한국산학기술학회논문지
Vol. 12, No. 3 pp. 1327-1335, 2011

Semantics in XML Data Processing

Min Jin[1*]
[1]**Division of Computer Science and Engineering, Kyungnam University**

XML 데이터 처리에서 시맨틱

진 민[1*]
[1]경남대학교 컴퓨터공학부

Abstract XML is good at representing data with hierarchical and self-describing structure. There is no inherent semantics with XML. However, semantics of XML has come up with us as XML is used in wide and advanced applications. This paper surveyed semantics in XML data processing environment. XML semantics can be categorized into four groups according to its usage; structural semantics, relational semantics, extended semantics, and semantic web. Relational database is still a good alternative for storing and managing large volume of XML documents. We propose an extended relational semantics in order to exploit it in managing XML documents such as query processing.

요 약 XML은 계층적이고 스스로 기술하는 구조를 가지는 데이터를 표현하기에 좋지만 XML 자체는 시맨틱을 가지고 있지 않다. 하지만 XML이 널리 새로운 응용에서 사용됨에 따라 XML 시맨틱이 필요하게 되었다. 이 논문은 XML 데이터 처리 환경에서 시맨틱을 조사하였다. XML 시맨틱은 사용 용도에 따라 구조적 시맨틱, 관계 시맨틱, 확장 시맨틱과 시맨틱 웹으로 분류할 수 있다. 그리고 질의 처리와 같이 XML 문서 관리에서 활용하기 위해 확장된 관계 시맨틱을 제안한다.

Key Words : XML semantics, Relational semantics, Extended semantics, Extended relational semantics, Semantic web, Query processing

I. Introduction

XML which was originally developed as a simplified form of SGML has become a de facto standard for exchanging data and modeling semi-structured data in Internet data processing environments. It has being widely used in several areas. Due to the fact that XML has the hierarchical structure, it can be easily applied to modeling applications with hierarchical structures. However, as the application area of XML is extended, syntactic structure is not enough to represent all requirements of applications faithfully. Semantic issues have been raised in the XML applications recently.

Relational databases are still used for storing and managing voluminous XML documents[1-5]. XML is transformed into relational tables and relational tables are published in XML documents. In addition to the structural discrepancy between relational database and XML, there are semantic difference between them. Enhanced applications need to represent some semantic issues as well as syntactic hierarchical structure. For example, in documents analysis in digital library, the issue of thematic level and rhetoric level is required[6]. Web is extended to semantic web for the new era of web application[7]. The

This work was supported by Kyungnam University Foundation Grant, 2009
*Corresponding author: Min Jin(mjin@kyungnam.ac.kr)
Received February 01, 2011 Revised March 03, 2011 Accepted March 10, 2011

sematic web is also based on the XML.

The term *semantics* used in XML applications has different meanings. This paper surveys kinds of semantics occurred with XML documents and classifies them into a few categories. We also proposes a method for representing semantics in XML documents. There are several kinds of semantics; structural semantics, relational semantics, extended semantics and semantic web. They are to be discussed in the following section. There are two methods in representing XML semantics; explicit and implicit method, which will be discussed with different semantic meanings.

The paper is organized as follows. Section 2 surveys and describes possible kinds of semantics in XML documents. Section 3 discusses a method for representing XML semantics. The conclusion is offered in section 4.

2. Semantics in XML documents

There have been a lot of research on XML semantics in many areas. The meaning of semantics is different from each other depending on the perspective. The meaning of XML semantics can be classified into a few categories as follows: structural semantics, relational semantics, extended semantics, and semantic web.

2.1 Possible semantics in XML

1) Structural semantics

Structural semantics are inherent characteristics of XML. Data in relational databases are represented as flat tables, whereas hierarchical structure can be expressed in XML. Twigs and tag paths are used for describing structural entities. Repetitive and recursive notations also are provided for depicting repeating groups.

2) Relational semantics

The volume of XML documents is getting larger, it has given rise to the need of database technologies for storing and managing XML documents. Relational databases are widely used in storing XML documents[1-3]. Relational data stored in relational databases also need to be transformed into XML documents as XML is used as a standard for data exchange and representation in data processing environments. Inherent relational characteristics in relational data can be lost in processing of transformation into XML documents. Tok[8-10] used this kind of semantics in XML query processing. They proposed ORA-SS(Object-Relationship-Attribute Model for Semistructured Data) data model which includes relational data model concepts with constructs capturing the hierarchical structure of XML data. ORA-SS model includes the following concepts which can not be specified by conventional XML schema or DTD: Attribute vs object class, Multivalued attribute vs object class, Identifier, IDREF or foreign key, N-ary relationship, Attribute of object class vs attribute of relationship type, view of XML document.

Most of these concepts are originated from relational data model although they use different terms from ones used in relational databases. These concepts are exploited in XML query processing such as twig pattern queries[9].

3) Extended semantics

As XML is widely used in applications where semi-structured data are needed, semantics of XML has been enhanced in order to accommodate the requirements of such applications. Semantic issues have been studied for general applications as well as specific applications[6,11-14].

Yokota[11] extended XML semantically for many applications by introducing special attributes. They extended XML from the perspectives of data construction, logical units and elements, and dynamic aspects. Special attributes were introduced for representing set and tuple constructors, self-description, identities and constraints, logical unit, logical elements, links and navigation, conditional elements, application constraints. For the analysis of document markup in digital library, semantics of XML was studied in several works[6,15,16]. Renear et al.[6,15]characterized the specific problems for semantics of XML mark up language. They addressed the following issues; class relationship, propagation, context and reference, ontological variation in reference, full and partial synonymy. They also proposed a model for markup semantics based on Prolog. Bayerl et al.[6] presented a formal semantic model of XML markup for analysis of scientific articles. They raised the need of formal description of semantics of text-oriented XML

documents and provided two kinds of semantic levels: the thematic level and functional level.

Semantics of XML was investigated from the perspective of XML applications. Class concept was introduced by object-oriented programming environments and attribute classification was introduced by engineering applications[11].

4) Semantic web

Semantic web is an XML-based general purpose knowledge representation scheme which is machine accessible. It is a good candidate of representing machine-readable data for the future web applications. The standard form of the syntax of documents as well as the semantics of their contents must be defined so that machine can understand and process the documents. Although XML is a meta language for expressing the syntax of structured documents, it does not represent semantics of data. RDF, RDF Schema, and ontological language are integrated with XML in the semantic web in order to provide semantic interoperability for new web applications[17-19]. RDF is essentially a data model. Its basic building block is an object-attribute-value triple, which is called a statement. XML is just a possible serialization of the graph model of RDF. RDF is domain independent. RDF Schema provides the vocabulary to enable to model specific domains. The relation of XML Schema to XML is not the same as RDF Schema to RDF. XML Schema imposes the structure of XML, whereas RDF Schema defines vocabulary and semantics of specific domains.

2.2 Representation of Semantics

Structural semantics such as the hierarchical structure is represented explicitly in XML documents. However, relational semantics which is not inherent in XML can not be expressed explicitly in XML, it is implied in the representation of XML documents. It should be extracted by software developers in the applications. Fig. 1 shows relationship between kinds of semantics and representation scheme. The explicit method is similar to the declarative semantics and implicit method is similar to procedural semantics[7]. The difference between declarative and procedural loosely coincides with the difference between RDF and conventional XML[7].

	structural semantics	relational semantics	extended semantics	semantic web
explicit	o		o	o
implicit		o	o	

[Fig. 1] XML semantics and representation

In relational database, attributes are clearly defined for a relation and the relationship between entities are represented in the model. The cardinality of the relationship is also specified in relational data model. However, XML doesn't have the mechanism for relationship. Data constructor such as tuple constructor is also missing in the ordinary XML data representation. Hence, methods like shared shredding, inlined shredding, and hybrid shredding have been proposed for storing XML documents in relational database[2,20]. Extended semantics are mostly represented in terms of explicit annotations for advanced applications. Generic or special tags and attributes were introduced for extended semantics.

1) Explicit method

Semantics is represented by tags or attributes in most advanced XML applications. Three operations such as conceptualization, restriction, and relation are introduced for semantic search via XML fragments[13]. These operations are denoted as [<tag></tag>] to enable to specify semantics explicitly. For example, the conceptual query [[<Animal></Animal>]] retrieves documents containing the annotation Animal, which applies to all subtypes of the concept animal. The notation [[<Instrument>bass<Instrument>]] is differentiated from [[<Animal>bass<Animal>]]. Semantic information can be represented as attributes such as <book lang=English>. Semantics is fixed in each ingredient of RDF/RDFS whereas semantic information is extracted by software designers in XML. Modeling primitives are provided in RDF/RDFS for representing objects, properties, and relationships. Application programs which deal with documents represented in RDF/RDFS can understand the given meaning of semantics inherent in RDF without additional processing.

2) Implicit method

Originally XML doesn't have any fixed facility for semantics except the structure, the application programs have to find out the semantics implied in the contexts of XML documents if necessary. We need to analyze document markup semantics in some applications such as digital library, in which the semantics is not represented explicitly. Application software designers identify and process actual document markup semantics found in existing XML documents, in which the semantics was not planted explicitly. As mentioned in the previous section, the issues of class relationship, propagation, context and reference, ontological variation, and synonymy were investigated in the process of extracting semantics for the application in digital library[15].

Especially relational semantics is usually represented implicitly in XML documents rather than explicitly since XML does not have any facility of relational models. Application programs are required to extract the relational semantics such as tuple construction and relationship in the context of XML documents for the purpose of using relational databases.

3. Extended relational semantics

In this study, we propose a solution of XML semantics based on relational semantics taking into consideration of further enhancement for advanced applications. Semantic web seems to be the most advanced solution for representing semantics of web documents in the new era of web applications. Actually semantic web was introduced to provide machine-processible information. In order to keep up with the semantic operability of the semantic web, documents written in XML should be rewritten from the scratch with drastic change. There are huge amount of web or data documents written in XML and relational databases are still used for managing and storing XML documents. Several methods such as basic, shared, and hybrid inlining for storing XML documents in relational databases have been proposed[2,21]. However a few problems are raised in these methods. These methods tried to cope with the structural discrepancy between XML and relational databases, there were still left some issues unsolved. Moreover, the semantic issue was not taken into consideration in the process of translation between them. Hence, we propose a solution of XML semantics on the way to the new ear of web applications. We analyse the semantics of XML documents from the perspective of relational semantics and represent the semantics with minor change of existing XML documents. This kind of semantic information can be used in the process of strong XML documents in relational databases. It is also exploited in the applications where relational semantic is required. This can be the basis from which other semantics can be derived for specific applications.

1) Entity, Attribute, and Class

It is important to identify entities in designing of relational databases. An entity is delineated in terms of attributes. An element in XML documents is usually mapped to an entity in relational database. All of elements are not mapped into entities in shredding methods[22]. Some elements such as element-only elements are not mapped to tables. The structural information can be lost in the transformation process. There are three kinds of elements in the process of transformation into relational databases; one mapped to a table, one inlined into a table as an attribute, one eliminated. In the XML documents in Fig. 2, *paper*, *book*, *author* element are transformed into tables; *year*, *title*, *name* elements are inlined into tables. *Reference* and *publication* elements are element-only elements so that they are eliminated in the transformation process[22].

The information of the element including the structure can be lost during the process of transformation into relational databases. This is not what we want from the perspective of XML semantics. The notion of entity should be represented in order to observe the intension of the XML document designer when XML documents are stored in relational databases. This goes the same way when XML documents are used in object-oriented applications. Elements are mapped to classes instead of entities in these applications. The appearance of *ID* attribute has the element mapped into entity like *paper* element as shown in Fig. 2.

```
<publication>
<paper paperID="1">
    <year>2003</year>
    <title>Associaion Inling…</title>
    <authors>
        <author>
            <name>Byung-Joo Shin</name>
            <email>raniman@zeus.kyungnam.ac.kr</email>
        </author>
        <author>
            <name>Min Jin</name>
            <email>mjin@kyungnam.ac.kr</email>
        </author>
    </authors>
    <reference>
        <paper paperID="2">
            <year>1999</year>
            <title>Relational Databases…</title>
            <authors>
                <author>
                    <name>Shanmugasundaram.J</name>
                </author>
            </authors>
        </paper>
        <book>
            <year>2000</year>
            <title>Professional XML…</title>
            <authors>
                <author>
                    <name>Williams.K</name>
                </author>
            </authors>
        </book>
    </reference>
<book>
    <year>2002</year>
    <title>C++ XML</title>
    <authors>
        <author>
            <name>Arciniegas.F</name>
        </author>
    </authors>
</book>
.
:
</publication>
```

[Fig. 2] An XML document

The notion of entity should be represented in order to observe the intension of the XML document designer when XML documents are stored in relational databases. This goes the same way when XML documents are used in object-oriented applications. Elements are mapped to classes instead of entities in these applications. The appearance of *ID* attribute has the element mapped into entity like *paper* element as shown in Fig. 2.

2) Relationship

Two important components of relational databases are entities and the relationships among the entities. It is crucial to grasp the relationships among entities in relational database applications. It is difficult to differentiate the relationships in XML documents among corresponding entities since there is no such notion in XML. An attribute/element of an element can be an attribute of the element or an attribute of a relationship between the element and its containing element. In the following fragment of an XML documents, *title* is an attribute of *course*, however *grade* is not an attribute of *course*, it is an attribute of the relationship between *student* and *course*. The notion that a relationship has attributes is not used any more in database modeling these days although it was used to. This is modeled as an separate association entity which is ID-dependent in relational database modeling process. However, we make relationships have attributes.

```
<student...
  <course ...
    <title>Database</title>
    <credit>3</credit>
    <grade>A</grade>
  ...
```

The maximum cardinality of the relationship is also represented for 1:N, for N:1, and for M:N. The above fragment is rewritten as follows.

```
<!ELEMENT course (title, credit, grade)>
...
<!ATTLIST course cardinality #PCDATA>
<!ATTLIST grade relationship #PCDATA>
...
  <student ...
    <course cardinality=(student, M..N)>
      <title>Database</title>
      <credit>3</credit>
      <grade relationship=(student, course)> A
</grade>
```

The cardinality of the relationship between *student* and *course* is *M:N* and *grade* is an attribute of the relationship. Element *Title* and *credit* belong to course. An intersection table is created for *M:N* relationship, which contains the relationship attribute if exists. Here, the intersection table contains the identifiers of *student* and *course*, and *grade* attribute. In the following fragment, the cardinality of the relationship between

professor and *student* is *1:N*. The identifier of *professor* is placed in the *student* table as a foreign key.

```
<professor
  ...
  <student cardinality=(professor, 1..N)>
    <studentNumber>2008100</studentNumber>
    <name>Gildong Hong</name>
    <gpa>3.8</gpa>
  ...
```

When an element has the *cardinality* attribute, the element is treated as an entity. The relationship between an entity and the entity specified in the cardinality attribute like *professor* in the above example, can have the meaning of *containment* or *refer-to*. The relationship between student and professor is *refer-to*, which means that there is not composition relationship between them. There are two kinds of containment; *has-a* and *aggregation*. *Has-a* relationship has the same life time such as the relationship between car and chassis. *Aggregation* relationship has different life time between the contained entity and the containing entity such as car and tires.

3) Identifiers

Although XML provides *ID* attribute and *IDREF* attribute, it is not strong enough to enforce referential integrity. We need more strict notion of identifiers in relational database applications. In the following documents, *title* is used as an identifier. It can be used as a foreign key in representing relationships. As mentioned in the above, the element with *ID* attribute is enforced to be mapped to an entity.

```
<!ELEMENT student ...
<!ELEMENT course ...
<!ATTLIST course title ID #REQUIRED>
 ...
<student ...
  <course title=Database >
    <credit>3</credit>
    <grade>A</grade>
  ...
```

4) Multivalued attributes

An element can be mapped to a single attribute, but multiple occurrences of it are mapped to multiple attributes. The multiple attributes are represented as a separate entity which has the identifier of the containing entity and the corresponding attributes. In the following fragment, *hobby** is mapped to multivalued attribute. It is represented as an entity with the identifier, *stuNumber*, and corresponding values.

```
<!ELEMENT student(name, address, hobby* >
<!ATTLIST student stuNumber ID #REQUIRED>
 ...
<!ELEMENT hobby #PCDATA>
 ...
<student stuNumber=20091234>
  <sname>Gildong Hong</sname>
  <address>449 Wolyoung-dong</address>
  <hobby>tennis</hobby>
  <hobby>piano</hobby>
</student>
```

4. Representation of relational semantics

Let us take a simple example, an XML document in Fig. 2. We put extended relational semantics like the one in Fig. 3. We assume that the maximum cardinality of the relationship between *paper* and *author* is *1:N*(though it seems to be unlikely) and that of the relationship between *book* and *author* is *M:N*. The following three tables are created in conventional shredding method[22].

```
Paper(parentID, parentCode, paperID, ... year, title,...)
Book(parentID, parentCode, ... year, title,...)
Author(parentID, parentCode, ... Name, email,...)
```

In extended relational semantics method, four tables are generated as follows.

```
Paper(paperID,..., year, title,...)
Book(...,year, title,...)
Author(...,Name, email, paperID,...)
Book-Author(title, Name)
```

```
<publication>
<paper paperID="1">
    <year>2003</year>
    <title>Associaion Inling…</title>
    <authors>
        <author cardinality=(paper, 1..N)>
            <name>Byung-Joo Shin</name>
            <email>raniman@zeus.kyungnam.ac.kr</email>
        </author>
        <author cardinality=(paper, 1..N) >
            <name>Min Jin</name>
            <email>mjin@kyungnam.ac.kr</email>
        </author>
    </authors>
    <reference>
        <paper paperID="2">
            <year>1999</year>
            <title>Relational Databases…</title>
            <authors>
                <author cardinality=(paper, 1..N)>
                    <name>Shanmugasundaram.J</name>
                </author>
            </authors>
        </paper>
        <book>
            <year>2000</year>
            <title>Professional XML…</title>
            <authors>
                <author cardinality=(book, M..N)>
                    <name>Williams.K</name>
                </author>
            </authors>
        </book>
    </reference>
<book>
    <year>2002</year>
    <title>C++ XML</title>
    <authors>
        <author cardinality=(book, M..N)>
            <name>Arciniegas.F</name>
        </author>
    </authors>
</book>
.
:
</publication>
```

[Fig. 3] An XML document with extended relational semantics

The identifier *paperID* of the *paper* is added to the *Author* table and *Book-Author* table is crated for *M:N* relationship between *book* and *author*. Let us consider the following queries which are related with the relationships.

① for $p in paper//author[name="Byung-Joo Shin"] return $p/title/text()

② for $a in paper[title="Association Inlining ... "]//author return $a/name/text()

③ for $b in book//author[name="Byung-Joo Shin"] return $b/title/text()

④ for $a in book[title="Professional XML"]//author return $a/name/text()

It is straightforward to process query ① with extended relational semantics since *author* table has *paperID* as foreign key. However, it is not easy in conventional shredding[22]. After getting the given author in the *author* table, we have to find the corresponding *parentCode* which means paper and get the *parentID*. Finally we get the title of the paper with the *paperID* in the paper table. If the *author* table doesn't have the *parentCode* which means paper, we have to trace a few tables to get the final result. For query ②, it is easy to find the corresponding paperID which has the given title in *author* table with extended relational semantics. Unfortunately we have the same problem as query ① with traditional shredding method. It goes the same way for query ③ and ④.

5. Conclusion

XML is good at representing data with hierarchical structures. As XML is widely used in applications where semi-structured data are needed, semantics of XML has been extended in order to accommodate the requirements of these applications. In this paper, we surveyed semantic issues in XML data processing. They can be categorized into four groups; structural semantics, relational semantics, extended semantics and semantic web. The relational database is one of the alternatives for storing large amount of XML documents. Hence, Relational database characteristics are useful for managing XML data, where relational semantics can be exploited in processing XML data. Since there is no inherent semantics with XML, extension of semantics is required in advanced XML application fields. Web technology is expected to be developed in order to meet requirements of future web applications. Semantic web is an XML-based general purpose knowledge representation scheme which is machine accessible for the future web era. The standard form of the syntax of documents as well as the semantics of their contents must be defined so that machine can understand and process the documents. Hence, all XML documents should be rewritten in semantic web from the scratch. We propose the extended relational semantics which can be exploited in processing XML data when the documents are stored in relational databases.

For the future work, the proposed semantics should be validated with large volume of XML documents in order to show the efficacy in processing XML data processing.

References

[1] K.S. Beyer et al., "DB2 goes hybrid: Integrating native XML and XQuery with relational data and SQL," *IBM Systems Journal*, Vol. 45, No. 2. pp. 271-298, 2006

[2] R. Krishnamurthy, R. Kaushik and J. F. Naughton, "XML-to-SQL Query Translation Literature: The State of the Art and Open Problems," *The 1st International XML Database Symposium* pp. 1-18, 2003

[3] R. Murthy et al., "Towards an Enterprise XML Architecture," *Proceedings of the ACM SIGMOD International Conference on Management of Data,* pp. 953-957, 2005

[4] M. Nicola and B. V. Linden, "Native XML Support in DB2 Universal Database," *Proceedings of the 31st VLDB Conference,* pp. 1164-1174, 2005

[5] S. Pal et al., "XQuery Implementation in a Relational Database System," *Proceedings of the 31st VLDB Conference,* pp. 1175-1186, 2005

[6] P. S. Bayerl, H. Lungen, D. Goecke, A. Witt, "Methods for the Semantic Analysis of Document Markup", *Proceedings of ACM Synposium on Document Engineering*, pp. 161-170, 2003

[7] S. Decker, S. Meljik et al., "The Semantic Web: The Roles of XML and RDF", *IEEE Internet Computing*, pp. 63-74, September·October, 2000

[8] T. W. Ling and G. Dobbie, "Using Semantics in XML Data Management", *Proceedings of International Workshop on Scalable Web Information and Integration and Service in Conjunction with DASSFA*, March 2007

[9] Z. Bao, H. Wu, B. Chen, T. W. Ling, "Using Semantics in XML Query Processing", *Proceedings of ICUIMC 2008*, pp. 157-162, 2008

[10] H. Wu, T. W. Ling, and B. Chen, "VERT: a semantic approach for content search and content extraction in XML query processing", *Proceedings of ER 2007*, 2007

[11] K. Yokota, T. Kunishima, and B. Liu, "Semantic Extensions of XML for Advanced Applications", *Proceedings of Information Technology for Virtual Enterprises*, pp. 49-57, Jan. 2001

[12] R. Nayak, "Investigating semantic measures in XML clustering", *Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, 2006

[13] J. Chu-carroll, J. Prager, K. Czuba, D. Ferrucci, and P. Duboue, "Semantic Search via XML Fragments: A High-Precision Approach to IR", *Proceedings of SIGIR 2006*, pp. 445-452, 2006

[14] M. Theobald, R. Schenkel, G. Weikum, "Exploiting Structure, Annotation, and Ontological Knowledge for Automatic Classification of XML Data", *Proceedings of International Workshop on the Web and Databases*, 2003

[15] A. Renear, D. Bubin, C.M.Sperberg-McQueen, C. Huitfeldt, "Towards a Semantic for XML markup", *Proceedings of ACM Synposium on Document Engineering*, pp. 119-126, 2002

[16] A. Renear, D. Dubin, and C. M. Sperberg-McQwueen, "XML Semantics and Digital Libraries", *Proceedings of Joint Conference on Digital Libraries*, pp. 303-305, 2003

[17] S. S. Sane and A. Shirke, "Generating OWL Ontologies from a Relational Databases for the semantic Web", *Proceedings of International Conference on Advances in Computing, Communication and Control*, pp. 157-162, 2009

[18] G. Antoniou and F. V. Harmelen, *A Semantic Web Primer*, MIT Press, 2008

[19] I. F. Cruz, H. Xiao, F. Hsu, "An Ontology-based Framework for XML Semantic Integration", *Proceedings of the International Database Engineering and Application Symposium*, 2004

[20] J. Shanmugasundaram, K. Tufte, G. He, C. Zhang, D. Dewitt, J. Naughton, "Rela-tional Databases for Querying XML Documents: Limitations and Opportunities", *Proceedings of the 25th VLDB Conference* pp. 302-314, 1999

[21] S. Amer-Yahia, F. Du, and J. Freire, "A Comprehensive Solution to the XML-to-Relational Mapping Problem", *Proceedings of International Workshop on Web Information and Data Management*, pp. 31-38, 2004

[22] B. J. Shin, M. Jin, "Association Inlining for Mapping XML DTDs to Relational Tables", *Proceedings of the 2004 International Conference on Computational Science and its Applications,* pp. 849-858, 2004

Min Jin [Regular member]

- Feb. 1982 : Seoul National U., Computer Science & Statistics B.S
- Feb. 1984 : KAIST, Computer Science, M.S
- Aug. 1997 : U. of Connecticut. Computer Science & Eng., Ph.D.
- March 1985 ~ current : Kyungnam U., Div. of Computer Science and Eng., Professor

<Research Interests>
Database, Data modeling, Object-oriented database
XML storage and processing