온톨로지에서의 그래프 레이블링을 이용한 효율적인 트랜지티브 클로저 질의 처리

(Efficient Processing of Transitive Closure Queries in Ontology using Graph Labeling)

김 종 남 † 정 준 원 † 민 경 섭 †† 김 형 주 †††
(Jongnam Kim) (Junwon Jung) (Kyeung-Sub Min) (Hyoung-Joo Kim)

요 약 온톨로지는 특정 개념에 대한 부가정보 및 개념간의 관계를 기술하는 방법으로서 고차원의 웹과 서비스를 실현하기 위한 시멘틱 웹, 그리고 지식관리 시스템을 비롯한 다양한 응용분야의 요구와 관심이 증가하면서 그 중요성이 대두되고 있다. 온톨로지에서 정보에 대한 접근은 특정 개념과 특정 관계를 가지는 데이타를 찾는 것이 주를 이루는데, 이러한 관계가 주로 트랜지티브 관계이기 때문에 트랜지티브 질의를 처리하는 것이 많은 비중을 차지한다. 또한 이와 같은 트랜지티브 클로저 질의 처리는 재귀호출의 형태로서 그 처리 비용 또한 매우 크다.

본 논문에서는 이와 같은 트랜지티브 클로저 질의의 효율적 처리를 지원하기 위한 방법으로써 그래프 레이블링을 이용한 전처리 기법을 제안한다. 이 기법은 저장 공간을 효율적으로 사용하고 알고리즘도 단순한 특징을 가지기 때문에 트랜지티브 클로저 질의에 대한 응답 시간을 줄이는 장점을 가지게 된다. 그리고 이와 같이 제안한 기법에 대해 기존 시스템들과 비교해 봄으로써 그래프 레이블링을 이용한 기법이 대용량 온톨로지에서의 트랜지티브 클로저 질의 처리에 효율적임을 보이고자 한다.

키워드 : 온톨로지, 트랜지티브 관계, 트랜지티브 클로저, 레이블링

Abstract Ontology is a methodology on describing specific concepts and their relationships, and it is being considered important more and more as semantic web and variety of knowledge management systems are being highlighted. Ontology uses the relationships among concepts to represent some concrete semantics of specific concept. When we want to get some useful information from ontology, we severely have to process the transitive relationships because most of relationships among concepts represent transitivity. Technically, it causes recursive calls to process such transitive closure queries with heavy costs.

This paper describes the efficient technique for processing transitive closure queries in ontology. To the purpose of it, we examine some approaches of current systems for transitive closure queries, and propose a technique by graph labeling scheme. Basically, we assume large size of ontology, and then we show that our approach gives relative efficiency in processing of transitive closure queries.

Key words : Ontology, Transitive relationship, Transitive closure, Labeling

1. 서 론

시맨틱 웹은 의미의 연결을 통해 정보를 제공하는 차세대 웹으로서 정보의 의미를 개념으로 정의하고 개념간의 관계성을 표현함으로써 정보를 공유한다. 이러한 시맨틱 웹을 현실화하기 위해서는 '잘 정의된 개념' 이 필요하고 이것이 바로 시맨틱 웹에서 온톨로지의 역할이다. 온톨로지의 대표적인 예인 야후의 계층적 카테고리나, Gene Ontology[1]를 보면 알 수 있듯이 온톨로지를 구성하는 방법에는 객체 지향 방법론이 응용되어 사용된다. 이러한 온톨로지에서의 주요한 관계(relationship)에는 'subClassOf'나 'subPropertyOf' 등이 있는데, 이는 트랜지티브 관계(transitive relationship)이므로 질

· 본 연구는 BK-21 정보기술 사업단과 정보통신부 ITRC(e-BIZ 기술연구센터)에 의해 지원되었음
† 학생회원 : 서울대학교 컴퓨터공학부
jnkim@oopsla.snu.ac.kr
jwjung@oopsla.snu.ac.kr
†† 정 회 원 : 서울대학교 컴퓨터공학부
ksmin@oopsla.snu.ac.kr
††† 종신회원 : 서울대학교 컴퓨터공학부 교수
hjk@oopsla.snu.ac.kr
논문접수 : 2004년 12월 22일
심사완료 : 2005년 7월 30일

의 처리시 해당 개념에 대한 인스턴스뿐 아니라 그 하위 클래스들도 모두 대상에 포함되게 된다. 즉 원래의 그래프 G=(N, E)에 대한 트랜지티브 클로저(transitive closure) $G^+=(N, E^+)$에 대해 질의를 하게 된다. 이론적으로, 어떤 그래프의 트랜지티브 클로저를 구하는 것은 해당 그래프에 대한 인접 행렬 표현을 생각했을 때 행렬의 곱의 복잡도에 해당하는 것으로, 대용량의 데이타를 가정했을 때 이를 동적으로 계산하기 보다는 미리 계산해 놓은 후 참조하는 방식이 일반적이다.

이러한 트랜지티브 클로저 질의는 재귀순환의 형태로서 대부분의 SQL 구현들이 이러한 재귀적 표현을 지원하지 못하므로, 이를 계속적인 SQL 문장으로 반복적으로 구해야 하고 이것은 시간 측면에서 성능의 큰 저하를 가져온다. 또한 현재 Jena[2]와 RDFSuite[3]을 포함한 일반적인 온톨로지 저장소들은 트랜지티브 클로저 질의를 보편적인 그래프 탐색과 메모리 캐쉬에 의존하고 있어서, 대용량의 온톨로지를 가정했을 때 트랜지티브 클로저 질의에 대한 성능이 좋지 못하다. 또한Gene Ontology시스템에서는 graph_path 테이블에 스키마 그래프의 트랜지티브 클로저를 미리 계산하여 저장하고 있으나 이는 공간 측면에서 비효율적 일 뿐 아니라 원래의 트리플(triple)과 그 데이타에서 유추된 트리플을 혼합하여 저장하므로 바람직하지 못하다. 또한 gene ontology처럼 온톨로지 내의 상속관계가 is_a, part_of 등 여러 개 일 수 있고, 관계가 점차 세분화 될 수 있는데 온톨로지 저장소들은 기본적으로 데이타를 트리플 형태로 저장하므로 결국 세분화된 관계에 대해서는 더 많은 트리플 간의 조인 연산이 필요하게 된다.

일반적으로, 온톨로지에서의 효율적인 트랜지티브 클로저 질의 기법은 다음 세 가지의 서로 배반적인 요건들을 만족시켜야 할 것이다.

1. 저장공간 효율성: 대용량 온톨로지를 가정
2. 검색 효율성: 일반적인 관계에 대해서 효율적인 검색이 보장 되어야 함.
3. 갱신 효율성: 점진적으로 갱신하는 비용이 다시 재계산하는 비용보다는 적어야 함.

본 논문은 다음과 같은 의미를 가진다.

1. 그래프 레이블링을 온톨로지에 적용하여 클래스(class)나 속성(property)들간의 상속관계를 레이블만 보고도 결정할 수 있게 한다. 일반적으로 그래프의 트랜지티브 클로저는 O(n2)의 공간을 차지하나 구간 기반 레이블링 기법은 O(n)으로 필요한 정보를 유지할 수 있다.
2. 생물 정보학에서의 사실상의 표준 온톨로지를 구현한 Gene Ontology시스템에서의 트랜지티브 클로저 질의에 대한 처리 방법과 널리 쓰이는 온톨로지 저장소인 Jena 시스템의 처리 방법을 설명하고 대용량의 온톨로지를 가정했을 때 이의 문제점을 분석한다.
3. 관계형 모델에 비해 비교적 변화가 자주 일어나는 온톨로지 모델에서 구간 기반 레이블링의 점진적인 갱신이 어떻게 가능한지 설명한다.
4. 대용량 온톨로지의 예로써Gene Ontology에 대한 실험을 통해 기존의 기법과 본 논문의 기법을 비교하고, 제안하는 기법의 우수성을 입증한다.
5. 그래프 레이블링은 트리 레이블링을 변형한 것이고 이는 주로 XML 질의처리에서 색인구조로 사용되는 반면, 본 논문에서 제안한 기법은 색인 구조로는 B+-tree를 사용하고 트랜지티브 클로저를 구하는데 그래프 레이블링을 이용했다는 면에서 다른 접근방식을 가지고 있다. 또한 온톨로지는 상속관계의 종류가 세분화 되는 추세이고 갱신에 대한 요구도 빈번해지는 상황에서, 본 논문의 기법은 4.2, 4.3절에서 다룬 것과 같이 두 가지 중요한 요구사항을 일관된 방법으로 수정할 수 있다.

본 논문의 구성은 다음과 같다.

2장에서는 관련연구에 대해 살펴보고, 3장에서 본 논문에서 사용되는 데이타 모델에 대해 설명하고 기존의 기법의 문제점을 설명한다. 4장에서는 본 논문이 제안하는 기법에 대해 알아본다. 5장에서는 실험을 통해 효율성을 설명하고, 6장에서는 결론 및 향후 연구를 기술한다.

2. 관련연구

현재까지 온톨로지 관련 연구는 데이타베이스 관점에서 바라보는 대용량을 가정했을 때의 질의 처리의 성능 개선보다는 인공 지능 관점에서의 보다 지능적인 또는 기계가 잘 이해할 수 있는 연구에 초점이 맞추어져 있고, 사실상 그 또한 실험적인 수준에 머물고 있다. 이는 실제적인 응용에서 대용량의 온톨로지 데이타를 찾기 어려운 측면 때문이기도 한데, 그런 의미에서 Gene Ontology[1]는 요즈음 각광 받고 있는 생물학 분야와 결합되어 비교적 대량의 온톨로지 데이타를 제공하고 있다는 점에서 흥미롭다.

온톨로지에 관한 표준으로 RDF/S, OWL 등이 W3C[4]에 의해 제정되어 활발히 연구되고 있으나, 아직까지 실용적인 응용들은 온톨로지의 약한 표현 형태인 택사노미(taxonomy) 정도에 머물고 있고, [5]에서는 이러한 택사노미의 계층 구조 탐색에 대해 세가지 레이블링 기법을 비교하고 있다. 실제로 계층 구조에 관한 효율적인 탐색은 기존의 지식관리 시스템에서도 다루어진 바 있으나[6] 이는 메모리를 효율적으로 이용하기 위한 압축기법에 치중하고 있고, 하나의 상속관계 만을 다

루므로 현재의 온톨로지 시스템에 대한 적용에는 미흡한 면이 있다. [7]은 역인덱스를 사용하여 관계형 데이타베이스를 효율적으로 활용하는데 이는 성능 측정에 대한 아이디어를 제공하고 있다.

현재의 대표적인 온톨로지 저장소로는 Jena와 Sesame를 들 수 있다. Jena[2]는 HP의 영국 브리스톨 연구소에서 개발한 시맨틱 웹 툴킷으로서 RDF/OWL API와 저장소를 비롯하여 간단한 추론 엔진을 구현하여 제공하고 있으나 추론은 성능이 미약한 상태이고 특히 트랜지티브 클로저 질의를 처리하는 모듈인 transitive reasoner는 트랜지티브 리덕션(transitive reduction)을 통해 공간 효율성을 높이고 트랜지티브 클로저에 대해서는 메모리 캐쉬를 하는 단순한 방법을 사용하고 있다. 하지만 이는 대용량의 온톨로지를 가정했을 때는 트랜지티브 클로저 계산에 필요한 트리플들이 모두 메모리로 올라올 수 없고, 따라서 필요할 때 마다 디스크 I/O를 수행해야 하고 이는 큰 부담이 된다. 반면 Sesame [8]는 Gene Ontology 시스템과 마찬가지로 트랜지티브 관계에 대해 별도의 테이블을 생성하고 이를 바탕으로 미리 계산해서 처리하는 방식을 사용한다.

Sesame는 European IST project On-To-Knowledge에 대한 연구 프로토타입으로 시작되었다. 기본적으로 RDF/RDFS 및 OWL 데이타를 다루기 위한 연산과 질의 및 저장을 지원하며 이와 같은 기능을 위한 인터페이스를 제공한다. Sesame에 관계형 데이타베이스를 이용하여 RDF 데이타를 저장할 경우 트리플로 나뉘어 저장되는 것은 Jena와 동일하나, 트랜지티브 클로저 질의 처리를 위한 별도의 테이블을 두고 미리 모든 클로저 트리플들을 계산해 놓는 방식으로 이는 GO 시스템과 동일하고 그 문제점을 3.2절에서 다룬다.

본 논문에서 사용하는 그래프 레이블링 기법은 기존의 XML 질의 처리를 위해 빈번히 사용되는 구간 기반 레이블링 기법을 그래프로 확장한 것이다[10]. 일반적으로XML에서의 레이블링 기법은 트리 모델을 가정한 상태에서 XPath질의를 처리하기 위해 두 노드가 조상/후손 관계인지 결정 하기 위한 역인덱스를 제공해주는 반면, 온톨로지에서의 트랜지티브 클로저 질의 처리는 한 노드의 모든 조상 또는 모든 후손을 효율적으로 찾는 것이 중요하다. 따라서 XML의 질의 처리와는 달리 색인 구조로는 B+-tree를 사용하고 트랜지티브 클로저를 구하는데 그래프 레이블링을 이용했다는 면에서 다른 접근방식을 가지고 있다.

3. 배경 지식

3.1에서는 Gene Ontology의 데이타 모델에 대해 설명하고, 3.2에서 추론의 기본 형태인 트랜지티브 클로저 질의에 대한 처리 기법을 알아 본 후, 3.3에서는 트랜지티브 클로저 질의의 처리에 대해 Jena 시스템에서 사용하는 방식에 대해 설명한다.

3.1 Gene Ontology 데이타 모델

Gene Ontology는 각각의 생물들에 대해 연구하던 여러 회사가 컨소시움을 형성하여 용어의 정리를 위해 만든 온톨로지이고, 따라서 대단히 방대한 양의 용어를 담고 있다. 생물학 분야에서의 용어를 크게 세 가지로 분류한 molecular function, biological process, celluar component을 기본으로 전체적으로 비순환 방향 그래프(DAG)로 표현된다. DAG는 자식 노드가 여러 개의 부모 노드를 가질 수 있다는 것이 특징이고, 하나의 자식 개념은 자신의 부모 개념에 대한 인스턴스이거나(is_a relationship) 또는 컴포넌트이다(part_of relationship). 자식 노드는 하나 이상의 부모를 갖고, 서로 다른 부모에 대해 다른 클래스의 관계를 갖게 될 것이다. 표준적인 데이타 모델로 각광을 받고 있는 XML만으로는 DAG 구조를 효율적으로 표현하기 힘들기 때문에, Gene Ontology는 rdf link등의 RDF 모델을 사용하고 있다.

3.2 트랜지티브 클로저 질의 처리

그림 1에서 보듯이, 그래프 G의 트랜지티브 클로저는 한 노드에서 갈 수 있는 모든 경로에 대해 새로운 간선을 만들어 준 그래프 G^*를 의미하고, 트랜지티브 리덕션은 반대로 간접적으로 갈 수 있는 경로를 제거해서 필수적인 간선만 나타내는 그래프 G^-를 의미한다. Gene Ontology의 그래프 구조를 DBMS에 저장하는 구조는 기본적으로 term과 term2term 테이블에 의해 유지되고, relationship 종류로는 is_a, part_of, develops_from이 있다. 온톨로지 질의 처리를 위해서는 그래프 탐색이 필요하게 되고, 이는 term2term 테이블에 대한 재귀적인 질의 형태가 되나, 대부분의 SQL 구현들은 이를 지원하지 못하므로 일반적으로 반복적인 SQL 호출의 형태가 되게 된다. 하지만, 이런 방식은 시간에 대한 비용이 대단히 커지므로 Gene Ontology에서는 임의의 노드에서 갈 수 있는 모든 트랜지티브 하위 클래스를 미리 계산해 놓고(즉, 트랜지티브 클로저), 이를 graph_path 테

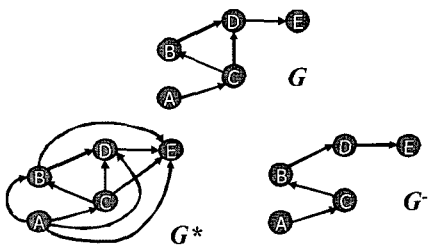

그림 1 트랜지티브 클로저(G^*)와 트랜지티브 리덕션(G^-)

이블에 저장한다. 정확히 말해서 노드 자신도 포함하는 리플렉시브 트랜지티브 클로저(reflexive transitive closure)를 계산한다.

예를 들어, 그림 2에서 모든 'engyme' 유전자를 찾으라는 질의에 답하기 위해서는 'engyme' 뿐만 아니라 'helicase'와 'DNA helicase' 등 engyme의 transitive closure에 속하는 모든 노드들의 유전자 또한 답이 된다. Ontology그래프의 노드가 n개라고 할 때 그래프 자체는 O(n)에 탐색이 가능하지만, 그것의 트랜지티브 클로저 그래프는 $\sum_{k=1}^{n} k$ = $O(n^2)$이 되는데 Gene Ontology 시스템에서는 그림 3의 G^*를 저장하고 있는 셈이므로 대용량의 Ontology를 가정했을 때 매우 공간 효율적이지 못하고 따라서 효율적인 기법이 요구된다.

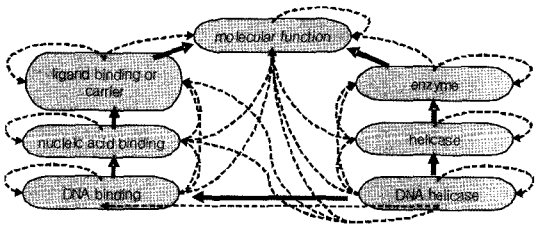

그림 2 Gene Ontology에서의 트랜지티브 클로저 질의

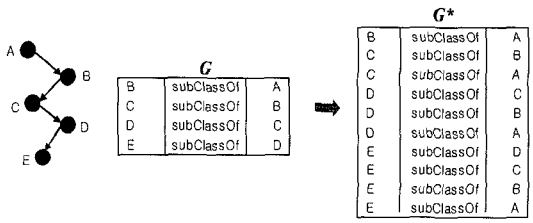

그림 3 트리플 관점에서의 트랜지티브 클로저

3.3 Jena에서의 트랜지티브 클로저 질의 처리

Jena[2]는 휴렛 팩커드(HP)에서 만든 시맨틱 웹 툴킷으로서, 기본적으로 RDF API 및 온톨로지 저장소의 기능과 추론 기능을 제공하고 있고 현재 가장 활발히 사용 및 연구되고 있는 시스템이다. 일반적으로 그림 3에서 만약 G^*를 동적으로 계산하려고 한다면 대용량의 온톨로지가 데이타베이스에 저장되어 있다고 가정했을 때 계산과정에서 엄청난 양의 I/O를 필요로 하게 되고 이는 시간 효율성의 저하로 나타난다.

대표적인 온톨로지 저장소인 Jena에서는 데이타베이스에서 트랜지티브 클로저 계산에 필요한 트리플들을 메모리로 복사해 와서 메모리 상에서 다시 재구성하는 방법을 사용하고 있다. 다시 말해 온톨로지 그래프의 트랜지티브 리덕션을 통해 필요한 저장공간을 줄이고 트랜지티브 클로저에 대해서는 그 그래프에서의 탐색을 통해 갈수 있는 모든 노드를 직접 방문하여 찾고, 이를 메모리에 캐시 하는 정책을 사용한다. 하지만 이는 대용량의 온톨로지를 가정했을 때는 트랜지티브 클로저 계산에 필요한 트리플들이 모두 메모리로 올라올 수 없고, 따라서 필요할 때 마다 디스크 I/O를 수행해야 하고 이는 큰 부담이 된다.

또한 온톨로지에서의 관계는 반드시 한 종류의 상속관계로 표현 될 필요는 없다. Gene Ontology 에서의 트랜지티브 관계는 is_a와 part_of, 그리고 develops_from이 있는데 앞으로 더욱 세분화(specialization) 될 예정이라고 한다. W3C의 표준 온톨로지 언어인 OWL에서는 *subClassOf*와 *subPropertyOf* 로만 상속관계가 표현되므로 Gene Ontology의 클래스간의 part_of 관계를 표현하기 위해서는 OWL의 Restriction을 이용해야 한다. 즉 Restriction으로 선언하고 onProperty에 part_of 관계임을 명시하는 방식인데 이것의 제점은 기존에 한 개의 트리플로 표현 가능했던 관계가 4개의 트리플

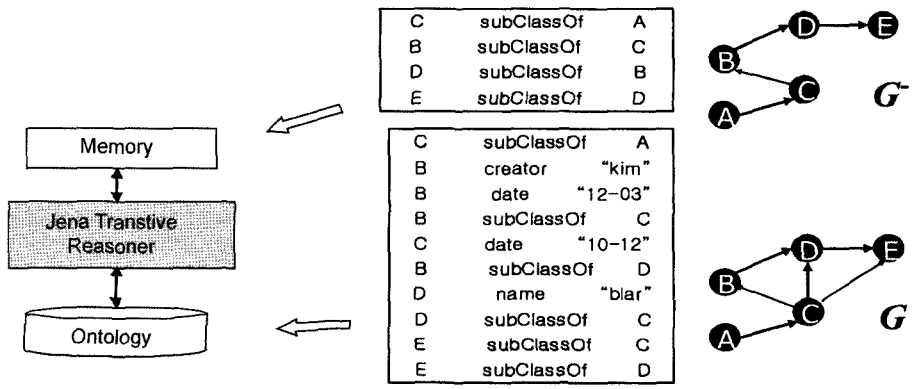

그림 4 Jena에서의 트랜지티브 클로저 질의 처리 방식

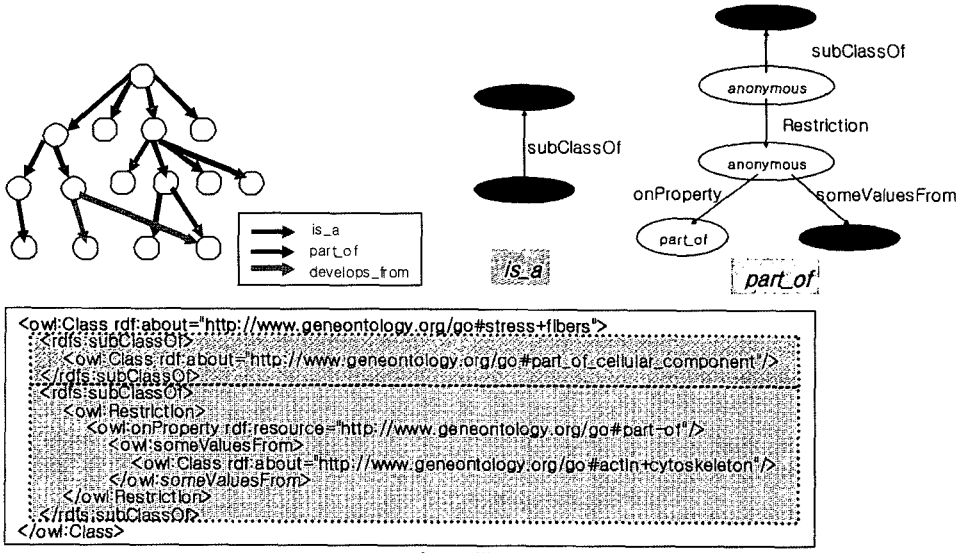

그림 5 Jena에서의 is_a와 part_of 처리 방식 비교

로 표현되게 되고 결국 어떤 클래스의 part_of 관계인 하위 클래스를 찾는 것은 4개의 트리플을 조인하는 연산을 발생시킨다. 단지 하나의 하위 클래스가 아니라 어떤 클래스의 트랜지티브 클로저를 계산하게 된다면 엄청난 양의 트리플을 더 찾아보게 되는 것이 되므로 이 또한 전체적인 응답시간의 저하를 가져오게 된다. 그림 5는 실제 Gene Ontology 파일에서 is_a와 part_of 관계의 트랜지티브 클로저가 트리플 레벨에서 어떻게 찾아지는 지를 보여주고 있다.

4. 그래프 레이블링을 이용한 트랜지티브 클로저 질의 처리 기법

4.1에서는 기존의 XML등에서 사용된 구간 기반 레이블링 기법을 그래프로 확장한 기법에 대하여 설명하고 4.2에서는 그것을 저장하기 위한 자료 구조와 알고리즘을 제안한다. 4.3에서는 온톨로지의 변경을 가정했을 때의 점진적인 갱신에 대해 기술하고 4.4에서 이론적인 분석을 한다.

4.1 그래프에 대한 구간 기반 레이블링 [10]

그래프가 하나의 연결 컴포넌트(connected component)로 구성되어 있다고 가정한다(forest의 경우는 가상의 노드를 루트로 만들면 동일해짐).

이때 레이블링의 절차는 다음과 같다. 레이블 포맷: (start_number, end_number)

1. 그래프를 깊이-우선 탐색(depth-first search)하면서 start_number와 end_number를 동일한 값으로 할당한다.
2. 자식 노드를 모두 방문하고 부분적인 루트를 방문하게 되면 자식 노드의 start_number중에 제일 큰 값을 end_number로 할당한다.
3. 동일한 과정을 전체 그래프의 루트로 돌아올 때 까지 반복한다.

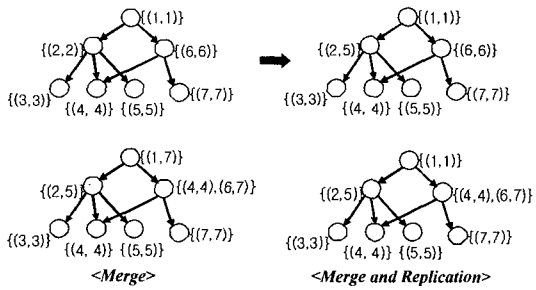

그림 6 구간 기반 레이블링 과정

결과적으로 한 노드를 구별하는 구별자(id)는 그 노드의 레이블에서 start_number가 된다. 이때, 중요한 것은 트리 간선이 아닌 그래프 간선에 대해서는 해당 자식 노드의 레이블 값을 추가하게 된다. 따라서 트리와 달리 일반적인 그래프에서는 부모가 여러 개인 노드가 있을 수 있으므로 노드의 레이블은 구간(interval)들의 집합으로 표현되고, 전체 레이블이 차지하는 공간은 평균적으로 원래 그래프와 같은 O(n) 임을 알 수 있다. 최악의 경우로 이등분할 그래프(bipartite graph)에서의 레

이블링은 $O(n^2)$이 될 수 있으나, 일반적인 온톨로지에서는 이런 경우를 가정하기 힘들다. 이러한 레이블링이 주는 장점은 그래프 탐색을 하지 않고도 레이블 정보만 보고 두 노드 사이의 연결 가능성(Reachability)을 결정할 수 있다는 점이고, 이는 원래 $O(n^2)$인 트랜지티브 클로저 그래프를 O(n)으로 압축하여 표현한 효과를 가져온다. 예를 들어 루트 노드 (1,7)과 단말 노드 (5,5)는 루트 노드의 구간이 단말 노드의 구간을 포함하므로 루트 노드에서 단말 노드로 가는 경로가 존재함을 의미하게 된다. 온톨로지 에서의 트랜지티브 클로저 질의는 단순한 접근 가능성 결정 보다는 한 노드에서 갈 수 있는 모든 노드를 구하는 것이 주가 되는데, 이는 적절한 자료구조를 이용하여 탐색범위를 최소화 할 수 있으며, 4.2절에서 설명한다.

4.2 자료 구조와 알고리즘

트랜지티브 클로저에 대한 탐색은 구간들에 대한 리스트에 대해 start_number에 B^+-tree 인덱스를 생성한 후 효과적으로 찾을 수 있다. 이때 리스트의 원소는 구간과 해당 노드의 URI를 쌍으로 갖고, 이에 대한 구현은 관계형 데이타 베이스를 이용할 수도 있고, 직접 인덱스를 구현할 수도 있다.

예를 들어 그림 7에서 (1,7)에 해당하는 노드의 모든 트랜지티브 하위 클래스를 찾기 위해서는 이미 레이블링 정보가 저장되어 있는 리스트를 탐색하면서 (2,5)에 포함되는 구간에 해당하는 URI를 찾으면 답이 된다. 이때 리스트가 start_number로 정렬되어 있다는 조건을 이용하여 start_number >= 2인 노드만을 탐색하면서 end_number<= 5인 노드를 찾기 때문에 전체 리스트에 대한 탐색보다 효율적이다.

또한 Gene Ontology의 예에서 보듯, 온톨로지에서의 트랜지티브 관계는 *is_a* 이외에도 *part_of*나 *develops_from* 등 좀 더 세분화 된 관계들이 존재 할 수 있는데, 3.3절에서 언급한 바와 같이 이러한 모든 관계가 OWL 스펙에서 subClassOf로 표현되기 위해서는 리덕션을 사용해야 하고, 이는 트리플 기반의 온톨로지 저장소에서 더 많은 트리플 탐색의 비용을 가져 온다.

따라서 레이블링을 하는 전처리 과정에서 온톨로지 내에 존재하는 모든 트랜지티브 릴레이션에 대해 각각 리스트를 유지하고 질의 처리시에 이용한다면, is_a 관계와 똑같은 비용으로 part_of에 대한 트랜지티브 클로저를 찾을 수 있다.

```
listSubclasses(target)
{
    for i = target.start to target.end
        find node of i
        add to result
    return result
}
```

```
listSupersubclasses(target)
{
    for each node s.t. node.end >= target.end
        if node.start <= target.start
            add to result
    return result
}
```

```
getNCA(target1, target2)
{
        let target1 to have larger postorder number
        for each node s.t. node.end >= target1.end
            if node.start <= target1.start and
                node.start <= target2.start
                return node
}
```

그림 9 트랜지티브 클로저 질의 알고리즘

4.3 점진적인 갱신

온톨로지는 사실상 스키마의 확장이고 관계형 모델에서의 스키마는 거의 변하지 않는다고 가정할 수 있지만,

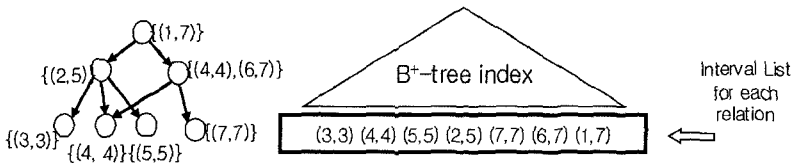

그림 7 레이블링된 데이타에 대한 저장 자료 구

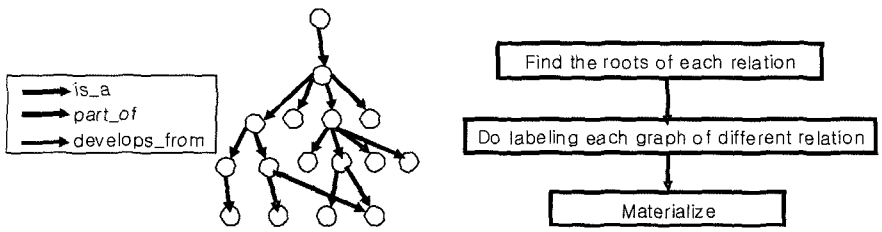

그림 8 온톨로지의 여러 관계에 대한 레이블링 고려

온톨로지의 경우는 비교적 갱신이 빈번히 일어날 수 있다. 예를 들어 야후나 ODP 카탈로그의 경우만 보더라도 필요에 의해 새로운 분류항목이 언제든지 추가 될 수 있어야 하고, 이때 4.1절의 레이블링을 그때 마다 다시 하는 것은 효율적이지 못하므로 점진적인 갱신이 필요하다. 사실 우리는 start_number를 반드시 연속적인 정수로 줄 필요는 없다. 즉 적당한 간격을 두어 할당하면 전체적으로 레이블링을 새로 하지 않고 새로운 노드에만 비어 있는 레이블을 할당할 수 있게 된다. 갱신을 삽입과 삭제의 경우로 나누어 생각해 보면, 삭제는 해당 노드를 그냥 지우면 되고, 삽입의 경우는 비어 있는 start_number값을 사용하면 해결된다. 그림 10은 gap = 10 일 때의 노드의 삽입을 보여주고 있다.

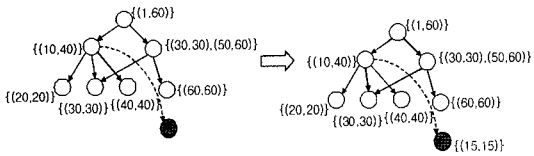

그림 10 새로운 노드가 삽입 될 때

삽입과 삭제를 고려하는 경우에는 어떤 클래스의 모든 하위 클래스를 구할 때 레이블 구간에 해당하는 그 하위 클래스들이 존재하지 않는 경우도 발생하게 된다. 예를 들어 그림 10에서 (10,40)의 하위 클래스는 노드 10부터 40까지에 해당하지만 실제로는 4개밖에 존재하지 않는다. 하지만 노드 ID에 대한 실제 URI는 해쉬 테이블에 저장되어 있고, 이런 경우는 해쉬 테이블에 해당 키값이 없는 경우이므로 상수 시간에 모든 하위 클래스를 찾을 수 있는 것에는 변함이 없다.

4.4 이론적 분석

4.4.1 공간 효율성

Gene Ontology 시스템처럼 모든 트랜지티브 클로저 그래프의 간선을 미리 계산해서 저장한다면 $\sum_{k=1}^{n} k$ = $O(n^2)$이 된다. Jena의 경우 그래프에 대한 트랜지티브 리덕션을 구하지만, 복잡한 네트웍과는 달리 온톨로지는 이로 인해 제거되는 간선의 수는 많지 않고 결국 O(n)의 공간 복잡도를 갖는다. 구간 기반 레이블링을 이용한 우리의 접근 방법도 노드 당 평균적으로 1개의 레이블을 유지하므로 O(n)의 공간 복잡도를 갖는다. 여기서 n은 트리플의 수 또는 노드의 수를 말한다.

4.4.2 시간 효율성

한 클래스의 모든 서브 클래스를 찾는 질의를 생각했을 때 메모리에서의 처리를 가정한다면 Jena의 경우 O(n)이 되는데 이는 답이 되는 노드를 찾기 위해서는 그 노드를 직접 방문해서 찾기 때문이다. 반면에 구간 기반 레이블링을 이용한 우리의 접근 방식은 그 클래스의 레이블만 보면 마치 해쉬 함수 처럼 답이 되는 클래스를 찾을 수가 있으므로 O(1)이 된다.

그림 11에서 Jena의 경우는 DFS로 재귀적으로 모든 해당 노드를 탐색하게 되고 이는 결국 답이 되는 노드의 수가 n이라고 할 때 O(n)의 시간 복잡도를 갖게 된다. 구간 기반 레이블링을 이용한 우리의 접근 방법은 한 노드의 레이블이 여러 구간이 될 수 있으나 결국 상수이고, 따라서 수행 시간도 O(1)의 시간 복잡도를 갖게 된다.

하지만, 모든 상위 클래스를 구하는 질의는 레이블만 보고 결정될 수 없기 때문에 이론적으로 Jena와 같은 O(n)이고, 이는 NCA의 경우도 마찬가지다. 하지만 트랜지티브 클로저 질의는 트랜지티브하게 갈 수 있는 모든 하위 클래스를 구하는 것이고, 상위 클래스에는 해당하지 않는다. 또한 현재 Jena는 메모리에서의 처리를 가정하고 있으므로, 만약 트랜지티브 클로저 질의 처리에 필요한 트리플들을 모두 메모리로 올릴 수 없을 정도의 대용량의 온톨로지를 가정한다면, 결국 Jena의 방식은 Gene Ontology 시스템의 방식과 큰 차이가 없게 된다.

또한 갈수록 복잡해져 가는 온톨로지에서 한 노드당 다수의 레이블이 존재하게 되었을 때에도 그래프 레이블링 기법은 효율적인데, 이는 그래프 레이블링 기법의 시간 복잡도가 노드의 fan-out에 영향을 받기보다는 전체 노드의 수에 의존하기 때문이다.

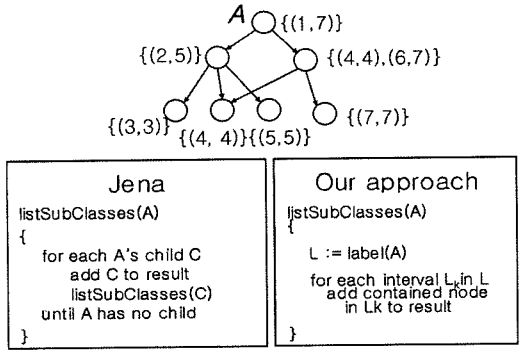

그림 11 트랜지티브 하위 클래스를 찾는 알고리즘

5. 실 험

실험은 Pentium Ⅲ 1.1Ghz CPU, 512MB 메모리, Windows XP 운영체제, 그리고 Jena2.1과 MySQL 4.0.20에서 수행하였다. 실험에 쓰인 데이타는 Gene Ontology의 term-db로서 이는 gene_product 정보를 포함하고 있는 association-db와는 달리 순수한 스키마

정보만을 가지고 있다. GONG[13] 프로젝트에서 제작한 DAML 포맷의 파일을 OWL로 변환한 후 실험 데이타로 사용하였다. 전체적으로 14000여개 정도의 클래스를 가지고 최대 깊이(depth) 13의 구조를 가지고 있는데 한 노드에서 나가는 간선의 수(fan-out)가 상당이 크고 여러 부모를 갖는 노드(term)들이 많기 때문에 트랜지티브 클로저 질의에 대한 실험용으로 적당하다고 볼 수 있다. 관계(relationship)는 19000 여개로서, is_a가 17000 여개이고 part_of가 2000 여개이다(표 1).

실험 방법은 세가지 방식을 네 가지 질의에 대해 비교하는 식으로 진행하였다.

세가지 방식은 naïve 방식과 Jena의 방식, 그리고 본 논문에서 제시하는 구간 기반 레이블링을 이용한 방식을 의미하고, 네 가지 질의는 표 2에서 보는 바와 같이 subclass(is_a), subclass(part_of), superclass, Nearest Common Ancestor(NCA)에 대해 수행하였다.

그림 12에서 전처리 시간을 비교해 보면, naïve 방식은 전처리 시간이 필요 없고 Jena에 비해서 본 논문의 방식이 4배 정도의 시간을 필요로 하지만, 이는 충분히 가능한 시간이다. 그림 13은 응답 시간에 대한 실험 결과로서, Jena가 메모리 처리 방식 위주이므로 그것을 naive방식처럼 수정해서 디스크 기반의 실험을 추가하였다. 첫번째 차트는 메모리 버전에 대한 결과이고 두번째 차트는 Jena를 naïve 방식처럼 수정하고 본 논문에서 제시하는 기법을 디스크 기반으로 했을 때의 응답시간에 대한 결과이다.

결과에서 보듯이 naive방식은 많은 SQL 문장으로 인해 디스크 I/O가 많이 발생하므로 규모가 큰 ontology

표 1 실험에 사용된 Gene Ontology 스펙

	Molecular function	Biological process	Cellular component	Total
Term	5399	7309	1304	14012
Edge	6856	11202	1644	19702

표 2 실험에 사용된 테스트 질의

Q1	Find all(is_a) subclasses of one class
Q2	Find all(part_of) subclasses of one class
Q3	Find all subclasses of one class
Q4	Find the nearest common ancestor of two classes

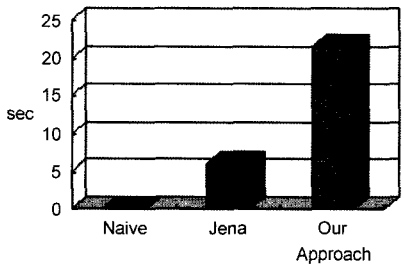

그림 12 전처리 시간 결과

에 대해서는 거의 불가능해 보이는 반면 Jena나 우리의 방식은 비교적 작은 응답 시간을 보이고 있는데 특히 엄밀한 의미의 트랜지티브 클로저 질의에 해당하는 Q1과 Q2에 대해 본 논문에서 제안한 방식이 훨씬 작은 응답 시간을 보이고 있다. Q3는 모든 상위 클래스를 구하는 질의에 해당하고 이는 4.4의 시간 복잡도 분석에서

표 3 실험 결과

	전처리시간 (sec)	응답시간 (sec)							
		Memory version				Disk version			
		Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
Naive	0	63	43	62	35				
Jena	6	23	10	22	5	33	20	30	15
Our	21	1	1	12	1	5	4	21	5

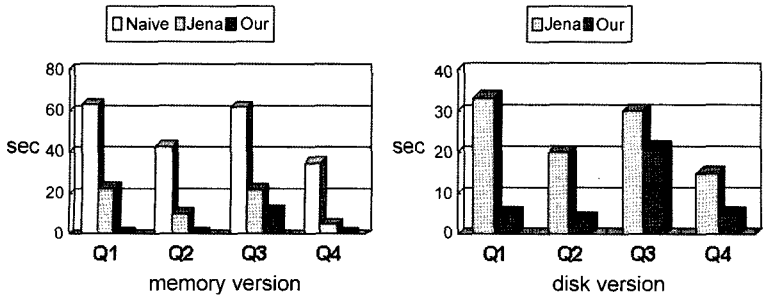

그림 13 질의 응답 시간 결과

보듯이 본 논문에서 제시하는 기법의 경우에도 답이 되는 모든 노드들을 리스트에서 찾아야 하므로 응답시간에 있어서 Jena의 방식과 큰 차이를 보이지 않고, Q4의 NCA를 찾는 질의는 Q3의 특수한 경우라고 볼 수 있다.

6. 결 론

본 논문에서는 그래프 레이블링 기법을 이용하여 그래프 기반의 데이타인 온톨로지 에서의 트랜지티브 클로저 질의에 대해 공간, 시간 효율적인 처리와 여러 트랜지티브 관계를 고려한 처리 방법에 대해 제안하였다. 본 논문의 접근 방법은 구간 기반 레이블링을 이용한 트랜지티브 클로저 그래프 압축과 실체화를 통해 대용량의 온톨로지에 대한 상당한 크기의 트랜지티브 클로저 그래프 정보를 적은 I/O로 메모리 내에서 처리할 수 있게 한다. Gene Ontology는 대용량의 데이타와 복잡하고 다양한 관계(relationship)로 인해 좋은 실험 환경을 제공하고, 본 논문이 제안한 기법은 빈번한 트랜지티브 클로저 질의들에 대한 성능을 개선시킬 수 있음을 대표적인 온톨로지 저장소인 Jena의 방식과 비교하여 이론적으로 그리고 실험적으로 보이고 있다.

또한 갱신 및 관리 비용에 있어서도 실체화된 자료의 갱신은 온톨로지의 갱신되는 정도를 고려해 처음에 설정된 간격에 의해 점진적인 갱신이 쉽게 가능하다.

본 논문에는 다루는 내용의 범위는 온톨로지의 추론 기능 중에 가장 기본적인 트랜지티브 추론에 대한 내용으로 W3C의 표준 온톨로지 언어인 OWL의 기능의 일부에 해당하지만, 트랜지티브 클로즈 질의 자체는 빈번하게 발생하면서 그 비용이 대단히 크다는데 큰 의의가 있고, 앞으로 다양하고 OWL의 기능을 풍부하게 사용하는 도메인이 많아지더라도 근본적인 요소(building block)로서 상위 시스템에 결합 될 수 있을 것이다. 추후에는 OWL의 DL(Description Logic) 기능을 충분히 활용할 수 있는 프레임웍을 가정한 상태에서의 이의 최적화에 관한 연구가 필요하다고 생각된다.

참 고 문 헌

[1] Gene Ontology(GO) Consortium: http://www.geneontology.org

[2] K. Wilkinson, C. Sayers, H. Kuno, D. Reynolds. "Efficient RDF Storage and Retrieval in Jena2," *SWDB* 2003.

[3] S. Alexaki, V. Christophides, G. Karvounarakis, D. Plexousakis. "The ICS-FORTH RDFSuite: Managing Voluminous RDF Description Bases," *Semantic Web Workshop* 2001.

[4] W3 Consortium (W3C): http://www.w3c.org

[5] V. Christophides, G. Karvounarakis, D. Plexousakis. "Optimizing Taxonomic Semantic Web Quries using Labeling Schemes," *Web Semantics: Science, Services, and Agents on the World Wide Web*, Vol. 1(2), 2004: 207-228.

[6] R. Agrawal, A. Borgida, H. V. Jagadish. "Efficient Management of Transitive Relationships in Large Data and Knowledge Bases," *Proceedings of the ACM SIGMOD International Conference on the Management of Data*, 1989:253-262.

[7] C. Zhang, J. Naughton, D. DeWitt, Q. Luo, G. Lohman. "On Supporting Containment Queries in Relational Database Management Systems," *Proceedings of the ACM SIGMOD International Conference on the Management of Data*, 2001:425-436.

[8] J. Broekstra, A. Kampman, F.V. Harmelen. "Sesame: An Architecture for Storing and Querying RDF Data and Schema Information," Semantics for the WWW, 2001.

[9] ODP Project: http://www.dmoz.org

[10] J. Kim, H.-J. Kim, "Efficient Processing of Regular Path Joins using PID," Information and Software Technology, 2002.

[11] I. Horrocks, S. Tobies, "Optimisation of Terminological Reasoning," Proceedings of International Workshop in Description Logics 2000.

[12] G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, "RQL: A Declarative Query Language for RDF," Proceeding of International World Wide Web Conference 2002.

[13] Gene Ontology Next Generation(GONG) Project: http://gong.man.ac.uk/

김 종 남

2003년 서울대학교 컴퓨터공학부(학사) 2005년 서울대학교 전기.컴퓨터공학부(석사). 관심분야는 데이타베이스, 시멘틱 웹, 온톨로지, XML

정 준 원

2000년 동국대학교 컴퓨터공학과(학사) 2003년 서울대학교 전기.컴퓨터공학부(석사). 2003년~현재 서울대학교 전기.컴퓨터공학부 박사과정 재학 중. 관심분야는 시멘틱웹, 온톨로지, XML, 데이타베이스

민 경 섭

1995년 한국항공대학교 전자계산학과(학사). 1997년 서울대학교 컴퓨터공학과(석사). 2005년 서울대학교 인지과학과(박사). 관심분야는 데이타베이스, XML, 지식 검색

김 형 주

1982년 서울대학교 전산학과(학사). 1985년 미국 텍사스 대학교 대학원 전산학(석사). 1988년 미국 텍사스 대학교 대학원 전산학(박사). 1988년 5월~1988년 9월 미국 텍사스 대학교 POST-DOC 1988년 9월~1990년 12월 미국 조지아 공과대학 조교수. 1991년~현재 서울대학교 컴퓨터공학부 교수