제25회 한국정보처리학회 춘계학술발표대회 논문집 제13권 제1호 (2006. 5)

효율적인 시멘틱 질의 처리를 위한 인덱싱 기법

김학수*, 차현석*, 손진현*
*한양대학교 컴퓨터학과
e-mail : {hagsoo, hscha, jhson}@cse.hanyang.ac.kr

Indexing Mechanism for Efficient Semantic Query Processing

Hak Soo Kim*, Hyun Seok Cha*, Jin Hyun Son*
*Dept. of Computer Science and Engineering, Hanyang University

요 약

RDF 는 트리플의 집합으로서 그래프 데이터 모델로 표현되며, 사용자는 RDF 그래프 모델로부터 정보를 검색하기 위해 시멘틱 질의 언어를 사용한다. 그러나, 이러한 접근 방식은 최악의 경우 전체 그래프 데이터 모델을 검색해야 되는 문제점이 발생한다. 이에 따라 최근의 연구에서는 시멘틱 질의를 효율적으로 처리하기 위해서 인덱스를 사용한다.

시멘틱 질의 언어(RDQL, SPARQL)의 핵심은 RDF 트리플에 대한 패턴을 기술함으로써 원하는 트리플 정보를 검색할 수 있게 하는 것이다. 따라서, 기존의 인덱스는 단일 트리플을 효율적으로 검색하는 데 초점을 둔다. 그러나, 트리플 패턴의 집합으로 질의가 표현될 경우에는 트리플 패턴 사이의 상관관계 때문에 조인비용이 많이 발생하는 문제점이 있다. 본 논문에서는 조인 비용이 발생되는 문제점을 해결하기 위한 인덱싱 기법을 제안한다. RDF 그래프 모델에서 유지해야 할 정보를 줄이기 위해서 RDF 그래프 모델에 존재하는 유사한 서브 그래프를 하나의 서브 그래프로 병합한다. 병합절차를 마친 여러 서브 그래프에 존재하는 모든 경로를 인덱스에 유지 함으로써 조인 비용을 제거한다.

1. 서론

시멘틱 웹 환경을 위한 대표적인 시멘틱 정보 관리 시스템으로는 JENA[3], Sesame[4], RDFStore[5] 등이 있다. 기존의 시스템에서는 시멘틱 정보를 검색하기 위해서 시멘틱 질의 언어를 제공하고 있으며, 질의를 효율적으로 처리하기 위해서 내부적으로 최대 6 개까지의 인덱스를 사용하고 있다. 기존의 인덱스들은 RDF 문서에서 하나의 트리플을 효율적으로 검색할 수 있도록 설계 되었다. 따라서 이러한 인덱스를 통해서 단일 트리플을 검색할 수 있기 때문에 하나의 트리플 패턴이 기술되는 단순한 질의의 경우 높은 성능을 얻을 수 있다. 하지만 하나이상의 상관관계가 있는 트리플 패턴이 기술되는 복잡한 질의의 경우 인덱스로부터 트리플을 검색 후 검색된 결과에 대한 조인 연산이 필요하다. 조인 연산은 인덱스로부터 검색된 결과가 많을수록 더 많은 비용을 가지게 되고 질의 처리를 지연시킨다.

본 논문에서는 RDF 그래프를 구성하는 각각의 트리플에 대한 정보에 대해서만 인덱스를 만드는 것이 아니라, 하나의 RDF 문서에 존재하는 모든 경로에 대한 인덱스를 구성해서 조인 비용을 없애고, 또한 기존의 색인 기법보다 적은 메모리 사용량을 가지는 인덱싱 기법을 제안한다.

본 논문의 구성은 다음과 같다. 2 장에서는 문제정의와 RDF 그래프가 가지는 특성에 따른 연구 및 동기, 3 장과 4 장에서는 본 논문에서 제안한 인덱싱 기법, 5 장에서는 제안한 방법의 효율성 분석, 6 장에서는 결론 및 향후 연구 과제를 제시하였다.

2. 동기부여 및 기여

RDF 그래프를 구성하는 트리플의 각 요소(주어, 술어, 목적어)를 키값으로 가지는 인덱스를 구축하게 되면, 단일 트리플의 요소를 찾는 질의가 요청 되었을 경우 한 번의 인덱스 접근을 통해서 해당되는 트리플을 찾고 그 트리플에서 찾고자 하는 요소를 찾을 수 있다. 이러한 인덱스들은 단일 트리플을 빠르게 찾기 위해서 설계되었기 때문에 단일 트리

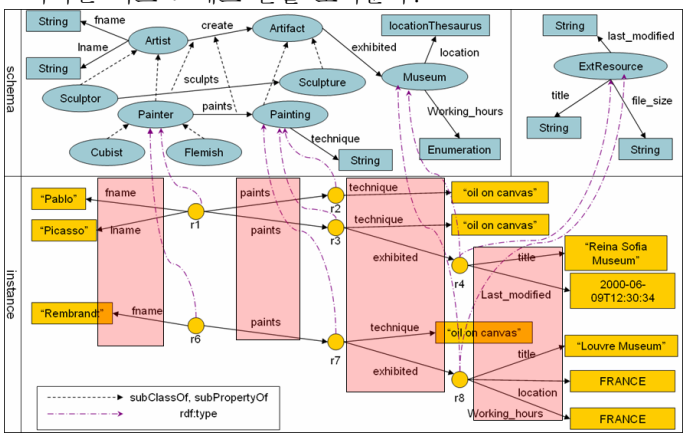
플의 요소를 찾는 형식의 질의에 대해서는 효율적인 응답시간을 가질 수 있다.

그래프 내에 존재하는 그래프 패턴이나 하나이상의 트리플로 구성되는 경로에 대한 질의가 요청되면 인덱스를 통해서 해당 트리플을 검색해내고, 검색한 트리플에 대한 조인 연산을 해야한다. 조인연산은 검색된 트리플이 많을 수록 비용이 커지게 된다. 질의 성능을 위해서 비용이 많이 발생하는 조인 연산을 배제하기 위해서는 RDF 그래프에 존재하는 전체 경로에 대해서 인덱스를 구축해야 하는데 이는 너무 많은 비용을 소모하게 된다. 하지만 RDF 그래프가 가지는 특징을 잘 이용하면 모든 경로 정보를 저장하지 않고도 효율적인 시멘틱 질의 처리가 가능한 인덱스를 구축할 수 있다.

RDF 그래프의 특징을 살펴보면, RDF 그래프는 스키마와 인스턴스 계층으로 구별될 수 있다. 스키마 계층에서는 특정 도메인에 대한 웹 자원의 메타 데이터 기술을 위한 클래스와 속성의 계층구조를 정의한다. 이러한 스키마 정보는 관계데이터베이스의 스키마와 같은 기능을 하기 때문에 유용하게 사용될 수 있지만 하나의 RDF 문서에서 스키마 정보는 집합으로 존재하고 이러한 집합 내의 서브 그래프 사이에 존재하는 관계성은 스키마 수준에서는 나타나지 않기 때문에 충분한 정보를 가지지 않는다. 하지만 인스턴스 수준에서는 스키마를 바탕으로 기술하기 때문에 중복된 경로가 많이 발생하게 된다. 이것은 스키마 수준에서 정의한 속성을 그대로 사용해서 실제 인스턴스를 기술하기 때문이다.

따라서, 정보의 손실 없이 이러한 중복을 제거한다면 최소한의 경로 정보만 유지할 수 있고, 이러한 정보를 바탕으로 인덱스를 구축한다면 적은 비용으로 단일 트리플의 요소를 찾는 질의뿐만 아니라 그래프 패턴이나 길이가 긴 경로에 대한 질의에도 조인 연산 없이 효율적으로 시멘틱 질의를 처리 할 수 있게 된다.

아래 그림은 실제 인스턴스 수준에서 중복된 경로가 나타나는 서브 그래프 들을 보여준다.



(그림 1) 유사한 유형의 RDF 서브 그래프

본 논문에서는 [그림 3]과 같이 비슷한 유형의 서브 그래프가 많이 나타나는 단일 문서에 대한 인덱싱 기법을 제안한다.

3. 서브 그래프 병합

서브 그래프를 병합하는 절차는 인덱스를 구축하기 전에 중복으로 인한 불필요한 경로를 제거하고 최소한의 경로만을 유지하기 위한 작업으로 다음과 같은 2 가지 절차로 수행된다.

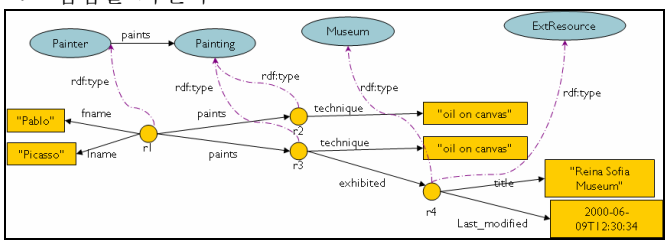
1. 서브 그래프 내부에서의 중복 제거

2. 서브 그래프 병합

위의 절차를 설명하기 위해서 다음과 같은 정의를 필요로 한다.

> 정의 1. 타입시퀀스 유사도
>
> 하나의 RDF 그래프에 존재하는 두 개의 서브그래프 U_1과 U_2가 있을때, 만약 U_1과 U_2가 같은 타입시퀀스 집합을 가지면 두 서브 그래프는 타입시퀀스가 유사하고 이 두 그래프는 병합될 수 있다.

여기에서 타입은 인스턴스 수준에서 기술된 웹 자원의 rdf:type 의 값을 말하는 것이고, 그 값은 RDF 스키마에서 기술한 클래스를 나타낸다. 만약 두 웹 자원의 rdf:type 값이 같다는 것은 같은 클래스의 인스턴스라는 것을 의미하고 따라서 웹 자원은 같은 속성을 가질 수 있게 된다. 따라서 두 서브 그래프의 타입 값이 같은 시퀀스로 나타난다면 두 서브 그래프는 결국 같은 속성을 가질 수 있기 때문에 병합하게 된다. 이것은 [그림 2]의 경우 {Painter.Painting.Museum|ExtResource}와 같은 타입 시퀀스 집합을 가진다



(그림 2) RDF 타입시퀀스 집합

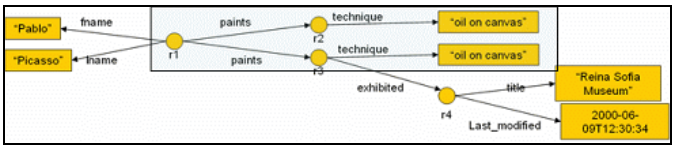
다음의 연산자들은 하나의 서브 그래프 내부에서의 중복제거, 두 서브 그래프의 병합에 사용된다.

> 정의 2. 연산자 +, -, ∪
>
> 두 개의 서브그래프 SG_1과 SG_2가 있을 때,
> - '+' 연산자는 SG_1에 SG_2의 간선과 노드를 추가하는 연산자이다. 단, SG_2의 노드의 라벨은 null로 설정되어 추가된다.
> - '-' 연산자는 SG_1과 SG_2를 비교해서 rdf:type 값이 같은 노드와 같은 간선을 제거하는 연산자이다.
> - '∪' 연산자는 SG_1에 SG_2를 병합하는 연산자이다. 단, 병합 후 SG_1의 노드의 라벨은 SG_1과 SG_2의 라벨을 순차적으로 가지는 집합 형태가 된다.

서브 그래프 병합은 서브 그래프 내부에서의 중복제거와 서브 그래프 병합의 두 단계로 구성된다.

1. 서브그래프 내부에서의 중복제거

서브 그래프 내부에서의 중복은 한 노드에서 같은 간선이 한 번 이상 나타날 수 있기 때문에 발생한다. 이것은 RDF 스키마에서 속성을 정의할 때 하나의 속성이 나타날 수 있는 횟 수에 대한 제한이 없기 때문이다. [그림 3]에서 웹 자원 r1 이 paints 라는 속성을 두 번 가지는 것을 알 수 있다. 현실에서도 한 명의 화가는 여러 작품을 그릴 수 있기 때문에 아래와 같은 형태의 그래프는 빈번하게 나타날 수 있다.

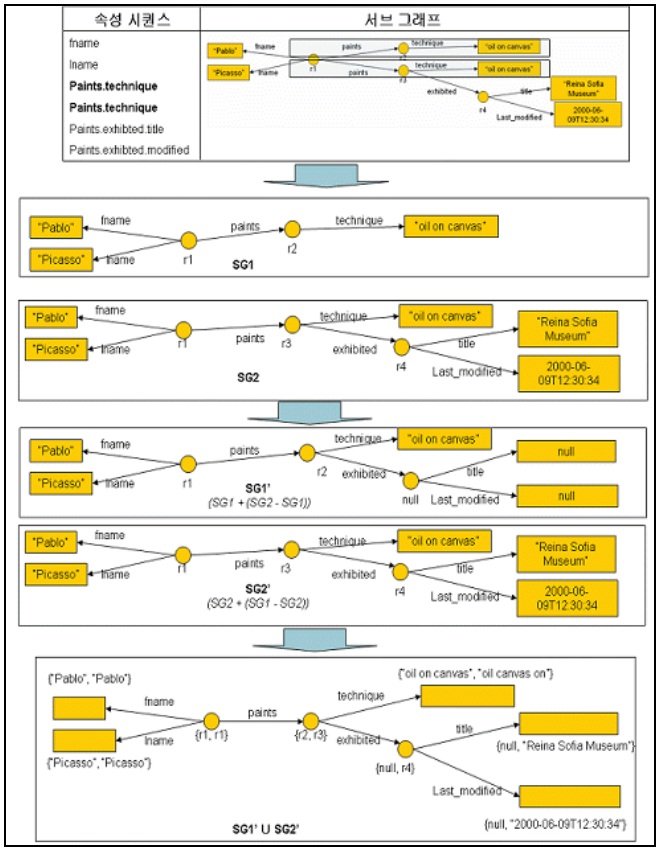


(그림 3) 그래프 내부에서의 중복

하나의 그래프 내부에서의 중복제거는 다음과 같은 순서로 이루어진다.

(1) 서브 그래프에서 루트 노드부터 단말 노드까지의 모든 경로 추출
(2) 추출한 경로에서 나타나는 속성의 시퀀스 추출
(3) 각 경로에서 추출된 속성의 시퀀스를 비교
(4) 중복제거

[그림 4]는 서브그래프 내부에서의 중복제거 절차를 보여준다.



(그림 4) 서브 그래프 내부에서의 중복 제거

2. 서브그래프 병합

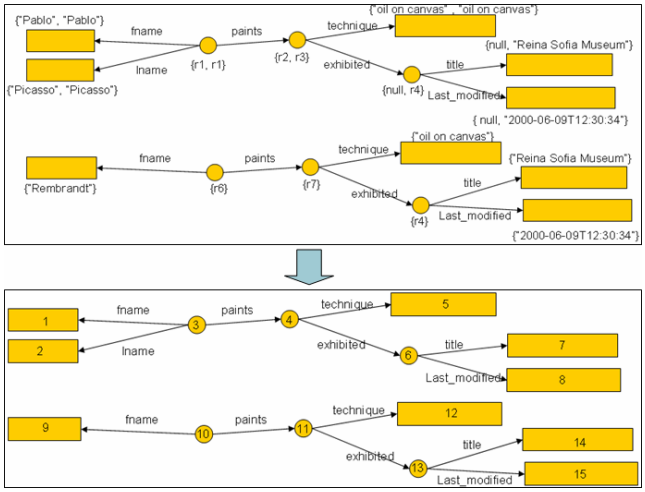
서브 그래프 병합 절차는 서브 그래프 내에서의 중복을 제거한 후 유사한 서브그래프가 존재할 경우 하나의 서브 그래프에 대해서만 경로를 유지하기 위한 절차로서 다음과 같은 순서로 실행된다.

(1) 각 서브 그래프로부터 타입 시퀀스 추출
(2) 추출한 타입 시퀀스 비교
(3) 그래프 병합

4. 인덱스 구성

서브 그래프 병합 후의 그래프를 기반으로 인덱스를 구성한다. 병합 절차를 통해서 수많은 비슷한 유형의 서브 그래프들이 하나로 병합된 후에는 최소한의 경로 정보만 남게 된다. 이 경로 정보를 이용해서 인덱스를 구성하게 되면 RDF 그래프 내에 존재하는 모든 경로를 유지하면서도 적은 메모리를 사용한다.

인덱스 구성은 먼저 병합된 그래프를 아래의 [그림 5]와 같이 각 서브 그래프 별로 번호가 겹치지 않게 번호를 부여한다.



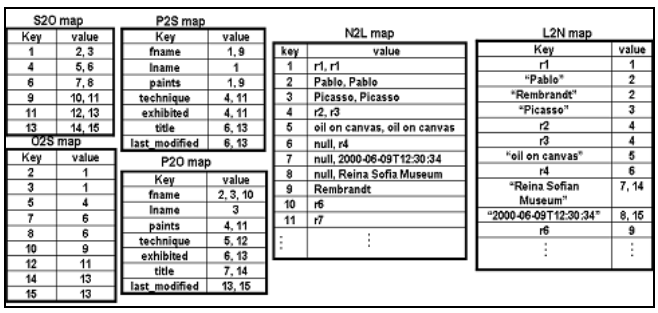
(그림 5) 병합된 서브 그래프에 번호 부여

서브 그래프의 각 노드에 번호를 부여한 후, 번호와 그 번호에 해당되는 레이블과 관계성을 맺어 주기 위해서 레이블 맵을 구성한다. 이러한 레이블 맵을 구성하는 구체적인 목적은 다음과 같다.

첫 째, 질의가 요청되었을 때, 질의에 기술된 웹 자원이 병합된 그래프의 어느 노드에 해당되는지를 알아내고, 결과로서 병합된 그래프의 노드로부터 실제 웹 자원을 추출하는데 사용된다.

둘 째, 질의가 요청되었을 때, WHERE 절에 기술된 경로의 시작 노드와 끝 노드를 찾아 한 번의 인덱스 접근으로 바로 두 노드 사이에 존재하는 경로를 검색할 수 있도록 하기 위함이다.

위와 같은 두 가지 목적을 위하여 S2O(Subject to Object), O2S(Object to Subject), P2S(Predicate to Subject), P2O(predicate to Object), N2L(Node to Label), L2N(Label to Node) 총 6 개의 맵이 사용된다. 이러한 맵은 해쉬 테이블로 구성된다. 6 개의 맵 중에서 L2N 을 제외한 나머지 맵은 모두 다 병합된 그래프의 수에 따라 맵의 크기가 결정된다. 하지만 L2N 은 기존의 그래프에 존재하는 모든 노드의 수만큼의 크기를 가진다. 아래의 [그림 6]은 6 개의 맵을 보여준다.



(그림 6) 6 개의 레이블 맵

인접행렬은 그래프에 두 노드 사이에 있는 경로를 바로 찾아 낼 수 있는 그래프 표현법이다. 무 방향 그래프의 경우 행렬의 상위 또는 하위 삼각형만 저장하여 기억 장소를 절약할 수 있지만 RDF 그래프는 방향 그래프이기 때문에 인접행렬을 사용하게 되면 메모리가 많이 낭비되는 단점이 있다.

하지만 본 논문에서는 RDF 그래프에 존재하는 모든 경로 정보를 유지하지 않고 먼저 서브 그래프 병합을 통해서 유지되어야 할 최소한 경로정보를 추출해서 그 정보만 유지하기 때문에 인접행렬의 메모리 낭비가 심하다는 점을 보완하여 인접행렬의 장점을 극대화해서 사용할 수 있다.

아래의 [표 1]과 [표 2]는 [그림 5]에 있는 그래프를 인접행렬로 나타낸 것이다.

<표 1> 병합된 첫 번째 서브 그래프

	1	2	3	4	5	6	7	8
1		fname	lname	paints	paints,4, technique	paints,4, exhibited	paints, 4 exhibited, 6, title	paints, 4, exhibited, 6, last_modified
2								
3								
4					technique	exhibited	exhibited, 6, title	exhibited, 6, last_modified
5								
6							title	last_modified
7								
8								

<표 2> 병합된 두 번째 서브 그래프

	9	10	11	12	13	14	15
9		fname	paints	paints, 11 technique	paints,11 exhibited	paints, 11 technique, 13, title	paints,11 exhibited, 13 last_modified
10							
11				techinique	exhibited	exhibited, 13, title	exhibited, 13, last_modified
12							
13						title	last_modified
14							
15							

본 논문에서 제안한 인덱스는 병합된 서브 그래프에 존재하는 모든 경로에 대해서 인덱스를 구성하기 때문에 조인 비용없이 임의의 두 노드 사이에 존재하는 경로를 한 번의 인덱스 접근으로 바로 가져올 수 있다. 항상 시작 노드와 끝 노드를 알아야 하기 때문에 질의에 따라서 6 개의 맵 중에서 1 개나 2 개의 맵을 이용해서 시작 노드와 끝 노드를 검색할 수 있다. 검색된 노드를 이용해서 인접행렬로부터 두 노드 사이의 경로를 가져온다. 그 경로에서 사용자가 찾고자 하는 노드나 속성을 찾는다. 속성일 경우 그 속성 자체가 결과가 되지만 노드일 경우에는 노드에 속하는 웹 자원을 검색하기 위해서 맵을 한 번 더 접근하게 된다.

5. 결론 및 향후 과제

논문에서는 단일 RDF 문서에 대한 인덱싱 기법을 제안하였다. 서브 그래프 병합을 통해서 RDF 문서 내에 존재하고 있는 모든 경로를 유지하면서도 상대적으로 적은 메모리를 사용한다. 또한 전체경로를 유지하기 때문에 조인 비용이 발생하지 않게 하였다. 이것은 기존의 인덱싱 기법이 질의에서 기술된 경로가 길어질수록 그에 따른 조인이 발생하는 문제점을 해결 하고자 한 것으로서 문서의 크기가 클수록 (즉 기존의 인덱스에서 유지해야할 트리플이 많을수록), 그래프 병합절차를 거친 후의 서브 그래프의 수가 작을수록 기존의 인덱싱 기법에 비해 많은 장점을 가질 수 있다.

향후 과제로는 조인이 없는 질의의 경우 데이터가 충분히 많지 않으면 질의처리 속도가 기존의 인덱싱 기법보다 오히려 더 느려지게 되는 문제점을 개선하기 위한 연구가 필요하다. 또한 서브 그래프 병합 시 모든 서브 그래프들을 비교해야 하기 때문에 인덱스를 구축하는데 걸리는 시간을 좀 더 개선시키는 연구가 필요하다.

참고문헌

[1] G. Klyne, J. Carroll: "Resource Description Framework (RDF): Concepts and Abstract Syntax" W3C Recommendation. Feb 2004.

[2] D. Brickley, R.V. Guha: RDF Vocabulary Description Language 1.0 : RDF Schema W3C Recommendation. Feb 2004.

[3] K. Wilkinson, C. Sayers, H. Kuno, Dave Reynolds: "Efficient RDF Storage and Retrieval in Jena2" . SWDB 2003, pp. 131-150, 2003. see http://jena.sourceforge.net/

[4] J. Broekstra, A. Kampman, F. van Harmelen: "Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema". In The Semantic Web - ISWC 2002, volume 2342 of Lecture Notes in Computer Science, pp. 54-68, 2002. See http://openrdf.org/

[5] A. Reggiori, D. van Gulik, Z. Bjelogrlic: "Indexing and retrieving SemanticWeb resources: the RDFStore model". EuropeWorkshop on SemanticWeb Storage and Retrieval - SWAD 2003, 2003.

[6] E. Pud'hommeaux, A. Seaborne, HP Labs Bristol, "SPARQL Query Language for RDF : W3C Working Draft 19 April 2005", W3C, 2005, see http://www.w3.org/TR/rdf-sparql-query/