인라이닝에 기반한 XML 스키마의 관계형 스키마 변환 기법

조 정 길†

요 약

데이터 중심의 XML 문서를 관계형 데이터베이스에 저장하고 관리할 경우에 XML 스키마로부터 관계형 스키마를 추출하는 것이 보다 시급한 일이다. 또한 생성된 테이블에 XML 문서를 분할하여 저장할 경우에 많은 널 값을 초래하거나 조인 비용의 증가를 가져오기 때문에 이에 대한 해결책이 필요하다. 이 논문에서는 XML 스키마로부터 관계형 스키마를 생성하는 Schema Hybrid Inlining 기법을 제안하였다. 제안된 기법은 XML 스키마 그래프를 바탕으로 관계형 스키마를 생성하게 되는데, 기존의 Inlining 방식을 확장하여 출현 지시자와 진입 차수에서의 테이블 생성 방법을 휴리스틱하게 매뉴얼 처리를 하며, 유도 관계에서 최종 노드인 조상 노드는 새로운 테이블을 생성한다. 또한 DTD 종속적인 저장 방식의 단점인 조인 연산 비용을 줄이기 위하여 중복을 활용한 분할 저장과 구조적 검색 기법을 개선한 관계 경로 요소 정보를 이용하였다.

A Transformation Technique of XML Schema into Relational Schema Based-on Inlining

Jung Gil Cho†

ABSTRACT

When any data-centric XML documents are stored and managed in RDBMS, schema extracting from XML Schema is an imminent problem. Furthermore, when they are stored in partitioned way on created table, lots of null values will be produced and/or be increased cost for join, so we need a solution to solve these problems. This paper proposes a Schema Hybrid Inlining technique to generate relational schema of XML documents. The suggested technique creates a relational schema based on the XML Schema graph. Also, the technique expands the legacy Inlining method by manual and heuristic processing table generation method of cardinality and in-degree. Then, an ancestor node, terminal node, and creating a new table on the derived relation in this technique. DTD-dependent storage method uses partitioned storing and relation path element information reformed structured-searching method to reduce joining operation cost that is a weak point of it.

키워드 : XML, XML 스키마(XML Schema), 인라이닝(Inlining), XML 스키마 변환(XML Schema Transformation)

1. 서 론

대표적인 반구조적 데이터(semistructured data)인 XML 문서들이 웹에 많아짐에 따라 데이터들간의 의미적, 구조적 관계를 설정하는 스키마를 추출하고 그에 따라 데이터를 구조화시켜 정보로서의 가치를 만들 수 있는 새로운 저장 기법들이 연구되어 왔다. 이러한 연구 바탕에 XML 문서를 데이터베이스에 저장하고 검색하기 위한 연구가 활발히 수행되고 있다[1-5].

XML 문서는 문서 중심의 XML 문서와 데이터 중심의 XML 문서로 구분된다. 전자는 엘리먼트간의 순서가 중요하고, 기존의 DTD 정보만 가지고도 충분하다. 그러나 후자는 엘리먼트간의 순서는 중요하지 않고 계층 구조만이 의미를 가지며, 오로지 문자열 형식만을 지원하는 DTD로는 부족함이 많다. 이에 새로운 스키마인 XML 스키마를 사용하면 매우 복잡한 구조의 XML 문서를 정의할 수가 있고 다양한 데이터 형식으로 표현할 수가 있으며, 무엇보다도 인스턴스 수준에서 동적으로 문서의 구조를 결정할 수 있는 방법을 제공하고 있다[6-8].

데이터 중심의 XML 문서는 관계형 데이터베이스를 이용하여 저장 관리할 경우에 많은 잇점이 있다. 상용화된 제품들이 많으며, 그 동안 많은 개발자들에 의하여 성능이 향상되었기 때문에 보다 효율적으로 XML 문서들을 저장하고 검색할 수가 있다[9]. 이러한 연구는 주로 DTD를 기반으로 하는 XML 문서를 관계형 데이터베이스에 저장하는 방법에 대하여 연구되어 왔으나 앞으로 XML 스키마가 데이터 중심의 XML 문서에 보다 자주 사용되리라 예상된다.

이 논문의 목적은 XML 스키마로부터 질의 시에 조인 연산 비용을 줄이고, 구조적 검색이 가능한 DTD 종속적인 저장 방식의 관계형 스키마를 생성하는 것이다. 따라서,

† 정 회 원 : 성결대학교 컴퓨터공학부 교수
논문접수 : 2003년 6월 9일, 심사완료 : 2004년 7월 13일

XML 데이터를 관계형 데이터베이스에 저장하기 위하여 XML 스키마로부터 DTD 종속적인 저장 방식의 관계형 스키마를 생성하는 기법인 SHI(Schema Hybrid Inlining)를 제안한다.

2. 관련 연구

기존의 관계형 데이터베이스에 XML 문서를 저장하기 위한 스키마 생성 기법에 관한 여러 연구들이 있었다. 이러한 XML 문서의 저장 방법은 크게 DTD나 XML 스키마와 같은 스키마 정보가 없는 경우와 있는 경우로 나눌 수가 있다.

스키마 정보가 없는 경우에는 XML 문서로부터 스키마 정보를 추출해 내어 관계형 테이블들을 생성하는 방법과 스키마 정보는 무시하고 XML 문서를 구성하는 요소들인 노드와 연결선들만을 저장하는 방법이 있다. 전자의 경우에는 STORED[1]가 있고, 후자의 경우에는 반구조적 데이터를 저장하기 위한 방법[2]과 Binary 방식[10, 3]이 있다.

스키마 정보가 있는 경우에는 DTD나 XML 스키마로부터 관계형 스키마를 생성하는 방법인 X-Ray[11], XML-DBMS [12], Inlining[4], Extended Inlining[5]등이 있다.

Inlining에서는 XML 문서의 DTD를 기반으로 관계형 스키마를 추출하는 기법을 제안하였다. DTD 그래프를 만들기 전에 세 가지 타입의 변환인 분배 변환, 단순 변환, 그룹 변환을 하여 DTD 단순화 작업을 한다. DTD 단순화 작업을 마친 후에 표현될 수 있는 정보들만을 이용하여 DTD 그래프와 엘리먼트 그래프를 만든다. DTD 그래프는 DTD의 구조를 표현한다. 노드는 DTD에 있는 엘리먼트, 속성, 연산자를 나타낸다. 각각의 엘리먼트는 DTD 그래프에 한번만 나타나며, 속성과 연산자는 DTD에서 나타난 횟수만큼 나타난다. 또한 DTD 그래프에서 사이클은 재귀를 나타낸다. 엘리먼트 그래프 생성 방법은 DTD 그래프에서 깊이 우선 탐색(DFT)을 하며, 테이블을 구축하려는 엘리먼트 노드에서 시작한다. 이때 각각의 노드는 처음 도착시에 'visited'로 표시하며, 각각의 노드는 일단 모든 자식들을 탐색하면 그것을 표시하지 않는다.

Inlining[4] 기법에서는 DTD를 관계형 스키마로 변환하기 위하여 세 가지 관계형-변환 알고리즘인 Basic Inlining 기법(Basic), Shared Inlining 기법(Shared), Hybrid Inlining 기법(Hybrid)을 제시하였다. Basic 기법은 엘리먼트 그래프를 깊이 우선 순회하면서 방문하여 엘리먼트들을 하나의 테이블에 인라인 시키는 방법을 사용하였다. Shared 기법은 DTD 그래프를 순회하면서 루트 노드에 해당하는 엘리먼트에서만 테이블을 생성한다. 자식 엘리먼트 노드들을 방문하면서 엘리먼트들을 인라인 시키는 것은 Basic과 같지만, 추가적으로 다른 테이블과 공유되어질 수 있는 엘리먼트들을 만나게 되면 인라인 시키지 않고 독립된 테이블로 처리하므로써 Basic의 단점을 해결하였다. Hybrid 기법은 앞의 두 기법을 혼합한 것이며, 가장 좋은 성능 평가를 얻은 기법이다. 이 기법은 DTD 구조를 이용함으로써 가능한 적은 수의 테이블로 XML 데이터를 사상시킨다. Shared에서 인라인 되지 않은 엘리먼트 중에서 *로 연결되어 있지도 않고 재귀적으로 연결된 경우가 아니라면 다른 테이블에 인라인 시키는 방법으로 Shared의 경우보다 조인의 횟수를 줄이는 성능 향상 효과가 있다.

Extended Inlining[5]에서는 DTD 기반의 Inlining을 확장하여 XML 스키마를 기반으로 관계형 스키마를 추출하는 기법을 제안하였다. XML 스키마 데이터 형식의 변환 처리와 단순화 작업을 하고, XML 스키마에서 표현될 수 있는 정보들만을 이용하여 스키마 그래프와 타입 그래프를 만든다. 데이터 형식들과 엘리먼트들간의 관계를 스키마 그래프로 표시하며, 스키마 그래프에서 특정한 타입에 대한 타입 그래프를 생성한다. 상속 관계 표시는 엘리먼트와 복잡 형식들간의 유도 관계를 표시하며, 타입 관계 표시는 엘리먼트와 데이터 형식들간의 관계를 나타낸다. 또한 내포 관계 표시는 데이터 타입과 엘리먼트간의 관계를 나타낸다.

Extended Inlining에서는 XML 스키마를 관계형 스키마로 변환하기 위하여 두 가지 관계형-변환 알고리즘인 Basic Extended Inlining과 Hybrid Extended Inlining을 제안하였다. Basic Extended Inlining 기법은 타입 그래프를 깊이 우선 순회하면서 방문하여 엘리먼트들을 하나의 테이블에 인라인 시키는 방법을 사용하고 있다. Hybrid Extended Inlining 기법은 Basic Extended Inlining 기법에서 테이블의 개수와 중복되는 칼럼의 수가 매우 커지는 단점을 해결하였다.

관계형 테이블을 모델링 할 때에 테이블의 수와 테이블에 인라인 되는 칼럼은 테이블의 성능 평가에 중요한 요소가 된다. 또한 XML 스키마의 데이터 구조나 출현지시자인 maxOccurs의 값에 따라 테이블 생성에 많은 영향을 미친다. 그리고 XML 데이터가 생성된 테이블에 따라 분할 저장되는 데에 따른 조인 횟수의 증가도 해결해야할 과제이다. 따라서 XML 스키마의 데이터 구조와 출현지시자인 maxOccurs의 값을 분석하여 테이블 생성에 최적의 조건을 제시하는 연구가 필요하다. 또한 테이블 생성에 중복되는 칼럼을 추가하여 분할 저장된 XML 문서의 조인 횟수를 줄이며, 분할된 테이블의 구조 정보 및 검색 정보를 표현하여 구조적 검색이 가능하도록 하는 연구가 필요하다.

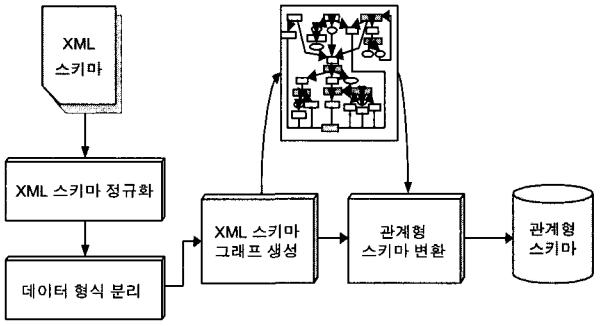
3. XML 스키마의 관계형 스키마 변환 기법

이 논문에서는 XML 스키마의 관계형 스키마 변환에서 문서의 구조 파악이 용이하도록 문서의 구조를 나타내는 엘

리먼트들과 데이터 형식들간의 관계를 표시한 XML 스키마 그래프와 테이블 생성 방법인 Schema Hybrid Inlining(SHI)를 제안한다. 제안된 SHI는 DTD 종속적인 저장 방식의 단점인 조인 연산 비용의 증가를 해결하기 위하여 출현 지시자와 진입 차수의 테이블 생성 기준을 휴리스틱하게 매뉴얼 처리를 하며, 중복을 활용한 분할 저장 방식과 유도 관계에 의한 테이블 생성 방식을 이용한다.

3.1 관계형 스키마 변환 절차

관계형 스키마 변환 절차는 (그림 1)과 같으며, XML 스키마에서 관계형 스키마를 변환하는 전체 과정을 나타낸 것이다.



(그림 1) 관계형 스키마 변환 흐름도

우선 XML 스키마 문서에서 필요 없는 요소들을 제거하기 위한 작업과 함께 실제적으로 이들을 XML 스키마 파일에서 제거하는 XML 스키마 정규화 작업을 하며, 엘리먼트, 데이터 형식, 속성을 분리하는 데이터 형식 분리 처리를 한다. 다음으로 각각의 형식이 분류되어 테이블에 저장된 데이터로 XML 스키마 그래프를 생성한다. 마지막으로 XML 스키마 그래프를 가지고 관계형 스키마 변환을 하여 관계형 스키마를 생성한다.

3.2 XML 스키마 정규화

정규화는 단순화와 데이터 형식 변환으로 처리된다. XML 스키마 단순화는 XML 스키마에서 관계형 테이블을 생성하는데 없어도 되는 요소들을 제거하여 XML 스키마를 관계형 스키마로 쉽게 생성할 수 있도록 하는 과정이다.

태그 안에 순서대로 열거되어 있는 엘리먼트들을 정의하는 <sequence>는 관계형 데이터베이스의 테이블로 매핑할 때에 스키마에 나타난 순서대로 왼쪽에서부터 오른쪽 칼럼으로 생성되기 때문에 스키마 파일에서 제거하여도 무방하다. <choice>는 여러 개의 자식 엘리먼트중에 한 개를 선택하므로 <sequence>와 같이 스키마 파일에서 삭제가 가능하다. 또한 <all>은 중첩이 불가능하고, 이 태그 안에 나타날 수 있는 출현 지시자인 minOccurs와 maxOccurs의 값은 0이나 1인 둘 중에서 하나만 올 수가 있다. 이 때에 기본값(minOccurs = "1", maxOccurs = "1")인 경우에는 <all>을 제거한다.

XML 스키마의 정규 데이터 형식에 비하여 변환하는 데이터베이스의 데이터 형식이 한정되어 있기 때문에, 보통 관계형 데이터베이스 시스템에서는 XML 스키마에서 제공하는 정규 데이터 형식을 완벽하게 지원하지는 못한다. 예를 들어, SQL Server 2000에서는 문자 데이터 형식을 위하여 char, varchar, text, nchar, nvarchar, ntext와 숫자 데이터 형식을 위하여 int등의 14가지 형식의 정규 데이터 형식이 있다. 따라서, XML 스키마의 데이터 형식이 double인 경우에 구축하여 놓은 형식 변환 정보 테이블을 사상하여 관계형 스키마의 정규 데이터 형식인 real로 변환한다.

3.3 XML 스키마 그래프

XML 스키마는 다양한 정보들을 담고 있고 매우 복잡하다. 이러한 XML 스키마로부터 관계형 테이블을 생성하기 위해서는 XML 스키마의 필요한 정보를 그래프로 표현한 XML 스키마 그래프를 생성한다.

(그림 2)는 "책"을 주제로 XML 문서를 작성하기 위한 기준 XML 스키마 문서이다.

```
<?xml version = "1.0" encoding = "UTF-8"?>
<schema xmlns = "http://www.w3.org/2001/XMLSchema"
  elementFormDefault = "qualified">
    <element name = "article">
      <complexType>
        <sequence>
          <element ref = "title"/>
          <element name = "author" type = "authorType"
            minOccurs = "0"  maxOccurs = "unbounded"/>
          <element name = "contactauthor"
            type = "contactauthorType" minOccurs = "0"/>
        </sequence>
      </complexType>
    </element>
    <complexType name = "authorType">
      <sequence>
        <element name = "name" type = "nameType"/>
        <element name = "address" type = "addressType"/>
      </sequence>
      <attribute name = "id" type = "ID" use = "required"/>
    </complexType>
    <element name = "book">
      <complexType>
        <sequence>
          <element ref = "booktitle"/>
          <element name = "author" type = "authorType"/>
        </sequence>
      </complexType>
    </element>
    <complexType name = "contactauthorType">
      <attribute name = 'authorID" type = "IDREF"/>
    </complexType>
    <complexType name = "editorType">
      <sequence>
        <element name = "monograph" type = "monographType"
```

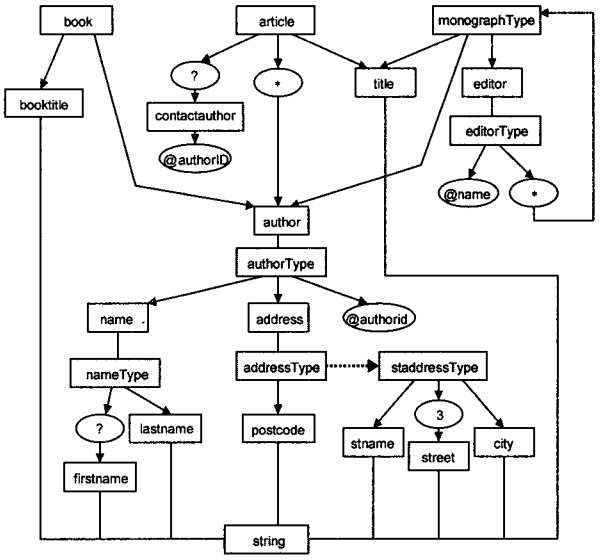
```
        minOccurs = "0" maxOccurs = "unbounded"/>
      </sequence>
      <attribute name = "name" type = "string" use = "required"/>
    </complexType>
    <complexType name = "monographType">
      <sequence>
        <element ref = "title"/>
        <element name = "author" type = "authorType"/>
        <element name = "editor" type = "editorType"/>
      </sequence>
    </complexType>
    <complexType name = "nameType">
      <sequence>
        <element ref = "firstname" minOccurs = "0"/>
        <element ref = "lastname"/>
      </sequence>
    </complexType>
    <complexType name = "addressType">
      <complexContent>
      <extension base = "staddressType">
          <sequence>
            <element name = "postcode" type = "postalcode"/>
          </sequence>
        </extension>
      </complexContent>
    </complexType>
    <complexType name = "staddressType">
      <sequence>
        <element name = "stname" type = "string"/>
        <element name = "street" type = "string" minOccurs = "1"
         maxOccurs = "3"/>
        <element name = "city" type = "string"/>
      </sequence>
    </complexType>
    <simpleType name = "postalcode">
      <restriction base = "string">
        <pattern value = "[0-9]{3}-[0-9]{3}"/>
      </restriction>
    </simpleType>
    <element name = "title" type = "string"/>
    <element name = "firstname" type = "string"/>
    <element name = "lastname" type = "string"/>
    <element name = "booktitle" type = "string"/>
</schema>
```

(그림 2) 책에 대한 XML 스키마

XML 스키마 엘리먼트와 데이터 형식을 위한 구조는 XML 스키마 그래프로 표현할 수 있다. XML 스키마 그래프에는 엘리먼트들과 데이터 형식들 간의 관계를 표시하고 데이터 형식들 간의 유도 관계를 나타낼 수 있다. DTD에서는 엘리먼트를 기본 요소로 하여 DTD 그래프를 작성한다. 그러나 XML 스키마에서는 엘리먼트가 데이터 형식에 따라 여러 번 나타날 수 있기 때문에 엘리먼트와 데이터 형식을 기본 요소로 하여 XML 스키마 그래프를 작성한다.

(그림 2)의 XML 스키마 문서를 XML 스키마 그래프로 표현하면 (그림 3)과 같다.

노드는 XML 스키마에 있는 엘리먼트, 데이터 형식, 속성, 출현 지시자를 나타낸다. 각각의 데이터 형식들은 XML 스키마 그래프에서 한번만 나타나며, 엘리먼트, 속성, 출현 지시자는 XML 스키마에서 나타난 횟수만큼 나타난다. XML 스키마 그래프에서 회색 직사각형노드는 데이터 형식을 나타내며, 흰색 직사각형 노드는 엘리먼트를 나타낸다. 또한 타원형 노드는 속성, 출현 지시자를 나타내는데, 속성의 이름에는 @을 추가한다. 출현 지시자인 minOccurs와 maxOccurs는 카디널리티로 표현한다. 출현 지시자 노드에서 ?는 minOccurs = "0"과 maxOccurs = "1"이며, *는 maxOccurs = "unbounded"로서 제한이 없음을 나타내고, minOccurs = "1"과 maxOccurs = "1"일 때에는 생략한다.



(그림 3) XML 스키마 그래프

그래프에서 연결선은 세 가지 종류로 구분한다. 데이터 형식의 상속을 나타내는 유도 관계는 화살표가 있는 점선으로 표시한다. 유도 관계 구조는 XML 스키마의 복잡 형식들을 대상으로 구성하며, 단순 형식은 모두 루트 형식에 해당한다. 단순 형식으로 상속된 단순 형식들도 루트 형식으로 처리하며, 이들간의 유도 관계는 표시하지 않는다.

또한 부모 요소인 엘리먼트나 복잡 형식과 자식 요소들인 엘리먼트, 속성, 출현 지시자들과의 관계를 나타내는 내포 관계는 화살표가 있는 실선으로 표시한다. 그리고 엘리먼트와 데이터 형식의 관계를 나타내는 형식 관계는 화살표가 없는 실선으로 나타낸다.

3.4 관계형 스키마 변환

3.4.1 테이블 생성 기준

테이블 생성 기준에 따라 XML 스키마 그래프를 순회하면서 다음과 같은 방법으로 관계형 스키마를 생성한다.

첫째, 진입 차수(in-degree)가 0인 엘리먼트 노드는 개별적인 테이블을 생성한다. 이는 루트 엘리먼트에 해당하는 경우로 어떤 다른 노드로부터도 도달하지 않기 때문이다.

둘째, 재귀 형태인 사이클이 형성되어 있는 경우에는 개별적인 테이블을 생성한다. 셋째, 주어진 데이터 형식에 포함되어 있는 엘리먼트의 출현 지시자인 maxOccurs의 값이 2 이상일 때 독립적인 테이블을 생성한다. 넷째, 진입 차수가 2 이상인 엘리먼트 노드는 독립적인 테이블 생성하고, 진입 차수가 1인 노드들은 인라인한다. 다섯째, 복잡 형식들의 유도 관계에서 확장에 의한 유도인 경우에 상속으로 연결된 최종 노드인 조상 노드는 개별적인 테이블을 생성한다.

3.4.2 XML 데이터의 중복을 활용한 분할 저장

관계형 스키마로 분할되어 저장된 XML 데이터에 대해 고전적인 방법을 적용함으로 조인 비용을 줄일 수가 있다. ID 칼럼 중복과 Value 칼럼 중복은 In-place 중복과 Separate 중복이 있다. In-place 중복은 조상 테이블의 ID 칼럼의 사본이 자손 테이블 내에 중복되는 것이고, Separate 중복은 중복되는 ID 칼럼들을 별도의 테이블로 유지하는 것이다. In-place 중복은 중복 데이터를 갖는 모든 테이블을 찾아서 갱신해야하는 단점은 있지만 하나의 경로 상에서의 한 노드의 중복이 여러 테이블에 일어나고 않고, 또한 갱신되어야 할 레코드의 수가 대량인 경우가 아닌 경우에는 유용하다. 따라서 빠른 갱신 수행을 위해서는 Value 칼럼과 ID 칼럼이 함께 중복되어야 하며, 테이블 생성에 ID 칼럼과 Value 칼럼의 In-place 중복을 활용하여 질의 처리 시에 효율성을 높인다.

예를 들어, (그림 3)의 XML 스키마 그래프에서 book 노드의 자식 노드인 authorType은 article 노드의 출현 지시자인 *에 의하여 새로운 테이블을 생성한다. 또한 authorType은 출현 지시자인 *을 제외하고도 book과 monographType에서의 진입 차수가 2가 되므로 두 테이블에 값이 중복되는 경우가 발생한다. 이러한 경우에는 Value 칼럼 중복의 경우로 staddressType을 가르키는 ID 칼럼과 함께 중복으로 처리한다. 또한 조상 테이블의 ID 칼럼과 테이블 이름의 사본인 parentID, ParentTable을 중복하여 처리한다. 테이블 생성 방법과 ID 칼럼과 Value 중복에 의해서 생성된 (밑줄 친 부분) book 테이블은 (그림 4)과 같다.

book(bookID, booktitle, authorType.nameType.firstname, authorType.nameType.lastname, authorType.addressType.postcode, authorType.@authorid, authorType.addressType.parentID, authorType.addressType.parentTable)

(그림 4) 중복에 의하여 생성된 book 테이블

3.4.3 유도 관계에 의한 테이블 생성 방법

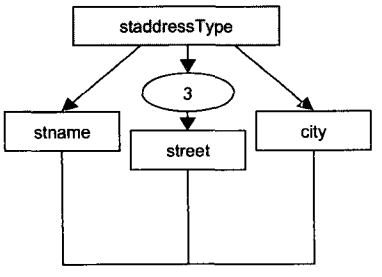
XML 스키마에서 데이터 형식의 상속은 유도 관계로서 표현한다. 이러한 유도 관계는 제한(restriction), 확장(extension)을 사용하여 정의할 수가 있다.

제한에 의한 유도에는 단순 형식과 복잡 형식이 유도가 되며, 확장에 의한 유도에는 복잡 형식만 유도가 된다. 제한에 의한 유도에는 상속하는 부모 데이터 형식에 제한을 가하는 경우이기 때문에 새로운 테이블을 추가할 필요가 없고 부모 노드에 칼럼으로 인라인한다. 그러나 확장에 의한 유도에는 기존의 내용 모델인 텍스트 내용, 엘리먼트 내용, 혼합 내용에 단순 형식, 복잡 형식, 자식 엘리먼트, 속성등이 추가되어 새로운 복잡 형식을 만든다. 이러한 경우에는 관계형 스키마 생성 시에 부모 복잡 형식에 테이블이나 칼럼이 추가된 상태이므로, 상속받은 곳에서 다시 추가하게 되면 중복이 발생한다. 그러므로, 유도 관계로 연결된 최종 단말 노드인 조상 노드는 새로운 테이블을 생성한다.

3.4.4 휴리스틱 매뉴얼 인라이닝

maxOccurs의 값이 2 이상인 엘리먼트는 상황에 따라 독립적인 테이블을 생성한다. 이는 Inlining의 엘리먼트 그래프에서 * 연산자를 만나는 경우인 다중 값을 갖는 엘리먼트와 같다. 예를 들어, (그림 3)에서 author 엘리먼트 노드와 형식 관계인 authorType 복잡 형식 노드는 개별적인 테이블을 생성한다.

그러나 maxOccurs의 값이 2, 3, 4 등과 같이 적고 또한 밑의 자식 노드의 규모가 작은 경우에는 상황에 따라 테이블에 칼럼으로 인라인시킨다. 이러한 경우는 실행자가 화면에서 시뮬레이션된 각 요소의 최적 결과를 보고 판단하여 매뉴얼 처리를 한다. 예를 들어, (그림 5)에서 출현 지시자인 maxOccurs가 3이지만 내포된 자식 노드가 한 개밖에 없기 때문에 street를 maxOccurs의 숫자만큼 staddressType에 인라인 시키는 것이 효율적인 방법이 된다.



staddressType(ID, stname, street, street, street, city)

(그림 5) 출현 지시자를 포함한 XML 스키마 그래프와 테이블

진입 차수가 2 이상인 엘리먼트 노드는 자식 엘리먼트의 규모인 너비(width)와 깊이(depth)에 따라 인라인과 테이블 생성을 선택한다. (그림 3)에서 authorType 노드는 * 출현 지시자가 없는 경우에 진입차수가 2가 되는데, 이러한 경우에 book과 monographType 테이블에 각각 칼럼으로 인라인하게 된다. 이때에 authorType 노드에 속해있는 name,

address, @authorid 노드가 각각의 테이블에 칼럼으로 중복 인라인 된다. 이런 방법은 복잡한 XML 스키마인 경우에 과중한 칼럼의 중복을 초래하여 심각한 이상 현상이 발생하게 된다. 이러한 경우에도 엘리먼트 출현 지시자와 같은 방법으로 시뮬레이션된 각 요소의 최적 결과를 보고 판단하여 매뉴얼 처리를 한다.

(그림 2)의 XML 스키마를 SHI의 테이블 생성 기준과 휴리스틱 매뉴얼 인라이닝에 적용하면 (그림 6)과 같은 관계형 스키마가 생성된다.

```
book(bookID, booktitle, authorType.nameType.firstname,
     authorType.nameType.lastname, authorType.addressType.postcode,
     authorType.@authorid, authorType.addressType.parentID,
     authorType.addressType.parentTable)
article(articleID, contactauthor.@authorID, title)
monographType(monographTypeID, title, authorType.nameType.firstname,
     authorType.nameType.lastname, authorType.addressType.postcode,
     authorType.@authorid, editorType.@name,
     authorType.addressType.parentID,
     authorType.addressType.parentTable)
authorType(authorTypeID, nameType.firstname, nameType.lastname,
     addressType.postcode, @authorid, parentID, parentTable,
     addressType.parentID, addressType.parentTable)
staddressType(staddressTypeID, stname, street, street, street, city)
```

(그림 6) 생성된 관계형 스키마 테이블

3.5 관계 경로 요소 정보

관계 경로 요소 정보는 (그림 3)의 직사각형 노드 중에서 테이블로 생성되는 루트 엘리먼트와 복잡 형식 노드에 대해서만 기록한다. 이것은 테이블로 생성되는 노드로부터의 경로 표현인 Q와 그 노드 요소에 포함된 엘리먼트, 복잡 형식, 속성 경로 표현의 집합인 {P_1, …, Pn}의 쌍으로 구성되며, (Q, {P_1, …, Pn})로 정의한다. 이 때에 Q와 {P_1, …, Pn}은 XPath 형식의 경로를 사용한다. Q의 경로는 절대 경로로 나타내고, {P_1, …, Pn}의 경로는 상대 경로로 Q의 경로를 갖는 테이블에 포함되어 있는 칼럼들의 경로나 하위 테이블의 경로를 나타낸다.

(그림 3)으로부터 생성한 관계 경로 요소 정보는 (그림 7)과 같이 표현되고, 함께 생성되는 관계형 스키마의 테이블들은 (그림 6)과 같다.

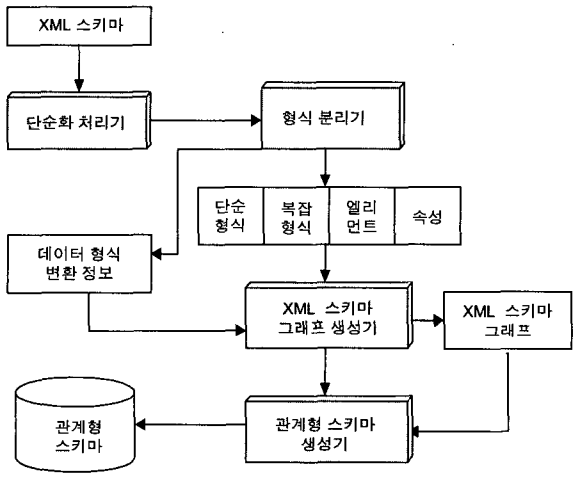
```
1 : (book, {bookID, booktitle, //authorType})
2 : (book/authorType, {authorTypeID, parentID, nameType.firstname,
     nameType.lastname, nameType.postcode, //addressType.staddressType,
     @authorid})
3 : (book/authorType/addressType/staddressType,
     {staddressTypeID, parentID, stname, street, city})
4 : (article, {articleID, contactauthor, contactauthor.@authorID,
     //authorType, title})
5 : (article/authorType, {authorTypeID, parentID, nameType.firstname,
     nameType.lastname, nameType.postcode, //addressType.staddressType,
     @authorid})
6 : (article/authorType/addressType/staddressType,
     {staddressTypeID, parentID, stname, street, city})
7 : (monographType, {monographTypeID, title, //authorType,
     editorType.@name})
8 : (monographType/authorType, {authorTypeID, parentID,
     nameType.firstname, nameType.lastname, nameType.postcode,
     //addressType.staddressType, @authorid})
9 : (monographType/authorType/addressType/staddressType,
     {staddressTypeID, parentID, stname, street, city})
```

(그림 7) 관계 경로 요소 정보의 예

4. 구현 및 고찰

4.1 구 현

이 논문에서 설계한 시스템 구성도는 (그림 8)과 같으며, 시스템을 구성하는 세부 모듈은 단순화 처리기, 형식 분리기, 엘리먼트 처리기, 단순/복잡 형식 처리기, 속성 처리기, XML 스키마 그래프 생성기, 관계형 스키마 생성기로 구성된다.



(그림 8) 시스템 구성도

단순화 처리기에서는 XML 스키마 문서에서 필요 없는 요소들을 제거하기 위한 작업과 함께 실제적으로 이들을 XML 스키마 파일에서 제거한다. 형식 분리기에서 엘리먼트는 전역 엘리먼트와 엘리먼트 그룹에서 처리되며, 데이터 형식은 단순 형식과 복잡 형식으로, 속성은 전역 속성과 속성 그룹에서 구분되어 처리된다. 단순 형식의 데이터 형식에서 변환되는 데이터 형식 변환 정보는 사용하려는 관계형 데이터베이스 시스템의 데이터 형식으로 사상하는데 이용한다. XML 스키마 그래프 생성기에서는 전 단계인 형식 분리기에서 분리한 엘리먼트, 단순 형식, 복잡 형식, 속성, 데이터 형식 변환 정보를 가지고 XML 스키마 그래프를 작성한다. 그리고 관계형 스키마 생성기는 XML 스키마 그래프와 엘리먼트의 정보를 참조하여 관계형 스키마를 생성한다.

4.2 실험 및 평가

구현한 시스템의 성능 평가는 테이블 크기, 성능 비교, 구조적 검색 지원으로 평가한다. 평가에서는 관계형 스키마를 생성하는 기존의 생성 방식인 Hybrid Inlining(HI) 기법과 Hybrid Extended Inlining(HEI) 기법, 그리고 이 논문에서 제안한 방식인 SHI 기법을 비교 대상으로 각각 적용하였다.

테이블 크기 평가를 위해 MPEG-7 표준의 MDS(Multimedia Description Schemas) Schema[13]를 대상으로 각 모델의 기법으로 관계형 테이블을 생성하였다. 실험 대상 데이터인 MDS Schema는 크기가 248KB이며, 301개의 complexType으로 기술되어있는 방대한 양의 XML 스키마 문서이다. 성능 비교를 위하여 MDS Schema의 DescriptionMetadataType 형식의 XML 스키마를 가지고 각 모델의 기법으로 관계형 스키마를 생성하였으며, 총 20.2MB 분량의 XML 문서 1500개를 각 기법으로 생성된 테이블에 저장하였다.

4.2.1 테이블 크기

이 논문에서 제안한 SHI에서는 테이블 생성 기준인 Auto와 휴리스틱 매뉴얼 인라이닝 방법인 Manual 처리로 구분하여 테이블을 생성하였다. Auto는 테이블 생성 기준에 따르며, Manual은 진입 차수와 출현 지시자인 두 요소의 자손 총 노드의 수를 자동으로 시뮬레이션하여 최적의 결과를 추출한다.

테이블 크기의 실험 결과는 <표 1>과 같다. HEI에서 평균 칼럼 수가 제일 적게 나타난 가장 큰 요인은 <choice> 태그의 처리를 모두 새로운 테이블로 생성했기 때문이다. SHI는 다른 기법과 달리 기본 스키마 정보를 위한 테이블로 4개가 추가되었으며, 중복을 활용한 분할 저장 방식으로 인하여 HEI에 비하여 칼럼 수가 늘어나는 요인이 되었다.

〈표 1〉 테이블 크기 실험 결과

	HI	HEI	SHI	
			Auto	Manual
총 테이블 수	346	288	336	280
테이블중의 최대 칼럼 수	200	71	195	82
테이블당 평균 칼럼 수	30	15	32	21
칼럼이 100개 이상 있는 테이블 수	14	0	6	0

SHI의 Auto와 Manual에서는 총 테이블의 수에서 많은 차이를 보이고 있다. 그 이유는 Manual의 경우에 진입 차수와 maxOccurs의 자손 총 노드 수를 감안하여 매뉴얼로 처리함으로써 Auto에 비하여 테이블의 수를 줄일 수 있었기 때문이다. 이렇게 함으로써 테이블의 생성 개수를 줄여서 질의 처리 시에 조인 횟수를 줄이는 효과를 얻을 수가 있었다.

4.2.2 성능 비교

HI, HEI, SHI Manual 기법을 다양한 종류의 SQL 질의를 통하여 평가하였으며, 평가 요소를 측정하기 위한 도구는 SQL Server 2000의 Query Analyzer를 사용하였다. 성능 평가를 위해 SQL 질의문 3개를 작성하여 각 기법을 수행하였다. Q1은 엘리먼트 id를 주고서 모든 정보를 읽으며, Q2는 중첩 깊이가 깊지 않은 하나의 엘리먼트의 값으로 Title을 읽으며, 또한 Q3은 중첩 깊이가 매우 깊은 하나의 엘리먼트의 값으로 GivenName을 읽는다. 예를 들어, 세 종류의 SQL 질의문을 T-SQL로 정의하면 다음과 같다.

```
Q1 : SELECT * FROM DescT WHERE Version = '1.1'
Q2 : SELECT Title FROM DescT AS d, CreaT AS
     c WHERE d.ID = c.parentID
Q3 : SELECT GivenName FROM DescT AS d, CreaT
     AS c, AddT AS a, InstT AS i
       WHERE a.ID = i.parentID AND c.ID = a.parentID
       AND d.ID = c.parentID
```

① 테이블 조인 수 분석

일반적으로 검색 시에 테이블의 수가 많으면 조인 수가 증가하고, 각 테이블당 중복된 칼럼수가 많으면 조인 수가 감소한다.

〈표 2〉 테이블 조인 수에 따른 성능 비교

질 의 문	HI	HEI	SHI
Q1	1	1	1
Q2	4	2	2
Q3	9	4	4

<표 2>의 실험 결과를 정리하면, 각 기법에서 생성한 테이블의 개수가 테이블 조인 수를 결정하게 되는데, HI에서는 생성한 테이블의 개수가 다른 두 기법보다 많기 때문에 테이블의 조인 수도 늘어나게 된다. Q1 질의에서는 간단한 조인만으로 이들 정보를 구할 수 있으므로 테이블의 조인 수가 모두 같음을 볼 수가 있다. 그리고, HEI와 SHI 에서는 생성한 테이블의 수나 테이블의 조인 수에서 성능 평가 결과가 같음을 알 수가 있다.

② 레코드 접근 수 분석

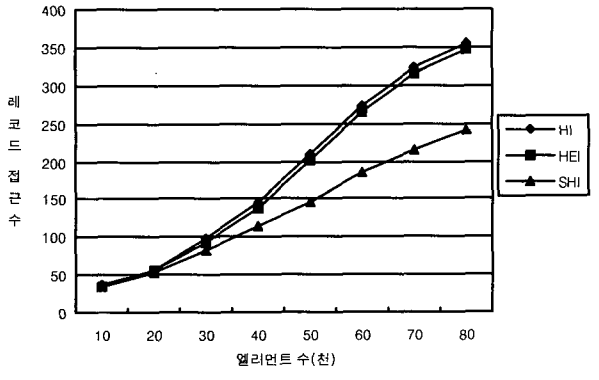
레코드의 접근 수가 많으면 실행 시간이 길어지고, 한개의 XML 인스턴스가 여러 개의 테이블에 나누어져 있는 분할 저장이 심하다는 것이다.

<표 3>의 SHI에서는 XML 인스턴스의 분할 저장을 줄이고, 칼럼의 중복을 활용함으로써 다른 기법에 비하여 레코드의 접근 수가 적음을 알 수가 있다.

〈표 3〉 레코드 접근 수에 따른 성능 비교

질 의 문	HI	HEI	SHI
Q1	56	64	35
Q2	123	136	69
Q3	355	347	241

(그림 9)는 HI, HEI, SHI에 대하여 질의문 Q3으로 엘리먼트 수에 따른 레코드 접근 수를 실험한 결과이다. 즉, 중첩의 깊이가 깊고 다수의 엘리먼트들이 질의에 필요하게 되며 여러 번의 조인 연산이 필요하게 되는 경우에 성능 차이가 매우 커짐을 알 수 있다.



(그림 9) Q3 질의문에 대한 레코드 접근 수의 비교

③ 실행 시간 분석

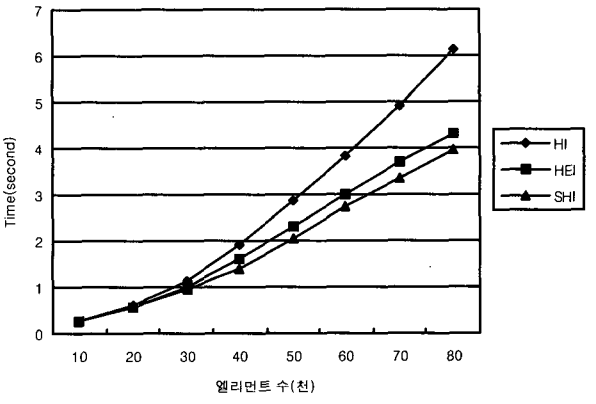
테이블의 조인 수와 레코드의 접근 수가 많으면 실행 시간이 길어진다. 이러한 이유는 대체로 XML 인스턴스의 분할 저장이 심할 경우에 발생한다.

〈표 4〉 실행 시간에 따른 성능 비교(second)

질 의 문	HI	HEI	SHI
Q1	0.248	0.250	0.251
Q2	4.92	3.89	3.62
Q3	6.12	4.297	3.97

<표 4>의 실험 결과를 정리하면, Q1의 경우처럼 검색 시에 킷값을 이용하여 전체 XML 문서에서 데이터 값만을 찾아내는 경우에는 전부 비슷한 성능을 보여주고 있다. Q2의 경우에는 중첩의 깊이가 깊지 않고 where 절에 포함하는 엘리먼트의 수가 적기 때문에 비록 조인 연산이 필요하더라도 HI보다 빠른 성능을 보이고 있다. Q3의 경우에는 여러 번의 복잡한 조인 연산이 필요하게 되므로 테이블의 수가 많은 HI에서 급격히 나빠진 성능을 보이게 된다.

(그림 10)은 HI, EHI, SHI 기법에 대하여 질의문 Q3으로 엘리먼트 수에 따른 실행 시간을 비교 실험한 결과의 그래프이다. SHI는 중첩의 깊이가 깊고 다수의 엘리먼트들이 질의에 필요하게 되며 여러 번의 조인 연산이 필요하게 되는 Q3의 경우에 HEI에 비하여 실행 시간이 빠름을 알 수 있다.



(그림 10) Q3 질의문의 실행 시간에 따른 성능 비교

4.2.3 구조적 검색 지원

관계 경로 요소 정보는 루트 노드로부터 경로에 의해 이름 붙여지며 각각의 요소를 가리키며, 테이블간의 관계 정보와 칼럼으로 사상된 요소 정보와 속성 정보들을 나타내어 효과적인 구조적인 검색이 가능하다. 제안된 SHI 기법의 관계 경로 요소 정보는 보다 효과적인 구조적 검색이 가능하며, SQL로 정형화하기 위한 정보를 제공하여SQL 질의 정형화가 용이하다. 따라서 다른 애플리케이션들이 접근하기 편리하며, XML 질의 언어의 표준인 XQuery 언어를 지원함으로서 범용성을 가진다. 또한 XML 문서의 구조정보를 저장하여 데이터베이스에서 XML 문서로 복원되는 능력인 XML 문서의 복원력도 가능하다. 이러한 구조적 검색 지원 평가는 SQL 질의 정형화, XML 문서의 복원력, 구조적 검색 지원 여부로 하였다.

〈표 5〉 구조적 검색 지원 평가

평가 요소	HI	HEI	SHI
SQL 질의 정형화	어려움	어려움	용이함
XML 문서의 복원력	미약	미약	가능
구조적 검색 지원 여부	불가능	불가능	가능

<표 5>는 구조적 검색 지원 평가이다. SHI에서의 관계 경로 요소 정보를 이용한 구조적 검색 기법은 DTD 독립적인 저장 방식에서 사용하는 기법을 개선한 방법이다. 따라서 HI와 HEI에서는 구조적 검색을 지원하지 못하기 때문에 비교 평가가 불가능하다. 그러나 DTD 독립적인 방식과 비교한 결과 이러한 관계 경로 요소 정보를 이용한 구조적 검색 기법은 DTD 독립적인 저장 방식과 비슷한 질의 처리 성능을 보였다. 또한 SQL 질의 정형화가 용이하고, XML

질의 언어인 XQuery를 통해서 기존의 관계형 데이터를 접근함으로서 기존 저장 방식들보다 좋은 호환성을 보였다.

4.3 평가 검토

제안된 SHI에 대한 성능 평가에서는 HI와 HEI 기법들과의 비교를 통해 생성된 관계형 테이블의 크기를 평가하였고, 성능을 비교 실험하였다. 실험 결과 SHI의 경우에 테이블 크기 측면인 테이블당 평균 칼럼수에서 HEI보다 40% 정도가 많았다. 그럼에도 불구하고 레코드의 접근 수를 약 30%정도 줄일 수 있었고, 약 10% 정도의 실행 시간을 단축할 수 있었다. 또한 테이블 생성시에 휴리스틱 매뉴얼 인라이닝 처리를 함으로서 테이블 수에서 HI보다 약 23%의 총 테이블 수를 줄일 수 있었다.

5. 결론 및 향후 연구

XML 모델링에서 DTD의 단점을 보완하여 표준화된 XML 스키마로부터 관계형 스키마를 생성하는 것이 보다 시급한 일이라 예상된다. 특히 XML 스키마에는 상속 개념이 있어서 이것을 잘 활용하면 재사용성을 높이는 효과를 가져올 수가 있다. 일반적인 XML 모델링에는 XML 문서 구조 갱신에 따른 구조 수정, 검색 효율성, 분할 저장으로 인한 대량의 조인 연산, 원문 복원력등의 문제점을 가지고 있다. 따라서 이 논문에서는 이러한 문제를 해결하기 위하여 전체적인 저장 구조와 검색의 효율성을 높이고, 데이터베이스 시스템의 장점을 최대로 살리는 XML 모델링을 제안하였다.

구현한 시스템을 통하여 실제 MDS Schema 문서로 관계형 테이블을 만들고 성능 비교 실험을 통해서 SHI 기법의 우수함을 보였다. SHI는 테이블 크기 측면인 테이블당 평균 칼럼수에서 HEI보다 40% 정도가 많았음에도 불구하고 레코드의 접근 수를 약 30%정도 향상시켰고, 약 10% 정도의 실행 시간을 단축할 수 있었다. 그리고 HI 기법과 HEI 기법 및 SHI 기법을 상호 비교함으로써, 어떠한 상황에서 어느 기법이 우수한 관계형 스키마를 생성하는지를 확인하였다. 또한 관계 경로 요소 정보를 이용함으로써 구조적 검색이 가능하고 SQL 질의 정형화가 용이해졌다. 따라서 제안한 기법을 적용한 스키마 변환 시스템은 인터넷 상에서 데이터 교환의 매개체로 XML을 이용하는 다양한 분야의 데이터베이스 구축과 정부 표준 문서와 같이 많은 종류와 카테고리로 나누어지는 문서 타입들을 관리하는 분야에 적용 가능하다고 판단된다.

향후 연구과제는 관계형 스키마 생성 시에 XML 스키마의 특징인 복잡한 유도 관계와 다형성을 고려한 변환이 필요하며, XML 스키마로써 객체 지향 스키마를 생성하는 연구가 필요하다.

참 고 문 헌

[1] A. Deutsch, M. F. Fernandez, D. Suciu, "Storing Semi-structured Data with STORED," *In Proc. of ACM SIGMOD Conference*, 1999.

[2] S. Abiteboul, P. Bunneman, D. Suciu, *Data on the Web : From Relations to Semistructured Data and XML*, Morgan Kaufmann, 1999.

[3] Albrecht Schmidt, Martin L. Kersten, Menzo Windhouwer, Florian Was, "Efficient Relational Storage and Retrieval of XML Documents," *WebDB*, 2000.

[4] J. Shanmugasundaram, H. Gang, K. Tufte, C. Zhang, D. DeWitt and J. Naughton, "Relational Databases for Querying XML Documents : Limitations and Opportunities," *Proceedings of the 25th VLDB Conference*, Edinburgh, Scotland, pp.302-314, 1999.

[5] 김정섭, XML Schema로부터 관계형 스키마의 자동 생성, 한국과학기술원 석사학위 논문, 2002.

[6] W3C, *XML Schema Part 0 : Primer*, http://www.w3.org/TR/ xmlschema-0, May, 2001.

[7] W3C, XML Schema Part 1 : Structures, http://www.w3.org/TR/ xmlschema-1, May, 2001.

[8] W3C, *XML Schema Part 2 : Datatypes*, http://www.w3.org/TR/xmlschema-2, May, 2001.

[9] 박성진, "XML 데이터베이스", 한국인터넷정보학회학회지, 제2권 제3호, Sept., 2001.

[10] D. Florescu and D. Kossmann, "Storing and Querying XML Data Using an RDBMS," *IEEE Data Eng. Bulletin*, 1999.

[11] Gerti Kappel, et al., "X-Ray - Towards Integrating XML and Relational Database System," *Proceedings of ER 2000*, pp.339-353, 2000.

[12] R. Bourret, C. Bornhovd, A. P. Buchmann, "A Generic Load/Extract Utility for Data Transfer between XML Documents and Relational Databases," *WECWIS'00*, San Jose, California, Jun., 2000.

[13] IBM, *MPEG-7 Schema Page*, http://pmedia.i2.ibm.com:8000/ mpeg7/schema, Apr., 2002.

[14] Chaiyanya Baru, "XViews : XML Views of Relational Schemes," *International Workshop on Internet Data Management*, 1999.

[15] Jon Duckett, et al., *Professional XML Schema*, Wrox, 2002.

[16] M. Volz, K. Aberer and K.Boem, "Applying a Felexible OODBMS-IRS-Coupling to Structured Document Handling," *In Proceedings of 12th ICDE*, 1996.

[17] Kevin Williams, et al., *Professional XML Databases*, Wrox, 2000.

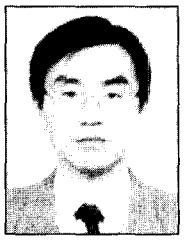
[18] 김재훈, 박석, "XML 데이터의 RDBMS로의 분할 저장시 중복을 활용한 효율적 질의 처리", 정보과학회데이터베이스 연구회논문지, May, 2002.

[19] 노준규, 관계 데이터베이스를 이용한 XML 스키마 저장 및 통합 관리 기법의 설계 및 구현, 경원대학교 석사학위논문, 2002.

[20] 민준기, XML 데이터의 효율적인 경로 인덱스와 구조 정보 추출, 한국과학기술원 박사학위 논문, 2002.

[21] 조정길, "UML 확장 메카니즘을 이용한 XML Schema 사상 명세", 컴퓨터산업교육학회논문지, 제3권 제2호, pp.167-178, 2002.

[22] 조정길, 조윤기, 구연설, "구조적 상이성 분석에 기반한 XML 문서 변환 시스템의 설계 및 구현", 정보처리학회논문지D, 제9-D권 제2호, pp.297-306, 2002.



조 정 길

e-mail : jkcho@sungkyul.edu

1987년 숭실대학교 전자계산학과(공학사)

1993년 숭실대학교 정보과학대학원 (이학석사)

2003년 충북대학교 대학원 전자계산학과 (이학박사)

1987년~1992년 DHL코리아 정보시스템실

1992년~1993년 에스콰이아 정보시스템부

1996년~2004년 남서울대학교 컴퓨터학과 겸임교수

1997년~1999년 한신텔리젠트 연구소 선임연구원

2004년~현재 성결대학교 컴퓨터공학부 전임강사

관심분야 : 객체지향 자료 모델링, XML 문서관리, 시멘틱 웹, 정보검색