



특허청구의 범위

청구항 1

온톨로지 데이터 파일에 대한 일반 정보를 저장하는 온톨로지테이블;과
클래스, 속성, 개체 개념을 포함하는 리소스를 하나의 테이블로 구성하는 리소스테이블;을 포함하되,
상기 각 개념의 인스턴스(Instance)마다 유일한 식별자(ID)를 지정하는 것을 특징으로 하는 관계형 데이터베이스기반 온톨로지 저장소.

청구항 2

청구항 1에 있어서, 상기 온톨로지 저장소는,
상기 리소스들이 사용하는 네임스페이스의 식별자(ID)와 이름을 저장하는 네임스페이스(namespace)테이블;
개체의 정보를 포함함과 동시에 해당 개체와 연관된 클래스의 정보를 포함하는 개체(individual)테이블;
상기 리소스테이블의 식별자(ID)를 참조하여 주어(subject), 술어(predicate), 목적어(object) 필드를 포함하여 트리플(triple)을 표현하는 RDF트리플테이블;
등가, 비등가, 계층관계에 따른 관계유형을 지정하는 관계(relation)테이블;
온톨로지 속성 특성에 따라 속성 간 관계 유형을 테이블로 구성하여 저장하는 속성테이블;
온톨로지의 속성 간 관계에 대한 제약을 지정하여 저장하는 속성계약테이블;
클래스간 제약을 저장하는 클래스계약테이블;
중 하나 이상을 포함하여 구성되는 관계형 데이터베이스기반 온톨로지 저장소.

청구항 3

청구항 2에 있어서,
상기 RDF트리플테이블은,
상기 리소스테이블의 식별자 값을 저장하며, Get ID 함수를 이용하여 상기 리소스테이블에 저장된 리소스 이름에 근거하여 해당 리소스 식별자를 리턴하는 관계형 데이터베이스기반 온톨로지 저장소.

청구항 4

청구항 2에 있어서,
상기 저장소는,
상기 리소스테이블에 ID필드, RDF트리플테이블의 Subjdet 필드, 개체테이블의 식별자(ID)필드, 속성테이블의 PROPID필드, 속성계약테이블의 PROPID필드, 클래스계약테이블의 CLSID 필드 중 어느 하나 이상에 인덱스를 지정하는 것을 특징으로 하는 관계형 데이터베이스기반 온톨로지 저장소.

청구항 5

대용량 온톨로지 저장과 효율적 검색을 위한 관계형 데이터 베이스 기반의 청구항 1 내지 4 중 어느 한 항의 온톨로지 저장소를 이용하여 검색을 수행하는 방법에 있어서,

상기 온톨로지 저장소의 스키마에 근거한 시맨틱 질의를 실체화 뷰(MView, Materialized View)를 구성하여 검색을 수행하는 관계형 데이터베이스기반 온톨로지 저장소를 이용한 검색방법.

청구항 6

대용량 온톨로지 저장과 효율적 검색을 위한 관계형 데이터 베이스 기반의 청구항 1 내지 4 중 어느 한 항의 온톨로지 저장소를 이용하여 검색을 수행하는 방법에 있어서,

상기 온톨로지 저장소의 스키마에 근거한 질의의 저장프로시저를 구성하거나, 상기 온톨로지 저장소의 스키마에 의하여 구성된 트리거(trigger)를 적용하여 검색을 수행하는 관계형 데이터베이스기반 온톨로지 저장소를 이용한 검색방법.

청구항 7

제 1 항 내지 제 4 항 중 어느 한 항의 관계형 데이터베이스기반 온톨로지 저장소를 이용하여 컴퓨터에 검색을 실행시키는 프로그램을 기록한 컴퓨터로 판독 가능한 기록매체.

청구항 8

사용자 어플리케이션을 구동하는 사용자단말;

상기 어플리케이션과 데이터베이스와의 통신을 수행하는 관계형 데이터베이스인터페이스;

상기 관계형 데이터베이스인터페이스를 통해 상기 사용자단말이 청구항 1 내지 4 중 어느 한항의 온톨로지 저장소 내부의 각 테이블을 이용해 유지관리를 진행하도록 하는 데이터베이스관리모듈;

상기 온톨로지 저장소에 저장된 온톨로지데이터에 대한 질의를 해석하여 SQL 쿼리로 전환하고, 쿼리결과를 처리하여 사용자단말에 전송하는 시맨틱쿼리모듈;

을 포함하여 구성되는 온톨로지 저장소 관리시스템.

청구항 9

청구항 8에 있어서,

상기 데이터베이스관리모듈은,

스키마관리모듈, 인덱스 관리모듈, 뷰관리모듈, 저장 프로시저관리모듈, 트리거관리모듈 중 어느 하나 이상을 포함하여 구성되는 온톨로지 저장소 관리시스템.

청구항 10

청구항 9에 있어서,

상기 시맨틱 쿼리모듈은,

온톨로지 해석모듈, 쿼리생성모듈, 쿼리결과처리모듈 중 어느 하나 이상을 포함하여 구성되는 온톨로지 저장소 관리시스템.

명세서

기술분야

대용량 온톨로지의 효율적인 저장 및 유지관리를 위한 관계형 데이터베이스 기반저장소의 스키마 구조 및 이를

이용하여 검색효율을 향상하는 방법에 관한 것이다.

배경 기술

- [0002] 온톨로지(Ontology)는 지능형 웹 서비스의 시맨틱 웹과 에이전트 기술을 통하여 자동적인 서비스의 발견, 호출, 결합 및 상호운영, 실행 감시 및 복구를 수행 할 수 있도록 컴퓨터가 지식을 추론하고 처리할 수 있게 하기 위한 서비스 간의 관계를 정의하는 서비스이다. 이와 같이, 정보를 사람이 처리하던 것이 자동화된 에이전트(agent)를 통해 처리될 수 있는 방안을 고려되고 있다. 시맨틱 웹(Semantic web)을 표현하기 위한 수단으로, 자원 명세 언어(Resource Description Language:RDF)가 표준으로 사용되고 있고, 웹 온톨로지 언어(Web Ontology Language:OWL)은 RDF로 표현할 수 없는 어휘들을 수용하여 온톨로지들의 병합 및 추론을 위한 방법을 제공하기 위해 현재 표준이 진행 중이다.
- [0003] 즉, 시맨틱 웹(Semantic web)은 사용자의 요구를 만족시키는 방법에 대한 보다 지능적인 지원을 위하여 컴퓨터가 웹 정보자원에 대한 이해와 정보의 검색, 추출, 해석 등을 진행할 수 있도록 하는 차세대 웹이다. OWL은 W3C에 의해 2004년 12월에 권고안으로 채택된 이후에 많은 연구가 진행되고 있다. 오늘날 온톨로지 기술은 생명과학, 인공지능, 지능형자원관리, 소셜네트워크 등 여러분야에 널리 사용되고 있다. 온톨로지의 응용과 함께 방대해지는 온톨로지 데이터를 효율적으로 검색하고 처리할 수 있는 저장소(Storage)에 대한 요구가 제기되고 있다. 현재 온톨로지를 저장하고 처리하는 매체는 주로 메모리, 파일, 관계형 데이터베이스(Relational Database) 등이 있다. 메모리 기반 저장방식은 온톨로지 데이터를 한꺼번에 메모리에 읽어 들여 메모리 내부에서 데이터의 검색작업을 진행하는 방식이다. 이런 메모리 기반 저장관리시스템의 예로는 OWLim과 OWLJessKB 등이 있다. 이 방식은 실행효율이 매우 높지만 컴퓨터가 탑재한 물리적 메모리의 제한으로 오직 제한된 규모의 온톨로지 데이터만 처리가능하고 메모리 특성상 영속적인 저장이 불가능하다. 파일 기반 저장방식은 현재 OntoEdit, Protege 등 여러 온톨로지 툴에서 광범위하게 지원하는데 실행이 간단하고 영속저장을 지원한다. 그러나 이런 방식은 XML 형태로 표현된 RDF(Resource Description Framework)데이터 파일 처리시 RDF모델의 전반적인 데이터 구조를 파악하기 위하여 파일에 대한 반복적인 스캔과 I/O 작업이 실행되어야 하는데 이는 대용량 파일처리 시스템의 성능에 엄청난 저하를 일으킬 수 있다. 그러므로 이런 방식은 규모가 크지 않는 온톨로지 데이터의 저장에 적합하다. 관계형 데이터베이스는 오랜시간 동안의 개발연구를 거쳐 기술이 상대적으로 성숙되었고 대규모 데이터의 효율적인 저장과 쿼리 및 관리를 지원하며 관련 제품들도 많이 존재한다. 현재 관계형 데이터베이스에 온톨로지를 저장하는 방식은 주로 세가지 방식이 존재하는데 각 방식에 대한 설명과 장단점은 다음과 같다.
- [0004] 1) 수평방식 : 데이터베이스에 오직 한 개의 테이블만 구성하고 테이블의 각 필드는 온톨로지의 각 개념을 저장한다. 즉 온톨로지에서의 한 개 인스턴스(Instance)는 테이블에서 한 개 행의 각 필드들로 구성된다. 이런 저장방식의 특징은 구조가 간단하고 테이블의 필드의 개수가 온톨로지 파일에 포함된 각 인스턴스의 개념의 총 합과 같다. 그러나 이러한 필드의 정의는 데이터베이스 테이블이 허용하는 필드개수 제한을 초과할 수 있다는 문제가 발생할 수 있다.
- [0005] 2) 수직방식 : 이 방식은 데이터베이스 내부에 RDF 트리플의 구조에 따라 한 개의 테이블을 구성하는데 테이블의 매개 필드는 RDF의 Subject-Predicate-Object에 대응된다. Jena Protege 등이 이런 방식을 사용하고 있다. 이런 방식은 설계가 간단하고 현재의 온톨로지 데이터는 모두 RDF Triple의 형식으로 표현가능하므로 통용성도 우수하다. 그러나 이런 방식은 가독성이 낮고 온톨로지 클래스 관계 등 쿼리 진행시 SQL의 설계가 복잡해지고 쿼리 진행시 테이블 전반에 걸쳐 검색을 진행하므로 조인 연산이 필요할 때 테이블에 대한 전체 검색이 여러차례 발생하게 되는데 이는 쿼리 효율의 급격한 저하를 일으킬 수 있다.
- [0006] 3) 분해방식 : 이 방식은 모델분해의 사상을 기반으로 여러 개의 테이블에 온톨로지 개념을 분해하여 저장한다. 이 방식은 크게 두 가지로 나뉘는데 하나는 클래스 기반의 분해방식으로써 온톨로지의 각 클래스는 하나의 테이블을 구성한다. 다른 하나는 속성기반의 분해방식으로써 온톨로지의 각 속성은 하나의 테이블을 구성한다. Sesame은 이런 방식을 이용하였다. 이 방식하에서 각 테이블의 규모는 상대적으로 작아지므로 기타 방식에 비해 응답시간이 빠르다. 그러나 온톨로지의 여러 개의 클래스 혹은 속성과 연관되는 복잡한 쿼리는 결과도출을 위하여 결과 집합에 대한 교집합 연산이 필요한데 이는 쿼리 효율을 저하시킬 수 있다.

발명의 내용

해결하려는 과제

- [0007] 본 발명은 상술한 문제를 해결하기 위하여 안출된 것으로, 특히 본 발명에서는 기존 온톨로지 저장방법에 대한

비교분석을 통하여 기존 관계형 데이터베이스 기반 저장 시스템의 문제점을 개선하고 새로운 기능들을 추가하여 확장한 온톨로지 저장소를 설계고, 기존 저장방식들의 스키마 확장 어려움과 데이터 중복 저장 등 구조적인 문제점을 해결하고자 온톨로지 저장을 위한 관계형 데이터베이스 스키마를 제공하는 것을 목적으로 한다.

[0008] 특히, 본 발명에 따른 스키마의 구성은 저장공간의 사용률, 검색의 효율, 질의 구성의 용이성과 향후 확장가능성 등 여러 방면에 대한 고려를 기반으로 진행되며, 이와 동시에 상기 스키마기반으로 인덱스, 실체화 뷰(Materialized View), 저장 프로시저(Stored Procedure), 트리거(Trigger) 등 기법을 적용하여 진일보된 성능 향상과 사용자의 편의를 위한 방법을 제공하는 것을 또 다른 목적으로 한다.

과제의 해결 수단

[0009] 상기의 문제점을 해결하기 위하여 본 발명에서는 온톨로지 데이터 파일에 대한 일반 정보를 저장하는 온톨로지 테이블;과 클래스, 속성, 개체 개념을 포함하는 리소스를 하나의 테이블로 구성하는 리소스테이블;을 포함하되, 상기 각 개념의 인스턴스(Instance)마다 유일한 식별자(ID)를 지정하는 것을 특징으로 하는 관계형 데이터베이스 기반 온톨로지 저장소를 제공할 수 있도록 한다.

[0010] 또한, 상기 온톨로지 저장소의 스키마에 근거한 시멘틱 질의를 실체화 뷰(MView, Materialized View)를 구성하여 검색을 수행하거나, 상기 온톨로지 저장소의 스키마에 근거한 질의의 저장프로시저를 구성하거나, 상기 온톨로지 저장소의 스키마에 의하여 구성된 트리거(trigger)를 적용하여 검색을 수행하는 관계형 데이터베이스 기반 온톨로지 저장소를 이용한 검색방법을 제공할 수 있도록 한다.

발명의 효과

[0011] 본 발명에 따르면, 온톨로지 기술과 응용에 따른 온톨로지 데이터의 방대화 추세에 따라 본 발명에서 제안하는 관계형 데이터베이스에서 온톨로지 저장을 위한 스키마 및 이를 기반으로 구성되는 각 방법은 대용량 온톨로지 데이터의 효율적인 저장과 검색가능성을 제고하며 사용자 어플리케이션의 설계에도 편의를 도모하여 시스템 개발 효율을 제고할 수 있는 효과가 있다.

도면의 간단한 설명

[0012] 도 1은 본 발명에 따른 온톨로지의 저장 및 유지를 위한 관계형 데이터베이스의 스키마 구조를 도시한 것이다.

도 2는 본 발명에 따른 온톨로지의 저장 및 유지를 위한 관계형 데이터베이스의 구성도이다.

도 3은 본 발명에 따른 실시예를 나타낸 것이다.

발명을 실시하기 위한 구체적인 내용

[0013] 아래에서는 첨부한 도면을 참조하여 본 발명이 속하는 기술분야에서 통상의 지식을 가진 자가 용이하게 실시할 수 있도록 본 발명의 실시 예를 상세히 설명한다. 그러나 본 발명은 여러 가지 상이한 형태로 구현될 수 있으며 여기에서 설명되는 실시 예에 한정되지 않는다. 그리고 도면에서 본 발명을 명확하게 설명하기 위해서 설명과 관계없는 부분은 생략하였으며, 명세서 전체를 통하여 유사한 부분에 대해서는 유사한 도면 부호를 붙여 설명하기로 한다.

[0014] 이하, 본 발명에서 실시하고자 하는 구체적인 기술내용에 대해 첨부도면을 참조하여 상세하게 설명하기로 한다.

[0015] 도 1은 본 발명에 따른 온톨로지 저장소(이하, '저장소'라 한다.)의 스키마 구조를 도시한 구조도이다.

[0016] 도시된 도면을 참조하면, 본 발명에 따른 저장소는 온톨로지 데이터 파일에 대한 일반 정보를 저장하는 온톨로지 테이블(10)과, 클래스, 속성, 개체 개념을 포함하는 리소스를 하나의 테이블로 구성하는 리소스테이블(20)을 포함하여 구성될 수 있다. 특히 이 경우 상기 각 개념의 인스턴스(Instance)마다 유일한 식별자(ID)를 지정하는 것을 특징으로 한다.

[0017] 나아가, 본 발명에 따른 저장소는 상기 리소스들이 사용하는 네임스페이스의 식별자(ID)와 이름을 저장하는 네임스페이스(namespace)테이블(30), 개체의 정보를 포함함과 동시에 해당 개체와 연관된 클래스의 정보를 포함하는 개체(individual)테이블(50), 상기 리소스테이블의 식별자(ID)를 참조하여 주어(subject), 술어(predicate), 목적어(object) 필드를 포함하여 트리플(triple)을 표현하는 RDF트리플테이블(40)과, 등가, 비등가, 계층관계에 따른 관계유형을 지정하는 관계(relation)테이블(70), 온톨로지 속성 특성에 따라 속성 간 관계 유형을 테이블로 구성하여 저장하는 속성테이블(60), 온톨로지의 속성 간 관계에 대한 제약을 지정하여 저장하는 속성제약

테이블(80), 클래스간 제약을 저장하는 클래스제약테이블(90) 중 하나 이상을 포함하여 구성될 수 있다.

- [0018] 즉, 본 발명에서는 온톨로지의 구조 특징에 대한 분석과 기존의 관계형 데이터베이스 기반 저장방식에 대한 참고를 통하여 온톨로지의 저장과 처리 효율 제고를 위한 관계형 데이터베이스 기반 저장소 스키마를 구성하는 것을 요지로 한다.
- [0019] 구체적으로는, 해당 스키마의 설계과정에서 필수로 온톨로지의 중요한 개념들인 클래스(Class), 속성(Property), 개체(Individual) 등은 단독으로 테이블을 구성함으로써 각 개념의 저장 독립성을 확보하고 향후 확장에 용이하도록 하였다.
- [0020] 또한, 각 개념의 인스턴스(Instance) 마다 유일한 식별자(Identity)를 지정하여 개념 간 상호인용시 이런 식별자를 사용하여 저장하여 데이터의 리던던트(Redundant)를 줄이도록 한다.
- [0021] 예를 들면, RDF트리플(Triple) 테이블의 주어(Subject), 술어(Predicate), 목적어(Object) 필드(Field)는 각 리소스(Resource)의 식별자로 표현된다. 또한 온톨로지 추론에 자주 사용하게 되는 상속, 포함, 동등 등 관계는 따로 필드를 제공하거나 전문 테이블을 구성하여 검색을 효율을 제고하고자 하였다.
- [0022] 상술한 본 발명에 따른 저장소 스키마를 구성하는 각 테이블의 구성 및 기능은 다음과 같이 구체적으로 설명할 수 있다.
- [0023] - 온톨로지(OWLontology) 테이블 : 온톨로지명, 버전정보 등 온톨로지 데이터파일에 대한 일반적인 정보를 저장하는데 사용된다.
- [0024] - 리소스(OWLResource) 테이블 : 리소스는 온톨로지에서 매우 기본적인 개념으로써 클래스, 속성 등은 모두 리소스로 설명될 수 있다. 본 발명에서 구성한 저장소 스키마는 리소스를 하나의 테이블로 구성하여 각 연관 테이블이 해당 리소스의 ID를 참조하여 사용할 수 있도록 하였다. 해당 ID는 주키(Primary Key)로 UNIQUE 속성을 부여한다. 또한, 리소스(OWLResource) 테이블은 타입필드를 지정하여 해당 리소스가 속한 범주(클래스, 속성 등)을 지정해 줄 수 있게 한다.
- [0025] - 네임스페이스(Namespace) 테이블 : 네임스페이스 역시 온톨로지에서 중요한 개념으로서 술어(Glossary)들의 오염을 방지해 준다. 네임스페이스 테이블의 구조는 비교적 간단하며 리소스들이 사용하게 되는 각 네임스페이스의 ID와 이름들을 저장한다.
- [0026] - 개체(OWLIndividual) 테이블 : 온톨로지에서 개체는 클래스의 한 개 인스턴스으로써 특정 클래스와 관계를 가진다. 상기 개체(OWLIndividual) 테이블은 개체의 정보를 포함함과 동시에 해당 개체와 연관된 클래스의 정보도 함께 포함한다.
- [0027] - RDF트리플(RDFTriple) 테이블 : 자원 명세 언어(Resource Description Language:이하, 'RDF') 즉, RDF 트리플은 온톨로지에서도 또 다른 하나의 중요한 개념이다. RDF 트리플 테이블은 상술한 '수직저장방식'과 같게 subject(주어), predicate(술어), object(목적어) 필드를 포함하여 트리플을 표현한다. 각 필드의 값은 각각 리소스(OWLResource) 테이블 내의 리소스 식별자를 참조한다.
- [0028] -관계(OWLRelation) 테이블 : 웹 온톨로지 언어(Web Ontology Language:OWL)의 개념 간 관계는 등가관계인 equivalentClass, equivalentProperty, sameAs 등과 비등가관계인 differentFrom, AllDifferent가 존재한다. 이외에 온톨로지에서도 계층을 나타내는 subClass, subProperty 및 superClass, superProperty 등 관계가 존재한다. 비록 등가/비등가/계층 관계에 따라 테이블을 따로 구성할 수 있지만 향후 관계에 대한 질의시 테이블간 조인을 줄이기 위하여 관계(OWLRelation) 테이블에 모두 저장하고 type 필드를 통하여 관계유형을 지정해 준다.
- [0029] - 속성(OWLProperty) 테이블 : 온톨로지에서의 속성은 리소스로 OWLResource 테이블에서 type을 통하여 나타낼 수 있다. 하지만 속성은 보통 domain와 range 두가지 값이 주어져야 하며, 이런 값은 온톨로지 추론에서 자주 사용되므로 OWLProperty 테이블을 단독으로 구성하여 저장하도록 한다.
- [0030] 또한, 온톨로지의 속성의 특징에는 속성유형에 따라 ObjectProperty, DatatypeProperty 및 속성 간 관계를 나타내는 SymmetricProperty, TransitiveProperty, FunctionalProperty, InverseFunctionalProperty 등이 포함된다. 이런 관계 유형은 type 필드를 구성하여 서술하도록 한다.
- [0031] 이외도 속성 간 관계에는 inverseOf가 있는데 이는 이진속성으로 OWLProperty 테이블에서 표현시 inverse 관계

를 나타내는 속성의 저장을 위한 필드가 필요하다. 그러나 inverseOf 제약이 없는 속성에 대해서는 해당 필드가 항상 NULL값이 되므로 공간의 낭비가 발생한다. 때문에 inverseOf는 OWLRelation 테이블에서 type 값을 "inverseOf"로 지정하여 저장하도록 한다.

[0032] - 속성제약(OWLPropertyRestriction) 테이블 : 온톨로지에서 속성 간 관계에 대한 제약은 Restriction, onProperty, allValuesFrom, someValuesFrom, hasValue 등을 포함한다. 또한, 속성개수에 대한 제약으로 minCardinality, maxCardinality, cardinality 등이 있다. 이런 속성에 대한 제약은 OWLPropertyRestriction 테이블에 저장하여 type 필드를 이용하여 제약의 유형을 묘사하고 value 필드를 이용하여 type 값이 "hasValue", "minCardinality", "maxCardinality", "cardinality"시 값을 지정하도록 한다.

[0033] - 클래스제약(OWLClassRestriction 테이블) : 온톨로지에서 클래스 간에도 intersectionOf, unionOf, complementOf, disjointWith 등이 존재한다. 이런관계는 OWLClassRestriction테이블에 저장하도록 한다.

[0034] 발명의 내용을 상세하게 설명하기 위하여 도1에서 상기 온톨로지 저장소 스키마의 구조와 각 테이블간의 연관관계를 보여주고 있으며, 도3은 이러한 실시예를 보여주고 있다.

[0035] 도3과 상기 스키마를 기반으로 온톨로지 시맨틱적 질의 진행과정에 대한 예를 표1의 동시협력설계에서의 부품관련 데이터로 설명하면 다음과 같다. 표의 데이터는 설계자 DesignUnit1이 부품 Part1(310)을 설계하고 설계자 DesignUnit2가 부품 Part2(320)를 설계한다. Part1은 특징 Hole1(311)을 포함하고 Part2는 특징 Boss1(321)을 포함하고 Hole1(311)과 Boss1(321)은 서로 맞물림을 나타낸다.

[0036] {표 1} 동시협력설계관계 예시

Subject	Predicate	Object
DesignUnit1	codesign:designs	Part1
DesignUnit2	codesign:designs	Part2
Part1	codesign:has	Hole1
Part2	codesign:has	Boss1
Hole1	codesign:matchTo	Boss1

[0037]

[0038] 다음은 SPARQL과 SQL로 각각 표현된 현재 설계하고 있는 파트들의 모든 특징을 검색하는 쿼리문이다.

[0039] SPARQL :

[0040] PREFIX codesign : http://dbsi.inha.ac.kr/codesign/

[0041] SELECT ?f

[0042] WHERE

[0043] { ?d codesign:designs ?p ?p codesign:has ?f

[0044] }

[0045] SQL :

[0046] SELECT RT2.object

[0047] FROM RDFTriple RT1, RDFTriple RT2, OWLResource RES

[0048] WHERE RT1.object = RT2.subject

[0049] AND RT1.predicate = GetID(RES, "designs")

[0050] AND RT2.predicate = GetID(RES, "has")

- [0051] AND RES.NSID = GetID(NAMESPACE, "http://dbsi.inha.ac.kr/codesign/")
- [0052] 여기서 RDF트리플(Triple) 테이블은 리소스(OWLResource) 테이블의 리소스 식별자 값만 저장하므로 GetID함수를 이용하여 OWLResource 테이블에 저장된 리소스의 이름에 근거하여 해당 리소스 식별자를 리턴(Return)한다. GetID함수의 표현식은 다음과 같다.
- [0053] **[GetID함수의 표현식]**
- [0054] CREATE FUNCTION GetID(TBL, varchar20) RETURNS INT
- [0055] AS "SELECT ID FROM TBL WHERE name = \$1" ;
- [0056] 위의 예는 한차례의 셀프 조인(Self Join)과정에서 첫번째 RDF Triple의 주어가 두번째 RDFTriple 테이블의 술어와 동일한 튜플을 선택하게 된다. 그리고 첫번째 테이블의 술어부가 "designs"이고 두번째 테이블의 술어부가 "has"인 경우의 튜플을 재 선택한 다음 그 결과를 출력하게 된다. 이렇게 구성된 SQL 결과의 질의는 시맨틱 질의와 같은 결과를 출력한다.
- [0057] 저장소의 진일보 성능최적화를 위하여 본 발명에서는 위에서 구성한 스키마에 기반하여 인덱스를 구성하는 방법과 뷰를 생성하는 방법 및 저장프로시저 사용 등 방법을 적용한다.
- [0058] 데이터베이스 인덱스는 검색의 효율을 제고할 수 있지만 데이터의 추가, 삭제 작업이 빈번히 발생시 시스템은 더욱 많은 시간을 들여 인덱스를 유지관리 하게 된다. 때문에 인덱스를 구성함에 있어서 데이터 검색이 자주 발생하는 필드에 인덱스를 지정하고 업데이트가 자주 발생하는 필드에는 인덱스 지정을 피하도록 하여야 한다.
- [0059] 리소스(OWLResource) 테이블은 저장소 관계형 데이터베이스 내부의 각 테이블들이 주로 참조하는 테이블로써 데이터의 검색이 제일 빈번히 일어나며 기타 테이블은 이 테이블의 ID 필드를 주로 참조한다.
- [0060] 그러므로 OWLResource 테이블에서 리소스(Resource)의 테이블의 식별자 필드는 인덱스로 된다. 이외 검색이 자주 일어나는 RDFTriple 테이블의 subject 필드, OWLIndividual 테이블의 식별자 필드, OWLProperty 테이블의 PROPID 필드, OWLPropertyRestriction의 PROPID필드 및 OWLClassRestriction의 CLSID 필드에 인덱스를 지정해 준다.
- [0061] 이외 검색효율을 높이는 방법으로 실체화 뷰(MView)를 적용한다. MView는 뷰의 결과가 물리적으로 저장되므로 복잡한 온톨로지 추론관련 질의가 자주 필요한 경우 사전에 MView를 생성하면 조인이 필요하던 질의를 뷰에서 조인이 엮는 질의로 변환할 수 있다. 본 발명 저장소 스키마 설계상 온톨로지에서의 기본개념인 리소스를 OWLResource 테이블에 저장하였는데 이로 하여 온톨로지 개념 간 관계 검색시 거의 모든 질의는 이 OWLResource 테이블과 연관된다.
- [0062] 데이터베이스 검색에서 테이블 간의 조인은 시스템 효율 저하의 주요 원인으로 볼 수 있다. 이런 테이블 조인이 자주 발생하는 부분에 대하여 MView를 생성하면 검색의 효율을 향상시킬 수 있다. 다음은 클래스 검색을 위한 class_mview를 생성하는 예이다.
- [0063] MView : class_mview(ID, namespace, name)
- [0064] CREAT MATERIALIZED VIEW class_Mview
- [0065] BUILD IMMEDIATE
- [0066] REFRESH ON COMMIT
- [0067] ENABLE QUERY REWRITE
- [0068] AS
- [0069] SELECT ID, NSID, NAME
- [0070] FROM OWLResource

[0071] WHERE type="class";

[0072] 기타 individual_mview, property_mview 등 MVIEW은 상술문구에서 WHERE의 type의 값을 "individual" 혹은 "property"로 바꾸면 되므로 자세한 내용은 생략하도록 한다.

[0073] 또한, 시맨틱 질의 처리의 효율적 지원을 위하여 저장프로시저를 적용한다. 프로시저사용은 클라이언트가 서버에 대한 쿼리 진행시 네트워크상에서 전송하는 쿼리문의 크기를 줄여줄 수 있을 뿐만 아니라 모듈화 구조로 컴파일 후 중복호출이 가능하여 프로그래머의 개발 효율도 높여줄 수 있다. 저장프로시저의 예로 두개체의 동등여부 판단을 들면 다음과 같다.

[0074] CREATE PROCEDURE is_equivalent_individual (individual1

[0075] varchar2(100), individual2 varchar2(100)) AS

[0076] DECLARE

[0077] ind1ID integer, ind2ID integer, result integer,

[0078] BEGIN

[0079] SELECT ind_mv1.ID INTO ind1ID,

[0080] ind_mv2.ID INTO ind2ID

[0081] FROM individual_mview ind_mv1,

[0082] individual1_mview ind_mv2

[0083] WHERE ind_mv1.name = individual1

[0084] AND ind_mv2.name = individual2;

[0085] SELECT res1ID INTO result

[0086] FROM OWLRelation

[0087] WHERE type = "sameAs"

[0088] AND (res1ID = ind1ID AND res2ID = ind2ID)

[0089] OR (res1ID = ind2ID AND res2ID = ind1ID);

[0090] IF result IS NOT NULL THEN

[0091] COMMIT;

[0092] RETURN TRUE;

[0093] END;

[0094] COMMIT;

[0095] RETURN FALSE;

[0096] END

[0097] 상술 저장프로시저는 먼저 주어진 값이 개체인지 여부를 판단한 후 두 개체가 sameAs 관계를 가지는지 여부를 판단한다. 이 두 조건을 모두 만족시 TRUE를 리턴하여 서로 동등함을 나타내고 그렇지 않으면 FALSE를 리턴하여 서로 같지 않음을 나타낸다.

[0098] 본 발명의 저장소 스키마는 온톨로지에 필요한 정보를 여러 테이블에 나누어 저장하도록 설계되었으므로 이런 테이블 간 데이터의 무결성을 유지하여야 할 필요가 있다. 데이터의 무결성 유지를 위하여 본 발명에서 상기 저

장소 스키마에 의하여 구성된 트리거(Trigger)를 적용한다. 예를 들면 Namespace 테이블에서 ID를 변경하였을 경우 OWLResource 테이블에서도 해당 NSID를 업데이트 해야 할 필요가 있다. 이를 완성하는 트리거 정의문구는 다음과 같다.

- [0099] CREAT OR REPLACE TRIGGER Update_NS_ID
- [0100] AFTER UPDATE OF ID ON NAVESPACE
- [0101] BEGIN
- [0102] UPDATE OWLResource SET NSID =: new.ID
- [0103] WHERE NSID := old.ID
- [0104] END.
- [0105] 이런 과정은 모두 자동적으로 진행되므로 관리자로 하여금 테이블간 데이터의 일관성, 무결성 유지관리를 보다 쉽게 해줄 수 있다.
- [0106] 도 2는 상기 각 기능들을 통합적으로 관리하여 연결해주는 대용량 온톨로지 저장소 관리시스템의 구조를 나타낸다.
- [0107] 도시된 도면을 참조하면, 본 발명에 따른 온톨로지 저장소 관리시스템은 사용자 어플리케이션(120)을 구동하는 사용자단말(100)
- [0108] 상기 어플리케이션(120)과 상기 온톨로지 저장소(1)와의 통신을 수행하는 관계형 데이터베이스인터페이스(100)과 상기 관계형 데이터베이스인터페이스를 통해 상기 사용자단말(100)이 상술한 본 발명에 따른 온톨로지 저장소 내부의 각 테이블을 이용해 유지관리를 진행하도록 하는 데이터베이스관리모듈(200) 및 상기 온톨로지 저장소(1)에 저장된 온톨로지데이터에 대한 질의를 해석하여 SQL 쿼리로 전환하고, 쿼리결과를 처리하여 사용자단말에 전송하는 시맨틱쿼리모듈(300)을 포함하여 구성된다.
- [0109] 아울러, 상기 데이터베이스관리모듈(200)은, 스키마관리모듈(210), 인덱스 관리모듈(220), 뷰관리모듈(230), 저장 프로시저관리모듈(240), 트리거관리모듈(250) 중 어느 하나 이상을 포함하여 구성될 수 있다. 나아가, 상기 시맨틱 쿼리모듈(300)은, 온톨로지 해석모듈(310), 쿼리생성모듈(320), 쿼리결과처리모듈(330) 중 어느 하나 이상을 포함하여 구성될 수 있다.
- [0110] 즉, 도 2에 도시도니 본 발명에 따른 관계형 데이터베이스 인터페이스(400)는 주로 어플리케이션과 데이터베이스간의 통신을 책임진다. 데이터베이스 관리모듈(200)은 관계형 데이터베이스 인터페이스(400)를 통하여 사용자로 하여금 저장소 내부 각 테이블에서 스키마관리, 인덱스, 실체화 뷰관리, 저장프로시저 및 트리거의 생성/삭제/업데이트 등 유지관리를 진행할 수 있도록 한다. 시맨틱 쿼리 모듈(300)은 저장소에 저장된 온톨로지 데이터에 대하여 질의를 해석하여 SQL 쿼리로 전환하고 쿼리 결과를 처리하여 사용자한테 보내주는 기능을 포함한다. 사용자는 사용자 어플리케이션(120)을 이용하여 상기 각 모듈의 기능을 호출하여 사용하게 된다.

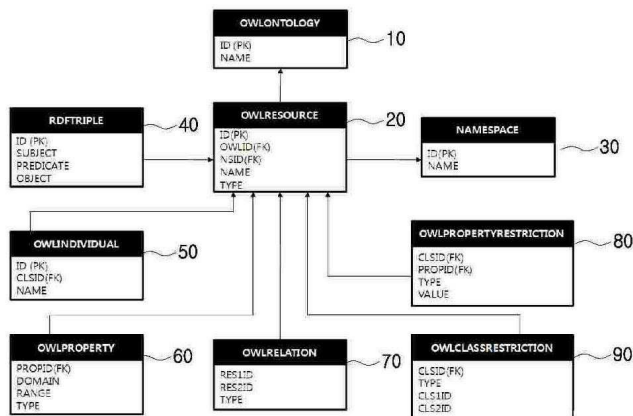
부호의 설명

- [0111] 1: 온톨로지 저장소
- 10: 온톨로지데이터
- 20: 리소스데이터
- 30: 네임스페이스데이터
- 40: RDF트리플 테이블
- 50: 개체데이터

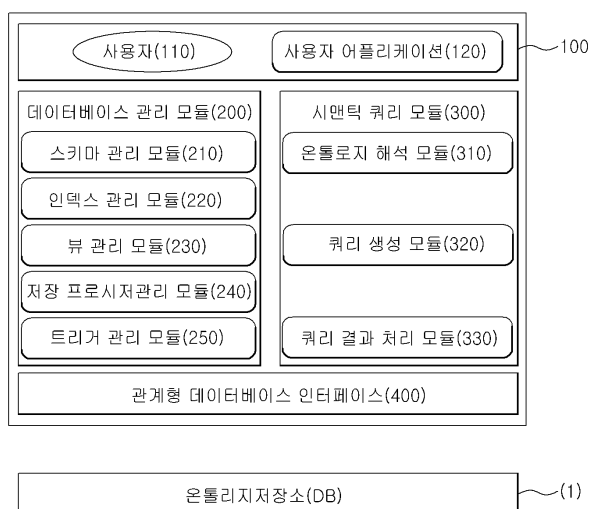
- 60: 속성테이블
- 70: 관계테이블
- 80: 속성제약테이블
- 90: 클래스제약테이블
- 100: 사용자단말
- 200: 데이터베이스 관리 모듈
- 300: 시맨틱 쿼리 모듈
- 400: 관계형 데이터베이스 인터페이스

도면

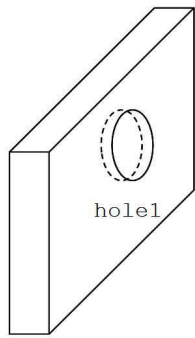
도면1



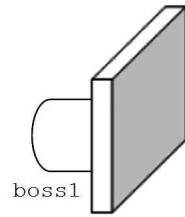
도면2



도면3



a) Part1



b) Part2