연역 객체 지향 데이터베이스 언어 구현을 통한 XML 데이터 처리에 관한 연구

김 성 규†

요 약

본 논문에서는 XML 데이터와 같은 비구조적인 데이터 처리와 추론을 필요로 하는 의미 웹(semantic web) 구축에 유리한 연역 객체 지향 데이터베이스(Deductive and Object-oriented Database) 언어 구현을 통해 XML 데이터 처리에 대해 알아본다. 대량 문서 관리와 데이터 교환에 가장 유용한 마크업 언어로 알려진 XML을 이용하여 XML 데이터 모델을 연역객체지향 데이터베이스 모델로 바꾸는 방법에 대해 알아본 다음 이 연역객체 지향 데이터베이스를 다시 Connection Graph로 바꾸고 Connection Graph Resolution을 이용하여 어떻게 질의에 답할 수 있는지를 기술한다. 또한 데이터베이스 내의 계층 지식을 이용하여 효율적이면서도 같은 답을 주는 질의로 바꾸는 방법을 제시하고 이 방법이 효율적이며 논리적으로 타당하다는 점을 증명한다.

On XML Data Processing through Implementing A Deductive and Object-oriented Database Language

Seonggyu Kim†

ABSTRACT

With the advent of XML and database languages armed with the object-oriented concept and deductive logic, the problem of efficient query processing for them has become a major issue. We describe a way of processing semi-structured XML data through an implementation of a Deductive and Object-oriented Database (DOODB) language with the explanation of query processing. We have shown how to convert an XML data model to a DOODB data model. We have then presented an efficient query processing method based on Connection Graph Resolution. We also present a knowledge-based query processing method that uses the homomorphism of objects in the database and the associative rule of substitutions.

키워드 : XML 데이터 처리, 연역 객체 지향 데이터베이스, 질의 처리, Connection Graph Resolution

1. 서 론

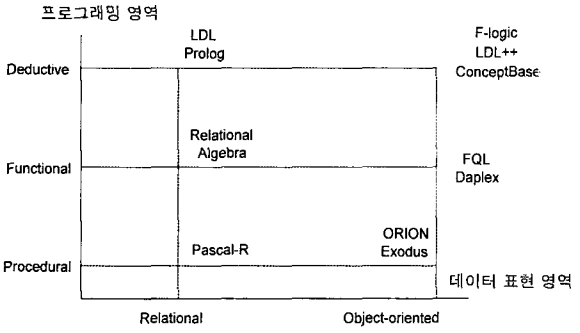
고도 정보화 사회를 주도할 국가 기반구조로서 새로운 사회 간접 자원인 다량의 다양한 데이터베이스 구축이 필요하다. 종래의 관계 데이터베이스는 학적 관리 시스템, 비행기 예약 시스템 등의 레코드 중심의 응용분야에서 많이 활용되어 왔으나, 컴퓨터 하드웨어와 소프트웨어 기술의 발달과 더불어 VLSI CAD나 CASE 도구의 설계, 스케줄링(scheduling), 멀티미디어 데이터베이스 등 새로운 데이터베이스 응용분야의 도래로 관계 데이터베이스의 한계는 이미 여러 문헌에서 고찰되고 있다[1, 3, 10, 12].

자료처리의 핵심은 어떤 형태의 자료를 표현할 수 있는가이다. 이제까지의 관계 데이터베이스 언어가 앞에서 언급한 새로운 데이터베이스 응용분야의 발달로 그 적용 한계에 이르러 새로운 표현 언어에 대한 연구가 활발하게 이루어지고 있다. 이 새로운 응용분야의 특징은 자료의 구조가 복잡하다는 것인데 이는 객체(Object)를 이용하여 해결할 수 있으며 이 객체 지향 개념이 미래 자료처리 언어의 기본이 된다[10, 12]. 또 다른 특징은 자료의 양이 방대하며 추론을 필요로 한다는 것인데 이는 긍정적인(positive) 데이터와 부정적인(negative) 데이터의 표현과 if then else의 rule 표현을 통하여 많은 정보를 효과적으로 다룰 수 있게 해줄 수 있는 논리(logic)를 이용한 추론 기능을 통해 해결할 수 있다. 불완전한 자료의 표현을 rule을 통해 나타낼 수 있으며 확률적인 자료의 표현도 이 rule을 확장하여 나타낼 수 있다.

(그림 1)의 데이터베이스 프로그래밍 언어 발달사에서 보

※ 이 논문은 1997년도 안양대학교 학술연구의 지원을 받아 연구되었음.
† 정 회 원 : 안양대학교 컴퓨터학과 교수
논문접수 : 2002년 9월 30일, 심사완료 : 2002년 12월 16일

듯 프로그래밍 영역의 논리(logic)와 데이터 표현 영역의 객체가 결합된 연역 객체 지향 데이터베이스 언어의 탄생이 필요하게 되었다. F-logic, LDL++ 그리고 ConceptBase와 같은 언어는 기존의 데이터베이스에서 볼 수 있는 여러 계층의 언어, 즉 데이터 언어(Data Language), 데이터 정의 언어(Data Definition Language), 데이터 조작 언어(Data Manipulation Language), SQL과 같은 질의 언어(Query Language), 응용 프로그램 언어(Application Language)를 가짐으로써 생기는 개념적 불일치(Impedance Mismatch)를 해소할 수 있는, 이들을 통폐합한 언어이다[3, 12].



(그림 1) 데이터베이스 프로그래밍 언어의 발달과정

SGML과 웹과의 만남으로 탄생한 XML은 대량 문서 관리와 데이터 교환에 가장 유용한 마크업 언어로 알려져 XML을 기반으로 하는 프로그래밍 기술이 발전함에 따라 인터넷, 전자상거래, 음악, 과학, 디지털 도서관, EDI, EC/CALS, 문서관리 그리고 웹 퍼블리싱 분야 등과 같은 매우 다양한 분야에서 새로운 응용에 XML을 적용하고 있으며 XML은 차세대 웹상의 표준문서로 자리 매김을 하고 있다. 더욱이 현재는 이러한 XML을 기반으로 웹이 단순한 정보의 전달 수단이 아닌 데이터베이스의 개념으로 생각하여 각 정보간의 관계를 명시함으로써 웹상의 문서 자체뿐만 아니라 그 문서의 의미도 추론 또는 이해하여 유익한 정보를 제공할 수 있는 의미 웹(semantic web)에 대한 개발이 진행 중이다[2, 8, 11].

이러한 XML과 객체 지향 개념과 연역 기능을 갖는 데이터베이스 언어의 등장으로 비구조적 데이터를 지원하는 이 들을 위한 효율적인 질의 처리기법에 관한 연구가 주된 관심사가 되었다. XML의 경우 XML Query 워킹 그룹을 통해 웹상에서 XML 문서에 대한 추출, 검색 및 가상 문서의 제공 등의 기능을 제공하는 질의어에 대한 권고안을 제정하고 있다. 이러한 질의 언어는 SGML에서는 존재하지 않으며 데이터베이스 질의 언어인 SQL, OQL과 같은 형식을 참조하고 XML 또는 비구조적 데이터 문서의 질의 언어로 제안된 Quilt, UnQL, XQL 그리고 YATL을 참조하여 설계 중이며[1, 2, 8, 14] XML 데이터를 관계 데이터베이스로 변환하여 단순한 질의를 처리하는 기법에 대한 연구도 진행중이다[7].

본 논문에서는 비구조적 XML 데이터의 관리 및 효율적인 질의 처리 그리고 추론을 필요로 하는 의미 웹의 개발에 사용할 수 있도록 XML 문서를 F-logic 데이터베이스로 바꾸어 질의를 처리한다. XML 문장을 F-logic 언어로 쓰인 데이터베이스로 바꾸고 F-logic 언어의 구문과 의미를 이용하여 계층 지식에 근거한 질의 처리기를 제시하고 또한 F-logic 언어로 작성된 데이터베이스는 단순히 구문만 고려하였던 기존의 관계 데이터베이스와는 전혀 다르기에 효율적으로 질의 처리를 위한 기법을 제시함으로써 연역 객체 지향 데이터베이스 언어의 구현을 통한 XML 데이터 처리에 대해 알아본다.

2. 배경 연구

2.1 F-logic 설명 및 질의처리방안

LDL++(Logic-based Data Language++)는 논리(logic) 프로그래밍과 관계 데이터베이스 기술을 결합한 LDL에 o-expansion 처리과정에 의해 객체 지향 개념을 첨가하여 만들어진 언어로 higher-order를 사용하는 F-logic에 비해 이해하기 어렵다는 단점이 있으며 ConceptBase는 다수 사용자를 위한 연역 객체 지향 데이터베이스로 이론적 완성도가 F-logic에 비해 떨어진다[3, 4, 12]. F-logic은 이론적으로는 성숙 단계에 접어들었지만 많은 기능을 포함하고 있어 실용성이 문제가 되고 있다. 본 논문에서는 비록 실용성은 떨어지지만 이론적 완성도가 높은 F-logic의 일부 기능을 이용하여 XML 데이터 처리를 위한 연역 객체 지향 데이터베이스 언어 구현에 대해 알아본다.

F-logic 언어는 알파벳 심벌(symbol)로 구성된 식(formula)으로 이루어지며 하나의 식은 연결자와 한정자로 연결된 데이터베이스 항목(database term)으로 구성된다. 데이터베이스 항목 T는 is-a 항목, Q : P 또는 데이터 항목 $P[A_1 \rightarrow T_1, \cdots, A_m \rightarrow T_m]$이다. 여기서 P와 Q는 ID 항목으로 객체를 나타내고 A_i는 ID 항목으로 객체의 속성을 나타내며 T_i는 데이터 항목이다. Is-a 항목은 객체 사이의 계층구조를 정의하며 데이터 항목은 객체에 속성 A_i에 정의된 값을 설정해준다. F-logic 언어에서 각 객체와 클래스 객체를 구분하여 나타내지는 않는다. 즉 객체 p가 p : q에서와 같이 객체 인스턴스(instance) 위치에 그리고 r : p에서와 같이 클래스 위치에 나타날 수 있다. 문법적으로는 맞지만 a[X → b]에서와 같이 변수(nonground) 레이블의 사용은 불가능하다. 그리고 클래스가 지정되지 않으면 가장 커다란 객체인 T가 생략된 것으로 간주한다. 또한 모든 데이터베이스 항목은 분자식(molecular formula)이다. 예를 들어 데이터베이스 항목 X[attr 1 → a, attr 2 → b]은 두 개의 원자(atomic literal) X[attr 1 → a]와 X[attr2 → b]로 이루어진 것을 의미한다. 본 논문의 예에서 편의를 위해 소문자로 시작하는 ID 항목은 상수인 경우에, 대문자로 시작하는 ID 항목은 변수인 경우에 사용한다.

[EXAMPLE 1] Formulas

1. student : person
2. "CS" : string
3. X : NBA-Leads ⇐ X [ppg → (> 20)]
4. mary[name → "Mary," employed → cs : dept[name → "cs," manager → phil]]
5. bob[name → "Bob," employed → cs : dept]

Is-a 항목인 student : person는 student는 person임을 의미하며 "CS" : string은 "CS"가 string임을 의미한다. Formula 3은 게임당 득점이 20점 이상인 사람은 NBA-Leads 클래스의 일원이 됨을 의미하며 formula 4는 mary가 cs 학과에서 일하는데 그 곳의 매니저는 phil이라는 사실을 나타내며 formula 5는 bob은 cs 학과에서 일을 한다는 사실을 의미한다. 만약 formula 4와 formula 5가 같은 데이터베이스 내에 저장된 내용이라면 bob의 매니저도 phil이 된다.

F logic언어로 나타낸 연역 객체 지향 데이터베이스는 GCI, EDB^0, IDB^0, 그리고 OHG의 합집합이다. EDB^0와 IDB^0는 연역 데이터베이스(Deductive Database)의 EDB(Extensional Database)와 IDB (Intensional Database)에 각각 해당되는데 연역 데이터베이스와는 달리 객체 식별자(Object Identifier)를 갖는다는 점이 다르다. OHG(Object Hierarchy Graph)는 객체 간의 *is-a* 관계를 나타내며 GCI(General Class Information)는 데이터베이스의 무결성 제약처럼 데이터베이스가 바른 상태에 있는지를 점검하는 정보이다.

2.2 XML(eXtensible Markup Language)

현재 HTML 기반의 웹 문서는 오직 디스플레이 위주로 재가공하여 사용할 수 없으며 의미정보를 표현하지 못해 인터넷상에서 의미 정보를 검색하는데 큰 문제점을 갖고 있다. XML은 HTML과 SGML의 장점을 가지면서 특히 인터넷상에서 의미정보를 전달하고 질의할 수 있는 막강한 장점을 갖고 있다. XML은 사용자가 필요로 하는 XML 정보를 일반 관계 데이터베이스에 XML 형태로 저장하고 XML을 이용하여 효율적으로 검색할 수 있게 해준다. 예를 들어 현재와 같은 EDI 전자상거래 환경에서는 서로 다른 데이터 구조(스키마)를 지니는 데이터베이스간의 데이터 교환이 필수적으로 발생하게 되는데 이때 XML은 각기 다른 데이터베이스에 자신의 데이터를 XML 데이터로 변환 저장하고, 저장된 XML 데이터를 검색할 수 있게 해준다.

XML은 인터넷상에서 데이터 교환을 위해 W3C(World Wide Web Consortium)에서 제안된 표준 언어로 XML은 본질적으로 다른 언어를 기술하기 위한 언어, 즉 메타언어이다. HTML은 태그의 종류가 한정되어 있는 반면 XML은 다른 사용자가 한 사용자가 작성한 태그들의 의미를 파악할 수 있도록, 또는 XML 문서가 태그 정의를 참조할 수 있도록 하는 선언 파일인 DTD(Document Type Definition)를 통해 문서의 내용 또는 데이터에 관련된 독자적인 태그를 사용자가 직접 정의할 수 있으며 그 태그를 다른 사람들이 사용하도록 할 수 있다[8, 13-15].

DTD를 이용하여 문서의 논리적 구조를 다양한 형식으로 표현이 가능하다. 하나의 문서로 각각의 목적에 맞게 스타일 시트를 적용시켜서 정보를 재가공할 수도 있다. XML의 DTD를 통한 구조적 유동성은 모든 형태의 데이터가 XML로 기술될 수 있게 해주며 웹에서 운용되는 모든 데이터가 동일한 형태로 통합, 저장, 처리될 수 있는 기반을 제공한다. XML은 또한 객체 지향 개념을 근간으로 하고 있을 뿐만 아니라 플랫폼과 응용 프로그램에 독립적인 데이터 형식이라는 장점을 갖고 있다.

XML 데이터는 논리적 구조를 가지는 데이터라고 할 수 있으나 이런 논리적 데이터가 어떻게 쓰일지는 정의되어 있지 않다. 단지 문서의 측면에서 브라우징을 할 경우 XSL을 이용해 스타일 정보를 정의할 뿐이다. 데이터 측면에서 원하는 데로 처리하기 위해서 XML을 다루기 위한 시스템은 XML 처리기와 응용 프로그램으로 구성된다. XML 처리기(processor)는 우선 XML 파일이 사양(Specification)을 지키는지 검사한다. 그런 다음 컴퓨터가 XML 파일을 해석하는데 필요한 문서 트리를 생성해 낸다. 즉, XML 처리기는 문서와 DTD를 읽어 문서의 정확성을 검증하고, 문서 내의 엘리먼트, 속성, 컨텐츠 트리, 엔터티 정보를 생성해 내며, 이를 이용하여 XML 응용은 해당 정보에 대한 특정 동작을 기술함으로써 XML 문서의 처리를 수행하도록 지원하는 기능을 제공한다. 이 문서 트리를 이용해서 컴퓨터는 처리 지시들을 차례로 뽑아낸다. 파서는 XML 처리기의 역할을 담당하며, 응용 프로그램은 파서가 뽑아낸 처리 지시에 따라 트리의 데이터를 처리한다.

XML 처리기를 이용하는 방법에는 XML 브라우저를 이용하여 문서를 파싱, 해석하여 XSL이나 CSS와 같은 스타일 시트 정보를 이용하여 표시하는 프로그래밍이 필요없는 방법과 XML 프로세서가 제공하는 API를 이용하여 XML 문서에 대한 검증, 갱신, 수정 등의 처리 작업을 할 수 있도록 프로그래밍 하는 방법으로 나누어진다. 전자의 방법은 XML 브라우저의 기능에 전적으로 의존하는 방법으로 다양한 XML 응용에 모두 적용하기에는 융통성이 적다. 반면 후자의 방법은 다양한 XML 응용에 따라 필요한 기능을 이용하여 XML 문서 정보를 다룰 수 있다.

XML 응용 개발을 위한 표준 API로 이벤트 기반의 SAX (Simple API for XML)와 객체 모델 기반의 DOM(Document Object Model)이 있는데 SAX 방법은 이벤트에 접근한 문서내에 특정 엘리먼트를 만났을 때 지정된 이벤트를 발생시켜 이를 이용하여 필요한 정보에 접근하는 방법이며 DOM 방법은 XML 문서를 파싱하여 문서의 구조정보와 컨텐츠 모두를 객체로서 메모리에 올려놓고 문서 전체에 대한 구조 정

보를 트리에 기반한 객체로 이용 가능하여 XML 문서를 구조적으로 변경하거나 검색하는 작업등에 적합하나 크기가 큰 XML 문서의 사용은 그에 따른 메모리 사용량이 증가한다는 단점이 있다. DOM을 이용하여 사용자는 문서내의 이벤트 및 문서 구조, 문서 내용, 문서의 보안 레벨 설정 등과 같은 문서의 모든 것에 대해 정의, 조작, 변경, 접근할 수 있다.

3 XML에서 F-logic으로의 전환

XML의 계층적인 구조를 갖는 특성 살리면서 문서(Document), 엘리먼트(Element), 애트리뷰트(Attribute) 단위의 조작을 통한 데이터 처리를 수행하기 위해서는 XML 데이터를 프로그램에서 쉽게 처리할 수 있게끔 자료구조화 하여야 하는데, 이는 객체 지향적인 자료구조로 만드는 것인 가장 알맞다. W3C 같은 곳에서도 이 자료구조를 DOM이라는 이름으로 하여 객체로 모델링을 하였다는 것으로도 확인할 수 있다.

XML 문서는 구조화된 문서이기 때문에 DOM 처리가 가능하다. DOM 처리를 위해 XML 데이터 모델은 계층구조를 가지게 되는데 XML의 최상위 노드는 문서(#document)가 되며 자식요소로 처리 명령(processing instruction) 노드와 문서형(Document type) 노드 그리고 주석(#comment) 노드를 가진다. 최상위 노드는 자식요소 이외에 속성들의 노드 목록인 애트리뷰트를 가진다. 다음으로 실제 문서의 내용을 이루는 요소인 엘리먼트를 가지는데 엘리먼트는 재귀적으로 구성할 수 있으며 속성을 가질 수 있는 부 엘리먼트로 구성된다. 엘리먼트와 부 엘리먼트 사이의 관계는 엘리먼트의 ID나 IDREF 속성으로 정의될 수 있다.

엘리먼트의 선언은 <!ELEMENT elem_name(strut_spec) >으로 주어지는데 struct_spec은 엘리먼트의 구조를 기술하며 엘리먼트의 속성은 다음과 같이 기술된다.

```
<!ATTLIST elem_name attr_name 1 attr_type 1 attr_constr 1
                                ...
                        attr_namen attr_typen attr_constrn>
```

이 때 attr_typei는 값 속성과 참조 속성을 구분하거나 ID 속성을 기술하거나 속성의 값이 스칼라(scalar)인지 여러 개의 값을 가질 수 있는 속성인지를 지정하며, 참조는 ID나 IDREF 속성에 의해 가능하다. Attr_constri는 속성이 반드시 필요한지 아니면 아닌지, 또는 단순 상속(monotonic inheritance)이 가능한지를 나타낸다.

이러한 사실을 바탕으로 XML에서 F-logic 언어로의 변환은 일반적으로 엘리먼트가 위에서 아래로 같은 레벨에서는 왼쪽에서 오른쪽으로 나타나는 상대적인 순서에 따라 element는 F-logic의 클래스나 단순 객체로, attribute는 객체의 속성으로 ID, IDREF 유형의 인스턴스는 객체인식자로, CDATA, NMTOKEN의 인스턴스는 값으로 각각 변환된다. F-logic으로 표현하기 위해 <oid> 태그와 ISO 통화 단위에서 사용되는 콜론(:)이 첨가되어 사용되는데 <oid> 태그는 XML 데이터를 유일하게 구분하기 위한 키 역할을 하며, 콜론은 클래스를 표시하기 위해 사용되어 phil : employee는 phil이 employee 클래스의 일원임을 의미한다.

[정의 1] Transformation

E를 Element, A를 Attribute 그리고 D를 각 attribute 유형에 따른 데이터 값이라 하면 XML 문서 T = <E, A, D>로 정의된다. 이때 T에서 F-logic으로의 변환은 다음의 규칙을 따른다.

1. E_i가 E_j내에 정의되어 있으면 $E_i : E_j$
2. A_l가 E_m의 Attribute로 정의되어 있고 인스턴스의 값이 D_n이면 $E_m[A_l \rightarrow D_n]$.

일반적으로 DTD의 내용은 F-logic의 OHG 부분을 기술하며 XML 인스턴스는 F-logic의 식을 구성한다. DTD 혹은 XML 스키마 정보가 없는 경우에도 XML 인스턴스로부터 p : q와 같이 주어지는 F-logic의 OHG를 구성할 수 있으며 DTD가 있을 때와 다른 점은 q가 항상 모든 객체를 포함하는 최상위 객체 ⊤가 된다는 점이다. 다음은 본 논문의 설명을 위한 샘플 DTD와 XML 인스턴스(instance)이다.

```
<!ELEMENT school (person+, dept+)>
<!ELEMENT person (employee+, faculty+, student+)>
<!ELEMENT person name #PCDATA>
<!ATTLIST person oid ID #REQUIRED>
<!ATTLIST employee works IDREF #REQUIRED>
<!ATTLIST faculty works IDREF #REQUIRED>
<!ELEMENT student EMPTY>
<!ATTLIST student major IDREF #REQUIRED>
<!ELEMENT dept (dname)>
<!ELEMENT dept dname ID #REQUIRED>
<!ATTLIST dept name #PCDATA mgr IDREF #REQUIRED>

<school>
<person>
<faculty name = "Chris"><oid> chris </oid>
                        <works> ee </works></faculty>
<faculty name = "Bob"><oid> bob </oid>
                        <works> cs 1 </works></faculty>
<faculty name = "Mary"><oid> mary </oid>
                        <works> cs 2 </works></faculty>
<employee name = "John"><oid> john </oid>
                        <works> cs 2 </works></employee>
<student name = "Sally"><oid> sally </oid>
                        <major> cs </major></student>
</person>
<dept name = "CS"><dname> cs 1</dname>
                    <mgr> phi l : employee </mgr></dept>
<dept name = "CS"><dname> cs 2 </dname>
                    <mgr> mary </mgr></dept>
<dept name = "EE"><dname> ee </dname>
                    <mgr> chris </mgr></dept>
</school>
```

이 XML를 [정의 1]에 의해 샘플 DTD와 인스턴스를 연역 객체 지향 데이터베이스 언어인 F-logic 언어로 바꾸면 다음과 같다.

- **Object Hierarchy Graph**

(1) bob : faculty (2) mary : faculty
(3) chris : faculty (4) faculty : employee
(5) phil : employee (6) sally : student
(7) student : person (8) employee : person

- **EDB**

(1) bob : faculty [name → "Bob," works → cs 1 : dept [dname → "CS," mgr → phil : employee]]
(2) mary : faculty [name → "Mary," works → cs 2 : dept [dname → "CS," mgr → mary : faculty]]
(3) chris : faculty [name → "Chris," works → ee : dept [dname → "EE," mgr → chris : faculty]]
(4) john : employee [name → "John," works → cs 2]
(5) sally : student [name → "Sally," major → cs]

Extensional 데이터베이스(EDB)는 XML을 이용하여 쉽게 나타낼 수 있지만 intensional 데이터베이스(IDB)를 어떻게 나타내느냐의 문제가 남아 있다. XML에서 사용자가 임의로 태그를 만들 수 있기 때문에 <IF> <THEN> </IF>, <AND> </AND>, <OR> </OR>, <NOT> </NOT>와 같은 IDB를 나타내기 위한 태그를 DTD로 정의하여 해결할 수 있으나 본 논문에서는 변환 결과에 XML로 표현 못하는 부분을 F-logic 언어의 구문에 맞게 GCI(General Class Information)와 IDB를 추가하여 질의를 처리한다.

- **GCI**

(6) faculty [supervisor → faculty]
(7) employee [supervisor → employee]

- **IDB**

(8) E [supervisor → M] ⇐ E : employee [works → D : dept [mgr → M : employee]]

- **Query**

(1) ? X : employee [supervisor → Y, works → D [dname → "CS"]]

4. F-logic 데이터베이스에서의 질의 처리

XML을 위한 데이터베이스는 새로운 데이터 모델을 지원하는 데이터베이스 시스템을 하부 저장 장치부터 질의어 처리기까지 모두 만드는 방법과 기존의 데이터베이스 시스템을 이용하여 XML 데이터를 저장하고 질의를 수행하는 방법으로 크게 나누어진다. XML은 기존 관계 데이터베이스의 테이블이나 객체 지향 데이터베이스의 클래스와는 다르게 그 구조가 가변적일 수 있고 질의 형태도 기존 데이터베이스 질의어와 다르기에 새로운 데이터 모델을 지원하는 전용 데이터베이스 시스템을 구현하는 것이 개념적 불일치 없이 잘 표현할 수 있고 질의할 수 있기에 후자보다 유리하다.

XML 데이터의 저장에 있어 F-logic언어로 표현하면 문서 내의 클래스, 엘리먼트의 의미를 보존하여 저장하는 방법을 사용하여 데이터베이스에 저장된 XML 문서에 의미 있는 질의를 쉽게 작성할 수 있다.

4.1 F-logic 데이터베이스에서의 Connection graph

F-logic에서의 질의 처리 기법으로 first-order 언어에서 사용되는 단순한 resolution 기법[8]을 사용할 수 있으나 본 논문에서는 *connection graph resolution* 기법[6]을 이용하여 질의를 처리한다. CG(Connection Graph) resolution 기법은 resolution 방법에 비해 resolution을 적용하기 위해 탐색해야 하는 공간을 작게 만들 수 있으며 resolution 순서를 조절하여 특정 탐색 공간에 대해 우선적으로 resolution을 행할 수 있으며 양방향으로 resolution을 행할 수 있다는 장점이 있다.

우선 질의 처리를 위해 F-logic 언어로 주어진 데이터베이스를 어떻게 CG로 만드는지에 대해 알아본다. F-logic 언어는 객체 지향적이라는 점이 first-order 언어와는 다르다. 따라서 객체가 무엇으로 구성되는지 알아야 한다. 객체의 구문적 구성원을 나타내기 위해 signature 표기법을 사용한다. Signature는 현 객체에 있는 심벌과 상위 클래스로부터 내려받은 심벌로 구성된다.

[정의 2] Signature

Signature $\Sigma = < U, <_U, P, O >$ 로

U, 유한한 실제 객체들의 집합,
$<_U$, U에 대한 부분 순서로 $O <_U O'$는 O가 O'의 하위 클래스임을 의미,
P, 속성과 값 쌍의 집합,
O, 객체 이름들의 집합.

하위 객체를 갖지 않는 객체들은 GCI나 IDB^0를 가질 수 없다. 즉, GCI나 IDB^0는 단지 클래스에만 정의된다. IDB^0는 하위 클래스에서 정의된 것이 상위 클래스에서 정의된 것을 대치할 수 있도록 색인된다. 이런 색인 정보는 CG를 만드는데 사용된다. GCI는 이런 색인이 필요 없는데 그 이유는 모두가 같은 최상위로 간주하기 때문이다.

한 클래스는 그 클래스에서 정의된 GCI와 IDB^0 그리고 상위 클래스로 부터 상속받은 GCI와 IDB^0를 포함하여야 한다. GCI는 변수를 가질 수 없으나 IDB^0는 변수를 가질 수 있다. [정의 1]에 의해 GCI와 IDB^0를 각각 Σ-GCI와 Σ-IDB^0라 하자.

[정의 3] Class Specification

한 클래스의 내역, $S = <\Sigma, c, \Phi, \Delta>$로 주어지는데

Σ, signature

c, $c \in O$인 클래스 이름

Φ, 클래스 c에서 정의 된 Σ-GCI와 클래스 c에 상속 된 Σ-GCI의 집합

Δ, 클래스 c에서 정의 된 Σ-IDB^{o}와 클래스 c에 상속 된 Σ-IDB^{o}의 집합

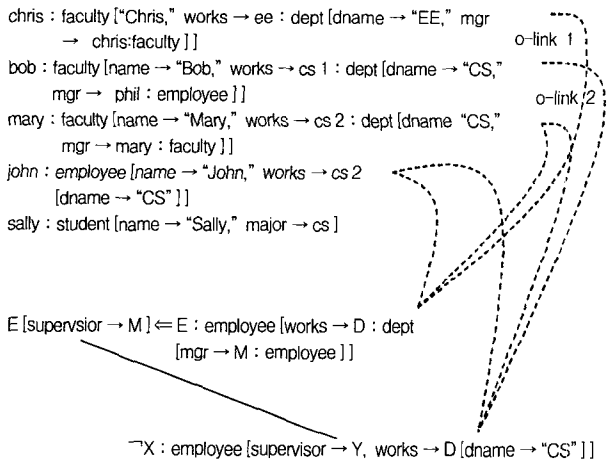
First-order 언어를 CG로 나타냈을 때 두 식(formula)이 연결되기 위해서는 상반적이면서 같아질 수 있는 속성을 가져야 한다. F-logic에서는 동종의 두 객체가 상반되며 한 객체의 속성이 다른 객체의 속성을 포함하거나 같아 질 수 있는 속성이 존재하면 된다. 두 객체 p, q가 $p \preceq_U q$ 또는 $p \succ_U q$ 이면 동종이라 한다.

[정의 4] Connection

동종의 객체 o와 o′ 사이에 connection 또는 link가 존재하기 위해서는 $\sum_{p \in o}$ 와 $\neg \sum_{p \in o'}$ 가 다음 조건을 만족시키면 된다.

1. unifier 존재하거나
2. $I(\sum_{p \in o}) \{\subseteq or \supset\} I(\sum_{p \in o'})$, 단 I는 어떤 경우에도 식을 참으로 만드는 해석(interpretation)이다.

(그림 2)는 [정의 1]에 의해 질의 ?X : employee [supervisor → Y, works → D [dname → "CS"]]가 더해진 예제 데이터베이스를 CG로 나타낸 것이다.



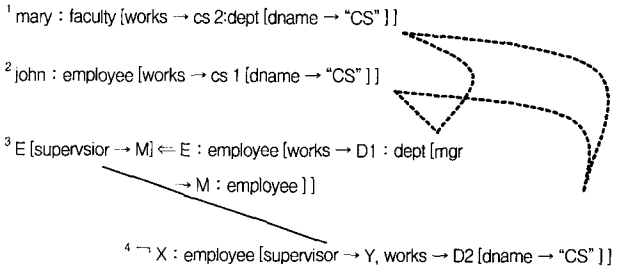
(그림 2) 생성된 CG

(그림 2)에서 보듯 두 종류의 link가 존재한다. 하나는 실선으로 표시된 일반적인 link이고 다른 하나는 점선으로 표시된 *object-link*(o-link)로 중첩된 객체를 연결한다. Bob의 경우 cs1에서 일하고 있는데 cs1의 mgr은 employee인 phil이다. 이는 GCI에서 faculty의 supervisor는 faculty이어야 한다는 제약 때문에 o-link가 형성될 수 없다. O-link는 그림에는 표시하지 않았지만 해당 속성 이름이 같이 표시한다. 예를 들어 객체 mary는 dname과 mgr o-link를 갖지만 faculty의 supervisor는 faculty이어야 한다는 GCI 때문에 객체 bob은 dname o-link만을 갖는다. 중첩된 객체에 resolution을 적용하기 위해서는 객체의 모든 속성(property)이 다른 객체의 상응하는 속성에 연결되어 있어야 한다. 모든 link가 존재할 때, 객체가 서로 다른 모든 o-link를 갖는다고 한다. 관계 데이터베이스의 중첩된 SELECT에서와 마찬가지로 중첩된 객체는 먼저 처리된다. 따라서 중첩된 객체에서 link 제거 가능 여부를 제일 먼저 알아본다. [Lemma 1]에 의해 (그림 2)에서 o-link 1과 o-link 2는 삭제된다.

[Lemma 1] o-link deletion : 서로 다른 모든 o-link를 갖지 않는 객체는 resolution에 사용될 수 없다.

[증명] resolution이 일어나기 위해서는 한 객체의 각 속성이 다른 객체의 해당 속성과 연결될 수 있어야 하는데 한 객체의 속성 중 다른 객체의 상응하는 속성과 연결되어 있지 않기에 resolution에 사용될 수 없다.

(그림 3)은 예제 데이터베이스로부터 Resolution에 사용되지 않을 connection을, 효율을 위해, 제거하고 난 다음 생성된 CG이다.



(그림 3) 불필요한 Connection을 제거한 후의 CG

4.2 CG Resolution을 이용한 질의 처리

질의를 처리하기 위해 CG resolution을 사용하는데 CG resolution에 의해 unsatisfiability에 도달하면 질의는 처리된 것이고 그 과정에서 얻은 substitution이 질의에 대한 답이 된다. 종래의 CG resolution과 달리 데이터베이스에 대한 질의의 답이 여러개 존재할 수 있기 때문에 CG 상의 connection이 더이상 존재하지 않을 때까지 resolution을 수행하여야 한다는 점이다.

CG resolution을 위해 연역 객체 지향 데이터베이스에서 connection에 의해 연결된 두 식을 같게 만들어야 하는데 이때 first-order 언어에서처럼 단순하게 구문을 동일하게 만드는 방법외에 구문을 같게 만들지 못하더라도 의미를 따져 같아 질 수 있는지도 고려해야 한다. 이렇게 의미를 따져 같아진 경우, 이를 minimal unifier라고 하며 Min, Max 함수는 이런 일을 도와준다.

[정의 5] Min and Max

$o_1 \leq_U o_2$일 때 $Min(o_1, o_2) = Min(o_2, o_1) = o_1$, $Max(o_1, o_2) = Max(o_2, o_1) = o_2$, 단 $o_1, o_2 \in \{O, 변수\}$. 만약 o_1과 o_2가 동종이 아니고, $o_3 <_U o_1$이고 $o_3 <_U o_2$이면서 o_4가 $o_3 <_U o_4 <_U o_1$와 $o_3 <_U o_4 <_U o_2$를 만족시키지 않으면 $Min(o_1, o_2) = o_3$, $Max(o_1, o_2) = \top$.

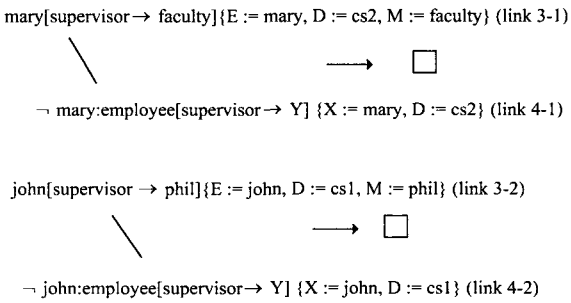
만약 함수 인자 중 하나가 변수이면 그 것이 Min 값이 되고 Max 값은 $\top$이 된다. 변수에 다른 인자 보다 작은 값이 들어갈 수 있기 때문이다. 둘 다 변수인 경우, 한 변수가 Min 값이 되고 Max 값은 최상위 객체 $\top$이 된다.

[Lemma 2] Minimal Unifier

동종인 [attr → T_1 : o]와 [attr → T_2 : o']에서 T_3를 Min(T_1, T_2), T_4를 Min(Max(T_1, T_2), Min(o, o'))라 하자. Minimal unifier는 [attr → T_3 : T_4]이다.

[증명] $T_1 <_U o$ 이고 $T_2 <_U o'$ 이기에 임의의 모델 I에 대해 $I[T_1] \subset I[o]$ 이고 $I[T_2] \subset I[o']$이다.

(그림 3)의 connection 4-1, 4-2, 3-1 그리고 3-2에 대해 CG Resolution을 수행시키면 (그림 4)를 얻게 된다. (그림 4)에서 질의에 대한 두 개의 답, mary [supervisor → faculty, works → cs 2]와 john [supervisor → phil, works → cs 1]을 구할 수 있다.



(그림 4) 4번의 Resolution 수행 후의 CG

5. 지식에 기반한 질의 처리

$I : O \rightarrow U$, 즉 OHG가 준동형(Homomorphism)을 이룬다는 점과 substitution이 결합 법칙을 만족한다는 점을 이용하여 Resolution에 사용될 식을 보다 효율적인 식으로 바꿀 수 있다.

[EXAMPLE 2] 다음 OHG와 질의를 고려해 보자.

(1) employee : person
(2) student : person
(3) workstudy : student
(4) workstudy : employee
(5) korea : foreign
? Y : employee [salary → (Z < 20000)] ∧ Y : student [origin → X : foreign, Name → N]

이 질의는 봉급이 2만불 이하인 사원과 외국에서 온 학생의 join을 필요로 한다. OHG에서 workstudy 클래스는 사원의 하위 객체이며 동시에 학생 클래스의 하위 객체임을 알 수 있다. Monotonic inheritance[2]의 정의에 의해, workstudy 클래스는 세 가지 속성, origin, salary 그리고 name을 갖게 된다. 이런 사실을 이용하여 주어진 질의를 같은 답을 내면서 보다 효율적인 질의 : ? Y : workstudy[origin → X : foreign, salary → (Z < 20000), name → N]으로 바꿀 수 있다. 이는 사원과 학생 클래스 사이에 Min값이 존재하기에 가능하였다. 즉, 공통 변수를 제한하는 두 클래스 사이에 Min 함수의 값이 존재하면 효과적인 질의로 바꿀 수 있다. 이때 공통 변수는 하나의 식 혹은 AND로 연결된 식들 사이에 존재하며 OR로 연결된 식들 사이에서의 같은 이름의 변수는 공통 변수라 하지 않는다.

[정의 6] Common Variable

질의 $Q \equiv ?subq_1 \wedge \cdots \wedge subq_i \cdots \wedge subq_j \wedge subq_k$ ($k \geq 1$) 라 하자. 변수 V는 다음의 경우 공통 변수라 한다.

1. $subq_i = p[\cdots, attr_{in} \rightarrow V : c, attr_{im} \rightarrow V : c', \cdots]$
2. $subq_i = p[\cdots, attr_{in} \rightarrow V : c, \cdots]$와 $subq_j = V : c'[\cdots, attr_{jm}, \cdots]$
3. $subq_i = p[\cdots, attr_{in} \rightarrow V : c, \cdots]$와 $subq_j = q[\cdots, attr_{jm} \rightarrow V : c', \cdots]$
4. $subq_i = V : c[\cdots, attr_{in}, \cdots]$와 $subq_j = V : c'[\cdots, attr_{jm}, \cdots]$ 단, $i \neq j$, $attr_{in} \neq attr_{jm}$, 그리고 $c, c', p, q \in S$.

공통 변수가 식에 있으면 [정의 6]에서와 같이 동일한 답을 주면서 효율적인 질의로 바꿀 수 있다. 1, 2, 3의 경우 c와 c'에서 Min값 c를 구해 c와 c'를 c로 대치한다. 4의 경우 Min값을 구한 후 상속(monotonic inheritance)이 이루어진다.

[정의 7] Query Reformation

$c \leq c$, $c \leq c'$이고 $p \leq c$, $p \leq c'$, $c \leq p$를 만족하는 클래스 p가 존재하지 않는다고 가정하자. [정의 5] 의 1, 2, 3, 4의 경우 각각 변형된 질의는 다음과 같다.

1. $subq_i = p[\cdots, attr_{in} \rightarrow V : c, attr_{im} \rightarrow V : c, \cdots]$
2. $subq_i = p[\cdots, attr_{in} \rightarrow V : c, \cdots]$와 $subq_j = V : c[\cdots, attr_{jm}, \cdots]$
3. $subq_i = p[\cdots, attr_{in} \rightarrow V : c, \cdots]$와 $subq_j = q[\cdots, attr_{jm} \rightarrow V : c, \cdots]$
4. $subq_{ij} = V : c[\cdots, attr_{in}, \cdots, attr_{jm}, \cdots]$, $subq_{ij}$는 $subq_i$와 $subq_j$의 join을 의미한다.

이 변형된 질의의 원래 질의와의 동등성과 효율성을 증명하여야 한다.

[Lemma 3] Equivalence

변형된 질의는 원래의 질의와 동등한 답을 준다.

[증명] {V : = p}가 원래 질의의 답이지만 변형된 질의의 답이 아니라 하자. p : c와 p : c'가 성립한다. 따라서 $p \leq c$이고 $p \leq c'$. $c \leq c$, $c \leq c'$이기에 $p = c$ 또는 $p < c$이다. 즉, {V : = p}도 변형된 질의의 답이 된다. 역의 경우도 비슷하게 증명된다.

[Lemma 4] Efficiency

변형된 질의의 성능이 보다 좋다.

[증명] 하나의 connection은 하나의 resolution을 의미하기에 변형된 질의가 보다 적은 수의 connection을 가짐을 보이면 되는데 이는 Min 값을 취하기에 분명하다.

6. 결 론

XML 데이터와 같은 비구조적인 데이터처리와 추론을 필요로 하는 의미 웹을 구축에 유리한 연역 객체 지향 데이터베이스(Deductive and Object-oriented Database)에서의 질의 처리 방안을 통해 XML 데이터 처리에 대해 알아보았다. 이를 위해 대량 문서 관리와 데이터 교환에 가장 유용한 마크업 언어로 알려진 XML을 이용하여 XML 데이터 모델을 연역 객체 지향 데이터베이스 모델로 바꾸는 방법을 기술하였고 이 연역 객체 지향 데이터베이스를 다시 Connection Graph로 바꾸고 Connection Graph Resolution을 이용하여 어떻게 질의에 답할 수 있는지를 기술하였다. 또한 데이터베이스 내의 계층 지식을 이용하여 효율적이면서도 같은 답을 주는 질의로 바꾸는 방법을 제시하고 이 방법이 효율적이며 논리적으로 타당하다는 점을 증명하였다.

본 논문에서 질의 처리시 단순 스칼라 유형의 데이터만을 목표로 하여 다루지 못한 set 유형의 데이터 처리와 대용량의 XML 데이터베이스 환경에서 보다 빠른 질의 처리를 위해 동시에 주어진 질의와 extensional 데이터베이스로부터 양방향으로 질의를 처리해 나가는 기법에 대한 연구가 향후 과제로 남아있다.

참 고 문 헌

[1] Abiteboul S. and et. al., "The Lorel Query Language for Semistructure Data," *International Journal on Digital Libraries*, Vol.1, No.1, pp.68-88, 1997.

[2] Allen C., "Application Integration with XML," *World Wide Web Journal*, Vol.2, No.4, pp.229-248, 1997.

[3] Kifer M., Lausen G. and Wu J., "Logical Foundations of Object-oriented and Frame-Based Languages," *Journal of the ACM*, Vol.42, No.4, pp.741-843. 1995.

[4] Kim S. and Henschen L., A Study on Search Space Reduction in Deductive and Object-oriented Databases, *Journal of Computing and Information*, Vol.1, No.1, pp.353-370, 1995.

[6] Kowalski R., "A Proof Procedure using Connection Graphs," *Journal of the ACM*, Vol.22, No.4, pp.572-595, 1975.

[7] Lam L. H. and Ng W., "The Development of Nested Relational Sequence Model to Support XML Databases," *Proceedings of the International Conference on Information and Knowledge Engineering '02*, pp.374-380, 2002.

[8] Laurent S., "XML Elements of Style," McGraw-Hill, 2000.

[9] Leitsch A., The Resolution Calculus, Springer, 1997.

[10] Marcus S. and Subrahmanian V. S., "*Foundations of Multimedia Database Systems*," *Journal of the ACM*, Vol.43, No.3, pp.474-523, 1996.

[11] May W., Modelling and Query Structure and Contents of the Web, Proceedings of the DEXA 99 Workshops, pp. 721-725, 1999.

[12] Ullman J. D., "A Comparison between Deductive and Object-Oriented Databases Systems," *Proceedings of the Second International Conference on Deductive and Object-Oriented Databases*, pp.263-277, 1991.

[13] World Wide Web Consortium, "The XML Data Model," 2000, http://www.w3.org/XML/DataModel.html.

[14] World Wide Web Consortium, "XQuery 1.0 and Xpath 2.0 Data Model," http://www.w3.org/TR/query-datamodel, 2001.

[15] World Wide Web Consortium, XSL Transformations, http://www.w3.org/TR/xslt, 1999.

김 성 규

e-mail : sgkim@aycc.anyang.ac.kr

1985년 홍익대학교 전자계산학과(학사)

1987년 서울대학교 계산통계학과(석사)

1994년 Northwestern 대학교 전산과(박사)

1987년~1996년 삼성종합기술원 선임연구원

1996년~1998년 안양대학교 컴퓨터공학과 전임강사

1998년~2001년 안양대학교 컴퓨터공학과 조교수

2001년~현재 안양대학교 컴퓨터공학과 부교수

관심분야 : 데이터베이스, 인공지능, World Wide Web