HBase를 이용한 분산 시맨틱 웹 데이터 저장소에 대한 연구

조대웅* , 김명호*
*숭실대학교 컴퓨터공학과
e-mail:jodw@ssu.ac.kr, kmh@ssu.ac.kr

A Study on Distributed Semantic Web Data Repository Using HBase

Daewoong Jo* , Myung Ho Kim*
*Department of Computing, SoongSil University

요 약

실시간으로 발생되는 대량의 데이터를 효율적으로 저장하기 위한 연구는 분산/병렬 처리를 위한 하둡 및 NoSQL과 관련한 빅 데이터 처리 기술을 통해 진행 중에 있다. 하지만 시맨틱 웹 분야에서 발생되는 대량의 데이터를 처리하기 위한 모델은 현재 연구가 진행되고 있지 않다. 본 논문에서는 시맨틱 웹 환경에서 발생되는 대량의 온톨로지 데이터를 빅 데이터 처리가 가능한 NoSQL 분야인 HBase 데이터베이스에 분산 저장할 수 있는 매핑 규칙을 제안한다. 이와 같은 매핑 규칙을 통해 시맨틱 웹 환경에서도 대량으로 발생될 수 있는 데이터들을 효율적으로 분산 저장 할 수 있다.

1. 서론

시맨틱 웹은 기계수준에서 데이터를 처리하기 쉽게 해줄 수 있는 기술로 W3C에서는 웹의 표준화와 함께 시맨틱 웹 기술을 위해 RDF, RDFs, OWL 등과 같은 온톨로지 언어를 표준화 하고 있다[1]. 시맨틱 웹 기술은 검색엔진과 데이터간의 상호 유기적인 관계를 매핑하는데 적합한 기술이다. 시맨틱 웹 기술을 이용하여 SNS(Social Networking Service)와 같은 서비스 모델에 온톨로지 언어를 사용하여 관계를 매핑 하거나 모바일 GPS정보, 유전정보와 같은 데이터를 효과적으로 다룰 수 있다. 하지만 이와 같이 대량으로 발생되는 정보를 효과적으로 저장하고 추출하는 것은 쉬운 일이 아니다. 따라서 현재 클라우드 컴퓨팅 기술인 분산/병렬 처리를 위한 하둡(Hadoop)과 빅 데이터를 저장할 수 있는 NoSQL형 데이터베이스를 이용하여 대량의 데이터 모델을 다루기 위한 연구가 진행 중에 있다[2].

시맨틱 웹 분야에서는 RDF 수준의 관계형 데이터베이스에 저장이 가능한 트리플 스토어에 관한 연구가 진행되고 있다[3]. 하지만 실시간적으로 대량의 데이터가 발생될 수 있는 모델에 대해 효과적으로 저장하고 분산 관리하기 위한 방법은 시맨틱 웹 분야에서는 현재 연구가 진행되고 있지 않다. 따라서 본 논문에서는 NoSQL형 데이터베이스 중 HBase 데이터베이스에 OWL 온톨로지를 효과적으로 저장할 수 있는 매핑 규칙을 제안한다. 매핑 규칙을 통해 OWL 온톨로지를 HBase 저장소에 저장이 가능한 형태로 변환이 가능하며 이를 통해 시맨틱 웹 환경에서도 대량으로 발생될 수 있는 데이터들을 분산 저장이 가능한 저장소를 갖추게 된다.

본 논문은 4장으로 구성되며 2장에서는 관련연구인 NoSQL과 HBase에 대해 알아보고, 3장에서는 시맨틱웹과 HBase간의 매핑 규칙 및 모델 알아보고 4장에서는 결론을 맺는다.

2. 관련연구

2장에서는 NoSQL과 분산 데이터 저장소인 HBase에 대해 알아본다.

2.1 NoSQL

NoSQL은 “Not Only SQL”의 약어로 데이터를 저장하는데 SQL만 이용해서 할 필요가 없다는 전제하에 나온 개념이다[4]. 즉, 데이터의 종류와 성격에 따라 기존의 관계형 데이터베이스가 필요한 데이터라면 관계형 데이터베이스를 이용하고 그렇지 않다면 무리하게 관계형 데이터베이스에만 데이터를 저장하는 것이 아니라 용도에 맞는 데이터 스토어를 사용하자는 것이다.

관계형 데이터베이스는 오랜 연구 끝에 성능과 데이터의 일관성 면에서 많은 장점을 가지고 있지만 현재와 같은 데이터가 폭증하는 시대에는 약점도 함께 내포하고 있다. 대량의 데이터 처리가 힘들고 테이블 갱신에 필요한 인덱스 생성이나 스키마 변경이 용이하지 않고, 컬럼을 확

장하기 어려운 경우가 있다. 이와 같은 관계형 데이터베이스의 약점을 보완해서 분산된 대량의 데이터 처리에 적합하고, 확장성에 초점을 둔 NoSQL 형 데이터베이스가 현재 발전이 되고 있다[5].

NoSQL형 데이터베이스에도 여러 종류가 있다. Key--Value형 데이터베이스, 문서형 데이터베이스, 컬럼형 데이터베이스와 같은 종류로 각각 발전이 되고 있다[6].

<표 1> NoSQL 데이터베이스 종류

Key-Value형 데이터베이스	문서형 데이터베이스	컬럼형 데이터베이스
DynamoDB	MongoDB	HBase
MemcacheDB	CouchDB	Cassandra
Redis	OrientDB	HyperTable

<표 1>은 각 특징에 따른 NoSQL 데이터베이스의 종류를 나타낸 것이다. 본 논문에서는 컬럼형 데이터베이스 종류중의 하나인 HBase를 이용하여 시맨틱웹 데이터를 분산 저장할 수 있는 방법을 제안한다.

2.2 HBase

HBase는 NoSQL 데이터베이스 중의 컬럼형 데이터베이스에 속하는 데이터베이스이다. 컬럼형 데이터베이스는 컬럼 단위로 데이터를 읽고 쓰는 처리에 뛰어 나다. 관계형 데이터베이스는 특정 조건을 만족시키는 데이터의 검색과 같은 행 단위로 데이터를 처리한다. HBase는 기존의 관계형 데이터베이스와는 다르게 컬럼 단위로 데이터를 처리하기 때문에 모든 행에 대한 특정 컬럼의 일괄 갱신 및 컬럼을 확장하는데 용이한 구조를 가지고 있다.

따라서 대용량 데이터를 다루는데 카산드라(Cassandra)와 함께 현재 가장 많이 사용되고 있는 NoSQL 데이터베이스이다. 카산드라는 성능위주의 데이터베이스로 일관성이 보장 되지 않을 수가 있다[7]. 하지만 HBase는 대량의 데이터 처리의 성능과 일관성을 보장하도록 설계되어 있다. 이는 HBase에는 분산 시스템을 통제하는 마스터를 두고 복제된 전체 데이터의 일관성을 관리하기 때문에 분산된 복제 데이터 사이의 일관성을 보장하며, 제약이 있지만 트랜잭션성 처리도 지원을 한다[8].

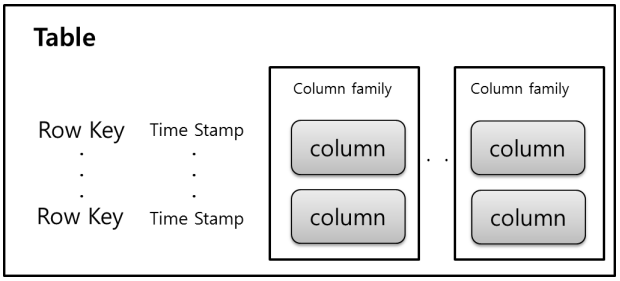
HBase는 대량의 데이터 처리와 분석에 이용되는 하둡(Hadoop)의 서브 프로젝트로 현재 아파치에 등록되어 있다. 따라서 HBase는 하둡의 HDFS(Hadoop Distributed File System) 및 맵리듀스(MapReduce)등과 함께 사용될 수 있도록 최적화 되어 있다. 이러한 분산/병렬 처리가 가능한 프레임워크 안에서 동작 가능한 HBase는 대용량의 데이터를 처리하기에 현재 가장 적합한 NoSQL 기술이다.

3. 시맨틱 웹과 HBase 매핑 모델

3장에서는 시맨틱 웹 환경을 구성하고 있는 OWL의 구조에 대한 설명과 HBase가 가지고 있는 기본적인 데이터 모델에 대해 설명하고, OWL과 HBase를 매핑할 수 있는 규칙에 대해 설명한다.

3.1 OWL과 HBase 데이터 모델

OWL은 W3C에서 온톨로지 구축을 위해 고안된 가장 상위 레벨의 언어이며 OWL 모델을 통해 기존의 RDF, RDFs와 같은 언어의 레벨도 흡수가 가능하다. OWL은 기본적으로 클래스(Class), 프로퍼티(Property), 객체(Individual)로 구성이 된다. 클래스는 공통된 속성을 가진 객체들을 담는 그릇이 되며 프로퍼티는 해당 객체들 사이의 관계를 표시하고 객체는 각 클래스에 속한 데이터이다. OWL 문법에 대한 자세한 설명은 [9]를 참조한다. 본 논문에서는 세 가지 기본적인 속성을 HBase 데이터 모델로 변환하고자 한다.



(그림 1) HBase 데이터 모델

(그림 1)은 HBase의 데이터 모델을 나타낸 그림이다. HBase의 데이터 모델은 행키(Row Key), 타임스탬프(Time Stamp), 컬럼 패밀리(Column family), 컬럼(Column) 형태로 구성이 된다. 행키는 순서에 따라 자동으로 정렬된 형태이며 컬럼 패밀리와 컬럼을 대표하는 키로 작용한다. 그리고 각 컬럼 패밀리는 컬럼들을 대표하는 속성을 나타낸다. 컬럼에는 컬럼에 해당하는 데이터들이 들어가게 된다.

3.2 OWL과 HBase 매핑 규칙

본 논문에서는 OWL의 세 가지 속성인 클래스, 프로퍼티, 객체들을 HBase 데이터 모델형태로 변환하고 변환을 위한 데이터 파일은 JSON[10] 형태로 한다. JSON으로 변환된 파일은 HBase Java API와 REST API를 이용하여 HBase로 저장이 가능하다.

<표 2> OWL과 HBase의 매핑 규칙 테이블

OWL	HBase
클래스	행키
프로퍼티(Object, Data)	컬럼 패밀리
rdf:type	
프로퍼티 이름	컬럼
resource	
rdf:resource	데이터

<표 2>는 OWL과 HBase의 매핑 규칙 테이블을 나타낸 것이다. OWL의 클래스는 HBase의 행키와 매핑을 한다. 이는 공통된 속성이 모여진 OWL 클래스는 HBase의 행키와 같은 역할을 수행할 수 있다. HBase의 컬럼 패밀리에는 두 가지 프로퍼티 속성인 오브젝트형과 데이터 프로퍼티 형을 대표 컬럼 패밀리로 한다. 그리고 rdf:type에 해당하는 객체 식별자는 Individual의 이름으로 다른 컬럼 패밀리로 구성한다. 이와 같은 방식을 통해 컬럼 패밀리는 프로퍼티형 컬럼 패밀리와 객체를 담는 컬럼 패밀리로 나뉘게 된다. OWL에는 프로퍼티의 기본 이름이 있으며 프로퍼티 이름을 통해 해당 리소스의 속성을 이해할 수 있다. 따라서 프로퍼티 이름이 프로퍼티형 컬럼 패밀리에 있는 컬럼명이 되고, Individual의 이름의 컬럼 패밀리에는 resource의 이름으로 해당 객체 데이터를 담는 컬럼으로 명명한다. OWL의 rdf:resource 속성은 데이터를 나타내는 것이므로 HBase의 컬럼안에 있는 데이터로 매핑 시킨다.

```
<owl:Class rdf:ID="Winery"/>

<owl:ObjectProperty rdf:ID="madeFromGrape">
  <rdfs:domain rdf:resource="#Wine"/>
  <rdfs:range rdf:resource="#WineGrape"/>
</owl:ObjectProperty>

<owl:Thing rdf:ID="Winery" />
<owl:Thing rdf:about="#Winery">
   <rdf:type rdf:resource="#Whitewine"/>
</owl:Thing>
```

(그림 2) OWL 데이터

```
Winery: {
  ObjectProperty: {
        madeFromGrape: {
              T1:Wine
              T2:WineGrape
        }
  }
  Individual : {
        resource:{ TI:Whitewine }
  }
}
```

(그림 3) HBase 데이터 모델이 적용된 JSON

(그림 2)는 OWL이 가지고 있는 3가지 속성이 표현된 wine 온톨로지를 나타낸 것이다. wine 온톨로지에는 Winery라는 하나의 클래스가 있고, 오브젝트 프로퍼티형의 madeFromGrape와 Winery 클래스에 속하는 Whitewine 객체로 구성 되어 있다.

(그림 3)은 (그림 2)에 있는 OWL 정보를 JSON 형태로 변환한 것이다. JSON에 대한 문법은 생략한다. (그림 3)에서는 Winery가 행키를 나타내고 2개의 컬럼 패밀리를 가지고 있다. 각각은 ObjectProperty와 Individual이다. 컬럼 패밀리 안에는 madeFromGrape 컬럼에 Wine 데이터와 WineGrape가 있다. T1과 T2는 타임스탬프를 나타낸다. 그리고 Individual 컬럼 패밀리 안에는 resource 컬럼에 Whitewine 객체가 데이터로 매핑된 모습이다. 이와 같이 나타내는 데에는 <표 2>에 있는 매핑 규칙 테이블을 토대로 JSON 형태로 변환을 하였으며 이와 같은 규칙을 통해 필요에 따라 XML형태로도 변환이 가능하다.

JSON 형태로 변환된 파일은 JSON 파일 처리를 통해 HBase에 저장이 될 수 있다. HBase에서는 Java 및 REST API를 함께 제공하여 Java 기반의 어플리케이션 혹은 웹을 통하여 HBase 데이터베이스로 저장이 가능하다.

4. 결론 및 향후 과제

본 논문에서는 대용량의 데이터를 분산 저장이 가능한 HBase 데이터베이스에 시맨틱 웹 정보를 저장할 수 있는 매핑 규칙을 제시하였다. 본 논문에서 제시한 규칙을 바탕으로 OWL 스키마와 인스턴스를 효율적으로 HBase 데이터 모델에 맞게 매핑이 가능하며 매핑된 데이터는 JSON 형태로 나옴으로써 JSON 파일 처리를 통해 HBase 저장소에 저장이 가능하다. 이와 같은 매핑 규칙을 통해 시맨틱 웹 환경에서 발생할 수 있는 대용량의 데이터를 분산 저장할 수 있다.

본 논문에서는 OWL이 가지고 있는 기본적인 요소들에 대해서만 매핑 규칙을 제시하였다. 하지만 OWL에는 많은

제약사항을 나타내는 규칙들이 있으며 향 후 에는 이러한 부분을 커버하기 위한 규칙 모델이 적용된 하나의 프레임 워크 안에서 동작 가능한 형태의 시스템으로 개발하는 것이 연구 과제로 남아 있다.

참고문헌

[1] W3C, “http://www.w3.org/”.

[2] Rick C, "Scalable SQL and NoSQL Data Stores," ACM SIGMOD, Volume 39 Issue 4, pp.12-27, 2010.

[3] Ying Y, Chen W, Aoying Z, Weining Q, Li M, Yue P Dorene R, and John S, “Efficiently querying rdf data in triple stores,” WWW’08, pp.1053-1054, 2008.

[4] Shashank T, “Professional NoSQL”, Wrox Press.

[5] Jaroslav P, “NoSQL Databases: a step to database scalability in Web environment,” iiWAS’11, pp.278-283, 2011.

[6] NoSQL “http://nosql-database.org/”.

[7] Avinash L, Prashant M, “Cassandra - A Decentralized Structured Storage System,” ACM SIGOPS Operating Systems Review, Volume 44 Issue 2, pp.35-40, 2010.

[8] Carstoiu D, Cernian A and Olteanu A, “Hadoop Hbase-0.20.2 Performance Evalution,” NISS, 4th, pp.84-87, 2010.

[9] OWL, “http://www.w3.org/TR/owl-features/”.

[10] JSON, “http://www.json.org/json-ko.html”.