관계형 데이터베이스 상품 정보 질의 처리를 위한 인덱싱

Towards a Indexing Structure for Querying Product Information in Relational Databases

이현자(Hyunja Lee)*, 심준호(Junho Shim)**

초 록

관계형 데이터베이스를 사용하여 상품 온톨로지의 저장과 관리가 현실적 방안이라고 할 때, 의미적 관계에 대한 온톨로지 질의를 효과적으로 처리하기 위해서는 인덱스의 도움을 받는 것이 필요하다. 상품 온톨로지에서 의미적 관계는 상품 정보 간의 포함관계 및 분류계층 구조상의 위치관계 등 전이적 특성을 지닌 관계를 포함한다. 이 논문에서는 상품 정보 간의 포함관계 및 전이적 특성을 갖는 의미적 관계에 대한 질의를 효율적으로 처리할 수 있고, 정보의 갱신에 유리한 넘버링 기법을 사용한 인덱스 방법을 제안한다.

ABSTRACT

The product information can be practically stored and managed, and also queried when we use a relational database. We need to develop a special indexing scheme in order to process the queries to ask for the semantic relationships of the product information. Such semantic relationships include ISA or taxonomy relationships that have the transitive properties. In this paper, we propose an index scheme to effectively process those queries with the transitive property. The proposed index scheme is based on a numbering scheme that has relatively low update cost.

키워드 : 상품 온톨로지, 전자 상거래, 인덱스, 관계형 데이터베이스
Product Ontology, E-Commerce, Index, Relational Database

본 연구는 지식경제부 및 정보통신연구진흥원의 대학 IT연구센터 육성・지원사업(IITA-2008-C1090-0801-0031)의 연구결과로 수행되었음.

* 주저자, 숙명여자대학교 컴퓨터과학과 박사과정
** 교신저자, 숙명여자대학교 정보과학부 컴퓨터과학전공 부교수

2008년 10월 21일 접수, 2008년 10월 31일 심사완료 후 2008년 11월 07일 게재확정.

1. 서 론

전자 상거래 도메인에서 온톨로지의 실용적 활용을 위해서는 온톨로지 정보의 저장과 온톨로지적 질의 처리가 가능해야 하고, 이에 대한 현실적 방안은 RDBMS를 이용하는 것이다. 상품 온톨로지는 상품 및 서비스, 상품 분류체계, 상품을 설명하는 속성, 속성과 관련해서 속성 값을 구체적으로 설명해주는 측정 단위를 주요 개념으로 한다[7].

상품 온톨로지 질의가 잘 처리되기 위해서는 상품 온톨로지의 주요 개념과 이들 간의 관계에 대한 정보가 RDBMS에 잘 저장되어 유지 관리되고, 관계의 특성을 반영한 질의 처리가 효과적으로 이루어질 수 있도록 인덱스의 도움을 받거나 추가적인 정보(예를 들면, 계층 구조 상에서의 위치 정보)를 함께 유지해야 한다. 특히, 온톨로지에서 포함관계 및 계층 구조적 관계에 관한 질의는 이들 관계의 특성인 전이적(transitive) 특성 정보가 질의 처리에 반영되어야 한다.

XML, RDF/S, OWL과 같은 그래프 구조적인 온톨로지 정보의 처리에서 중요한 포함관계 혹은 계층구조 관계에 대한 처리를 위한 연구가 꾸준히 진행되고 있다[3, 4, 8, 11, 13~15]. 이들 연구는 트리 혹은 그래프 구조적인 데이터에서의 포함관계, 계층 구조에 관한 질의인 조상-후손 관계에 대한 질의 등 전이적 특성을 고려해야 하는 질의 처리를 위한 인덱싱 방법을 제안한다. 이들 연구의 인덱싱 방법은 그래프에서의 노드 혹은 간선에 숫자(혹은 이름)을 붙인 넘버링 기법(numbering scheme 또는 labeling scheme)을 사용한다. 넘버링 기법을 사용한 인덱스 방법이 그래프 구조의 질의에서 효율성이 뛰어나지만, 그래프에 새로운 정보가 노드로서 삽입될 때, 갱신 비용이 많이 드는 단점이 있다. 갱신 비용 을 줄이기 위한 방법으로, 인접한 노드 간에 일정 간격의 여유를 두어 넘버링하는 방안을 제시하고 있다[4, 14]. 그러나 이 방안은 노드에 넘버링을 할 때, 정수 타입을 사용하므로, 인접한 두 노드 사이에 여유 간격(공간)을 주더라도 어느 정도의 간격(즉, 번호의 차이)을 두어야 할 지 예측하기 힘들다. 두 노드 사이에 삽입될 새로운 노드에 넘버링할 수 있는 정수가 없다면, 최악의 경우, 전체 노드를 전부 넘버링해야 하기 때문에, 결국 해야 하는 갱신을 지연시킬 뿐이고, 근본적인 비용 문제를 해결하지는 못한다. 본 논문에서 제안하는 인덱스는 실수 타입으로 넘버링을 하여, 인접한 두 노드 사이를 더욱 정밀하게 넘버링할 수 있도록 함으로써 고비용의 갱신 문제를 해결할 수 있다.

본 논문에서는 상품 온톨로지에서의 ISA 관계, 혹은 component-of 관계와 같이 전이적 특성을 고려한 계층 구조에 관한 질의 처리에 효율적인 넘버링 기법을 사용한 인덱싱 방법을 제안한다.

본 논문의 구성은 제 2장에서는 관련연구이며, 제 3장에서는 상품 온톨로지의 RDB 스키마를 소개하고, 제 4장은 상품 온톨로지에서의 상품간의 계층적 관계 처리를 위한 인덱싱 방법을 제안한다. 이어서, 인덱스의 갱신 방법과 DAG로의 확장방법을 소개한다. 제 5장은 제안한 인덱스의 성능에 관한 실험 평가 결과를 보여주고, 제 6장에서 이 논문의 결론을 맺는다.

2. 관련 연구

이장에서는 그래프 구조적인 정보의 질의 처리를 위해 넘버링 기법을 사용하는 연구에 관해 소개한다.

XML뿐만 아니라 RDF/S, OWL, 온톨로지 정보의 저장과 질의 처리에 관한 최근 연구는 안정성과 실용성을 보장하는 RDBMS를 바탕으로 한다. 순서가 있고 방향성 있는 그래프 구조적인 데이터의 DBMS로의 저장과 질의 처리에서 중요한 것은 그래프 구조 정보를 잘 저장하여 포함관계 및 계층구조에 대한 질의가 효과적으로 처리될 수 있는지의 여부다. 이러한 질의 처리를 위해서는 특별한 인덱스가 필요하며[3], 인덱스 구축에 넘버링 스키마(또는 라벨링 스키마) 기법[3~5, 8, 13~15]을 사용할 수 있다.

[15]의 연구는 XML로 표현된 정보를 RDBMS에 저장하고 이에 대한 질의 처리에 관한 것으로 XML 문서의 포함관계에 관한 질의 처리를 효율적으로 처리하는 인덱스 방법을 제안한다. 제안된 인덱스는 XML 문서의 요소(Element)와 텍스트(text)가 출현하는 순서에 따라 각각 번호를 붙인 역 인덱스이다. XML의 요소를 위해서는 e-index 테이블에 정보를 저장하고, 텍스트의 정보는 t-index 테이블에 저장한다. XML내의 요소를 노드로 해서 구성된 트리 구조의 XML 문서를 깊이 우선 탐색 방법의 순서대로 노드에 번호를 붙이는 넘버링 스키마방법을 사용한다.

Tatarinov et al.은 [14]에서 XML 트리에 넘버링 기법 세 가지를 소개한다. 전역(Global) 순서, 지역(local) 순서, 듀이[6] 순서가 그 세 가지이며, 전역 순서 방법은 트리의 루트를 시작으로 깊이 우선 탐색의 순서로 넘버링하고, 지역 순서 방법은 트리에서 형제(sibling) 노드들의 순서만을 넘버링한다. 이 방법으로는 조상노드나 손자노드를 찾기가 어렵다. 전역 순서와 지역 순서방법의 장단점을 보완한 것이 듀이순서 방법이다. Dewey decimal 분류[6]에 기반한 순서로 넘버링 하는 것으로, 트리의 루트에서 각 노드의 경로를 백터로 할당하는 것이다(즉, 루트노드 1, 자식노드 1.1, 손자노드 1.1.1...). 지역 순서와 듀이 순서의 넘버링 방법은 새로운 노드 추가로 인한 트리의 갱신이 필요할 때, 노드 전체를 새로 넘버링 할 필요가 없으므로, 전역 순서 방법보다 트리의 갱신에 있어서 유리하다.

[3]은 RDF/S의 포함관계 및 조상-후손 관계에 대한 질의 처리를 위해서 접두어(prefix) 넘버링 방식과 간격(interval) 넘버링 방식이 효율적임을 실험으로 증명해 준다.

접두어 방식은 부모노드의 정보를 자식노드가 유지하는 방법으로 듀이 순서 방식이 바로 여기에 해당된다. 간격 방식은 노드에 부여하는 시작번호와 끝번호 간에 차이(간격)을 두어 넘버링하는 기법으로 [1, 5, 8, 15]이 모두 이 방법에 해당된다.

[5]는 트리에서의 조상-후손 관계 검색을 위해 넘버링 기법을 처음 사용하였고, 트리를 전위 순서와 후위 순으로 각각 탐색하여 트리의 모든 노드에 <preorder, postorder> 두 개의 번호를 지정한다. 조상-후손 관계에 있는 두 노드는 전위 순서 방법으로는 조상노드 번호가 후손 노드의 번호보다 더 작고, 후위 순서일 때는 조상의 노드 번호가 후손의 번호보다 더 큰 관계가 항상 성립함을 이용해서 조상-후손관계를 알아낸다. [8]은 [5]

의 넘버링 방법의 확장으로 XML 트리의 각 노드에 <order, size>의 번호를 깊이 우선 탐색 방법으로 지정한다. order는 전위 순서를 의미하며, size는 해당 노드의 모든 자식 노드를 합한 수를 의미한다.

그 외 그래프 구조적인 온톨로지 데이터에 대한 인덱스 방법으로 suffix arrary[11]와 [13]이 있다.

Suffix array는 RDF/S의 그래프의 모든 경로를 배열에 저장하고 질의 처리에 이용한다. 노드와 에지에 레이블링된 DAG 형태의 그래프인 RDF/S에서 스키마경로, 인스턴스 경로, 클래스경로, 프로퍼티경로를 각각 추출하고 특별한 기호를 붙여 구별하고 배열로 저장하여 인덱스로 사용한다.

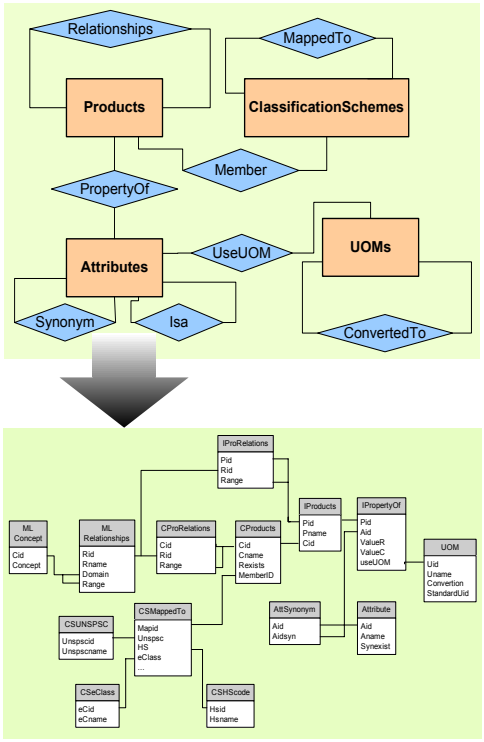
3. 상품 온톨로지의 RDB스키마

상품 온톨로지 모델은 OWL로 표현할 수 있을 뿐만 아니라, EER, UML과 같은 개념적인 모델 표현이 가능하며, 이러한 표현은 RDB 스키마로의 변환을 용이하게 한다[9].

<그림 1>은 [7]기반의 상품 온톨로지 모델을 RDB 스키마로 변환한 하나의 예이다.

RDBMS에서 상품 온톨로지 질의가 잘 처리되기 위해서는 우선, 상품 온톨로지의 개념들간의 다양한 관계-ISA, componentOf, mappedT0, PropertyOf, Member-of 등-에 관한 정보가 잘 저장되어야 하므로 잘 설계된 RDB 스키마가 중요하다.

<그림 1>[10]의 RDB 스키마 중에서 다음 장의 인덱스 예제에 필요한 부분을 간략히 소개한다.



〈그림 1〉 상품 온톨로지 모델을 RDB 스키마로의 변환 예

CProducts(Cid int, Cname char, MemberID)
IProducts(Pid int, Pname char, Cid int)

CProducts의 Cid와 Cname 필드는 상품 클래스 아이디와 상품 클래스의 이름을 의미한다. MemberID 필드는 상품 분류 스키마에 대한 테이블과 연관 있는 필드로 이 논문에서는 자세한 설명은 생략한다.

IProducts 테이블은 모든 상품 인스턴스에 대한 정보 테이블이다. Pid는 상품 인스턴스 아이디, Pname은 상품 인스턴스 이름, Cid는 CProducts 테이블의 상품 클래스 아이디를 참조하여, 해당 인스턴스가 속한 상품 클래스 정보를 저장한다.

4. 상품 온톨로지에서의 ISA 관계를 위한 인덱싱 방법

일반적으로 온톨로지 질의 유형은 도달 가능(Reachability), 조상 및 후손(Ancestor/Sucessor)의 집합, 그리고 최소 공통 조상(least common ancestors)에 관한 질의로 구분할 수 있다[12]. 이들 질의는 그래프에서의 구조적 관계 및 전이적 특성에 대한 처리가 가능해야 하며, RDBMS에서는 이를 위한 효과적인 인덱스를 필요로 한다.

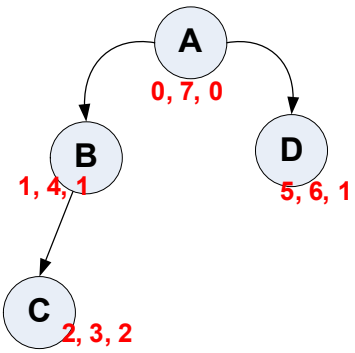
본 논문에서는 상품 온톨로지의 조상 및 후손 관계의 계층 구조에 관한 질의 처리를 위해 넘버링 기법[3~5, 8, 13, 15]을 사용한 인덱스 방법을 제안한다.

개념간의 조상 및 후손 관계의 계층 구조에 관한 질의 처리는 이들 관계의 전이적 특성을 반드시 고려하여 처리되어야 한다. 상품 온톨로지에서 전이적 특성을 지닌 상품 의미적 관계로는 상품간의 ISA 관계, component-of 관계가 대표적이다.

그래프 구조적인 상품 온톨로지에서 상품간의 전이적 관계의 계층 구조를 트리 또는 DAG로 표현할 수 있다. 문제를 단순화하기 위해 본 논문에서는 일단, 트리로 가정하고 넘버링 기법을 적용한다.

다음은 상품간의 ISA 관계에 적용한 넘버링 기법에 대한 설명이다.

<그림 2>를 상품 온톨로지에서의 상품간의 ISA 트리 관계(포함 관계)라고 가정한다. A노드는 상품 클래스 중에서 최상위 클래스이며 개념적으로만 존재하는 클래스이다. C노드와 D노드는 자식 노드가 없는 잎 노드이다. 노드 간의 ISA 관계 정보를 위한 넘버링은 루트 노드인 A노드를 번호 0으로 시작해서 깊이 우선 탐색하여 각 노드에 번호를 부여한다. 노드를 처음 방문한 순서대로 시작번호를 부여하고, 탐색 순서에 따라 되돌아 방문하는 순서대로 끝 번호를 지정한다.



〈그림 2〉 상품 온톨로지에서 상품간의 ISA 트리

시작번호, 끝번호와 함께 트리에서의 레벨 번호도 함께 지정한다. 루트 노드의 레벨을 0으로 한다.

이런 방식으로 상품온톨로지 ISA 트리 내의 모든 노드에 세 쌍의 번호를 부여한다. 각 노드의 세 쌍의 번호는 서로 비교연산을 통해 노드 간의 포함관계를 판별할 수 있다.

예를 들어, 그림의 ISA 트리에서 두 노드 B, C의 세 쌍의 번호를 비교해보자. 노드B, C의 번호는 각각 B(1, 4, 1), C(2, 3, 2)이다. 트리에서 B노드는 C노드의 부모 노드이다.

두 노드의 시작번호를 비교해 보면 B노드가 C노드보다 작으며, 끝 번호는 B노드가 C노드 보다 크다. 이러한 관계가 성립되면 B노드는 C노드의 조상 노드이다. 특히, B노드의 레벨 번호는 C노드보다 1 작고 이러한 경

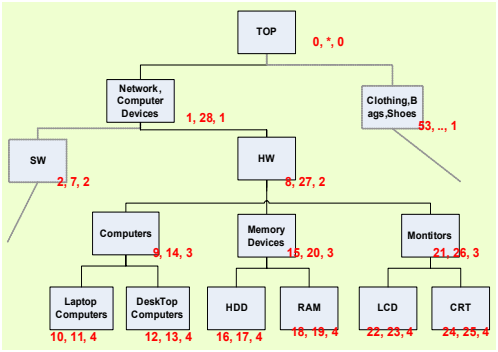
우, B노드는 C노드의 부모 노드이다. 이와 같이 각 노드의 세 쌍의 번호를 서로 비교하면, 조상-후손(부모-자식)노드를 판별할 수 있다.

위의 조건을 일반화하여 표현하면 다음과 같다.

ISA 트리의 임의의 두 노드 A(startNOA, endNOA, levelA), B(startNOB, endNOB, levelB)이라고 할 때,

조건식 1 : startNOA < startNOB이고 endNOA > endNOB이면, A노드는 B노드의 조상 노드이다.

조건식 2 : startNOA < startNOB이고 endNOA > endNOB이고 levelB = levelA + 1을 만족하면 A노드는 B노드의 부모 노드이다.



〈그림 3〉 상품 온톨로지에서 상품간의 계층구조의 한 예

<그림 3>은 상품 온톨로지에서 상품간의 ISA 관계를 보여주는 한 예이다. 위 그림에서의 관계를 ISA가 아닌 relatedTo와 같은 또다른 의미적 관계로의 해석이 가능하나, 본 논문에서는 ISA 관계로 가정한다. 본 논문의 그림 표현 기호는 [9]의 EER 표현을 따른다.

그림에서 Top 클래스는 ISA 관계를 이루고 있는 모든 상품 클래스의 최상위 클래스이며 개념적인 클래스이다. Top 노드를 시작으로 깊이 우선 탐색을 하면서 노드마다 번호를 붙여나간다. 탐색하면서 처음 노드를 방문할 때의 번호를 시작번호로, 되돌아 방문할 때의 번호를 끝 번호로 지정한다. 그리고 레벨 번호는 트리에서의 깊이에 대한 정보로 top노드를 0번으로 시작해서 각 노드에 부여한다. 넘버링의 순서는 트리를 깊이 우선 탐색방법으로 모든 노드를 방문한 후, Top노드로 되돌아오면서 끝나게 된다. 따라서 Top노드의 끝번호는 상품 온톨로지의 ISA 관계에 포함되는 모든 상품의 개수보다 훨씬 크며, Top노드의 끝번호는 새로운 노드(상품 클래스)가 트리에 확장될 것을 감안해서 상당히 큰 수로 지정한다. 이러한 고려는 트리의 갱신(새로운 노드의 삽입) 혹은 확장에 유리하도록 하는 것이다. 이렇게 해서 상품온톨로지 ISA 트리 내의 모든 노드는 그림에 표시한 것처럼 세 쌍의 번호가 붙여진다.

상품 온톨로지의 ISA 관계를 위한 인덱스 테이블을 ISAindex라고 할 때, RDB 스키마는 다음과 같다.

ISAindex (cid number startNO number, endNO number, level number)

cid 필드는 상품의 id로 앞서 소개한

CProducts 테이블의 상품 id번호를 참조한다. StartNO 필드, endNO 필드, 그리고 level 필드는 깊이 우선 탐색 방법으로 트리를 순회하면서 부여한 세 쌍의 번호이다. 각 노드의 세 쌍의 번호는 다음과 같이 서로 비교하여 포함관계를 알아낼 수 있다.

상품 온톨로지에서 임의의 두 상품 클래스를 P, Q라고 가정한다.

두 상품 클래스의 RDB 스키마는 각각 P(cidA, startNOP, endNOP, levelP), Q(cidQ, startNOQ, endNOQ, levelQ)이다.

조건식 1 : startNOP < startNOQ이고 endNOP > endNOQ이면, P 상품은 Q 상품의 상위 클래스이다.

조건식 2 : startNOP < startNOQ이고 endNOP > endNOQ이고 levelQ = levelP + 1을 만족하면, P 상품은 Q 상품의 부모 클래스이다.

그림에서 HW 클래스의 번호는 (8, 27, 2)이고, RAM 클래스는 (18, 19, 4)이다.

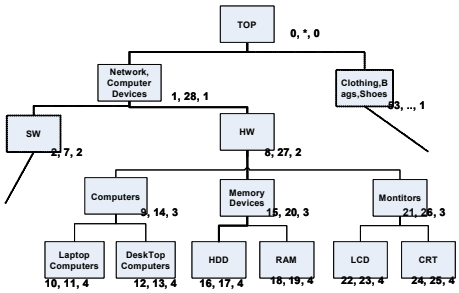
두 클래스의 세 쌍의 번호를 비교하면, 8 < 18 이고 28 > 19이므로 RAM은 HW의 후손 관계가 성립한다.

다음의 두 예제는 RDBMS에서 이 인덱스를 사용한 ISA 관계의 질의를 SQL로 표현하고 질의를 실행한 결과를 나타내고 있다.

질의 :
'HDD' 상품의 모든 조상에 해당하는 상품은?

SQL :
SELECT cp1.cname
FROM ISAindex, ISAindex AS c1,
Cproducts AS cp, Cproducts AS cp1
WHERE cp.cname = 'HDD' And c1.cid = cp.cid
And ISAindex.sno < c1.sno And
ISAindex.eno > c1.eno And
ISAindex.cid = cp1.cid ;

결과 :

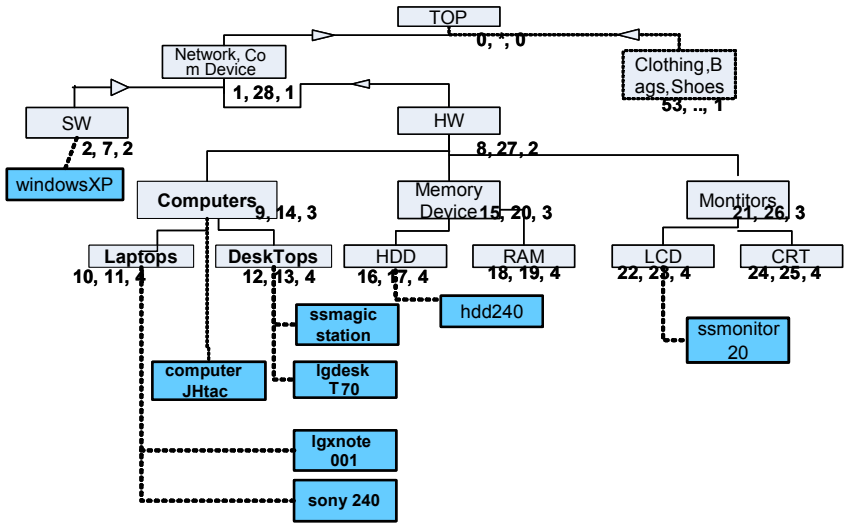


위 그림에서, 각 노드의 세 쌍의 번호는 질의처리 과정에서 HDD의 (16, 17, 4)와 서로 비교된다. 비교의 결과가 앞의 조건 식에 만족하면 결과를 출력한다.
즉, 시작번호는 HDD의 시작번호인 16보다 작고,끝번호인 17보다 크면 HDD 상품의 조상 클래스에 해당하는 상품이다. 따라서, Memory Devices(15, 20, 3), HW(8, 27, 2), Network, and Computer Devices(1, 28, 1) 노드가 조건에 만족하므로 HDD의 조상 상품(클래스)이다. 아래의 테이블은 SQL 질의 처리로 출력된 결과다.

	cname
	Network, Comp
▶	HW
	memory device

의미 :
모든 컴퓨터 상품(인스턴스)를 검색하라

SQL :
SELECT ip.pid, ip.pname
FROM IProducts AS ip, [SELECT ci1.cid, ci1.sno, ci1.eno, ci1.level FROM CProducts AS cp, ISAindex AS ci1, ISAindex AS ci2 WHERE cp.cname = 'Computers' And cp. cid = ci2.cid and ci2.sno <= ci1.sno And ci2. eno >= ci1.eno And ci2.level <= ci1.level]. AS ci3 WHERE ci3.cid = ip.cid ;



이 예제는 인스턴스를 포함한 질의이다. 이 질의에 대한 응답으로서 기대되는 결과는 컴퓨터(computers) 클래스와 ISA 관계를 갖고 있는 데스크탑(DeskTops) 클래스, 랩탑(Laptops) 클래스의 모든 인스턴스를 반환하는 것이다. 우선, ISAindex 테이블의 상품 클래스의 세 쌍의 수를 서로 비교해서 해당 하위 클래스를 찾아낸 다음, 컴퓨터 클래스를 포함한 결과클래스 아이디(cid)를 통해 모든 인스턴스가 저장된 IProduct 테이블의 인스턴스 각각의 cid와 일치하는 인스턴스를 반환해 낸다.
질의 처리 결과는 컴퓨터 클래스 인스턴스인computerJHtac, 랩탑과 데스크탑 클래스 인스턴스인 lgxnote001, sony240 ssmagicstation, 과 lgdeskT70을 아래와 같이 모두 올바르게 출력해 내고 있다.

pid	pname
11	computerJHtac
2	lgxnote001
9	sony240
1	ssmagicstation
10	lgdeskT70

4.1 상품온톨로지 ISA트리의 갱신

상품 온톨로지 ISA 트리를 깊이 우선 방법으로 순회하면서 각 노드에 일련의 번호를 부여하기 때문에 새로운 노드를 트리에 추가할 때, 최악의 경우에는, 전체 노드의 번호를 모두 갱신해야만 한다. 반면, 노드를 삭제할 때에는 어떠한 경우라도, 해당 노드를 제거하기만 하면 된다.

새로운 노드를 추가할 때 발생할 수 있는 이와 같은 문제점을 개선하기 위해 본 논문에서는 다음의 방법을 제안한다.

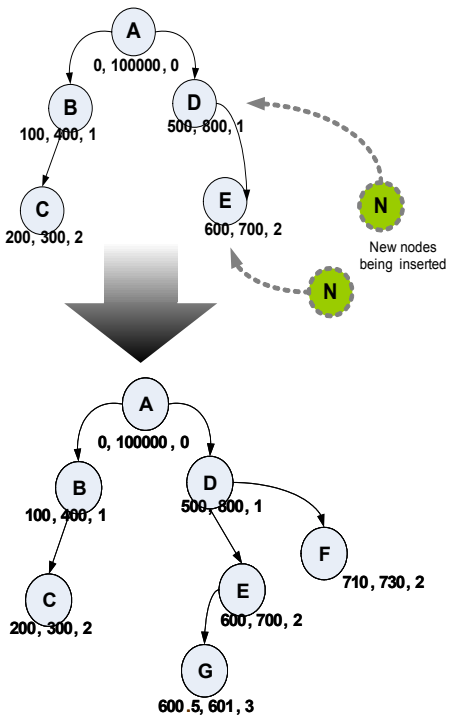
- 정수가 아닌 실수형으로 넘버링한다.
- 인접한 노드라고 할지라도 어느 정도의 간격을 두고 넘버링한다.

트리를 구성할 때 각 노드에 부여하는 번호의 간격을 여유 있게 한다. 뿐만 아니라, 노드와 노드사이에 더 많은 공간 여유를 주기 위해 번호를 정수가 아닌 실수로 부여한다. 두 실수 사이에는 무수히 많은 실수가 존재하므로, 새로운 노드에 부여할 충분한 공간의 여유가 생긴다. 그러나 실수표현에 있어서 허락하는 유효자리수만큼만 나타낼 수 있기 때문에 만약 4byte 혹은 8byte의 저장공간(용량) 이라면 정수 타입이든 실수 타입이든 넘버링에 사용할 수 있는 수의 가지 수는 차이가 없을 수도 있다. 하지만, 정수를 사용했을 때는 두 노드 사이에 여유 간격을 두더라도 어느 정도의 간격을 두어야 할 지 예측하기 힘들고, 무한정 간격을 둘 수도 없다. 실수 타입으로 넘버링을 할 경우에는 인접한 두 노드 사이를 더욱 정밀하게 넘버링 할 수 있는 장점이 있다.

이러한 장점이 효과를 발휘하려면, 유효숫자를 늘려야 하는데, 이것은 해당필드의 크기를 증가시키는 결과를 초래해, 결국 질의 처리 성능에 영향을 미칠 수 있다.

하지만, 직접 실험을 해 본 결과, double 타입(8byte)의 실수 타입을 사용한 넘버링과 long 타입(4byte)의 정수 타입을 사용해서 넘버링했을 때 성능에서의 차이가 없음을 확인할 수 있었다(실험환경 : MS-ACCESS 2007의 기본 셋팅, 115,996노드와 10레벨의 트리구조를 실험 데이터 셋으로 사용함, 참조 제 5장의 실험 평가).

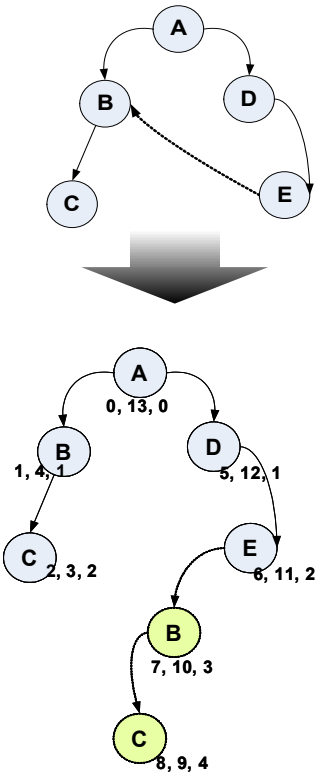
<그림 4>는 본 논문에서 제안한 넘버링 방법으로 상품간의 ISA 트리 갱신에 유리하다. 각 노드의 번호는 일정 간격을 두고 부여



〈그림 4〉 상품 온톨로지의 ISA 트리의 갱신을 위한 넘버링 제안 방법

한다. (이전 그림의 노드 번호를 시작번호와 끝 번호에 한해서 100배 비례적으로 증가 시킨 그림임) 이렇게 하면 각 노드의 번호 사이에 간격이 생기므로, 새로운 노드를 삽입해야 할 때, 기존의 노드의 번호를 갱신하지 않아도 된다. 인접한 두 노드 간에 어느 정도의 여유 간격을 두고 넘버링하는 아이디어는 [4]와 [14]에서의 넘버링 방법과 유사하다. 그러나 본 논문에서는 여기에 더하여 실수 데이터 타입으로 넘버링함으로써, 두 노드 간의 차이를 더욱 정밀하게 넘버링할 수 있도록 하여 많은 노드를 삽입하여도, 새로운 노드를 삽입하는데 드는 비용은 거의 O(1)로 일정하게 유지할 수 있는 장점이 있다.

4.2 상품온톨로지 DAG로의 확장



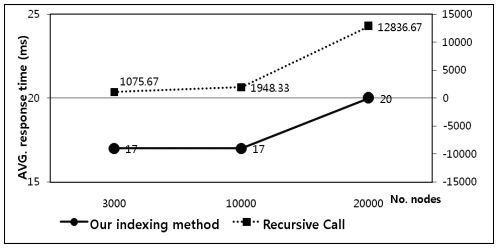
〈그림 5〉 상품 온톨로지의 ISA tree의 DAG로의 확장

지금까지 트리로 가정한 상품 온톨로지 ISA 트리를 그래프(DAG)로 어렵지 않게 확장할 수 있다.

<그림 5>와 같이 노드 E에서 B로 향하는 간선이 있으므로, B노드는 E노드의 자식 노드가 되고, C노드는 E노드의 자손 노드가 되고, 이를 위 그림의 아래와 같이 트리로 표현할 수 있다. B노드와 C노드에 원래의 번호 외에 새로운 번호 (7, 10, 3)과 (8, 9, 4)를 각각 부여한다. 그러나 이 방법은 <그림 5>와 같이 들어오는 간선이 E노드에서 B노드이지만, B노드 뿐 아니라, B노드의 하위 노드 전체에 새로운 번호를 부여하여야 하므로 저장 공간을 많이 필요로 하게 된다. 이런 방식으로 미리 계산된 TC(transitive closure set)에 드는 비용은, 50,000노드의 데이터를 갖고 실험했을 때 시간에 관해 $O(n^3)$와 공간에 관해 $O(n^2)$이다[2, 13]. 저장 공간의 비용 문제를 개선하기 위해서 [13]에서는 노드 B에만 새로운 번호를 할당하고 그 하위 노드는 쿼리 실행시간 동안에 해당 번호를 계산하는 방법을 제안한다. 그러나 이 방법은 상대적으로 쿼리 시간이 많이 걸리는 단점이 있다.

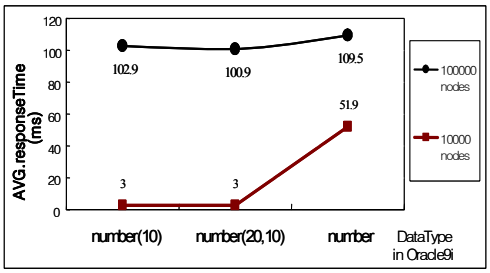
5. 실험 평가

이 장에서는 제안한 인덱스를 사용하여 질의 했을 때, 전이적 특성을 지닌 관계에 대한 처리의 성능 개선 효과를 실험을 통해 입증해 본다. 실험은 두 가지인데, 첫 번째는 제안 인덱스를 사용했을 때와 인덱스 없이 질의 처리할 때의 성능 차이 비교 실험이며, 두 번째는 실수형으로 넘버링하여 필드크기가 커짐에 따라 성능의 차이가 어떠한지를 비교해 보는 실험이다.



〈그림 6〉 제안된 인덱스를 사용했을 때의 성능 비교

전이적 특성을 지닌 의미적 관계에 대한 질의 처리는 인덱스가 없을 때에는 대개 반복적인 조인을 통해 이루어진다. <그림 6>은 제안한 인덱스를 사용했을 때와 재귀적인 호출을 통해 조인을 했을 때의 성능에서의 차이를 보여 주고 있다. 노드수가 많아질수록 재귀적 조인의 처리 시간이 현격히 오래 걸린다는 것을 볼 수 있다. 이 실험의 실행은 intel core2 1.83 GHz CPU, 1G의 RAM을 장착한 하드웨어에, 운영체제는Window-Vista, 프로그래밍 환경은 JDK 1.4v, RDBMS는 MS-Access2007의 기본 세팅 환경에서 이루어 졌다.



〈그림 7〉 필드 크기와 노드 수에 따른 성능 비교

<그림 7>은 오라클 9i 환경에서 NUMBER 데이터 타입(decimal형)을 사용하고 필드크기를 달리 지정했을 때, 노드 수에 따라 제안한 인덱스를 사용한 질의 처리 성능이 어떻게 달라지는지를 실험한 결과이다. 오라클에서 number(p, s)의 데이터 타입은 p는 정밀도, s는 크기를 의미한다. 예를 들어, number(5, 3)인 데이터 타입이라면 소수점 이하의 유효 자리수는 3자리까지 가능하며, 이를 포함하여 표시할 수 있는 전체 자리수는 5자리이다. 정밀도와 크기를 지정하지 않으면, -10^{38}~10^{37} 범위의 수를 표현할 수 있다.

<그림 7>에서 Number(10), number(20, 10), number(지정안함)의 세 가지 경우로 필드 크기를 달리 했을 때, 필드 크기가 클수록 성능이 좋지 않음을 볼 수 있다. 그러나 노드 수가 10,000개인 경우와 100,000개인 경우를 비교하여 볼 때, 노드 수가 많을수록 필드 크기에 따른 성능의 차이가 거의 없음을 확인할 수 있다. 즉, 노드 수가 많아질수록 실수 데이터 형을 사용해도 성능의 차이가 거의 없으므로 인덱스 갱신에 있어서 유리한 실수 데이터 형을 사용하는 것이 효과적임을 알 수 있다.

〈표 1〉 인덱스 갱신 비용

Operation	삽입	삭제	데이터형
Time Complexity	O(1)	O(1)	실수형
	O(n)	O(1)	정수형

<표 1>은 실수 데이터 타입을 사용하여 넘버링한 제안한 인덱스의 갱신에 드는 비용을 표로 보여주고 있다. 정수 타입을 사용하여 일련의 번호로 넘버링 했을 때, 그래프의 갱신 비용은 최악의 경우, O(n)이다. 여기서 n은 노드의 수를 의미한다. 본 논문에서 제안한 인덱스는 전체 노드 전부를 새로 넘버링할 필요가 거의 없으므로 삽입과 삭제에 있어서 O(1)의 비용이 들 뿐이다.

6. 결 론

본 논문은 상품 온톨로지의 ISA, compo-

nent-of 관계와 같은 전이적 특성을 지닌 관계 질의의 효율적 처리를 위해 넘버링 기법을 사용한다 인덱스 방법을 제안한다. 본 논문에서 제안한 넘버링 기법의 인덱스 방법은 갱신에 있어서 유리한 다음의 두 가지 특징이 있다.

1) 인접한 노드간에 여유 간격을 두어 넘버링한다.
2) 정수가 아닌 실수형으로 넘버링하여 인접한 두 노드 사이의 넘버링을 더욱 정밀하게 할 수 있다.

위와 같은 특징으로 새로운 노드가 추가될 때 인덱스 갱신 비용이 거의 들지 않으며, 정밀한 표현의 실수 데이터 형을 사용하더라도, 노드 수가 많아 질수록 질의 처리 성능의 차이는 거의 없음을 실험을 통해 입증했다. 제안한 인덱스 방법은 상품 온톨로지 관계 트리 뿐 아니라 DAG 확장 형태에서도 사용할 수 있다.

참 고 문 헌

[1] R. Agrawal, A. Borgida, H. V. Jagadish, Efficient management of transitive relationships in large data and knowledge bases, In Pro. of the 1989 ACM SIGMOD, 1989.

[2] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, Introduction to Algorithms, MIT Press, 2001.

[3] V. Christophides, D. Plexousakisa, M. Scholl, and S. Tourtounis, On labeling schemes for the semantic web. In Pro. of the 13th World Wide Web Conference, 2003, pp. 544-555.

[4] Shu-Yao Chien, Zografoula Vagena, Donghui Zhang, Vassilis J. Tsotras, Carlo Zaniolo, Efficient Structural Joins on Indexed XML Documents, In Pro. of VLDB, 2002.

[5] Paul F. Dietz. Maintaining order in a linked list. In Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing, pp. 122-127, San Francisco, California, May 1982.

[6] Online Computer Library Center. Introduction to the Dewey Decimal Classification, http://www.oclc.org/oclc/fp/about/about_the_ddc.htm.

[7] Taehee Lee, Ig-hoon Lee, Suekyung Lee, Sang-goo Lee, Dongkyu Kim, Jonghoon Chun, Hyunja Lee, and Junho Shim, Building an Operational Product Ontology System, Electronic Commerce Research and Applications, 2006.

[8] Q. Li and B. Moon, Indexing and querying XML data for regular path expressions, In Proc. of VLDB 2001, 2001.

[9] Hyunja Lee and Junho Shim, Conceptual and Formal Ontology Model of e-Catalogs, In Pro. of 6th International Conference on Electronic

Commerce and Web Technologies (EC-Web 2005), in conjunction with DEXA, 2005.

[10] HyunjaLee and Junho Shim, Storing and Querying for Product Ontology, Sookmyung Women's Univ. technical report, 2008.

[11] A. Matono, T.Amagasa, M. Yoshikawa, and S. Uemura, An Indexing Scheme for RDF and RDF Schema based in Suffix Array, In Pro. of SWDB, 2003.

[12] Silke Trißl, Ulf Leser, Querying Ontologies in Relational Database Systems, In Pro. of DILS 2005, pp. 63-79.

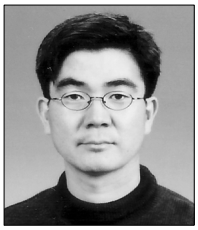
[13] Silke Trißl, Ulf Leser, Fast and practical indexing and querying of very large graphs, In Pro. of ACM SIGMOD 2007, pp. 845-856.

[14] Igor Tatarinov, Stratis Viglas, Kevin S. Beyer, Jayavel Shanmugasundaram, Eugene J. Shekita, and Chun Zhang, Storing and querying ordered XML using a relational database system, In Pro. of ACM SIGMOD 2002, pp. 204-215.

[15] C. Zhang, J. Naughton, D. Dewitt, Q. Luo, and G. Lohman, On Supporting Containment Queries in Relational Database Management Systems, In Pro. of ACM SIGMOD, 2001.

저 자 소 개

이현자 (E-mail : hyunjalee@sm.ac.kr)
1996년 숙명여자대학교 컴퓨터과학과 (학사)
2004년 숙명여자대학교 컴퓨터과학과 (이학석사)
2004년~현재 숙명여자대학교 컴퓨터과학과 박사과정
관심분야 데이타베이스, 전자상거래, 상품정보, 온톨로지



심준호 (E-mail : jshim@sookmyung.ac.kr)
1990년 서울대학교 계산통계학과 (학사)
1994년 서울대학교 계산통계학과 전산과학전공 (이학석사)
1998년 Northwestern Univ, USA, Electrical and Computer Eng, (공학박사)
1999년 Computer Associates Int'l, USA, R&D Staff
1999년~2001년 Drexel Univ, USA, Assistant Professor
2001년~현재 숙명여자대학교 정보과학부 컴퓨터과학전공 부교수
관심분야 데이타베이스, 전자상거래, 상품정보, 온톨로지