한국시뮬레이션학회 논문지
Vol. 17, No. 2, pp. 31-47 (2008. 6)

계층 구조를 고려한 Jena Plug-in 저장소의 평가를 위한 실험 및 시뮬레이션

신희영[1] · 정동원[2†] · 백두권[1†]

Experiment and Simulation for Evaluation of Jena Storage Plug-in Considering Hierarchical Structure

Heeyoung Shin · Dongwon Jeong · Doo-Kwon Baik

ABSTRACT

As OWL (Web Ontology Language) has been selected as a standard ontology description language by W3C, many ontologies have been building and developing in OWL. The Jena developed by HP as an Application Programming Interface (API) provides various APIs to develop inference engines as well as storages, and it is widely used for system development. However, the storage model of Jena2 stores most owl documents not acceptable into a single table and it shows low processing performance for a large ontology data set. Most of all, Jena2 storage model does not consider hierarchical structures of classes and properties. In addition, it shows low query processing performance using the hierarchical structure because of many join operations. To solve these issues, this paper proposes an OWL ontology relational database model. The proposed model semantically classifies and stores information such as classes, properties, and instances. It improves the query processing performance by managing hierarchical information in a separate table. This paper also describes the implementation and evaluation results. This paper also shows the experiment and evaluation result and the comparative analysis on both results. The experiment and evaluation show our proposal provides a prominent performance as against Jena2.

Key words : Ontology, Jena, Plug-in, OWL, Relational database, Ontology storage, Hierarchical structure

요 약

W3C에서 표준 온톨로지 서술 언어로 OWL을 채택함에 따라 많은 온톨로지들이 OWL로 기술 및 구현되고 있다. 이와 관련된 기술 중 Jena는 HP에서 개발한 API로서 저장소는 물론 추론 엔진을 개발할 수 있는 다양한 API를 제공하고 있으며 현재 많은 시스템 개발에 이용되고 있다. 그러나 Jena2의 저장 모델은 단일 테이블에 문서의 정보를 저장하기 때문에 대용량의 온톨로지 데이터 처리에 있어 성능이 저하되는 문제점을 지닌다. 무엇보다 클래스와 프로퍼티의 계층적 구조를 고려하지 않기 때문에 계층 구조를 이용한 질의 처리 시 잦은 조인 연산으로 인해 성능이 급격하게 저하된다. 따라서 본 논문에서는 이러한 문제점들을 해결하기 위해 기존의 Jena2 API를 그대로 이용하면서 Plug-in 형식으로 적용할 수 있는 새로운 OWL 온톨로지 관계형 데이터베이스 모델을 제안한다. 제안 모델은 클래스(Class), 프로퍼티(Property), 인스턴스(Instance)의 정보들을 의미적으로 분류하여 저장하며 계층적 정보들에 대해서도 개별적으로 관리함으로써 질의 처리 성능을 향상시킨다. 또한 기존모델과 이 논문에서 제안하는 모델과의 실험 및 시뮬레이션을 통해 비교 분석 한다. 실험 및 시뮬레이션 결과에서, 제안 시스템이 Jena2보다 나은 성능을 보였다.

주요어 : 온톨로지, Jena, 플러그인, OWL, 관계형 데이터베이스, 온톨로지 저장소, 계층 구조

* 이 연구에 참여한 연구자는 '2단계 BK21 사업'의 지원을 받았으며 또한 이 논문은 "2006년 정부(교육인 적자원부)의 재원으로 한국 학술진흥재단의 지원을 받아 수행된 연구임" (KRF-2006-311-D00776).

2008년 3월 17일 접수, 2008년 5월 24일 채택

[1] 고려대학교 컴퓨터학과
[2] 국립군산대학교 정보통계학과
주 저 자 : 신희영
교신저자 : 백두권, 정동원
E-mail; shintul@software.korea.ac.kr, djeong@kunsan.ac.kr, baik@software.korea.ac.kr

1. 서 론

최근 시맨틱 웹(Semantic Web)에 대한 관심이 증대되면서 W3C(World Wide Web Consortium) 표준으로 개발된 웹 온톨로지 서술 언어(RDF, RDFS, OWL 등) 및 관련 기술에 대한 연구가 활발히 진행되고 있다. 시맨틱 웹이란 현재의 인터넷과 같은 분산환경에서 리소스(웹 문서, 각종 파일, 서비스 등)에 대한 정보와 자원 사이의 관계-의미 정보(Semantics)를 기계(컴퓨터)가 처리할 수 있는 온톨로지 형태로 표현하고, 이를 자동화된 기계(컴퓨터)가 처리하도록 하는 프레임워크이자 기술이라 정의할 수 있다[1]. 시맨틱 웹이 실현되면 검색어 포함 유무에 의한 검색결과가 아닌 정보의 '의미(개념)'을 이용한 검색이 가능해져 사용자가 원하는 정보만을 제공해 줄 수 있다. 온톨로지는 시맨틱 웹을 구현하기 위한 가장 중요한 요소로서, 온톨로지는 자원(Resources)들을 정의하고 자원들이 어떻게 연관되어 있는지를 표현한다. 지금까지 온톨로지를 기반으로 한 많은 연구가 다양한 분야에서 진행되어 왔다[2-7]. W3C에서는 온톨로지 표현과 교환의 용이성을 위해 온톨로지 서술 언어를 개발하였으며, 온톨로지 서술을 위한 대표적인 언어로는 RDF(Resource Description Framework)[8], RDFS(Resource Description Framework Schema)[9], DAML+OIL[10], OWL(Web Ontology Language)[11]등이 있다. 특히 W3C는 2001년웹 온톨로지 워킹그룹을 구성하여 OWL에 대한 표준화 연구를 진행해 왔으며 2004년에 OWL을 W3C 표준으로 제정하였다. OWL은 RDFS와 DAML+OIL의 언어로서 보다 다양하고 세밀하게 자원들의 개념 및 관계를 서술할 수 있고, 추론을 지원함으로써 강력한 표현력을 발휘한다. W3C가 OWL을 표준 온톨로지 언어로 지정함에 따라 추후 대부분의 온톨로지 데이터들은 OWL로 서술될 전망이다[12]. 이로 인해, 최근 RDF, RDFS, OWL, SPARQL[13] 등의 언어를 위한 프로그램 환경에서 규칙 기반 추론 엔진을 포함하고 있는 자바 프레임워크인 Jena[12]에 대한 관심이 높아지고 있다.

Jena는 HP 시맨틱 웹 연구소의 Brian Bride에 의해 개발된 시맨틱 웹 프레임워크이다. RDF, RDFS 및 OWL을 위한 프로그래밍 환경과 기본적인 RDF 파서를 제공하며 내부적으로 규칙 기반의 추론엔진을 포함하고 있다. Jena2는 저장 모델 자동 생성 시 물리적 스키마 구조가 비교적 단순하여, 이로 인해 많은 OWL 관련 시스템 개발에 이용되고 있다. 하지만 비정규화된 단일구조로서 단일테이블에 정보를 저장함에 따라 대용량의 웹 온톨로지에 대한 저장 및 질의 시 성능이 저하되는 문제점을 지닌다[5,14].

본 논문에서는 단순 선택 연산(Simple Selection)은 물론 조인 연산(Join Selection)이 요구되는 질의에 대해서 질의 응답의 성능이 저하되는 Jena2 저장모델의 문제를 해결하기 위한 새로운 온톨로지 관계형 데이터베이스 모델을 제안한다. 제안하고자 하는 저장 모델은 기본적인 Jena2 프레임워크를 그대로 사용할 수 있도록 Jena2의 Plug-in 형식으로 제공되고 보다 효율적으로 저장 및 질의 처리를 제공한다. 특히 계층 구조를 관리함으로써 계층 구조를 고려해야 하는 질의 처리에 보다 나은 성능을 보인다. 계층 구조를 고려한 저장소로는 대표적으로 Sesame를 들 수 있다[21]. 그러나 Sesame는 이미 기존의 여러 연구[17-20]에서 Jena2에 비해 매우 낮은 성능을 확인 할 수 있다. 또한 본 논문의 목적은 Jena2 저장소의 성능을 향상시키는 것이다. 이러한 이유로 Sesame를 비교 대상에서 배제하고 Jena2와 제안 저장 모델과의 실험 결과로 비교평가를 수행한다. 또한 추가적으로 계층적 구성이 갖는 성능향상 정도를 수학적으로 분석하기 위해 각 질의 패턴을 관계대수로 표현하고 이를 계산 모델화 하여 시뮬레이션 하고 실제 실험과 비교평가 하였다. 실제 실험에 대한 평가 항목은 질의 별 결과 개수, 완전성 및 정확성 등으로 이를 통해 제안하는 저장 모델이 Jena2 저장 모델보다 우수함을 보인다.

본 논문의 구성은 다음과 같다. 제 2장에서는 Jena2 프레임워크와 저장 모델에 대해서 구체적으로 서술한다. 제 3장에서는 제안하는 저장 모델에 대해서 자세히 기술한다. 제 4장에서는 Jena2의 저장 모델과 제안하는 저장 모델간의 실제 실험과 시뮬레이션을 통한 성능 비교 및 평가에 대하여 기술한다. 제 5장에서는 결론 및 향후 연구 과제에 대하여 기술한다.

2. 관련 연구

이 장에서는 시맨틱 웹 프레임워크인 Jena2의 기본적인 구조와 Jena2 API의 RDB 기반 저장 모델에 대해서 기술한다.

2.1 Jena2 구조

그림 1은 Jena2의 전체적인 구조이다. Jena2는 자바로 구현되어 자바언어가 갖는 모든 이점을 가지고 있으며, 기본적인 RDF 파서도 제공한다. 내부적인 추론은 그래프 매칭 방법을 이용하여 접근한다. Jena2 프레임 워크의 가장 주요한 부분은 RDF API이다. 이 외에 추론 관련 API,

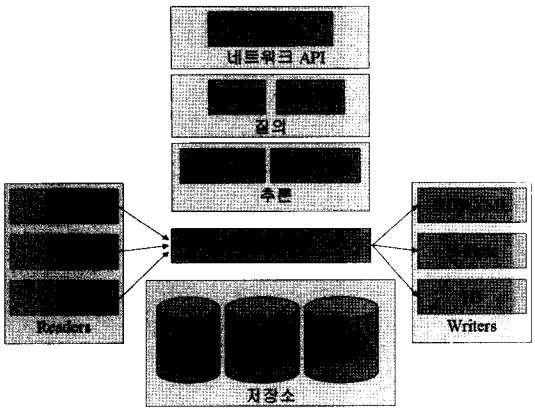


그림 1. Jena2 구조

질의 언어, 네트워크 API 및 저장소 등으로 구성되어 있다. RDF API는 RDF 그래프에 대한 생성, 처리, 그리고 질의를 지원하고 메인 메모리와 관계형 데이터베이스와 버클리 데이터베이스 같은 여러 저장 기술들을 지원한다.

그림 1에서 RDF API를 통하여 Readers와 Writers는 RDF를 읽고 RDF/XML, N3, N-트리플(Triples)로 저장할 수 있다. ARP는 Jena2에 포함되어 있는 RDF/XML 파서이며, RDF/XML 문법으로부터 생성된다. 그림 1에서 RDF API 상위에는 추론, 질의, 네트워크 API 등 3개의 계층이 존재한다. 추론계층은 RDF 그래프에 정보를 추가하여 사용된다. 예를 들어 RDF-S와 DAML은 기존에 존재하는 RDF 그래프에 정보를 추가하여 하위 클래스의 관계로 구현되어 있다. 질의계층은 RDF 그래프를 위해 개발된 질의 언어인 RDQL(RDF Data Query Language)[15]과 다른 질의 언어 등으로 구성된다.

시맨틱 질의 언어의 핵심은 RDF 트리플에 대한 패턴을 기술함으로써 원하는 트리플 정보를 검색할 수 있다. 따라서 기존의 인덱스는 단일 트리플을 효율적으로 검색하는데 초점을 두고 있다. 네트워크 API는 원격 RDF 데이터베이스에 대한 접근(질의, 수정 연산 등)을 가능하게 한다.

2.2 Jena2 저장소 구조

Jena2는 관계형 데이터베이스 기반의 웹 온톨로지 저장 모델을 제공한다. 그림 2는 Jena2 관계형 데이터베이스 모델을 관계형 데이터 모델로 표현하였다. Jena2 프레임워크를 통해서 OWL 온톨로지 데이터 집합을 저장할 때 기본적으로 jena_long_lit, jena_gntn_stmt, jena_long_uri, jena_gntn_reif, jena_prefix, jena_graph, jena_sys_

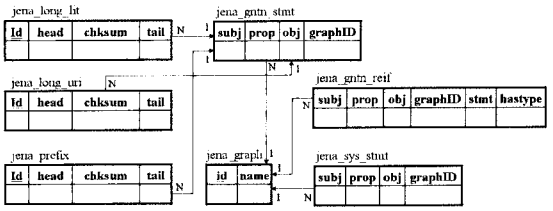


그림 2. Jena2 관계형 데이터 모델

그림 3. Jena2 관계형 데이터베이스 실제 저장 데이터

stmt 등 총 7개의 테이블이 생성된다. 먼저 OWL 파일의 데이터들은 Subject-Property-Object의 컬럼으로 구성된 하나의 테이블에 분류되어 jena_gntn_stmt의 단일 테이블에 저장된다. 리터럴 정보나 URI 정보의 크기가 256 바이트 이상일 경우, 리터럴과 URI 정보는 jena_gntn_stmt 테이블이 아닌 jena_long_lit와 jena_long_uri 테이블에 각각 저장된다. 공통적으로 사용하는 접두사에 대해서는 jena_prefix 테이블에 저장되며 jena_gntn_stmt에서 이를 참조하도록 구성된다.

이와 같은 Jena2 저장 모델 구조는 단순 정보 검색을 위한 단순 질의연산은 물론 조인 연산이 요구되는 질의시 성능 저하를 초래한다. 이는 웹 온톨로지를 구성하는 클래스(Class), 프로퍼티(Property), 인스턴스(Instance) 등의 세부 정보들을 각각의 의미 특징과는 무관하게 트리플(Subject-Property-Object)의 형태로 단순하게 저장되기 때문이다. 이러한 구조는 검색 및 수정과 같은 연산을 위해 막대한 비용을 발생시킨다. 뿐만 아니라 계층 구조적 의미를 파악하기 위해서 처리되는 과정에서도 위와 같은 이유로 단일 테이블 전체를 조인해야 하기 때문에 높은 비용을 요구한다. 더욱이 웹 온톨로지의 용량이 커질수록 처리 비용 역시 증가한다.

2.3 Jena2 저장소 문제점

이 장에서는 Jena2 관계형 데이터 모델에 대한 문제점에 대해서 예를 통해서 구체적으로 기술한다. 그림 3은

OWL 온톨로지 데이터 집합을 Jena2 저장소 모델을 통해 관계형 데이터베이스에 실제 저장된 테이블의 결과이다. 예를 들어, 클래스 정보인 Uv::http://www.Department0.University0.edu/Lecturer3을 검색할 경우 테이블 전체를 검색한다. 이 테이블에는 26개의 클래스 정보뿐 아니라 50000개 이상의 인스턴스와 프로퍼티의 정보들이 한꺼번에 모두 저장되어있기 때문에 1개의 클래스를 검색하기 위해서 50000번 이상의 조회과정이 이루어진다. 클래스 정보를 별도의 테이블로 관리한다면 불필요한 비교연산을 줄임으로써 보다 나은 처리 성능을 제공할 수 있다. 특히 질의를 처리하는 과정에서 조인 연산을 사용해야 하는 경우 클래스, 프로퍼티, 인스턴스 등 50000개 이상의 세부 정보의미와 특징과는 무관하게 트리플 구조의 단일테이블 전체를 조인해야 한다. 이로 인해 성능이 급격하게 저하되는 문제점을 지닌다.

3. JeSPi

이 장에서는 Jena2 프레임워크의 문제점을 보완하기 위해 제안된 저장 시스템인 JeSPi(Jena Storage Plug-in for Enhanced Query Processing)에 대해 기술한다. Jena2의 데이터베이스 저장 모델은 단순한 형태의 저장구조를 지니며 계층 구조를 고려하지 않기 때문에 단순 질의는 물론 조인 연산이 요구되는 경우 높은 처리 비용이 요구된다. OWL, RDF, RDF-S등의 웹 온톨로지는 Class와 Property등의 다양한 계층적 구조를 가지고 있다. 이는 사용자가 상위개념을 통해 하위의 다양한 검색을 지원하고, 세부적인 정보까지 검색 결과로 얻을 수 있는 장점을 지니고 있다. 본 논문에서 제안하는 JeSPi는 Jena2의 다양한 API를 그대로 활용할 수 있는 Jena2의 Plug-in 형태로 개발되었으며 앞서 언급한 문제점을 해결할 수 있는 저장 모델이다. 특히 클래스와 프로퍼티 정보들의 계층 구조를 고려하여 저장할 수 있도록 지원함으로써 기존의 Jena2 프레임워크에서 계층 구조에 대한 처리에서 발생할 수 있는 처리비용을 줄일 수 있다. 이 장에서는 JeSPi를 구성하고 있는 개선된 OWL 온톨로지 관계형 데이터베이스 모델에 대하여 서술하고 JeSPi의 핵심 컴포넌트인 어댑터와 컨버터의 구조 및 기능에 대해서 구체적으로 기술한다.

3.1 JeSPi 프레임워크

그림 4는 기존의 Jena2 API를 그대로 이용하면서 Plug-in형식으로 제공하는 JeSPi의 프레임워크를 보여준다. 컴포넌트는 Jena2 API를 이용하여 OWL 문서가 제

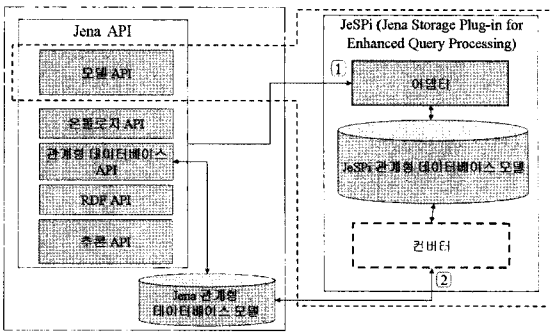


그림 4. JeSPi 프레임워크

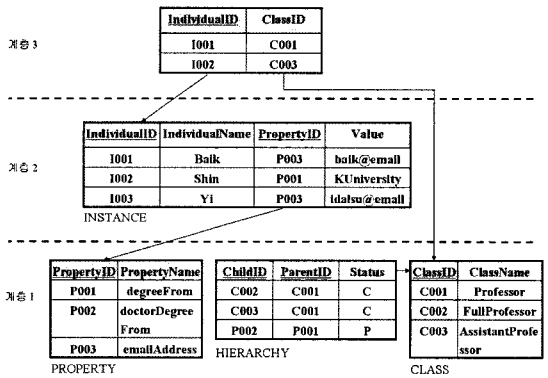


그림 5. JeSPi 관계형 데이터 모델

안하는 어댑터를 거쳐 새로운 개념의 JeSPi 관계형 데이터베이스 모델 형태로 저장된다. 이 컴포넌트는 기존의 Jena2 API를 이용하여 제안하는 개념을 Plug-In 형식으로 적용함으로써 JeSPi 관계형 데이터베이스 모델을 생성 및 변환할 수 있다. 본 논문에서는 첫 번째 컴포넌트인 어댑터에 대해 보다 큰 비중을 두고 이 저장 모델을 구현 후 비교평가를 수행한다.

결과적으로, JeSPi는 새롭게 저장되는 OWL 문서와 기존에 이미 저장되었던 OWL 문서의 관계형 정보들을 보다 검색 성능이 향상된 JeSPi 관계형 데이터베이스 모델로 수용할 수 있는 장점을 지닌다.

3.2 JeSPi 관계형 데이터베이스 모델 구조

제안하는 JeSPi 관계형 데이터베이스 모델은 기존의 Jena2 프레임워크에서 단일테이블로 생성되었던 구조를 OWL 문서의 의미론적 입장에서 클래스, 프로퍼티, 인스턴스로 분류하여 구조화한다. 그림 5는 제안하는 JeSPi 관계형 데이터베이스 모델의 전체적인 구조를 관계형 데이터 모델로 표현하고 있으며 크게 세 개의 계층으로 구

성된다. 본 논문에서 제안하는 JeSPi 관계형 데이터베이스 모델의 테이블 구조에 대한 적용 범위는 OWL DL에만 한정한다. OWL DL은 여러 OWL 언어 구문의 사용상 제약 사항을 포함하는 OWL의 하위 언어이다.

OWL DL은 클래스, 데이터타입, 데이터타입 속성, 객체 속성, 주석 속성(Annotation Properties), 온톨로지 속성(예: 온톨로지 임포트 및 버전 정보에 관련된 속성들), 개체, 데이터 값 및 사전 정의 어휘 간에 상호 배타적인 분리를 전제한다. 예를 들면, 클래스는 개체가 될 수 없는 특징들이 있다. OWL Full에서는 각 요소의 관계가 유연하며 클래스와 프로퍼티 그리고 인스턴스 간의 관계를 큰 제약조건 없이 표현할 수 있다. 이러한 특성은 저장 모델 정의를 매우 어렵게 한다. 무엇보다 현재 대부분의 저장 모델이 DL에 한정되어 개발되어 왔으며 그 실용성이나 정확성 측면에서도 DL을 우선적으로 채택하고 있기 때문에 본 논문에서도 OWL DL에 한정한다.

제안한 저장 모델은 다섯 개의 테이블로 구성된다. 실제 데이터는 UBA Tool을 이용해서 실제 온톨로지 데이터 셋인 LUBM을 생성한다. LUMB은 대학과 교수 학생에 대한 정보를 포함하고 있다. OWL 문서의 의미를 클래스, 프로퍼티, 인스턴스 그리고 클래스와 프로퍼티의 계층관계를 정의한 각각의 데이터 정보들을 테이블에 저장한다. 계층 1에서는 CLASS, PROPERTY 그리고 HIERARCHY 테이블로 구성되어 있다. CLASS 테이블에는 클래스 정보들이 저장되고 PROPERTY 테이블에는 프로퍼티 정보들이 저장된다. 그리고 각각의 CLASS와 PROPERTY의 상태를 C(클래스)와 P(프로퍼티)로 구분하여 계층관계에 대한 정보들을 HIERARCHY 테이블에 저장하여 표현한다. 이는 질의에 포함되어 있는 특정 클래스 정보의 하위 클래스 정보까지 추론하여 질의결과를 생성할 때 보다 빠른 연산을 가능하게 한다. 왜냐하면 Jena2의 관계형 데이터베이스 저장 모델을 통해 저장 될 때에는 계층관계에 대한 #type, #first, #rest정보 등이 추가적인 레코드로 저장되지만 JeSPi의 경우는 계층 구조에 필요한 정보만을 저장함으로써 불필요한 비용을 줄일 수 있기 때문이다. 이외에도 OWL에서 정의하고 있는 restrictions이나 axioms과 같은 다양한 제약조건을 정의하고 있다. 이러한 조건에 대한 판단이 요구되는 질의를 보다 빠르게 처리할 수 있도록 하기 위해서는 이러한 조건들을 추가적인 테이블을 통해 정의하여 관리할 필요가 있다. 그러나 본 논문에서는 계층구조에 초점을 두어 rdfs:subPropertyOf와 rdfs:subClassOf 조건만을 다루고 이 외의 조건들에 대해서는 향후 연구 내용으로 남겨 둔다.

계층 2의 INSTANCE 테이블에서는 IndividualID-IndividualName-PropertyID-Value 형태의 인스턴트와 해당 프로퍼티에 대한 정보들로 구성되며 계층 3은 CLASS 테이블과 INSTANCE 테이블의 관계를 사상시키는 테이블로 구성된다. 예를 들어, 계층 1의 CLASS 테이블에는 Professor, FullProfessor, AssistantProfessor의 데이터 레코드가 저장되고 PROPERTY 테이블에는 degreeFrom, doctorDegreeFrom, emailAddress의 데이터 레코드가 저장된다. 그리고 HIERARCHICAL 테이블에는 Professor의 자식클래스로 FullProfessor와 AssistantProfessor 사이의 관계가 저장된다. 이때 계층 2에서는 계층 1에서 삽입된 레코드정보를 이용하여 INSTANCE 테이블의 정보를 표현할 수 있다. 인스턴스가 Baik인 데이터는 PROPERTY 테이블에서 P003인 emailAddress를 참조하고 Baik 인스턴스 데이터의 emailAddress 프로퍼티인 Baik@email이 함께 저장된다. 즉, 인스턴스들의 프로퍼티와 이에 해당하는 value를 저장하여 인스턴스들의 정보를 표현한다. 계층 3의 CONCEPT 테이블에서는 I001의 INSTANCE 테이블의 정보를 참조하여 관계를 가지고 있는 클래스 정보인 C001 Professor를 저장한다.

제안한 저장 모델은 클래스와 클래스 간의 계층관계, 프로퍼티와 프로퍼티 간의 계층관계, 클래스, 프로퍼티, 인스턴스 간의 관계를 명확하게 표현할 수 있다. 따라서 데이터 이해의 용이성 및 질의 모델링의 편의성이 높고, 의미적으로 각각의 테이블로 분류하였기 때문에 질의 시 Subject-Property-Object형식으로 단일 테이블에 일괄 저장되었던 Jena2보다 높은 성능을 보인다. 이를 증명할 수 있는 성능 평가 결과에 대해서는 제 4장에서 상세하게 기술한다.

추가적으로, 기존 온톨로지 저장소를 Jena2에 적용하여 사용하는 방법을 고려해 볼 수 있다. 그러나 이러한 접근 방법은 다음과 같은 문제점을 야기한다. 첫째, Jena2에서 제공하는 다양한 API에 대한 활용성 저하 문제이다. 본 논문의 가장 큰 장점 중 한가지는 Jena2 API를 그대로 활용하면서 보다 나은 저장 모델을 Plug-in 형식으로 제공하는 것이다. 기존의 연구방식을 단순히 Jena2에 적용할 경우에는 시맨틱 웹과 관련된 상위의 애플리케이션을 개발할 수 있도록 패키지로 모듈화 되어있는 Jena2 API를 활용할 수 없는 문제점이 발생한다. 둘째, 기존의 연구방식과 Jena2 저장소방식의 인터페이싱 과정에서 많은 오버헤드가 발생된다. 셋째, 기존의 각 시스템마다 저장하는 방식이 서로 다르기 때문에 기존의 저장방식을 단순히 Jena2에 적용하기에는 문제점이 발생된다. 예를 들어

Sesame의 경우, 온톨로지 문서가 들어오면 문서로부터 리소스를 추출한 후에 추출한 리소스로부터 RDF트리플 형태로 변환한다. 하지만 Jena2의 경우에는 온톨로지 문서를 저장할 때 우선 그 문서들을 모델화(메모리 상에 생성되는 방향성 그래프)시킨 후 저장한다. 위와 같은 문제점으로 인해 본 논문에서는 상위 시맨틱 웹 어플리케이션을 개발하는데 편리함을 제공해주는 Jena2의 API 장점을 그대로 활용하면서 저장 성능을 더 개선할 수 있는 접근 방법을 이용한다.

3.3 어댑터 (Adapter)

그림 6은 Jena2 API를 이용하여 JeSPi를 구성할 수 있도록 제시한 JeSPi를 구성하는 컴퓨넌트 중 하나인 어댑터의 구조를 나타낸다.

JeSPi를 통해서 로드 된 OWL 온톨로지 데이터 집합은 Jena2 API에 의해 기본적인 온톨로지 모델을 생성한다. 온톨로지 모델은 어댑터의 추출 및 변환 단계를 거쳐 JeSPi 관계형 데이터베이스로 변환한다. JeSPi 관계형 데이터베이스 모델을 기반으로 관계형 데이터베이스 테이블을 구조화하여 생성한 후 데이터들을 미리 정의된 규칙 기반으로 저장한다.

어댑터는 OWL 데이터 정보에 대해 추출하고 분류하는 부분과 데이터를 임시 저장할 수 있는 메모리 영역, 그리고 관계형 데이터베이스에 변환하여 저장할 수 있도록 지원하는 부분 등 세 부분으로 구성되어 있다.

OWL 데이터 정보에 대해서 추출하고 분류하는 부분에서는 Jena2 프레임워크의 API를 이용하여 읽어 들인 OWL 데이터 집합의 의미를 분석한 후 요구하는 정보들을 JeSPi 관계형 데이터베이스 모델의 테이블 구조로 저장될 수 있도록 정보를 분류한다. 분류된 데이터 정보들을 임시로 저장할 경우 메모리 영역에 데이터를 보관한다. JeSPi 관계형 데이터베이스 모델을 생성하고 테이블을 생성하는 과정을 거쳐 분류된 데이터 정보들을 테이블에 저장한다. 이와 같이 OWL 데이터 집합을 어댑터의 저장

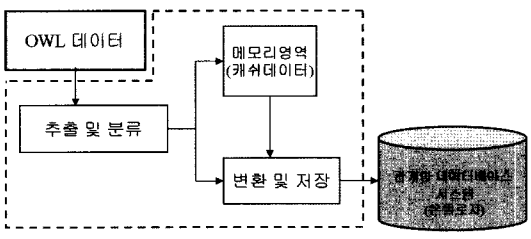


그림 6. 어댑터 구조

프로세스를 통해 JeSPi 관계형 데이터베이스에 저장하면 Jena2에서 제공하는 API를 그대로 사용할 수 있고 단일화로 구성된 저장 모델의 단점을 보완할 수 있다.

4. 비교평가 : 실험 및 시뮬레이션

이 장에서는 제안 저장 모델에 대한 평가를 위해 수행한 실험 및 시뮬레이션 결과에 대하여 기술한다. 실제 웹 온톨로지 데이터 셋을 관계형 데이터베이스에 저장하고 검색하는 실험을 통해 Jena2와 JeSPi 저장모델을 비교한다. 실제 실험은 실험 환경에 의존적이다. 따라서 이 논문에서는 보다 정형화 된 모델을 정의하고 이를 바탕으로 시뮬레이션을 수행한다. 이를 위해 각 질의 패턴을 관계대수를 이용하여 수학적으로 분석하여 시뮬레이션 모델을 정의한다.

4.1 실험

이 장에서는 Jena2와 JeSPi의 저장 모델에 대한 실험 및 비교평가 결과에 대하여 기술한다. JeSPi는 Jena2의 다양한 API를 그대로 활용하면서 Jena2의 저장소 성능을 향상시키기 위해 제안되었다. 따라서 비교평가를 위해 JeSPi와 기존 Jena2 저장소 간 성능 비교가 반드시 이루어져야 하며 이러한 이유로 인해 Jena2를 평가를 위한 비교 대상으로 선택한다.

4.1.1 실험 환경

실험 및 비교평가를 위한 시스템 환경은 다음과 같다.

- CPU : Pentium 4(2.81GHz)
- 메모리 크기 : 512MB
- 힙 메모리 크기 : 1024MB
- 하드디스크 크기 : 100GB
- 운영체제 : Windows Server 2K OS
- 언어 : Java
- Java JDK 버전 : Java JDK 1.6.0
- 데이터베이스 관리시스템 : Oracle 9i

본 논문에서는 Lehigh 대학교에서 제공하는 온톨로지 생성 도구인 UBA(University-Bench Artificial Data Generator)에 의해 생성된 데이터 집합을 이용하여 실험 및 평가를 수행한다. UBA는 SWAT 프로젝트의 결과에 대한 평가를 위해 개발된 도구이며[16] 많은 실험들이 이 도구를 이용하여 생성된 데이터 집합을 사용한다[17-20]. UBA는 LUBM(n,s)의 형태로 데이터를 생성하며 n과 s

는 각각 대학 수와 시드 값을 의미한다. 만일 LUBM(1,0)과 같은 인수 값을 이용하여 데이터 값을 생성할 경우, 대학 수가 1임을 의미하며 대학은 다수 개의 학과, 교수, 학부, 대학원 등의 정보들로 구성된다.

UBA를 이용하여 OWL 데이터 집합을 사용하는 이유는 타당한 크기의 온톨로지를 생성하여 실험할 수 있다는 점이다. 웹 상에서 배포되고 있는 온톨로지들의 경우 그 크기가 작기 때문에 보다 정확한 실험결과를 얻기 어렵다. 그러나 UBA 도구의 경우 매우 방대한 크기의 온톨로지 생성을 가능하게 하여 대용량 온톨로지의 성능을 평가하는데 적합하다.

본 논문에서는 UBA에 의해 생성된 온톨로지인 LUBM(1,0) 집합을 이용한다. LUBM(1,0) 집합을 생성하게 되면 총 15개의 OWL 웹 문서가 기본으로 구성된다. 실험을 위해 온톨로지 크기를 각각 문서 파일 1개, 5개, 10개, 15개로 분류하며 각각의 경우에 대하여 실험 및 비교평가를 수행한다. LUBM(1,0)이 포함하는 총 15개의 OWL 온톨로지 데이터 집합들을 4개의 그룹으로 나누어 각각에 대하여 실험하였다. 본 논문에서는 파일 개수에 따라 LUBM(1,0,1), LUBM(1,0,5), LUBM(1,0,10), LUBM(1,0,15)로 정의한다.

UBA를 통해서 LUBM(1,0) 보다 큰 대용량의 데이터 집합을 생성하기 위해서는 LUBM(5,0), LUBM(10,0) 등을 생성하여 이용할 수 있다. 그러나 본 논문의 실험에서는 LUBM(1,0)만으로도 충분한 질의 성능 차이를 도출할 수 있다. 결국 실험 결과와 같이 데이터의 양이 늘어남에 따라 결과가 충분히 예측이 가능하기 때문에 LUBM(1,0) 데이터 집합에 제한하여 실험을 하였다.

실험 평가는 다음과 같이 질의 응답 시간, 질의 결과 개수, 완전성 및 정확성 등에 대하여 이루어진다.

- 질의 응답 시간(Query Response Time) : 이 실험의 가장 중요한 평가 항목으로 질의 응답시간을 평가한다. 단위는 보다 자세한 측정을 위해서 실험 단위를 ms로 한다. 경우에 따라 질의 응답 시간이 약간씩 편차가 발생 할 수 있으므로 하나의 질의문 당 총 5번의 질의 시간을 측정하여 평균을 구하였다.

- 질의 결과 개수(The Number of Results (Tuples)) : 질의문에 대한 질의 결과를 평가함으로써 Jena2 저장구조와 JeSPi 저장구조에 대해서 정확한 질의가 이루어졌는지를 평가한다.

- 완전성(Completeness) : 질의하는 과정에서 원하는 질의 항목들이 완벽하게 검색되었는지에 대해서 평가한다. 원하는 항목의 데이터 결과가 모자라거나 없어진 것이 있는지 확인하는 검사의 평가가 이루어진다.

- 정확성(Accuracy) : 질의하는 과정에서 원하는 질의 항목들을 정확하게 검색되었는지에 대해서 평가한다. 원하는 질의 결과 외에 다른 결과가 추가되어 검색되지 않았는지에 대한 정확성 평가가 이루어진다.

실험을 위한 5가지 패턴의 질의를 정의하여 이용한다. 질의 패턴은 질의 대상과 계층 구조에 의해 정의되며 정의된 Q-P(Query-Pattern)은 다음과 같다.

- Q-P1 : C(Class): 모든 클래스 정보 검색
- Q-P2 : CI(Class-Instance): 특정 단일 클래스의 인스턴스 검색
- Q-P3 : CH(Class-Hierarchical): 특정 클래스의 모든 하위 클래스 검색
- Q-P4 : PH(Property-Hierarchical): 특정 프로퍼티의 모든 하위 프로퍼티 검색
- Q-P5 : CHI(Class-Hierarchical-Instance): 특정 클래스의 모든 하위 클래스에 해당하는 모든 인스턴스 검색

위 질의 패턴에 따라 실험을 위해 정의한 실제 질의 예제를 SPARQL 언어로 표현하면 표 1과 같다. 표 1에서, 첫 번째 질의패턴인 Q-P1은 전체 클래스에 대한 정보만을 검색하는 질의문으로 Professor, AssistantProfessor, AssociateProfessor,VisitingProfessor 등의 클래스 정보들만 검색된다.

두 번째 질의패턴인 Q-P2는 특정 클래스 정보에 속하는 인스턴스 정보들을 검색하는 질의문이다. 클래스 객체가 AssistantProfessor의 emailAddress를 검색하면 AssistantProfessor0@department0.University0.edu 등 특정 클래스 객체에 대한 해당 email정보가 검색된다.

세 번째 질의패턴인 Q-P3는 특정 클래스 정보에 대한 하위 클래스들을 검색하는 질의문이다. Professor 클래스 객체에 대한 모든 하위 클래스 정보들을 질의하여 AssistantProfessor, AssociateProfessor, Chair, FullProfessor 등이 검색된다.

네 번째 질의패턴인 Q-P4는 특정 프로퍼티에 대한 하위 프로퍼티정보를 검색하는 질의문으로 degree From의

표 1. 각 질의 패턴을 위해 정의한 질의 예제

질의패턴	설 명

Q-P1	select ?subj
	where {
	?subj ?Prop Uv::http://www.w3.org/2002/07/owl#Class
	}

	모든 Class를 검색하는 단순질의를 통해 질의 응답속도를 실험.

Q-P2	select ?Obj
	where {
	?subj Uv::http://www.lehigh.edu/~zhp2/2004/0401/univ- bench.owl# emailAddress ?Obj
	?subj ?prop Uv::http://www.Department0.University0.edu/AssistantProfessor7
	}

	AssistantProfessor7의 email 주소를 검색.

Q-P3	select ?class
	where{
	{ ?class Uv::http://www.w3.org/2000/01/rdf-schema#subClassOf ?Obj.
	?Obj Uv::http://www.w3.org/2000/01/rdf-schema#subClassOf
	Uv::http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#Professor
	}
	union all
	{ ?class Uv::http://www.w3.org/2000/01/rdf-schema#subClassOf
	Uv::http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#Professor
	}
	}

	모든 교수클래스를 검색.

Q-P4	select ?
	where{
	{ ?subj Uv::http://www.w3.org/2000/01/rdf-schema#subPropertyOf?Obj.
	?Obj Uv::http://www.w3.org/2000/01/rdf-schema#subPropertyOf
	Uv::http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#degreeFrom
	}
	union all
	{ ?subj Uv::http://www.w3.org/2000/01/rdf-schema#subPropertyOf
	Uv::http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#degreeFrom
	}
	}

	DegreeFrom에 속하는 모든 프로퍼티를 검색.

Q-P5	select ?ins
	where{
	?ins rdf:type ?subj
	{
	?subj Uv::http://www.w3.org/2000/01/rdf-schema#subClassOf ?Obj.
	?Obj Uv::http://www.w3.org/2000/01/rdf-schema#subClassOf
	Uv::http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#Professor
	}
	union all
	{ ?subj Uv::http://www.w3.org/2000/01/rdf-schema#subClassOf
	Uv::http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#Professor
	}
	}

	모든 교수들을 검색하시오. 즉 모든 Professor 클래스의 하위 클래스를 검색하고 그에 해당하는 인스턴스를 검색.

표 2. 질의 응답 시간

단위 : ms

모델분류 / 데이터	Jena2					JeSPi				
	P-Q1	P-Q2	P-Q3	P-Q4	P-Q5	P-Q1	P-Q2	P-Q3	P-Q4	P-Q5
LUBM(1,0,1)	506.4	1417.6	515.8	516.0	1118.4	500.0	891.0	500.0	500.0	503.0
LUBM(1,0,5)	678.2	2096.8	516.0	524.6	1443.8	500.0	1153.2	500.0	500.0	512.8
LUBM(1,0,10)	681.4	3406.4	516.0	596.0	9481.0	500.0	1818.8	500.0	500.0	2971.8
LUBM(1,0,15)	825.0	4603.0	516.0	624.2	11796.6	500.0	2034.4	500.0	500.0	3772.6

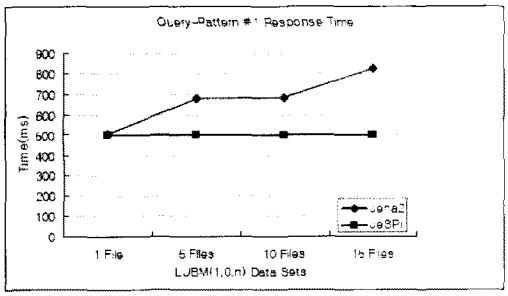


그림 7. P-Q1의 실험 결과

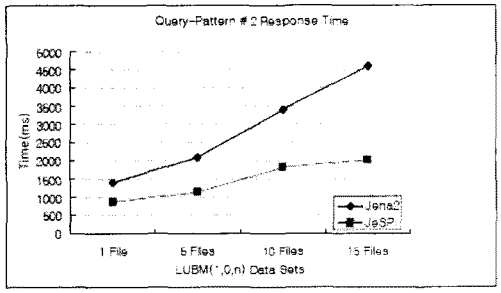


그림 8. P-Q2의 실험 결과

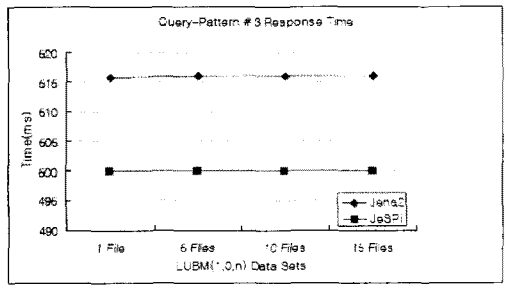


그림 9. Q-P3의 실험 결과

프로퍼티에 대한 하위 프로퍼티 정보들인 doctoralDegree-From, mastersDegreeFrom, undergraduateDegreeFrom 등이 검색된다.

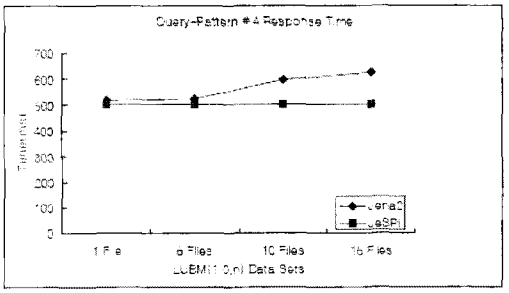


그림 10. P-Q4의 실험 결과

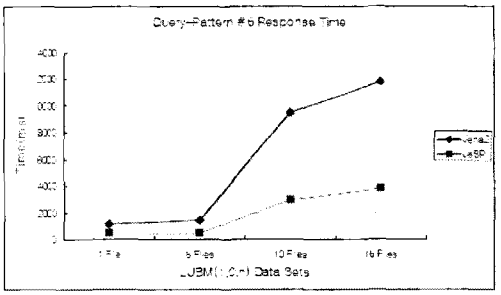


그림 11. P-Q5의 실험 결과

마지막으로 다섯 번째 질의패턴인 Q-P5는 특정 클래스에 대한 하위의 모든 클래스들에 속한 모든 인스턴스들을 검색하는 질의문이다. Professor 클래스 객체의 모든 하위 클래스들(AssistantProfessor, AssociateProfessor, Chair, FullProfessor 등)에 속한 모든 인스턴스들이 검색된다.

4.1.2 실험 결과

이 절에서는 앞서 정의한 각 평가 항목에 대한 실험 결과에 대하여 기술한다.

(1) 질의 응답 시간에 대한 실험 결과

표 2는 5개의 질의 패턴을 이용하여 각 데이터 집합에 대하여 실험한 전체 결과를 보여준다.

그림 7부터 그림 11까지는 표 2를 바탕으로 실험결과

를 그래프 모델로 표현한 것이다. 그림 7에서 알 수 있듯이, Q-P1, Q-P3, Q-P4에서는 응답속도가 최소 6~325ms 정도의 차이로 JeSPi 환경에서 빠른 응답속도를 확인할 수 있다. 이는 Jena2와 JeSPi 환경 모두 별도의 테이블에서 subClassOf와subPropertyOf에 대한 계층적 정보들을 관리하기 때문에 JeSPi의 성능이 우수함을 확인할 수 있다. Q-P2와 Q-P5에서도 역시 보다 나은 성능을 확인할 수 있다.

JeSPi가 Jena2에 비해 향상된 성능을 보이는 이유를 정리하면 다음과 같다. 첫째, 클래스와 관계된 인스턴스를 검색하는 과정에서 조인 연산이 이루어진다. 이 과정에서 Jena2는 단일구조 테이블 전체에 대해서 조인 연산이 요구된다. 그러나 JeSPi는 의미 별로 정의된 테이블 중 필요한 테이블만을 선택하여 조인 연산이 이루어지기 때문에 질의 응답 시간이 빠르다. 둘째, 하위 클래스를 조인하는 과정에서 JeSPi는 불필요한 정보를 제거하고 계층관계의 정보만 저장한 테이블을 사용하기 때문에 역시 Jena2보다 빠른 질의 응답 시간을 보인다.

그림 7과 그림 10에서 파일의 크기가 증가 할수록 Jena2에서는 질의 응답 시간이 증가하였으나 JeSPi에서는 변화되지 않았다. 이러한 이유는 UBA를 이용하여 데이터 집합을 생성할 시에 기본적으로 생성되는 15개의 데이터 파일 클래스와 프로퍼티 정보들은 모두 일정한 수로 제한되어 있기 때문이다. 즉, LUBM(1,0,1)과 LUBM(1,0,15) 데이터 집합의 다른 점은 제한된 클래스와 프로퍼티 데이터 수를 활용하여 얼마나 많은 인스턴스 데이터들을 생성했는지에 대한 차이이다.

그림 11에서는 인스턴스를 검색하는 질의문이기 때문에 LUBM(1,0,1)에서 LUBM(1,0,15)로 갈수록 인스턴스 수가 증가하여 이에 대한 질의 응답 시간이 증가하게 된다. 표 3의 각 질의 패턴 별 결과 개수를 참조하면 쉽게 이해될 수 있다.

그림 9에서, Jena2와 JeSPi사이에서 질의 응답 시간의 차이가 나는 이유는 JeSPi에서는 각 클래스와 프로퍼티 간 계층관계에 필요한 정보만을 저장한다. 이에 반해 Jena2의 관계형 데이터베이스 저장 모델에서는 계층관계에 대한 #type, #first, #rest 정보 등을 함께 저장하기 때문에 질의 속도에서 차이가 발생한다. 하지만 위에서 서술한 바와 같이 데이터 집합이 증가되어도 클래스와 프로퍼티의 개수는 제한적으로 동일하기 때문에 Jena2와 JeSPi 모델 모두 질의 응답 시간은 변화가 없다.

(2) 질의 결과 개수, 완전성 및 정확성에 대한 실험 결과

표 3은 Jena2와 JeSPi 저장 모델을 질의 결과 개수, 결과의 완전성 및 정확성에 대한 실험 결과를 보여준다. 본 논문에서 완전성이란 특정 질의가 주어졌을 때 실제 얻어져야 하는 모든 결과를 생성했는지에 대한 문제를 의미한다. 반면 정확성이란 원하는 결과들만을 생성하였는지에 대한 문제에 초점을 두게 된다. 예를 들어, 실제 얻어져야 하는 결과 집합을 {a, b, d}라고 가정하고 질의의 결과 집합이 {a, b, c, d}라고 할 때 완전성은 100%가 된다. 그러나 정확성은 전체 결과 중 3개만이 정확한 결과이고 1개는 정확한 결과가 아니므로 3/4 * 100 = 75%가 된다.

결과 개수, 완전성, 정확성 등에 대한 실험결과를 기술하는 이유는 두 가지 이다. 첫째, 물리적인 변환 시 기존의 기능을 정확히 수용하고 있는지에 대해서 평가가 이루어져야 한다. OWL 온톨로지 데이터 집합을 관계형 데이터베이스 저장소로 물리적인 변환을 할 때 기존에 처리되었던 검색이 데이터 손실 없이 정확히 기능화 되는지에 대한 평가 지표로 활용된다. 아무리 성능이 향상되어도 데이터의 손실이 생긴다면 성능향상의 목적이 타당성을 잃기 때문이다.

둘째, 성능향상에 대한 신뢰성을 제공한다. 데이터의 손실이 없는 것을 함께 평가함으로써 성능향상에 대한 신뢰성을 보장하기 때문이다. 표 3은 전 항목에 대해서 완전성과 정확성 모두 100%를 나타낸다. 이는 Jena2와 JeSPi의 저장 모델 각각에 대해서 질의문에 대한 의미가 정확히 일치하는 것을 증명한다.

표 3. 질의 결과 개수에 대한 실험 결과

데이터셋 / 질의패턴	LUBM(1,0,1)		LUBM(1,0,5)		LUBM(1,0,10)		LUBM(1,0,15)	
	Jena2	JeSPi	Jena2	JeSPi	Jena2	JeSPi	Jena2	JeSPi
Q-P1(C)	43							
Q-P2(CI)	10	10	46	46	94	94	146	146
Q-P3(CH)	6							
Q-P4(PH)	3							
Q-P5(CHI)	34	34	147	147	294	294	447	447

표 3의 질의 결과 개수에서, 전체의 클래스를 검색하는 Q-P1과 특정 클래스의 모든 하위클래스를 검색하는 Q-P3에서는 파일의 개수가 추가되더라도 전체 클래스의 결과에 대해서는 응답시간의 큰 차이를 보이지 않았다. 이는 LUBM(1,0,1) 데이터 집합에서는 한정된 클래스 항목에 대해서 인스턴스 정보들만이 추가되기 때문이다. 또한 특정 프로퍼티의 모든 하위 프로퍼티를 검색하는 Q-P4 역시 같은 비슷한 결과를 유지하였다. 하지만 인스턴스의 결과를 질의하는 Q-P2와 Q-P5의 질의문은 저장하는 파일의 개수가 추가될수록 결과 값은 증가되었다.

4.2 시뮬레이션

이 장에서는 4.1절에서의 실험 및 평가에 추가적으로 계층적 구성이 갖는 성능향상 정도를 수학적으로 분석하여 제시하고, 시뮬레이션 결과가 실제 실험 결과와 얼마나 일치하는 지에 대한 평가내용을 서술한다.

4.2.1 시뮬레이션 모델 정의

이 논문에서 시뮬레이션을 위한 계산 모델은 각각의 저장 구조에 대한 질의 결과를 생성하기 위한 관계 대수를 기반으로 정의한다.

표 4에서 정의한 관계대수를 바탕으로 표 5과 같은 계산모델을 유도할 수 있다. 이러한 모델을 통해서 관계형 데이터베이스에 저장되어 있는 웹 온톨로지 데이터들의 조회 횟수를 산정하여 질의 응답 속도를 비교할 수 있다.

표 5에서 n(J)는 Jena2의 단일 테이블의 레코드 개수를 의미하고, n(C), n(P), n(I)는 각각 Class, Property, Instance 테이블의 레코드 개수를 의미한다. 또한 n(HC), n(HP)는 각각 Hierarchy Class(계층적인 클래스)와 Hierarchy Property(계층적인 프로퍼티) 테이블의 레코드 개수를 나타낸다.

표 5의 Q-P1에서 클래스 정보만을 검색하였으므로 클래스 정보를 검색하기 위한 조회 횟수를 계산 모델화 하

표 4. 각 질의 패턴 별 관계 대수 표현

Q-P1	Jena2	$\Pi_{jena.s}\left(\sigma_{jena.o='dass'}(JENA)\right)$
	JeSPi	$\Pi_{dassname}(CLASS)$
Q-P2	Jena2	$\Pi_{jena.o}\left(\sigma_{jena.o='assistantprofessor7'\wedge jena.p='emailaddress'}(JENA)\right)$
	JeSPi	$\Pi_{ins.value}\left(\sigma_{ins.name='assistantprofessor7'\wedge ins.property='emailaddress'}(INSTANCE)\right)$
Q-P3	Jena2	$\Pi_{jena.s}\left(\sigma_{jena.o=jena.s\wedge jena.o='professor'}(JENA\times JENA)\right)\cup$ $\Pi_{jena.s}\left(\sigma_{jena.o='professor'}(JENA)\right)$
	JeSPi	$\Pi_{da.name}\left(\sigma_{hie.childid=hie.parentid\wedge cls.name='professor'}(HIERARCHY\times CLASS)\right)\cup$ $\Pi_{da.name}\left(\sigma_{cla.name='professor'}(CLASS)\right)$
Q-P4	Jena2	$\Pi_{jena.p}\left(\sigma_{jena.p=jena.p\wedge jena.p='degreefrom'}(JENA\times JENA)\right)\cup$ $\Pi_{jena.p}\left(\sigma_{jena.p='degreefrom'}(JENA)\right)$
	JeSPi	$\Pi_{pro.name}\left(\sigma_{hie.childid=hie.parentid\wedge pro.name='degreefrom'}(HIERARCHY\times PROPERTY)\right)\cup$ $\Pi_{pro.name}\left(\sigma_{pro.name='degreefrom'}(PROPERTY)\right)$
Q-P5	Jena2	$\Pi_{jena.s}\left(\sigma_{jena.s=jena.o}(JENA)\left(\sigma_{jena.p=jena.p\wedge jena.p='degreefrom'}(JENA\times JENA)\right)\right)\cup$ $\Pi_{jena.s}\left(\sigma_{jena.s=jena.o}(JENA)\left(\sigma_{jena.p='degreefrom'}(JENA)\right)\right)$
	JeSPi	$\Pi_{con.insname}\left(\sigma_{con.name=da.name}(CONCEPT)\left(\sigma_{hie.childid=hie.parentid\wedge cls.name='professor'}(HIERARCHY\times CLASS)\right)\right.$ $\Pi_{con.insname}\left(\sigma_{con.name=da.name}(CONCEPT)\left(\sigma_{cla.name='professor'}(CLASS)\right)\right)$

s = subject column; o = object column; p = property column; ins = instance table; cla = class table; pro = property table; con = concept table; hie = hierarchy table;

표 5. 계산 모델

Q-P	Model	Formula
Q-P1	Jena	*n(J)*c*
	JeSPi	*n(C)*c*
Q-P2	Jena	*n(J)*c + n(J)*c*
	JeSPi	*n(I)*c + n(I)*c*
Q-P3	Jena	*n(J)*c *n(J)*c + n(J)*c*
	JeSPi	*n(C)*c * n(HC)*c + n(C)*c*
Q-P4	Jena	*n(J)*c * n(J)*c + n(J)*c*
	JeSPi	*n(P)*c * n(HP)*c + n(HP)*c*
Q-P5	Jena	*n(J)*c *n(J)*c + n(J)*c + n(J)*c*
	JeSPi	*n(C)*c * n(HC)*c + n(C)*c + n(I)*c*

(*c*는 random value, 단 $0 < c < 1$)

였다.

Q-P2에서는 첫 번째 조건인 property가 emailaddress에 해당하는 데이터를 검색 후, 검색된 데이터에서 assistantprofessor를 재검색한다. 즉, 첫 번째 조건을 검색하기 위한 조회 횟수와 두 번째 조건을 검색하기 위한 조회 횟수를 합하여 Q-P2의 조회횟수를 산정한다.

특정 클래스의 계층적인 클래스들을 모두 검색하는 Q-P3에서는 Jena2 저장소 환경의 경우 특정 클래스의 계층적 정보를 얻기 위해서 셀프 조인을 통해 특정 클래스 하위의 클래스들을 검색할 수 있다. 하지만 JeSPi 저장 환경에서는 계층적인 정보만을 별도로 테이블에 저장시킴으로써 Jena2에서 검색하는 조회 횟수보다 적은 조회 횟수로 검색이 가능하다. 조회하는 횟수가 상대적으로 적다는 것은 그만큼 데이터에 접근하는 비용이 절약된다는 의미이므로 질의 응답 시간이 단축 되었다는 것을 유추할 수 있다.

Q-P5는 특정 클래스의 하위 클래스를 검색하고 검색된 클래스에 속하는 모든 인스턴스들을 검색한다. 즉, Q-P3과 같이 계층적인 클래스를 조회하고 검색된 클래스들의 인스턴스를 검색하기 때문에 계층적인 클래스를 조회하는 횟수에 인스턴스를 조회한 횟수를 합하여 최종 조회횟수를 산정한다.

4.2.2 시뮬레이션 환경 및 결과

Jena2와 JeSPi의 저장 모델에 대한 시뮬레이션 평가를 위해 가상의 웹 온톨로지 데이터 셋을 생성한다. Jena2와 JeSPi 저장소 모델에 대해 생성되는 규칙은 다음과 같이 정의한다.

- 초기 클래스는 총 10개
  : n(Class)=10;
- 하나의 클래스에 대한 계층적 클래스는 5개
  : n(Hierarchy_class) / 1 Class = 5
- 하나의 클래스에 계층적 프로퍼티는 5개
  : n(Hierarchy_property) / 1 Class=5
- 하나의 클래스에 인스턴스 프로퍼티는 5개
  : n(Instance_property) / 1 Class=5

위와 같이 초기화된 웹 온톨로지 데이터 셋을 바탕으로 클래스와 인스턴스를 각각 5배, 10배씩 증가시켰으며, 각 데이터가 증가됨에 따라 클래스, 인스턴스, 프로퍼티 등의 개수도 정의된 규칙에 맞게 생성한다.

첫 번째 온톨로지 데이터 셋은 초기화된 10개의 클래스를 시작으로 클래스의 개수를 5배씩 증가시켰다. 하나의 클래스에는 5개의 계층적인 클래스가 존재하는 규칙에 따라 전체 클래스의 개수는 50, 250, 1250, 6250 개로 증가하게 된다. 또한 클래스 하나당 5개의 인스턴스가 존재하는 규칙에 따라 인스턴스의 개수도 250, 1250, 6250, 31250 개로 증가한다. 그리고 계층적인 클래스와 프로퍼티의 개수도 함께 증가한다.

두 번째 온톨로지 데이터 셋은 클래스 개수는 고정된 상태로 인스턴스의 개수만 10배씩 증가시킨다. 클래스의 개수에는 변화 없이 인스턴스의 수만 증가시켰기 때문에 계층적인 클래스의 개수는 그대로 유지되며, 인스턴스는 250, 1250, 6250, 31250 개로 증가하고 Jena2의 레코드 개수 역시 625, 6250, 62500, 625000 으로 증가한다.

이처럼 클래스와 인스턴스의 개수를 증가시킨 이유는 표 1에서 정의한 질의를 처리 시 클래스와 인스턴스 수의 증가가 데이터를 처리하는 시간에 큰 영향을 미치는 요소이기 때문이다.

그림 12, 13, 14, 15, 16은 시뮬레이션의 첫 번째 실험 결과로, 클래스를 5배씩 증가함에 따른 웹 온톨로지 데이터 셋의 조회 횟수를 각 질의 패턴 별로 시뮬레이션한 결과이다.

Q-P1과 Q-P2의 시뮬레이션 결과인 그림 12와 그림 13에서 JeSPi 저장소 환경보다 Jena2 저장소 환경에서의 레코드 조회 횟수가 더 많이 요구된 것을 확인 할 수 있다. 이는 웹 온톨로지를 의미별로 분류하여 클래스 정보만을 클래스 테이블에 따로 저장시켜 놓았기 때문이다. 따라서 JeSPi에서 클래스 관련 정보를 검색 시 클래스 테이블 만을 검색함으로서 Jena2 보다 적은 조회횟수가 요구된다.

그림 14, 15, 16에서 클래스의 개수가 증가 할수록 두 그래프가 더욱 큰 차이가 보이는 이유는 계층적인 구조를 검색하는 질의 패턴이기 때문이다. Jena2의 경우는 계층

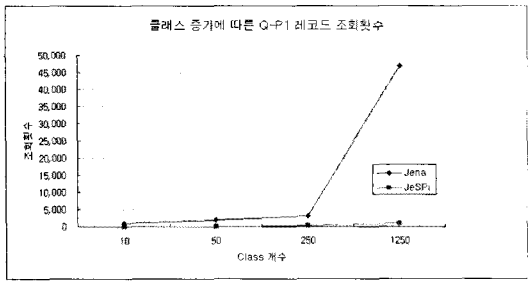


그림 12. 클래스증가에 따른 P-Q1 시뮬레이션 결과

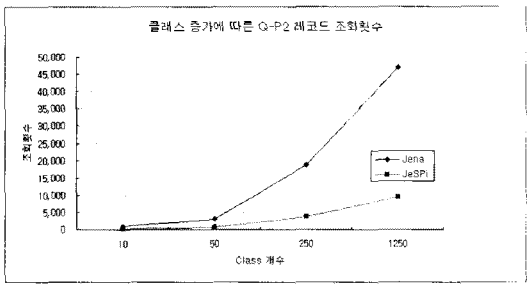


그림 13. 클래스증가에 따른 P-Q2 시뮬레이션 결과

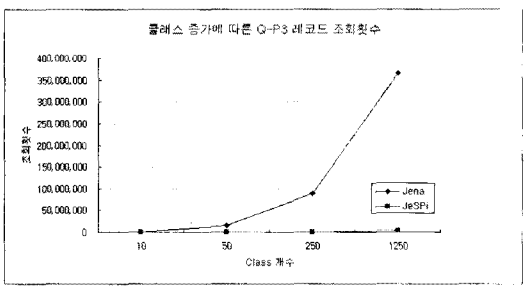


그림 14. 클래스증가에 따른 P-Q3 시뮬레이션 결과

적인 구조를 파악하기 위해서 자체조인을 해야 하지만 JeSPi 저장구조에서는 클래스와 프로퍼티 각각의 계층구조만을 저장할 수 있는 별도의 테이블에 저장되기 때문에 계층적인 구조의 정보들을 파악하는데 있어서 Jena2보다 적은 조회횟수를 요구한다. 따라서 적은 조회횟수를 요구하는 만큼 질의 처리 시간을 절약할 수 있다.

그림 9와 그림 14는 같은 질의 패턴에 대해서 실험하고 시뮬레이션한 결과이다. 그림 9에서는 Jena2와 JeSPi 간에 유사한 실험 결과를 확인할 수 있지만, 그림 14에서는 다소 차이가 발생하였다. 실험의 경우, 사용된 LUBM 데이터 셋이 생성될 때, 클래스의 개수는 일정하게 정해져있고 단지 인스턴스의 개수만이 증가하며 생성되었다. 그러나 시뮬레이션의 경우는 클래스의 개수도 함께 증가

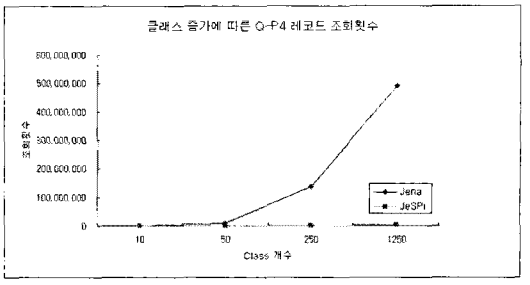


그림 15. 클래스증가에 따른 P-Q4 시뮬레이션 결과

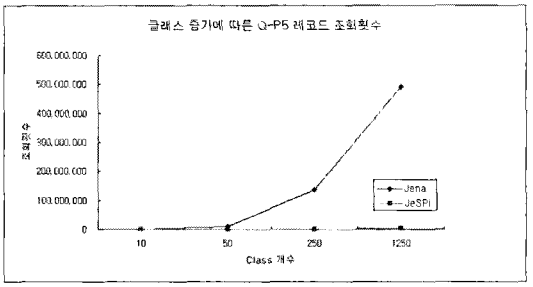


그림 16. 클래스증가에 따른 P-Q5 시뮬레이션 결과

하며 많은 수의 클래스에 대해서 시뮬레이션되었기 때문에 Jena2와 JeSPi간에 차이를 보인다.

클래스의 개수를 증가시킴에 따라 조회 횟수의 결과를 시뮬레이션으로 실행해본 결과, Q-P1부터 Q-P5까지 모두 데이터의 개수가 증가 할수록 Jena2 저장소 모델과 JeSPi저장소 모델사이에는 더 큰 폭으로 조회 횟수의 차이가 생겼다. 이와 같은 결과는 4.1절에서 질의 패턴별 질의 응답 시간의 결과와 유사하다.

그림 17, 18, 19, 20, 21은 시뮬레이션의 두 번째 실험 결과로, 인스턴스의 개수를 5, 50, 500, 5000개 등, 10배씩 증가함에 따른 웹 온톨로지 데이터 셋의 조회 횟수를 각 질의 패턴 별로 시뮬레이션한 결과이다. 앞의 시뮬레이션 결과와 마찬가지로 Q-P1과 Q-P2에 대해서 Jena2 저장소 환경의 레코드 조회 횟수가 JeSPi 저장소 환경보다 더 많은 조회 횟수를 요구하였다. 이 역시 웹 온톨로지를 의미별로 분류하여 클래스 정보들을 클래스 테이블에 따로 저장시켜 놓았기 때문에 클래스 관련 정보를 검색 시, Jena2의 단일 테이블 전체를 검색하는 것이 아닌, 클래스 테이블만을 검색함으로서 Jena2 보다 적은 조회횟수가 요구된다. 또한 그림 19, 20, 21에서 인스턴스의 개수가 증가 할수록 두 그래프가 더욱 큰 차이가 보이는 이유 역시 계층적인 구조를 검색하는 질의 패턴이기 때문이다.

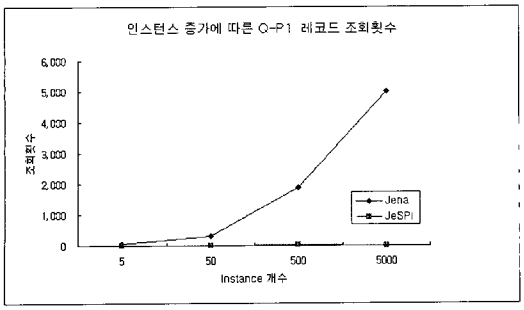


그림 17. 인스턴스증가에 따른 P-Q1 시뮬레이션 결과

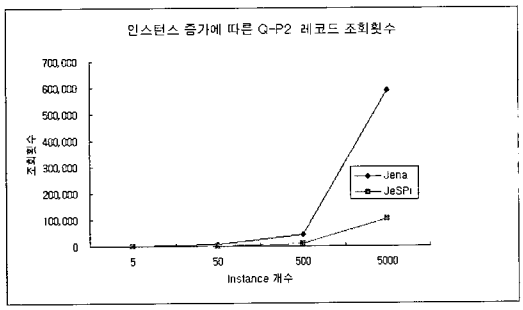


그림 18. 인스턴스증가에 따른 P-Q2 시뮬레이션 결과

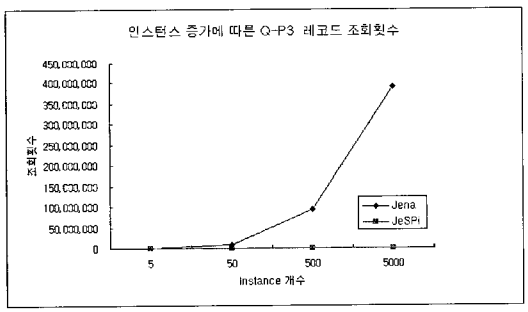


그림 19. 인스턴스증가에 따른 P-Q3 시뮬레이션 결과

즉, 인스턴스를 증가시킴에 따라 Q-P1부터 Q-P5까지 모두 JeSPi 환경이 Jena2환경보다 적은 조회횟수가 요구되는 것을 확인할 수 있다. 인스턴스의 개수를 증가시킴에 따라 조회 횟수의 결과를 시뮬레이션으로 실행해본 결과, 역시 Q-P1부터 Q-P5까지 모두 데이터의 개수가 증가할수록 Jena2 저장소 모델과 JeSPi저장소 모델사이에는 더 큰 폭으로 조회 횟수의 차이가 발생하였다. 이와 같은 결과 역시 4.1절에서 질의 패턴별 질의 응답 시간의 결과와 유사하다.

결론적으로 질의 응답 시간을 체크하기 위한 실험과 조회 횟수를 체크하는 시뮬레이션 모두 온톨로지 데이터

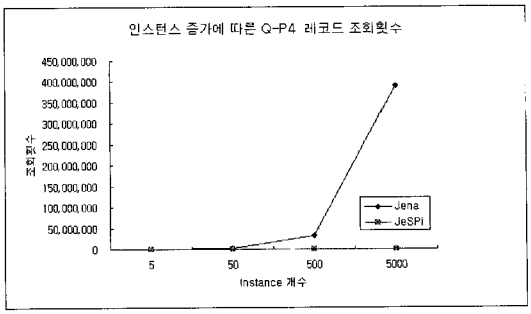


그림 20. 인스턴스증가에 따른 P-Q4 시뮬레이션 결과

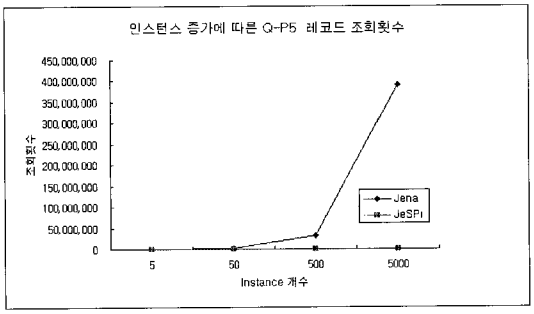


그림 21. 인스턴스증가에 따른 P-Q5 시뮬레이션 결과

셋이 증가 할수록 JeSPi저장소의 검색 성능이 Jena2저장소의 검색 성능보다 우수한 것을 확인할 수 있다.

4.3 평가

앞 절에서의 실험과 시뮬레이션을 통해 Jena2 저장소 모델과 JeSPi 저장소 모델의 성능 및 비교평가는 표 6과 같다.

Jena1에서는 트리플 데이터(Subject-Property-Object)를 저장하기 위해 정규화된 데이터베이스 스키마를 사용하였기 때문에 저장 크기 면에서는 효율적인 장점이 있으나, 질의 시 Statement 테이블과 리터럴 테이블 그리고 리소스 테이블 간에 많은 횟수의 조인 연산을 필요로 해야 하는 단점이 발견된다. 그러나 Jena2의 경우 Jena1의 문제점을 해결하기 위해 의도적으로 비정규화 함으로써 Jena1과 비교하여 데이터의 검색 시간과 조인 연산의 횟수는 줄일 수 있는 장점이 있지만, 하나의 테이블 공간이 커지는 단점이 있고 이는 단순 선택 연산 시 성능이 저하되는 문제점을 지닌다. 또한 비정규화가 높아 여전히 조인 연산 시 불필요한 정보를 액세스하게 되고 이에 따른 성능 저하를 가져온다.

제안 모델은 OWL에서 정의한 클래스, 프로퍼티, 인스턴스, 계층 구조의 개념을 이용하여 OWL 문서의 데이터

표 6. 비교 평가 결과

비교항목	Jena2 저장모델	제안모델
검색 (질의) 속도	느림	빠름
데이터 중복성	높음	낮음
데이터 이해의 용이성	낮음	높음
질의 모델링의 편의성	낮음	높음
모델 변환의 용이성	낮음	높음

를 관계형 데이터베이스에 저장함으로써 단순정보검색 질의 시 Jena2에서 비 정규화된 테이블 구조로 저장할 때보다 질의 응답 속도를 향상시킬 수 있다. 또한 조인 연산 시 두 테이블의 크기로 인하여 조인비용이 발생하는 문제점을 해결함으로써 빠른 검색 및 질의 속도를 보장할 수 있다. 기존 모델의 경우, 데이터 정보가 단일 테이블로 구성되어 있기 때문에 데이터에 대한 중복성이 높지만 제안된 모델의 경우에는 데이터의 속성을 분류함으로써 중복을 제거하여 데이터의 중복성이 낮다.

데이터 이해의 용이성이란 저장소의 데이터 구조가 쉽게 이해할 수 있도록 구성되어 있는 정도를 의미한다. Jena2의 관계형 데이터베이스 모델은 단일 테이블로 저장되기 때문에 데이터에 대해서 직관적으로 파악하기 어려우며 분류를 거친 후에 사용자가 이해할 수 있다. 하지만 제안되는 모델에서는 의미상으로 이해하기 편리하게 구조화 되어있기 때문에 사용자로 하여금 직관적으로 파악할 수 있도록 지원하며, 질의 모델링을 하는 개발자로 하여금 편리하게 모델링 할 수 있도록 제공된다. 또한 문서들에 대해서 의미상으로 분류되어 있기 때문에 다른 모델로 변환할 때에도 역시 용이하다. 마지막으로 제안한 저장 모델은 rdfs:subClassOf 조건과 rdfs:subProperty 조건의 계층 관계만을 별개의 테이블에 저장하여 관리함으로써 계층 구조에 대한 조건 판단이 요구되는 질의에 대한 처리 에서 Jena2보다 속도를 향상시킬 수 있다.

질의 모델링의 편의성이란 실제 모델을 설계하는 작업의 단계에서 모델링의 편의를 제공하는 것을 의미한다. 하나의 단일화된 테이블로 구성되어있는 Jena2와는 다르게 JeSPi는 OWL 온톨로지 데이터 집합을 클래스, 프로퍼티, 인스턴스로 구분하여 저장함으로써 특정 데이터들과의 관계성 추출 및 이해가 쉽다. 또한 데이터의 의미적 특성을 고려하여 저장하였기 때문에 모델을 설계하는 과정에서 저장구조에 대한 이해를 보다 용이하게 한다.

모델 변환의 용이성이란 이미 온톨로지 데이터 집합이 저장된 모델을 변화시킬 경우 용이한 정도를 의미한다. Jena2 저장소 모델은 하나의 단일화된 테이블에 데이터 정보들이 저장되기 때문에 저장 시에는 편리하지만 이를 변환하는 과정에서는 관계성 도출 및 추가적 연산에 대해서 비용이 소모된다. 하지만 JeSPi 모델의 경우 문서의 의미적 특성에 따라 관계적 표현까지 정규화 되어 저장되었기 때문에 추가적 비용을 줄일 수 있는 장점이 있다.

Jena2와 JeSPi 저장모델 사이에서 실제 실험을 통한 질의 응답 시간과 관계 대수를 수식화한 시뮬레이션을 통한 조회횟수의 결과에서 알 수 있듯이, 5개의 질의 패턴 모두에서 제안된 JeSPi 저장모델의 검색 속도가 Jena2 저장모델의 검색 속도보다 우수함을 확인할 수 있다.

5. 결론 및 향후 연구

본 논문에서는 Jena2의 관계형 데이터베이스 모델의 효율성을 높이기 위한 Plug-in 형식으로 새로운 저장 모델을 제안하였다. 그리고 기존 Jena2모델과 제안하는 모델에 대해 질의 응답 시간, 질의 결과 개수, 완전성 및 정확성 등의 항목으로 실험하고 결과를 비교 평가하였다.

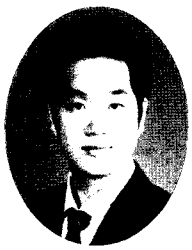
실험 및 시뮬레이션의 비교 평가 결과에서, 제안한 저장 모델이 Jena2보다 향상된 성능을 보였다. 이는 단순 연산과 조인 연산 과정에서 Jena2는 단일구조 전체에 대해서 연산이 이루어지지만 JeSPi는 의미 별로 정리된 테이블에서 필요한 테이블만을 선택하여 조인 연산이 이루어지기 때문이다. 또한 계층 구조에 대해서 불필요한 정보는 제거하고 계층관계의 정보만 저장한 테이블을 별도로 관리하기 때문에 역시 Jena2보다 나은 질의 처리 성능을 보였다.

향후 연구 과제는 다음과 같다. 먼저 기존의 Jena2와 JeSPi 저장 모델은 특정 환경에 종속된 저장 모델을 생성한다. 하지만 변화되는 다양한 환경과 어플리케이션에 따라서 저장 및 질의 응답 시간을 단축시킬 수 있는 저장 모델 생성 방법이 요구된다. 따라서 변화되는 다양한 환경에 따라 효율을 높일 수 있는 모델 생성기에 대한 연구와 함께 Sesame, DLDB 등과 같은 다양한 시스템들과의 비교 연구가 요구된다.

참 고 문 헌

1. Berners-Lee Tim, Hendler James, Lassila Ora (2001) The semanticweb Sci Am 284(5):34-43.
2. 이준하, 박용범, "온톨로지를 이용한 개념형 소프트웨어 프

로세스 데이터베이스 설계 및 구현", 한국정보처리학회, 한국정보처리학회논문지D, 제14D권 제2호, pp. 203-210, 2007.
3. 김종우, 김형도, 윤정희, 정현철, "기업간 비즈니스 프로세스 등록저장소를 위한 메타데이터 온톨로지 설계", 한국정보처리학회논문지D, 제14D권 제4호, pp. 435-446, 2007.
4. 민영근, 김인수, 이복주, "온톨로지를 이용한 의미 기반 정보 채움 시스템", 한국정보처리학회논문지B, 제14B권 제4호, pp. 295-302, 2007.
5. Hak Soo Kim, Hyun Seok Cha, Jungsun Kim, andJin Hyun Son, "Development of the Efficient OWL Document Management System for the Embedded Applications," Springer-Verlag, Lecture Notes In Computer Science (LNCS), Vol. 3597, pp. 75-84, Jul. 2005.
6. 김수경, 안기홍, "추론기반의 웹 온톨로지를 이용한 지능형 이미지 검색", 한국정보처리학회, 한국정보처리학회 학술대회논문집, 제27회 춘계학술발표대회, pp. 521-524, 2007.
7. 서희철, 김현기, 장명길, 개인용 미디어 관리를 위한 시맨틱웹 기반 온톨로지, 한국정보처리학회, 한국정보처리학회 학술대회논문집, 제27회 춘계학술발표대회, pp. 419-422, 2007.
8. RDF/XML Syntax Specification, http://www.w3.org/TR/rdf-syntax-grammar, Feb. 2004.
9. RDF Vocabulary Description Language 1.0: RDF Schema, http://www.w3.org/TR/rdf-schema, Feb. 2004.
10. DAML+OIL Reference Description W3C Note, http://www.w3.org/TR/daml+oil-reference, Dec. 2001.
11. OWL Web Ontology Language Reference, http://www.w3.org/TR/owl-ref, Feb. 2004.
12. Jena - A Semantic Web Framework for Java, http://jena.sourceforge.net
13. SPARQL Query Language for RDF, http://www.w3.org/TR/rdf-sparql-query, Mar. 2007.
14. Myung-Jae Park and Chin-Wan Chung, "Property Based OWL Storage Schema in Relational Databases," in Technical Report, CS/TR-2005-247, Div. of Computer Science, KAIST, Dec. 2005.
15. RDQL - A Query Language for RDF, http://www.w3.org/Submission/RDQL, Jan. 2004.
16. SWAT Projects- the Lehigh University Benchmark (LUBM), http://swat.cse.lehigh.edu/projects/lubm/
17. Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin, "An Evaluation of Knowledge Base Systems for Large OWL Datasets," Springer-Verlag, Lecture Notes in Computer Science (LNCS), Vol. LNCS 3298, pp. 274-288, 2004.
18 Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin, "LUBM: A Benchmark for OWL Knowledge Base Systems," Journal of Web Semantics, Vol. 3, No. 2, pp. 158-182, 2005.
19. Dongwon Jeong, Myounghoi Choi, Yang-Seung Jeon, Youn-Hee Han, Young-Sik Jeong, and and Sung-Kook Han, "A Novel Memory-Oriented OWL Storage System," Springer-Verlag, Lecture Notes in Computer Science (LNCS), Vol. LNCS 4331, pp. 542-549, Dec. 2006.
20. Dongwon Jeong, Myounghoi Choi, Yang-Seung Jeon, Youn-Hee Han, Laurence T. Yang, Young-Sik Jeong, and Sung-Kook Han, "Persistent Storage System for Efficient Management of OWL Web Ontology," Springer-Verlag, Lecture Notes in Computer Science (LNCS), Vol. LNCS 4611, pp. 1089-1097, Jul. 2007.
21. Jeen Broekstra, Arjohn Kampman, and Frank van Harmelen, "Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema," Lecture Notes In Computer Science (LNCS), Vol. 2342, pp. 54-68, June 2002.
22. 신희영, 정동원, 김진형, 백두권, Jena2 기반의 효율적인 OWL Ontology 관리를 위한 저장 모델, 2007 한국컴퓨터종합학술대회 논문집 제34권 제1호(C), 2007. 6



신 희 영 (shintul@software.korea.ac.kr)

2006 연세대학교 컴퓨터공학과(공학사)
2006～현재 고려대학교 컴퓨터학과 석사

관심분야 : 데이터베이스, XML, 시뮬레이션, 시맨틱 웹, 온톨로지

정 동 원 (djeong@kunsan.ac.kr)

1997 군산대학교 컴퓨터과학과(이학사)
1999 충북대학교 전산학과(이학석사)
2004 고려대학교 컴퓨터학과(이학박사)
1999～2000 ICU 부설 한국정보통신교육원(전임강사)
2000～2001 지구넷 부설 연구소(선임연구원)
2002～2004 TTA 표준화 위원회-데이터연구반 SG08.02(특별위원)
2004 고려대학교 정보통신기술연구소(연구조교수)
2005 펜실베니아 주립대학(PostDoc.)
2004～현재 TTA 메타데이터 프로젝트 그룹, PG 606(위원)
2006～현재 ISO/IEC JTC1/SC32 국내위원회(전문위원)
2005～현재 군산대학교 수학정보통계학부 정보통계학전공(교수)

관심분야 : 데이터베이스, 시맨틱 웹, 시맨틱 그리드, 시맨틱 GIS, 유비쿼터스 컴퓨팅, 보안

백 두 권 (baik@software.korea.ac.kr)

1974 고려대학교 수학과(이학사)
1977 고려대학교 대학원 산업공학과(공학석사)
1983 Wayne State Univ.(전산학석사)
1985 Wayne State Univ.(전산학 박사)
1986～현재 고려대학교 컴퓨터학과(교수)
1989～현재 한국정보과학회(이사/평의원/부회장)
1991～현재 한국시뮬레이션학회(이사/부회장/감사)
1991～현재 ISO/IEC JTC1/SC32 국내위원회(위원장)
2002～2004 고려대학교 정보통신대학(초대학장)
2002～2004 한국시뮬레이션학회(회장)
2001～현재 행자부 등록 (사)도산아카데미(원장)
2004～현재 정통부 등록 (사)한국정보처리학회(부회장)
2005～현재 정통부 등록 (사)한국정보과학회(부회장)
2005～현재 교육부 등록 (사)홍사단(공의원)
2005～현재 산자부 등록 (사)한국시뮬레이션학회(고문).

관심분야 : 데이터베이스, 소프트웨어 공학, 시뮬레이션, 메타데이터, 정보 통합, 정보 보호 등