디지털산업정보학회 논문지 제7권 제3호-2011년 9월

효율적인 온톨로지 추론 질의를 지원하는 OWL 저장 모델

김 연 희* · 이 애 정**

OWL Storage Model to Support Efficient Ontology Reasoning Query

Kim, Youn Hee · Lee, Ae Jung

〈Abstract〉

In the Semantic Web, storage models are required to efficiently store and retrieve metadata and ontology represented using OWL that can provide expressive power and reasoning support. In this paper, we propose an OWL storage model that can store and retrieve many restrictions and semantic relations defined on ontology with metadata. In addition, we propose some methods and rules to improve query processing efficiency of the proposed storage model. The proposed storage model can store and process large amounts of ontology and metadata because it consists of tables based on the relational database. And the proposed model can quickly provide more accurate results to users because of performing two different types of ontology reasoning and using the prime number labeling scheme to easily identify hierarchy relationships between classes or properties. The comparative evaluation results show that our storage model provides better performance than the existing storage model.

Key Words : OWL, Storage Model, Ontology Reasoning, Query Processing

Ⅰ. 서론

시맨틱 웹은 기존 웹의 여러 한계점을 극복하고 보다 정확하면서 지능적인 정보 검색 기능을 제공하기 위해 제시된 것이다[1]. 시맨틱 웹에서는 모든 정보 자원을 URI(Uniform Resource Identifier)로 식별한다. 그리고 메타데이터와 온톨로지를 이용해 정보 자원의 의미를 개념적으로 정의하고 정보 자원들 간의 다양한 의미적 연관성을 표현함으로써 사람뿐만 아니라 컴퓨터도 정보 자원의 의미를 이해하고 해석할 수 있도록 한다. 온톨로지는 메타데이터를 정형화된 형태로 기술하기 위해 필요한 개념과 개념들 간의 의미적 관계를 정의하는 역할을 담당한다. 메타데이터는 온톨로지에 정의된 개념을 이용해 실제 정보 자원의 의미와 정보 자원들 간의 의미적 연관성을 기술한 것이다. 시맨틱 웹에서는 온톨로지에 대한 추론을 통해 메타데이터에 직접 기술되지는 않았지만 숨겨져 있는 새로운 정보를 유도해낼 수 있기 때문에 보다 정확하고 사용자 요구에 적합한 정보 검색이 가능하다. 따라서 정보 자원에 대한 의미 있는 해석과 추론에 기반을 둔 지능적인 정보 검색 기능을 실현하기 위해서는 메

* 부천대학교 e-비즈니스과 강의전담교수
** 한양여자대학교 정보경영과 교수(교신저자)

타데이터와 온톨로지를 컴퓨터가 이해할 수 있는 정형화된 형태로 표현하고 관리하는 것이 중요하다.

메타데이터와 온톨로지를 정형화된 형태로 기술하기 위해 RDF(Resource Description Framework), RDF 스키마, OWL(Web Ontology Language) 등의 다양한 언어들이 제안되었다. 그 중 다른 언어들에 비해 풍부한 표현력과 모델링 요소를 가지고 있는 OWL이 W3C의 권고안으로 채택되었고 현재 OWL 2.0까지 발표되었다[2]. OWL은 개념과 개념들 간의 다양한 관계를 자세하게 정의할 수 있기 때문에 보다 강력한 추론 기능을 제공할 수 있고 메타데이터와 온톨로지를 함께 기술할 수 있다는 장점이 있다. 따라서 OWL의 장점을 그대로 수용하면서 OWL로 기술된 메타데이터와 온톨로지를 효율적으로 저장하고 검색할 수 있는 저장 모델의 필요성이 대두되었다. 특히 다양한 분야에서 OWL로 기술된 메타데이터와 온톨로지가 사용되고 그 크기가 대용량화되면서 관계형 데이터베이스에 기반을 둔 저장 모델에 대한 관심이 커지고 있다. 하지만 관계형 데이터베이스에 기반을 둔 기존 OWL 저장 모델들은 온톨로지 추론을 이용한 질의를 고려하지 않거나 비효율적으로 처리하는 경우가 많아 OWL의 특징이 완벽하게 반영되지 못하는 문제점을 가지고 있다.

따라서 본 논문에서는 OWL로 기술된 온톨로지와 메타데이터를 각각의 특성에 따라 저장하고 추론을 통해 유도되는 새로운 정보까지 고려하는 확장된 질의의 효율적인 처리를 지원하는 저장 모델을 제안한다.

본 논문은 다음과 같이 구성된다. 2장에서는 대표적인 기존 OWL 저장 모델의 특징과 문제점을 제시한다. 3장에서는 온톨로지 추론 질의의 효율적인 처리를 지원하기 위해 본 논문에서 제안한 OWL 저장 모델을 소개한다. 그리고 4장에서는 실험을 통해 본 논문에서 제안한 OWL 저장 모델의 성능을 평가한 결과를 제시하고 5장에서 결론을 맺는다.

Ⅱ. 관련연구

OWL로 기술된 대용량의 온톨로지와 메타데이터를 저장하고 검색하기 위해 관계형 데이터베이스를 이용하는 시스템이 지금까지 많이 소개되었다. 초기의 시스템들은 온톨로지와 메타데이터의 특성을 구별하지 않고 주어(subject)-서술어(predicate)-목적어(object)의 트리플 구조로 구성된 하나의 테이블에 모든 내용을 저장했다. 하지만 그러한 저장 방식은 질의 처리 시 많은 투플을 포함하는 하나의 테이블을 대상으로 여러 번 자기 조인이 발생하기 때문에 비효율적인 문제점이 있다. 그러한 문제점을 해결하고 온톨로지와 메타데이터의 특성을 반영하면서 검색 성능을 향상시키기 위해 온톨로지에 정의된 내용에 따라 테이블을 분해하여 정규화된 저장 스키마를 설계하고 그에 따라 메타데이터의 내용을 저장하는 다양한 시스템들이 제안되었다. 이들 중에서 본 논문의 목적과 부합하는 대표적인 시스템으로 DLDB[3], XPOS[4], ONTOMS[5], OntoRDB[6] 등이 있다.

DLDB[3]는 온톨로지에 정의된 모든 개념(클래스)과 개념들 간의 관계(프로퍼티)를 각각 별도의 테이블로 구성하고 메타데이터로 기술된 관련 내용을 테이블에 저장한다. 그리고 별도의 추론기를 이용해서 온톨로지에 정의된 클래스나 프로퍼티의 계층 구조에 기반을 둔 추론 기능을 지원한다. 하지만 추론기를 이용하여 질의 처리 시 추론 기능을 제공하기 때문에 검색 시간이 오래 걸리고 대용량 데이터에 대한 효율적인 추론이 이루어지지 못한다. 그리고 DLDB는 OWL의 풍부한 표현력을 이용해 온톨로지에 정의할 수 있는 다양한 제약 사항이나 의미적 관계 등은 고려하지 않기 때문에 이러한 정보에 대한 검색과 추론이 가능하지 않은 문제점이 있다.

XPOS[4]는 클래스, 프로퍼티, 트리플, 인스턴스의 4개 테이블로 구성된다. XPOS는 클래스나 프로퍼티의 계층 구조를 이용하는 질의를 효율적으로 처리하기 위해 추론기를 이용하지 않고 클래스나 프로퍼티의 계층 구조를 XPath 형태로 표현하여 미리 테이블에 저장한다. 하지만

DLDB와 마찬가지로 XPOS도 클래스와 프로퍼티의 기본적인 정의 내용과 계층 구조 외에 OWL로 표현이 가능한 다양한 제약 사항과 의미적 관계는 고려하지 않는 문제점을 가지고 있다.

ONTOMS[5]는 클래스나 프로퍼티의 계층 구조는 물론 OWL로 표현 가능한 프로퍼티의 역관계(inverse), 대칭관계(symmetric), 이행관계(transitive)를 이용한 추론 기능까지 지원한다. ONTOMS는 온톨로지와 메타데이터에 직접 기술되어 있는 정보와 함께 추론을 통해 유도된 새로운 정보를 테이블에 미리 저장해둠으로써 질의 처리 시간을 줄이고 검색의 정확도를 향상시키는 장점이 있다. 그러나 ONTOMS는 온톨로지에 정의된 모든 클래스와 여러 개의 값을 가지는 모든 프로퍼티마다 별도의 테이블을 구성하기 때문에 너무 많은 개수의 테이블이 필요한 문제가 있다. 그리고 여러 개의 값을 가지는 프로퍼티인지를 확인하기 위해 테이블을 구성하기 전에 메타데이터의 모든 내용을 살펴봐야 하는 부담이 있다. 또한 클래스를 정의할 때 사용되는 unionOf, intersectionOf, complementOf, oneOf, Restriction 등의 제약 사항이나 클래스의 인스턴스인 개체들 간의 관계를 표현하는 sameAs, differentFrom 등과 같이 OWL로 표현할 수 있는 클래스나 개체들의 다양한 의미 관계 자체에 대한 검색이나 그것을 이용한 추론이 가능하지 않다.

OntoRDB[6]는 9개의 테이블로 구성되고 메타데이터에 기술된 내용과 함께 OWL의 풍부한 표현력을 이용해 온톨로지에 정의된 대부분의 제약 사항과 의미적 연관관계를 모두 저장한다. 따라서 온톨로지에 정의된 내용 자체에 대한 질의는 물론 온톨로지 추론을 통해 새롭게 유도된 메타데이터 내용에 대한 질의도 처리가 가능하다. 하지만 질의 처리를 할 때마다 테이블에 저장된 내용을 토대로 추론이 이루어지기 때문에 추론된 내용을 미리 테이블에 저장해두는 방식에 비해 질의 처리 시간이 오래 걸린다. 그리고 추론을 위해서는 여러 테이블에 대한 조인이 필요하기 때문에 질의 내용을 SQL 언어로 표현하는 것이 복잡해질 수 있다.

본 논문에서는 관계형 데이터베이스를 이용한 기존 OWL 저장 시스템들의 장점을 극대화하고 앞서 제시된 문제점을 해결할 수 있는 새로운 OWL 저장 모델을 제안한다.

Ⅲ. OWL 저장 모델

본 논문에서는 OWL로 표현할 수 있는 온톨로지의 모든 정의 내용과 메타데이터에 직접 기술된 내용은 물론 온톨로지 추론을 통해 유도되는 새로운 메타데이터 정보를 함께 저장하여 검색의 효율성을 높일 수 있는 OWL 저장 모델을 제안한다. 특히, 본 논문에서는 온톨로지 추론을 통해 얻어지는 새로운 메타데이터 정보를 미리 테이블에 저장하여 사용자 요구에 부합하는 정확한 검색 결과를 빠르게 제공하고자 한다.

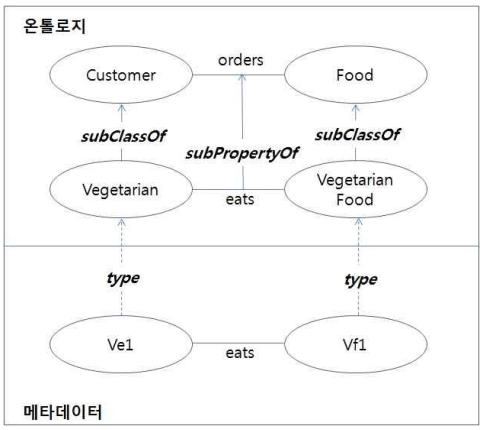
본 절에서는 본 논문에서 제안한 OWL 저장 모델을 소개하고 제안한 저장 모델과 관련한 레이블링 기법과 추론 및 저장 규칙들을 설명한다.

3.1 추론 규칙

질의를 처리할 때 온톨로지 추론을 실시간으로 수행하는 것 보다는 미리 처리하여 그 결과를 테이블에 저장한 후 이용하는 것이 효율적이다. 그러나 가능한 모든 추론의 결과를 테이블에 미리 저장하게 되면 질의를 처리할 때 너무 많은 데이터를 다루어야 하기 때문에 오히려 성능을 저하시킬 수 있다. 따라서 본 논문에서는 가능한 온톨로지 추론의 유형을 질의 처리 시에 수행해야 하는 것과 미리 수행해야 하는 것으로 구분하여 처리한다. 추론 유형을 분류하는 기준은 처리의 복잡도이다. 추론 처리가 상대적으로 복잡하고 많은 양의 데이터를 대상으로 하는 경우는 미리 처리하여 테이블에 그 결과를 저장해 둠으로써 질의 처리 시간을 줄일 수 있다.

<그림 1>은 OWL로 기술된 레스토랑 관련 온톨로지

와 메타데이터의 간단한 예를 그래프 모델의 형태로 표현한 것으로 이후 본 논문에서 제안한 추론 규칙을 설명하기 위해 사용한다.



<그림 1> 레스토랑에 대한 온톨로지와 메타데이터 예

3.1.1 질의 처리 시 수행하는 추론 규칙

사용자의 질의를 처리하는 과정에서 수행하게 되는 온톨로지 추론은 세부적으로 다음과 같이 3가지의 유형으로 분류된다.

(1) 타입 전달

<그림 1>에서 "Vegetarian" 클래스는 "Customer" 클래스의 하위 클래스(subClassOf)이고 정보 자원인 "Ve1"은 "Vegetarian" 클래스 타입의 개체(인스턴스)로 정의되어 있다. 따라서 직접적으로 선언된 것은 아니지만 클래스 계층 관계를 통해 "Ve1"가 "Customer" 클래스 타입의 개체라는 것을 추론할 수 있기 때문에 사용자가 "Customer" 클래스 타입의 개체를 검색하고자 하면 "Ve1"이 결과로 반환되어야 한다. 본 논문에서는 이와 같이 하위 클래스 타입으로 선언된 개체가 상위 클래스 타입으로 추론되는 것을 계층 구조에 따른 타입 전달이라고 정의한다.

OWL에서는 다른 클래스들의 합집합(unionOf)과 교집합(intersectionOf) 관계를 이용해 새로운 클래스를 정의할 수 있고 이러한 클래스간의 합집합과 교집합 관계는 계층 관계로 이해할 수 있다[7]. 예를 들어 "Person" 클래스가 "Man" 클래스와 "Woman" 클래스의 합집합으로 정의되어 있다면 "Man" 클래스와 "Woman" 클래스는 "Person" 클래스의 하위 클래스로 추론할 수 있고 일반적인 계층 구조와 마찬가지로 타입 전달이 가능하다. 본 논문에서는 클래스들 간의 합집합과 교집합의 관계를 일반적인 계층 관계로 변환하여 질의 처리 시 타입 전달을 수행한다.

본 논문에서는 동등 관계(equivalentClass)로 정의된 클래스들에 대해서도 타입 전달을 수행한다.

(2) 타입 추론

클래스 타입이 정의되어 있지 않고 타입 전달도 가능하지 않은 개체에 대해 클래스 타입을 결정해야 하는 경우 OWL의 풍부한 표현력을 바탕으로 온톨로지에 정의된 다양한 내용을 이용해 타입을 추론할 수 있다. 본 논문에서는 온톨로지에 정의된 프로퍼티의 도메인(domain) 및 레인지(range) 클래스와 oneOf, hasValue, allValuesFrom을 이용해 정의된 제한 사항을 이용하여 개체의 타입을 추론한다.

예를 들어 <그림 1>에서 "eats" 프로퍼티의 도메인(domain) 클래스는 "Vegetarian"이고 레인지(range) 클래스는 "Food"로 정의되어 있다. 따라서 특별히 타입이 선언되지 않은 "A" 개체가 "B" 개체와 "eats" 프로퍼티 관계를 맺고 있다면 "A" 개체는 "Vegetarian" 클래스 타입으로, "B" 개체는 "Food" 클래스 타입으로 추론한다.

또 다른 예로 "VegetarianFood" 클래스가 oneOf를 이용해 "mushroom1"과 "tofu1" 개체만을 멤버로 가지도록 온톨로지에 정의되어 있다면 "mushroom1"과 "tofu1" 개체는 모두 "VegetarianFood" 클래스 타입으로 추론한다.

(3) 관계 전달

<그림 1>에서 “eats” 프로퍼티는 “orders” 프로퍼티의 하위 프로퍼티(subPropertyOf)이고 “Ve1” 개체는 “Vf1” 개체와 “eats” 관계를 맺고 있다. 따라서 직접적으로 선언된 것은 아니지만 프로퍼티 계층 관계를 통해 “Ve1” 개체는 “Vf1” 개체와 “orders” 관계도 맺고 있음을 추론할 수 있다. 본 논문에서는 이와 같이 하위 프로퍼티의 관계를 맺고 있는 개체들 간에 상위 프로퍼티의 관계가 전달되는 것을 관계 전달이라고 정의한다.

본 논문에서는 동등 관계(equivalentProperty)로 정의된 프로퍼티들에 대해서도 타입 전달을 수행한다.

3.1.2 질의 처리 전 미리 수행하는 추론 규칙

테이블을 구성하는 과정에서 미리 수행하여 그 결과를 저장하는 온톨로지 추론은 세부적으로 다음과 같이 2가지의 유형으로 분류된다.

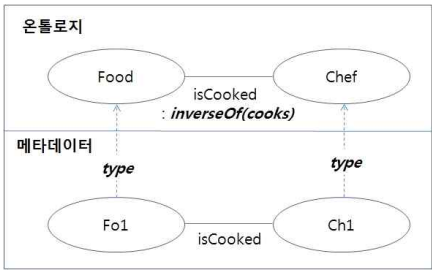
(1) 관계 추론

본 논문에서는 온톨로지에 정의된 프로퍼티의 역관계(inverse), 대칭관계(symmetric), 이행관계(transitive)를 이용하여 개체들 간의 새로운 관계를 추론한다.

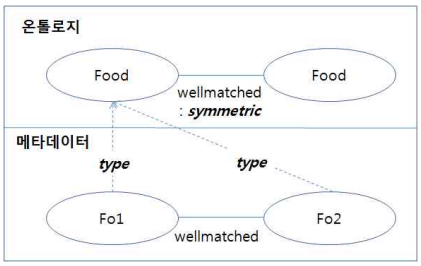
<그림 2>에서 “isCooked” 프로퍼티는 “cooks” 프로퍼티와 역관계로 정의되어 있다. 따라서 메타데이터에 “Fo1” 개체가 “Ch1” 개체와 “isCooked” 관계를 맺고 있다는 (Fo1 isCooked Ch1)의 트리플 문장이 기술되어 있다면 “Ch1” 개체와 “Fo1” 개체가 “cooks” 관계를 맺고 있음을 의미하는 숨겨진 (Ch11 cooks Fo1) 트리플 문장을 추론한다.

<그림 3>에서 “wellmatched”는 대칭 프로퍼티로 정의되어 있다. 따라서 메타데이터에 (Fo1 wellmatched Fo2) 트리플 문장이 기술되어 있다면 숨겨진 (Fo2 wellmatched Fo1) 트리플 문장을 추론한다.

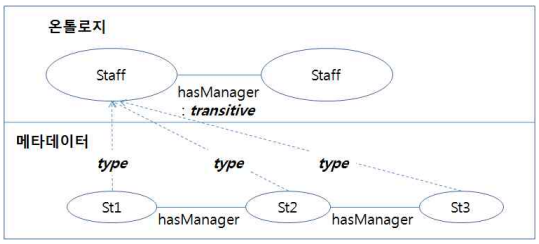
<그림 4>에서 “hasManager”는 이행적 프로퍼티로 정의되어 있다. 따라서 메타데이터에 직접 기술된 두 개의 트리플 문장 (St1 hasManager St2)와 (St2 hasManager St3)을 통해 숨겨진 (St1 hasManager St3) 트리플 문장을 추론한다.



<그림 2> 프로퍼티의 역관계 예



<그림 3> 프로퍼티의 대칭관계 예



<그림 4> 프로퍼티의 이행관계 예

그리고 본 논문에서는 hasValue 제한을 통해 새로운 프로퍼티 관계를 추론한다. 예를 들어 온톨로지에서 “A” 클래스를 정의할 때, “A” 클래스의 모든 개체는 “v1” 개체와 “p” 프로퍼티 관계를 맺어야 한다는 것을 hasValue로 제한하였다면 메타데이터에 직접 기술되어 있지 않아도 “A” 클래스 타입의 모든 개체는 “v1” 개체와 “p” 프로퍼티 관계를 맺고 있다는 것을 추론한다.

(2) 개체의 동일성과 이질성 추론

본 논문에서는 직접적으로 기술되어 있지 않더라도 온톨로지에 정의된 다른 내용을 바탕으로 개체간의 동일성(sameAs)이나 이질성(differentFrom)을 판단한다. 개체의 동일성은 함수적(functional) 프로퍼티, 역함수적(inversefunctional) 프로퍼티를 통해 추론할 수 있다. 그리고 개체의 이질성은 클래스간의 서로소(disjointWith) 관계를 통해 추론할 수 있다.

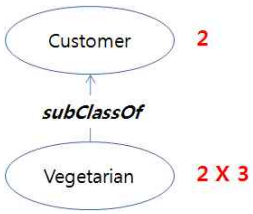
예를 들어 특정 개체에 대해서 반드시 하나의 개체와만 관계를 맺을 수 있는 함수적 프로퍼티로 선언된 "p" 프로퍼티에 대해 메타데이터에 개체 "x1"와 관련해 (x1 p a1)과 (x1 p b1) 트리플 문장이 기술되어 있다면 "a1"과 "b1"는 동일한 개체로 판단한다.

또 다른 예로 "Man" 클래스와 "Woman" 클래스가 disjointWith 관계로 정의되어 있다면 "Man" 클래스의 개체와 "Woman" 클래스의 개체는 서로 다른 이질적인 개체로 판단한다.

3.2 레이블링 기법을 이용한 계층 구조 표현

본 논문에서는 온톨로지 추론에서 가장 중요하고 빈번하게 다루어지는 클래스나 프로퍼티의 계층 관계를 빠르게 판단하기 위해 소수 레이블링 기법을 이용하여 추론에 소요되는 시간을 줄이고자 한다.

소수 레이블링은 XML이나 RDF 스키마에서 계층 구조를 표현하기 위해 사용된 기법으로 계층 구조를 트리의 형태로 표현하고 각각의 노드에 고유한 소수 값을 부여한 후 부모의 소수 레이블 값에 부여받은 소수 값을 곱하여 자신의 레이블로 결정하는 방법이다[8]. 이 방법은 두 노드의 소수 레이블 값으로 나누기 연산을 수행하였을 때 그 나머지가 0이면 두 노드가 계층 관계를 맺고 있다는 것을 빠르게 판단할 수 있다. 예를 들어 <그림 5>에서 "Vegetarian" 클래스는 자신에게 부여된 소수값 3에 부모인 "Customer" 클래스의 소수 레이블인 2를 곱한 결과인 6이 자신의 소수 레이블이 된다.



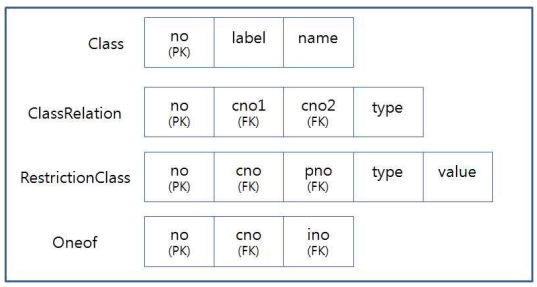
<그림 5> 소수 레이블 부여 예

본 논문에서는 동등 관계인 클래스들에 대해 같은 소수 레이블을 부여함으로써 동등 관계에 대한 추론 질의 처리를 보다 빠르게 처리하도록 한다. 동등 관계의 프로퍼티들에 대해서도 같은 소수 레이블을 부여하여 동등 관계임을 빠르게 판단할 수 있도록 한다.

3.3 저장 스키마 설계

본 논문에서 제안한 OWL 저장 모델은 11개의 테이블로 구성되어 있다. 11개의 테이블 중 7개의 테이블은 온톨로지에 정의된 내용을 저장하고 4개의 테이블은 메타데이터에 기술된 내용을 저장한다.

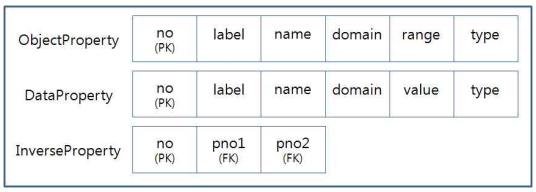
<그림 6>은 본 논문에서 제안한 OWL 저장 모델에서 온톨로지에 기술된 클래스의 정의와 클래스간의 관계 정의 내용을 저장하는 4개 테이블들 스키마를 보여준다. 본 논문에서 제안한 OWL 저장 모델의 모든 테이블은 일련번호를 기본키로 사용한다.



<그림 6> 클래스 정의 내용을 저장하기 위한 테이블 스키마

<그림 6>에서 Class 테이블은 OWL에 정의된 모든 클

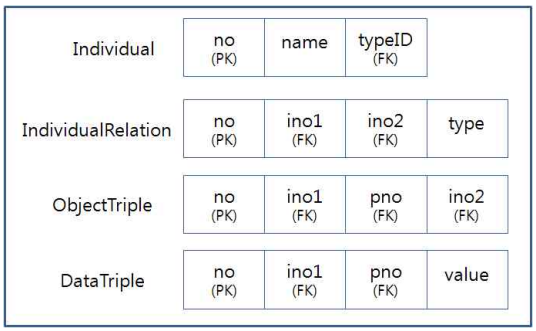
래스의 소수 레이블과 이름 정보를 저장한다. 그리고 ClassRelation 테이블은 두 클래스간의 관계 정보를 저장하기 위해 Class 테이블의 기본키를 외래키로 참조하는 cno1과 cno2 필드를 포함한다. 그리고 type 필드에서 클래스간의 관계를 union, intersection, complement, disjoint 타입으로 구별하여 표현한다. RestrictionClass 테이블은 프로퍼티에 대한 제한 사항을 가지는 클래스의 정보를 저장하기 위해 Class 테이블의 기본키와 Property 테이블의 기본키를 외래키로 참조한다. 그리고 제한 사항은 allValue, someValue, hasValue, cardinality, maxcardinality, mincardinality 타입으로 구별하여 type 필드에 표현하고 제한된 값은 value 필드에 저장한다. Oneof 테이블은 멤버 개체가 제한된 클래스 정보를 저장하기 위해 Class 테이블의 기본키와 Individual 테이블의 기본키를 외래키로 참조한다.



<그림 7> 프로퍼티 정의 내용을 저장하기 위한 테이블 스키마

<그림 7>은 본 논문에서 제안한 OWL 저장 모델에서 온톨로지에 기술된 프로퍼티의 정의와 프로퍼티간의 관계 정의 내용을 저장하는 3개 테이블들 스키마를 보여준다. 일반적으로 프로퍼티는 개체들 간의 관계를 표현하는 ObjectProperty와 개체가 가지는 특성 값을 표현하는 DataProperty로 구분할 수 있기 때문에 본 논문에서는 프로퍼티의 두 가지 유형 차이를 반영하여 별도의 테이블로 구성한다. ObjectProperty 테이블은 온톨로지에 정의된 모든 ObjectProperty의 소수 레이블과 이름, 도메인과 레인지 클래스, 타입 정보를 저장한다. 도메인과 레인지 클래스는 Class 테이블의 기본키를 외래키로 참조하여 표현하고 프로퍼티의 타입은 symmetric, transitive, functional, inversefunctional으로 구분한다. DataProperty 테이블은 레인지 클래스 대신 프로퍼티 값의 데이터 유형을 저장한다. InverseProperty 테이블은 프로퍼티간의 역관계를 저장한다. InverseProperty 테이블의 pno1과 pno2 필드는 ObjectProperty 테이블의 기본키를 참조한다. 역관계는 다른 프로퍼티 타입과는 달리 두 개 프로퍼티 간의 관계를 표현하기 때문에 별도의 테이블로 구성한다.

메타데이터에는 개체의 클래스 타입이 선언되고 개체간의 프로퍼티 관계가 트리플 구조의 문장 형태로 기술된다. 그러한 특성에 따라 <그림 8>에 제시된 4개의 테이블에 메타데이터의 내용을 저장한다.



<그림 8> 메타데이터 내용을 저장하기 위한 테이블 스키마

<그림 8>에서 Individual 테이블은 개체의 이름과 함께 Class 테이블의 기본키를 외래키로 참조하여 개체의 클래스 타입을 저장한다. IndividualRelation 테이블은 개체간의 동일성과 이질성을 저장하는데 직접적으로 기술된 것뿐만 아니라 본 논문의 3.1절에서 제시한 추론 규칙에 따라 유도된 새로운 내용도 함께 저장한다. IndividualRelation 테이블은 두 개체간의 동일성과 이질성 관계 정보를 저장하기 위해 Individual 테이블의 기본키를 참조하는 ino1과 ino2 필드를 포함한다. 개체와 개체의 프로퍼티 관계는 트리플 구조로 구성된 ObjectTriple 테이블에 저장한다. 마찬가지로 트리플 구조로 구성된 DataTriple 테이블에는 개체가 가지는 프로퍼티의 특성

값을 저장한다. 그리고 관계 추론에 의해 유도된 새로운 메타데이터 내용도 유형에 따라 ObjectTriple 테이블과 DataTriple 테이블에 나누어 저장한다. ObjectTriple 테이블과 DataTriple 테이블에서 개체는 Individual 테이블의 기본키를 외래키로 참조하여 표현한다. 그리고 프로퍼티는 ObjectProperty 테이블이나 DataProperty 테이블의 기본키를 외래키로 참조하여 표현한다.

<표 1> 제안한 OWL 저장 모델을 위한 저장 과정과 규칙

	저장 규칙
1단계	클래스와 프로퍼티의 계층 관계를 소수 레이블로 표현
2단계	온톨로지에 정의된 내용을 테이블로 구성 규칙1) 클래스 정의를 테이블에 저장 (Class, ClassRelation, Oneof 테이블) 규칙2) 제한된 클래스 정의를 테이블에 저장 (RestrictionClass 테이블) 규칙3) 프로퍼티 정의를 테이블에 저장 (ObjectProperty, DataProperty 테이블) 규칙4) 프로퍼티 관계를 테이블에 저장 (InverseProperty 테이블)
3단계	메타데이터에 기술된 내용을 테이블로 구성 규칙1) 개체 정의를 테이블에 저장 (Individual, IndividualRelation 테이블) 규칙2) 개체에 대해 기술된 내용을 테이블에 저장 (ObjectTriple, DataTriple 테이블)

<표 1>은 본 논문에서 제안한 OWL 저장 모델에 온톨로지와 메타데이터를 저장하기 위한 저장 과정과 규칙의 요약한 것이다.

Ⅳ. 성능 평가

본 논문에서 제안한 저장 모델은 OWL로 표현 가능한 온톨로지의 모든 정의 내용과 메타데이터에 기술된 내용을 정규화된 테이블로 저장한다. 그리고 온톨로지 추론으로 새롭게 생성된 메타데이터 정보까지 저장과 질의의 대상으로 확대하여 사용자 질의에 부합하는 정확한 검색 결과를 빠른 시간 내에 제공하는 것을 목표로 한다. 따라서 본 논문에서 제안한 OWL 저장 모델의 성능을 평가하기 위해 유사한 목표를 가지고 제안된 기존의 관계형 데이터베이스 기반의 저장 모델 중 가장 최근에 제안된 OntoRDB[6]를 선택하여 다양한 질의 유형별로 처리 시간을 비교하였다.

본 논문에서 제안한 OWL 저장 모델의 스키마와 OntoRDB 시스템의 저장 스키마를 인텔 Core2 Duo T8300(2.4GHz) CPU와 2GB 메모리로 구성된 컴퓨터에서 Windows XP 운영체제와 함께 설치된 오라클 10g에 각각 구성하고 Visual Studio를 이용해 C 언어로 질의 처리 모듈을 구현하였다.

본 논문에서는 제안한 OWL 저장 모델의 질의 처리 성능을 평가하기 위해서 OWL이 제공하는 다양한 의미적 관계를 모두 포함하고 있는 온톨로지를 직접 설계하였다. 실험을 위해 설계한 온톨로지는 레스토랑 주제와 관련된 것으로 18개의 클래스와 11개의 프로퍼티로 구성되어 있으며 계층 관계는 물론 본 논문에서 고려하는 다양한 의미적 관계를 모두 포함한다. 그리고 OWL 인스턴스 생성기를 직접 구현하여 설계한 레스토랑 온톨로지에 따라 세 가지 형태의 메타데이터를 생성하고 설계한 온톨로지와 함께 실험에 사용하였다. 실험 데이터로 사용한 메타데이터는 포함되어 있는 개체의 개수와 트리플 개수에 따라 <표 2>와 같이 세 가지로 구분된다. <표 2>에서 트리플의 개수는 직접 기술된 트리플의 개수와 추론을 통해 미리 추출한 트리플의 개수를 합한 것이다.

<표 2> 실험에 사용된 세 가지 메타데이터

	데이터1	데이터2	데이터3
개체 수	2,510개	25,010개	250,010개
트리플 수	2,600개	26,000개	260,000개

본 논문에서 제안한 OWL 저장 모델의 성능을 여러 가지 측면에서 측정하기 위해 온톨로지 추론을 지원하는 질의와 함께 온톨로지에 정의된 내용 자체에 대한 질의

에 대해서도 처리 시간을 비교하였다. <표 3>은 실험에 사용한 8가지 질의 유형을 요약한 것이다.

<표 3> 실험에 사용된 질의

	질의 내용
Q1	"Customer" 클래스의 하위 클래스 검색
Q2	"Food" 클래스와 동등 관계의 클래스 검색
Q3	"orders" 프로퍼티의 도메인 클래스 검색
Q4	"hasWork" 프로퍼티의 maxcardinality 검색
Q5	"Customer" 클래스 타입의 개체 검색
Q6	"orders" 프로퍼티 관계를 맺고 있는 개체 쌍 검색
Q7	"cooks" 프로퍼티 관계를 맺고 있는 개체 쌍 검색
Q8	"hasManager" 프로퍼티 관계를 맺고 있는 개체 쌍 검색

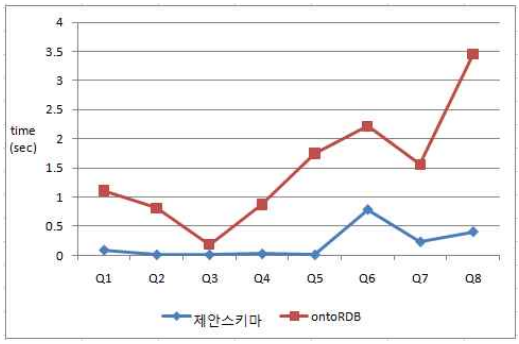
질의 처리 시간에 영향을 줄 수 있는 데이터베이스의 캐싱 기능을 제외하기 위해 모든 질의는 데이터베이스 서비스가 구동된 직후 바로 처리하고 초 단위로 시간을 측정하였다. 그리고 제안한 저장 모델과 OntoRDB 시스템의 저장 모델에 대해 인덱스의 사용을 배제하고 각 질의마다 같은 방법으로 10번씩 수행한 후 평균 처리 시간을 계산한 결과를 비교하였다.

본 논문에서 제안한 저장 모델과 OntoRDB 시스템의 저장 모델 모두 <표 3>의 모든 질의 유형에 대해 처리 결과로 같은 개수의 결과 데이터가 반환되었다.

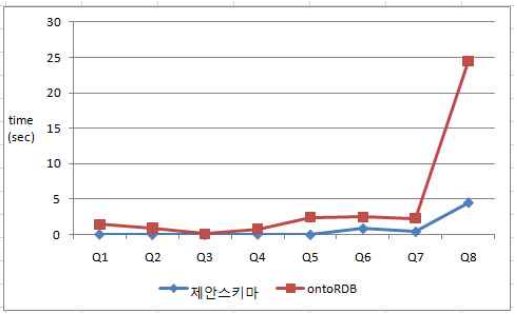
<그림 9>는 레스토랑 온톨로지와 첫 번째 실험 메타데이터를 대상으로 본 논문에서 제안한 저장 모델 스키마와 OntoRDB 시스템의 저장 스키마에 대해 <표 3>에서 제시한 8개 질의의 처리 시간을 비교한 그래프이다.

<그림 10>은 레스토랑 온톨로지와 두 번째 실험 메타데이터를 대상으로 본 논문에서 제안한 저장 모델 스키마와 OntoRDB 시스템의 저장 스키마에 대해 <표 3>에서 제시한 8개 질의의 처리 시간을 비교한 그래프이다.

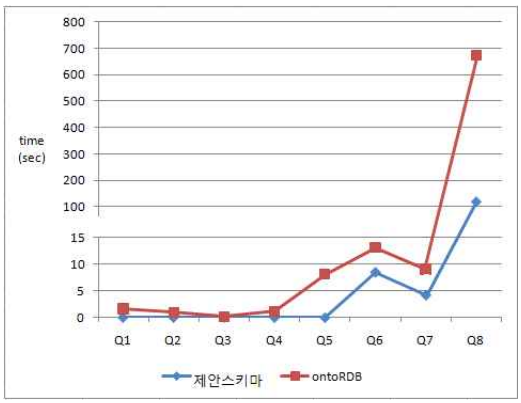
<그림 11>은 레스토랑 온톨로지와 세 번째 실험 메타데이터를 대상으로 본 논문에서 제안한 저장 모델 스키마와 OntoRDB 시스템의 저장 스키마에 대해 <표 3>에서 제시한 8개 질의의 처리 시간을 비교한 그래프이다.



<그림 9> 첫 번째 실험 데이터에 대한 질의 처리 시간 비교



<그림 10> 두 번째 실험 데이터에 대한 질의 처리 시간 비교



<그림 11> 세 번째 실험 데이터에 대한 질의 처리 시간 비교

<그림 9>, <그림 10>, <그림 11>에서 그래프로 제시된 비교 결과를 통해 모든 유형의 질의에 대해 본 논문에서 제안한 저장 모델이 OntoRDB의 저장 모델에 비해 빠른 질의 처리 시간이 소요되는 것을 확인할 수 있다.

본 논문에서 제안한 저장 모델은 질의 처리 시 클래스

나 프로퍼티의 계층 구조를 이용한 추론을 실시간으로 수행하기 때문에 전체적인 질의 처리 시간의 비효율성을 초래할 수 있다. 하지만 본 논문에서는 이러한 문제를 해결하기 위해 소수 레이블링 기법을 이용하여 계층 관계에 대한 빠른 판단이 가능하기 때문에 클래스나 프로퍼티의 계층 구조에 기반을 둔 추론을 빠르게 처리할 수 있다. 그래서 세 가지 실험 데이터 모두에 대해 계층 구조에 기반을 둔 추론이 특별히 더 중요한 역할을 하는 Q1, Q2, Q5, Q6, Q7, Q8 질의가 OntoRDB 시스템의 저장 모델에 비해 빠르게 처리된 것을 확인할 수 있다.

<그림 9>, <그림 10>, <그림 11>에서 보면 제안한 저장 모델에서도 다른 질의에 비해 Q6, Q7, Q8 질의의 처리 시간이 더 많이 걸리는 것을 확인할 수 있다. 이것은 Q6, Q7, Q8의 질의가 관계 추론을 통해 추출되어 미리 저장된 많은 양의 데이터를 처리 대상으로 하기 때문이다. 그러나 본 논문에서 제안한 저장 모델은 미리 추론 결과를 저장해두기 때문에 질의 처리 시에 추론을 수행해서 결과를 반환해야 하는 OntoRDB 시스템의 저장 모델에 비해 Q6, Q7, Q8 질의의 처리 시간이 다른 질의의 처리 시간과 그 차이가 크지 않다. 특히 이행관계로 정의된 프로퍼티를 대상으로 질의하는 Q8의 경우 처리 시간이 가장 많이 요구되지만 본 논문에서 제안한 저장 모델은 OntoRDB 시스템의 저장 모델과 비교할 때 상대적으로 처리가 빠르다는 것을 확인할 수 있다.

즉, 서로 다른 크기의 실험 데이터와 다양한 유형의 질의로 처리 시간을 비교하여 성능을 측정한 결과 본 논문에서 제안한 OWL 저장 모델을 이용하면 사용자가 원하는 검색 결과를 정확하게 제공하면서도 온톨로지 추론을 지원하는 질의 처리를 빠르게 수행할 수 있다.

V. 결론

시맨틱 웹에서 정보 자원의 개념과 의미적 관계를 정의하는 온톨로지와 메타데이터의 역할은 중요하다. 특히 풍부한 표현력과 추론 기능을 가지고 있는 OWL로 기술된 온톨로지와 메타데이터를 효율적으로 저장하고 검색하기 위한 시스템이 필요하다. 그러나 대부분의 기존 저장 시스템들은 OWL이 제공하는 다양한 제한 사항과 의미적 관계를 고려하지 않고 많은 시간이 소요되는 온톨로지 추론을 지원하지 않는 경우가 많았다. 따라서 본 논문에서는 OWL의 장점을 그대로 수용하면서 OWL로 표현할 수 있는 제한 사항과 의미적 관계의 정의를 메타데이터와 구분하여 관리할 수 있는 저장 모델을 제안하였다. 본 논문에서 제안한 OWL 저장 모델은 대용량의 온톨로지와 메타데이터를 관리할 수 있도록 관계형 데이터베이스에 기반을 두고 있다. 그리고 온톨로지 추론을 통해 추출되는 새로운 정보까지 저장과 질의의 대상으로 확장하기 때문에 사용자가 원하는 정확한 검색 결과를 제공할 수 있다. 하지만 온톨로지 추론이 질의 처리에 비효율성을 초래할 수 있기 때문에 본 논문에서는 이러한 문제를 해결할 수 있는 여러 가지 방법과 규칙들을 저장 모델과 함께 제안하였다. 본 논문에서 제안한 저장 모델은 레이블링 기법을 이용해 클래스나 프로퍼티의 계층 관계를 쉽게 판단하고 온톨로지 추론을 질의 처리 시 실시간으로 이루어지는 추론과 질의 처리 전에 수행해서 그 결과를 미리 저장하는 추론 규칙으로 분류하고 각 특성에 맞게 처리함으로써 빠른 질의 처리 시간이 가능하도록 하였다. 그리고 여러 크기의 실험 데이터에 대해 다양한 질의 유형으로 실험적 평가를 수행하여 제안된 OWL 저장 모델의 처리 효율성을 증명하였다.

향후에는 다른 OWL 저장 시스템과의 질의 처리 효율성을 비교하는 실험과 함께 데이터 적재 시간에 대한 실험적 평가를 수행하여 제안 저장 모델의 성능을 다각도로 분석하고 온톨로지가 변화하는 동적 환경에서의 활용 여부를 판단하기 위한 연구를 계속 진행하고자 한다.

참고문헌

[1] 권준희・김성림, “시맨틱 웹 환경에서의 레벨화된 컨텍스트 온톨로지를 이용한 추천 기법,” 디지털산업정보학회논문지, 제5권, 제2호, 2009, pp. 95-100.

[2] OWL 2 Web Ontology Language Primer, “http://www.w3.org/TR/2009/REC-owl2-primer-20091027/”

[3] Y. Guo, Z. Pan, and J. Heflin, “An Evaluation of Knowledge Base Systems for Large OWL Datasets,” Third International Semantic Web Conference, LNCS 3298, 2004, pp. 274-288.

[4] 김진형・정동원・백두권, “XPOS: 효율적인 질의 처리를 위한 XPath 기반의 OWL 저장 모델,” 한국정보학회논문지, 제35권, 제3호, 2008, pp. 243-256.

[5] M. Park, J. Lee, C. Lee, J. Lin, O. Serres, and C. Chung, “An Efficient and Scalable Management of Ontology,” Lecture Notes In Computer Science, Vol. 4443, Springer-Verlag, 2007, pp. 975-980.

[6] 이훈・유상봉, “효율적인 온톨로지 저장과 처리를 위한 관계형 데이터베이스 설계,” 한국정보기술학회논문지, 제8권, 제9호, 2010, pp. 143-151.

[7] Dean Allemang and James Hendler, Semantic Web for the Working Ontologist, Morgan Kaufmann, 2008.

[8] 김선영・권동섭・이석호, “소수 레이블을 이용한 RDF/RDFS 인덱스 구조,” 한국정보과학회, 학술발표논문집, 제32권, 제1호, 2005, pp. 82-84.

■ 저자소개 ■

김 연 희
Kim, Youn Hee

2007년 3월~현재 부천대학교 e-비즈니스과 강의전담교수
2006년 8월 홍익대학교 컴퓨터공학과(공학박사)
2002년 2월 홍익대학교 컴퓨터공학과(공학석사)
2000년 2월 홍익대학교 컴퓨터공학과(공학사)

관심분야 : 시맨틱 웹, XML, 분산 데이터베이스, 모바일 데이터베이스
E-mail : yhkim@bc.ac.kr

이 애 정
Lee, Ae Jung

1993년 3월~현재 한양여자대학교 정보경영과 교수
2000년 2월 홍익대학교 전자계산학과(이학박사)
1982년 2월 연세대학교 전산학과(공학석사)
1975년 2월 서강대학교 생물학과(이학사)

관심분야 : 시맨틱 웹, 분산 시스템, 멀티미디어 데이터베이스
E-mail : ajlee@hywoman.ac.kr

논문접수일 : 011년 8 월 18 일
수 정 일 : 2011년 9 월 1 일
게재확정일 : 2011년 9 월 5 일