

A CONCEPTUAL SCHEMA BASED ON RDFS-OWL AND CONSISTENCY CONSTRAINT CHECKING WITH XQuery*

OVILIANI YENTY YULIANA

Department of Computer Science and Information Engineering,
National Taiwan University of Science and Technology, Taipei, Taiwan
E-MAIL: d9915801@mail.ntust.edu.tw

Abstract:

This project assignment proposes a novelty (1) how to define Object Role Modeling (ORM) constraints as a conceptual schema using Resource Description Framework Schema (RDFS) and Web Ontology Language (OWL) in RDF/XML syntax as a semantic web; (2) how to define ORM constraints using XQuery for checking consistency XML data with the constraints. All defined RDFS, OWL, and XQuery are well validated. In addition, the defined XQueries can be used for checking consistency XML data with the constraint. Several ORM constraints still are not supported by RDFS and OWL. Current XQuery syntax can not be used to define several ORM constraints.

Keywords:

ORM Constraints; RDFS; OWL; XQuery

1. Introduction

Nowadays XML is gradually accepted as a standard for representing, accessing, and exchanging data through internet applications. However, accessing and exchanging XML data typically are not completed with XML schema (XSD) and semantic information. In addition, data in XML databases as a text file is very easy to modify. Without XSD and semantic information, it is very difficult to keep consistency data in XML databases. Therefore, it is impossible to produce quality information.

There are several researches as a background of this project. Reengineering XML Databases using Object Role Modeling (ORM) [1] as conceptual data model into the fifth normal form is conducted by [2]. In addition, [3] extended the revise engineering method using concept single and multi association rules. Furthermore, [4] proposed how to cover ORM constraints on forward engineering that do not supported by XSD syntaxes. In 2001, Berners-Lee proposed a *semantic web* as an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation [1]. As far as I know, there is not a tool that has a feature to forward

engineering using ORM as a data model.

The project assignment aims are to (1) define the forward engineering ORM constraints that still are not covered by [4] using Resource Description Framework Schema (RDFS) and Web Ontology Language (OWL) [5] into RDF/XML format by example; (2) define XQuery [6,7] for checking consistency constraints XML data by example; (3) validate RDF/XML and XQuery using a validator; and (4) conduct experiment to evaluate XQuery.

2. Conceptual schema based on RDFS-OWL

ORM has rich constraints [1], i.e. uniqueness, mandatory or optional, value (data type, data pattern, enumeration, and range), set-comparison (subset, equality, and exclusion), subtype (partition, exclusive, and exhaustive), frequency, and ring (reflexive, symmetric, transitive, irreflexive, asymmetric, antisymmetric, and intransitive). XSD (physical level) syntaxes can only define uniqueness, mandatory or optional, and value constraints. However, XSD does not support set-comparison, subtype, frequency, and ring constraints [4].

RDFS and OWL are needed to define set-comparison, subtype, frequency, and ring constraints in a conceptual level in order to support semantic web application. In this project, RDF/XML syntax will be used to define the ORM constraints based on RDFS and OWL. RDFS only supports subtype constraint in context sub class and super class definition. To define partition, exclusive, exhaustive, and ring constraints, OWL is needed. A foundation layer for the semantic web is shown in Figure 1.

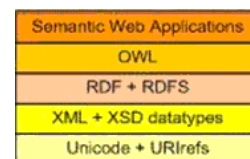


Figure 1. The foundation layer for the semantic web

*Extended Version of A Conceptual Schema Based XML Schema with Integrity Constraints Checking Paper, IEEE Computer Society Proceeding of ICHIT 2008, Daejeon-Korea

The project will use ORM meta data that produced by [2,3,4] as shown in Figure 2. In general, the meta data is reversed engineering into RDF/XML as shown in Figure 3. RDF/XML is divided into header, class elements, and property elements that is divided into object properties and data type properties. To define class elements, **Constraint**, **ObjectType**, **RoleType**, and **SubtypingObject** tables are needed. **Role** and **RoleCosntraint** tables are needed to define object properties. At last, **ConstraintValue** table is needed to define data type properties. The main concern project is how to define ORM constraints that were not covered by [4]. Therefore the detail syntax in RDF/XML will not show in detail.

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

Figure 2. ORM meta data

- Header
- Class Elements
 - Constraint
 - ObjectType
 - RoleType
 - SubtypingObject
- Property Elements
 - Object properties
 - Role
 - RoleConstraint
 - Data type properties
 - ConstraintValue

Figure 3. RDF/XML general format

There are three ORM subtypes, i.e. partition, exclusive, and exhaustive. Subtypes example can be shown in Figure 4, Figure 6, and Figure 8 respectively. Define every subtype and super type at a class element. In addition, define **subClassOf** super type (**Academic**) in every subtype class element (e.g. **Professor** and **SeniorLecturer**). To define exclusive subtype, add **disjointWith** tag at each exclusive subtype class (**SeniorLecturer**) as show in Figure 5. To define exhaustive subtype, add **unionOf** tag at super type class element (**Person**) as show in Figure 7. To combine exclusive and exhaustive constraints, i.e. partition, use **disjointWith** and **unionOf** tags just discussed. For instance see Figure 9.

ORM frequency constraint example is shown in Figure 10. It is defined using **cardinality** tag at **Restriction** in class elements definition, e.g. see Figure 11. There are three types ORM set-comparison constraint, i.e. subset, equality,

and exclusion. However, RDFS and OWL only support to define a subset constraint. It defines using **subPropertyOf** tag at **InverseFunctionProperty** on property element, i.e. object property. The ORM subset constraint dan RDF/XML subset constraint can be shown in Figure 12 and Figure 13 respectively. **ObjectProperty Car.Drives** is a subset of **ObjectProperty Person.Owns**.

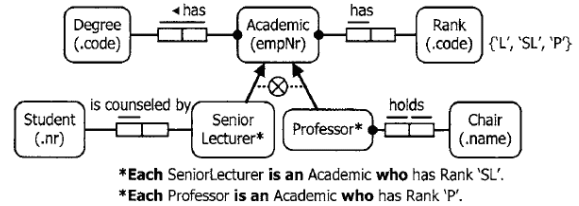


Figure 4. ORM diagram exclusive subtype constraint

```
...
<owl:Class rdf:ID="Academic">
  ...
</owl:Class>
<owl:Class rdf:ID="Professor">
  <rdfs:subClassOf rdf:resource="#Academic"/>
</owl:Class>
<owl:Class rdf:ID="SeniorLecturer">
  <rdfs:subClassOf rdf:resource="#Academic"/>
  <owl:disjointWith rdf:resource="#Professor"/>
</owl:Class>
...
```

Figure 5. RDF/XML exclusive subtype constraint

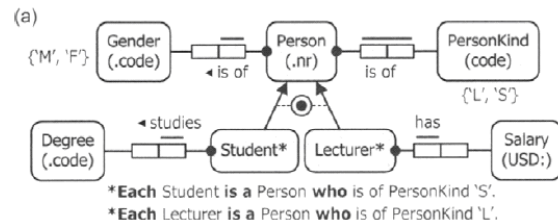


Figure 6. ORM diagram exhaustive subtype constraint

```
...
<owl:Class rdf:ID="Student">
  <rdfs:subClassOf rdf:resource="#Person"/>
</owl:Class>
<owl:Class rdf:ID="Lecturer">
  <rdfs:subClassOf rdf:resource="#Person"/>
</owl:Class>
<owl:Class rdf:ID="Person">
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Student"/>
    <owl:Class rdf:about="#Lecturer"/>
  </owl:unionOf>
</owl:Class>
...
```

Figure 7. RDF/XML exhaustive subtype constraint

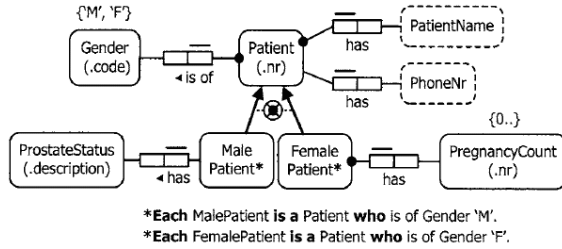


Figure 8. ORM diagram partition subtype constraint

```
...
<owl:Class rdf:ID="MalePatient">
  <rdfs:subClassOf rdf:resource="#Patient"/>
</owl:Class>
<owl:Class rdf:ID="FemalePatient">
  <rdfs:subClassOf rdf:resource="#Patient"/>
  <owl:disjointWith rdf:resource="#MalePatient"/>
...
</owl:Class>
<owl:Class rdf:ID="Patient">
...
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#MalePatient"/>
    <owl:Class rdf:about="#FemalePatient"/>
  </owl:unionOf>
</owl:Class>
...
```

Figure 9. RDF/XML partition subtype constraint

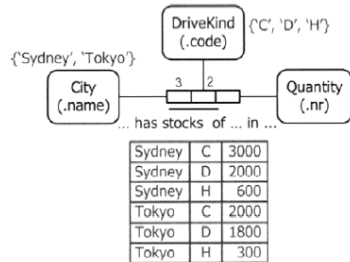


Figure 10. ORM diagram frequency constraint

There are seven ORM ring constraints, i.e. reflexive, symmetric, transitive, irreflexive, asymmetric, antisymmetric, and intransitive. However RDFS and OWL do not support for defining AntiSymmetry, InTransitive, and Acyclic constraints. In general, attribute domain and range tag base on the same class element. Symmetric constraint example is shown in Figure 14. Person object type is defined on class element **Person**. Ring constraint symmetric Friend is defined on ObjectProperty element **Friend** with **SymmetricProperty** attribute. The RDF/XML symmetric constraint Figure 14 can be shown in Figure 15. In addition, transitive constraint is shown in Figure 16 and RDF/XML is shown in Figure 17. To define RDF/XML

transitive similar with RDF/XML symmetric, only change symmetric with **TransitiveProperty** attribute.

```
...
<owl:Class rdf:ID="Stock">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#Stock.City"/>
      <owl:cardinality rdf:datatype="&xsd:nonNegativeInteger">3
    </owl:Restriction>
  </rdfs:subClassOf>
  ...
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#Stock.DriveKind"/>
      <owl:cardinality rdf:datatype="&xsd:nonNegativeInteger">2
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
...
```

Figure 11. RDF/XML frequency constraint

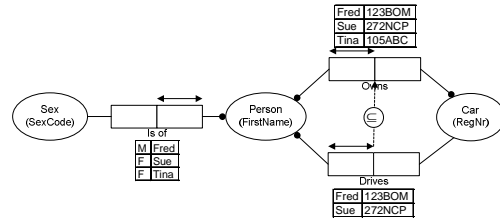


Figure 12. ORM diagram subset constraint

```
...
<owl:ObjectProperty rdf:ID="Person.Owns">
  <rdfs:domain rdf:resource="#Person"/>
  <rdfs:range rdf:resource="#Car"/>
  <owl:inverseOf rdf:resource="#Car.Owns"/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="Person.Drives">
  <rdfs:domain rdf:resource="#Person"/>
  <rdfs:range rdf:resource="#Car"/>
  <owl:inverseOf rdf:resource="#Car.Drives"/>
</owl:ObjectProperty>
<owl:InverseFunctionalProperty rdf:ID="Car.Owns">
  <rdfs:domain rdf:resource="#Car"/>
  <rdfs:range rdf:resource="#Person"/>
  <owl:inverseOf rdf:resource="#Person.Owns"/>
</owl:InverseFunctionalProperty>
<owl:InverseFunctionalProperty rdf:ID="Car.Drives">
  <rdfs:domain rdf:resource="#Car"/>
  <rdfs:range rdf:resource="#Person"/>
  <rdfs:subPropertyOf rdf:resource="#Person.Owns"/>
  <owl:inverseOf rdf:resource="#Person.Drives"/>
</owl:InverseFunctionalProperty>
...
```

Figure 13. RDF/XML subset constraint

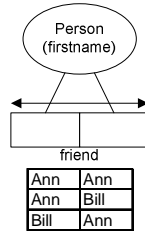


Figure 14. ORM diagram symmetric constraint

```

...
<owl:Class rdf:ID="Person">
...
</owl:Class>
...
<owl:ObjectProperty rdf:ID="Friend">
  <rdf:type rdf:resource="&owl:SymmetricProperty"/>
  <rdfs:domain rdf:resource="#Person"/>
  <rdfs:range rdf:resource="#Person"/>
</owl:ObjectProperty>
...

```

Figure 15. RDF/XML symmetric constraint

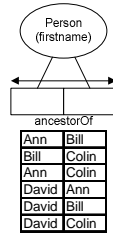


Figure 16. ORM diagram transitive constraint

```

...
<owl:Class rdf:ID="Person">
...
</owl:Class>
...
<owl:ObjectProperty rdf:ID="AncestorOf">
  <rdf:type rdf:resource="&owl:TransitiveProperty"/>
  <rdfs:domain rdf:resource="#Person"/>
  <rdfs:range rdf:resource="#Person"/>
</owl:ObjectProperty>
...

```

Figure 17. RDF/XML transitive constraint

OWL does not support defining reflexive, irreflexive, and asymmetric constraints. Therefore, OWL 2 must be used. All defined RFD/XML must valid. However, ALTOVA Semanticworks does not support to validate OWL 2. Manchester validator will be used, therefore reflexive, irreflexive, and asymmetric constraints will be defined using Manchester syntax. Reflexive constraint example can be shown in Figure 18 and Manchester reflexive constraint

can be shown in Figure 19. Property characteristic **ReflexiveObjectProperty** is used to define reflexive constraint, for example see line 14 in Figure 19. Irreflexive and asymmetric constraints can be defined using the same way, just change the property characteristic with **IrreflexiveObjectProperty** and **AsymmetricObjectProperty** respectively.

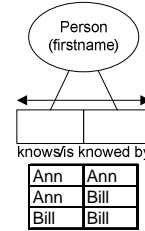


Figure 18. ORM diagram reflexive constraint

1. Prefix(:=<http://example.com/owl/families/>)
2. Prefix(otherOnt:=<http://example.org/otherOntologies/families/>)
3. Prefix(xsd:=<http://www.w3.org/2001/XMLSchema#>)
4. Prefix(owl:=<http://www.w3.org/2002/07/owl#>)
5. Ontology(<http://example.com/owl/families>
6. Declaration(NamedIndividual(:Ann))
7. Declaration(NamedIndividual(:Bill))
8. Declaration(Class(:Person))
9. Declaration(ObjectProperty(:knows))
10. Declaration(DataProperty(:FirstName))
11. ObjectPropertyDomain(:knows :Person)
12. DataPropertyDomain(:FirstName :Person)
13. DataPropertyRange(:FirstName xsd:string)
14. **ReflexiveObjectProperty**(:knows)
15. ClassAssertion(:Person :Ann)
16. ObjectPropertyAssertion(:knows :Ann :Ann)

Figure 19. Manchester reflexive constraint

3 Consistency constraint checking with XQuery

The previous section discussed how to define semantic of ORM constraints as a documentation system. The defined semantic of ORM constraints are used to people or machine which cooperative work on web application can better understanding each other. Another aim purpose in this project is how to checking XML data consistency with the defined constraint. Therefore XQuery will be used. Generally XQuery syntax can be shown in Figure 20 [6,7]. There is a limitation in XQuery syntax so in this project only define XQuery for several constraints.

XQuery for checking consistency example ORM subset constraint in Figure 21 is shown in Figure 22. XQuery for checking consistency example ORM subtype constraint in Figure 8 is shown in Figure 23. In addition, XQuery for checking consistency example ORM frequency constraint in Figure 24 is shown in Figure 25. Moreover,

XQuery for checking consistency example ORM irreflexive constraint in Figure 26 is shown in Figure 27. At last, XQuery for checking consistency example ORM asymmetric constraint in Figure 28 is shown in Figure 29.

```

FLWORExpr ::= (ForClause | LetClause)+ WhereClause?
              OrderByClause? "return" ExprSingle
ForClause ::= "for" "$" VarName TypeDeclaration? PositionalVar? "in"
              ExprSingle ("," "$" VarName TypeDeclaration? PositionalVar?
              "in" ExprSingle)*
LetClause ::= "let" "$" VarName TypeDeclaration? ":" ExprSingle
              ("," "$" VarName TypeDeclaration? ":" ExprSingle)*
WhereClause ::= "where" ExprSingle

```

Figure 20. XQuery syntax in general

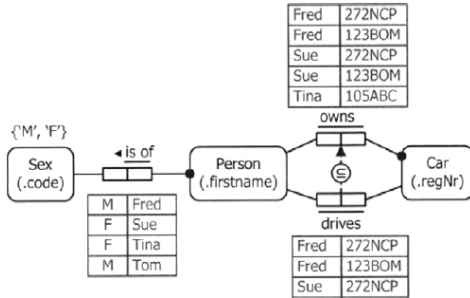


Figure 21. Another ORM diagram subset constraint

```

for $j in fn:doc("SubSetData.xml")//Drives
let $p := fn:doc("SubSetData.xml")//Owns
  [FirstName=$j/FirstName and RegNr=$j/RegNr]
return
if (fn:exists($p)) then()
else (
  <InvalidSubSet>
    { $j/FirstName }
    { $j/RegNr }
    <Note>Invalid Subset</Note>
  </InvalidSubSet>
)

```

Figure 22. XQuery for checking subset constraint

```

for $j in fn:doc("SubTypeData.xml")//MalePatient
let $p := fn:doc("SubTypeData.xml")//Patient
  [Nr= $j/Nr and Code="M"]
return
if (fn:exists($p)) then()
else (
  <InvalidSubTypeMale>
    { $j/Nr }
    { $p/Code }
    <Note>Invalid Sub Type</Note>
  </InvalidSubTypeMale>
)

```

Figure 23. XQuery for checking subtype constraint

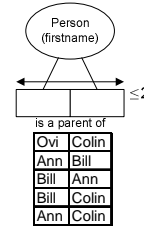


Figure 24. Another ORM diagram frequency constraint

```

for $j in fn:distinct-values(
  fn:doc("OccurFreqData.xml")//Person/ParentOf)
let $p := fn:doc("OccurFreqData.xml")//Person
  [ParentOf = $j]
return
if (fn:exists($p) and count($p/ParentOf)>2) then
  <InvalidMaxOccurece>
    <ParentOf> { $j } </ParentOf>
    <CoutParentOf> { fn:count($p/ParentOf) } </CoutParentOf>
    <Note>The occurrent frequency must be less than or equal to
      two</Note>
  </InvalidMaxOccurece>
else ()

```

Figure 25. XQuery for checking frequency constraint

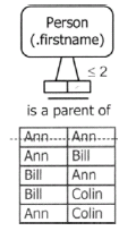


Figure 26. Another ORM diagram irreflexive constraint

```

for $e in doc("IrreflexiveData.xml")//IrreflexiveData/Person
where $e/FirstName=$e/ParentOf
return
  <InvalidIrreflexive>
    { $e/FirstName }
    { $e/ParentOf }
    <Note>Invalid irrefelxive constraint</Note>
  </InvalidIrreflexive>

```

Figure 27. XQuery for checking irreflexive constraint

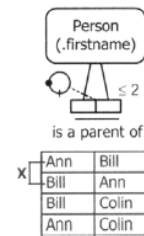


Figure 28. Another ORM diagram asymmetric constraint

```

for $r in fn:doc("ASymmetricData.xml")/ASymmetricData/Person
let $a := fn:doc("ASymmetricData.xml")/ASymmetricData/Person
[FirstName = $r/ParentOf and ParentOf = $r/FirstName]
return
if (fn:exists($a)) then
  <InvalidASymmetric>
    {$r/FirstName}
    {$r/ParentOf}
  <Note>Invalid Asymmetric</Note>
</InvalidASymmetric>
else ()

```

Figure 29. XQuery for checking asymmetric constraint

4 Experiment and result

ALTOVA Semanticworks 2011 [8] was used to validate RDF/XML syntaxes for the defined semantic ORM conceptual level using RDFS and OWL. The experiments show that all defined semantic ORM constraints are well validated as shown in Figure 30. The defined XQueries will be validated and executed by ALTOVA XMLSpy 2011 [9]. All of the defined XQueries can give information about inconsistency XML data with the constraint. For example, see the table data Figure 26. The XQuery in Figure 27 is run and give the information as shown in Figure 31. After deletion the first row data, the XQuery will show the information in Figure 32.


OWL Full →  This ontology is well-formed.

Figure 30. Validated RDF/XML syntaxes

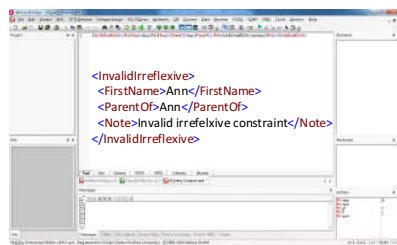


Figure 31. Information inconsistency irreflexive XML data

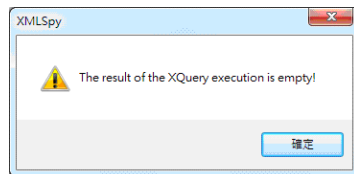


Figure 32. XML data consistent with irreflexive constraint

5. Conclusions

RDFS and OWL can be used to define ORM subtype,

frequency, subset, symmetric, and transitive constraints. OWL 2 can be used to define ORM reflexive, irreflexive, and asymmetric constraints. However, OWL 2 still can not define ORM equality, exclusion, antisymmetry, intransitive, and acyclic constraints. The defined subset, equality exclusion, subtype, occurrence, irreflexive, and asymmetric XQueries can be used for checking consistency XML data with the constraints.

Acknowledgements

I wish to thank to Prof. Yi-Leh WU for his continuous guidance, encouragement, patient, and support.

References

- [1] Halpin T. A. Antony J. M. and Tony M., Information Modeling and Relational Databases the Second Ed., Morgan Kaufmann, USA, 2008.
- [2] Yuliana O. Y. and Suphamit C., "XML Schema Re-Engineering Using a Conceptual Schema Approach", IEEE Computer Society Proceeding of ITCC 2005 Conference, Las Vegas, pp. 255-260, April 2005.
- [3] Yuliana O. Y. and Suphamit C., "Deriving Conceptual Schema from XML Databases", IEEE Computer Society Proceeding of ACIIDS 2009 Conference, Dong Hoi, pp. 40-45, April 2009.
- [4] Yuliana O. Y. and Suphamit C., "A Conceptual Schema Based XML Schema with Integrity Constraints Checking", IEEE Computer Society Proceeding of ICHIT 2008 Conference, Daejeon, pp. 19-24, August 2008.
- [5] W3C Recommendation (27 October 200p), OWL 2 Web Ontology Language Primer, retrieved at 17 October 2010, web site: <http://www.w3.org/TR/owl2-primer/>.
- [6] W3C Recommendation (14 December 2010), XQuery 1.0: An XML Query Language Second Edition, retrieved at 18 December 2010, web site: <http://www.w3.org/TR/2010/REC-xquery-20101214/#id-sequencetype-syntax>.
- [7] W3C Recommendation (14 December 2010), XML Syntax for XQuery 1.0 (XQueryX) Second Edition, retrieved at 18 December 2010, web site: <http://www.w3.org/TR/xqueryx/>.
- [8] ALTOVA SemanticWorks 2011, retrieved at 10 December 2010, web site: <http://www.altova.com/semanticworks.html>.
- [9] ALTOVA XMLSpy 2011, retrieved at 10 December 2010, web site: <http://www.altova.com/xmlspy.html>.