

# Generating the Good XML Schema from Relational Database by using String Matching Algorithms

Myint Myint Lwin, Thi Thi Soe Nyunt, Yuzana

University of Computer Studies, Yangon

*lwin.myintmyint@gmail.com, thithisn@gmail.com, yuzana@gmail.com*

## Abstract

*Data exchanging is involving as an important role in the Web application and Extensible Markup Language (XML) is also accepted as the standard for exchanging information between the heterogeneous systems. The XML schema acts as the standard format between the sender and receiver for the XML documents. As a result, the design of the schema document is very important because most of the XML documents are validated with their schema. In this paper, the two string matching algorithms (Maximum Consecutive Substring at right and any) are proposed for detecting the common attributes in the relations. Then, these string matching algorithms are applied for detecting the common attributes in the relations. The similar attributes are grouped and converted as element group in XML schema document for converting the relational database to XML schema document. The resulted XML schema is more modular; more understand for the human readers and reduce the maintainability effort.*

## 1. Introduction

Nowadays, Web applications are very popular and XML is also tightly couple with the Web. XML is the standard language and useful for the exchanging information over the Internet. The main advantages of XML are text-based and structured format of data. In real world, most of data are stored in relational database because of many users are familiar with the relational database. They cannot dispensable the relational database because it is the mature technology and

has many advantages such as concurrency and data consistency etc. On the other hand, the Web technology is currently popular and useful in business applications that are based on the Web technology. Some business's information and data are required to convert into the XML format for exchanging data between the business organizations. However, converting to XML format from RDB is non-trivial task.

Many researchers have been proposed and presented the various converting methods between XML format and relational database. They considered the different point of views such as structural or semantic. But the earlier converting methods did not consider the good XML schema design (maintainability effort or understanding of human reader). This paper presents the efficient converting method and generates the good XML schema by considering the important design factors.

This paper presents about introduction in section 1. The section 2 describes the related works. The background theory of XML and motivation are presented in section 3 and 4. In section 5, the architecture of the proposed system is described and concludes in Section 6.

## 2. Related Works

Many translation methods have been proposed taking into account structural and/or semantic aspects [8]. Flat Translation (FT) is the simplest translation method which converts relations to XML. In this method, each relations is converted into an element (E) and each relation attributes is either converted to a subelement (element approach) or attribute

(attribute approach) of E. It is not efficient because it does not apply nesting idea. Nesting-based Translation (NeT) [5] was proposed to solve the problems found in FT. It utilized the nested structure from the flat relational model by using the nest operator such as “\*” and “+”. As a result, the resulting DTD is more efficient and useful for decreasing data redundancy. However, NeT considers tables one at a time and cannot provide a whole relational schema where many relations are interconnected with each other through various other dependencies. FT and NeT are structured method and they did not consider the semantic aspects. Constraints-based Translation (CoT) algorithm [6] was developed to solve the problem occurred in NeT. It uses Inclusion Dependency (INDs) of relational schema which based on the foreign key constraints. It is mostly associated with the usage of sub-elements and IDREF attributes for translation purpose. Moreover, it considers not only the structural part such as tables and columns but also the semantic part such as constraints and referential integrity (RI). But it can only provide the explicit RI. If the implicit RI exists, it cannot produce an exact XML document. The ConvRel algorithm [3] detects the relation between tables and extracts the referential integrity by applying the idea of parent-child relationship. It can provide the N:M relationships modeled as a combination between a nested structure and keyref. All of the above algorithms did not evaluate complexity of their resulted XML schema documents and did not consider the reusability and maintainability effort. The proposed system considers the all aspects of structural, semantic and maintainability efforts. The quality of XML schema documents includes size of the document, line of code, number of simple type or complex type etc. Finally, Dilek Basci and Sanjay Misra [4] proposed also a metric called Schema Entropy (SE) metric based on entropy concept. Although many measuring metrics are developed to measure the quality of XML schema documents, all of above papers did not measure their schema complexity. The SE metrics is developed to measure the complexity of XML schema document and for the good

XML schema design that reduces the maintainability efforts.

This paper proposed the good design for XML schema documents to reduce the large amount of maintainability effort by using the three string matching methods for grouping the same attribute. Moreover, it can provide more structure and reduce the maintainability efforts.

### 3. Theory Background

XML is the most suitable language for Web-based data exchange and XML schema also acts as the major role in the World Wide Web application. The XML instance documents can be validated against the associated schema definition. An XML Schema is a document which describes the rules for XML document. A structure of an XML document can be defined as Document Type Definition (DTDs), XML Schema Definition (XSD), XML Data Reduced (XDR). The most popular XML schemas are DTD and XSD.

#### 3.1. The role of the XML schema on the Web

XML schema is essential that both parts have the same expectations about the content when sending data from a sender to a receiver. With the XML schema, the sender can describe the data in a way that the receiver will understand.

#### 3.2 Important design factors of good XML schema

As the consequence of converting the data from relational database into XML format, the efficient converting methods are required. In paper [10] described the important design factors as following.

(i) **Information preservation:** it is fundamental for converted XML Schema that should be retained structural and semantic information of the relational database entirely.

(ii) **Highly nested structure:** nesting is important in XML documents because it allows navigation of the paths in the document tree structures to be processed efficiently.

(iii) **No redundancy:** there is no data redundancy in the XML documents that conform to the target XML schema, thus no inconsistency will be introduced while updating the XML documents.

(vi) **Consideration of dominant applications:** the structure of XML document should be enough or compatible with the dominant applications can be guaranteed to be processed efficiently.

(v) **Reversibility of design:** the original design can be obtained from the target XML schema, which is fundamentally important to data integration.

#### 4. Motivation

Traditionally, a lot of data are stored in relational database and their technology is strong and settled. However, sometime they are not suitable for the Web application and need to convert the XML format that is compatible with the Web technology. But the relational database and XML technology are challenge with each other with their advantages and disadvantages. For this reasons, most of the users cannot dispensable the usage of relational database and still use it today. But sometime, the relational data are needed to exchange between different business and conversion to XML format is required.

The researchers have been focused on the converting between relational and XML format. They proposed many converting method from relational database to XML schema. The earlier methods have been applied the structural or semantic point. But they did not measure their converted XML schema and did not consider the schema design factors such as modular, maintainability effort, easier for human reader. Based on our investigation, some of the attributes in tables are generally same. But their attribute names are different according to the desire of database designer. For example, empname,

stdname, staffname are variation of name but the database designer set the attribute name as their desired. This case prevents the converting from relational database to XML schema and introduces the redundancy tags and increases maintainability effort.

Therefore, the proposed system converts the relational database into the XML schema by focusing on the important design factors. To support the essential design factors, the proposed system detect the general same attributes in the relation using the string similarity measuring methods. Then the same attributes are grouped and created as element group in the schema document to reduce code and more structure.

#### 5. Architecture of the Proposed System

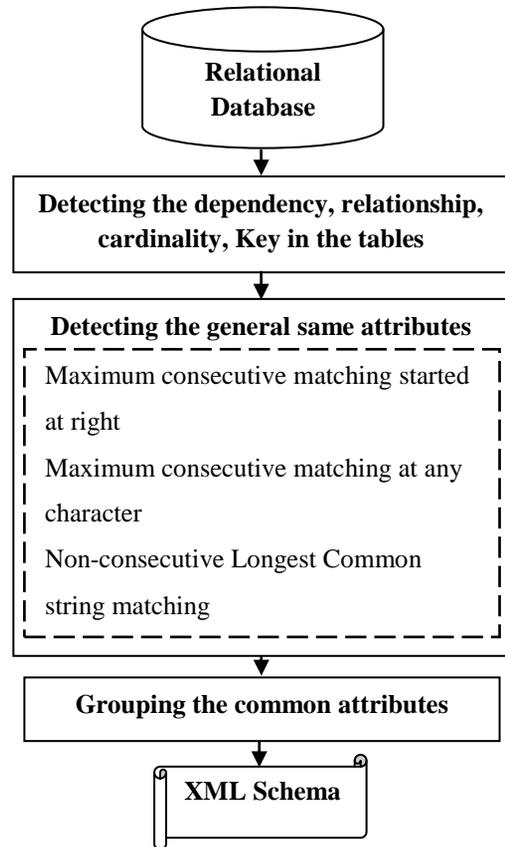


Figure 1: Architecture of the proposed system

The architecture of the proposed system is described in Figure 1. The relational database is given as input and the final output is good XML schema document. The first step of the proposed system is detecting the dependencies between the attributes of the relation by analyzing the keys of the relations to get the highly nested structure. Then the relationships and cardinality of the relations are detected to create the relationship between the elements in the schema. And then the same attributes in the relations are grouped by using similarity measuring methods (non-consecutive string matching, maximum consecutive matching started at left, maximum consecutive matching started at right and maximum consecutive matching at any character and). They are described as follow.

```

Algorithm 1: non-consecutive Longest Common Substring
Input   :  $S_1, S_2$  // two strings to compare
Output  :  $S_c$  //common any character but not consecutive

Begin
   $i \leftarrow 0$ 
  While ( $i < S1.length$  and  $S2.length > 0$ )
    If  $S1_i \subseteq S2$  Then
       $S_c \leftarrow S_c.concat(S1_i)$ 
       $S2 \leftarrow S2 \setminus S1_i$ 
    End
     $i \leftarrow i+1$ 
  End
  Return  $S_c$ 
End

```

**Figure 2: Non-consecutive longest common substring algorithm**

The algorithm 1 accepts two strings as input and finds the common characters from the input strings but these characters are not consecutive. And then it produces the longest common substring.

**Algorithm 2:** Maximum consecutive Substring matching started at left

Input :  $S_1, S_2$  // two strings to compare  
Output :  $S_{MCSI}$  // maximum consecutive substring

```

Begin
   $d \leftarrow |S_1|, t \leftarrow |S_2|$ 
  If  $d > t$ 
     $r_i \leftarrow S_2, s_j \leftarrow S_1$ 
  Else
     $r_i \leftarrow S_1, s_j \leftarrow S_2$ 
  End if
  While  $|r_i| \leq 1$ 
    If  $r_i \in s_j$ ; that is  $s_j \cap r_i = r_i$ 
      return  $r_i$ 
    Else
       $r_i \leftarrow r_i \setminus c_k$  that is, remove the right-most character from  $r_i$ 
    End if
  End while
End

```

**Figure 3: Maximum consecutive substring at left algorithm**

The algorithm 2 gets the two input strings and produces the maximum consecutive string. It produces the maximum left consecutive string.

**Algorithm 3:** Maximum consecutive Substring matching started at right

Input :  $S_1, S_2$  // two strings to compare  
Output :  $S_{MCSn}$  // maximum consecutive substring

```

Begin
   $d \leftarrow |S_1|, t \leftarrow |S_2|$ 
  If  $d > t$ 
     $r_i \leftarrow S_2, s_j \leftarrow S_1$ 
  Else
     $r_i \leftarrow S_1, s_j \leftarrow S_2$ 
  End if
  While  $|r_i| \geq 1$ 
    If  $r_i \in s_j$ ; that is  $s_j \cap r_i = r_i$ 
      return  $r_i$ 
    End if
  End while
End

```

```

Else
    ri ← ri \ ck that is, remove the
    left-most character from ri
End if
End while
End

```

**Figure 4: Maximum consecutive substring at right algorithm**

The algorithm 3 has the similar manner of algorithm 2 because it also gets the two input strings and produce the maximum consecutive string. It produces the maximum right consecutive string.

Based on the discovery, the prefixes of the some attributes are influenced by the table's name. For example Sname, Saddress, Stdphone are the attributes of student table. The prefixes S and Std are obstacles to calculate the similarity of attributes. They are required to remove from the attribute to improve the accuracy of the similarity values. Therefore, the algorithm 1 is applied to remove the prefix of the some attributes in the relations.

The algorithm 4 is also similar the previous algorithms. But it accepts the two strings and produces the maximum consecutive string that is started at any character and consecutive.

**Algorithm 4:** MaxConConsecutiveAny  
Input : S1, S2 // two strings to compare  
Output : S<sub>MCSany</sub>  
Begin  
If S1.length < S2.length then  
    pattern ← S1  
    target ← S2  
Else  
    pattern ← S2  
    target ← S1  
End If  
While ( i < target.length)  
    C ← pattern.Char(i)  
    tempMax ← tempMax.concat (C)  
    If tempMax ⊆ target then  
        If tempMax.length > Max.length then  
            Max ← tempMax  
        End If  
    End While

```

Else
    tempMax ← tempMax \ leftmostchar
    i ← i+1
End If
End While
Return Max
End

```

**Figure 5: Maximum consecutive substring at any algorithm**

Then the similar attributes are normalized using the following normalized equations to get the accurate similarity values [1].

The equation (1) is used for normalizing the similar attributes that are produced by algorithm 1.

$$v_1 = NS_c(r_i, s_j) = \frac{\{length(S_c(r_i, s_j))\}^2}{length(r_i) \times length(s_j)} \quad \text{---(1)}$$

The equation(2) is applied for normalizing the similar attributes that are produced by algorithm 2.

$$v_2 = NS_{MCSn}(r_i, s_j) = \frac{\{length(S_{MCSn}(r_i, s_j))\}^2}{length(r_i) \times length(s_j)} \quad \text{---(2)}$$

The equation(3) is used for normalizing the similar attributes which are produced by algorithm 3.

$$v_3 = NS_{MCSany}(r_i, s_j) = \frac{\{length(S_{MCSany}(r_i, s_j))\}^2}{length(r_i) \times length(s_j)} \quad \text{----(3)}$$

After normalization the strings, the value of each normalize are evaluated the similarity use the below equation. The similarity of the two strings is:

$$\alpha = w_1 v_1 + w_2 v_2 + w_3 v_3 \quad \text{.....(4)}$$

where  $\alpha$  is the similarity value of two strings. Then,  $w_1, w_2, w_3$  are weights of each normalized value and  $w_1 + w_2 + w_3 = 1$ . The similar attributes are grouped which are satisfied the threshold value 0.5 and create them as the element group in

schema documents. Finally, the good XML schema is generated to reduce the complexion of code.

## 5.1 Illustration of XML schema document

The following tables are in the relational database.

SUPPLIER (S#, SNAME, STATUS, CITY)  
 SPJ (S#, P#, J#, QTY)  
 PROJECT (J#, JNAME, CITY)  
 PART (P#, PNAME, COLOR, WEIGHT, CITY)

In the above tables, SNAME, JNAME, PNAME are generally same because they are represented for name. Therefore, NAME and CITY are common in some tables and group them as group element in XML schema document. The final result of XML schema is described in Figure 6.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:group name="GROUP">
    <xsd:sequence>
      <xsd:element name="NAME"
        type="xsd:string"/>
      <xsd:element name="CITY"
        type="xsd:string"/>
    </xsd:sequence>
  </xsd:group>

  <xsd:element name="database">
    <xsd:element name="SPJ">
      <xsd:ComplexType>
        <xsd:sequence >
          <xsd:element name="S#"
            type="xsd:Integer">
            <xsd:sequence maxOccur="Unbound">
              <xsd:element name="SUPPLIER">
                <xsd:ComplexType>
                  <xsd:element name="STATUS"
                    type="xsd:Integer"/>
                  <xsd:group ref="GROUP"/>
                </xsd:ComplexType>
              </xsd:element>
            </xsd:sequence>
          </xsd:ComplexType>
        </xsd:element>
      </xsd:ComplexType>
    </xsd:element>
  </xsd:element>
</xsd:schema>
```

```
</xsd:sequence>
</xsd:element>

<xsd:element name="J#"
  type="xsd:Integer">
  <xsd:sequence maxOccur="Unbound">
    <xsd:element name="PROJECT">
      <xsd:ComplexType>
        <xsd:group ref="GROUP"/>
      </xsd:ComplexType>
    </xsd:element>
  </xsd:sequence>
</xsd:element>

<xsd:element name="P#"
  type="xsd:Integer">
  <xsd:sequence maxOccur="Unbound">
    <xsd:element name="PART">
      <xsd:ComplexType>
        <xsd:element name="COLOR"
          type="xsd:string"/>
        <xsd:element name="WEIGHT"
          type="xsd:Integer"/>
        <xsd:group ref="GROUP"/>
      </xsd:ComplexType>
    </xsd:element>
  </xsd:sequence>
  <xsd:element name="QTY"
    type="xsd:Integer"/>
  </xsd:sequence>
</xsd:ComplexType>
</xsd:element>

</xsd:element>
</xsd:schema>
```

**Figure 6: The generated xml schema document using element group**

The output of the XML schema is more structure and easier for human reader by grouping the generally same attributes in the relations. In real world, many relations are included in the database. When more same attributes are involved in the database, the number of tags can be reduced in the schema document and provide more modular the code.

## 6. Conclusion and Future work

The proposed system is presented with two new string matching algorithms to obtain the generally common attributes in the database. The resulted XML schema is presented with example. The proposed system will reduce the tags of the common attributes in each element such as SNAME, PNAME and JNAME in XML schema document. Therefore, the generated XML schema is more modular and provides the more understandability of the human reader. Measuring the complexity and quality (maintainability effort, reusability etc) of the resulted XML schema design and measuring the accuracy of the proposed algorithms will be described in our ongoing tasks.

## References

- [1] Aminul Islam, Diana Inkpen, Iluju Kiringa, "Applications of corpus-based semantic similarity and word segmentation to database schema matching", Volume 17 Issue 5, Springer-Verlag New York, Inc. Secaucus, NJ, USA, August 2008.
- [2] Andrew McDowell, Chris Schmidt, Kwok-Bun Yue, Analysis and Metrics for XML Schema, CSREA press 2004, ISBN: 1932415277, USA, 2004, pp. 538-544.
- [3] Angela Cristina Duta, Ken barker and Reda Alhaji, "ConvRel: relationship conversion to XML nested structure", Proceedings of the 2004 ACM symposium on Applied computing, ACM New York, NY, USA, 2004.
- [4] Dilek Basci and Sanjay Misra, "Entropy as a Measure of Quality of XML Schema Document", The International Arab Journal of Information Technology, Vol. 8, No1, January 2011.
- [5] Dongwon Lee, Murali Mani and Wesley W. Chu, Nesting-based relational-to XML schema translation, In Proceedings of International Workshop on the Web and Databases, 2001, pp. 61-66.
- [6] Dongwon Lee, Murali Mani and Wesley W. Chu, NeT & CoT: Translating relational schemas to XML schemas using semantic constraints, In Proceedings of the 11<sup>th</sup> ACM International Conference on Information and Knowledge Management, 2002, pp. 282-291.
- [7] Dagwon Lee , Murali Mani and Wesley W. Chu, "Effective Schema Conversions between XML and Relational Models", In European Conf. on Artificial Intelligence (ECAI), Knowledge Transformation Workshop (ECAI-OT), 2002.
- [8] Jinhyung Kim, Dongwon Jeong and Doo-Kwon Baik, A Translation Algorithm for Effective RDB-to-XML Schema Conversion Considering Referential Integrity Information, Journal of Information Science and Engineering 25, 2009, pp 137-166.
- [9] Joseph Fong, Anthony Fong, HK Wong and Philip Yu, Translating relational schema with constraints into XML schema, International Journal of Software Engineering and Knowledge Engineering IJSEKE, Volume 16, Issue 2, 2006, pp 201-243.
- [10] Rui Zhou, Chengfei Liu and Jianxin Li , Holistic constraint-preserving transformation from relational schema into XML schema, Proceedings of the 13th international conference on Database systems for advanced applications, Springer-Verlag Berlin, 2008.