상품 온톨로지 저장을 위한 관계형 데이터베이스 논리적 설계

Logical Design of Product Ontology in Relational Databases

이현자(Hyunja Lee)*, 심준호(Junho Shim)**

초 록

전자 상거래 도메인에서 온톨로지가 실용적으로 활용되기 위해서는 온톨로지를 잘 저장 관리하고 사용자가 기대하는 온톨로지 질의에 대한 응답 처리가 가능해야 한다. 다양한 장점과 신뢰성을 바탕으로 한 관계형 데이타베이스 기반의 상품 온톨로지의 저장이 현실적 실용 방안이라고 할 때, RDBMS에서 온톨로지 질의 처리가 가능하기 위해서는 상품 온톨로지의 개념과 개념들간의 관계에 대한 정보가 잘 처리될 수 있도록 저장되어야 한다. 이 논문에서는 관계형 데이타베이스 기반에서 확장성과 효율적인 상품 온톨로지의 의미적 관계에 대한 질의 처리를 위한 데이터베이스 설계 방법을 제안한다.

ABSTRACT

In order to use a RDBMS as the repository for product information, it is essential that the ontological query be processed effectively and answered properly, satisfying the user's expectations for an ontological query process. To be well-processed ontological queries, the key point is whether the various semantic relationships among the concepts of the product information are likewise well-processed. For feasibly processing a semantic query, we need to consider how the product ontology data is designed and stored in RDBMS. In this paper, we present how a RDB schema for the product ontology could be designed and queried.

키워드 : 상품 온톨로지, 전자 상거래, 관계형 데이터베이스 시스템, 논리적 설계
Product Ontology, e-commerce, RDBMS, Logical Design

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

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

2008년 11월 04일 접수, 2009년 02월 02일 심사완료 후 2009년 02월 09일 게재확정.

1. 서 론

상품 온톨로지 모델링은 전자 상거래 도메인에서 의미 있는 관련 개념(concept)과 개념간의 관계를 명확하게 정의하여 해당 정보의 저장과 쿼리를 구현할 수 있도록 도움을 준다[6]. 대용량의 데이터의 처리, 트랜잭션, 로킹, 보안 매커니즘 등의 신뢰성 있고 안정적인 서비스 지원 및 확장성이 가능한 RDBMS 사용은 온톨로지의 저장과 질의 처리를 위해 현실적 방안으로 받아들여지고 있다[7]. 그러나 사용자가 기대하는 상품 온톨로지적 질의 처리가 뒷받침되어야 한다. 상품 온톨로지적 질의는 상품 온톨로지에서의 개념과 개념간의 의미적 관계에 대한 질의를 의미한다.

데이터베이스 설계과정에는 데이터베이스의 엔터티(entity)와 이들간의 관계(relationship)로 표현되는 개념적 모델을 대상 데이터베이스 시스템이 지원하는 데이터베이스의 논리적 모델로 변경하는 과정이 필요한데 이를 논리적 설계(logical design)라 한다. 통상적으로 논리적 설계는 표현 저장하려는 데이터베이스 정보의 논리적 표현만 고려하는 것이 아니라, 실제 사용하는 데이터베이스 시스템이 데이터베이스를 물리적(physical)으로 어떻게 표현 저장되는지를 고려해야 효율적으로 데이터를 저장하고 질의 처리를 수행할 수 있게 된다[10].

상품 온톨로지는 도메인의 개념과 이들간의 관계가 그래프 구조로 표현될 수 있다. XML, RDF/S, OWL등과 같은 그래프 구조적인 온톨로지 정보를 RDBMS에 저장하기 위한 많은 연구가 진행되어 왔다.

본 논문에서는 그래프 구조적인 상품 온톨로지 모델을 기반으로 상품 온톨로지 질의의 효율적 처리가 가능한 RDBMS 기반의 상품 온톨로지를 위한 저장구조를 제안한다. 특히 상품 온톨로지 질의의 SQL 표현과 처리 결과를 예를 들어 소개한다. 이 논문의 구성은 다음과 같다. 제 2장에서는 관련 연구에 대해 언급하며, 제 3장에서는 상품 온톨로지 모델을 바탕으로 한 RDB 스키마 구조를 소개하고, 제 4장은 상품 온톨로지의 질의 예와 이에 대한 SQL 표현과 질의처리를 소개한다. 제 5장에서 결론을 맺는다.

2. 관련 연구

그래프 구조적인 XML 또는 RDF/S, OWL 등 온톨로지 데이터를 RDBMS에 저장하기 위한 많은 연구가 진행되어 왔다[8, 3, 13, 2, 1, 4, 11, 12]. 이 장에서는 그래프 구조의 저장에 관한 연구와 질의, 그리고 전자상거래 도메인에서의 상품정보 저장구조에 대한 특징에 대해 소개한다.

2.1 그래프 구조 정보의 저장

2.2.1 XML 저장 기법

이제까지 연구되고 있는 XML의 저장 구조는 크게 에지(edge)중심구조, 바이너리(binary) 구조, 유니버셜(universal) 구조, 인라이닝(inlining) 구조로 나뉘어진다[4, 5, 11].

Floresu와 Kossmann은 XML을 순서가 있고, 이름이 있는 방향성 그래프로 보고 RDBMS의 저장 형태를 XML 그래프의 에지에 대한 저

장 방법 3가지와 XML의 그래프의 리프(leaf) 노드에 해당하는 값(예를 들면, 스트링 값)에 대해 저장 방법 2가지로 제안하고 있다[4]. 이들이 제안하는 방법은 단순하지만, 그래프 구조적인 온톨로지 데이터인 RDF/S 혹은 OWL 등의 저장 방법 연구에 많이 참조되고 있다.

XML 문서의 모든 에지와 노드를 하나의 테이블에 저장하는 방법(수직저장 방법), 같은 이름의 에지들을 그룹화하여 에지 별로 테이블을 따로 두는 저장 방법(바이너리 저장 방법), 그리고 바이너리 저장 방법의 스키마에서 모든 에지 테이블을 외부 조인하여 생긴 결과와 같은 구조가 되는 것으로 모든 에지를 하나의 테이블에 저장하는 방법(유니버셜 방법)이 있다.

값에 대한 저장 방법 2가지 중 하나는 값 데이터의 타입을 나누어서 저장하는 분리방법이다. 즉, 스트링 타입의 값일 때와 정수 타입의 값을 서로 각각의 테이블에 저장한다. 두 번째 값 저장 방법은 인라이닝 방법으로 에지 저장 방법 중 바이너리 저장 스키마와 첫 번째 값 저장 방법을 외부 조인한 결과와 같은 스키마로 에지와 해당 값을 한 테이블에 저장하는 방법이다[11].

2.2.2 RDF/S, OWL 관련 저장 기법

RDFSuite[1]는 RDF의 파싱, 저장, 쿼리를 지원해 주는 도구이다. 저장 DB로 PostgreSQL을 사용하며, RDF의 트리플 구조를 하나의 테이블의 저장하는 '일반적 표현'(수직 저장 구조)과 네 개의 테이블에 각각 class, property, subclass, subproperty 정보를 저장하는 '특별한 표현' 스키마를 갖고 있다. Subclass와 subproperty 정보를 별도의 테이블로 저장 관리하여 분류계층에 관한 쿼리가 가능하도록 한다.

DLDB[9]는 RDF/S 저장을 위해 클래스 별로 분리된 테이블을 두고 property 테이블과 연결하는 통상 혼합방식 혹은 하이브리드(hybrid) 방식이라 불리는 저장 구조를 제안한다.

Sesame[2]의 RDB(My-SQL)의 저장 구조는 하나의 테이블에 RDF/S의 트리플 구조를 저장하고, 별도의 테이블에 저장된 class, property, subclass, literal 등을 참조하는 형태이다.

Jena[8] 는 하나의 테이블에 RDF 트리플을 저장하고, 별도의 리터럴, 리소스 테이블을 참조한다. Jena의 업그레이드 버전인 Jena2[13]는 질의 처리시의 성능개선을 위해 리소스 URI와 리터럴 값의 길이에 제한기준(threshold)을 두고 길이가 긴 경우에만 별도의 테이블에 저장하는 방식을 사용한다. 또한 군집화를 이루는 속성을 중심으로 속성 테이블 구조를 별도로 유지한다.

그 외 [12]는 RDBMS 기반의 온톨로지 데이터의 저장과 쿼리에 관한 연구로, 온톨로지를 뿌리가 있고, 방향성 있는 트리 혹은 DAG로 간주하여, 노드 테이블과 에지 테이블로 분리하여 온톨로지를 저장한다.

2.2 상품정보의 저장

Agrawal 등은 전자 상거래 데이터의 특징 고려해 볼 때, RDBMS의 전통적인 방식인 하나의 테이블에 객체 데이터(상품)를 위한 필드 하나와 모든 속성을 필드로 갖는 수평 저장보다는 상품-속성-속성값의 수직적 방식으로 저장할 것을 제안한다[2]. 전자 상거래

에서의 상품, 속성의 정보의 특징은, 상품을 설명하는 속성의 종류가 무수히 많다는 것과, 상품 클래스별로 군집화하는 경향이 있다는 것이다. 이러한 특성을 기존의 수평구조의 테이블에 저장할 때 발생하는 문제점은 다음과 같다.

- **너무 많은 칼럼 수 :** DB2나 Oracle과 같은 RDBMS는 칼럼수를 1012개로 제한하고 있으나, 전자 카타로그의 속성들을 종합해보면 수는 5000개 이상이 된다. 하나의 테이블의 칼럼 수로는 너무 많다.
- **많은 Null 값 :** 객체 데이터(상품)마다 속성이 군집화하는 특성이 있어서, 해당되지 않는 속성에 대해서는 null 값으로 채우게 된다. 그렇게 되면 저장에 있어서도 오버헤드가 발생하고, B+트리와 같은 인덱스를 사용하려 해도 고비용을 감수해야 한다.
- **스키마 변경의 어려움 :** 새로운 카타로그, 상품, 속성이 추가되는 경우, 스키마를 변경해야 하는 불합리한 점이 있다.
- **성능에서의 단점 :** 데이터 레코드 자체가 많은 필드를 갖고 있는 것은 단지 몇 개의 필드만 쿼리에 참가한다 해도 성능이 나빠질 수 있는 요인이 된다.

2.3 온톨로지 질의

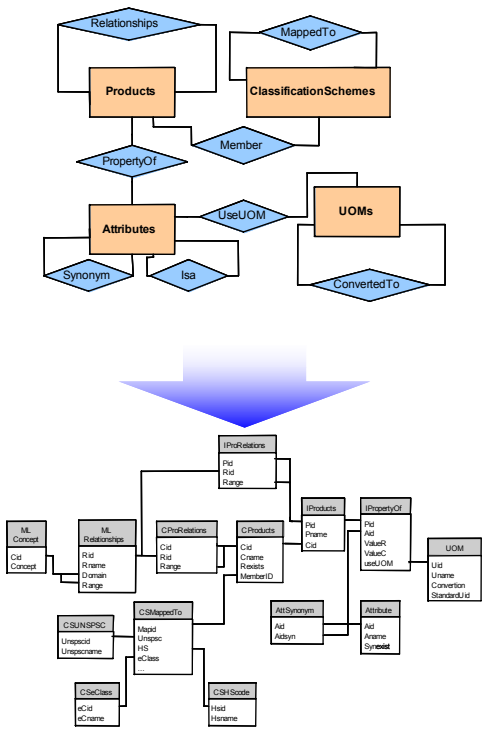
Trißl[12]은 그래프 구조적인 온톨로지 질의를 도달가능성(reachability), 조상-후손의 집합(ancestors-descendents), 최소 공통 조상(least common ancestor)의 세 종류로 분류하고 있는데, 도달가능성은 온톨로지 그래프 구조에서 두 노드(도메인의 개념)간에 길이와 상관없이 경로가 있는지를 묻는 질의이고, 조상-후손 찾기는 주어진 노드의 모든 부모 노드 혹은 자식 노드를 찾는 것과 동일하다. 포함관계 및 분류계층 구조에 관한 질의이며, 최소 공통 조상은 주어진 노드들의 공통된 기원을 찾는 질의이다.

Broekstra[2]은 RDF/S와 같은 XML 기반의 온톨로지 데이터의 질의를 문법적(syntactic) 레벨, 구조적(structure) 레벨, 의미적(semantic) 레벨에서의 질의로 분류한다. 문법적 레벨의 질의는 XML의 문법적 구조기반의 질의이기 때문에 유일한 표현식이 아닌 XML의 실제 표현되어 있는 구조 상태를 알아야 질의가 가능하다. 구조적 레벨에서의 질의는 RDF의 트리플 구조 기반의 질의로, 주어-술어-목적어 중 어느 것이든 질의의 대상이 될 수 있다. 의미적 레벨 질의는 RDF/S 그래프에서 명시된 의미뿐 아니라, 추론을 통해 얻어질 수 모든 지식을 결과로서 기대한다.

3. 상품 온톨로지 모델과 RDB 스키마

상품 온톨로지 질의 처리가 가능하기 위해서는 우선, 상품 온톨로지 정보가 RDBMS에 잘 저장되어야 하며, 이는 잘 모델링된 상품 온톨로지를 바탕으로 이루어 질 수 있다.

<그림 1>은 상품 온톨로지 모델과 이를 바탕으로 한 RDB 스키마이다. 상품 온톨로지는 개념과 개념간의 관계가 그래프로 표현될 수 있으며, 위의 상품 온톨로지 모델을 바탕으로 RDB 스키마를 설계할 수 있다.



〈그림 1〉 상품 온톨로지 모델(메타 클래스 모델)과 RDB 스키마의 한 예

상품 온톨로지 모델은 메타 모델로, 메타 클래스 레벨, 클래스 레벨, 인스턴스 레벨의 세 레벨로 나누어 볼 수 있다[7].

메타 클래스 모델은 상품 온톨로지의 주요 개념과 개념간의 관계에 대한 표현이며, 클래스 레벨은 메타 클래스 모델의 인스턴스라고도 할 수 있으며, 인스턴스레벨 데이터의 스키마를 제공한다. 예를 들면, <그림 1>의 'Products(상품)' 개념의 구체적 상품들, 즉, 하드웨어, 구두, 책상, 조명기기, 소프트웨어 등등을 의미한다. 인스턴스 레벨은 클래스 레벨의 인스턴스이며, 예를 들면 소프트웨어의 한 인스턴스로 'windowXP'가 있다(인스턴스로 인식되는 객체를 시리얼 고유 번호를 가진 'windowXP' 각각을 인스턴스로 해석할 수도 있으나, 이 논문에서는 'windowXP'를 단지 하나의 인스턴스로 간주한다. 마찬가지로 동일 디자인, 동일 브랜드, 동일 디자이너에 의해 만들어진 가방이 각각 고유번호를 갖고 있더라도 하나의 인스턴스로 간주한다. 이는 전자 상거래에서 상품 거래를 할 때, windowXP의 시리얼 번호를 보고 선택하거나, 명품가방의 고유번호를 골라서 상품을 선택하지는 않기 때문이다).

그래프 구조적인 상품 온톨로지의 효율적 처리를 위한 본 논문의 상품 온톨로지 RDB 스키마는 저장 방식에서 다음과 같은 특징이 있다.

- **(메타)클래스 레벨의 데이터와 인스턴스 데이터의 분리 저장** : 질의 유형에 따라 대용량의 인스턴스 데이터 테이블을 참조하지 않아도 된다.
- **상품 인스턴스-속성-속성값의 트리플 정보의 수직 저장 방식** : 속성의 다중값 처리가능, 속성의 변화와 추가 정보에 따른 스키마 변경에 따른 고비용의 처리를 줄인다.
- **미리 계산된 관계 정보 테이블 : 상품과 상품, 혹은 속성간의 관계,** 예를 들면 synonym, component와 같은 symmetric 혹은 transitive 특성을 갖는 의미적 관계 정보는 미리 테이블에 저장하여 질의 처리시간을 줄인다.
- **UOM간의 대표 UOM 설정과 저장 :** 서로 다른 UOM간의 관계에 대한 의미적 질의 처리의 효율성을 높인다.

4. 상품 온톨로지의 RDB 스키마기반의 질의 예제

상품 온톨로지에서의 질의는 RDB 기반 상품 온톨로지 모델에 기반해서 크게 2가지로 구분한다. 이는 인스턴스를 포함하느냐의 여부에 따른 분류이다.

- (메타)클래스 레벨 질의

상품 온톨로지 모델의 메타 클래스와 클래스 레벨에서의 질의를 의미한다.

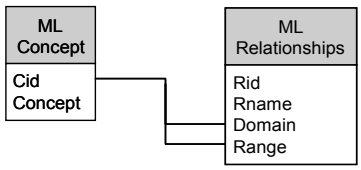
예) ◦ 상품 온톨로지의 메타 클래스 레벨에서의 개념들간에 존재하는 모든 관계를 출력하라.
◦ '데스크탑'과 'purchaseSet' 관계를 갖는 모든 상품(클래스)을 찾아라.

- 인스턴스 레벨 질의

인스턴스를 포함한 모든 질의는 인스턴스 레벨 질의를 뜻한다.

예) ◦ 무게가 1.8kg 이하이고, 모니터 크기는 12인치 이상인 노트북의 모든 인스턴스(노트북 객체)를 찾아라.
◦ HS 분류 코드가 "xxxx..."인 분류에 매핑되는 UNSPSC 코드로 분류되는 모든 개별 상품을 구하라.

<그림 2>는 상품 온톨로지 RDB 스키마의 일부분이다. MLConcept 테이블에는 메타 레벨의 개념인 상품 온톨로지의 주요 개념 정보가 저장되고, MLRelationships 테이블에는 상품 온톨로지의 모든 관계 정보를 저장한다. MLConcept에서 Cid는 주요 개념의 아이디이고 Concept 필드는 주요 개념의 이름을 의미한다. MLRelationships 테이블에서 Domain 필드와 Range필드는 MLConcept의 Cid 필드를 참조한다. Rid와 Rname 각각 관계 아이디와 관계이름을 저장하는 필드이다.



〈그림 2〉 메타 클래스 레벨 정보를 위한 상품 온톨로지의 RDB 스키마

본 논문의 질의 예제들은 RDB 스키마 기반의 상품과 상품 간의 관계 정보에 관한 상품 온톨로지 질의와 MS-Access2003 버전에서의 SQL 표현과 질의 처리 실행결과이다.

다음의 예제는 RDB스키마를 기반으로 (메타)클래스 레벨에서의 질의와 SQL 표현이다.

● (메타)클래스 레벨에서의 질의

의미 :
상품 온톨로지에서 개념들간의 모든 관계를 리스트해라

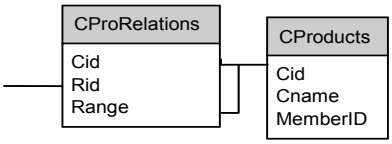
SQL :

```
SELECT mr.rname, mc.concept as domain,
mc1.concept as range
FROM Mrelationships mr, MConcepts mc,
MConcepts mc1
Where mr.domain = mc.cid and mc1.cid =
mr.range
```

결과 :

rname	domain	range
PropertyOf	Attributes	Products
Synonym	Attributes	Attributes
Isa	Attributes	Attributes
MappedTo	ClassificationSc	ClassificationScemes
Member	Products	ClassificationScemes
UseUOM	Attributes	UOM
ConvertedTo	UOM	UOM
Relationships	Products	Products

이 질의는 <그림 1>의 상품 온톨로지의 메타 클래스 레벨의 개념과 개념간의 다양한 정보를 저장하고 질의한 결과이다. 상품, 상품분류스키마, 속성, UOM이 상품 온톨로지의 주요 개념들이며, 이들간에는 PropertyOf, Synonym, Isa, MappedTo 등의 관계가 있다.



〈그림 3〉 상품 온톨로지의 상품과 상품간의 관계 정보를 저장한 RDB 스키마

<그림 3>은 상품에 관한 정보와 상품간의 관계에 대한 정보를 저장하는 RDB 스키마이다. CProducts의 Cid와 Cname 필드는 상품 클래스 아이디와 상품 클래스의 이름을 의미한다. MemberID 필드는 상품 분류 스키마를 위한 RDB 테이블과 연관 있는 필드로 자세한 설명은 생략한다. CProRelationships 테이블의 Cid, Range 필드는 CProducts 테이블의 Cid를 참조하고, Rid 필드는 MLRelationships 테이블의 Rid를 참조한다.

아래의 질의는 RDB 스키마 기반의 상품과 상품 간의 관계 정보에 관한 질의와 SQL 표현이다.

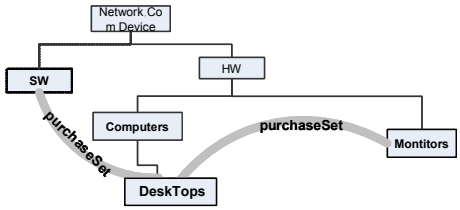
• (메타) 클래스 레벨의 질의

의미 :
'Desktops'과 'purchaseSET' 관계가 있는 모든 상품 클래스는?

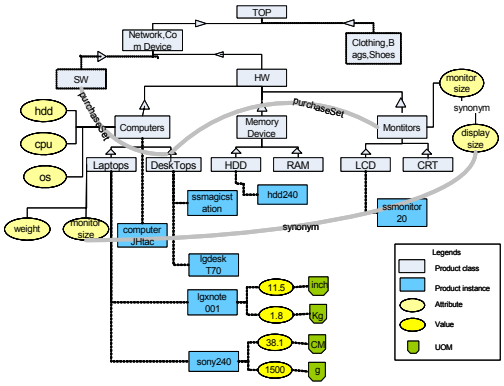
SQL :
SELECT cp.cname, mr.rname, cp2.cname
FROM CProducts AS cp, CProRelations AS cpr, MRelationships mr, CProducts cp2
WHERE cp.cname = 'Desktops' And mr.rname = 'purchaseSet' and mr.rid = cpr.rid and cp.cid = cpr.cid and cpr.range = cp2.cid

결과 :

	cp.cname	rname	cp2.cname
	Desktops	PurchaseSet	Monitors
▶	Desktops	PurchaseSet	SW



그림의 상품 온톨로지의 상품정보와 상품간의 관계 정보가 CProducts, CProRelations 에 저장된다. 온톨로지 그래프에서 DeskTops과 Monitors 클래스 그리고 DeskTops과 SW(software)가 purchaseSet 관계가 있으므로, 위의 SQL 쿼리가 처리되면 결과는 위와 같다.

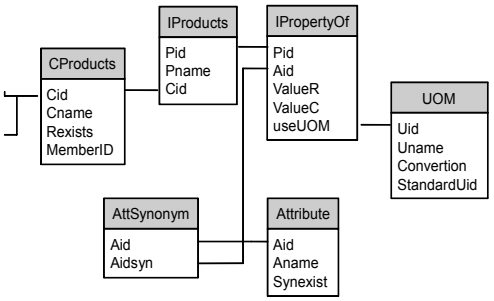


〈그림 4〉 인스턴스를 포함한 상품 온톨로지 모델의 한 예

<그림 4>는 다음의 질의 예제(인스턴스)를 위한 상품 온톨로지 모델이다. 이 논문에서 그림의 기호는 [LS05]의 EER 기반 표현에 따른다.

<그림 5>는 상품 온톨로지의 인스턴스들의 저장을 위한 RDB 스키마이다. 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.



〈그림 5〉 상품 온톨로지의 인스턴스 레벨의 데이터 저장을 위한 RDB 스키마

IPropertyOf 테이블은 각 상품 인스턴스와 관련된 속성과 속성 값, 사용 측정 단위를 함께 저장한다. 속성 값이 어떤 타입이든 간에 하나의 테이블로 저장하며, 실수(혹은 정수) 타입일 때는 ValueR 필드에 값을 저장하며, 텍스트 타입의 속성 값이면 ValueC에 값을 저장한다(실제의 저장구조는 타입에 따라 테이블을 분리하여 저장하지만, 논문에서는 설명의 간편성을 위해 하나의 테이블에 분리된 필드로 대신한다). Pid 필드는 상품 인스턴스 아이디, Aid는 속성 아이디로 Attributes 테이블을 참조한다. UseUOM은 속성과 관련한 UOM 정보를 저장하는 필드이다. 만약 '160GB'의 'xx HDD 2595'인 하드 디스크 상품을 새로운 레코드로 삽입한다고 가정할 때, 'xx HDD 2595'은 상품 인스턴스 이름이며 '160GB'의 속성값 '160'을 ValueR의 필드에 UOM 인 'GB'를 UseUOM 필드에 각각 저장한다.

Attributes 테이블은 속성 이름과 속성 아이디, synonym 관계를 갖고 있는지의 여부를 synexist 필드에 저장한다. AttSynonym 테이블은 속성간의 synonym 관계를 저장한다. Aid, AidSyn은 모두 Attributes 테이블의 속성아이디를 참조한다. <그림 6>은 Attributes 테이블과 AttriSynonym 테이블의 스냅샷이다. 'monitorsize'와 'displaysize'속성이 synonym관계라고 할 때, 이에 대한 정보가 Synonym 테이블에 저장된 모습이다. synonym 관계가 symmetric한 특성이 있으므로, 임의의 레코드(x, y)가 저장되면 반드시 (y, x)도 저장되어야 한다.

	Aid	Aname	Synexist
+	8	weight	1
+	7	monitorsize	1
+	6	displaysize	1

AID	AIDsyn
6	7
7	6

〈그림 6〉 Attribute 테이블과 AttSynonym 테이블 스냅샷

UOM 테이블은 속성에 사용되는 측정 단위를 저장한다. UID는 UOM 아이디, Uname은 UOM 이름 즉, 'cm(CM)', 'inch', 'kg' 등을 의미한다. standardUID는 표준 단위로 무게를 측정하는 단위, 길이를 측정하는 단위 등 단위 그룹별로 대표 단위를 정해 놓은 것으로 도메인 전문가에 의해 지정될 수 있다. <그림 7>에서 UID = 0인 M(미터)의 표준단위는 UID = 2인 CM(센티미터)로 정의되어 있음을 볼 수 있다. 길이에 관련된 측정단위들인 '미터', '센티미터', '인치'등은 'CM'를 표준단위로 정하고, 마찬가지로 컴퓨터 관련 단위

중 용량은 'MB'를 표준 단위로 정할 수 있다.

conversion필드는 해당 UOM이 표준 단위로 변환할 때 환산되는 단위 값을 저장한다. 1CM(센티미터) = 0.01M(미터)이므로 첫 번째 레코드의 conversion 값은 0.01이다. Conversion 필드는 질의에서 입력되는 측정 단위가 무엇이건 간에 상관없이 비교 연산이 가능하도록 한 것으로, 질의 처리 과정에서 모든 값은 표준 단위로 환산된 값을 연산에 적용함으로써, 서로 다른 측정단위를 갖는 두 상품의 속성 값에 대한 비교나 연산이 가능해 진다 Uname = 'CM'은 자기 자신이 표준 단위이므로 conversion 값은 '1'이다. 결국, conversion 필드는 convertedTo 관계처리에 필요한 선형 방정식연산을 할 수 있도록 마련한 필드이다.

예를 들어 인스턴스 레벨의 질의 2에서는 1.8kg 이하의 무게에 대한SQL 표현으로는 표준단위를 사용하여, '*u.conversion×ipo.valueR <= 1.8×u1.conversion*'의 식을 써서 환산된 값이 연산 비교가 된다.

<그림 7>에서 UID = 7인 kg의 standardUID가 kg이므로, conversion 값은 1이며, '1×1.8<=1.8×1의 식이 처리과정에 사용된다. 만약, 1500g의 무게를 속성으로 갖는 노트북이라면 g의 conversion값 0.001을 참조하여, 0.001×1500<=1.8×1이 계산된다.

	UID	Uname	Conversion	standardUID
	0	M	0.01	2
	1	mm	10	2
	2	CM	1	2
	3	inch	0.3937	2
	4	MH	1024	4
	5	GB	1	6
	6	MB	1	6
	7	kg	1	7
	8	g	0.001	7

〈그림 7〉 UOM 테이블의 스냅샷

아래의 질의는 인스턴스를 포함한 인스턴스 레벨의 질의이다.

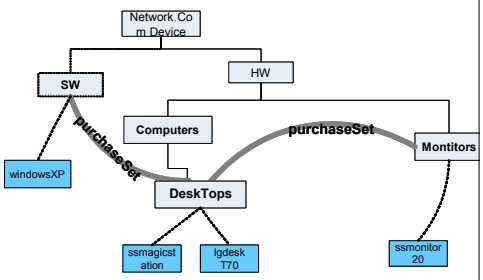
• 인스턴스 레벨의 질의 1

의미 :
'Desktops'과 'purchaseSET'관계인 상품 클래스의 모든 인스턴스?

SQL :
SELECT distinct cp.cname, ip.pname, pr.rname, cp2.cname, ip2.pname
FROM CProducts AS cp, CProRelations AS cpr, PRelationships AS pr, CProducts AS cp2, IProducts ip2, IProducts ip
WHERE cp.cname = 'Desktops' And pr.rname = 'purchaseSET' And cpr.rid = pr.rid And cp.cid = cpr.cid And cpr.range = cp2.cid and cp2.cid = ip2.cid and cp.cid = ip.cid;

Answer :

cp.cname	ip.pname	rname	cp2.cname	ip2.pname
Desktops	lgdeskT70	PurchaseSet	Monitors	ssmonitor20
Desktops	lgdeskT70	PurchaseSet	SW	windowsXP
Desktops	ssmagicstation	PurchaseSet	Monitors	ssmonitor20
Desktops	ssmagicstation	PurchaseSet	SW	windowsXP



이 질의 예제는 상품 클래스 DeskTops와 Monitors 또는 DeskTops과 SW(software)간에 도메인의 의미적 관계인 purchaseSet 관계가 있음을 찾아내고, 각각의 해당 인스턴스를 구한다.

• 인스턴스 레벨의 질의 2

의미 :
무게가 1.8kg 이하이고, 모니터 크기가 12인치 이상인 노트북의 모든 인스턴스?

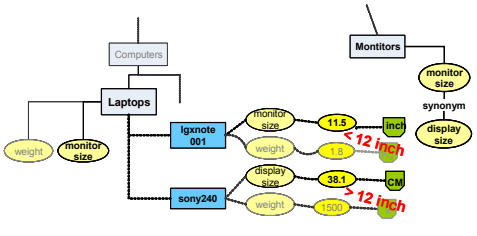
SQL :

SELECT cp.cname, ip.pname, att.aname, ipo.valueR, u.uname
FROM CProducts AS cp, IProducts AS ip, Attribute AS att, IPropertyOf AS ipo, UOM AS u, UOM u1
*WHERE cp.cname = 'laptops' And ip.cid = cp.cid And att.aname = 'weight' and ip.pid = ipo.pid and att.aid = ipo.aid and ipo.useUOM = u.uid and u.conversion*ipo.valueR< = 1.8*u1.conversion and u1.uname = 'kg'*

INTERSECT

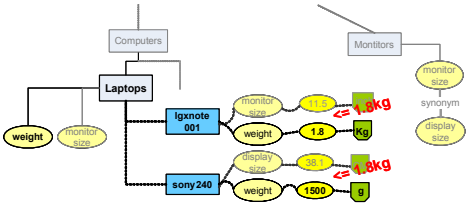
SELECT cp.cname, ip.pname, att1.aname, ipo.valueR, u.uname
FROM Attribute AS att, AttSynonym AS asyn, Attribute AS att1, CProducts AS cp, IProducts AS ip, IPropertyOf AS ipo, UOM AS u, UOM u1
*WHERE att.aname = 'displaysize' And att.aid = asyn.aid And asyn.aidsyn = att1.aid And att1.aid = ipo.aid And ip.pid = ipo.pid And cp.cname = 'laptops' And ip.cid = cp.cid And u.conversion*ipo.valueR> = 12 *u1.conversion And u.uid = ipo.useUOM and u1.uname = 'inch';*

중간 결과 :



cname	pname	aname	valueR	uname
Laptops	sony240	monitorsize	38.1	CM

INTERSECT



cname	pname	aname	valueR	uname
Laptops	lgxnote001	weight	1.8	kg
Laptops	sony240	weight	1500	g

결과 :

(Laptops, Sony240)

이 예제는 서로 다른 UOM을 사용하고, 서로 다른 속성을 갖는 두 상품의 속성값에 대해 UOM 간의 convertedTo 관계, 속성간의 synonym(유사) 관계를 고려하여, 의미적 질의를 처리하는 것이다. 질의의 의미를 두 부분으로 나눌 수 있다.

첫 번째 부분은 1.8kg 이하의 노트북을 찾는 것이고, 두 번째 부분은 12인치 이상의 모니터 크기인 노트북을 찾아서 두 조건을 동시에 만족하는 것을 결과로 되돌려 주어야 한다. 상품 클래스와 상품 인스턴스, 속성, UOM이 질의에 모두 포함되므로 속성간의 관계(여기서는 synonym)와 UOM 간의 관계(convertedTo)를 고려해야 한다. 1.8이하, 12이상과 같은 비교연산이 가능해야 하므로 1.8, 12의 실수(정수) 값과 kg, '인치'와 같은 UOM은 분리된 필드(분리된 테이블)에 저장되어야 하고 이미 분리하여 저장했다.

속성에서 '무게'는 '노트북 무게', 'weight'등 같은 의미의 다양한 표현이 가능하고, 마찬가지로, '모니터 크기'는 'display size' 'monitor size'등으로 표현되어 사용되고 있을 수 있으므로, 속성간의 synonym 관계에 대한 정보를 AttriSynoym 테이블에서 얻는다. 'kg'이나 '인치'는 'g'이나 'CM' 처럼 측정단위를 달리 사용하고 있을 수 있으므로, UOM 간의 convertedTo 관계를 고려하여 이를 질의 처리 과정에 반영해야 하며, 이것은 conversion 필드가 그 역할을 하고 있다. 실제 질의자가 어떤 단위 체계로 입력을 하든, DB에 어떤 측정 단위로 저장되어 있든 간에 상관없이 처리될 수 있다.

5. 결 론

온톨로지 저장 및 관리 시스템에 요구되는 사항은 대용량 온톨로지 데이터의 확장성과 효율적인 온톨로지 질의의 처리능력이다. 본

논문은 그래프 구조적인 상품 온톨로지 모델을 기반으로 온톨로지 의 저장과 질의 처리가 RDBMS에서 효과적으로 이루어질 수 있도록 데이터베이스 저장 설계 방안을 제안하였다.

상품 온톨로지 모델에 기반한 저장구조는 대용량의 상품-속성-속성값 정보의 확장성을 고려하여 수직적 저장방식을 제안하고, 클래스-인스턴스 정보의 분리 저장, 상품 정보 간의 관계의 미리 계산된 정보 저장, 대표 UOM 설정 저장 등의 방법을 통해 상품 온톨로지 질의 처리가 효율적으로 이루어 지도록 하고 있다. 본 문에서의 질의 예는 상품 온톨로지 질의, SQL 표현, 질의 처리 실행 결과를 제시함으로써 제안한 저장구조의 효과와 효율성을 보여준다.

참 고 문 헌

[1] Sofia Alexaki, Vassilis Christophides, Gregory Karvounarakis, Dimitris Plexousakis, Karsten Tolle, The ICS-FORTH RDFSuite : Managing Voluminous RDF Description Bases, SemWeb, 2001.

[2] Rakesh Agrawal, Amit Somani, Yirong Xu, Storage and Querying of E-Commerce Data, VLDB, 2001.

[3] Jeen Broekstra, Arjohn Kampman, Frank van Harmelen, Sesame : A generic Architecture for Storing and Querying RDF and RDF Schema, International Semantic Web Conference, 2002.

[4] Daniela Florescu, Donald Kossmann, Storing and Querying XML Data using an RDMBS, IEEE Data Eng. Bull, Vol. 22, No. 3, 1999.

[5] Anuradha Gali, Cindy X. Chen, Kajal T. Claypool, Rosario Uceda-Sosa, From Ontology to Relational Databases, ER (Workshops), Vol. 4, 2004.

[6] Hyunja Lee and Junho Shim, Conceptual and Formal Ontology Model of e-Catalogs, EC-Web, 2005.

[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] Brian McBride, Jena : A semantic web toolkit, IEEE Internet Computing, 2002.

[9] Zhengxiang Pan, Jeff Heflin, DLDB : Extending Relational Databases to Support Semantic Web Queries, PSSS, 2003.

[10] Raghu Ramakrishnan, Johannes Gehrke, Database Management Systems, 3rd ed., McGraw-Hill, 2003.

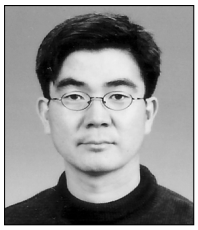
[11] Jayavel Shanmugasundaram, Kristin Tufte, Chun Zhang, Gang He, David J. DeWitt, Jeffrey F. Naughton, Relational databases for querying XML documents : Limitations and opportunities. VLDB, 1999.

[12] Silke Trißl, Ulf Leser, Querying Ontologies in Relational Database Systems, DILS, 2005.

[13] Kevin Wilkinson, Craig Sayers, Harumi A. Kuno, Dave Reynolds, Efficient RDF Storage and Retrieval in Jena2, SWDB, 2003.

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 & Computer Eng, (공학박사)
1999년~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