A Generation of XML Schema from Entity-Relationship Model

Chang Suk Kim
Kongju National University
Dept. of Computer Education
Chungnam, Kongju 314-702
Republic of Korea
e-mail: csk@kongju.ac.kr

Dae Su Kim
Hanshin University
Dept. of Computer Science
Kyunggi, Osan 447-791
Republic of Korea
e-mail: daekim@hanshin.ac.kr

Kwang-Baik Kim
Silla University
Dept. of Computer Science
Busan, 617-736
Republic of Korea
e-mail: kbkim@silla.ac.kr

***Abstract*—The XML is emerging as standard language for data exchange on the Web. Therefore a demand of XML Schema(W3C XML Schema Spec.) that verifies XML document becomes increasing. However, XML Schema has a weak point for design because of its complication despite of various data and abundant expressiveness. This paper shows a simple way of design for XML Schema using a fundamental means for database design, the Entity-Relationship model. The conversion from the Entity-Relationship model to XML Schema can not be directly on account of discordance between the two models. So we present some algorithms to generate XML Schema from the Entity-Relationship model. The algorithms produce XML Schema codes using a hierarchical view representation. An important objective of this automatic generation is to preserve XML Schema's characteristics such as reusability, global and local ability, ability of expansion and various type changes.**

I. INTRODUCTION

The XML(eXtensible Markup Language) is emerging as standard language for data exchange on the Web. It has great advantages such as extensibility, portability and the possibility to add semantics to data within the document itself. To have these characteristics, XML have to model data or documents using a DTD(Document Type Definitions) which describes document structural constraints. But DTD syntax is different from XML document syntax. So DTD is limited to model and describe a document structure since it does not support XML namespace and various data types. Recently a demand of XML Schema(W3C XML Schema Spec.) that verifies XML document becomes increasing. XML Schema supports various data types which can be represented a complex document. The syntax of XML Schema is the same as XML document against DTD has EBNF type syntax of its own.

However, XML Schema has a weak point for design because of its complication despite of various data and abundant expressiveness. Thus, it is difficult to design a complex document reflecting the usability, global and local facility and ability of expansion[1, 2].

This paper shows a simple way of design for XML Schema using a fundamental means for database design, the Entity-Relationship model. The conversion from the Entity-Relationship model to XML Schema can not be directly on account of discordance between the two models. So we present some algorithms to generate XML Schema from the Entity-Relationship model. The algorithms produce XML Schema codes using a hierarchical view representation. An important objective of this automatic generation is to preserve XML Schema's characteristics such as reusability, global and local ability, ability of expansion and various type changes.

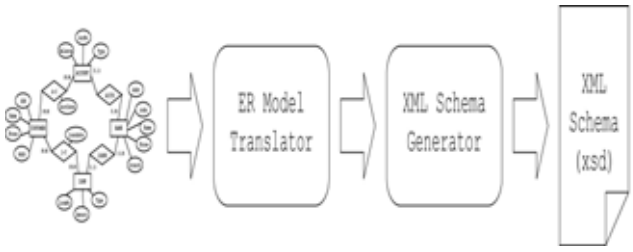

Figure 1. Generation of XML Schema from E-R model.

Entity-Relationship model is not accord with XML Schema structure, so the transformation XML Schema from Entity-Relationship model can not be processed directly.

Before to generate XML Schema Entity-Relationship model should be translated hierarchical view. The target XML Schema (xsd file) is generated from the hierarchical view using proposed transformation rule and constraints.

II. RELATED WORKS

The researches about generating DTD from Entity-Relationship model are accomplished at UCLA (EXPRESS project), University of Applied Sciences (DB2XML project) and Stanford University (Lore Project) [3]. But it is rare to study generating XML Schema from Entity-Relationship model[4].

Elmasri[5] introduced a design methodology for XML Schema that based on well-understood conceptual model. Elmasri proposed entity migration to transform Entity-Relationship model. So some migrated original entities are disappeared. This cause some problem to preserve XML Schema's characteristics such as reusability and ability of expansion.

III. TRANSFORM HIERARCHICAL STRUCTURE FROM E-R MODEL

A. Transformation Overview

Transformation algorithms about hierarchical view from Entity-Relationship model are described. This algorithm has a characteristic to use reusability of duplicated entities.

Transformation Overview is as follows:

- BFS(Breadth First Search) algorithm searches a root entity
- Transform graph-style E-R representation to hierarchical structure
- Arrange E-R representation

B. BFS Scan

It needs to transform graph-style E-R representation to hierarchical structure. To do this BFS(Breadth First Search) algorithm searches a root entity which is designated by operator(Figure 2 and Figure 3).

C. Hierarchical structure model

After to processing BFS scan, we can get a hierarchical structure model as shown Figure 4. When we scan from ACCOUNT entity to BANK entity, BANK entity is known as duplicated entity. Then the BANK entity generated BANKGroup entity. All the attributes of BANKsuch as branch, phone, name, code, addr are moved into BANKGroup entity(Figure 5). BANK entity refers BANKGroup entity as shown Figure 4.

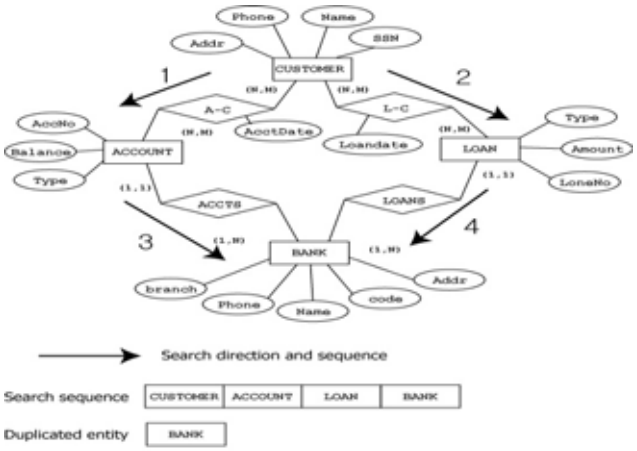

Figure 2. BFS scan.

Algorithm BFSScan
Begin search designated root entity;
do
{
Search designated root entity;
if (duplicated entity){
Save duplicated entity into duplicated queue (duplicateQue);
}else{
Save normal entity into queue (bfsQue);
}
}while(in searching process);

Figure 3. BFS scan algorithm.

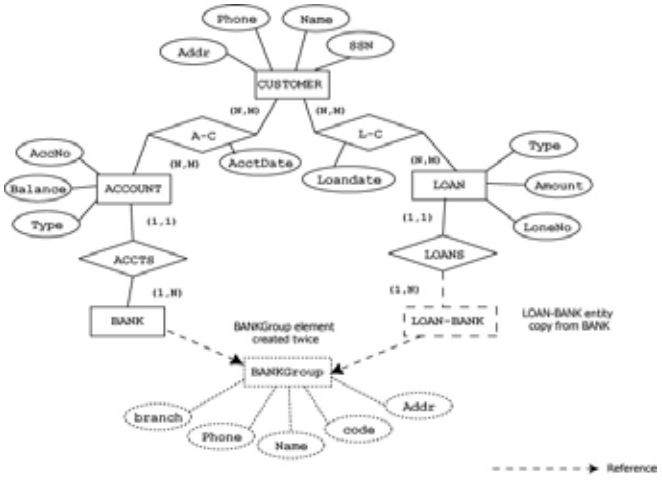

Figure 4. Hierarchical structure model.

```
algorithm GenerateHierView
for(number of entities){
  if(child entity of current entity is duplicated){
    if(first duplication){
      generate new group entity; // named as "node-nameGroup"
      move to the new generated entity of current child entity;
      current child entity refer new generated entity;
    }else{
      copy duplicated entity; // named as "parent entity-child entity"
      duplicated entity refer generated group entity already;
    }
  }
}
```

Figure 5. Algorithm GenerateHierView.

IV. GENERATION XML SCHEMA FROM HIERARCHICAL STRUCTURE

A. Assumptions

This chapter describes XML Schema generation from hierarchical structure model. In general, it is possible to generate several XML Schemas from the same hierarchical structure model. So we assume some restrictions.

- Entity is generated as designated complex type and global style and it is named as 'entity-nameType'.
- Attributes are generated as simple type. If current entity has a child entity, it includes the child entity.
- If mapping constraints is one-to-one, then the occurrence generated as minOccurs = "1" maxOccurs = "1". Else if mapping constraints is one-to-N, then the occurrence generated as minOccurs = "1" maxOccurs = "unbounded".
- If group entity are existed, the group entity should be declared global.

B. XML Schema file and Schema declaration

Following algorithm creates XML Schema file and Schema declaration. All created XML Schema documents have a prefix 'xs:' which differentiate same named entities.

Algorithm createXmlSchemaDoc

XML Schema file creation;

write("<?xml version="1.0" encording="UTF-8"?>");

write("<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">");

Following code is generated by above algorithm createXmlSchemaDoc.

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

C. Root entity creation

Root element is defined as designated complex type and element type is named 'rootType'. The compositor is defined <sequence> and root entity is named 'root-entity-nameType'. Occurrence generated as minOccurs = "1" maxOccurs = "unbounded". In this example, root type's name is CUSTOMER.

D. Group element creation

If hierarchical structure model have group elements, 'group-elementGroup' are generated. The compositor is defined <sequence>, names are defined the same names of attribute and type is generated as 'xs:string' (Figure 6).

Root element is defined as designated complex type and element type

```
algorithm createGroup
for(number of group){
  write(creation group element 'group-elementGroup');
}
```

Figure 6. Algorithm createGroup.

```
<xs:group name="BANKGroup">
  <xs:sequence>
    <xs:element name="Addr" type="xs:string"/>
    <xs:element name="code" type="xs:string"/>
    <xs:element name="Name" type="xs:string"/>
    <xs:element name="Phone" type="xs:string"/>
    <xs:element name="Branch" type="xs:string"/>
  </xs:sequence>
 </xs:group>
```

The resulting codes are generated from Entity-Relationship model in Figure 2 using proposed algorithms.

```
<?xml version="1.0" encording="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="CUSTOMERDoc" type="rootType"/>
 <xs:complexType name="rootType">
   <xs:sequence>
     <xs:element name="CUSTOMER" type="CUSTOMERType" minOccurs="0" maxOccurs="unbounded">
   </xs:sequence>
 </xs:complexType>

 <xs:complexType name="CUSTOMERType">
   <xs:sequence>
```

```
    <xs:element name="Ssn" type="xs:string"/>
    <xs:element name="Name" type="xs:string"/>
    <xs:element name="Phone" type="xs:string"/>
    <xs:element name="Addr" type="xs:string"/>
    <xs:element name="ACCOUNT" type="ACCOUNTType" minOccurs="1" maxOccurs="unbounded"/>
    <xs:element name="LOAN" type="LOANType" minOccurs="1" maxOccurs="unbounded"/>
   </xs:sequence>
  </xs:complexType>
  <xs:complexType name="ACCOUNTType">
   <xs:sequence>
    <xs:element name="Balance" type="xs:string"/>
    <xs:element name="AccNo" type="xs:string"/>
    <xs:element name="Type" type="xs:string"/>
    <xs:element name="AcctDate" type="xs:string"/>
    <xs:element name="BANK" type="BANKType" minOccurs="1" maxOccurs="1"/>
   </xs:sequence>
  </xs:complexType>

  <xs:complexType name="LOANType">
   <xs:sequence>
    <xs:element name="LoanNo" type="xs:string"/>
    <xs:element name="Amount" type="xs:string"/>
    <xs:element name="Type" type="xs:string"/>
    <xs:element name="LoanDate" type="xs:string"/>
    <xs:group ref="BANKGroup"/>
   </xs:sequence>
  </xs:complexType>

  <xs:complexType name="BANKType">
   <xs:sequence>
    <xs:group ref="BANKGroup"/>
   </xs:sequence>
  </xs:complexType>

  <xs:group name="BANKGroup">
   <xs:sequence>
    <xs:element name="Addr" type="xs:string"/>
    <xs:element name="code" type="xs:string"/>
    <xs:element name="Name" type="xs:string"/>
    <xs:element name="Phone" type="xs:string"/>
    <xs:element name="Branch" type="xs:string"/>
   </xs:sequence>
  </xs:group>

</xs:schema>
    <xs:element name="Name" type="xs:string"/>
    <xs:element name="Phone" type="xs:string"/>
    <xs:element name="Branch" type="xs:string"/>
   </xs:sequence>
  </xs:group>

</xs:schema>
```

V. EXPERIMENTATION AND DISCUSSION

The proposed algorithms for transforming hierarchical structure and generating XML Schema are implemented by Java programming language (JDK 1.4.2.). We built some Java classes such as GenenrateXmlSchema, MakeE, MakeR etc. Our approach comparing to previous study has advantages such as reusability and expansion. Figure 7 show redundant element creation of the existing XML Schema generation method.

```
<xs:complexType name="ACCOUNTType">
  <xs:sequence>
    ……
    <xs:element name="BankAddr" type="xs:string"/> <!—elements are duplicated -->
    <xs:element name="Bankcode" type="xs:string"/>
    <xs:element name="BankName" type="xs:string"/>
    <xs:element name="BankPhone" type="xs:string"/>
    <xs:element name="Bankbranch" type="xs:string"/>
  </xs:sequence>
 </xs:complexType>
 <xs:complexType name="LOANType">
  <xs:sequence>
    <xs:element name="Loandate" type="xs:string"/>
    <xs:element name="Amount" type="xs:string"/>
    <xs:element name="LoneNo" type="xs:string"/>
    <xs:element name="Type" type="xs:string"/>
    <xs:element name="BankAddr" type="xs:string"/> <!—elements are duplicated -->
    <xs:element name="Bankcode" type="xs:string"/>
    <xs:element name="BankName" type="xs:string"/>
    <xs:element name="BankPhone" type="xs:string"/>
    <xs:element name="Bankbranch" type="xs:string"/>
  </xs:sequence>
 </xs:complexType>
</xs:schema>
```

Figure 7. XML Schema code with duplicated elements.

VI. CONCLUDING REMARKS

This paper showed a simple way of design for XML Schema using the Entity-Relationship model. The conversion from the Entity-Relationship model to XML Schema can not be directly on account of discordance between the two models. So we presented some algorithms to generate XML Schema from the Entity-Relationship model. The algorithms produce XML Schema codes using a hierarchical view representation. An important objective of this automatic generation is to preserve XML Schema's characteristics such as reusability, global and local ability, ability of expansion and various type changes. Finally we showed the reusability compare to existing approach.

ACKNOWLEDGMENTS

This work was supported by grant R01-2002-000-00068-0 from the Basic Research Program of the Korea Science and Engineering Foundation in Republic of Korea.

REFERENCES

[1] Jon Duckett, etc., *Professional XML Schemas*, Wrox, 2001.

[2] Kevin Williams, etc., *Professional XML Databases*, Wrox, 2001.

[3] Dongwon Lee, "Schema Conversion Methods between XML and Relational Models", *Knowledge Transformation for the semantic Web*, 2003.

[4] Garsten Kleiner and Udo Lipeck, "Automatic Generation of XML DTDs from Conceptual Database Schemas", *Informatik 2001 - Wirtschaft und Wissenschaft in der Network Economy - Visionen und Wirklichkeit*, 2001.

[5] Ramez Elmasri, "Conceptual Modeling for Customized XML Schema", *Proceedings of the 21st International Conference on Conceptual Modeling 2002*, page 429-443.