# OWL 요소와 질의 패턴에 대한 관계 분석에 웹 온톨로지 저장소의 구현 및 평가

(Implementation and Evaluation of a Web Ontology Storage based on Relation Analysis of OWL Elements and Query Patterns)

정 동 원 † 최 명 회 †† 정 영 식 ††† 한 성 국 †††

(Dongwon Jeong) (Myounghoi Choi) (Young-Sik Jeong) (Sung-Kook Han)

**요 약** W3C에서 OWL을 웹 온톨로지 기술을 위한 표준으로 채택함으로써 OWL 웹 온톨로지를 효과적으로 저장할 수 있는 저장 모델에 대한 필요성이 제기되었다. 지금까지 Jena, Sesame, DLDB 등과 같은 관계형 모델 기반의 저장 시스템이 개발되었나 여전히 여러 가지 문제점을 지니며, 특히 비효율적인 질의 처리 성능을 제공한다. 질의 처리 성능이 저하되는 구조적인 문제점을 요약하면 다음과 같다. Jena의 경우, 정규화가 이루어지지 않은 매우 단순한 구조를 지니며 단일 테이블에 대부분의 정보를 저장한다. 이는 단순 검색은 물론 조인 연산이 요구되는 질의 처리시 불필요한 정보까지 비교함으로써 성능을 급격하게 저하시킨다. Sesame와 같은 저장소는 지나치게 정규화 된 구조를 지니기 때문에 질의 처리시 많은 조인 연산이 요구된다. 심지어 단순한 특정 클래스를 검색할 경우에도 많은 조인 연산이 요구된다. 이 논문에서는 이러한 기존 저장모델의 비정규화 혹은 지나친 정규화로 인해 발생하는 질의 처리 성능 저하 문제를 해결할 수 있는 저장 모델을 제안한다. 이를 위해 OWL 요소와 질의 패턴과의 관계를 분석하고 기존 저장 모델의 구조적인 문제점을 분석한다. 이러한 분석 결과를 통해 정의된 제안 모델은 적정 수준의 정규화 된 구조를 지니며 조인 연산이나 불필요한 정보에 대한 비교를 최소할 수 있는 구조를 제공한다. 질의 처리 성능 실험을 위해 LUBM 데이타 셋을 이용하며, 검색 대상 및 대상의 계층 관계를 고려한 질의 유형을 정의한다. 추가적으로, 제안된 저장모델의 데이타 손실 여부를 확인하기 위해 질의 결과의 정확성 및 완전성에 대해 실험하고 그 결과를 기술한다. 비교 평가 결과에서, 제안 모델이 기존 저장 모델보다 나은 성능을 보였다.

**키워드** : 웹 온톨로지, OWL, 저장소, 질의 응답 시간, 실험, 평가

***Abstract*** W3C has selected OWL as a standard for Web ontology description and a necessity of research on storage models that can store OWL ontologies effectively has been issued. Until now, relational model-based storage systems such as Jena, Sesame, and DLDB, have been developed, but there still remain several issues. Especially, they lead inefficient query processing performance. The structural problems of their low query processing performance are as follow: Jena has a simple structure which is not normalized and also stores most information in a single table. It exponentially decreases the performance because of comparison with unnecessary information for processing queries requiring join operations as well as simple search. The structures of storages (e.g., Sesame) have been completely normalized. Therefore it executes many join operations for query processing. The storages

† 종신회원 : 군산대학교 정보통계학과 교수
djeong@kunsan.ac.kr
†† 학생회원 : 군산대학교 정보통계학과
cmh775@kunsan.ac.kr
††† 종신회원 : 원광대학교 전기전자정보 공학부 교수
ysjeong@wonkwang.ac.kr
skhan@wonkwang.ac.kr
논문접수 : 2007년 8월 21일
심사완료 : 2008년 2월 11일


정보과학회논문지: 데이타베이스 제35권 제3호(2008.6)

require many join operations to find simply a specific class. This paper proposes a storage model to resolve the problems that the query processing performance is decreased because of non-normalization or complete normalization of the existing storages. To achieve this goal, we analyze the problems of existing storage models as well as relations of OWL elements and query patterns. The proposed model, defined with the analysis results, provides an optimal normalized structure to minimize join operations or unnecessary information comparison. For the experiment of query processing performance, a LUBM data sets are used and query patterns are defined considering search targets and their hierarchical relations. In addition, this paper conducts experiments on correctness and completeness of query results to verify data loss of the proposed model, and the results are described. With the comparative evaluation results, our proposal showed a better performance than the existing storage models.

**Key words** : Web Ontology, OWL, Storage, Query Response Time, Experiment, Evaluation

## 1. 서 론

최근 시맨틱 웹에 대한 관심이 높아지면서 W3C에서 개발한 웹 온톨로지 기술 언어, 온톨로지 질의 언어 및 관련 기술에 대한 연구가 활발하게 진행되고 있다[1]. 시맨틱 웹은 보다 지능화 된 웹을 구축하여 대량의 정보를 시스템 간에 자동으로 처리할 수 있도록 하기 위한 기반 기술이다. 이러한 시맨틱 웹을 구현하기 위해서는 컴퓨터가 이해할 수 있는 언어로 웹을 구성하여야 한다.

보다 지능화 된 웹을 구축하기 위한 대표적인 웹 온톨로지 기술 언어(Web Ontology Description Language)로는 RDF(Resource Description Framework)[2], RDFS(RDF Schema)[3], DAML(DARPA Agent Markup Language)+OIL(Ontology Interface Layer)[4] 및 OWL(Web Ontology Language)[5] 등이 있다. OWL은 RDFS와 DAML+OIL을 확장한 언어로서 추론 기능과 함께 풍부한 표현 기능을 제공한다. 따라서 OWL은 보다 풍부하게 자원들의 개념 및 관계를 표현할 수 있다[6].

이러한 OWL 특성에 기인하여, 온톨로지 편집기, 추론 엔진, 저장소, API 등 OWL과 관련된 매우 다양한 연구들이 진행되고 있다[7-18]. 특히, 방대한 양의 웹 온톨로지를 효율적으로 저장하고 관리할 수 있는 OWL 웹 온톨로지 저장소에 대한 연구가 활발하게 진행되고 있다.

OWL 웹 온톨로지 저장소는 크게 파일 시스템과 관계형 데이타베이스를 기반으로 한 접근 방법으로 분류할 수 있다. 파일 시스템을 기반으로 온톨로지를 저장할 경우, 온톨로지가 지니는 특성을 그대로 저장할 수 있다는 장점을 지닌다. 그러나 질의 처리시 많은 오버헤드를 지니며 대용량의 온톨로지 관리 측면에서 많은 문제점을 초래한다. 관계형 데이타베이스 기반의 OWL 웹 온톨로지 저장소는 관계형 데이타베이스의 장점을 그대로 수용하면서 현재 대부분의 데이타가 관계형 데이타베이스 내에서 관리되고 있다는 현실적인 상황에 적합한 접근 방법이다. 지금까지 Sesame[8,9], Jena[7,12,13], DLDB[10,11], OWLJessKB[17] 등과 같은 관계형 데이타베이스를 기반으로 한 저장소가 개발되어 왔다.

Jena는 단순한 형태의 저장소 모델을 제공하며, 무엇보다 사용자 정의 저장소 개발 및 추론 엔진 등의 개발이 가능한 다양한 API를 제공한다는 특징을 지닌다. Sesame는 웹 기반의 온톨로지 저장 시스템으로 사용자가 편리하게 온톨로지를 관리할 수 있다는 장점을 지닌다. DLDB는 Lehigh 대학의 SWAT 프로젝트의 일부로 개발된 저장소로서, 메모리 및 영구 저장소(관계형 데이타베이스) 기반의 온톨로지 저장 관리 기능을 제공한다. 추가적으로, 개발 시스템 및 비교 대상 시스템들과의 비교 평가를 위해 UBA 도구를 개발하여 제공하며, 이 도구를 이용하여 사용자는 대용량의 OWL 웹 온톨로지를 생성할 수 있다. OWLJessKB는 자바로 개발된 추론 엔진인 Jess를 기반으로 OWL 웹 온톨로지를 처리할 수 있도록 확장한 저장 시스템이다.

이 시스템들이 다양한 장점을 제공하지만 여전히 다음과 같은 문제점을 지닌다.

- 단일 테이블 기반의 데이타 관리: Jena의 경우, 단일 테이블에 모든 데이타를 트리플 형식으로 저장 및 관리한다. 이는 질의 처리시 불필요한 데이타에 대한 비교 연산을 요구하게 된다. 특히 조인이 요구되는 질의가 주어질 경우, 불필요한 많은 데이타들이 조인 연산에 이용되며 이는 성능을 급격하게 저하시킨다.
- 저장소 모델의 복잡성: Sesame의 경우, 온톨로지 저장을 위한 스키마 구조가 매우 복잡하게 정의되어 있다. 이는 질의 시 많은 조인 연산을 요구하게 되며 질의 처리 성능을 저하시키는 요인이 된다. 또한 사용자가 스키마 구조를 이해하는데 어려움이 있으며 데이타 간 상호 관계성을 파악하는데 많은 어려움이 따른다.

기존 시스템의 질의 처리 성능이 저하되는 가장 근본

적인 이유를 저장 모델의 구조적 특징에서 찾을 수 있다. Jena는 OWL 요소들간의 관계를 고려하지 않은 단순한 구조, 즉 정규화가 전혀 이루어지지 않은 구조를 지니기 때문이며, Sesame는 지나친 정규화로 인한 복잡한 구조를 제공하기 때문이다. 따라서 OWL 요소에 대한 관계 분석을 통해 적정 수준 정규화 된 구조를 정의한다면 보다 나은 질의 처리 성능을 제공할 수 있다.

웹 온톨로지의 관리에 있어 중요한 요소는 저장 시간, 질의 응답 시간 및 정확한 결과 생성이라 할 수 있으며, 실용성 측면에서 질의 응답 시간 및 질의 결과의 정확성 문제는 매우 중요한 평가 요소이다. 특히 질의 처리 성능은 대용량 온톨로지를 처리하고 이용함에 있어 가장 중요한 요소이며 이 논문에서도 질의 처리 성능을 개선할 수 있는 저장 모델에 초점을 둔다.

이 논문에서는 기존 시스템들의 성능 저하 문제를 개선할 수 있는 저장 모델을 제안한다. 제안 모델은 적정 수준의 정규화 된 구조를 제공함으로써 개선된 질의 처리 성능을 제공할 수 있도록 설계된다. 이를 위해 먼저 기존의 온톨로지 저장 모델에 대하여 분석한다. 기존 저장 모델에 대한 분석 결과와 OWL 요소와 질의 패턴에 대한 관계 분석을 통해 적정 수준의 정규화 구조를 정의한다. 정의한 저장 모델을 지원하는 저장소를 구현하고 기존 시스템과의 비교 평가를 수행한다. 기존 시스템과의 실질적인 비교 평가를 위해 다양한 크기의 온톨로지 셋을 이용한다. Lehigh 대학에서는 SWAT 프로젝트의 결과로서 개발한 시스템인 DLDB 시스템에 대한 평가를 위해 온톨로지 생성 툴인 UBA(Univ-Bench Artificial Data Generator)를 개발하였고 이를 통해 생성한 온톨로지 셋을 실험에 이용하였다. 이 논문에서도 UBA에 의해 생성된 데이타 셋을 이용하여 실험 및 평가를 수행한다.

이 논문의 구성은 다음과 같다. 제2장에서는 RDB 기반 시스템들의 저장 구조 및 문제점에 대하여 기술하고 제3장에서는 이 논문에서 제안하는 저장 구조의 특징, 구조, 주요 알고리즘 및 구현 결과에 대하여 서술한다. 제4장에서는 기존 시스템과 제안 시스템에 대한 실험 및 비교 평가 결과에 대하여 기술하며 제5장에서는 결론 및 향후 연구에 대하여 기술한다.

## 2. 관련 연구

이 장에서는 관계형 데이타베이스를 이용하여 웹 온톨로지를 저장하는 저장 시스템들의 저장 구조와 특징에 대하여 기술한다. 즉, Jena의 전체적인 시스템 구조, 실제 저장소의 관계형 모델 및 문제점에 대하여 기술하고 Sesame에서 제공하는 온톨로지 저장 모델과 문제점에 대하여 기술한다.

### 2.1 Jena

Jena2는 시맨틱 웹 응용프로그램을 구축하기 위한 자바 프레임워크로서 RDF, RDFS, DAML+OIL, OWL, 규칙 기반 추론엔진, 질의를 위한 RDQL 및 SPARQL 엔진 등을 포함하는 응용프로그램 개발 API이다. 또한 Jena2는 오픈 소스로서 HP 연구실에서 시맨틱 웹 연구의 일환으로 개발하였으며, 현재 2.5 버전까지 개발되어 있는 상태이다[8].

그림 1은 Jena2에서 기본적으로 제공하는 기본적인 저장 모델을 보여준다[13]. 그림 1에서, Jena_GiTi_Stmt 테이블은 온톨로지 데이타인 트리플들을 저장한다. 이 테이블은 Jena_Graph 테이블의 GraphID에 따라 테이블이 생성된다. 리터럴이나 URI 정보가 256 바이트 이상일 경우, 이를 Jena_Long_Lit 테이블과 Jena_Long_

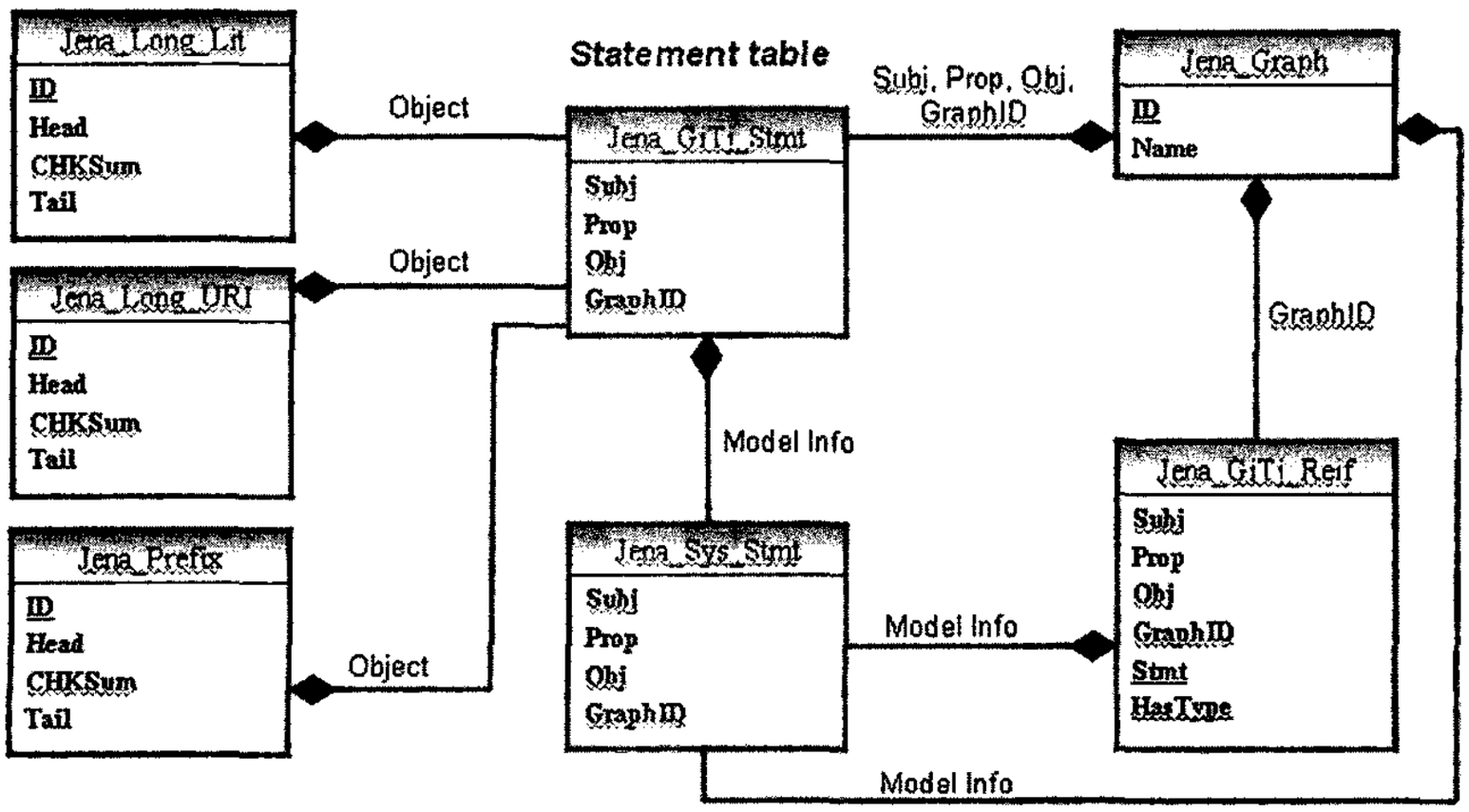



그림 1 Jena2 저장 스키마

URI 테이블에 각각 저장하여 이를 참조한다.

비록 Jena2가 많은 유용한 API를 제공하지만 특정 테이블에 모든 정보를 저장 관리하기 때문에 질의 처리 시 불필요한 데이타에 대한 비교 연산을 요구하게 된다. 특히 조인 연산이 요구되는 질의가 주어질 경우, 무관한 정보들이 조인연산에 이용되어 질의 처리 성능을 급격하게 저하시킨다.

### 2.2 Sesame

Sesame는 웹 온톨로지를 저장하고 이에 대한 질의 및 추론 연산을 수행할 수 있는 오픈 소스 자바 프레임워크이다[8,9]. Sesame는 클라이언트 응용프로그램이 HTTP 프로토콜을 통해서 통신할 수 있는 서버로서 사용된다. Sesame는 실제 저장소로서 다양한 RDBMS와의 연동을 위한 API를 자체적으로 지원한다.

그림 2는 관계형 데이타베이스를 위해 Sesame에서 제공하는 저장 구조를 보여준다. Sesame는 Jena와는 달리 보다 정규화 된 구조를 제공한다. TRIPLES 테이블에 트리플 정보를 저장하며 RDFS에서 제공하는 계층 구조를 별도의 두 테이블, subClassOf와 subPropertyOf에 저장하여 관리한다. 그러나 Sesame는 정보들간 복잡한 관계성으로 연결되어 있어 실제 질의 처리 시 많은 조인 연산을 요구함으로써 매우 낮은 성능을 보인다. 즉, 지나치게 OWL 요소를 분리된 테이블에 관리함으로써 많은 조인 연산이 요구되고 이에 따른 오버헤드를 지닌다.

## 3. 저장 모델 정의

이 장에서는 기존 온톨로지 저장 모델의 문제점을 보완하기 위해 제안하는 저장 모델의 개념, 구조 및 주요 알고리즘에 대하여 기술한다.

### 3.1 제안 모델의 특징

기존 저장 모델의 문제점은 구조적인 특성에서 유래하며, 이를 요약하면 (1) OWL 요소의 관계나 특성을 고려하지 않고 비정규화 된 저장 구조를 지니거나 (2) 혹은 지나치게 정규화 된 구조를 지님으로써 질의 처리시 많은 조인 연산을 요구한다. 이를 개선하기 위해 적정 수준의 정규화가 이루어진 저장 모델을 정의해야 한다.

그렇다면 유효 적정 수준의 정규화를 통해 구조화된 저장 구조를 어떻게 정의할 것인가? 이를 위해서는 먼저 OWL 요소와 OWL 온톨로지에 대한 질의 패턴에 따른 처리 성능이 어떠한 상호 연관성을 지니는지에 대한 분석이 요구된다.

먼저 OWL 웹 온톨로지에 대한 질의 패턴을 고려해 보자. OWL 요소에서 가장 중요한 요소는 Class와 Individual이다. Class는 객체의 개념을 정의하는 요소이며 Individual은 Class에 대한 인스턴스를 의미한다. 따라서 질의는 임의의 클래스를 검색하는 연산, 특정 클래스에 해당하는 Individual 요소의 값, 즉 인스턴스를 검색하는 연산 혹은 그 역의 관계로 정의할 수 있다. 또한 Class는 계층 구조를 이룰 수 있다. 따라서 특정 클

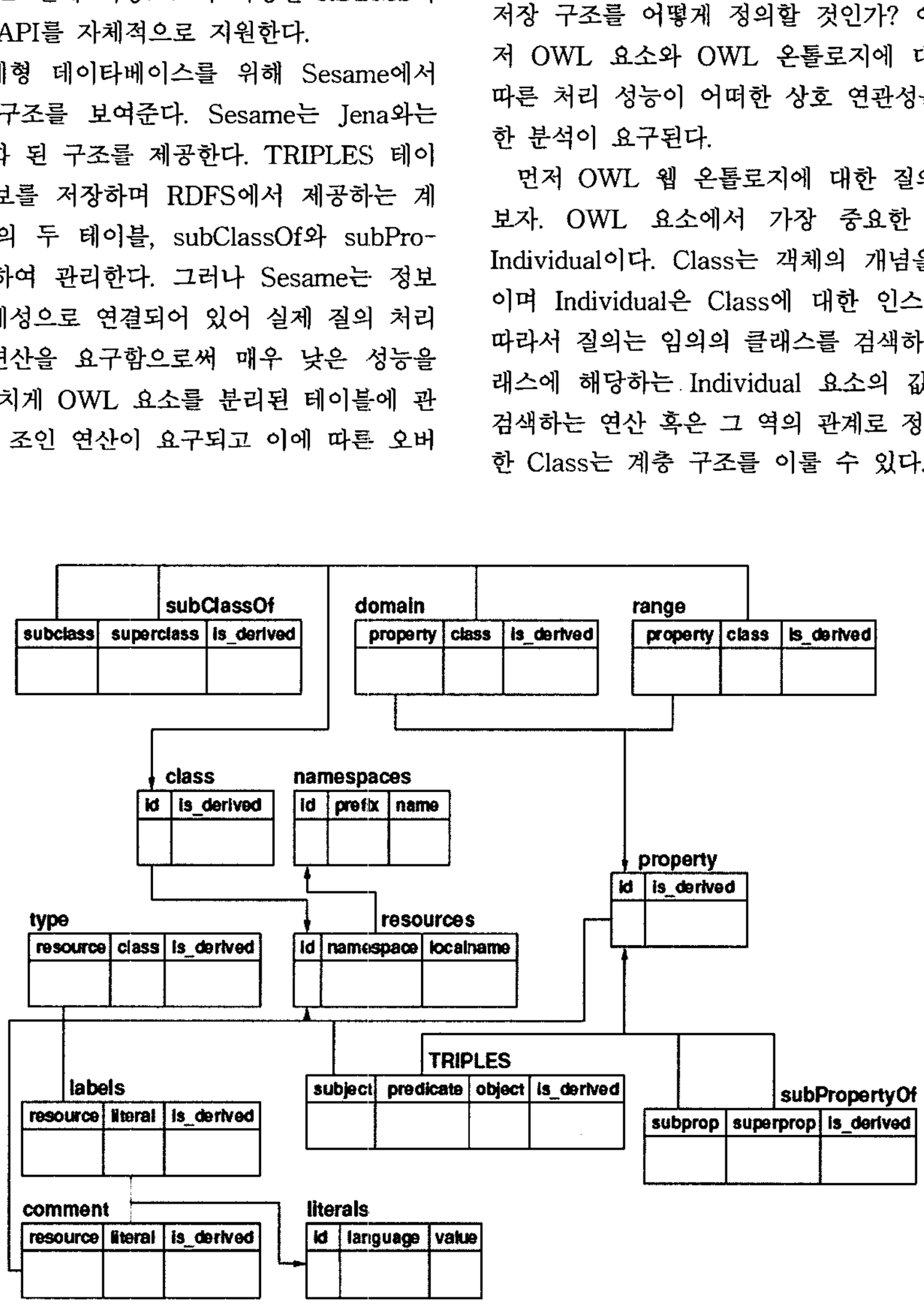



그림 2 Sesame의 구조

래스의 모든 하위 클래스를 검색하는 연산과 그 하위 클래스에 해당하는 모든 Individual의 값을 검색하는 연산이 가능하다. 이 때 OWL 요소인 Property의 값들 중에서 Class와 Class 간의 프로퍼티를 정의하는 값들만이 연산에 요구된다. 즉, Individual과 Individual 간의 관계성을 표현한 프로퍼티는 연산에 불필요한 값들이 된다. 따라서 이를 분리하여 관리할 필요가 있다.

이러한 분석 결과를 바탕으로 기존 저장 모델의 성능이 저하되는 구조적인 특성을 분석한 결과는 다음과 같다. 먼저, Jena는 256 바이트 이상의 URI와 리터럴에 대해서만 정규화를 제공하고 나머지 OWL 요소들에 대해서는 정규화가 이루어지지 않은 구조이다. 이는 앞선 질의 패턴에 대한 질의 처리시 불필요한 다른 OWL 요소의 값들이 비교 연산에 참여함으로써 처리 성능을 저하시키는 요인으로 작용한다.

Sesame는 정규화가 잘 이루어져 있다. 그러나 이러한 구조는 실제 질의 처리시 많은 조인 연산을 통해 질의 성능을 저하시킨다. 예를 들어, 특정 클래스에 대한 정보를 검색하는 질의 패턴을 고려해 보자. 이러한 단순 질의 처리를 위해 Sesame는 resources, class, domain, range 및 namespaces 테이블에 대한 조인 연산이 요구된다.

이 논문에서는 이러한 OWL 요소와 질의 유형과의 관계 및 기존 저장 모델의 구조적인 문제점에 대한 분석 결과를 토대로 새로운 저장 모델을 정의한다. 제안 모델의 구조적인 특성을 정리하면, 먼저 URI나 name-space와 같은 정보는 OWL 요소 값과 분리하지 않고 함께 관리하며, OWL에서 range와 domain은 모두 Class이므로 동일한 테이블에 함께 저장한다. Property는 Class 및 Individual을 위한 각각의 테이블로 분리하여 관리하며 OWL 요소에 대한 정의 속성도 각 해당 요소 정의 테이블에 함께 관리한다. 마지막으로, OWL 요소들 중에서 가장 중요한 Class와 Individual에 대한 값들을 분리하여 관리함으로써 단순 검색은 물론 두 요소 간 관계 비교가 요구되는 검색의 경우 효율적인 질의 처리가 가능하도록 저장 모델을 정의한다.

### 3.2 프레임워크

그림 3은 제안하는 저장소를 위한 프레임워크를 보여준다. 전체적으로, 먼저 사용자에 의해 입력된 OWL 온톨로지 데이타는 추출 및 분류 과정을 거쳐 메모리에 적재된다. 이 역할은 메모리 로딩 모듈에서 담당하며 이를 통해 메모리 기반 시스템 개발이 가능하다. 메모리 상에 저장된 데이타는 바로 사용자의 질의에 대한 결과를 생성하여 반환해 줄 수 있으며, 따라서 매우 빠른 처리 성능을 제공한다.

일단 데이타가 메모리에 적재되면, 적재된 데이타는

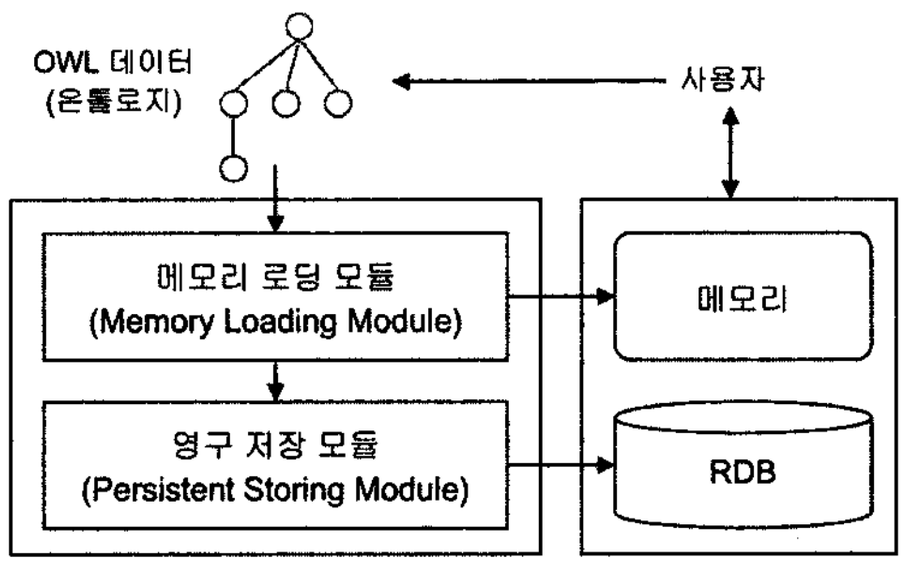



그림 3 제안 저장 시스템의 프레임워크

관계형 데이타베이스와 같은 저장소에 저장된다. 관계형 데이타베이스 스키마 구조는 이 논문의 가장 핵심적인 부분으로서, 저장 구조에 따라 질의 처리 성능이 크게 달라진다. 이에 대한 상세한 내용은 다음 절에서 다룬다.

그림 3에서 묘사하고 있는 구조는 대부분의 저장 시스템이 지니는 일반적인 구조라 할 수 있다. 그러나 처리 모듈, 즉 메모리 로딩 모듈이나 영구 저장 모듈의 구현 방법이나 이를 위한 알고리즘은 매우 다양하게 구체화된다. 또한 각 모듈은 저장 모델, 즉 물리적인 저장 구조에 따라 달라지게 된다.

메모리 로딩 모듈과 영구 저장 모듈과의 관계에서 고려할 사항 중 하나는 메모리에 로딩된 데이타의 물리적인 저장 시점이다[14]. 소규모의 온톨로지를 메모리에 로딩하여 데이타베이스에 저장할 경우, 모든 데이타를 로딩한 이후에 데이타베이스 연결을 통해 물리적으로 저장하는 방법이 보다 효율적이다. 그러나 대용량 온톨로지의 경우, 메모리 용량과 시스템의 성능에 따라 임계값을 설정하여 가장 최적의 시점에 물리적으로 저장해야 한다. 메모리에 로딩하는 데이타 용량을 너무 적게 설정하여 잦은 데이타베이스 연결이 이루어진다면 오히려 성능이 저하된다. 반면, 지나치게 큰 용량을 로딩하여 저장할 경우, 데이타베이스와의 연결에 따른 부하는 줄일 수 있으나 로딩 시스템의 성능을 급격하게 저하시켜 전체적인 성능을 저하시킬 수 있다. 그러나 이러한 로딩 과정은 일반적으로 오프라인 상태에서 이루어지는 경우가 대부분이며, 또한 이 논문의 초점과 다른 문제이므로 이 논문에서는 다루지 않는다.

### 3.3 저장소 구조 정의

이 절에서는 제안하는 저장 시스템과 저장소 구조에 대하여 정의한다. 제안하는 저장 시스템의 전체적인 구조는 정의 1과 같다[15].

**정의 1.** OWL 온톨로지를 저장 관리하기 위한 저장 시스템, PSSM (Persistent Store System Model)은 네 개의 컴포넌트로 구성된다.

PSSM = (IN, OP, MDM, PDM)

IN은 입력으로 주어지는 OWL 웹 문서 파일, 즉 OWL 웹 온톨로지를 의미한다. OP는 OWL 온톨로지를 분석하여 클래스와 인스턴스 등을 추출 및 분류하여 메모리 로딩한 후, 이를 다시 영구 저장소에 저장하는 연산들의 집합이다. MDM(Memory Data Model)은 OWL 온톨로지를 분석하여 메모리에 로드 시 생성되는 메모리의 데이타 구조를 의미한다. MDM 구조는 자바에서 제공하는 자료 구조를 이용하여 구현하였을 뿐, 정의 2에서 정의하고 있는 RDM 구조와 동일한 구조를 지닌다. 즉, MDM과 RDM은 동일한 메타 모델을 지닌다. 따라서 메모리 상의 데이타를 물리적인 저장소로 저장하는 연산을 보다 용이하게 구현할 수 있다.

PDM(Persistent Data Model)은 물리적인 저장 모델로서 이 논문의 가장 핵심적인 부분이며 PDM 구조는 아래와 같이 정의된다.

**정의 2.** PDM은 관계형 데이타베이스 모델을 표현한 것으로 다음과 같이 정의된다.

PDM = (RDM, Meta, Plain)

온톨로지 관리는 클래스나 인스턴스와 같은 실제 온톨로지 정보는 물론 온톨로지 타입이나 OWL 파일(Original OWL Documents) 등과 같은 메타 정보에 대한 관리를 요구한다. RDM(Relational Data Model)은 실제 온톨로지 정보를 저장하기 위한 관계형 스키마 구조를 의미한다. 온톨로지 타입과 같은 일부 정보를 제외한 실제 온톨로지 데이타는 모두 RDM에 저장된다. Meta는 온톨로지 타입 정보를 저장하기 위한 스키마들로 구성되며, 온톨로지에 대한 프로파일 정보, 정의된 prefix 정보 및 온톨로지에 대한 설명 등과 같은 부가 정보를 포함한다. 따라서 RDM 내의 실제 온톨로지 데이타 중 일부 혹은 전체가 온톨로지 타입에 사상된다.

그림 4는 이러한 OWL 데이타 집합을 관계형 데이타베이스에 저장하기 위한 PDM의 전체적인 구조를 보여준다. PDM은 세 영역으로 구성된다. 첫 번째 영역인 OWL Data 영역은 클래스, 인스턴스 등과 같은 정보를 저장하는 부분이다. 이 논문에서 사용하는 클래스와 인스턴스는 OWL 표준 문서에서의 Class와 Individual로서 각각 정의된다. 이 영역은 PDM의 RDM에 해당한다. Class_Definition과 Class_Property 및 Class_Axiom 릴레이션들은 클래스와 관련된 정보를 포함한다. Individual_Definition과 Individual_Property 및 Individual_Axiom 릴레이션들은 인스턴스 정보를 저장하기 위해 이용된다.

메타모델 구조에 대한 이해를 돕기 위해 일부 릴레이션에 대한 구체적인 구조에 대하여 기술한다. 먼저 Class_Definition은 클래스 정보를 저장하며 (className, classType, classDef)로 구성된 스키마를 지닌다. 각각 클래스 이름, 클래스 유형 및 클래스에 대한 정의를 나타낸다. 클래스 타입은 OWL에서 정의한 단순 클래스인지 논리 형식의 클래스인지를 구분하기 위한 속성이다. Class_Axiom은 클래스의 도메인(domin)과 영역(range) 및 관계를 나타내기 위한 구조로서 (domainClass, rangeClass, relation)으로 표현된다. 특히 relation은 도메인과 영역 간의 관계를 표현하는 속성으로 subClassOf, sameAs, disjointWith와 같은 관계를 나타낸다. OWL 요소들 중에서 중요한 또 다른 요소인 Individual은 Individual_Definition에 기본적인 정보를 저장한다. Individual은 클래스의 실제 객체인 인스턴스를 의미하며 따라서 해당 클래스가 무엇인지에 대한 정보를 반드시 포함해야 한다. 이 논문에서 IndividualDefinition은 (IndividualName, ClassName)으로 구성된다.

두 번째 영역인 Meta Information 영역은 온톨로지 타입에 대한 정보를 관리하는 영역으로 PDM의 Meta에 해당한다. 마지막으로, Original OWL 영역은 입력으로 주어진 OWL 문서 파일의 집합으로서 PDM의 Plain 컴포넌트에 해당한다.

### 3.4 온톨로지 저장 알고리즘 및 구현 결과

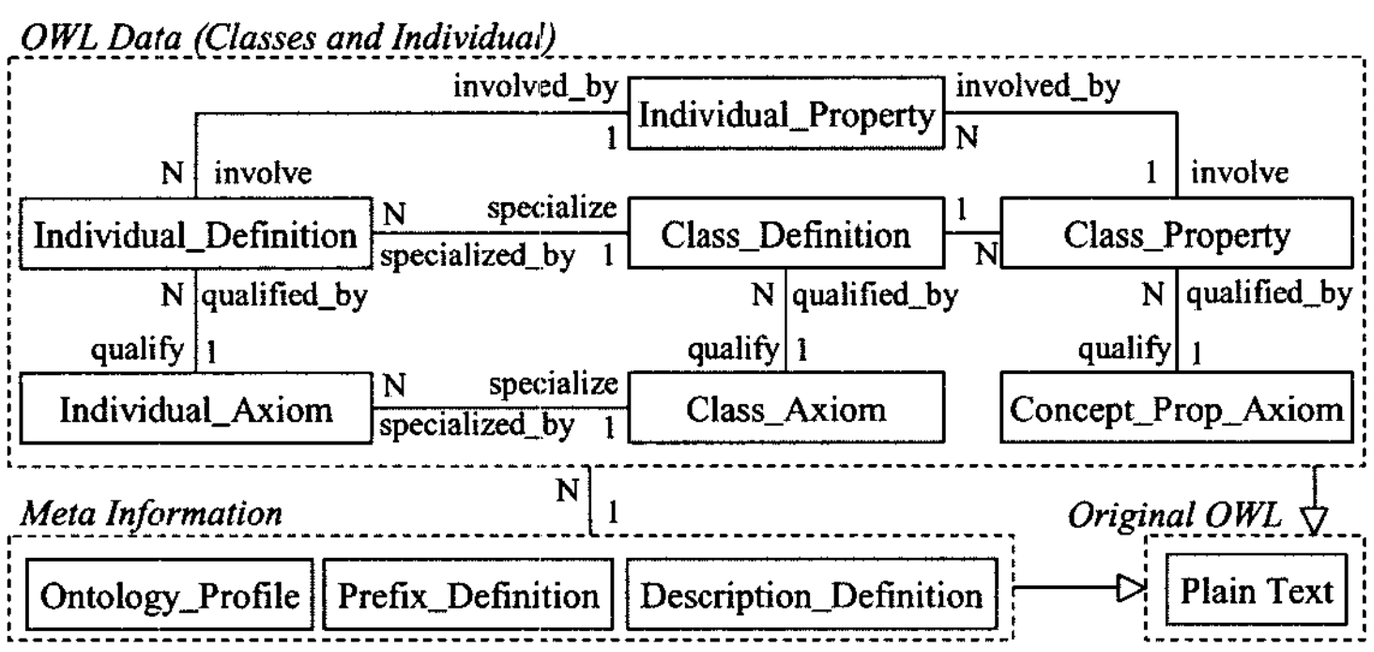



그림 4 관계형 스키마 구조를 위한 메타모델

이 논문에서 제안하는 저장 모델에 OWL 데이타를 저장하는 알고리즘은 크게 2 단계로 구성된다. 첫 번째 단계는 OWL 문서에 대한 분류 작업(1~13줄)이며, 두 번째 단계는 분류된 OWL 문서의 클래스와 인스턴스를 변환하여 연구 저장소에 저장하는 단계(14~24줄)이다. 전체적인 알고리즘은 다음과 같다.

**OWL 온톨로지 로딩을 위한 전체 알고리즘**

```
INPUT : I /**OWL document) */
OUTPUT : R          /**R indicates PDM */
START Algorithm {
 1: /** Reading a specific OWL document */
 2: owlText = read(I); //owlText: OWL stream
 3: boolean vr = validator.check(owlText);
 4: // OWL document is not valid, return an error
 5: if(!vr) {
 6:    ExceptionHandler.show(INVALID_ERROR);
 7:    return;
 8: }
 9:
10: /** Extracting and classifying OWL elemetnts */
11: EClassifier classifier new EClassifier(owlText);
12: //Creating a Mapper object
13: Mapper mapper = classifier.createMapper();
14: /** Loading OWL data (memory or database) */
15: if(MEM_TAG) { //Into both
16:    memManager.load(mapper);
17:    persistentManager.load(memManager);
18: }
19: else { // Into only persistent DB
20:    persistentManager.load(memManager);
21: }
22:
23: /** Generating statistical information */
24: StatisticalManager.createInfo(persistentManager);
}END Algorithm
```

알고리즘에서 중요한 부분은 EClassifier 객체를 생성 부분이다. EClassifier 객체는 입력된 OWL 문서에서 요소를 추출하여 분류하는 역할을 수행한다. 예를 들어, 아래와 같은 OWL 문서가 있다고 가정해 보자.

```
<owl:Class rdf:ID="Wine">
 <rdfs:subClassOf rdf:resource="&food;PotableLiquid"/>
 <rdfs:label xml:lang="en">wine</rdfs:label>
 <rdfs:label xml:lang="fr">vin</rdfs:label>
 ...
</owl:Class>
<owl:Class rdf:ID="Pasta">
 <rdfs:subClassOf rdf:resource="#EdibleThing" />
 ...
</owl:Class>
```

위와 같은 문서가 주어졌을 때, EClassifier 객체는 Wine이라는 클래스를 식별하고 rdf:ID와 같은 불필요한 내용을 제거한다. 이와 같이 분류된 집합들이 Mapper 객체에 전달되고 Mapper 객체는 이를 이용하여 메모리에 로드하거나 혹은 물리적으로 정의된 해당 테이블에 각 데이타 값을 저장하게 된다.

그림 5는 앞서 기술한 저장 모델과 저장 알고리즘에 따라 저장되는 과정과 저장된 결과를 보여준다. "Loading" 탭은 특정 OWL 문서를 선택하여 로딩하는 기능으로서, "View Processing Results"를 통해 처리되는 과정이 보여진다. "Statistical Info" 탭은 저장된 정보, 즉 클래스나 인스턴스 등의 개수를 보여주는 부가 기능을 제공한다. 마지막으로, "View Results"는 처리 과정을 통해 저장된 실제 데이타를 보여준다. 왼쪽은 주요 OWL 요소로서 실제 물리적인 테이블 명을 의미하며 오른쪽 화면은 해당 테이블, 즉 각 OWL 요소의 실제 값을 보여준다.

## 4. 평 가

이 장에서는 실험환경 및 실험결과에 대하여 기술한다. 이 논문에서의 주 평가 항목은 질의 응답 시간(Query Response Time)이며 아울러 질의 결과에 대한 정확성(Correctness)과 완전성(Completeness)에 대하여 평가한다.

### 4.1 실험환경

이 논문에서의 실험 대상 시스템은 Jena와 Sesame이다. 두 시스템 외에도 DLDB[10,11], OWLJessKB[17] 등과 같은 다양한 관계형 모델 기반 저장 시스템들이 제안되어 있으나 구체적인 저장 모델이나 API 혹은 소스가 공개되지 않아 비교 대상에서 제외한다. 특히, OWLJessKB는 기존 연구 결과에서, Sesame에 비해 매우 낮은 성능을 보였고 실험 대상으로서 무의미하므로 대상에서 제외한다.

실험을 위한 온톨로지는 Lehigh 대학의 온톨로지 생성 툴인 UBA(University-Bench Artificial Data Generator)를 이용하여 생성한다. UBA 툴은 DLDB 시스템의 평가를 위해 개발된 온톨로지 생성 툴로서 다양한 크기의 온톨로지 셋을 생성할 수 있다. 이 논문에서는 실험을 위해 LUBM(1,0), LUBM(2,0), LUBM(3,0) 데

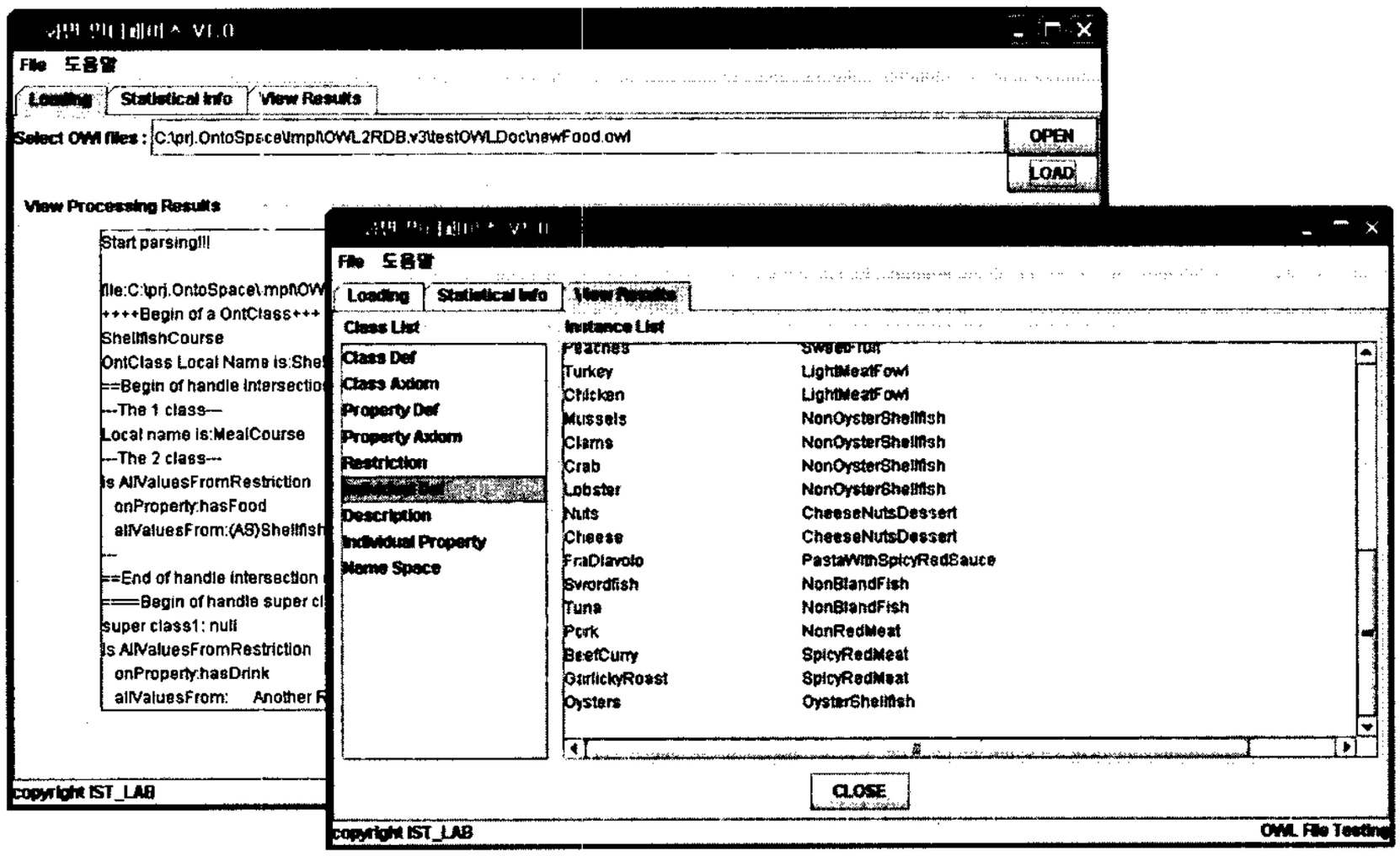



그림 5 구현 화면: 저장 과정 및 저장 결과

표 1 실험 데이타 셋 정보

| 항목 / 데이타 셋 | 파일 수 | 인스턴스 수 | 온톨로지 크기 |
|---|---|---|---|
| LUBM(1,0) | 15 | 103,074 | 8.02 MB |
| LUBM(2,0) | 34 | 237,142 | 18.40 MB |
| LUBM(3,0) | 50 | 348,006 | 27.10 MB |

표 2 질의 성능 평가 실험을 위한 질의 패턴 정의

| 질의 타입 | 질의 타입 설명 |
|---|---|
| C | 특정 클래스 검색 |
| CI | 특정 단일 클래스의 인스턴스 검색 |
| CH | 특정 클래스의 하위클래스 검색 |
| CHI | 하위클래스들까지의 인스턴스 검색 |

이타 셋을 이용한다. 각각의 데이타 셋은 15개, 34개, 50개의 OWL 문서 파일로 구성되어 있으며 각 데이타 셋에 대한 세부 정보는 표 1과 같다.

이 데이타 셋만을 이용하는 이유는 사용하는 데이타 규모만으로도 성능의 차이를 비교 평가할 수 있기 때문이다. 4.2절의 실험 결과에서 알 수 있지만, 위 데이타 셋만으로도 성능의 차이를 명확하게 확인할 수 있다. 추가적인 이유로서 질의 결과를 수동으로 파악할 때 발생할 수 있는 오류를 방지하고 이를 통해 실험 결과의 신뢰성을 제공하기 위함이다. 정확성과 완전성을 평가하기 위해서는 수동으로 예상되는 질의 결과를 확인해야 한다. 따라서 온톨로지의 크기가 지나치게 방대할 경우, 수동으로 정확한 결과 셋을 파악함에 있어서 정확성을 보장하기 어렵다. 결론적으로, 이 논문에서는 충분한 성능 차이를 확인할 수 있으면서 정확성과 완전성을 판단할 수 있는 적정 크기의 온톨로지를 이용한다.

이 논문에서 실험을 위한 시스템 환경은 다음과 같다. 추가적으로, 이 논문에서 비교 대상 시스템인 Sesame의 버전은 1.2.7 버전을 사용하고, Jena2의 경우 2.4 버전을 이용한다.

- CPU : Pentium D 3.0 GHz
- 메모리 크기(RAM) : 2GB
- 하드디스크 크기(HDD) : 250GB
- 플랫폼 : Windows XP Professional OS
- 자바 환경 : JAVA SDK 1.6.0
- 이용 DBMS : MySQL 4.1

질의 처리 성능 평가를 위한 질의 패턴은 매우 다양하게 정의될 수 있다. 이 논문에서는 질의 응답 시간을 평가하기 위해 네 개의 질의 타입을 정의한다(표 2). 표 2에서, 첫 번째 질의 타입(C)은 클래스 정보를 검색하는 단순 질의로서 하나의 OWL 요소만을 대상으로 검색한다. 이를 통해 Jena와 같이 모든 정보를 하나의 테이블에 포함하는 구조와 Sesame와 같이 여러 테이블에 분리하여 관리하는 저장소와의 성능을 비교 평가할 수 있다.

나머지 질의 타입은 두 개 이상의 OWL 요소가 관여하는 질의에 대한 처리 성능을 실험하기 위한 것으로, Jena와 Sesame는 물론 제안 저장 모델 또한 조인 연산이 요구된다. 이를 통해 비정규화가 이루어지지 않은 Jena 구조의 성능과 지나치게 정규화 된 구조를 지니는 Sesame와의 비교 평가가 가능하다. 특히 질의 타입, CH 및 CHI의 경우, 클래스의 계층 구조를 파악하여 결과를 반환하는 질의로서, sameAs, disjointWith와 같이 클래스와 클래스 간의 관계를 이용한 질의들에 대한 성능 평가를 수행하기 위한 질의 패턴이다.

이 논문에서, 실험을 위해 표 2에서 정의한 네 가지 질의 패턴만을 이용하는 이유는 다음과 같다. 먼저 대부분의 질의가 클래스나 인스턴스를 대상으로 이루어진다는 측면을 고려한다. 이에 따라 특정 클래스를 검색하는 질의와 인스턴스를 검색하는 질의로 분류한다. 또한 클래스 계층 구조를 고려하여 클래스와 인스턴스를 검색하는 질의 형태를 패턴 분류의 요인에 추가한다. 이는 단순히 임의의 클래스에 속하는 인스턴스를 검색하는 질의 패턴이 이용될 수 있으나, 특정 클래스의 모든 하위 클래스들에 속하는 인스턴스를 대상으로 하는 질의 패턴이 이용될 수 있기 때문이다.

다음 표 3은 표 2에서 정의한 네 가지 타입을 위해 정의한 실제 질의를 보여준다.

표 3 평가를 위한 실제 질의

| 질의(질의 타입) | 질의 설명 |
|---|---|
| Q1(C) | 모든 클래스를 검색 |
| Q2(CI) | 모든 대학원생을 검색 |
| Q3(CH) | 모든 교수 클래스를 검색 |
| Q4(CHI) | 모든 교수들을 검색 |

표 3에서, Q1(C)은 온톨로지에 정의되어 있는 모든 클래스 정보를 검색하는 질의이며, Q2(CI)는 GraduateStudent 클래스에 해당하는 모든 인스턴스를 검색하는 질의이다. 이 질의의 경우, 먼저 클래스 정보를 검색하고 해당 클래스와 연관되어 있는 인스턴스 정보를 검색하기 위한 조인 연산이 이루어진다. Q3(CH)는 클래스 계층 구조를 고려한 질의로서, 모든 교수에 해당 하위 클래스를 검색하는 질의이다. 따라서 Professor 클래스의 하위 클래스인 AssistantProfessor, AssociateProfessor 등의 모든 클래스를 검색하게 된다. Q4(CHI)는 교수, 즉 Professor 클래스의 모든 하위 클래스를 검색하고 각 클래스의 인스턴스를 검색하는 질의이다.

### 4.2 실험결과

이 논문에서의 주 비교 평가 항목은 질의 응답 시간으로서 이에 대한 실험 결과는 표 4와 같다.

그림 6은 각 시스템의 질의 응답 시간에 대한 실험 결과(표 4)를 그래프 모델로 표현한 것이다. 각 질의에 대한 비교 대상 시스템들의 성능 실험에 대한 평가 결과는 다음과 같다.

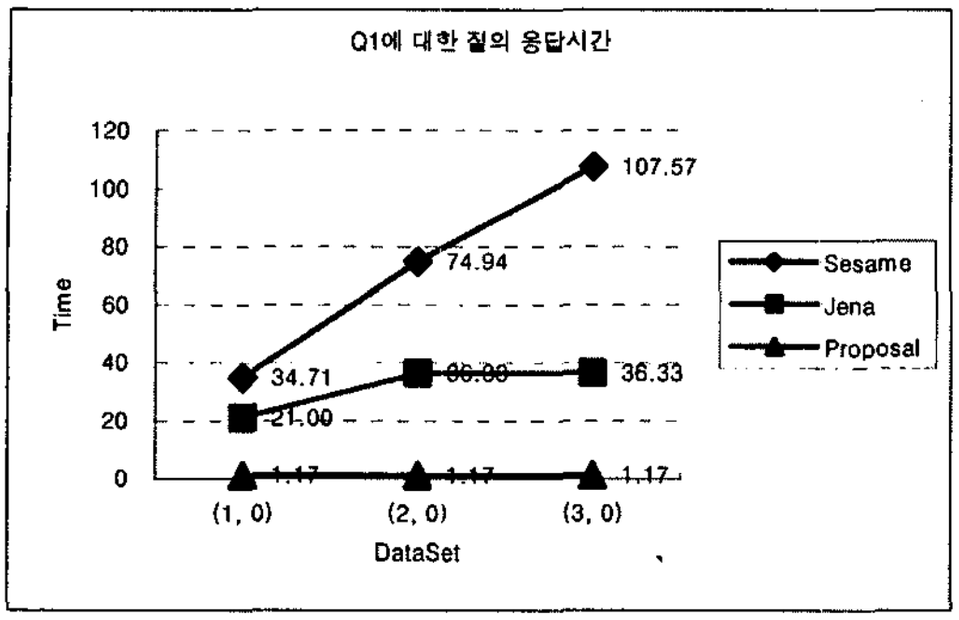
그림 6(a)의 Q1을 이용한 실험 결과에서, Sesame는 테이블을 검색하기 위해 여러 테이블을 이용하므로 다수의 조인 연산이 필요하다. 이러한 이유로 다른 시스템보다 검색 시간이 현저하게 증가한다. Jena는 하나의 테이블에 모든 인스턴스 정보를 저장하기 때문에 모든 클래스 검색 시 테이블에 저장된 모든 인스턴스를 검색하게 되며, 불필요한 데이타까지 비교한다. 제안 모델은 하나의 테이블에 모든 클래스 정보를 저장하므로 단일 테이블만을 검색한다. 따라서 조인 연산을 요구하는 다른 시스템에 비해 나은 성능을 보인다. LUBM(1,0) 셋에서 제안 시스템의 응답 시간(1.17ms)을 1로 보았을 때, Jena는 약 20배, Sesame는 약 30배 이상의 응답 시간의 차이를 보이며 데이타 셋의 증가에 따라 응답 시간의 차이가 증가하는 것을 알 수 있다. LUBM(3,0) 셋의 경우, 제안 시스템과 Sesame의 응답 시간의 차이가 약 100배 이상임을 알 수 있다.

그림 6(b)의 특정 단일 클래스의 모든 인스턴스를 추출하는 Q2 질의에 대해, Sesame는 특정 클래스를 검색 시 여러 테이블을 참조하게 되고 이를 통해 검색된 특정 클래스에 해당하는 인스턴스를 검색하기 위해 다시 여러 테이블을 참조한다. 따라서 여러 번의 조인 연산을 수행하게 되며 데이타 셋이 증가할수록 응답 시간의 선형적인 증가현상을 보인다. Jena는 클래스와 인스턴스 정보가 동일 테이블에 존재하기 때문에 조인 연산시 불필요한 클래스 정보에 대해서도 비교 연산이 수행된다. 따라서 데이타 셋이 증가할수록 응답 시간의 선형적인 증가를 보인다. 특정 단일 클래스를 검색하고 해당 인스턴스를 검색하기 위해 참여하는 클래스와 인스턴스 수가 Jena보다 적고 Sesame보다 조인 연산이 적기 때문에 검색 시간이 단축된다. LUBM(1,0) 셋에서, Jena는 약 20배, Sesame는 약 25배 이상의 응답 시간을 요구한다.

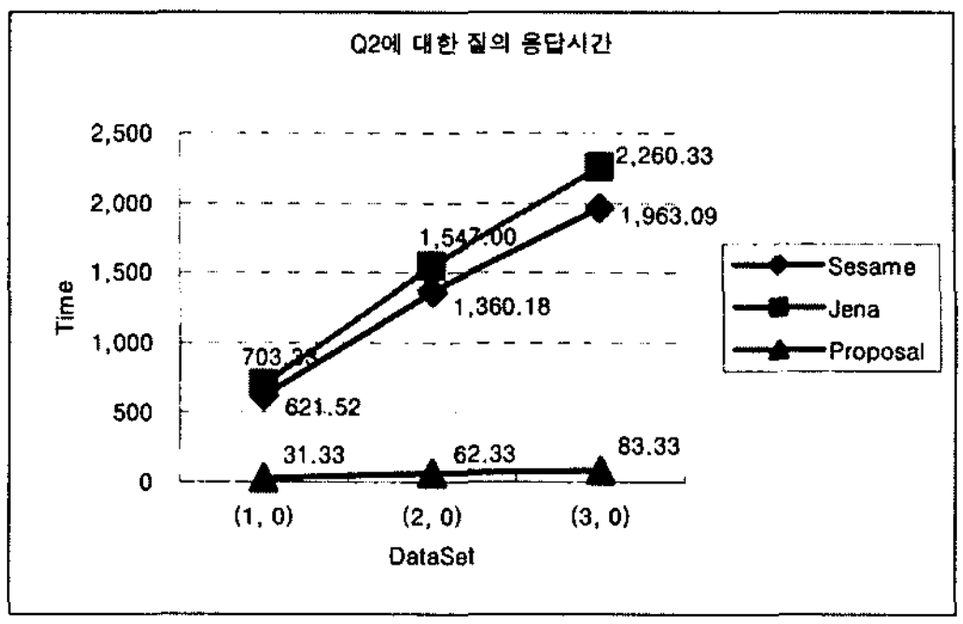
표 4 질의 응답 시간에 대한 실험 결과 (단위: ms)

| 질의 / 데이타 | Q1(C) | | | Q2(CI) | | | Q3(CH) | | | Q4(CHI) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | S | J | P | S | J | P | S | J | P | S | J | P |
| LUBM (1, 0) | 34.00 | 21.00 | 1.17 | 621.00 | 703.00 | 31.33 | 36.50 | 36.67 | 0.43 | 1,267.00 | 620.00 | 510.00 |
| LUBM (2, 0) | 74.00 | 36.00 | 1.17 | 1,360.00 | 1,547.00 | 62.33 | 76.60 | 57.67 | 0.43 | 2,779.00 | 1,312.00 | 1,151.00 |
| LUBM (3, 0) | 107.00 | 36.00 | 1.17 | 1,963.00 | 2,260.00 | 83.33 | 111.00 | 62.67 | 0.43 | 4,036.00 | 1,953.00 | 1,682.00 |

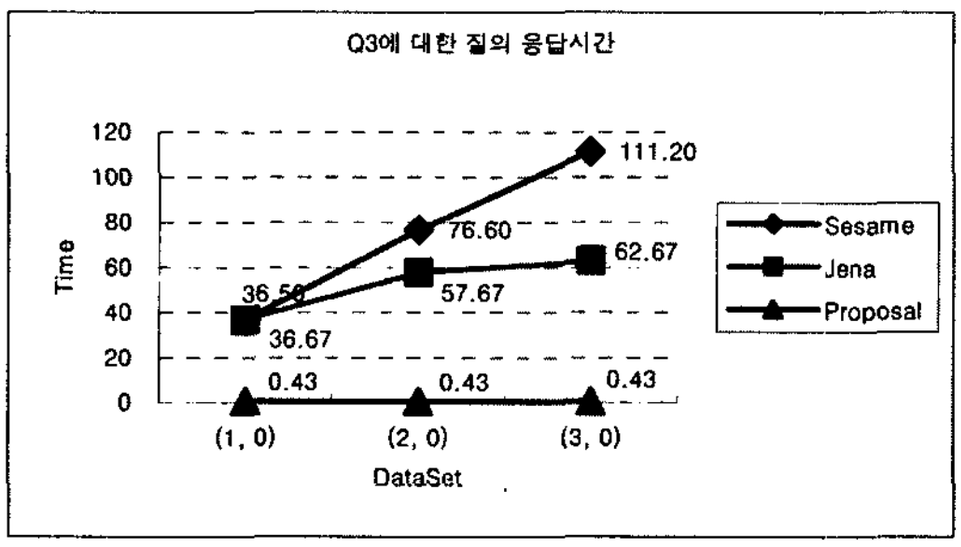
· S : Sesame, · J : Jena, · P : 제안 시스템



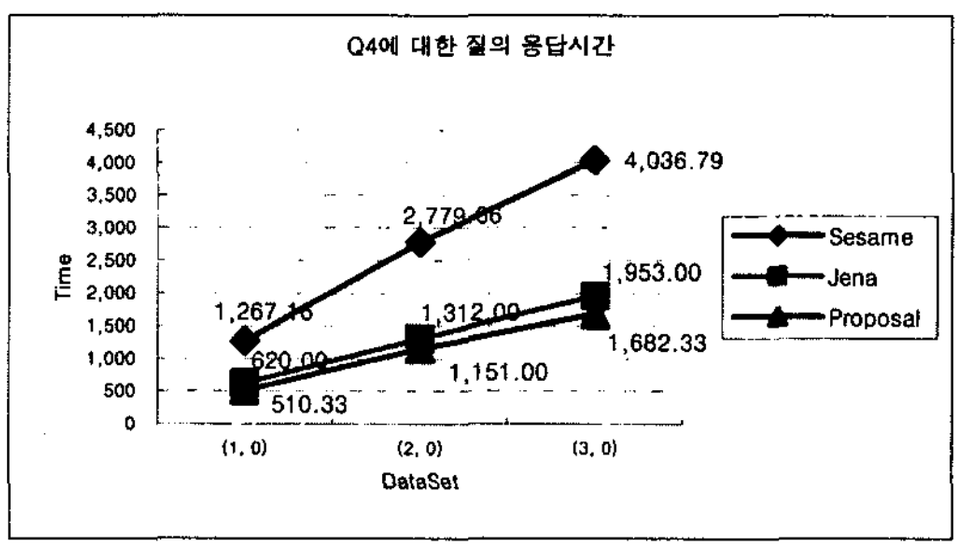

(a) Q1에 대한 응답시간




(b) Q2에 대한 응답시간




(c) Q3에 대한 응답시간




(d) Q4에 대한 응답시간

그림 6 각 질의 별 실험 결과 그래프

특정 클래스의 모든 하위 클래스를 추출하는 Q3의 질의 처리 결과인 그림 6(c)에서, Sesame는 특정 클래스 검색은 해당 하위 클래스 검색을 위해 많은 조인 연산을 요구한다. Jena는 특정 클래스 검색한 후, 해당 클래스의 하위 클래스를 검색하기 위하여 Prof 필드의 값이 "subClassOf"인 클래스를 모두 검색한다. 이러한 과정을 반복함으로써 최종 결과를 얻게 되며 이 과정에서 비교 연산의 증가로 인해 질의 처리 성능이 저하된다.

마지막으로, 특정 클래스의 하위 클래스들에 해당하는 모든 인스턴스를 추출하는 Q4의 질의 결과인 그림 6(d)에서, Sesame는 Q3에서 추출한 결과의 instanceOf 관계에 있는 모든 정보를 검색하여 최종 결과를 생성하며 데이타 셋이 증가할수록 시간이 두 배씩 증가한다. Jena는 Q2와 같이 자신의 테이블에서 subClassOf 관계를 검색하고 다시 그에 대한 인스턴스를 검색하기 때문에 Q2에서와 같이 데이타 셋이 증가할수록 질의 응답 시간이 증가하게 된다. 제안 시스템은 Q3의 결과와 인스턴스만을 저장하는 테이블과의 조인 연산을 수행함으로써 최종 결과를 반환하며, Sesame나 Jena에 비해 짧은 응답 시간을 요구한다. LUBM(1,0) 셋에서, Jena는 약 1.2배, Sesame는 약 2배 이상이며, LUBM(3,0) 셋의 경우에는 제안 시스템과 Sesame의 응답 시간의 차이가 약 2.5배 이상이다.

결론적으로, Jena2는 OWL 요소 관계를 고려하지 않고 모든 데이타를 하나의 테이블에 저장하는 비정규화된 구조를 지니기 때문에 질의 처리시 관련 없는 정보까지 참조하게 되며 성능을 저하시킨다. Sesame의 경우, OWL 요소 관계를 고려하여 정규화 된 구조를 제공하지만 질의 패턴과의 관계를 고려하지 않음으로써 질의 처리시 많은 조인 연산이 요구되어 성능을 저하시키는 요인으로 작용했음을 알 수 있다. 반면 제안 모델은 OWL 요소의 관계를 고려하여 구조화 하고 질의 패턴을 고려하여 정규화함으로써 단순 질의는 물론 조인 연산이 요구되는 질의에 대한 처리 측면에서 보다 나은 성능을 보였다.

추가적으로, 제안 모델과 기존의 모델과의 질의 결과에 대한 정확성 및 완전성을 평가한다. 표 5는 각 질의 별 실제 반환되어야 하는 결과 개수를 보여주며, OWL 파일을 수동으로 분석하여 얻어진 결과이다.

표 5 각 질의 별 수동 분석 결과(반환되어야 하는 튜플 수)

단위: 개

| Query / Dataset | Q1(C) | Q2(CI) | Q3(CH) | Q4(CHI) |
|---|---|---|---|---|
| LUBM(1, 0) | 43 | 1,874 | 6 | 447 |
| LUBM(2, 0) | 43 | 4,319 | 6 | 1,032 |
| LUBM(3, 0) | 43 | 6,390 | 6 | 1,507 |

표 6 질의에 따라 반환 된 튜플 수 (단위: 개)

| Query / Data sets | Q1(C) | | | Q2(CI) | | | Q3(CH) | | | Q4(CHI) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | S | J | P | S | J | P | S | J | P | S | J | P |
| LUBM(1, 0) | 28 | 43 | 43 | 1,874 | 1,874 | 1,874 | 3 | 6 | 6 | 447 | 447 | 447 |
| LUBM(2, 0) | 28 | 43 | 43 | 4,319 | 4,319 | 4,319 | 3 | 6 | 6 | 1,032 | 1,032 | 1,032 |
| LUBM(3, 0) | 28 | 43 | 43 | 6,390 | 6,390 | 6,390 | 3 | 6 | 6 | 1,507 | 1,507 | 1,507 |

· S : Sesame, · J : Jena, · P : 제안 시스템

표 6은 비교 평가를 위해 정의한 질의 별로 각 시스템이 결과로서 반환한 튜플 수를 보여준다.

정확성은 질의 결과에 대한 오차율을 의미하며 완전성은 손실률, 즉 검색되어야만 하는 모든 데이타를 반환하는지를 의미한다. 표 7과 표 8은 각 시스템의 정확성과 완전성을 백분율로 표현한 것으로 다음과 같은 공식에 의하여 계산된다.

- 정확성 계산식:

$$\frac{\text{Retrieved \& Correct Results}}{\text{All Results Retrieved by Each System}} \times 100$$

- 완전성 계산식:

$$\frac{\text{Retrieved \& Correct Results}}{\text{All Results by Manual Retrival}} \times 100$$

정확성은 검색된 전체 결과 개수를 기준으로 올바른 결과의 비율을 의미한다. 완전성은 실제 반환되어야 하는 전체 결과 개수를 기준으로 각 시스템에 의해 검색된 결과들 중에서 올바른 결과만을 고려하여 백분율로 환산하게 된다. 위 식에서, 검색한 정확한 개수는 표 6에서 질의에 의해 검색된 인스턴스 중에서 표 5에서 검색된 인스턴스와 일치하는 개수이다. 수동 검색한 전체 개수는 표 5에서 검색된 개수이고, 검색한 전체 개수는 표 6에서 검색된 인스턴스 개수이다.

표 7의 Q1에 대해서 Jena와 제안 시스템은 100%의 정확성을 보이는 반면, Sesame는 33%정도의 정확성을 보인다. 또한 Q3의 경우, Jena와 제안 시스템과는 달리 Sesame는 50%의 정확성을 보인다. 이러한 결과는 각각의 시스템이 OWL 파일을 얼마나 정확하게 분석하여 저장하는지를 확인할 수 있도록 해 주며, 이는 질의 응답 시간에 대한 실험 결과의 신뢰성을 제공한다.

표 8의 완전성 평가에서, 대부분은 요구하는 모든 결과를 반환한다. 그러나 Q1에 대해서 Jena와 제안 시스템이 100%로 모든 결과를 검색하여 반환해 주는 반면, Sesame는 50%의 결과만을 반환해 준다.

결과적으로, 지금까지 기술한 실험 결과를 바탕으로, 제안 저장 모델이 기존 저장 모델에 비해 질의 처리 성능, 정확성 및 완전성이 우수함을 알 수 있다.

## 5. 결론 및 향후 연구

이 논문에서는 기존 관계형 데이타베이스 기반 저장 시스템의 문제점 분석하고 이를 개선하기 위한 저장 모델을 제안하였다. 이를 위해 기존 저장 모델의 구조적인 문제점 및 OWL 요소와 질의 패턴과의 관계를 분석하였다. 마지막으로, 기존 저장 모델과의 정량적인 평가를 위한 실험 및 평가를 실시하였다. 실험을 위한 평가 항목은 질의 응답 시간, 정확성 및 완전성이며 이를 통해 각 시스템의 성능은 물론 질의 처리 결과에 대한 신뢰성에 대하여 평가하였다. 실험 결과에서, 이 논문에서 제안한 저장 모델은 Jena 및 Sesame와 비교하여 나은 질의 처리 성능을 보였다. 또한 정확성과 완전성 측면에서도 신뢰할 수 있는 결과를 보였다. 특히, 질의 응답 시간에 대한 성능 평가 결과에서, 기존 시스템들은 중첩된 조인 연산과 단일 테이블에서의 조인 연산으로 인해

표 7 정확성 평가 결과 (단위: %)

| Query / Dataset | Q1(C) | | | Q2(CI) | | | Q3(CH) | | | Q4(CHI) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | S | J | P | S | J | P | S | J | P | S | J | P |
| LUBM(1, 0) | 33 | 100 | 100 | 100 | 100 | 100 | 50 | 100 | 100 | 100 | 100 | 100 |
| LUBM(2, 0) | 33 | 100 | 100 | 100 | 100 | 100 | 50 | 100 | 100 | 100 | 100 | 100 |
| LUBM(3, 0) | 33 | 100 | 100 | 100 | 100 | 100 | 50 | 100 | 100 | 100 | 100 | 100 |

· S : Sesame, · J : Jena, · P : 제안 시스템

표 8 완전성 평가 결과 (단위: %)

| Query / Dataset | Q1(C) | | | Q2(CI) | | | Q3(CH) | | | Q4(CHI) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | S | J | P | S | J | P | S | J | P | S | J | P |
| LUBM(1, 0) | 50 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| LUBM(2, 0) | 50 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| LUBM(3, 0) | 50 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |

· S : Sesame, · J : Jena, · P : 제안 시스템

제안 저장 모델에 비해 낮은 성능을 보였다. 결론적으로, OWL 요소와 질의 패턴 분석을 통해 적정 수준 정규화 된 제안 모델이 구조적으로 보다 나은 모델임을 알 수 있다.

향후 연구 과제로서, 메모리에 저장된 데이타를 영구 저장소로 저장하는 최적 시점에 대한 연구가 요구되며 이는 보다 향상된 온톨로지 저장 성능을 제공할 것이다. 추가적으로, 제안 저장 시스템의 활용성 재고를 위해 다른 저장 모델과의 연계 방법에 대한 연구도 요구된다.

## 참 고 문 헌


[1] Berners-Lee, T., Hendler, J., and Lassila, O., "The Semantic Web," Scientific American, May 2001.

[2] RDF/XML Syntax Specification, http://www.w3.org/TR/rdf-syntax-grammar, Feb. 2004.

[3] RDF Vocabulary Description Language 1.0: RDF Schema, http://www.w3.org/TR/rdf-schema, February 2004.

[4] DAML+OIL Reference Description W3C Note, http://www.w3.org/TR/daml+oil-reference, December 2001.

[5] OWL (Web Ontology Language), http://www.w3.org/2004/OWL/, 2007.

[6] W3C, http://www.w3.org/, 2007.

[7] Jena2, http://jena.sourceforge.net/, 2007.

[8] Sesame, http://www.openrdf.org/, 2007.

[9] Broekstra, J., Kampman, A., and Harmelen, F.v., "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.

[10] Pan, Z. and Heflin J., "DLDB: Extending Relational Databases to Support Semantic Web Queries," In Workshop on Practical and Scaleable Semantic Web Systems, The Second International Semantic Web conference (ISWC2003), 2003.

[11] SWAT Projects - the Lehigh University Benchmark (LUBM), http://swat.cse.lehigh.edu/projects/lubm/

[12] Jena Inference Engine, http://Jena.sourceforge.net/inference/index.html, 2007.

[13] SourceForge.net, "Jena2 Database Interface - Database Layout," November 2004.

[14] Jeong, D., Choi, M., Jeon, Y.-S., Han Y.-H., Yang, L.T., Jeong, Y.-S., and Han, S.-K., "Persistent Storage System for Efficient Management of OWL Web Ontology," Springer-Verlag, Lecture Notes in Computer Science (LNCS),Vol. LNCS 4611, pp. 1089-1097, July 2007.

[15] Jeong, D., Choi, Jeon, Y.-S., Han Y.-H., Jeong, Y.-S., and Han, S.-K., "A Novel Memory-Oriented OWL Storage System," Springer-Verlag, Lecture Notes in Computer Science (LNCS), Vol. LNCS 4331, pp. 542-549, December 2006.

[16] Protégé, http://protege.stanford.edu/, 2007.

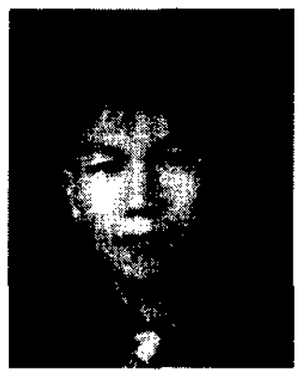
[17] OWLJessKB, http://edge.cs.drexel.edu/assemblies/software/owljesskb/, 2007.

[18] 3store, http://www.aktors.org/technologies/3store/, 2007.



**정 동 원**

정보과학회논문지 : 데이타베이스
제 35 권 제 2 호 참조

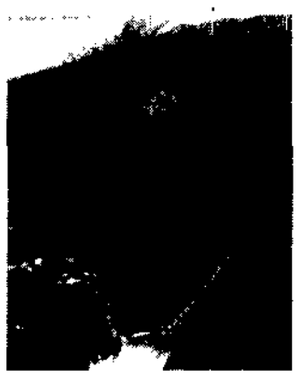


**최 명 희**

2006년 군산대학교 정보통계학과 졸업(학사). 2008년 군산대학교 대학원 정보통계학과 졸업(석사). 2007년~현재 네이버시스템㈜ 연구원. 관심분야는 시맨틱 웹, LBS, 정보보안, DB

**정 영 식**

1987년 고려대학교 수학과 졸업(학사). 1989년 고려대학교 대학원 전산학과 졸업(석사). 1993년 고려대학교 대학원 전산학과 졸업(박사). 1997년 미시간 주립대학교 객원교수. 2004년 웨인 주립대학교 객원교수. 1993년~현재 원광대학교 전기전자 및 정보공학부 교수. 관심분야는 그리드컴퓨팅, 시맨틱그리드, 모바일 서비스 솔루션, 분산병렬시스템



**한 성 국**

1979년 인하대학교 전자공학과(공학사) 1983년 인하대학교 전자공학과(공학석사). 1988년 인하대학교 전자공학과(공학박사). 1990년~1992년 University of Pennsylvania 방문교수. 2003년~2004년 University of Innsbruck와 DERI 연구교수. 1984년~현재 원광대학교 전기전자 및 정보공학부 교수. 관심분야는 온톨로지, 시맨틱 웹 서비스, 지식표현 및 추론, 자연언어처리