산업기술연구(강원대학교 산업기술연구소 논문집), 제21권 A호, 2001.
Journal of Industrial Technology, Kangwon Nat'l Univ.,Korea, No. 21 A, 2001.

DTD를 이용한 XML문서의 효율적인 스키마 추출 기법

An Efficient Schema Extracting Technique Using DTD in XML Documents

안 성 은* 최 황 규**
Ahn, Sung-Eun Choi, Hwang-Kyu

Abstract

XML is fast emerging as the dominant standard to represent and exchange data in the Web. As the amount of data available in the Web has increased dramatically in recent years, the data resides in different forms ranging from semi-structured data to highly structured data in relational database. As semi-structured data will be represented by XML, XML will increase the ability of semi-structured data. In this paper, we propose an idea for extracting schema in XML document using DTD.

키워드 : *XML, DTD, 반구조적 데이터, 카디널리티 연산자*
Keywords : *XML, DTD, Semi-structured data, Cardinality operator*

1. 서론

XML[12]은 W3C에 의해 제안된 SGML의 한 부분집합으로써 HTML과 SGML의 단점을 보안한 웹 상에서 데이터를 교환하기 위한 표준 언어이다.

웹 상에서 다루어지는 데이터의 양이 급속하게 증가하면서 데이터의 형식이 전혀 구조화되지 않은, 즉 각 데이터들간에 절대적인 공통 구조가 미리 고정되어 있지 않고, 불완전하고 불규칙적인 구조 정보를 포함하고 있는 반 구조적 데이터(Semi-structured Data)들이 증가하고 있다[8,9]. XML은 이러한 반 구조적인 데이터들을 정의하는 표준으로 등장하고 있으며 그 유용성을 증가시킬 것이다.

XML은 DTD(Document Type Definition)를 통해서 문서자체에 문서의 구조를 기술하고 있다[12]. 문서의 구조를 사용자가 원하는 대로 정의할 수 있으며, 이러한 구조적 유동성은 다양한 형태의 데이터를 XML로 표현될 수 있게 해준다. 즉, 웹에서 운용되는 데이터가 동일한 형태로 저장, 처리 될 수 있음을 의미한다. 앞으로 웹에서 반 구조적 데이터를 대표할 XML문서들이 많아지면 그 데이터들간의 의미적, 구조적 관계를 설정하는 스키마를 추출하여 그에 따라 데이터를 구조화 시켜 정보로써의 가치를 만들 수 있는 새로운 저장 기법들이 필요하다.

본 논문의 나머지 부분의 구성은 다음과 같다. 2절에서는 반 구조적 데이터와, XML데이터 모델에 대해 설명한다. 3절에서는 관련연구를 살펴보고, 4절에서는 스키마 추출방법을 제안한다. 마지막으로 5장에서 결론을 맺는다.

2. 반 구조적 데이터(semi-structured data)

웹 상에서 이용 가능한 데이터들 중에는 기존 데이터베이스 시스템의 스키마와 같이 정확하고 일관적인 공통구조는 아니지만 각 데이터마다 어떤 일련의 내재하는 구조가 존재하는 데이터를 반 구조적 데이터(semi-structured data)라 한다[8,9]. 즉, 앞에서 언급한 것처럼 각 데이터들간에 절대적인 공통 구조가 미리 고정되어 있지 않고, 불완전

* 강원대학교 컴퓨터정보통신공학과 석사과정
** 강원대학교 전기전자정보통신공학부 교수, 공학박사

하고 불규칙적인 구조 정보를 포함하고 있는 데이터를 말한다.

2.1 반 구조적 데이터의 주요 특성

반 구조적 데이터는 그 이질적인 성격 때문에 비록 데이터자체는 작더라도 스키마가 상당히 커질 수 있다. 기존의 데이터 베이스 시스템에선 스키마와 데이터와의 구별이 명확했다. 즉, 스키마는 데이터베이스의 구조를 묘사하는 역할을 하고 데이터는 데이터 베이스의 한 인스턴스(instance)이다. 하지만 반 구조적 데이터에서는 스키마와 데이터와의 많은 차이점들이 사라진 졌다. 즉, 스키마가 매우 크고, 스키마가 자주 변화하고, 스키마에 대한 제약요소가 사라졌다.

2.2 반 구조적 데이터(semi-structured data)의 예

대표적인 반 구조적인 데이터로는 HTML로 작성된 문서를 예로 들 수 있다. HTML문서는 일반 텍스트 문서와는 달리 태그에 의한 구조정보를 포함한다. 즉, 전체 데이터의 공통 구조가 아니라 각 데이터마다 자신의 데이터 구조만을 표현하고 있다. 기본적으로 HTML문서를 구성하는 html, title, body 태그를 구성하고는 있지만 데이터는 body 태그 내에 사용자의 의도에 따라 여러 다양한 형태의 이질적인 데이터를 태그에 의한 구조정보를 이용해 나타낸다.

반 구조적 데이터는 본질적인 특성상 기존의 관계형 데이터 모델이나 객체 지향형 데이터 모델로 모델링하는 것은 어려우며 대부분의 경우에 있어서 관련 데이터 집합의 공통적인 구조를 미리 결정하는 것이 불가능하다[8,9]. 따라서, 반 구조적인 데이터를 위한 데이터 모델이 필요하며 다음절에서 다루도록 하겠다.

2.3. XML데이터 모델

반 구조적 데이터를 모델링(modeling) 하는데 있어서 가장 근본적인 문제는 어떠한 모델(model)을 선택하느냐 이다. 아주 엄격하고 잘 정리된 모델이나 아니면 간단하고 유연한 모델 중 선택을 해야 할 것이다.

(1) OEM(Object Exchange Model)

반 구조적 데이터 모델로 제안된 것은 스텐포드 대학의 TSIMMIS 프로젝트[6]와 Lore[5]에서 사용된 OEM(Object Exchange Model)[5,7]이 표준적인 모델로 사용되고 있다. OEM은 자기 기술적이고 라벨이 붙은 그래프 형태로 표현 될 수 있다. OEM 데이터 모델에서 모든 존재 물은 객체이고, 각각의 객체는 유일한 OID(Object Identifier)와 그 객체를 기술하는 텍스트 라벨(label)과 타입(type), 값(value)을 가진다. 단일 타입으로는 *integer, real, string, image, html, audio,* 등등이 있을 수 있으며, 복합 타입으로는 OEM 객체들의 집합을 갖는다. OEM은 중첩된 객체와 객체 식별자만을 제공하는 단순한 모델이므로 정보 통합 시 사용되는 여러 가지 다른 데이터 모델로의 변환이나 병합이 간단하지만 객체들 사이에 순서를 표현하지 못하는 단점이 있다.

실제로 많은 반 구조적 데이터를 저장하기 위한 연구에서[1,2,3,4] OEM과 비슷한 모델들이 제안되었다.

(2) Labeled Tree Model

Labeled Tree Model[11]은 화살표(edge)에 레이블(label)이 붙은 트리로 표현되며 트리의 각 노드는 연결자로 데이터 구조를 나타낸다. 레이블에는 데이터 속성, 데이터 값 등의 기본적인 정보를 기술하며 트리의 단말 에지(terminal edge)에 있는 레이블은 데이터 값(data value)이고 비 단말 에지(non-terminal edge)에 있는 레이블은 데이터 속성(data attribute)을 나타낸다. 또한, Labeled Tree Model은 순환구조를 허용하지만 순서를 표현하지 못하는 단점을 가진다. 이 모델의 수정된 모델이 관련연구에서 사용되었으며 본 논문에서도 이와 비슷한 모델을 사용하였다.

3. 관련연구

기존의 데이터베이스에 XML데이터를 저장하기 위해서는 스키마를 추출하는 것이 가장 중요하다. 몇몇 기존의 연구를 통해서 스키마 추출방법에 대해 알아보자.

3.1 관계형 DBMS 저장 및 검색

(1) 데이터모델

[4]의 OEM모델과 유사한 간단한 형태의 Graph Data Model을 사용한 경우로 이 모델 에서는 반 구조적 데이터가 라벨화된 방향성 그래프(directed labeled graph)로 표현된다. 노드는 개체를 나타내며 각각의 개체는 유일한 식별자를 가지며 개체에서 나가는 화살표(edge)은 개체의 에트리뷰트(attribute)를 나타낸다, 즉 에지(edge)들은 에트리

뷰트들의 이름 값으로 라벨화 된다.

그림 3-1의 XML코드는 한 가족의 구성원에 대한 정보이다. 각각의 person은 유일한 식별자(id)와 나이 속성을 가지며 다른 정보는 하위 요소에 나타난다. person 1은 하위요소로 child를 가지지만 person 2는 person 4에 대한 포인터(pointer)만을 가진다.

위의 XML문서를 데이터 모델로 맵핑(mapping)하는데는 몇 가지 문제점이 있다. 첫째로 속성(attribute)과 하위요소(sub-element)사이에 순서의 적절함을 해결 해야 한다. 둘째로 속성과 하위요소

```
<person id=1 age=55>
        <name>Peter</name>
        <address>4711 Fruitdale Ave.</address>
        <child>
                <person id=3 age=22>
                        <name>John</name>
                        <address>5361 Columbia Ave.</address>
                        <hobby>swimming</hobby>
                        <hobby>cycling</hobby>
                </person>
        </child>
        <child>
                <person id=4 age=7>
                        <name>David</name>
                        <address>4711 Fruitdale Ave.</address>
                </person>
        </child>
</person>

<person id=2 age=38 child=4>
        <name>Mary</name>
        <address>4711 Fruitdale Ave.</address>
        <hobby>painting</hobby>
</person>
```

그림 3-1. XML 문서 예

를 구별하는 방법이다. 셋째로 참조 값에 대한 모델링 방법이 있다. 이 논문에서는 XML문서에서의 하위 요소(element)와 속성(attribute)를 구분하지 않았으며 참조 값은 그래프에서 화살표로 표현된다. 위의 XML문서를 OEM과 유사한 Label Tree Model로 나타내면 [그림 3-2]와 같으며 소스의 순서 값을 주기 위해 번호를 주었다.

(2) 스키마 추출방법

XML문서를 기존의 관계 데이터 베이스에 저장하기 위해 스키마를 추출하는 방법에는 여러 가지가 있을 수 있다.

가장 간단하게는 위에서 언급했던 XML문서(가족 구성원의 정보)의 모든 요소(element, attribute, id, pointer, text)들을 하나의 테이블에 저장하는 방법이 있다. 이때 테이블에는 구조정보를 포함하기 순서 값(각 id가 구성하는 그래프 나타나는 노드들의 순서)을 포함한다. 이 논문에서는 이러한 테이블을 edge 테이블이라 칭했다.

두 번째 방법으로는 같은 이름을 가지는 요소들을 하나의 테이블에 함께 저장하는 방식이다. 즉, 가족 구성원의 정보에 관한 XML문서에서 age, name, address, child, hobby같은 요소를 하나의 테이블에 저장하는 방법으로 많은 테이블을 필요로 한다.

세 번째 방법으로는 첫 번째 방법의 target값들을 따로 분리해 정수형 데이터와 문자열데이터의 테이블에 저장하는 방법이다.

마지막으로는 두 번째 방법에서의 value테이블과 edge 테이블을 조인하는 방법으로 이 방법이 가장 성능이 좋았다.

3.2 객체-관계형 DBMS 저장 및 검색

(1) 데이터모델

그림 3-2는 XML문서를 간단한 트리 구조로 나타낸 것이며 트리 구조의 노드 타입을 Element, Attribute, Text의 세 가지 형태로 구분하였다. 물론, 주석(comment)등 기타 다른 노드 타입이 존재할 수 있지만 이 논문에서는 그러한 것들은 고려하지 않았으며 XML문서를 저장하는 데이터 베이스 스키마는 DTD나 Element 타입과는 독립적이다. element노드는 라벨(label)로 이름(name)값을 가지며 자식 노드를 하나이상 가질 수 있으며 없을 수 도 있다. Attribute노드는 라벨로 속성 이름(name)값과 value값을 가지며 자식 노드를 가지지 않는다. 만약 속성 값이 여러 개 일 경우 그 순서를 구분하지 않는다. Text노드는 라벨로 문자열 값을 가진다.

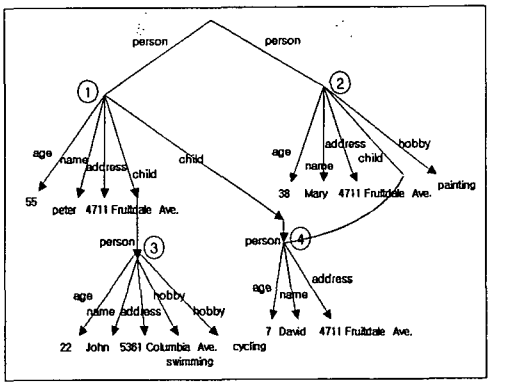

그림 3-2. XML 문서 트리

(2) 스키마 추출방법

XML문서를 저장하는 릴레이션은 Element Attribute, Text, Path 네 가지로 구성된다.

Element 릴레이션은 Element노드들에 대한 정보를 저장하는데 데이터 베이스 속성은 문서식별자(docID), 경로식별자(pathID), 인덱스, 위치정보이다. Attribute 릴레이션은 Attribute노드들에 대한 정보를 저장하는데 데이터 베이스 속성으로는 문서식별자(docID), 경로식별자(pathID), attribute 값, 위치정보이다. Text 릴레이션은 Text노드에 대한 정보를 저장하며 데이터 베이스 속성으로는 문서식별자(docID), 경로식별자(pathID), Text 값, 위치정보를 가진다. 마지막으로 Path 릴레이션은 간단한 경로구조에 관한 정보를 저장하는데 데이터베이스 속성으로는 경로표현(pathexp), 경로식별자(pathID)를 가진다.

트리 구조를 노드로 분해하여 각각의 노드들의 타입에 따라 저장을 하면 다음과 같은 장점을 얻을 수 있다. 즉, XML문서를 저장하는 데이터 베이스 스키마는 DTD와 노드 타입에 독립적이다. 따라서 어떠한 XML 문서도 네게의 릴레이션 테이블을 기반으로 하여 관리 할 수 있다.

4. DTD를 이용한 스키마 추출 기법

XML문서는 XML데이터들의 스키마 역할을 하는 DTD를 가지고 있다. 물론 DTD가 강제적인 요소는 아니지만 도서목록이나 은행의 고객관리정보와 같은 데이터를 XML문서로 만들 때에는 문서의 구조에 관한 정보를 가지고 있는 DTD가 반드시 필요할 것이다. 우리는 XML문서의 구조 정보를 담고 있는 DTD (Document Type Definition)를 이용하여 XML문서의 스키마를 추출하고자 한다

4.1 기본 아이디어(basic idea)

XML문서의 DTD(Document Type Definition)의 요소(element)선언부 에는 각 노드와 자식노드의 수를 결정해주는 카디널리티 연산자(Cardinality Operator)를 가지고 있다[12]. 본 논문에서는 Cardinality Operator를 CO 라고 약칭하겠다. CO를 이용해 각 노드간의 개수 범위를 설정하여 DTD 그래프에 적용함으로써 스키마를 추출 하고자 한다.

기호	노드의 개수 범위	예
(none)	(1)	ELEMENT A
?	(0,1)	ELEMENT A?
*	(0,1,2,3...)	ELEMENT A*
+	(1,2,3,4...)	ELEMENT A+

그림 4-1. 카디널리티 연산자

4.2. CO(Cardinality Operator)

그림 4-1에서 (none)은 노드가 단 한 개만 존재하고, ?는 있거나 없거나 둘 중 한가지 경우이다. *는 없거나 여러 개 존재하고, +는 한 개 이상 여러 개 존재한다는 의미이다.

CO를 부모노드에서 자식노드까지 적용시키기 위해서 그림 4-2와 같은 아이디어를 냈다. 즉, 그림 4-2와 같이 부모와 자식노드의 '합 연산자'를 표현 할 수 있다. 각 연산자의 합 연산을 표시한

(none)을 n으로표기	
nn →n (1)	?+ →* (0,1,2,3,...)
n? →? (0,1)	** →* (0,1,2,3,...)
n* →* (0,1,2,3,...)	*n →* (0,1,2,3,...)
n+ →+ (1,2,3,4,...)	*+ →* (0,1,2,3,...)
?? →? (0,1)	++ →+ (1,2,3,4,...)
?n →? (0,1)	+? →* (0,1,2,3,...)
?* →* (0,1,2,3,...)	

그림 4-2. 카디널리티 연산자 조정

것으로 괄호 안의 숫자들은 노드의 발생 범위를 나타낸다. 예를 들어 다음과 같은 DTD가 있다.

```
<!ELEMENT dblab (professor+,student+,project*) >
<!ELEMENT professor (name,office,phone*) >
```

여기서 professor/phone에 대한 CO는 +* → * 이다. 즉, professor/phone은 (0,1,2,3,...)의 노드 개수 범위를 갖는다. 위와 같은 방법으로 루트노드에서 어떤 위치에 있는 자식노드까지의 노드 개수 범위를 n, ?, *, +로 나타낼 수 있다.

4.3 절대 스키마 그래프

그림4-3은 본 논문에서 사용할 Book Catalog.dtd 이다. 위의 DTD에 OEM과 유사한 directed labeled graph를 그린 후에 연산자를 적용하면 그림4-4와 같다. 그림4-4에 합 연산자를 적용하여 그래프를 그리면 그림4-5 와 같으며 각각의 CO는 XML문서의 노드 범위를 나타낸다. 즉, +로 표시된 노드는 문서 내에서 1번 이상 나타나고, * 로 표시된 노드는 0번 이상 나타날 것이다. 즉, 간단하게 그래프만으로 전체 문서에서 반드시 존재하는 엘리먼트를 확인 할 수가 있고, 각 엘리먼트의 발생 빈도를 확인 할 수 있다. 즉, 그림4-5에서와 같이 "+"와 "n"연산자를 가지는 노드들을 따로 분리해서 그래프를 그리면 그림4-6과 같다. 이러한 그래프를 "절대 스키마 그래프"라 칭하겠다. 절대 스키마는 전체 XML문서들에서 반드시 존재하는 노드들만으로 구성된 것으로, 절대 스키마를 바탕으로 관계형 데이터베이스에 저장하기 위한 릴레리션 스키마(relation schema)를 추출한다.

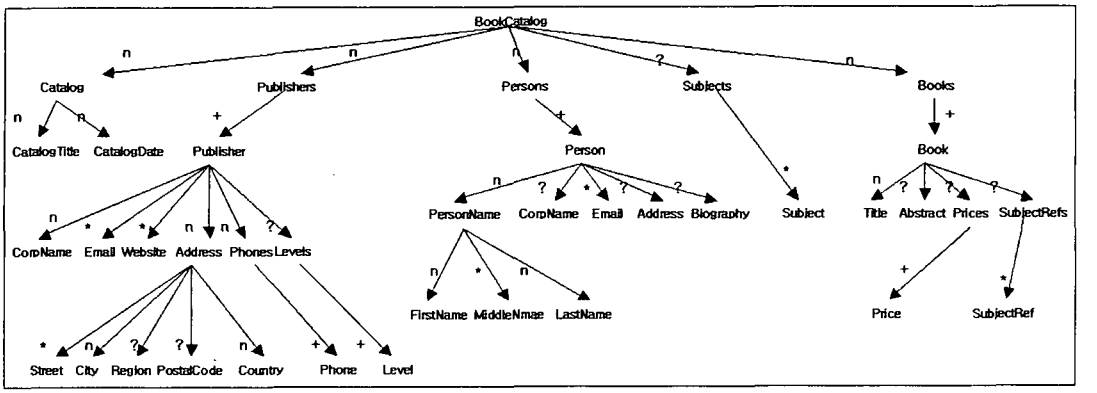

그림 4-4. Book Catalog 문서에 대한 Directed Labeled Graph

```
<!ELEMENT BookCatalog (Catalog, Publishers, Persons, Subjects?, Books) >
<!ELEMENT Catalog (CatalogTitle, CatalogDate) >
<!ELEMENT CatalogTitle (#PCDATA) >
<!ELEMENT CatalogDate (#PCDATA) >
<!ELEMENT Publishers (Publisher+) >
<!ELEMENT Publisher (CorpName, Email*, Website*, Address, Phones, Levels?) >
<!ELEMENT Address (Street*, City, Region?, PostalCode?, Country) >
<!ELEMENT Street (#PCDATA) >
<!ELEMENT City (#PCDATA) >
<!ELEMENT Region (#PCDATA) >
<!ELEMENT PostalCode (#PCDATA) >
<!ELEMENT Country (#PCDATA) >
<!ELEMENT Phones (phone+) >
<!ELEMENT Phone (#PCDATA) >
<!ELEMENT Levels (Level+) >
<!ELEMENT Level (#PCDATA) >
<!ELEMENT Persons (Person+) >
<!ELEMENT Person (PersonName, CorpName?, Email*, Address?, Biography?) >
<!ELEMENT PersonName (FirstName, MiddleName*, LastName) >
<!ELEMENT FirstName (#PCDATA) >
<!ELEMENT MiddleName (#PCDATA) >
<!ELEMENT LastName (#PCDATA) >
<!ELEMENT Subjects (Subject+) >
<!ELEMENT Subject (#PCDATA) >
<!ELEMENT Books (Book+) >
<!ELEMENT Book (Title, Abstract?, Prices?, SubjectRefs?) >
<!ELEMENT Title (#PCDATA) >
<!ELEMENT Abstract (#PCDATA) >
<!ELEMENT Prices (Price+) >
<!ELEMENT Price (#PCDATA) >
<!ELEMENT SubjectRefs (SubjectRef+ >
<!ELEMENT SubjectRef (#PCDATA) >
```

그림 4-3. Book Catalog XML 문서 예

4.4 릴레이션 스키마(relation schmea)

Book Catalog.dtd를 사용하는 XML문서들은 절대 스키마 그래프에서 그 구조를 파악 할 수 있듯이 Catalog, Publishers, Persons, Books 그룹으로 나누어 볼 수 있으며, 각각의 그룹을 하나의 테이블로 맵핑(mapping) 할 수가 있다. 맵핑 방법은 기존에 연구되었던 방법들이 사용 될 수 있다. [1,4]

Catalog릴레이션
Catalog(CatalogTitle, CatalogDate)

Publishers 릴레이션
Publishers(Publisher_id, CorpName, Address.City, Address.Country, Phones.Phone)

Persons릴레이션
Persons(Person_id, Person.Firstname, Person.Last Name)

Books릴레이션
Books(Book_id, Title)

기존의 방법들은 주로 문서 중심의 접근 방법으로써 하나의 문서에만 적용 가능한 스키마를 추출하는 기법들이 였다. 하지만 본고에서의 위와 같은 릴레이션 스키마는 동일한 DTD를 사용하는 XML 문서들의 동일한 요소들을 나타내는 스키마이다.

5. 결 론

본 논문에서는 DTD를 이용해 절대 스키마를 추출함으로써 동일한 DTD를 사용하는 XML문서들에 공통적으로 반드시 존재하는 노들을 생성하였고, 절대 스키마를 이용해 XML문서를 관계형 데이터 베이스 시스템에 저장하기 위한 릴레이션 스키마를 생성하였다. 본 고에서는 XML문서의 DTD의 속성(attribute)값은 고려하지 않았으며, 향후 DTD의 속성 선언부에 IDREF, IDREFS, #REQUIRED, #IMPLIED등의 타입 값을 고려할 예정이다.

감사의 글

이 논문은 정보통신부의 정보통신 우수대학원 지원사업의 일부 지원으로 이루어졌음

참고문헌

[1] J. Shanmugasundaram, H. Gang, Kristin Tufte, C. Zhang, D. J. DeWitt, and J. F. Naughton, "Relational databases for querying XML documents : Limitation sand opportunities", In VLDB'99 Edinburgh,

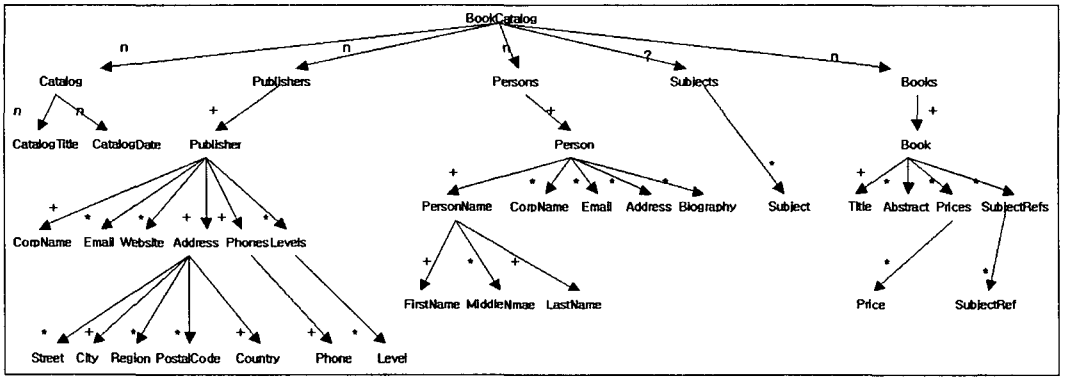

그림 4-5. 카디널리티 연산자의 적용 예

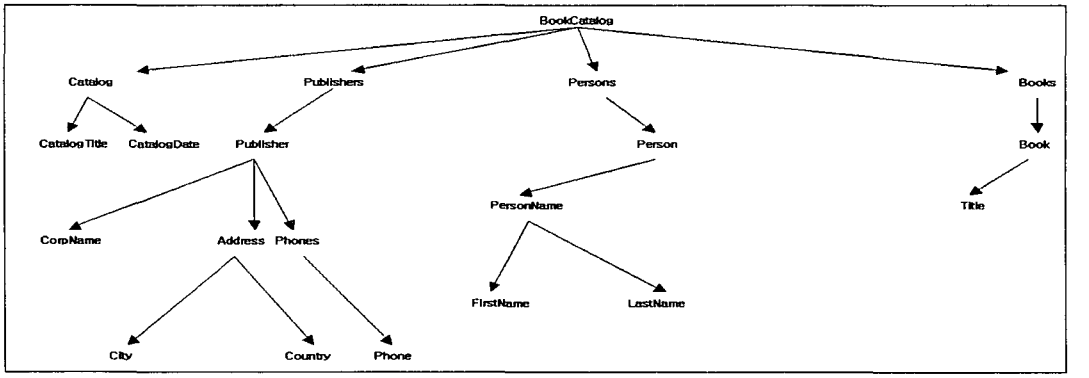

그림 4-6. 결과 그래프

Scotland, 1999.

[2] T. Shimura, M. Yoshikawa, S. Uemura, "Storage and Retrieval of XML Documents Using Object-Relational Databases", DEXA 1999.

[3] A. Deutsch, M. F. Fernandez, D. Suciu, "Storing Semistructured Data with STORED", SIGMOD Conference 1999.

[4] D. Florescu and D. Kossmann, "Storing and Querying XML Data Using an RDBMS", IEEE Data Eng.

[5] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, J. Widom. "Lore: A database management system for semi structured data", Technical report, Stanford University Database Group, February 1997.

[6] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom, "The TSIMMIS Project: Integration of Heterogeneous Information Sources", In Proc. of IPSJ Conference, pages 7-18, 1994.

[7] S. Abiteboul, D. Quass, J. McHugh, J. Widom, J. Wiener, "The Lorel Query Language for Semistructured Data" International Journal on Digital Libraries, 1996.

[8] S. Abiteboul, "Querying Semi-Structured Data", In Proceedings of ICDT, Jan 1997.

[9] P. Buneman, "Semistructured Data : a tutorial", In Proceedings of PODS, Tucson, Arizona, May 1997.

[10] J. Widom. "Data Management for XML : Research Directions", In IEEE Data Engineering Bulletin, volume 22(3), 1999.

[11]P. Buneman, S. Davidson, G. Hillebrand, D. Suciu, "A query language and optimization technique for unstructured data", In proceedings of ACM-SIGMOD International Conference on Management of Data, 1996.

[12] T. Bray, J. Paoli, C. M. Sperberg-McQueen, "Extensible Markup Language(XML) 1.0 (Second Edition)" , http://www.w3.org/TR/REC-xml, W3C Recommendation 6, October 2000.