OWL 데이터 검색을 위한 효율적인 저장 스키마 구축 방법[1]

우은미° 박명제 정진완
한국과학기술원 전자전산학과 전산학전공
(emwoo°, jpark, chungcw)@islab.kaist.ac.kr

A Storage Schema Construction Technique For Efficient Retrieval of OWL Data

Eun-Mi Woo° Myung-Jae Park Chin-Wan Chung
Division of Computer Science, Department of Electrical Engineering & Computer Science,
Korea Advanced Institute of Science and Technology

요 약

온톨로지 언어를 기반으로 하는 시맨틱 웹 상에서 온톨로지 데이터에 대한 질의를 효율적으로 처리하기 위해서는 질의 처리에 효과적인 저장 스키마의 구축이 필요하다. 본 논문에서는 온톨로지 데이터의 검색을 위한 효과적인 저장 스키마 구축 기법을 제안하고자 한다. 아울러 본 논문에서 제안하는 저장 스키마를 위한 질의 기법을 제안하고자 한다. 온톨로지 데이터를 추출할 때에는 계층 구조를 기반으로 질의 수행을 해야 한다. 따라서 본 논문에서는 계층 정보를 유지하기 위해 별도의 XML 문서를 기존의 넘버링 기법을 기반으로 생성하여 하위 정보의 추출을 효과적으로 지원하고자 한다. 실험을 통한 질의 처리 성능을 통해서 본 논문에서 제안하는 저장 스키마 기법과 질의 처리 기법이 효과적임을 보인다.

1. 서론

월드 와이드 웹(World Wide Web)은 놀랄만한 속도로 발전을 거듭하고 있다. 현재의 웹은 사용자의 수가 점점 늘어나고 공유하는 정보의 양이 증가함에 따라, 사용자가 원하는 정확한 정보를 제공하지 못하는 문제점이 제기되고 있다. 그 이유로는 현재의 웹이 사람이 보고 이해할 수 있도록 설계되었기 때문이다. 이런 웹의 문제점을 해결하고자 시맨틱 웹(Semantic Web)이 제안되었다. 시맨틱 웹은 웹상의 정보에 관련 의미를 부여해서 사람 뿐 아니라 컴퓨터가 정보의 의미를 분석하는 것을 가능하게 한다.[1] 이런 시맨틱 웹을 구축하기 위해서는 웹상에 존재하는 정보들의 의미를 형식적으로 기술할 수 있는 온톨로지 언어를 사용해야 한다. 대표적인 온톨로지 언어로는 W3C의 권고안인 OWL[2]이 있으며 시맨틱 웹상에서 정보를 검색하기 위해서는 OWL 데이터를 효율적으로 저장하고 검색하는 방법이 필요하다.

OWL에 대한 연구는 현재 진행 중에 있거나 시작 단계에 있다. OWL 이전에 제안된 DAML+OIL[3]을 지원하는 시스템으로는 DLDB[4], 그리고 RDF[5] 저장, 검색 시스템인 Sesame[6]을 확장시킨 BOR[7]가 대표적이다. OWL이 이전 온톨로지 언어에 비해 달라진 점은 정보 표현 범위가 넓어지고 추론 기능이 강화되었다는 것이다. 시맨틱 웹 상에서 OWL의 특성을 잘 이용하기 위해서는 OWL 데이터를 위한 시스템이 요구되어 진다.

따라서 본 논문은 시맨틱 웹을 구축하기 위한 온톨로지 언어인 OWL 데이터를 저장, 검색하는 기법을 제안하고자 한다. 특히 OWL 데이터를 위한 저장 스키마를 설계함에 있어서 효율적인 질의 처리를 위한 저장 스키마를 제안하고자 한다. 그리고 설계된 저장 스키마에 적절한 질의 처리 기법 또한 제안하고자 한다. 또한 클래스(class)나 프로퍼티(property)간의 계층 정보를 추출하는 데 있어서 기존의 넘버링(numbering) 기법을 사용하여 효과적으로 하위 구조를 고려할 수 있도록 한다. 마지막으로 실험 결과를 통하여 기존의 시스템에 비해 제안된 기법들이 효율적임을 보인다.

2. 관련 연구

DLDB[4]는 Lehigh University의 Semantic Web & Agent Technologies Lab에서 개발한 온톨로지 데이터 저장, 검색 시스템으로 특별히 DAML+OIL을 지원하며 RDBMS를 사용한다. 온톨로지 데이터를 저장하기 위해서는 문서 상에 선언된 클래스나 프로퍼티 각각에 대해서 테이블을 생성하며 클래스와 프로퍼티의 계층 정보를 유지하기 위해서는 별도의 계층 테이블을 생성해서 부모/자식(super/sub) 관계의 정보를 저장한다.

현재 본 연구실에서 개발중인 시스템인 OWL 저장, 검색 시스템[8]은 OWL을 지원하며 RDBMS와 XML[9] 저장소를 함께 사용한다. 클래스의 인스턴스(instance)에 관한 정보들은 모두 하나의 인스턴스 테이블에 저장된다. 계층 정보는 별도로 추출하여 XML 문서로 만들어서 XML 저장소에 저장한다. 질의 시 계층 정보 추출을 위해서는 XPath[10] 질의를 사용하여 해당 결과를 추출하고, 사용자가 원하는 온톨로지 데이터의 추출을 위해서는 추출한 계층 정보와 함께 SQL 질의를 만들어 RDBMS를 통해 수행하여 최종 결과를 얻게 된다.

본 연구의 전체적인 접근 방법은 관련 연구의 OWL 시스템과 유사하다. 하지만 기존 시스템의 성능보다 향상된 성능을 위한 효율적인 저장 스키마, 계층 정보 관리 기법, 그리고 검색 기법을 제안한 데에 차이를 두고 있으며 그 차이를 중심으로 본 논문의 저장, 질의 기법을 설명하고자 한다.

3. OWL 데이터의 저장 기법

그림 1은 본 논문이 기반한 OWL 저장 처리기[8]의 전체 구조를 보여준다. OWL 데이터의 저장을 위해 RDBMS와 XML 저장소를 함께 사용한다. OWL 문서가 파서를 통해 파싱되면 클래스와 프로퍼

[1] 본 연구는 정보통신부의 대학 IT연구센터(ITRC) 지원을 받아 수행되었습니다.

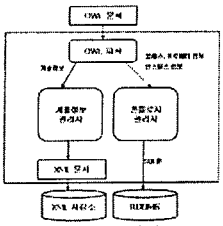


그림 1. 저장처리기 구조

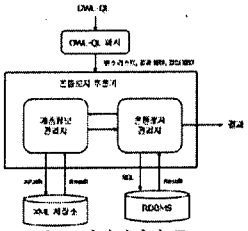


그림 2. 질의처리기 구조

티 그리고 인스턴스 정보는 RDBMS에 저장하며 클래스와 프로퍼티에 관한 계층 정보는 XML 저장소에 저장한다.

3.1 계층 정보의 저장

OWL 문서는 계층 구조를 이루고 있으므로 온톨로지 데이터의 효율적인 검색을 위해서는 클래스와 프로퍼티의 계층 정보를 유지하는 것이 요구된다. XML 문서는 XPath 나 XQuery[11]등의 XML 질의 언어를 통해서 특정 엘리먼트의 하위 구조를 손쉽게 추출해 낼 수 있다는 장점을 가진다. 따라서 본 연구에서는 OWL 문서 내의 클래스와 프로퍼티의 계층 정보를 포함하는 XML 문서를 생성하여 사용함으로써 질의 처리시에 계층 정보의 추출을 용이하게 한다. 이때 Dietz' s Numbering 기법[12]을 사용하여 기존의 시스템에서 구성하는 XML 문서에 비해 계층 정보 추출시의 효율성을 높였다. 이 넘버링 기법의 특징은 각 엘리먼트의 전위(preorder)와 후위(postorder) 정보를 기반으로 조상/자손(ancestor/descendant)간의 관계를 쉽게 추출할 수 있다는 것이다. 특정 엘리먼트의 자손 엘리먼트들은 특정 엘리먼트의 전위보다 크거나 같고 후위보다 작거나 같다는 특성을 가지고 조상/자손간의 관계를 효율적으로 알 수 있다.

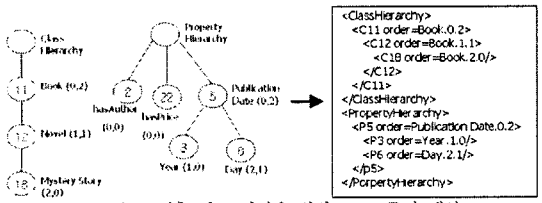


그림 3. 계층 정보 저장을 위한 XML 문서 생성

그림 3은 XML 트리로 표현된 클래스와 프로퍼티의 계층 구조를 기반으로 XML 문서를 생성한 예를 보여준다. XML 트리에서 원 안의 숫자는 클래스나 프로퍼티가 가지는 고유한 식별자이며 각 노드는 클래스나 프로퍼티의 이름과 함께 (preorder, postorder)값을 가진다. XML 문서를 구성 시 각 엘리먼트의 애트리뷰트에는 전위, 후위의 값과 최상위 클래스 및 프로퍼티의 이름이 포함된다.

3.2 저장 스키마

질의 시 추출해야 하는 온톨로지 데이터들을 저장하기 위해서 각각의 최상위 클래스/프로퍼티에 대한 테이블을 각각 생성한다. 그리고 하위 클래스/프로퍼티의 정보들은 해당 최상위 클래스/프로퍼티에 대해 생성된 테이블에 저장된다. 그림 3의 경우 Book_TABLE, hasAuthor_TABLE, hasPrice_TABLE, PublicationDate_TABLE이 생성되며 Novel과 Mystery Story 클래스의 경우는 Book_TABLE에, Year와 Day 프로퍼티는 PublicationDate_TABLE에 저장된다.

4. OWL-QL 을 이용한 질의 처리 기법

그림 2는 질의 처리기의 구조도를 나타낸다. 사용자가 입력한 OWL-QL[13] 질의는 파싱을 거쳐 검색해야 할 온톨로지에 관한 정보를 얻는다. 질의에 사용된 클래스나 프로퍼티의 하위 정보들은 XPath 질의를 통해서 얻고 사용자가 원하는 온톨로지 정보를 추출하기 위해서는 추출한 계층 정보와 함께 SQL문을 만들어 질의를 수행한다.

4.1 SQL질의로의 변환

4.1.1 OWL-QL 질의의 패턴 분류

OWL-QL의 질의 패턴(Query Pattern)은 크게 4가지로 나뉜다. 그림 4는 각 특징에 따라 분류한 OWL-QL 질의의 패턴을 보여준다.

패턴1: (type ?x Book)
패턴2: (hasAuthor ?x DanBrown)
패턴3: (hasAuthor TheDaVinciCode ?y)
패턴4: (hasAuthor ?x ?y)

그림 4. OWL-QL 질의의 4가지 패턴

사용자가 얻고자 하는 결과가 클래스의 인스턴스인지, 프로퍼티의 값인지 또는 두 가지 모두인지에 따라 4가지 패턴으로 나뉜다.

(type ?b Book)(hasAuthor ?b DanBrown)(hasPrice ?b ?p)(Year ?b ?y)
패턴 1 | 패턴 2 | 패턴 4 | 패턴 4

그림 5. OWL-QL 질의 예제

그림 5는 OWL-QL 질의의 예이다. 각각의 질의 패턴에는 해당 패턴번호를 표기해 놓았다. 그림 5의 질의는 Dan Brown의 책(Book), 가격(hasPrice), 출판년도(Year)에 관한 질의이다.

4.1.2 계층 정보의 추출

질의에 사용된 클래스와 프로퍼티 각각에 대해서 XPath 질의를 만들고 XML 저장소로부터 계층 정보를 얻는다.

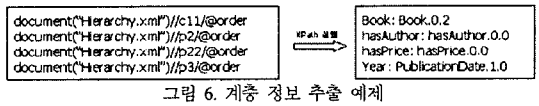


그림 6. 계층 정보 추출 예제

그림 6은 그림 5의 질의에 사용된 클래스와 프로퍼티에 대한 XPath 질의와 그 결과이다. Book 클래스의 경우 최상위 클래스가 Book이고 전위가 0, 후위가 2임을 알 수 있다.

4.1.3 타입에 따른 SQL 변환

하나의 질의 패턴은 하나의 SQL문으로 변환된다. SQL문으로 변환 시 각 패턴에 따라 변형된 SQL 변환 기법이 적용된다.

패턴 1,2,3은 하나의 변수를 가지고 있는 경우이다. 만약 동일한 변수를 가진 질의 패턴이 있다면 각각 SQL문으로 변환 후 중간에 INTERSECT 연산자를 적용한다. 그림 5의 질의에서 첫 번째와 두 번째의 질의 패턴은 동일한 변수 w를 가진다. 따라서 두 질의 패턴에 대한 SQL문 사이에는 INTERSECT 연산자를 적용한다. INTERSECT 연산자를 통해 불필요한 중간 튜플들을 제거함으로써 마지막 단계에 조인에 포함하는 튜플의 수를 줄인다.

패턴 4는 두 개의 변수를 가지고 있는 경우이다. 이때 두 개의 변수 중 적어도 하나의 변수에는 패턴 1,2,3에서 사용된 변수이다. 그림 5의 질의에서 세 번째, 네 번째 질의 패턴의 w 변수는 이미 앞서 사용된 변수이므로 이미 변수에 대한 중간 결과를 가지고 있다. 따라서 패턴 4의 질의 패턴에 대한 SQL문 변환 시에는 중간 결과를

얻은 변수의 값과 매칭이 되는 것만을 결과로 얻도록 한다.

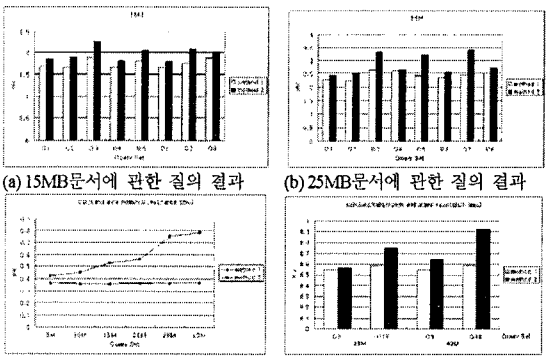
각 SQL문에 대한 결과 추출은 XPath 질의의 결과로 얻은 최상위 클래스/프로퍼티의 테이블로부터 이루어진다. 그리고 클래스나 프로퍼티의 하위 구조까지 고려해서 결과를 추출하기 위해서 각 SQL문에는 전위와 후위의 범위를 조건으로 준다.

5. 성능평가

5.1 실험 환경

실험은 CPU가 2.56GHZ이고 512MB의 메모리를 가지는 펜티엄 4급의 컴퓨터를 사용하였다. 실험에 사용된 OWL 문서는 Univ-Bench Artificial data generator(UBA)에 의해 생성된 것이며 OWL 데이터에 대한 질의는 DAML+OIL에 관한 벤치마킹 논문[14]을 참조하였다. 질의 1과 2, 4, 6은 계층 구조 유무에 따른 하나의 클래스/프로퍼티에 대한 질의를, 질의 3, 7은 하나의 클래스와 여러 개의 프로퍼티에 관한 것이다. 질의 5, 8은 여러 개의 클래스/프로퍼티에 관한 질의를, 질의 11은 하나의 클래스의 인스턴스 추출에 관한 질의이다. 마지막으로 질의 9는 계층 구조가 없는 경우, 질의 10은 많은 레벨의 계층 구조를 가진 경우의 질의이다. 본 연구(method1)는 관련 연구에서 소개한 OWL 저장, 검색 시스템(method2)[8]과 성능을 비교하였다.

5.1 실험결과



(a) 15MB문서에 관한 질의 결과 (b) 25MB문서에 관한 질의 결과

(c) 하나의 클래스에 대한 질의 (d) 계층 정보 추출 성능

그림 7. 질의 처리 성능

그림 7은 각각의 질의에 대한 성능을 그래프로 표기한 것이다. (a), (b)는 15MB와 25MB의 OWL 문서에 대해서 질의 1~8까지를 수행시킨 결과이다. 그래프를 살펴보면 모든 질의에 대해서 전반적으로 본 연구의 시스템의 성능이 좋으며 크게는 약 1.3배의 성능 차이를 보인다. 그 이유는 비교 시스템의 경우 하나의 테이블에 모든 클래스의 인스턴스 정보를 저장하므로 검색해야 하는 튜플의 수가 많지만, 본 연구의 경우 최상위 클래스와 프로퍼티마다 테이블을 생성하므로 비교 시스템에 비해 검색해야 하는 튜플의 수가 적다.

질의 3, 5, 7에 대한 성능은 특히 좋다. 이 질의들은 여러 개의 클래스나 프로퍼티를 가지는 경우로 본 연구는 중간 튜플의 수를 줄이고자 패턴별 다른 SQL 변환 기법을 적용하였기 때문에 위의 특정 질의에 대해서 더 좋은 성능을 얻는다.

(c)에서 하나의 클래스의 인스턴스들을 추출하는 경우 본 연구의 시스템은 문서 크기에 영향을 많이 받지 않는 반면, 비교 시스템은 문서 사이즈가 증가함에 따라 하나의 테이블에 저장되는 데이터의 양이 점점 증가함으로 성능에 크게 영향을 받는다.

(d)의 경우, 많은 계층 구조를 가지는 질의 10에서 더 큰 성능 차이가 나는 것으로 보아 본 연구에서 제안한 계층 정보 추출 기법이 더 효율적임을 알 수 있다. 비교 시스템의 경우 XML 문서에서 추출한 하위 클래스나 프로퍼티 정보들을 SQL문 생성시 WHERE절의 조건으로 넣어서 튜플들의 애트리뷰트의 값이 그 정보들과 매칭이 되는지 일일이 확인하는 과정을 거친다. 하지만 본 시스템의 경우는 전위와 후위 범위 사이의 튜플들만을 추출하도록 하였으므로 간단한 하위 구조의 검색이 가능하다.

6. 결론

본 연구에서는 시맨틱 웹 상에서의 효율적인 OWL 질의 처리를 위한 저장 스키마의 구축과 질의 기법을 제안하였다. 최상위 클래스와 프로퍼티에 따라 테이블을 분리하여 하위 정보들은 해당하는 최상위 클래스나 프로퍼티에 관한 테이블에 저장하도록 하였다. 그리고 기존의 넘버링 기법을 이용하여 전위와 후위의 범위를 가지고 간단하게 계층 정보를 추출하였다. 또한 질의 시에는 사전에 불필요한 중간 튜플의 수를 줄여서 최종적으로 이루어지는 조인의 수를 줄였다. 실험 결과, 모든 질의에 대해서 전반적으로 본 시스템의 성능이 좋으며 크게는 약 1.3배의 성능 차이가 나는 것을 알 수 있다. 특히 여러 개의 클래스나 프로퍼티를 가지는 경우에 더 좋은 성능을 얻을 수 있었다. 또한 계층 구조를 많이 가지는 경우의 질의에 대해 더 효율적임을 알 수 있었다.

7. 참고문헌

[1] Tim Berners-Lee, James Hendler, and Ora Lassila, The Semantic Web, Scientific American, May 2001

[2] Michael K. Smith, Chris Welty, and Deborah L. McGuinness, OWL Web Ontology Language Guide, W3C Proposed Recommendation, http://www.w3c.org/ TR/ 2003/PR-owl-guide-20031215, 2003

[3] Deborah L. McGuinness, Rechard Fikes, and James Hendler, DAML+OIL: An Ontology Language for the Semantic Web, IEEE INTERNET SYSTEMS, pp. 72-80, September/October 2002

[4] Zhengziang Pau, and Jeff Heflin, DLDB: Extending Relational Databases to Support Semantic Web Queries, In Workshop on Practical and Scaleable Semantic Web System, ISWC, pp. 109-113, 2003

[5] Frank Manola, and Eric Miller, RDF Primer, W3C Recommendation, http://www.w3c.org/TR/rdf-primer, 2004

[6] Jeen Broekstra, and Arjohn Kampman, Sesame: An Architecture for Storing and Querying RDF Data and Schema Information, Proceedings of the first International Semantic Web Conference(ISWC), 2002

[7] Kiril Simov, and Stanislav Jordanov, BOR: a Pragmatic DAML+OIL Reasoner, IST Project IST-1999-10132 On-To-Knowledge

[8] Chun-Hee Lee, Jun-Ki Min, Myung-Jae Park, Ji-Hyun Lee, Eun-Mi Woo, and Chin-Wan Chung, A Storage and Query Processing Technique For Ontology Data, Technical Report CS/TR-2004-211, Department of Computer Science, KAIST, December 23, 2004

[9] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, and Eve Maler, Extensible Markup Language(XML) 1.0, W3C Recommendation, http://www.w3c.org/ TR/REC-xml, 2004

[10] James Clark, and Steve DeRose, XML Path Language(XPath) Versioin 1.0, W3C Recommendation, http://www.w3c.org/TR/Xpath, 1999

[11] Scott Boag, Don Chamberlin, Mary F. Fernandez, Daniela Florescu, Jonathan Robie, and Jerome Simeon, XQuery 1.0: An XML Query Language, Working Draft, http://www.w3c.org/TR/2002/WD-query-20020816

[12] Paul F. Dietz, Maintaining order in a linked list, Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing, pages 122-127, San Francisco California, May 1982

[13] Richard Fkies, Patrick Hayes, and Horrocks, OWL-QL – A Language for Deductive Query Answering on the Semantic Web

[14] Yuanbo Guo, Jeff Heflin, and Zhengxiang Pan, Benchmarking DAML+OIL Repositories, The Semantic Web – ISWC 2003, LNCS 2870. Springer, 2003, pages 613-627