# RDF 데이타에 대한 효율적인 키워드 검색 기법
## (An Efficient Keyword Search Method on RDF Data)

김 진 하 † 송 인 철 †† 김 명 호 †††
(Jinha Kim) (Inchul Song) (Myoung Ho Kim)

**요 약** 최근 문서나 웹 페이지뿐만 아니라 관계형 데이타나 XML 데이타, RDF 데이타 같은 구조화된 데이타에 대해서도 키워드 검색을 지원하고자 하는 연구가 활발히 진행되고 있다. 본 논문에서는 RDF 데이타에 대한 효율적인 키워드 검색 기법을 제안한다. 제안하는 기법은 먼저 RDF 데이타의 크기를 줄여 키워드 검색 성능을 높이고 키워드 검색 결과로 관련 있는 정보를 함께 반환해 주기 위해 RDF 데이타에서 관련 있는 노드와 에지를 묶어 새로운 RDF 그래프를 생성한다. 또한 키워드 검색 과정에서 키워드 검색의 결과를 정렬하기 위해 RDF 데이타 그래프의 노드와 에지에 키워드와의 연관도를 부여할 때, RDF 온톨로지 데이타의 특성을 활용함으로써 보다 사용자의 의도에 부합하는 키워드 검색 결과를 반환한다. 실제 RDF 데이타를 사용한 성능 비교 결과는 제안하는 기법이 RDF 데이타의 크기를 최대 2배까지 줄이고 기존 기법에 비해 검색 속도가 최대 5배 빠르다는 것을 보여준다.

**키워드** : 키워드 검색, RDF, 온톨로지

***Abstract*** Recently, there has been much work on supporting keyword search not only for text documents, but also for structured data such as relational data, XML data, and RDF data. In this paper, we propose an efficient keyword search method for RDF data. The proposed method first groups related nodes and edges in RDF data graphs to reduce data sizes for efficient keyword search and to allow relevant information to be returned together in the query answers. The proposed method also utilizes the semantics in RDF data to measure the relevancy of nodes and edges with respect to keywords for search result ranking. The experimental results based on real RDF data show that the proposed method reduces RDF data about in half and is at most 5 times faster than the previous methods.

**Key words** : keyword search, RDF, ontology

## 1. 서 론

인터넷의 발전과 대중화로 인해 인터넷 상에 방대한 양의 데이타가 홀러넘치게 되었다. 인터넷에서 이러한 방대한 양의 데이타를 검색하는 기법으로 키워드 검색이 널리 사용되어 왔다. 최근에는 관계형 데이타베이스나 XML 데이타베이스에 저장된 구조화된 데이타에 대해서도 키워드 검색을 지원하고자 연구가 활발히 진행되고 있다. 구조화된 데이타에 대한 키워드 검색에서는 사용자가 키워드를 입력하면 구조화된 데이타로부터 키워드를 모두 포함하는 하부구조를 추출해서 반환한다. 예를 들어, 관계형 데이타베이스에 대한 키워드 검색에서는 사용자가 키워드를 입력하면 외래키 참조(foreign key reference)를 따라 튜플들을 조인한 결과 중에서 모든 키워드를 포함하는 결과를 반환한다[1-5]. 트리 구조나 그래프 구조를 가지는 XML에 대한 키워드 검색 기법에서는 모든 키워드를 포함하는 서브트리 혹은 서브그래프를 반환한다[6-13].

구조화된 데이타에 대해 키워드 검색을 지원하게 되면 다음과 같은 장점이 있다. 첫째, 사용자는 데이타를

· 이 논문은 2007년도 정부(교육과학기술부)의 재원으로 한국과학재단의 지원을 받아 수행된 연구임(No. R0A-2007-000-10046-0)

† 정 회 원 : NHN 서비스관리시스템팀 품질관리시스템개발팀
sweetorgel@gmail.com
†† 학생회원 : KAIST 전산학과
icsong@dbserver.kaist.ac.kr
††† 종신회원 : KAIST 전산학과 교수
mhkim@dbserver.kaist.ac.kr

논문접수 : 2008년 2월 21일
심사완료 : 2008년 9월 30일



정보과학회논문지: 데이타베이스 제35권 제6호(2008.12)

검색 할 때 데이타의 내부 구조를 알 필요가 없다. 예를 들어, 기존에 관계형 데이타베이스에서 질의를 작성할 때는 테이블 이름이나 테이블의 칼럼 이름에 관한 정보를 미리 알고 있어야 했지만 키워드 검색에서는 그럴 필요가 없다. 둘째, 사용자는 특별한 질의 언어를 배울 필요가 없다. 반면에 관계형 데이타베이스를 질의하기 위해서는 SQL을 배워야 하고 XML 데이타베이스를 질의하기 위해서는 XQuery를 배워야 한다.

최근 기계도 이해할 수 있는 차세대 지능형 웹인 시맨틱 웹(Semantic Web)이 출현하면서 새로운 종류의 구조화된 데이타인 RDF 데이타가 많이 생겨나고 있다. RDF(Resource Description Framework)[14]는 컴퓨터가 이해할 수 있도록 자원(resource)에 관한 정보 혹은 속성을 그래프 구조를 통해 기술하는 언어다. 여기서 자원이란 웹 페이지, 사람, 문서, 데이타베이스 혹은 추상적인 개념 등 다양한 식별 가능한 대상을 가리킨다. RDFS(RDF Vocabulary Description Language)[15] 는 어떤 종류의 자원을 설명하고 있는 것인지, 즉 어떤 클래스에 속하는 자원을 설명하고 있는 것인지, 그리고 특정 클래스에 속하는 자원은 어떤 속성을 가질 수 있는지를 기술하는 언어다. 본 논문에서는 RDF와 RDFS를 사용해 생성한 데이타를 통칭해 RDF 데이타라고 부른다. RDF 데이타에 대한 질의 언어로는 원하는 데이타를 그래프 패턴 형태로 기술하는 SPARQL[16]이 있다.

RDF 데이타에 대해서도 키워드 검색을 지원하게 되면 SPARQL이라는 질의 언어를 배우지 않아도 사용자들이 손쉽게 RDF 데이타를 검색할 수 있게 되어 시맨틱 웹의 사용을 촉진시킬 수 있을 것이다. 그러나, 기존의 구조화된 데이타에 대한 키워드 검색 기법을 RDF 데이타에 그대로 적용하는 데에는 몇 가지 문제가 있다.

먼저, 관계형 데이타베이스에 대한 키워드 검색 기법은 관계형 데이타베이스에 특화된 기법으로 RDF 데이타에 적용할 수 없다. 또한, XML 데이타베이스에 대한 키워드 검색 기법 중 XML을 트리 구조로 다루는 기법은 그래프 구조를 가지는 RDF 데이타에 적용할 수 없다. 그 밖에 나머지 그래프 구조 데이타에 대한 키워드 검색 기법은 클래스 정보와 같은 RDF 온톨로지 데이타의 특성을 고려하지 못하므로 사용자가 원하는 결과를 제대로 반환하지 못한다.

최근 RDF 데이타에 대한 키워드 검색을 지원하고자 하는 연구가 이루어지고 있지만 이들 연구에서 제안하는 기법은 단순히 RDF 그래프 데이타에서 키워드와 관련 있는 노드를 추출하여 반환하거나[17], 최종 결과로 SPARQL 질의를 생성할 뿐 기존 그래프 기반 키워드 검색 기법과 유사하다[18,19].

본 논문에서는 RDF 데이타의 특성을 고려한 RDF 데이타에 대한 효율적인 키워드 검색 기법을 제안한다. 제안하는 기법에서는 먼저 전처리 단계로 온톨로지 축약을 수행한다. 전처리 단계에서는 RDF 온톨로지 데이타의 크기를 줄여 키워드 검색 속도를 향상시키고 키워드 검색 결과로 관련 정보를 함께 반환하기 위해 원본 RDF 온톨로지 그래프에서 서로 관련 있는 노드와 에지를 묶어 축약된 온톨로지를 구성한다. 다음 단계는 사용자가 키워드를 입력하면 전처리 단계를 통해 생성된 축약된 온톨로지에 대해 실제 키워드 검색을 수행하는 키워드 검색 단계다. 키워드 검색 단계에서는 먼저 키워드 검색 결과 정렬을 위해 RDF 데이타 그래프의 노드와 에지에 키워드와의 연관도를 부여한다. 기존 기법과는 다르게 제안하는 기법에서는 RDF 온톨로지 데이타의 특성을 활용하여 키워드 연관도를 더욱 정확하게 계산한다. 마지막으로 키워드와의 연관도가 부여된 RDF 데이타 그래프에 대해 기존의 그래프 기반 키워드 검색 알고리즘을 실행하여 키워드 검색 결과를 얻고 결과를 키워드와의 연관도에 따라 정렬하여 사용자에게 반환한다.

본 논문에서는 실제 RDF 데이타를 사용한 실험을 통해 제안하는 기법과 기존 그래프 기반 키워드 검색 기법 간의 성능을 비교하였다. 성능 비교 결과는 제안하는 기법이 온톨로지 축약을 통해 RDF 데이타의 크기를 최대 2배까지 줄이고 기존 기법에 비해 검색 속도가 최대 5배 빠르다는 것을 보여준다.

본 논문의 구성은 다음과 같다. 먼저 2장에서는 관련 연구를 설명한다. 다음으로 3장에서는 제안하는 기법의 데이타 모델과 시스템 구조를 설명한다. 4장에서는 전처리 단계인 온톨로지 축약 과정을 설명하고 5장에서는 키워드 검색 과정을 설명한다. 6장에서는 성능 평가 결과를 제시하고 마지막으로 7장에서는 결론을 내린다.

## 2. 관련 연구

구조화된 데이타에 대해 키워드 검색 지원하는 연구가 최근에 활발히 이루어지고 있다. 관계형 데이타베이스에서 키워드 검색을 지원하는 연구는 크게 스키마 기반 키워드 검색 기법과 그래프 기반 키워드 검색 기법으로 나눌 수 있다. 스키마 기반 기법은 외래키 참조를 따라 튜플들을 조인한 결과 중 모든 키워드를 포함하는 결과를 어떻게 생성할 것인지를 관계형 데이타베이스의 테이블 스키마 정보에 기반해 결정한다[1-5]. 스키마 기반 기법은 관계형 데이타베이스의 특성을 활용한 것으로서 RDF 데이타에 적용하기 힘들다. 그래프 기반 기법은 각 튜플을 그래프의 노드로 나타내고 두 튜플이 주키와 외래키의 관계를 따라 조인이 되는 경우 두 튜플을 나타내는 노드 간에 에지를 추가하여 관계형 데이타베이스를 그래프 구조로 표현한다[6-8]. 그래프 기반

기법은 클래스 정보와 같은 RDF 데이타의 특성을 고려하지 못한다.

XML 데이타베이스에 대한 키워드 검색 기법은 트리 기반 기법과 그래프 기반 기법으로 나눌 수 있다. 트리 기반 기법은 XML을 트리 구조로 나타내고 모든 키워드를 포함하는 서브트리를 찾는다[9,11-13]. 트리 기반 기법은 그래프로 모델링되는 RDF 데이타에 적용할 수 없다. 그래프 기반 기법에서는 ID-IDREF를 고려하여 XML을 그래프 구조로 나타낸다[10]. 그래프 기반 기법은 관계형 데이타베이스에서 그래프 기반 기법과 마찬가지로 클래스와 같은 RDF 데이타의 특성을 고려하지 못한다.

RDF 데이타에 대한 키워드 검색 기법은 노드 검색 기법과 SPARQL 생성 기법으로 나눌 수 있다. 노드 검색 기법은 단순히 RDF 데이타에서 키워드와 관련 있는 노드를 추출해 반환한다[17]. SPARQL 생성 기법은 키워드 검색 결과로 SPARQL을 생성해 낸다[18,19]. SPARQL 생성 기법은 최종 결과로 SPARQL을 생성해 낼 뿐 기존의 그래프 기반 키워드 검색 기법과 크게 다르지 않다.

## 3. 데이타 모델 및 시스템 구조

### 3.1 데이타 모델

본 논문에서는 RDF 데이타를 그래프로 모델링한다[20]. RDF 그래프는 주어(subject), 목적어(object), 술어(predicate)(속성(property)이라고도 부른다) 이 세 요소로 구성되는 트리플(triple)의 모음이다.[1] 트리플의 주어와 목적어는 그래프의 노드가 되고 속성은 주어 노드와 목적어 노드를 잇는 라벨이 붙은 에지(labeled edge)가 된다. RDF 그래프에서 노드는 URI 참조(URI reference), 리터럴 또는 공백(blank)이 될 수 있다. RDF 그래프에서 속성은 반드시 URI 참조여야 한다. RDF 그래프 노드 n의 텍스트 n.text는 URI 참조 노드인 경우 URI 참조를 문자열로 나타낸 것이고 리터럴 노드인 경우에는 텍스트 그 자체다. RDF 그래프 에지 e의 텍스트 e.text는 에지 라벨의 URI 참조를 문자열로 나타낸 것이다.

표 1은 "이름이 'jhkim'인 사람의 이메일은 'jinha@dbserver.kaist.ac.kr'이다"라는 정보를 RDF 트리플로

표 1 RDF 트리플의 예

| 주어 | 속성 | 목적어 |
|---|---|---|
| http://dbserver.kaist.ac.kr/people#jhkim | http://purl.org/dc/element/1.1/email | jinha@dbserver.kaist.ac.kr |

1) 본 논문에서는 술어라는 용어 대신 속성이라는 용어를 사용한다.

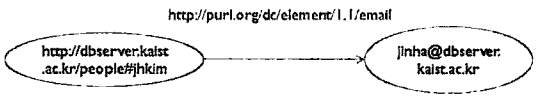



그림 1 RDF 그래프 예

나타낸 것이다. 그림 1은 이 트리플을 그래프로 그린 것이다. 트리플을 그래프로 나타낼 때 에지 방향은 주어에서 목적어 쪽을 향한다. 본 논문에서는 URI 참조를 접두어 붙은 이름(prefixed name)[21]으로 쓰거나 혼동을 주지 않는다면 접두어를 생략한 이름만으로 쓴다. 예를 들어, 표 1에 나와 있는 트리플에서 http://dbserver.kaist.ac.kr/people#jhkim은 people:jhkim으로 쓰거나 더 줄여서 jhkim으로 쓴다.

본 논문에서는 RDF 그래프를 무방향 그래프(undirected graph)로 취급한다. 일반적으로 그래프 기반 키워드 검색 기법에서는 주어진 키워드를 모두 포함하는 구조를 찾기 위해 그래프를 탐색한다. 이 때 그래프에서 특정 방향을 향하는 에지가 존재하지 않을 경우 그 방향을 따라 갈 때 도달하는 노드나 에지는 결과에 포함되지 않는다. 그러나, RDF 그래프에서는 주어진 에지의 양방향에 존재하는 정보가 모두 중요하다. 예를 들어, 그림 1에 나와 있는 RDF 그래프에서 사용자가 "jhkim"라는 키워드를 입력한 경우에는 주어 노드에서 시작해 에지 방향을 따라가서 찾을 수 있는 이메일 정보를 함께 반환하는 것이 사용자에게 유익할 것이고 사용자가 "jinha@dbserver.kaist.ac.kr"라는 이메일을 입력한 경우에는 목적어 노드에서 시작해 에지 반대 방향으로 따라가서 찾을 수 있는 이메일의 소유자에 대한 정보를 반환하는 것이 유익할 것이다.

RDFS는 클래스와 속성에 대한 정보를 기술하는 데 사용할 수 있는 어휘를 제공한다. RDFS에서 제공하는 주요 어휘에는 새로운 클래스를 정의하는 데 사용되는 URI 참조인 rdfs:Class 및 rdf:type, 새로운 속성을 정의하는 데 사용되는 URI 참조인 rdf:Property, rdfs:domain, rdfs:range 등이 있다. RDF 관련 표준 문서에서는 클래스의 인스턴스를 인디비쥬얼(individual)이라고 부른다. RDFS에서 제공하는 모든 어휘에 대한 자세한 설명은 [15]에 나와 있다.

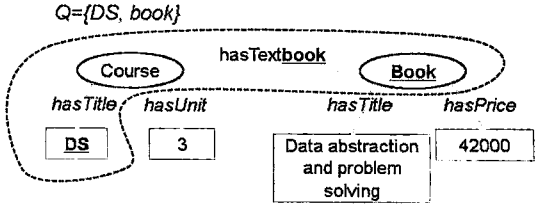



그림 2 최소 서브트리의 예

본 논문에서는 속성 중 rdfs:domain의 값으로 rdfs:Literal이나 그 하위클래스를 가지는 속성을 리터럴 속성(Literal Property: LP)이라고 부르고 나머지 속성을 비리터럴 속성(Non-Literal Property: NLP)이라 부른다. LP 속성은 목적어가 일반 텍스트인 속성이고 NLP 속성은 목적어가 일반 텍스트가 아닌 특정 클래스의 인디비쥬얼인 속성이다. 보통 LP 속성은 특정 인디비쥬얼에 대한 정보를 텍스트로 기술하는 데 사용되고 NLP 속성은 주어 인디비쥬얼과 목적어 인디비쥬얼의 관계를 기술하는 데 사용된다. 본 논문에서는 RDF 그래프를 그릴 때 주어 노드와 NLP 속성의 목적어 노드는 타원으로 그리고 LP 속성의 목적어 노드는 사각형으로 그린다.

본 논문에서는 RDF 데이타에 대해 다음 두 가지를 가정한다. 첫째, 모든 인디비쥬얼은 특정 클래스에 속한다. 둘째, 모든 속성에는 rdfs:domain과 rdfs:range가 정의되어 있다.

### 3.2 질의 시맨틱

질의 시맨틱은 기존의 그래프 기반 키워드 검색 기법에서 제안된 질의 시맨틱을 따른다[6-8]. 키워드 검색 질의 Q는 키워드 $k_1$, …, $k_m$으로 구성된다. 키워드 검색 질의 Q의 결과는 다음과 같이 정의된다. 키워드 질의 Q = ($k_1$, …, $k_m$)와 무방향 그래프 G가 주어졌다고 하자. 각 키워드 $k_i$에 대해 키워드 $k_i$를 포함하는 노드의 집합을 $S_i$라고 하자[2]. 그러면, 키워드 질의 Q의 결과는 각 $S_i$로부터 하나 이상의 노드를 포함하는 G의 최소 서브트리(minimal subtree)다. 최소 서브트리란 각 $S_i$로부터 하나 이상의 노드를 포함하는 서브트리 중에서 어떤 노드나 에지를 하나라도 제거하면 모든 키워드를 포함하지 못하게 되는 서브트리를 말한다. 그림 2는 주어진 RDF 그래프에 대한 질의 Q={DS, book}의 결과인 최소 서브트리(점선으로 둘러싸인 부분)를 보여준다.

### 3.3 시스템 구조

그림 3은 제안하는 키워드 검색 시스템의 전체 키워드 검색 과정을 도식화한 것이다. 제안하는 키워드 검색 처리 과정은 크게 전처리 단계(Pre-Processing Phase)와 키워드 검색 단계(Keyword Searching Phase)로 나뉜다.

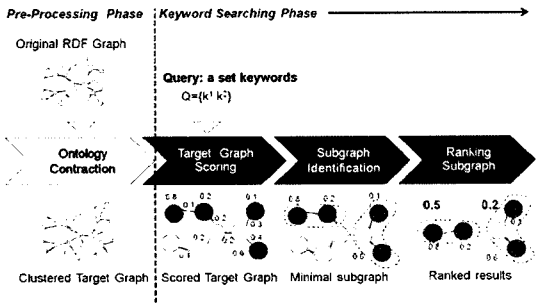



그림 3 키워드 검색 개요

2) 노드 n이 키워드 $k_i$를 포함한다는 말의 의미는 노드 n의 텍스트 n.text에 키워드 $k_i$가 나타난다는 의미다.

전처리 단계에서는 원본 RDF 그래프(Original RDF Graph)를 키워드 검색에 더 적합한 형태로 변환하는 온톨로지 축약(Ontology Contraction) 과정이 수행된다. 온톨로지 축약 과정에서는 효율적인 키워드 검색을 위해 키워드 검색의 대상이 되는 RDF 데이타의 크기를 줄이고 키워드 검색 결과로서 관련 정보를 함께 반환하기 위해, 원본 RDF 그래프에서 관련 있는 노드를 묶어 클러스터링된 대상 그래프(Clustered Target Graph)를 생성한다. 또한 클러스터링된 대상 그래프에 대해 그래프에서 나타나는 각 키워드를 키워드가 나타나는 노드나 에지에 매핑하는 역 인덱스(Inverted Index)를 생성한다.

키워드 검색 단계에서는 사용자로부터 키워드를 입력받아 키워드 검색을 수행한다. 이 단계의 첫 번째 과정은 키워드 검색 결과의 정렬을 위해 클러스터링된 대상 그래프의 노드와 에지에 각 키워드와의 연관도를 부여하는 대상 그래프 점수부여(Target Graph Scoring) 과정이다. 다음 과정은 서브그래프 식별(Subgraph Identification) 과정이다. 이 과정에서는 점수가 매겨진 대상 그래프(Scored Target Graph)에서 키워드 검색의 결과인 최소 서브트리를 찾는다. 마지막으로 서브그래프 순위 매기기(Ranking Subgraph) 과정에서는 서브그래프 식별 단계를 통해 찾아진 최소 서브트리의 순위를 매기고 순위에 따라 사용자에게 키워드 검색 결과를 정렬해서 보여준다.

## 4. 온톨로지 축약 단계

온톨로지 축약 단계에서는 원본 RDF 데이타 그래프에서 서로 관련이 있는 노드와 에지를 묶어 축약된 온톨로지를 생성한다. 온톨로지 축약은 키워드 검색에 있어서 몇 가지 장점을 제공한다.

먼저, 온톨로지 축약은 관련 있는 노드를 묶어 하나의 노드로 만들기 때문에 RDF 데이타의 크기를 줄인다. 처리해야 하는 RDF 데이타의 크기가 줄면 키워드 검색 과정에서 성능 향상을 얻을 수 있다. 둘째, 온톨로지 축약 후 관련된 정보가 모여 있게 되어 더욱 풍부한 키워드 검색 결과를 제공할 수 있다. 키워드 검색 질의 Q={DS, book}에 대한 결과를 보여주는 그림 2를 다시 보자. 그림 2에서 점선으로 둘러싸인 부분은 온톨로지 축약을 수행하지 않은 상태에서 기존 키워드 검색 기법을 사용해 얻은 결과다. 그 결과는 DS, hasTitle, Course, hasTextbook, Book을 차례로 연결하는 트리(경로)다.

여기서 사용자는 키워드 검색을 통해 DS라는 과목의 책과 관련된 일부 정보를 얻었다. 만약 사용자가 책의 제목이나 가격 같은 더욱 자세한 정보를 알고 싶을 때는 시스템에 추가적인 정보를 요청해야 한다. 반면에 온톨로지 축약을 수행하여 관련 정보를 모아 놓으면 애초에 키워드 검색 결과에 관련 정보를 포함시켜 반환해 줄 수 있다. 예를 들어, Book이라는 노드와 이 노드에 hasTitle과 hasPrice로 연결되어 있는 책 제목 및 가격 정보를 묶어 두면 키워드 검색 질의 결과로서 책의 정보가 반환될 때 책의 제목 및 가격 정보가 항상 함께 반환될 것이다.

그림 4는 온톨로지 축약 알고리즘 OntCon를 보여준다. OntCon의 입력값은 RDF 그래프를 트리플로 나타낸 트리플 집합이다. Jena[22]와 같은 RDF API에서는 RDF 데이타 그래프를 트리플 집합으로 접근할 수 있는 API를 제공한다. OntCon 알고리즘에서 트리플 t에 대해, t.subj는 주어 노드를, t.obj는 목적어 노드를, t.prop는 속성을 나타낸다. Jena RDF API는 특정 노드가 리터럴 노드인지 여부를 검사하는 기능도 제공한다.

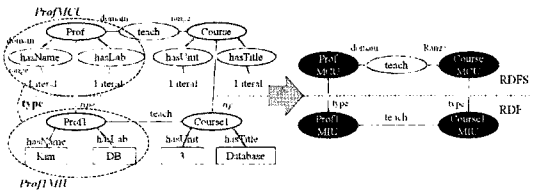
```
[입력값] RDF 그래프를 트리플로 나타낸 트리플 집합 T
[출력값] 클러스터링된 대상 그래프 G = (N, E)
[변수]
N := {} // node set of G
E := {} // edge set of G
[알고리즘]
OntCon
for each t ∈T
    if t.subj is in N
        subj <- N[t.subj]
    else
        allocate a new subj
        subj.text <- t.subj.text
        add subj to N

    if t.obj is a literal node
        // node grouping
        append t.prop.text and t.obj.text to subj.text
    else
        if t.obj is in N
            obj <- N[t.obj]
        else
            allocate a new obj
            obj.text <- t.obj.text
            add obj to N

        create an edge e = {subj, obj}
        e.text <- t.prop.text
        add e to E

return G = (N, E)
```

그림 4 온톨로지 축약 알고리즘




(a) Original graph (b) Clustered target graph

그림 5 온톨로지 축약의 예

RDF 그래프가 주어지면, OntCon 알고리즘은 주어진 RDF 그래프에 들어 있는 트리플의 개수만큼 for 문을 수행하고 종료한다. OntCon 알고리즘은 목적어 노드가 리터럴인 트리플의 주어 노드에 목적어 노드와 속성을 통합하기 때문에 원본 RDF 그래프에서 이러한 트리플마다 노드와 에지를 하나씩 줄인다. 목적어 노드가 리터럴인 트리플의 개수를 L이라고 하면, OntCon 알고리즘은 원본 RDF 그래프에서 L 개의 노드와 L 개의 에지를 줄인다. 원본 RDF 그래프에 목적어 노드가 리터럴인 트리플이 많을수록 그래프 크기는 더욱 줄어든다.

클러스터링된 대상 그래프에서 원본 RDF 그래프에서 클래스였던 노드가 주변 노드와 함께 통합된 것을 클래스 단위(MCU: Meaningful Class Unit) 노드라고 부르고 원본 RDF 그래프에서 인디비쥬얼이었던 노드가 주변 노드와 통합된 것을 의미 있는 인디비쥬얼 단위(MIU: Meaningful Individual Unit) 노드라고 부른다. 클러스터링된 대상 그래프에서는 MCU 노드와 MIU 노드는 서로 간에 비리터럴 속성 에지로 연결된다. 클러스터링된 대상 그래프에서 MIU 노드 i가 MCU 노드 c와 rdf:type 속성으로 연결되어 있는 경우 MIU 노드 i의 타입은 MCU 노드 c다.

그림 5는 온톨로지 축약의 예를 보여준다. 그림 5(a)는 Prof(교수)가 Course(수업)을 강의한다는 정보를 나타내는 원본 RDF 그래프다. 그림 5(b)는 클러스터링된 대상 그래프의 모습을 보여준다.

## 5. 키워드 검색 단계

키워드 검색 단계는 사용자 키워드를 입력하였을 때 전처리 단계를 통해 생성된 클러스터링된 대상 그래프를 대상으로 키워드 검색을 수행하는 단계다. 키워드 검색 단계는 키워드 검색 결과 정렬을 위해 클러스터링된 대상 그래프에 키워드와의 연관도를 부여하는 대상 그래프 점수부여 과정, 키워드 검색의 결과인 최소 서브트리를 찾는 서브그래프 식별 과정, 그리고 최종적으로 사용자에게 키워드 검색 결과를 정렬해서 보여주는 서브그래프 순위 매기기 과정으로 구성된다.

키워드 검색의 대상인 클러스터링된 대상 그래프에는

MCU 노드와 MIU 노드, 이렇게 두 종류의 노드가 존재한다. 이 중 MCU 노드는 클래스 정보를 나타내며 MIU 노드가 어떤 클래스에 속하는지를 설명하는 태그의 역할을 수행한다. 반면에 MIU 노드는 자원에 대한 실제 설명을 담고 있다. 사용자가 관심 있는 정보는 MIU 노드에 들어 있는 자원에 대한 설명 부분이다. 따라서 본 논문에서는 키워드 검색의 대상을 클러스터링된 대상 그래프에서 MIU 노드로 한정한다. 키워드 검색 대상에는 포함되지 않지만 클러스터링된 대상 그래프에서 MCU 노드 정보는 키워드 검색 단계 중 대상 그래프 점수부여 과정에서 활용된다.

### 5.1 대상 그래프 점수부여

키워드 검색 단계의 첫 번째 과정은 대상 그래프 점수부여 과정이다. 이 과정에서는 클러스터링된 대상 그래프의 노드와 에지에 각 키워드와의 연관도를 부여한다. 클러스터링된 대상 그래프의 노드에 점수를 부여할 때는 키워드 검색의 대상이 되는 MIU 노드에만 점수를 부여한다.

#### 5.1.1 노드 점수 계산

본 논문에서는 키워드 k와 MIU 노드 i의 연관도를 구할 때 다음의 두 가지 요소를 고려한다.

- 키워드 발생빈도(KF: Keyword Frequency): MIU 노드 i의 텍스트 내에 키워드 k가 나타나는 횟수를 의미한다. KF는 MIU 노드 i가 키워드 k를 얼마나 잘 설명하는지를 나타내는 수치다. MIU 노드 i의 KF가 클수록 키워드 k는 MIU 노드 i와 연관성이 높다.
- 노드 발생빈도(NF: Node Frequency): MIU 노드 i가 MCU 노드 c를 타입으로 가진다고 하자. 노드 발생빈도는 MCU 노드 c를 타입으로 가지는 MIU 노드 중 키워드 k가 나타나는 노드 수를 의미한다. NF는 키워드 k가 키워드 k에 관련 있는 MIU 노드와 관련 없는 MIU 노드를 얼마나 잘 구분할 수 있는지를 나타내는 수치다. 키워드 k가 MIU 노드 i와 동일한 MCU 노드를 타입으로 가지는 다른 MIU 노드에 많이 나타나는 경우 키워드 k는 MIU 노드 i와 연관성이 낮다.

상기 두 가지 요소를 고려한 MIU노드 i의 키워드 k와의 연관도 점수 $Score^k(i)$는 다음과 같다.

$$Score^k(i) = F_k(i) \times INF_k(i) \quad (1)$$

$$F_k(i) = \frac{KF_k(i)}{\max_j KF_k(j)} \quad (2)$$

$$INF_k(i) = \frac{N(i)}{NF_k(i)} \quad (3)$$

식 (1)에서 $F_k(i)$는 MIU 노드 i의 텍스트에 키워드 k가 나타난 횟수($KF_k(i)$)를 0에서 1.0 사이 값으로 표준화(키워드 k의 최대 출현 횟수인 $\max_j KF_k(j)$로 나눔)한 것이다. $INF_k(i)$는 MIU 노드의 타입인 MCU 노드 c와 동일한 타입을 가지는 MIU 노드 중 키워드 k가 나타나는 노드 수($NF_k(i)$)를 MCU 노드 c를 타입으로 가지는 MIU 노드 수(N(i))로 나눈 값을 뒤집은 역 노드 발생빈도(Inverted Node Frequency)다. 본 논문에서는 제안하는 노드 점수 계산 방법을 KF-INF라 부른다.

제안하는 KF-INF 노드 점수 계산 방법은 기존 IR 분야에서 널리 사용되는 TF-IDF(Term Frequency-Inverted Document Frequency) 방법[23]과 유사하다. TF-IDF 방법은 문서에 키워드와의 연관도 점수를 부여하는 방법인 반면 KF-INF 방법은 클러스터링된 대상 그래프에서 노드에 키워드와의 연관도 점수를 부여하는 방법이다. TF-IDF에서 TF 요소는 KF-INF에서 KF 요소에, IDF 요소는 INF 요소에 대응된다.

본 논문에서 제안하는 KF-INF 방법이 기존의 TF-IDF 방법과 다른 점은 KF-INF 방법에서는 도메인별 키워드 중요도를 고려한다는 점이다. 주어진 키워드가 관련 있는 문서와 관련 없는 문서를 얼마나 잘 선별할 수 있는지를 나타내는 선별도는 그 키워드가 어느 도메인에서 사용되었는지에 따라 달라진다. 예를 들어, "database"라는 키워드의 경우 데이타베이스 연구 분야의 자료에는 많이 나타날 것이지만 네트워크 연구 분야의 자료에는 상대적으로 적게 나타날 것이다. 이 경우 네트워크 연구 분야에서 나타난 "database"라는 키워드가 선별 효과가 더 뛰어나다고 봐야 한다. 하지만, 기존 TF-IDF 방법에서는 키워드가 사용된 도메인을 고려하지 않기 때문에 연구 분야에 상관 없이 "database"라는 키워드의 선별 효과를 동일하다고 본다. 반면에, 제안하는 KF-INF 방법에서는 키워드 k가 나타나는 MIU 노드 수를 구할 때 그 대상을 동일한 클래스에 속하는 MIU 노드만으로 한정함으로써 키워드 k의 노드 선별도를 더욱 정확하게 계산한다.

MIU 노드 점수는 추후 에지 점수를 계산할 때 사용될 뿐만 아니라 키워드 검색 결과의 우선 순위를 계산하는 데도 사용되기 때문에 표준화되어야 한다. 이를 위해 키워드 k에 대한 각 MIU 노드 점수를 MIU 노드 점수 중 가장 큰 값으로 나누어 0에서 1사이의 값으로 만든다.

#### 5.1.2 에지 점수 계산

클러스터링된 대상 그래프에서 모든 에지는 비리터럴 속성(NLP: Non-Literal Property)이다. 두 MIU 노드 i와 j를 연결하는 NLP 에지 e와 키워드 k가 주어졌다고 하자. 본 논문에서는 키워드 k와 NLP 에지 e의 연관도

를 구할 때 다음 요소들을 고려한다.

1. 에지 e의 텍스트에 키워드 k가 많이 나타날수록 에지 e와 키워드 k의 연관도가 높다. 즉, 키워드 k가 많이 나타나는 에지가 키워드 k를 잘 나타낸다고 본다.
2. MIU 노드 i나 j가 높은 점수를 가지고 있을수록 에지 e의 점수도 높다. 즉, 중요한 노드를 연결하는 에지가 더 중요하다고 본다.
3. MIU 노드 i나 j에 다른 에지가 많이 연결되어 있을수록 에지 e의 점수는 낮아진다. 즉, 다른 설명이 많이 붙어 있는 노드에 에지 e가 붙어 있을 경우 그 노드에 대한 에지 e의 설명이 가지는 중요도가 떨어진다고 본다.

이런 세가지 요소를 모두 반영하는 MIU 노드 i와 MIU 노드 j를 연결하는 에지 e = {i, j}의 키워드 k와의 연관도 점수 $Score_k(e)$는 다음과 같다.

$$Score_k(e) = KF_k(e) + c \times \sum_{m \in \{i,j\}} \frac{1 + \sum_{q \in Q} Score_q(m)}{|NB(m)|} \quad (4)$$

식 (4)에서 $KF_k(e)$는 에지 e의 텍스트에 키워드 k가 출현한 횟수를 의미한다. 식 (4)에서 나머지 부분은 에지 e가 MIU 노드 i와 MIU 노드 j로부터 받아오는 점수를 나타낸다. 노드 i로부터 받아오는 점수의 양은 다음과 같이 구한다. 먼저 기본 점수 1을 준 후 노드 i의 키워드별 점수를 모두 합하고($1+\Sigma_{q \in Q}$ $Score_q(i)$) 이를 노드 i에 연결되어 있는 이웃 노드 수(|NB(i)|)로 나눈다. 노드 j로부터 받아오는 점수의 양도 동일한 방식으로 계산한다. 마지막으로 노드 i와 노드 j로부터 받아오는 점수 양을 더한 후 여기에 에지 e가 노드 i와 j로부터 받아오는 점수의 양을 조절하기 위해 c를 곱한다. 본 논문에서는 c의 값으로 1을 사용하였다.

에지 점수도 키워드 검색 결과의 우선 순위를 계산하는 데 사용되기 때문에 표준화되어야 한다. 이를 위해 각 에지 점수를 에지 점수 중 가장 큰 값으로 나누어 0에서 1사이의 값으로 만든다.

### 5.2 서브그래프 식별

서브그래프 식별 과정에서는 대상 그래프 점수부여 과정을 거쳐 생성된 점수가 매겨진 대상 그래프에 대해 키워드 검색을 수행하여 최소 서브트리를 찾는다.

본 논문에서는 키워드 검색 결과를 찾는 데 기존의 그래프 기반 키워드 검색 알고리즘[6-8]을 사용한다. 주어진 그래프에서 키워드 $k_i$를 포함하는 노드 집합을 $S_i$라고 할 때 각 $S_i$로부터 하나 이상의 노드를 포함하는 최소 서브트리를 찾는 문제는 NP-Complete인 그룹 스타이너 트리 문제(group Steiner tree problem)로 알려져 있다[24,25]. 따라서, 기존 그래프 기반 키워드 검색 연구에서는 여러 가지 휴리스틱 알고리즘이 제안되었다. 기존 그래프 기반 키워드 검색 알고리즘은 일반적으로 다음과 같이 작동한다[8].

1. $E_i$를 키워드 $k_i$로부터 도달할 수 있는 노드 집합이라고 하자.
2. 초기에는 키워드 $k_i$를 포함하는 노드만 $E_i$에 들어 있다.
3. 키워드 검색의 각 단계에서 다음 작업을 수행한다. 먼저 이전에 방문한 노드 중 하나를 선택하고(v라고 하자) 그 노드에 연결되어 있는 에지 중 하나를 따라간다. 에지를 따라가면 방문하게 되는 노드를 u라고 하면 u를 방문한 것으로 표시하고 v를 포함하고 있는 $E_i$에 u를 추가한다.
4. 노드 x를 방문했다고 하자. 각 $E_i$에 대해 노드 x가 $E_i$에 들어 있거나 $E_i$에 들어 있는 어떤 노드와 x 사이에 에지가 존재하는 경우 알고리즘은 최소 서브트리의 루트인 x를 찾은 것이다.

### 5.3 서브그래프 순위 매기기

서브그래프 식별 과정을 통해 최소 서브트리 여럿을 얻으면 사용자가 입력한 키워드와 관련성이 높은 순서대로 사용자에게 반환한다. MIU 노드 집합 I과 에지 집합 E로 구성되는 최소 서브트리 t의 점수 Score(t)는 다음과 같이 계산한다.

$$Score(t) = \sum_{i \in I} Score(i) + c \times \sum_{e \in E} Score(e) \quad (5)$$

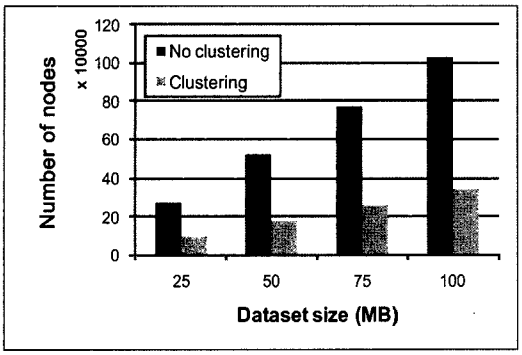
식 (5)에서 c는 노드 점수와 에지 점수의 중요도를 조절하기 위한 상수다. 본 논문에서는 c 값으로 1을 사용한다.

## 6. 평가

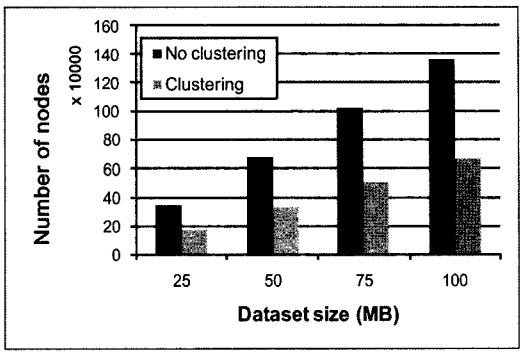
본 논문에서는 실제 RDF 데이타를 사용한 실험을 통해 제안하는 기법과 기존 그래프 기반 키워드 검색 기법 간의 성능을 비교하였다. 성능 비교 실험에서는 제안하는 기법과 [7]에서 제안된 역방향 키워드 검색(Backward keyword search, 줄여서 Backword KS) 기법의 키워드 검색 성능을 비교하였다. 제안하는 기법이 역방향 키워드 검색 기법과 다른 점은 원본 RDF 그래프에 대해 전처리 과정을 수행하여 클러스터링된 대상 그래프를 생성하고 이 그래프를 이용해 키워드 검색을 수행한다는 점과 키워드 검색을 수행하는 과정에서 5.1절에서 설명한 방식을 사용해 대상 그래프의 노드와 에지에 키워드와의 연관도 점수를 부여한다는 점이다.

표 2 실험 데이타의 통계자료

| 데이타 | 리터럴 노드 수 | 비리터럴 속성 수 | 리터럴 속성 수 | 노드 수 | 에지 수 |
|---|---|---|---|---|---|
| 25 MB | 174,053 | 174,053 | 169,241 | 270,246 | 343,294 |
| 50 MB | 345,383 | 335,970 | 345,383 | 524,541 | 681,353 |
| 75 MB | 512,885 | 504,851 | 512,885 | 773,222 | 1,017,736 |
| 100 MB | 691,798 | 669,802 | 691,798 | 1,027,905 | 1,361,600 |




(a) 노드 수




(b) 에지 수

그림 6 전처리 단계를 통한 그래프 크기 감소 효과

### 6.1 실험 환경

실험은 Dual Xeon CPU 2.8GHz, 메모리 3G인 PC에서 수행하였다. 실험 프로그램은 Java 6.0 언어로 작성하였고 실험 수행 시 Java 가상 머신의 최대 힙 크기(maximum heap size)를 1024MB로 설정하고 실험하였다. RDF API로는 Jena2.5.4[22]을 사용하였다.

실험 데이타는 DBLP Bibliography 웹 사이트[26]에서 제공하고 있는 논문 정보를 RDF 데이타로 변환한 SwetoDblp[27] 데이타의 2007년 8월 버전을 사용하였다. SwetoDblp 데이타의 전체 크기는 951MB이며 전체 데이타를 25MB 단위로 자른 소규모 데이타도 존재한다. 실험에서는 이 소규모 데이타를 조합하여 25MB, 50MB, 75MB, 100MB 크기의 데이타를 만들어 사용하였다. 표 2는 실험 데이타와 관련된 통계자료를 보여준다. 100MB 데이타의 경우 약 100만개의 노드와 130만개의 에지로 구성되는 RDF 그래프를 담고 있다.

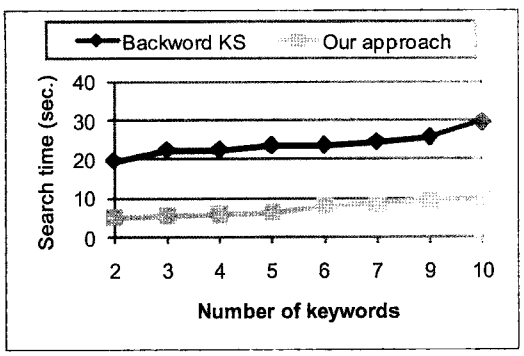
### 6.2 데이타 크기 실험

데이타 크기 실험에서는 제안하는 기법의 전처리 단계를 통해 서로 관련이 있는 노드와 에지를 클러스터링함으로써 얻을 수 있는 RDF 그래프 크기 감소 효과를 측정해 보았다. 실험에 사용한 RDF 그래프의 크기는 25MB, 50MB, 75MB, 100MB다. 그림 6(a)는 제안하는 기법의 전처리 단계를 통해 RDF 그래프를 클러스터링하기 전(No clustering)과 클러스터링한 후(Clustering)의 RDF 그래프의 노드 수를 측정한 결과를 보여준다. 실험 결과는 클러스터링된 대상 그래프가 원본 그래프에 비해 3분의 1 가량 줄어진 노드 개수를 가진다는 것을 보여준다. 그림 그림 6(b)는 에지 수 측정 결과를 보여준다. 에지 수는 클러스터링 후 반 정도로 줄어들었다.

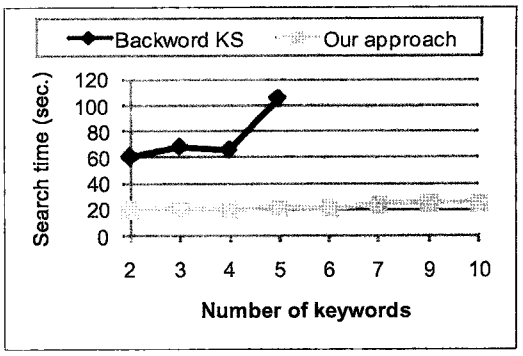
### 6.3 키워드 개수 실험

키워드 개수 실험에서는 키워드 수가 늘어남에 따라 제안하는 기법(Our approach)과 기존 기법(Backword KS)에서 키워드 검색 수행 속도가 어떻게 변화하는지를 측정했다. 이 실험에서 사용한 데이타 셋은 25MB 데이타 셋과 50MB 데이타 셋이다. 각 실험에서 데이타 셋과 키워드 수가 주어지면, 주어진 데이타 셋에서 주어진 개수만큼의 키워드를 무작위로 선택한 후 실험하였다. 키워드 중에서 영어 알파벳 이외의 문자를 포함하거나, 길이가 2 이하인 키워드는 실험에서 제외하였다. 이렇게 영어 단어가 아니거나 길이가 짧은 키워드를 제외하면, 25MB 데이타 셋에서는 약 1만 2천 개의 키워드가, 50MB 데이타 셋에서는 약 1만 7천 개의 키워드가 추출된다. 모든 실험은 100 번 반복해서 실험하였고 결과 값으로 100 번 실험한 결과의 평균 값을 취했다.

그림 7(a)는 25MB 데이타 셋에서, 그림 7(b)는 50MB 데이타 셋에서 키워드 개수가 변함에 따라 두 기법의 검색 시간이 어떻게 변하는지를 측정한 결과를 보여준다. 실험 결과를 보면 키워드 개수가 증가함에 따라 두 기법 모두에서 검색 성능이 서서히 감소한다. 두 기법 간 검색 성능 차이는 25MB 데이타 셋에서는 최대 4배, 50MB 데이타 셋에서는 최대 5배다. 특히, 50MB 데이타 셋에서는 키워드 개수가 6개가 넘어갈 경우 기




(a) 25MB 데이타 셋




(b) 50MB 데이타 셋

그림 7 키워드 개수에 따른 검색 시간

존 기법에서는 메모리 부족으로 인해 키워드 검색을 수행할 수 없었다.

## 7. 결 론

본 논문에서는 RDF 온톨로지 데이타에 대한 효율적인 키워드 검색 기법을 제안하였다. 제안하는 기법은 크게 전처리 단계와 키워드 검색 단계로 나뉜다. 전처리 단계에서는 사용자가 입력한 키워드와 직접적으로 관련된 정보뿐만 아니라 주변 정보도 함께 제공하고 또 RDF 그래프의 크기를 줄여 키워드 검색 성능을 높이기 위해 원본 RDF 그래프에서 관련 있는 노드들을 묶어 새로운 그래프를 생성한다. 키워드 검색 단계에서는 사용자에게 보다 정확한 결과를 전달하기 위해 키워드가 사용되는 도메인을 고려해 전처리 단계를 거친 그래프의 각 노드와 에지에 키워드와의 연관도 점수를 부여하고 기존 그래프 기반 키워드 검색 알고리즘을 사용해 키워드 검색 결과를 구한 후 사용자에게 반환한다.

본 논문에서는 제안하는 기법과 기존 그래프 기반 키워드 검색 기법과의 성능 비교 실험을 통해 제안하는 기법의 우수성을 보였다. 제안하는 기법은 전처리 단계를 통해 키워드 검색 대상이 되는 원본 그래프의 크기를 최대 반 정도 줄였으며, 기존 기법과 키워드 검색 성능에 있어서는 최대 5배의 성능 향상이 있었다.

## 참 고 문 헌

[1] Agrawal, S., et al., "DBXplorer: A System for Keyword-Based Search over Relational Databases," In Proc. of International Conference on Data Engineering, pp. 5-16, 2002.

[2] Hristidis, V. and Papakonstantinou, Y., "DISCOVERY: Keyword Search in Relational Databases," In Proc. of International Conference on Very Large Data Bases, pp. 670-681, 2002.

[3] Hristidis, V., et al., "Efficient IR-Style Keyword Search over Relational Databases," In Proc. of International Conference on Very Large Data Bases, pp. 850-861, 2003.

[4] Liu, F., et al., "Effective Keyword Search in Relational Databases," In Proc. of ACM SIGMOD Conference, pp. 563-574, 2006.

[5] Luo, Y., et al., "Spark: top-k keyword query in relational databases," In Proc. of ACM SIGMOD Conference, pp. 115-126, 2007.

[6] Bhalotia, G., et al., "Keyword Searching and Browsing in Databases using BANKS," In Proc. of International Conference on Data Engineering, pp. 431-440, 2002.

[7] Kacholia, T., et al., "Bidirectional Expansion For Keyword Search on Graph Databases," In Proc. of International Conference on Very Large Data Bases, pp. 505-516, 2005.

[8] He, H., et al., "BLINKS: ranked keyword searches on graphs," In Proc. of ACM SIGMOD Conference, pp. 305-316, 2007.

[9] Guo, L., et al., "XRANK: Ranked Keyword Search over XML Documents," In Proc. of ACM SIGMOD Conference, pp. 16-27, 2003.

[10] Hristidis, V., et al., "Keyword Proximity Search on XML Graphs," In Proc. of International Conference on Data Engineering, pp. 367-378, 2003.

[11] Xu, Y. and Papakonstantinou, Y., "Efficient Keyword Search for Smallest LCAs in XML Databases," In Proc. of ACM SIGMOD Conference, pp. 537-538, 2005.

[12] Liu, Z. and Chen, Y., "Identifying meaningful return information for XML keyword search," In Proc. of ACM SIGMOD Conference, pp. 329-340, 2007.

[13] Liu, Z., et al., "XSeek: A Semantic XML Search Engine Using Keywords," In Proc. of International Conference on Very Large Data Bases, pp. 1330-1333, 2007.

[14] http://www.w3.org/TR/REC-rdf-syntax

[15] http://www.w3.org/TR/rdf-schema
[16] http://www.w3.org/TR/rdf-sparql-query
[17] Rocha, C., et al., "A Hybrid Approach for Searching in the Semantic Web," In Proc. of International World Wide Web Conference, pp. 374-383, 2004.
[18] Zhang, L., et al., "Semplore: An IR Approach to Scalable Hybrid Query of Semantic Web Data," In Proc. of International Semantic Web Conference, pp. 652-665, 2007.
[19] Anyanwu, K., et al., "SPARQ2L: towards support for subgraph extraction queries in rdf databases," In Proc. of International World Wide Web Conference, pp. 797-806, 2007.
[20] http://www.w3.org/TR/rdf-concepts
[21] http://www.w3c.org/TR/REC-xml-names
[22] http://jena.sourceforge.net
[23] Yates, B. and Neto, B., "Modern Information Retrieval," ACM Press, New York, 1999.
[24] Ding, B., et al., "Finding Top-k Min-Cost Connected Trees in Databases," In Proc. of International Conference on Data Engineering, pp. 836-845, 2007.
[25] Kimelfeld, B. and Sagiv, Y., "Finding and approximating top-k answers in keyword proximity search," In Proc. of PODS Conference, pp. 173-182, 2006.
[26] http://www.informatik.uni-trier.de/~ley/db
[27] http://lsdis.cs.uga.edu/projects/semdis/swetodblp

김 진 하
2006년 성균관대학교 정보통신공학부 학사. 2006년~2008년 KAIST 전산학과 석사. 2008년~현재 NHN 서비스관리시스템랩 품질관리시스템개발팀. 관심분야는 시맨틱 웹, 키워드 검색 등

송 인 철
2004년 아주대학교 정보 및 컴퓨터공학부 학사. 2004년~2006년 KAIST 전산학과 석사. 2006년~현재 KAIST 전산학과 박사과정. 관심분야는 센서 네트워크, 시맨틱 웹, 키워드 검색 등

김 명 호
정보과학회논문지 : 데이타베이스
제 35 권 제 4 호 참조