IE Interfaces
Vol. 17, No. 3, pp. 330-337, September 2004.

A Meta-Model for the Storage of XML Schema using Model-Mapping Approach

Hoontae Kim[1†] · Taesoo Lim[2] · Keunhee Hong[3] · Suk-Ho Kang[4]

[1]Dept. of Industrial and Systems Engineering, Daejin University, Pocheon, 487-711
[2]LG Electronics PRC, Pyongtaek, 451-713
[3]OpenTide Korea, Seoul, 135-845
[4]Dept. of Industrial Engineering, Seoul National University, Seoul, 151-742

모델 매핑 접근법을 이용한 XML 스키마 저장 메타모델에 대한 연구

김훈태[1] · 임태수[2] · 홍근희[3] · 강석호[4]

[1]대진대학교 산업시스템공학과 / [2]LG 생산기술원 / [3]오픈타이드 코리아 / [4]서울대학교 산업공학과

Since XML (eXtensible Markup Language) was highlighted as an information interchange format, there is an increasing demand for incorporating XML with databases. Most of the approaches are focused on RDB (Relational Databases) because of legacy systems. But these approaches depend on the database system. Countless researches are being focused on DTD (Document Type Definition). However XML Schema is more comprehensive and efficient in many perspectives.

We propose a meta-model for XML Schema that is independent of the database. There are three processes to build our meta-model: DOM (Document Object Model) tree analysis, object modeling and storing object into a fixed DB schema using model mapping approach. We propose four mapping rules for object modeling, which conform to the ODMG (Object Data Management Group) 3.0 standard. We expect that the model will be especially useful in building XML-based e-business applications.

Keyword: XML Schema, meta-model, model mapping approach, DOM tree, object modeling

1. Introduction

Much business information has been generated on the web and companies have recognized a necessity for storing useful information created from business transactions. Since XML(eXtensible Markup Language) was highlighted as an information interchange format(IIF), much information has been transformed into XML documents. Then it was transferred between businesses and stored into a database. As a result, companies have been requesting the incorporation of XML into their databases(Bourret 2003). Those works are classified into a Relational mapping and an Object-Oriented mapping according to the type of database. RDB is preferred because it is

This work was supported by grant No.R01-2002-000-00155-0 from the Basic Research Program of the Korea Science and Engineering Foundation.
† Corresponding author : Hoontae Kim, Dept of Industrial and Systems Engineering, Daejin University, San 11-1, Sundandong, Pocheon, Gyeonggi, Korea, 487-711, Fax : 82-31-539-2000, E-mail: hoontae@daejin.ac.kr
Received August 2003, accepted July 2004 after 1 revision.

commonly used because a lot of advanced RDB techniques can be applied(Bertino and Catania, 2001; Florescu and Kossmann, 1999; Shanmugasundaram *et al.,* 1999; Yoshikawa *et al.,* 2001), whereas OODB is more proper for processing the complex data structure of XML(Christophides *et al.,* 1994; Chung *et al.,* 2001; Goldman *et al.,* 1999; Lin *et al.,* 2000). However, these previous approaches are limited to a specific database type. It is also the fact that most of those works are focused on XML instance and just some for DTD (Document Type Definition). However it is expected that XML Schema will substitute for DTD because of its richer resource sets for powerful information interchange as a document meta-data(Roy and Ramanujan, 2001).

This paper suggests a database-independent meta-model to store XML Schema document using DOM (Document Object Model) tree analysis, object modeling and model-mapping approach. Model-mapping approach uses a fixed DB schema so that it stores and easily updates meta-data of dynamic and structurally variant documents.

This paper explains related works and terminologies in chapter 2, suggests modeling procedures in chapter 3, explains each step of the process in chapter 4, 5, 6, and specific characteristics and future works are presented in chapter 7.

2. Related Works

2.1 XML Schema

XML meta-data describes the logical structure, contents and constraints of XML document. DTD derived from SGML(Standard Generalized Markup Language) is one of the XML meta-data description languages and has been widespreadly used so far. However, it has a lot of problems such as insufficient data types, different syntax from XML, lack of namespace, etc. These problems have shown much limitation on the usage of DTD and led the birth of a new XML meta-data description language, the XML Schema.

XML Schema was approved as a W3C(World Wide Web Consortium) Recommendation in May 2001. It extends the document model of DTD and strengthens the functionalities as an IIF. It supports 19 built-in types including string, integer, float, double, Boolean, etc. and 25 derived built-in types. In addition, it effectively provides the flexibility of data type definition with user-defined type.

XML Schema is more comprehensive and efficient in several perspectives. As Roy and Ramanujan (2001) points out, its characteristics can be explained with the following 4 perspectives.

Data Type. XML Schema provides not only rich data types but also user-defined types. Therefore, it is easy to generate XML document having appropriate syntax and semantics.

Syntax. XML Schema follows XML syntax so that just one parser can process both XML meta-data and instances.

Reusability. XML Schema supports inheritance, that is, the partial or overall of existing XML Schema can be reused by inheritance.

Namespace. XML Schema prevents conflicts problems caused by naming duplication using namespace. In addition, it is easy to generate valid documents using various namespaces defined by several XML Schema.

2.2 Previous Researches

We classify previous researches into Object-Oriented models and Relational models according to the modeling method. In addition, there are structure-mapping and model-mapping approaches for database schema design of XML document. In the structure-mapping approach, a database schema(in this paper, meta-model means database schema) is defined for each XML document schema. It is suitable when we store a large number of XML documents that conform to a limited number of document structures and when the document structure is static. Contrastingly, in the model-mapping approach, database schemas represent constructs of the XML document model. That is, a fixed database schema is used to store the structure of all XML documents. It is appropriate for storing a large number of dynamic and structurally-variant XML documents like numerous sophisticated Web applications (Yoshikawa *et al.,* 2001). <Table 1> and <Table 2> summarize some representative features and limitations of the model suggested by previous works.

This paper suggests an integrated model that is independent of the modeling method. As a solution, it applies the DOM tree analysis to extract structural characteristics from XML Schema and

performs object modeling to build database-independent data model, which is compliant with ODMG 3.0 Standard(Cattel and Barry, 2000). Furthermore, it designs a fixed database schema according to the model-mapping approach. Yhoshkawa *et. al.*(2001) suggested modeling and querying methods using the model-mapping approach. They focused on DTD and tried to make a RDB model. However, this paper suggests a database-independent meta-model to store XML Schema document. In addition, while they enumerated all the paths and stored the informations in a table, we decomposed the link informations to store them more efficiently.

3. Modeling Method

3.1 Building Process

Our procedure to build XML Schema meta-model is divided into three parts-first, DOM tree analysis of XML Schema and then, clustering of each node, second, four meta modeling rules applying to clustered nodes and the last, relational and object-oriented mapping for storing the generated ODMG 3.0 compliant schema into both databases according to the fixed database schema. <Figure 1> illustrates the overall procedure. In reality, meta modeling for primitive data type and facet is previously performed before the entire process, for it does not require a specific XML Schema document.

3.2 Fixed Database Schema

In model-mapping approach, a fixed schema is used to store the structure of all XML documents. Each XML document structure is stored as the data in the database according to the fixed database schema. We have designed two database schemas for both relational and object-oriented

Table 1. Object-Oriented models for XML database

Model	Mapping methodology	Target document	Features	Limitations
Christophides *et al.*, 1994	Structure mapping	DTD	- Create classes for all elements declared in DTD	- Abuse classes - Exclude the storage of additional information.(comment, tag order, etc.)
Goldman *et al.*, 1999	Structure mapping	XML instance	- Apply OEM(Object Exchange Model) data model - Introduce Dataguide about path information	- Lack of document update - Need more space to store Dataguides than actual XML data
Chung *et al.*, 2001	Structure mapping	DTD	- Apply inheritance: solve null value problem	- Exclude the storage of element order information

Table 2. Relational models for XML database

Model	Mapping methodology	Target document	Features	Limitations
Florescu *et al.*, 1999	Structure mapping	XML instance	- Use directed labeled graph. - Store node position and path information	- Expensive to re-compose data - Expensive and complex to update
Shanmugasundaram *et al.*, 1999	Structure mapping	XML instance	- Classify XML document - Data-centric - Document-centric - Hybrid representation	- Exclude the storage of additional information. (comment, tag order, etc.)
Bertino *et al.*, 2001	Structure mapping	XML instance	- Store according to the structural characteristics of DTD	- Exclude the storage of element order information
Yoshikawa *et al.*, 2001	Model mapping	XML instance	- Use directed labeled graph - Store path information	- Lack of full-text search - Lack of document update

databases, which are similar to each other. <Table 3> shows specific tables and classes consisting of both fixed schemas.

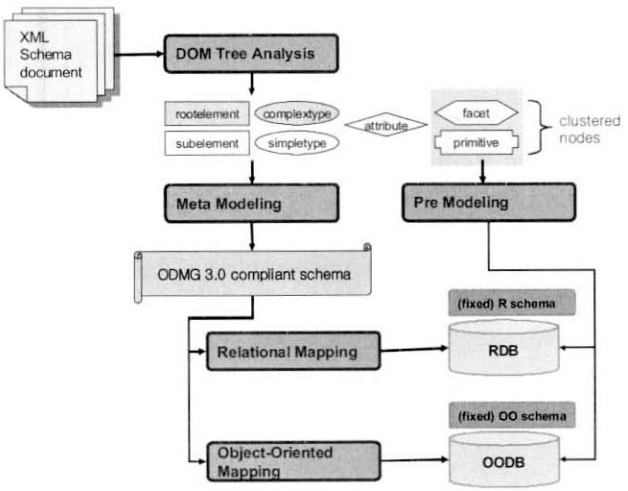


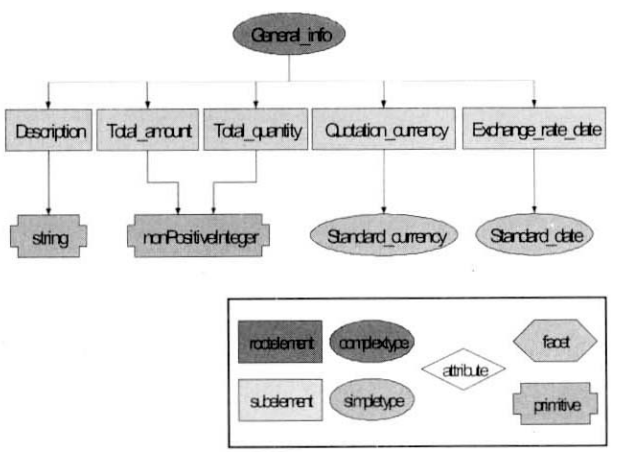
Figure 1. Building procedure of XML Schema meta-model.

4. DOM Tree Analyses

When XML Schema is loaded, DOM tree analysis is activated to grasp the document structure. Originally, DOM is the method to represent elements of XML hierarchically as a tree using several types of node(http://www.w3.org/TR/2003/WD-DOM-Level-3-Core-20030226/). In this paper, we classify these nodes into 7 types including rootelement, subelement, complextype, simpletype, attribute, primitive, and facet to analyze the structural characteristics concretely and precisely. <Figure 2> illustrates an example of DOM tree analysis. General_info of <Figure 2(a)> is one of the complex types in RFQ.xsd We composed a RFQ(Request For Quotation) document, which was introduced at the eCatalog technical committee of Korea Integrated Forum on Electronic Commerce on February 22, 2002. and <Figure 2(b)> is the result of DOM tree analysis for it.

```
<complexType name=General_info>
    <element name=Description   type=string/>
    <element name=Total_amount
        type=nonPositiveInteger/>
    <element name=Total_quantity
        type=nonPositiveInteger/>
    <element name=Qoutation_currency
        type=Standard_currency/>
    <element name=Exchange_rate_date
        type=Standard_date/>
</complexType>
```

(a) General_info type



(b) DOM tree analysis of General_info type

Figure 2. Example of DOM tree analysis.

Table 3. Fixed database schemas for XML Schema

Tables in RDB	Classes in OO DB	Role
RootElementRepository	RootElement	Extent of root elements
ComplexTypeRepository	ComplexType	Extent of complex types
SubElementRepository	subEelement	Extent of elements except a root element
AttributeRepository	Attribute	Extent of attributes
SimpleTypeRepository	SimpleType	Extent of simple types
PrimitiveTypeRepository	PrimitiveType	Extent of primitive types

This step analyzes the XML tree structure after loading the whole XML document. Even if the step requires large initial loading time, it is more efficient than other XML parsing approaches such as SAX(Simple API for XML) in the case of the high frequent access to XML document.

5. Meta Modeling

Meta modeling is the stage to generate data model independent of database. This paper suggests 4 mapping rules and these rules are compliant with ODMG 3.0 standard. First rule is regarding to class creation. The other rules specify type, element, built-in type, and attribute transformations.

Class Creation Rule. "*Three classes are always generated. Those are RootElementRepository, Simple-Type-Repository and ComplexTypeRepository classes.*"

RootElementRepository, SimpleTypeRepository and ComplexTypeRepository classes store the information about root element, simple type and complex type, respectively.

Type and Element Transformation Rule. "*ComplexType and SimpleType are defined as the attribute of ComplexTypeRepository and SimpleType-Repository class, respectively. Elements contained in the ComplexType are defined with struct declarer.*"

A ComplexType is defined as an attribute of ComplexTypeRepository class after being re-defined with struct type declarer and the name and type name of the generated attribute have to be identical. <Figure 3> illustrates the transformation of type and element.

<Figure 3(a)> is some part of RFQ.xsd to show how element, complex type and simple type are transformed. As you see <Figure 3(b)>, root element RFQ, complex type and simple type are stored as an attribute of RootElementRepository, ComplexTypeRepository, SimpleTypeRepository, respectively. In addition, the type of attribute in the ComplexTypeRepository class is declared with struct type declarer.

Primitive Data Type Mapping rule. "*A built-in primitive data type is transformed into an ODMG 3.0 literal. However, it is applied separately for common data type, constrained data type, decomposed data type and combined data type.*"

Built-in data types are derived from anySimpleType that restricts anyType, the super type of XML Schema built-in data type hierarchy. It is classified into primitive data types derived directly from anySimpleType and data types extended from those primitive types. Refer to (http://www.w3.org/XML/Schema#dev) for the details.

As the XML Schema is not developed just for ODMG 3.0 standard, all data types of XML Schema don't correspond to those of ODMG standard. The XML Schema supplies plentiful data types to express both document and data

```
<element name=RFQ    type=RFQType/>
<complexType name=RFQType>
... </complexType>
<complexType name=Delivery_info>
    <element name=Delivery_terms_code
     type=Standard_terms_code/>
    <element name=Delivery_location
     type=string/>
</complexType>
<simpleType name=Doc_num base=string>
</simpleType>
<simpleType name=Standard_date base=date
></simpleType>
```

(a) RFQ.xsd

```
class RootElementRepository {
    attribute RFQType RFQ;
};
class ComplexTypeRepository {
    attribute RFQType RFQType;
    attribute Delivery_info Delivery_info;
};
struct Delivery_info {
    Standard_terms_code Delivery_terms_code;
    string Delivery_location:
};
class SimpleTypeRepository {
    attribute string Doc_num;
    attribute date Standard_date;
};
```

(b) Meta modeling

Figure 3. Type and element transformation.

centric characteristics. On the other hand, the ODMG standard provides more generic types than the XML Schema. We divide the built-in data types into four categories as follows:

- *Common data type.* It is the data type that is directly mapped into the ODMG literal such as float, boolean, double, string, etc. It is possible to directly map a float type into a float literal. The typedef declarer specifies it as follows:

```
typedef float float;
```

- *Constrained data type.* It is the data type that can be mapped by adding some restrictions to the ODMG literals such as nonPositiveInteger, positiveInteger, negativeInteger, nonNegative-Integer, etc. For instance, nonPositiveInteger is specified by long long type and restricted by the constraint that its value should be equal or less than zero. We design meta-model describing the type definition for the constrained data type and its indication for the stored procedure as follows:

```
typedef long long nonPositiveInteger;
class constraint_register{
  attribute list<constraint> constraints;
  void register();
};
struct constraint{
  string type;
  string check_method;
};
```

- *Decomposed data type.* This type is mapped onto an ODMG 3.0 literal by combining several built-in data types of XML Schema. The XML Schema decomposes a date type into several types, such as gYearMonth, gYear, gMonthDay, gDay, and gMonth. On the contrary, ODMG 3.0 only provides a date literal. Therefore, the functions to combine those types need to be provided as follows:

```
typedef date gYearMonth, gYear, gMonth,
gMonthDay, gDay;
class constraint_register {
  attribute list<constraint> constraints;
  void register();
  date date_combine(); //a method for
combining several
        separated types.
};
```

- *Combined data type.* It is the data type that is defined by re-combining other types such as NOTATION, NMTOKENS, IDREFS, ENTITIES and so on. It is defined by using struct, set and list of ODMG literal according to the combination type as follows:

```
typedef set<QName> NOTATION;
typedef list<NMTOKEN> NMTOKENS;
typedef list<IDREF> IDREFS;
typedef list<ENTITY> ENTITIES;
```

Attribute Transformation Rule. "*Attribute is defined using struct type declarer.*"

In XML Schema, ComplexType can have attributes and subelements. Those attributes need to be stored in the struct for the ComplexType because it has its own name and type. However, it is difficult to distinguish it from the elements in the complex type (refer to Type and element transformation rule). Therefore, it is defined with additional struct type declarer. <Figure 4>

```
<element name=RFQ
  type=RFQType/> </element>
<complexType name=RFQType>
...
  <attribute name=Quotation_req_num
        type=Doc_num/>
  <attribute name=Quotation_req_date
   type=Standard_date/>
</complexType>
```

(a) RFQ.xsd

```
class ComplexTypeRepository {
    ...
    attribute RFQTypeAttribute RFQTypeAttribute;
};
struct RFQTypeAttribute {
    Doc_num Quotation_req_num;
    Standard_date Quotation_req_date;
};
```

(b) Meta modeling

Figure 4. Attribute transformation.

illustrates the example of attribute transformation. <Figure 4(a)> shows two attributes of RFQType complex type. They are declared with struct type declarer and then, defined as an attribute of Complex-TypeRepository like <Figure 4(b)>.

6. Database Mapping

After object modeling, the information extracted from XML Schema is stored according to the fixed database schema. Both databases store the information according to the following rules.

- RootElementRepository class is mapped into RootElementRepository table in RDB and Root-Element class in OODB.
- ComplexTypeRepository class is mapped into ComplexTypeRepository table and ComplexType class.
- Complex type declared by struct is mapped into subElementRepository table and subElement class.
- Attribute type declared by struct is mapped into AttributeRepository table and Attribute class.
- SimpleTypeRepository class is mapped into Sim-pleTypeRepository table and SimpleType class.
- Primitive data type declared by typedef is mapped into PrimitiveTypeRepository table and Primitive-Type class.

<Figure 5> illustrates the tables, fields, primary keys, and foreign keys in the relational database schema as stated in 3.2, and the mapping example for <Figure 3(b)> meta model is shown in <Figure 6>. An OODB mapping is similar to RDB mapping, therefore we omit it.

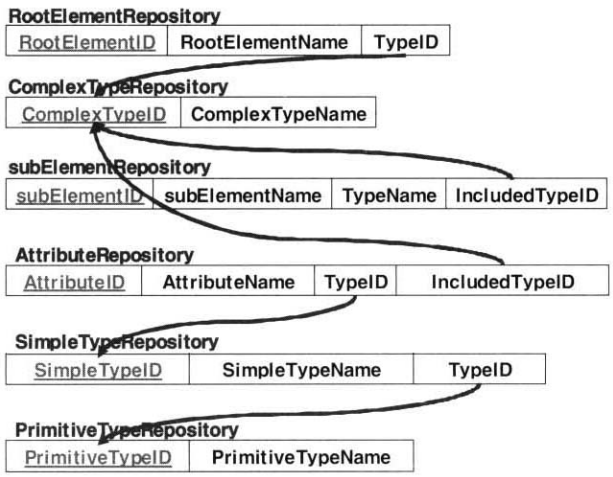


Figure 5. Relational database schema.

7. Conclusions

This paper proposes a meta-model for the storing of XML Schema, which is independent of a specific database. To build our model, we used the model-mapping approach, DOM tree analysis and object-oriented modeling.

First, model-mapping approach fixes the meta-model so that it prevents from abusing classes and tables. It makes it easy to update dynamic XML document and as a result increases the efficiency of information management and retrieval.

RootElementRepository Table

RootElementID	RootElementName	TypeID
RE_1	RFQ	CT_1

ComplexTypeRepository Table

ComplexTypeID	ComplexTypeName
CT_1	RFQType
CT_2	Delivery_info

PrimitiveTypeRepository Table

PrimitiveTypeID	PrimitiveTypeName
PT_1	string
PT_2	date
PT_3	nonPositiveInteger
PT_4	decimal

SimpleTypeRepository Table

SimpleTypeID	SimpleTypeName	TypeID
ST_1	Doc_num	PT_1
ST_2	Standard_date	PT_2
ST_3	Standard_terms_code	PT_4

subElementRepository Table

subElementID	subElementName	TypeName	IncludedTypeID
SE_1	Delivery_terms_code	Standard_terms_code	CT_2
SE_2	Delivery_location	string	CT_2

Figure 6. Storage of relational database.

Secondly, a lot of time was spent on the DOM tree analysis. Although a lot of time was spent, we can precisely and concretely grasp the structural characteristics of the XML document And last, object-oriented modeling easily catches the tree structure of XML document. We designed an effective and neutral meta- model following object-oriented ODMG 3.0 standard.

References

Bertino, E. and Catania, B.(2001), Integrating XML and databases, *IEEE Internet Computing*, 5, 84-88.

Bourret, R.(2003), XML and Databases, Tech. report, Technical Univ. Darmstadt, http://www.rpbourret.com/xml/

Cattell, R.G.G. and Barry, K. D.(2000), The object data standard: ODMG 3.0., *Morgan Kaufmann Publishers*

Christophides, V., Abiteboul, S., Cluet, S. and Scholl, M. (1994), From structured documents to novel query facilities, *SIGMOD Rec*, 23, 313-324.

Chung, T., Park, S., Han, S. and Kim, H.(2001), Extracting Object-Oriented Schemas from XML DTDs Using Inheritance, *The 2nd International Conference on Electronic Commerce and Web Technologies(EC-Web) with LNCS*

Document Object Model(DOM) Level 3 Core Specification (2003), http://www.w3.org/TR/2003/WD-DOM-Level-3-Core-20030226/

Extensible Markup Language(XML) Schema.(2001) http://www.w3.org/XML/Schema#dev

Florescu, D. and Kossmann, D.(1999), A performance evaluation of alternative mapping schemes for storing XML data in a relational database, Tech. Rep. 3680, INRIA

Florescu, D. and Kossmann, D.(1999), Storing and querying XML data using an RDBMS, *IEEE Data Engineering Bulletin*, 22, 27-34.

Goldman, R., Mchugh, J. and Widom, J.(1999), From semistructured data to XML: migrating the Lore data model and query language, *The 2nd International Workshop on the Web and Databases*, 25-30.

Lin, H., Risch, T. and Katchaounov, T.(2000), Object-Oriented mediator queries to XML data, *The 1st International Conference on Web Information Systems Engineering*, 38-45.

Roy, J. and Ramanujan A(2001)., XML Schema Language: Taking XML to the next level, *IEEE IT Pro*, 37-40.

Shanmugasundaram, J., Tufte, K., Gang, H., Zhang, C., Dewitt, D. and Naughton, J.(1999), Relational databases for querying XML documents: limitations and opportunities, *The 25th Conference on Very Large Data Bases*, 302-314.

Yoshikawa, M., Amagasa, T., Shimura, T. and Uemura, S. (2001), XRel: A path-based approach to storage and retrieval of XML documents using relational databases, *ACM Transactions on Internet Technology*, 1, 110-141

김 훈 태
서울대학교 산업공학과 학사
서울대학교 산업공학과 석사
서울대학교 산업공학과 박사
현재: 대진대학교 산업시스템공학과 부교수
관심분야: BPM, web services, 생산정보시스템

홍 근 희
한국외국어대학교 산업공학과 학사
서울대학교 산업공학과 석사
현재: 오픈타이드코리아 e-Business Consulting Group 컨설턴트
관심분야: Information Strategic Planning, e-Business Strategic Planning

임 태 수
서울대학교 산업공학과 학사
서울대학교 산업공학과 석사
서울대학교 산업공학과 박사
현재: LG 생산기술원 디자인엔지니어링그룹 선임연구원
관심분야: 생산정보시스템, 지식정보 시스템, 전자상거래표준화

강 석 호
서울대학교 물리학과 학사
미국 University of Washington 석사
미국 Texas A&M University 박사
현재: 서울대학교 산업공학과 교수
관심분야: Intelligent Manufacturing Systems, e-business