Journal of KIIT. Vol. 15, No. 11, pp. 31-38, Nov. 30, 2017. pISSN 1598-8619, eISSN 2093-7571 31
http://dx.doi.org/10.14801/jkiit.2017.15.11.31

3차원 R*-tree를 활용한 링크드 데이터 인덱싱 기법

이 용 주*

Linked Data Indexing Techniques Using 3D R*-tree

Yong-Ju Lee*

이 논문은 2016년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임
(No. 2016R1D1B02008553).

요 약

시맨틱 웹을 구현하기 위한 가장 실용적인 접근법으로 링크드 데이터가 그동안 많은 관심을 받아왔지만, 링크드 데이터 영역은 아직까지 많은 이슈들이 존재하고 있다. 링크드 데이터는 RDF 그래프 구조로 모델링되기 때문에 기존의 관계 데이터베이스 관리시스템이나 웹 기술들을 직접 적용할 수가 없다. 본 논문에서는 중앙집중 방식과 분산 방식 사이의 하나의 하이브리드 방법을 제안한다. 이 방법은 MBB(Minimum Bounding Box) 근사 기법을 기반으로 한 이단계 색인 구조로써 고도로 분산되어 있는 링크드 데이터를 효율적으로 검색할 수 있고 저장 용량도 상당히 줄일 수 있다. 우리는 실제 링크드 오픈 데이터셋을 사용하여 이단계 인덱스 구조를 기존의 인덱스 구조들과 성능 분석을 수행하였다. 실험 결과 제안된 방식은 많은 불필요한 자원들을 신속하게 제거할 수 있으므로 기존의 방식보다 성능이 더 우수함을 보여준다.

Abstract

Linked Data have gained much attention as a very pragmatic approach to implement the Semantic Web. However, there are still a lot of issues in the area of Linked Data. Since Linked Data are modeled as RDF graph structures, we cannot directly adopt solutions from traditional RDBMS (Relational Database Management System) or Web technologies. This paper proposes a hybrid method between the centralized approach and the distributed approach. By using the two-step index structure based on the MBB (Minimum Bounding Box) approximation, our approach can efficiently retrieve the distributed Linked Data and significantly reduce the amount of storage space. We evaluate the two-step index structure with existing index structures using a real Linked Open Dataset. The experimental results show that the proposed approach outperforms the existing approaches due to its ability to rapidly reduce a large amount of irrelevant resources.

Keywords
linked data, index structure, 3D R*-tree, RDF, SPARQL, join query

* 경북대학교 컴퓨터학부
- ORCID: http://orcid.org/0000-0002-1705-4967

· Received: Aug. 16, 2017, Revised: Oct. 11, 2017, Accepted: Oct. 14, 2017
· Corresponding Author: Yong-Ju Lee
School of Computer Science and Engineering, Kyungpook National University, 80, Daehak-ro, Buk-gu, Daegu, 41566, Korea,
Tel.: +82-53-950-7285, Email: yongju@knu.ac.kr

Ⅰ. 서 론

위키피디아(Wikipedia)에서 링크드 데이터(Linked Data)는 “구조화된 데이터를 웹상에 게시하는 하나의 방법으로써, 시맨틱(Semantic) 쿼리를 통해 데이터가 상호 연결되고 더욱 유용해질 수 있다”라고 정의하고 있다[1]. 기술적으로 링크드 데이터의 핵심 아이디어는 HTTP(Hypertext Transfer Protocol)의 URI(Uniform Resource Identification) 사용이다. URI는 웹 문서들을 식별하는 것뿐만 아니라 임의의 실세계 객체들을 탐색할 수 있다. 링크드 데이터를 구축하기 위해서는 비구조적인 데이터를 구조화하는데 시맨틱 형태로 표현해야 한다. 여기서 자원을 구조화한다는 것은 데이터를 RDF(Resource Description Framework) 형태로 표현하는 것이다.

RDF는 웹 자원의 메타(Meta) 정보를 표현하기 위한 데이터 모델로서, RDF 내에 있는 모든 데이터 아이템들은 주어-술어-목적어 형태의 트리플 구조를 갖는다. 여기서 주어가 정보의 자원이라면, 술어는 자원의 속성, 목적어는 해당 자원이 가리키는 값(또는 대상)이 된다. 그리고 정보의 자원은 URI를 통해 고유 식별자를 가지고 여러 가지 속성들을 이용하기 위해 자원들 간의 링크를 생성한다.

링크드 데이터를 위한 저장 및 색인 기법에 대한 연구는 최근 데이터베이스 분야에서 새롭고 도전적인 과제로 인식되어 지고 있다. 그러나 RDF 트리플은 그래프로 모델링되기 때문에 기존의 관계 데이터베이스관리시스템(RDBMS)이나 XML 기법들이 직접 적용될 수 없으므로 이 분야는 현재 많은 오픈 문제들을 포함하고 있다. 따라서 우선적으로 분산되고 변동성이 심한 초대용량 링크드 데이터에 대한 압축된 형태의 새로운 데이터 저장 및 인덱싱 기법이 필요하고, 이 기법들을 활용하여 데이터 연관성이 중요한 링크드 데이터 질의 처리 최적화 기술들이 요구된다.

현재 RDF 데이터의 저장이나 색인에 관한 연구들은 이미 많이 발표되어 있는 상황이나 링크드 데이터를 처리하는 방법들은 그렇게 많지 않다. 보통 링크드 데이터 색인 방법들은 쿼리 실행 프로세스 동안 데이터 링크의 존재는 무시한다. 대신 이러한 방법들은 쿼리를 실행하는 동안 조회할 URI들을 식별하기 위하여 미리 수집된 인덱스에 의존한다. 따라서 데이터 자체를 저장하는 기존의 인덱스 구조들(예, Quad, RDF-3X, Hexastore, 등)과는 달리 여기에서 논의되는 인덱스 방법은 URI들을 데이터에 대한 포인터로서 인덱스하는 데이터 구조를 사용한다.

이러한 인덱스를 이용한 리소스의 선택은 관련성에 기반을 둔다. 즉, URI를 조회하여 검색된 데이터가 쿼리 결과에 기여한다면 그 URI는 해당 쿼리에 관련된다고 말할 수 있다. 관련 없는 URI들로부터 주어지는 데이터는 쿼리 결과를 계산하는데 필요 없으므로 이러한 URI들의 조회를 피하면 쿼리 실행 비용이 상당히 줄어들 수 있다. 이러한 맥락에서 본 연구의 초점은 관련된 모든 URI들은 포함하지만 관련 없는 URI들은 가능한 적게 포함시키는 인덱스된 모든 URI들의 부분 집합을 식별하는 것이다.

이러한 인덱싱 방법은 전통적인 데이터베이스 쿼리 프로세싱 기법으로부터 많은 영향을 받았다. 예를 들면, Umbrich 등[2]은 링크드 데이터 요약을 위해 다차원 히스토그램(Multidimensional Histogram) 개념을 적용시켰고, 우리는 이러한 방법의 성능을 개선한 확장된 다차원 히스토그램(MDH*: Extended Multidimensional Histogram)[3]을 제안한 바 있다.

그러나 MDH* 기법의 가장 큰 문제점은 데이터의 집합이 균일하게 분포되어 있다고 가정하는 데 있다. 실제로 링크드 데이터는 밀도가 희박하거나 많은 클러스터(Cluster)를 포함할 수 있기 때문에 이러한 불규칙 분포를 위한 새로운 인덱스 구조가 요구된다. 본 논문에서는 MDH*의 단점을 극복하기 위해 기존 문헌으로부터 잘 알려져 있는 R*-Tree 다차원 공간 인덱스 구조를 활용하여 가장 최선의 링크드 데이터 색인 기술을 제안한다.

Ⅱ. 관련 연구

링크드 데이터 탐색과 저장에 관한 기존의 연구는 크게 세 가지 접근 방식, 즉 RDF 트리플을 중앙 저장소에 복사 관리하는 방식(중앙집중 방식), 분산된 SPARQL 엔터포인터 상에서 쿼리를 수행하는 방식(분산 방식), 그리고 압축된 로컬 인덱스 구조를 사용하는 방식(인덱스 방식)으로 분류할 수 있다.

첫째, 중앙집중 방식(Centralized Approach)은 웹

문서 검색엔진과 비슷한 방식으로 중앙 저장소에 모든 데이터를 복사하는 방법이다. 이러한 저장소를 사용하면 우수한 쿼리 응답 시간을 제공할 수 있다. 그러나 많은 단점들이 있을 수 있는데, 쿼리 결과가 최신의 데이터를 반영하지 못하고, 모든 데이터를 저장함에 따라 저장 비용이 많이 들며 이로 인해 성능이 저하될 수 있다. 또한 사용자는 저장소에 복사된 링크드 오픈 데이터(LOD, Linked Open Data)의 일부분만 사용하게 된다.

둘째, 분산 방식(Distributed Approach)은 기존 관계형 연합 시스템(Federation System)에서의 작업과 크게 다르지 않다. 이 방식은 몇 가지 장점을 제공하는데, 복사된 데이터를 동기화할 필요가 없고, 쿼리는 최신의 데이터로 더욱 역동적으로 처리될 수 있으며, 데이터를 인덱싱하고 통합하는데 시간이 걸리지 않고 새로운 리소스를 쉽게 추가할 수 있다. 또한 이러한 시스템은 저장 공간이 거의 필요 없다. 그러나 단점으로는 모든 게시자들이 그들의 링크드 데이터에 대한 신뢰할 수 있는 SPARQL 엔드포인트를 제공한다고 보장할 수 없는 것이다.

셋째, 인덱스 방식(Indexed Approach)은 중앙집중 방식과 분산 방식 사이의 하나의 절충안이라 할 수 있다. 즉 모든 RDF 트리플들을 중앙 저장소에 복사하는 대신에 로컬에 압축된 형태의 인덱스 구조를 만드는 것이다. 실제로 중앙집중 방식은 분명히 더 나은 성능을 제공할 수는 있지만 변화가 심한 링크드 데이터에서 쿼리된 결과가 최신이 아닐 수가 많다. 인덱스를 사용함으로써 분산되어 있는 데이터 리소스들을 효율적으로 검색할 수 있고, 온 디맨드(On-demand) 방식으로 데이터가 엑세스될 때 실제로 필요로 하는 데이터만 신속하게 줄일 수 있다. 표 1은 중앙집중 방식, 분산 방식, 그리고 인덱스 방식을 요약・정리한 것이다.

III. 링크드 데이터 인덱싱 기법

본 연구에서는 효율적인 링크드 데이터 질의 처리를 위하여 2장에서 기술된 인덱스 방식을 확장시킨다. Umbrich 등은 링크드 데이터 요약을 위해 다차원 히스토그램 기법을 제안하였는데, 이 기법을 요약하면 RDF 트리플이 해시 함수에 의해 3차원 공간상의 점으로 변환되고, 검색 시에는 쿼리 트리플 패턴이 3차원 공간상의 선(또는 면적)으로 변환된다. 따라서 검색 패턴에 관련이 있는 URI들은 선이나 면적에 걸쳐 있는 버켓(Bucket)에 존재해 있게 된다.

Harth 등[14]은 균일 분포만 지원되는 다차원 히스토그램을 개선하기 위해 QTree를 사용하였다. 이 기법은 다차원 히스토그램과 R-tree를 결합한 것으로서 두개 데이터 구조의 이점 모두를 취하고 있다. QTree는 다양한 크기의 버켓 사이즈를 가질 수 있는데 이는 희소한 집합이나 많은 클러스터가 존재하는 공간에서 효율적으로 처리될 수 있다.

하지만, 다차원 히스토그램과 QTree와 같은 근사(Approximation) 인덱싱 기법들은 인덱스 단계에서 정확한 필터링을 수행하지 못하므로 많은 불필요한 후보 객체, 즉 잘못된 히트(False Hit)들이 존재할 수 있다. 이로 인해 단순 쿼리 패턴에는 큰 무리가 없어 보이나 복잡한 조인(Join) 질의 처리 때에는 데이터 검색 시간이 상당히 늦어지는 문제점을 내포하고 있다.

표 1. 중앙집중 방식, 분산 방식, 그리고 인덱스 방식
Table 1. Centralized, distributed, and indexed approaches

Type	Description	Related Work
Centralized	Copying all data into a centralized local system	Quad[4], RDF-3X[5], Hexastore[6], Matrix[7], Path[8], PIG[9]
Distributed	Performing queries over SPARQL endpoints provided by Linked Data publishers	DARQ[10], SemWIQ[11], Federator[12], LiveExploration[13]
Indexed	A hybrid method between the centralized approach and the distributed approach	MDH[2], QTree[14]

즉 조인 질의 처리 시에 정확한 필터링이 불가능하여 각각의 서브쿼리에서 반환되는 많은 RDF 트리플 데이터를 모두 비교하여야 하는 문제점을 야기한다. 이로 인해 조인 연산이 필요한 복합 질의 처리 시에 고비용의 결과를 초래한다.

본 연구에서는 복합 질의 처리의 성능 향상을 위해 다차원 히스토그램 개념을 확장·발전시킨다. 확장된 인덱스 기법을 구축하기 위해서는 먼저 리소스에 의해 제공되는 RDF 트리플들을 3차원 공간의 점으로 변환한다. 이 방법은 RDF 트리플의 개별 구성요소들에 대해 해시 함수(예, 자바의 hashCode())를 적용하여 각 트리플에 대한 3개의 숫자를 얻는다. 예를 들면, (김철수, liveIn, 서울)이 (652, 143, 484)와 같이 변환된다. 다음 단계에서 3차원 공간이 분리된 영역으로 분할되며(여기서 분할된 영역들을 버켓이라 정의한다), 각 버켓에는 해당 영역에 속하는 변환된 RDF 트리플 정보와 URI 엔트리들이 포함된다. 한편, 쿼리에 대한 트리플 패턴이 주어지면 쿼리에도 동일한 해시 함수를 적용하여 숫자 트리플을 계산하고, 쿼리 트리플과 관련된 버켓들을 결정한다. 최종적으로 결정된 버켓들을 룩업함으로써 실제로 쿼리와 연관성 있는 리소스들을 검색할 수 있다.

여기서 우리는 기존의 QTree의 단점을 보완한 이단계 색인 구조(Two-step Index Structure)를 제안한다. 이단계 색인 구조는 그림 1과 같이 불규칙 3차원 공간 포인터들을 효율적으로 관리할 수 있는 MBB(Minimum Bounding Box) 기반 색인 구조에 버켓 내 포인터들을 구조화할 수 있는 링크드 리스트(Linked List) 구조를 결합시켰다. MBB 기반 색인 구조로는 현재 가장 널리 사용되고 있는 R*-tree[15]가 선택되었으며, 기존 2차원만 지원되던 R*-tree 구조를 3차원이 지원되도록 다시 프로그래밍하였다.

3차원 R*-tree는 리프 노드와 중간 노드로 구성되어 있으며, 리프 노드에는 버켓 식별자와 MBB, 그리고 링크드 리스트에 대한 주소 지시자를 포함하고 있다. 중간 노드는 자식 노드의 주소 지시자와 자식 노드의 박스 영역을 포함하고 있다. R*-tree 하부에 있는 링크드 리스트는 구성 요소의 식별자와 URI 쌍들을 포함하고 있으며, 링크드 리스트의 실제 구현은 2차원 배열을 사용한다.

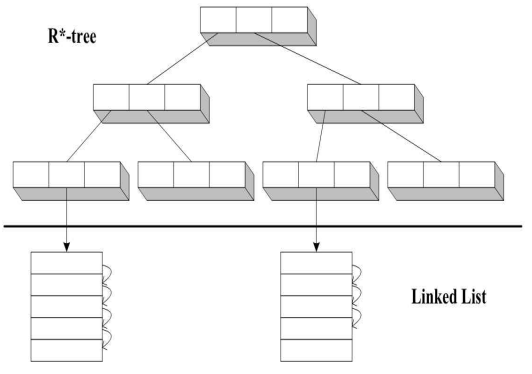

그림 1. 이단계 색인 구조
Fig. 1. Two-step index structure

RDF는 일반적으로 주어, 술어, 목적어의 트리플 구조, 즉 (s, p, o)로 표현되는데, 본 연구에서는 링크드 데이터 복합 질의를 위하여 출현 횟수 기반 다차원 히스토그램 개념[3]을 이차원 색인 구조에 적용시켰다. 본 시스템에서는 3차원 좌표가 이단계 색인 구조에 하나씩 삽입되며, 링크드 리스트에서는 트리플 좌표와 URI 쌍들의 리스트가 테이블 형태로 저장된다. 여기서 트리플 테이블의 각 리소스는 조인 쿼리의 속도 향상을 위해 3차원 좌표에 두 개의 출현 횟수 숫자를 추가로 첨가하였다. 출현 횟수는 s#과 o#으로 기술되는데, s#은 RDF 데이터셋에서 목적어 o가 다른 RDF 트리플에서 주어 s에 출현되는 횟수를 나타내고, 마찬가지로 o#은 s가 o에 출현되는 횟수를 나타낸다. 따라서 RDF 트리플 (s, p, o)는 (x, y, z, s#, o#)로 표현되는데, 여기서 x, y, z는 해시 함수에 의해 변환된 x, y, z 축 상에 존재하는 좌표 값들이고 s#과 o#은 어커런스 값들이다.

Ⅳ. 인덱스 기반 조인 질의 처리

SPARQL은 RDF 데이터 모델을 위한 표준 질의 언어이며, SPARQL에서 트리플 패턴(Triple Pattern)이란 주어, 술어, 목적어 위치에 접두사 ?를 가지고 있는 쿼리 변수를 포함할 수 있는 RDF 트리플을 말한다. SPARQL 쿼리에서는 8개의 트리플 패턴이 있다: (s, p, ?o), (?s, p, o), (s, ?p, o), (s, ?p, ?o), (?s, ?p, o), (?s, p, ?o), (?s, ?p, ?o), (s, p, o). 이들 중에서 (?s, ?p, ?o)는 전체 스캔이 필요하며 (s, p, o)는

일치하는 트리플의 수가 0 또는 1이다. 따라서 이 둘을 제외한 6개 트리플 패턴의 성능을 고려할 필요가 있다.

이단계 색인 구조를 이용한 트리플 패턴 쿼리는 다음과 같이 2단계로 수행된다. (1) 질의가 주어지면 우선 R*-tree를 사용하여 쿼리를 만족하지 않는 버켓들을 신속하게 제거한다(여과 단계). 그러나 단순한 MBB 근사 기법으로는 정확한 객체들을 탐색할 수는 없다. (2) MBB 안에 있는 리소스 집합으로부터 쿼리와 매칭되는 정확한 객체들을 찾는다(정제 단계). 링크드 데이터 환경에서 단일 트리플 질의는 비교적 처리가 간단하지만, 여러 개의 트리플 패턴이 포함된 복합 조인 연산은 상당히 복잡하여 시간이 많이 걸릴 수 있다. 일반적으로 SPARQL 질의는 포함되는 트리플 패턴의 개수에 따라 필요한 조인 연산수도 증가하기 때문에 대부분 많은 조인 연산을 필요로 한다.

조인 알고리즘은 RDBMS로부터 잘 알려져 있는 다양한 기법들(예, merge join, hash join, nested-loop join 등)을 활용하여 구현할 수 있다. 본 연구에서는 이들 중 보편적인 nested-loop 조인 알고리즘을 사용하였으며, 어커런스 기반 히스토그램을 활용함으로써 복합 조인 연산 때 요구되는 데이터 비교 연산을 상당히 줄일 수 있다. 본 논문에서 제안하는 인덱스 기반 조인 질의 처리를 설명하기 위해 예를 들어 (s, p, ?o) ⋈ (?s, p, ?o) 형태의 질의 처리 과정을 고려해 본다.

다음 SPARQL 쿼리는 김철수의 친구들이 참여한 프로젝트들을 질문하고 있다. 여기서 질의문은 변수 ?k에 의해 조인되는 두 개의 트리플 패턴으로 구성된다.

```
SELECT ?p WHERE {
    user:김철수   foaf:knows   ?k .
    ?k   foaf:project   ?p          .  }
```

이러한 형태의 SPARQL 질의문에 대한 적절한 결과를 반환하기 위해서는 다음과 같은 처리 과정이 필요하다(그림 2 참조).

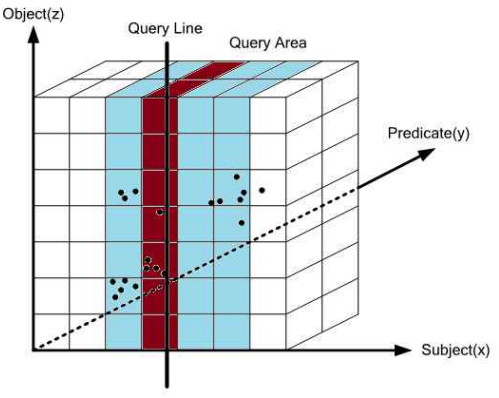

그림 2. 질의 선과 질의 면
Fig. 2. Query line and query area

① (s, p, ?o) 형태의 트리플 패턴을 처리하기 위해 먼저 해시 함수를 적용하여 3차원 공간상의 좌표로 변환시킨다. 그러나 RDF 트리플과는 달리 이 쿼리 패턴에는 ?o 변수가 포함되어 있다. 따라서 이러한 변수 때문에 포인터 대신 x, y축 상에 z 값으로 이루어진 질의 선(Query Line)이 사용된다. 결국 R*-tree를 사용하여 이 선에 걸쳐 있는 객체들을 탐색한다.

② (?s, p, ?o) 형태의 트리플 패턴을 처리하기 위해서 ①과 비슷한 과정을 거치지만 여기서는 ?s와 ?o 변수가 포함되어 있다. 따라서 이는 y축 상에 존재하는 x, z 값으로 이루어진 질의 면(Query Area)이 사용된다. 결국 R*-tree를 사용하여 이 면에 걸쳐 있는 객체들을 탐색한다.

③ 위의 두 과정을 통해 탐색된 객체들에 대하여 조인 연산을 실행하는데, 본 논문에서 제안하는 어커런스 기반 조인 알고리즘의 수행 과정은 다음과 같다.

과정③의 조인 연산을 수행하기 위해 과정①의 결과물의 목적어(o)와 과정② 결과물의 주어(s)를 비교하여 그 둘이 동일하면 최종 결과물로 선택된다. 먼저 과정①의 첫 번째 결과물에서 s#을 체크하여 s# ≠ 0이 아닌 경우 과정②의 결과물과 조인 작업을 수행한다. 이때 과정①의 o와 과정②의 s가 일치하는지 조사하는데, 조인 조건이 충족되면 그 튜플을 선택하여 최종 결과물에 첨가하고 s#을 1 줄여

준다. 이 과정은 s#이 0이 될 때까지 반복된다. s#이 0이 되면 비록 과정②의 나머지 결과 데이터가 남아 있더라도 비교를 수행하지 않고 과정①의 다음 비교 대상으로 전환하게 된다. 따라서 본 알고리즘은 쿼리와 일치하지 않는 것으로 확인된 불필요한 탐색을 신속하게 제거할 수 있으므로 최종 조인 반환시간을 상당히 감소시킬 수 있게 된다.

Ⅴ. 실험 분석

본 논문에서 제안하는 이단계 색인 구조의 성능을 분석하기 위해 여러 가지 실험을 수행하였다. 이를 위해 대표적인 중앙집중 방식과 분산 방식, 그리고 인덱스 방식으로 잘 알려져 있는 Quad[4]와 LiveExploration[13], 그리고 QTree[14]를 이단계 색인 구조(이하 2Step이라 한다)와 비교·분석하였다. 본 실험에서는 실질적이고 의미 있는 결과를 얻기 위해 다양한 RDF 자료가 사용되었다. 이러한 RDF 자료는 현존하는 링크드 오픈 데이터에서 사용되고 있는 실제 링크드 데이터[16]로부터 선택되었다.

본 논문에서는 두 가지 실험을 수행하였는데 첫 번째 실험에서는 쿼리 탐색 시간을 측정하고 두 번째 실험에서는 저장 공간을 측정하였다.

그림 3은 각 방식의 쿼리 실행 시간을 보여 주고 있다. 세로축은 시간(단위: 초)을 가로축은 RDF 트리플 수(단위: 천개)를 나타낸다. 그림에서 Quad, 2Step, Qtree가 비교적 만족할 만한 성능을 얻은데 반해 Live는 가장 느리다. 2Step과 Qtree는 Quad보다는 약간 느린데 이는 Quad가 로컬에 전체 데이터를 복사하여 빠른 반응시간을 보장하는 반면 2Step과 Qtree는 인덱스 수행은 빠르나 결국 최종 결과물은 URI를 이용한 원격 탐색을 수행하기 때문이다. 2Step과 Qtree의 성능 비교는 트리플 수가 적을 때는 거의 비슷해 보이지만 트리플 수가 늘어남에 따라 성능 차이가 조금씩 커지는 것을 관찰할 수 있다. Live의 성능이 제일 나쁜 이유는 모든 쿼리가 데이터 공급자에 의해 제공되는 SPARQL 엔터포인터 상에서 수행되기 때문이다. 실험 결과 Quad와 2Step이 Live보다 평균 75%와 59%의 탐색 시간 향상을 보인다.

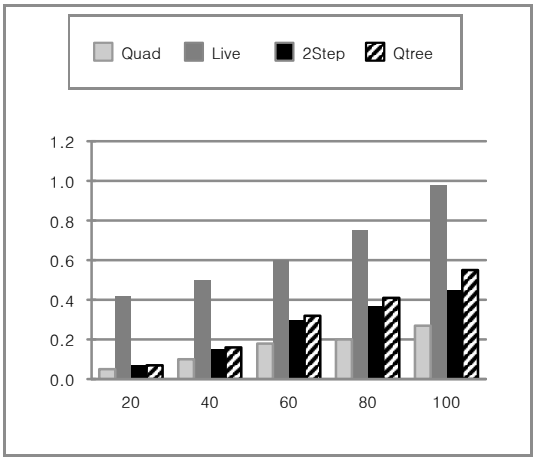

그림 3. 쿼리 탐색 시간
Fig. 3. Query searching time

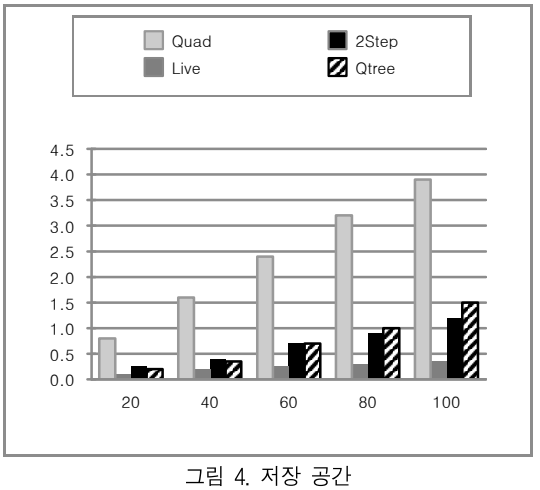

그림 4. 저장 공간
Fig. 4. Storage space

Live, 2Step, Qtree의 가장 큰 이점은 저장 공간의 절약이다. 그림 4에서 데이터 저장 공간을 보여 주기 위해 세로축은 저장량(단위: MB)을 가로축은 RDF 트리플 수(단위: 천개)를 나타낸다. 그림에서 알 수 있듯이 Live, 2Step, 그리고 Qtree가 Quad에 비해 상당히 적은 저장 공간을 필요로 하는데 그 이유는 셋 다 전체 RDF 데이터를 저장할 필요가 없기 때문이다. 2Step과 Qtree가 Live보다 조금 더 많은 공간을 요구하는데 이는 2Step과 Qtree는 분산 방식인 Live와는 달리 보조 색인 구조를 가지고 있기 때문이다. 2Step과 Qtree를 비교하면 두 기법의 저장 공간은 거의 비슷하지만 특이한 점은 트리플

수가 적을 때는(60 이하) Qtree가 약간 낮지만 이후부터는 Qtree가 조금씩 높게 나타난다. 한편, Quad는 중앙 저장소에 RDF 트리플 전체를 복사하기 때문에 가장 많은 저장 공간을 필요로 한다. 실험 결과 Live와 2Step은 Quad에 비해 평균 90%, 71% 저장 공간의 절약을 보인다.

결론적으로 2Step과 Qtree가 최상의 성능은 아니지만 실험 결과에서 알 수 있듯이 전반적으로 성능이 양호한 편이다. 이들 색인 기법들의 주요 이점은 압축된 저장 공간으로 신속한 검색을 지원하는 것이다. 따라서 인덱스 방식이 중앙집중 방식과 분산 방식보다는 더 효율적이고 유용하다 할 수 있다. 특히, 인덱스 방식 중 본 논문에서 제안하는 2Step과 기존의 Qtree를 비교했을 때 트리플 수가 적을 때는 비슷한 성능을 보이지만 트리플 수가 증가함에 따라 2Step의 성능이 더 좋아지는 것을 확인할 수 있다.

Ⅵ. 결 론

효율적인 링크드 데이터 질의는 시맨틱 웹 분야에서 가장 도전적인 목표 중 하나이다. 본 논문에서는 링크드 데이터 저장 및 검색을 위하여 기존의 색인 구조를 확장하였다. 제안된 색인 구조는 질의를 만족시키지 못하는 버켓들을 신속히 여과시킬 수 있는 R*-tree와 정확한 결과를 효율적으로 정제시킬 수 있는 링크드 리스트의 이단계로 구성되어 있다. 이단계 색인 구조(즉, 2Step)는 기존의 중앙집중 방식과 분산 방식, 그리고 인덱스 방식인 Qtree에 비해 최상의 성능은 아니지만 쿼리 탐색 시간과 저장 공간 두 측면에서 전반적으로 양호한 성능을 보여주고 있다.

한편, 링크드 데이터는 시간이 지남에 따라 상당히 많은 양의 데이터 변경이 일어날 수 있으므로 데이터 변동성에 관한 내용은 제안되는 인덱스 구조에서 무시할 수 없는 중요한 사항이다. 지금까지 몇 개의 논문(예, Umbrich[6] 및 Svoboda [17])에서 인덱스 관리 문제에 대해 간단하게 언급은 하였으나 이를 자세히 다루고 있는 연구는 거의 찾아볼 수가 없다. 따라서 변동이 심한 링크드 데이터의 변경을 효율적으로 지원할 수 있는 동적 레이블링 방법과 같은 연구가 향후 필요하다.

Referencees

[1] http://www.wikipedia.org, [Accessed: Aug. 16. 2017.]

[2] J. Umbrich, K. Hose, M. Karnstedt, A. Harth, and A. Polleres, "Comparing data summaries for processing live queries over Linked Data", World Wide Web, Vol. 14, No. 5-6, pp. 495-544, Oct. 2011.

[3] Y. J. Lee, "EMH(Extended Multidimentional Histogram) Based on Occurrence for Querying Linked Data", Journal of KIIT, Vol. 13, No. 3, pp. 121-128, Mar. 2015.

[4] A. Harth and S. Decker, "Optimized index structures for querying RDF from the Web", The 3rd Latin American Web Congress (LA-Web), pp. 71-81, Nov. 2005.

[5] T. Neumann and G. Weikum, "RDF-3X: A RISC-style engine for RDF", The 34th International Conference on Very Large Data Bases (VLDB), pp. 647-659, Aug. 2008.

[6] C. Wess, P. Karras, and A. Bernstein, "Hexastore: Sextuple indexing for semantic Web data management", The 34th International Conference on Very Large Data Bases (VLDB), pp. 1008-1019, Aug. 2008.

[7] M. Atre, V. Chaoji, M. J. Zaki, and J. A. Hendler, "Matrix "Bit" loaded: A scalable lightweight join query processor for RDF data", The 19th International Conference on World Wide Web (WWW), pp. 41-50, Apr. 2010.

[8] B. Liu and B. Hu, "Path queries based RDF index", The 1st International Conference on Semantics, Knowledge and Grid, pp. 91-93, Nov. 2005.

[9] T. Tran and G. Ladwig, "Structure index for RDF data", Workshop on Semantic Data Management (SemData@VLDB), Sep. 2010.

[10] B. Quilitz and U. Leser, "Querying distributed RDF data sources with SPARQL", The 5th European Semantic Web Conference (ESWC),

Vol. 5021, Lecture Notes in Computer Science, pp. 524-538, Jun. 2008.

[11] A. Langegger, W. Wob, and M. Blochl, "A semantic middleware for virtual data integration on the Web", The 5th European Semantic Web Conference (ESWC), Vol. 5021, Lecture Notes in Computer Science, pp. 493-507, Jun. 2008.

[12] O. Gorlitz and S. Staab, "Federated data management and query optimization for linked open data", New Directions in Web Data Management 1, Springer Berlin Heidelberg, pp. 109-137, Jan. 2011.

[13] O. Hartig, "An overview on execution strategies for Linked Data queries", Datenbank Spektrum, Vol. 13, No. 2, pp. 89-99, Jul. 2013.

[14] A. Harth, K. Hose, M. Karnstedt, A. Polleres, K. U. Satler, and J. Umbrich, "Data summaries for on-demand queries over Linked Data", The 19th International Conference on World Wide Web (WWW), pp. 411-420, Apr. 2010.

[15] N. Beckmann, H. P. Kriegel, R. Schneider, and B. Seeger, "The R*-tree: An efficient and robust access method for points and rectangles", The 1990 ACM SIGMOD International Conference on Management of Data, pp. 322-331, May 1990.

[16] http://linkeddata.org/data-sets, [Accessed: Jul. 10. 2017]

[17] M. Svoboda, "Efficient querying of distributed Linked Data", The 2011 Joint EDBT/ICDT PhD Workshop, pp. 45-50, Mar. 2011.

저자소개

이 용 주 (Yong-Ju Lee)

1985년 2월 : 한국과학기술원 산업공학과 정보검색전공(공학석사)
1997년 8월 : 한국과학기술원 정보및통신공학과 컴퓨터공학전공(공학박사)
1998년 8월 ~ 현재 : 경북대학교 IT대학 컴퓨터학부 교수
관심분야 : 시맨틱 웹, 빅데이터, 웹 데이터베이스