빅데이터 활용을 위한 클라우드 기반의 링크드 데이터 인덱싱 시스템

이민아 ^O^ 정진욱 김응희 김홍기
서울대학교 의생명지식공학연구실
minalee@snu.ac.kr^O^, cool_uk@bike.snu.ac.kr eungheekim@snu.ac.kr hgkim@snu.ac.kr

Linked Data Indexing System for Big Data Processing on the Cloud System

Mina Lee^O^ Jinuk Jung Eung-hee Kim Hong-gee Kim
Biomedical Knowledge Engineering Laboratory, Seoul National University

요 약

2000년대 초반 등장한 시맨틱 웹 기술은 최근 재조명을 받고 있다. 이는 초기에 구축된 시맨틱 데이터와 최근에 구축하는 시맨틱 데이터의 양적 비교를 통해서도 알 수 있다. 그러나 기존의 시맨틱웹 기술은 대용량 데이터를 처리하는데 어려움이 많아, 이를 처리하기 위한 기술이 중요한 문제로 대두되고 있다. 본 논문에서는 앞에서 말한 바와 같이, 기존 RDF Repository의 대안으로, 다양한 데이터 베이스를 복합적으로 사용하였다. RDF 데이터를 효율적으로 처리하기 위해, NoSQL DB와 메모리 기반 관계형 DB를 활용하여 시스템을 구성하였다. 또한, 사용자가 이에 대한 별도의 지식 없이 기존의 SPARQL 질의를 그대로 사용하여, 원하는 결과를 얻을 수 있는 시스템을 제안한다.

1. 서론

웹은 2004년, 팀 오레일리가 제창한 web 2.0이라는 키워드와 함께, 현재까지 큰 변화를 겪어왔다. 웹을 하나의 플랫폼으로 강조한 웹 2.0은 사용자참여와 공유를 바탕으로, 컨텐츠가 중심이 되는 인터넷 환경을 뜻한다[1]. 특히, 웹 2.0의 발전은 시맨틱웹의 대중화와 함께 새로운 국면을 맞이했다. 시맨틱웹 온톨로지 기술을 활용하여, 분산되어 있는 컨텐츠에 일관적인 메타 데이터를 기술함으로써 정보를 통합하고 재사용을 가능하게 했다[2]. 이렇게 통합된 정보들은 시맨틱웹 목적에 맞게 RDF Repository에 저장되고, SPARQL Endpoint를 통해 데이터가 외부에 공개되고 있다.

또한, 웹 2.0의 성공과 모바일 기기의 발전은 데이터의 양의 급증으로 이어져있으며, 이로 대용량 데이터 처리를 위한 다양한 시스템이 새롭게 등장했다. 그러나, 기존 시맨틱웹 기술은 대용량 데이터를 고려하지 않고 설계 되었기 때문에 대용량 트리플 데이터를 저장소를 구현하는데 어려움이 있다.

본 논문에서는 대용량의 링크드 데이터를 빠르게 처리하기 위해서 오픈 소스 프레임워크를 활용한 시스템을 설계 구현하였다. 본 시스템은 관계형 DB와 NoSQL DB를 조합한 저장소를 구성하고, 구현된 SPARQL 파서를 이용한 SPARQL Endpoint를 제공함으로써, 분산형 시맨틱 저장 환경을 제공한다.

2. 관련 연구

NoSQL은 Not Only SQL의 약자로, 기존 관계형 데이터 베이스와 차별적인 저장 방식을 가진 데이터 베이스를 지칭한다[3]. NoSQL은 관계형 데이터 베이스가 가진 복잡한 구조와 용량의 한계를 해결하기 위한 목적으로, Non-relational, Distributed, Schema-free, Scalable한 특징을 가지고 있다. NoSQL은 저장 방식을 기준으로 Key-Value, 문서방식, Column방식, 그래프 방식으로 분류한다. 특히, NoSQL DB는 기존의 관계형 데이터 베이스에 비해 데이터를 읽고 쓰는데 빠른 퍼포먼스를 보이기 때문에, 다른 DB에 비해 대용량의 데이터를 처리하는데 빠른 응답이 가능하다.

최근 들어 대용량의 링크드 데이터를 처리하기 위해서 NoSQL을 이용해 시스템을 구현하기 위한 연구들이 계속해서 진행되고 있다[4]. 다만, NoSQL의 특성상 기존의 관계형 데이터 베이스와는 달리 '관계'를 표현하는데 한계가 있다. 기존의 연구들은 그 한계를 인정하고, JOIN 쿼리와 같이 관계 기반의 복잡한 질의를 제한하여 RDF 저장소로써 대체하기에 한계가 있다.

3. 대용량 링크드 데이터 인덱싱 시스템의 구현

그림1은 링크드 데이터 인덱싱 시스템의 개요를 보여주는 구조도이다. 본 논문은 구조도의 순서에 따라 데이터 입력, 데이터 포맷 변환, 데이터 저장, 데이터 처리로 순으로 본 시스템의 기능에 대해 설명한다.

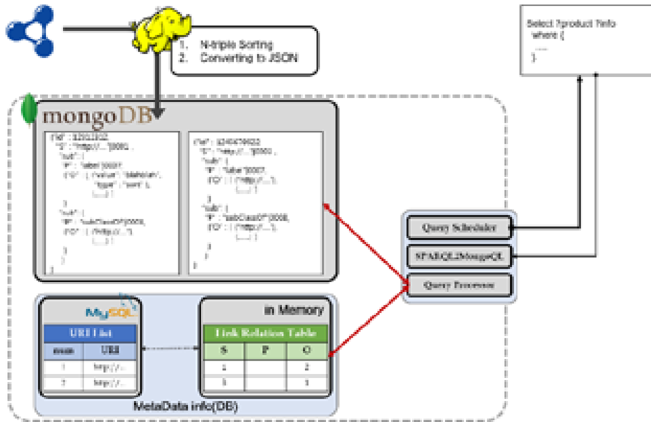

그림1. 링크드 데이터 인덱싱 시스템 구조도

3.1 입력 데이터 (N-triples 파일)

본 시스템이 입력 받는 링크드 데이터 파일은 RDF 트리플 리스트로 구성되어 있다. RDF 트리플은 주어(Subject), 술어(Predicate), 목적어(Object)로 구성된 구조적인 문장을 가리킨다. #기호(또는 /기호) 앞부분은 Namespace이며 일반적으로 도메인(분야) 또는 출처를 나타낸다. RDF 트리플을 기술하는 방식은 RDF/XML과 N-triples 등이 있다.

RDF/XML는 동일한 Subject를 가진 Predicate과 Object를 Subject 서브 레이어에 작성하는 트리 구조를 가지며, 공간이 절약된다는 장점이 있다. 그러나 트리플을 순차적으로 빠르게 읽고 처리하는 대용량 트리플 리스트에는 적합하지 않다.

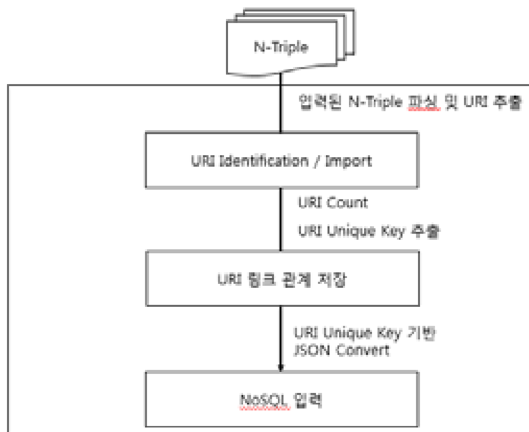

그림 2. 데이터 입력 순서도

N-triple은 하나의 트리플을 한 줄에 <Subject>, <Predicate>, <Object> .의 형식으로 구성되어 있다. 동일한 Subject를 매번 작성하기 때문에 RDF/XML에 비해 차지하는 공간이 크다는 단점에도 불구하고, 트리플을 순차적으로 읽기 용이하여 대용량 트리플 리스트를 처리하는데 적합하다. 따라서 본 연구는 N-triple 포맷의 링크드 데이터 파일을 지원한다.

3.2 데이터 포맷 변환

본 연구에서는 문서방식의 NoSQL DB에 데이터를 저장하고, 데이터에 질의를 하기 위해서는 N-Triple포맷의 RDF 트리플 데이터를 JSON 형태로의 변환하는 작업이 선행되어야 한다.

RDF 트리플 데이터를 JSON으로 표현하는 표준 포맷은 RDF/JSON방식과 JSON-LD방식 두 가지로 나뉘어진다[5,6].

RDF/JSON과 JSON_LD 방식은 Key 값으로 URI를 사용하며, 이를 그대로 NoSQL에서 사용할 경우, 인덱스를 구축이 되지 않아 URI검색에 많은 시간이 소요된다. RDF/JSON 방식의 경우, 동일 Predicate에 해당하는 Object를 하나로 묶어 사용하는 JSON_LD와 달리 중복된 Predicate를 모두 작성하기 때문에 문서 저장 시 많은 용량을 차지한다. 대용량 데이터의 경우에는 효율적인 데이터 처리를 위해 저장소 저장 용량을 고려하여 적합한 JSON포맷 변환이 불가피하다.

대용량 N-Triple 데이터 변환은 Hadoop Framework를 이용하여 처리되며, 변환 작업은 데이터 정렬 작업과 데이터 포맷 변환으로 나누어진다. 데이터 변환 직전, 데이터를 효율적으로 관리하기 위해 데이터 정렬 작업이 선행된다.

```
{“s”, “http://example.org/about”,
   "sub" : {
      “p” : “http://purl.org/dc/terms/title” ,
         {"o" : [ {“value”: Anna’s Homepage”,
                  “type” : "literal",
                  “lang” : “ko" },
                     {....} ]
         }
```

표1. 변환된 RDF 트리플 데이터 구조

3.3 데이터 저장

JSON 포맷으로 변환된 트리플 데이터는 자체적으로 개발된 ‘JSON2REPO’ 프로그램을 통해 해당 저장소가 허용하는 크기에 맞게 병합되어 저장소에 입력된다. 이 때 데이터는 원본 형태로 저장되는 것이 아니라 관계성을 표현하지 못하는 NoSQL의 한계를 보완하기 위해 #Namespace의 공통된 URI는 분리되어 관계형 데이터베이스에 저장된다. 각 Subject와 Object의 관계 정보를 in-Memory RDB에 저장하는 것을 통해 각 트리플간의 관계 표현과 응답속도를 높일 수 있다.

In-Memory RDB는 시스템의 메모리를 이용하기 때문에 메모리의 크기보다 큰 데이터를 저장할 수 없다. 트리플 데이터의 특성상 URI는 긴 String 타입으로, 그대로 저장할 경우 시스템 메모리 용량을 초월할 가능성이 높다. 본 시스템에서는 외부 RDB에 URI와 Object를 저장하고, 이에 대한 Key를 in-Memory RDB에 저장함으로써 이 문제를 해결할 수 있다. 복합형 DB를 활용함으로써 RDB가 제공하는 그래프 데이터 검색의 장점을, NoSQL이 가지는 대용량 데이터에 대한 빠른 응답 속도와 Full-Text Search의 이점을 모두 취할 수 있게끔 구현하였다.

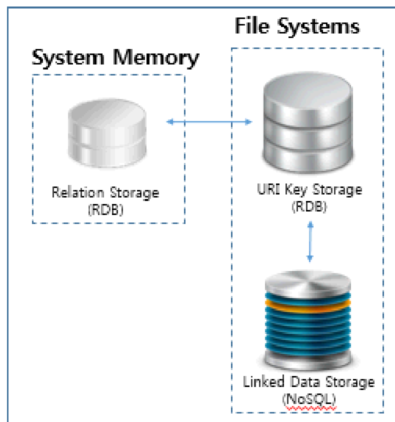

그림3. 링크드 데이터 저장소 구조도

3.4 데이터 처리

저장된 JSON 포맷의 트리플 데이터는 SPARQL 질의를 통해 불러오게 된다. NoSQL 질의는 SPARQL 질의와 다른 형태를 가지므로, SPARQL질의의 Subset Parser를 구현하여 SPARQL 질의를 NoSQL 질의로 변환한다. 사용자가 입력한 SPARQL 질의는 질의 분할기(Query Splitter)에 의해 NoSQL로 질의 할 수 있는 범위만큼 나뉘어진다. 이때, 분할된 질의를 Query Scheduler의 기준에 따라 질의 순서를 정렬하여 저장소에 보내게 된다.

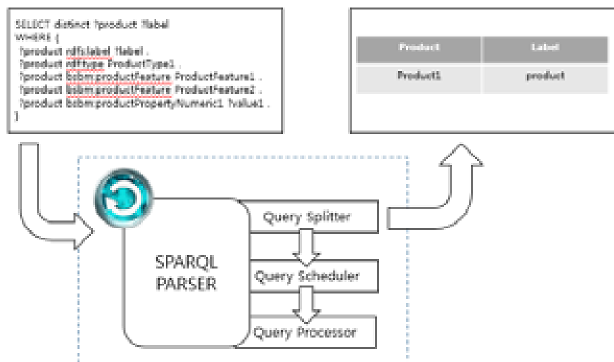

그림 4. SPARQL질의 처리 순서도

Query Scheduler는 모든 데이터에 각각의 쿼리를 보내면서 소요하는 시간을 단축시킴으로써 효율적인 데이터 처리를 가능하게 한다. 처리방식은 각각의 변환된 쿼리 한 줄이 처리하는 데이터의 양이 가장 적은 순서대로 정렬하고, 그에 적합한 데이터와 일치하는 데이터를 한정 지어 나머지 쿼리를 수행한다.

Query Processor는 한 줄로 나뉘어진 쿼리를 각각 NoSQL, RDB, in-Memory DB에 전송하여 수행된 결과를 SPARQL질의 문장에 맞게끔 조합하는 역할을 수행한다. 이에 따른 결과로써, 사용자는 최종적으로 SPARQL 질의에 따른 응답을 받게 된다.

4. 결론 및 향후 연구

본 논문에서는 대용량 트리플 데이터의 효율적인 활용을 위한 인덱싱 시스템을 설계 구현하였다. 3.2-3장에서는 데이터를 저장하기 위해서 N-Triple 포맷의 링크드 데이터를 변환하고, 변환한 데이터를 분리하여 각각 NoSQL, RDB, in-Memory에 저장하는 것에 대해 설명하였다. 3.4장은 사용자가 저장된 링크드 데이터를 SPARQL 질의를 통해 사용을 가능하게 만드는 데이터 처리 기술에 대해 알아보았다.

본 연구의 향후 과제는 본 시스템과 연계되어 있는 다른 시스템에서 비정기적으로 제공되는 링크드 데이터를 동적으로 갱신을 가능하게 하는 기술을 연구하는데 중점을 두고 진행될 것이다.

Acknowledge

본 연구는 미래부가 지원한 2013년 정보통신·방송(ICT) 연구개발사업의 연구결과로 수행되었음

참고 문헌

[1] T. O'Reilly, "What is Web 2.0 : Design Patterns and Business Models for the Next Generation of Software", 2005.

[2] C. Bizer, T. Heath, T. Berners-Lee, "링크드 데이터 - the story so far", International Journal on Semantic web and Information Systems, Vol. 5, No. 3, 2009.

[3] C. Rick, Scalable SQL and NoSQL data stores, ACM SIGMOD Record, 2011, 39.4: 12-27.

[4] M. Hausenblas, R. Grossman, A. Harth, P. Cudré-Mauroux, Large-scale 링크드 데이터 Processing-Cloud Computing to the Rescue?, In CLOSER (pp. 246-251), 2012.

[5] RDF 1.1 JSON Alternate Serialization https://dvcs.w3.org/hg/rdf/raw-file/default/rdf-json/index.html

[6] JSON-LD, http://www.w3.org/TR/json-ld-syntax/