관계형 데이터베이스 뷰 정의로부터 온톨로지 클래스와 계층 관계 생성 기법

(Generating Ontology Classes and Hierarchical Relationships from Relational Database View Definitions)

양 준 석 † 김 기 성 † 김 형 주 ††
(Junseok Yang) (Kisung Kim) (Hyoung-Joo Kim)

요 약 온톨로지는 시맨틱 웹을 구현하기 위해 중요한 역할을 하지만 이를 구축하는 작업은 많은 시간을 필요로 한다. 그러므로 기존 웹의 데이터 중 많은 양을 차지하고 있는 관계형 데이터베이스로부터 온톨로지를 자동으로 생성하는 연구들이 진행되고 있다. 기존의 연구들은 데이터베이스 스키마와 저장된 데이터 분석을 통한 온톨로지 생성에 대한 연구들이 주를 이룬다. 이러한 연구들은 데이터베이스 스키마 중 테이블과 제약조건만을 분석하여 온톨로지 스키마를 생성하며, 뷰 정의를 고려하지 않는다. 그러나 뷰는 데이터베이스 설계자가 데이터베이스를 사용하는 도메인을 고려하여 정의하므로, 뷰 정의를 고려할 경우 추가적인 클래스와 상하위 관계를 생성할 수 있다. 그리고 이렇게 생성된 클래스는 온톨로지에 대한 질의 처리와 통합에 유용하게 사용될 수 있다.

본 논문에서는 기존의 방법들을 분석하여 클래스와 상하위 관계 생성을 정형화하였으며, 뷰 정의를 분석하여 기존의 방법을 통해 생성된 온톨로지에 추가적인 클래스와 상하위 관계를 생성하는 방법을 제안한다. 또한 제안하는 방법을 이용해 예제 데이터베이스 스키마로부터 생성된 온톨로지의 결과 분석을 수행하고, 이를 통해 뷰 정의로부터 의미 있는 클래스와 상하위 관계가 추가적으로 생성되었음을 보인다.

키워드 : 온톨로지, 온톨로지 자동 생성, 스키마 맵핑

Abstract Building ontology is the key factor to construct semantic web. However, this is time-consuming process. Hence, there are several approaches which automatically generate the ontologies from relational databases. Current studies on the automatic generation of the ontologies from relational database are focused on generating the ontology by analyzing the database schema and stored data. These studies generate the ontology by analyzing only tables and constraints in the schema and ignore view definitions. However, view definitions are defined by a database designer considering the domain of the database. Hence, by considering view definitions, additional classes and hierarchical relationships can be generated. And these are useful in answering queries and integration of ontologies.

In this paper, we formalize the generation of classes and hierarchical relationships by analyzing existing methods, and we propose the method which generates additional classes and hierarchical relationships by analyzing view definitions. Finally, we analyze the generated ontology by applying our method to synthetic data and real-world data. We show that our method generates meaningful classes and hierarchical relationships using view definitions.

Key words : Ontology, Ontology generation, Schema mapping

· 본 연구는 BK-21 정보기술 사업단 및 지식경제부 및 정보통신산업진흥원의 대학 IT연구센터 육성·지원사업(NIPA-2010-C1090-1031-0002)의 연구결과로 수행되었음

† 비 회 원 : 서울대학교 컴퓨터공학부
jsyang@idb.snu.ac.kr
kskim@idb.snu.ac.kr
†† 종신회원 : 서울대학교 컴퓨터공학부 교수
hjk@snu.ac.kr

논문접수 : 2010년 1월 18일
심사완료 : 2010년 9월 24일

정보과학회논문지: 데이타베이스 제37권 제6호(2010.12)

1. 서 론

1998년 팀 버너스 리가 시맨틱 웹이라는 비전을 제안한 이래, 이를 구현하기 위한 많은 연구가 진행되고 있다. 시맨틱 웹은 기존의 웹이 진화한 형태로, 웹 상에 존재하는 정보와 서비스에 잘 정의된 의미를 부여하여 사람뿐만 아니라 기계도 웹을 이해할 수 있도록 하고, 더 나아가 기계를 이용한 자동화된 검색과 통합 등을 목표로 하고 있다.

온톨로지는 이런 시맨틱 웹을 실현하기 위한 중요한 역할을 한다. 온톨로지란 지식 도메인을 표현하기 위한 하나의 방법으로, 컨셉과 그들 사이의 관계를 정의 및 설명할 수 있도록 한다. W3C는 이러한 온톨로지를 웹 상에서 구축하고 배포할 수 있도록 하기 위한 표준으로 RDF(S), OWL 등을 제안했다.

그러나 시맨틱 웹을 구현하는 데는 온톨로지 구축이 중요하며, 온톨로지를 구축하는 작업은 도메인 전문가와 많은 시간을 요구 하는 등 여러 가지 어려움이 있다. 그러므로 이러한 온톨로지를 구축하기 위한 한 가지 방법으로 기존의 데이터로부터 온톨로지를 생성하는 연구들이 진행되고 있다. 기존의 웹에는 HTML, XML, 관계형 데이터베이스, 온톨로지 등 여러 가지 형태의 데이터들이 존재한다. 그 중 많은 양을 차지하고 있는 데이터는 관계형 데이터베이스로, 정적인 웹의 500배에 해당하는 데이터가 데이터베이스에 저장되어 있으며, 그 중 75%가 관계형 데이터베이스인 것으로 추정되고 있다[1]. 그러므로 관계형 데이터베이스로부터 온톨로지를 생성하는 작업은 기존의 웹에서 시맨틱 웹으로 진화하는데 있어서 중요한 문제이다.

관계형 데이터베이스로부터 온톨로지를 생성하는 작업은 도메인에 적합한 온톨로지를 생성하기 위해 도메인 전문가가 직접 수행할 수 있다[2,3]. 그러나 데이터베이스 스키마가 복잡하며 방대한 양의 데이터가 저장된 경우 이와 같은 작업 방식은 온톨로지를 생성하는데 있어 많은 시간을 필요로 한다. 그러므로 이러한 작업을 자동으로 수행하는 방법에 대한 연구들[3-7]이 진행되고 있다. 관계형 데이터베이스로부터 온톨로지를 자동으로 생성하는 기존의 연구들은 먼저 데이터베이스 스키마와 저장된 데이터 분석을 통해 온톨로지 스키마를 생성한 다음, 생성된 온톨로지 스키마에 맞게 저장된 데이터로부터 온톨로지 데이터를 생성한다. 기존의 연구들은 이러한 과정 중 온톨로지 스키마를 생성하는데 초점을 두고 있다.

그러나 기존의 연구들은 온톨로지 스키마 생성시 데이터베이스 스키마 중 테이블과 제약조건만을 고려하였으며, 뷰 정의는 고려하지 않았다. 그러나 뷰 정의는 테이블과 달리 추가·삭제가 용이한 가상의 테이블이며, 데이터베이스 설계자가 데이터베이스를 사용하는 도메인을 고려하여 정의한다. 그러므로 뷰 정의를 고려하여 온톨로지를 생성할 경우 기존의 방법을 통해 생성된 온톨로지에 추가적인 클래스와 상하위 관계를 생성할 수 있다. 그리고 이렇게 생성된 클래스는 온톨로지에 대한 질의 처리와 통합에 유용하게 사용될 수 있다.

이에 따라, 우리는 먼저 기존의 방법들을 분석하여 클래스와 상하위 관계 생성을 정형화하고, 이를 뷰 정의에 적용하는 방안에 대해 연구하였다. 그리고 뷰 정의로부터 추가적인 클래스와 상하위 관계를 생성하는 방법을 제안하며, 우리가 제안한 방법이 실제로 의미 있는 클래스와 상하위 관계를 생성함을 예제 데이터베이스 스키마와 실제 응용프로그램의 데이터베이스 스키마에 대한 결과 분석을 통해 보인다. 또한 의미가 없는 클래스를 생성하거나, 클래스로써 의미가 있는 뷰를 클래스로 생성하지 않는 경우도 있음을 결과 분석을 통해 보인다. 그리고 기존 연구와의 비교를 통해 우리가 제안한 방법이 기존 연구에서는 얻을 수 없는 결과를 얻음을 보인다.

우리가 제안하는 방법은 데이터베이스 스키마로부터 온톨로지 스키마를 자동으로 생성하는 방법에 관한 연구로, 이렇게 생성된 온톨로지에서 의미 관계가 적합한지 판별하는 것에 대해서는 기존의 방법들과 마찬가지로 고려하지 않는다.

본 논문의 구성은 다음과 같다. 2장에서는 관계형 데이터베이스로부터 온톨로지 자동 생성에 대한 관련 연구들에 대해서 살펴본다. 3장에서는 관계형 데이터베이스 스키마 모델과 온톨로지 스키마 모델을 정의하고, 예제 데이터베이스 스키마를 정의한다. 4장에서는 기존 연구들의 방법에 대해서 분석하고 이를 정형화한다. 5장에서는 제안하는 방법인 뷰 정의 분석을 통한 클래스와 상하위 관계 생성에 대해 설명하고, 6장에서는 우리가 제안한 방법을 이용해 예제 데이터베이스 스키마로부터 생성된 온톨로지에 대한 결과 분석을 수행하며, 기존 연구와의 비교를 위해 실제 응용프로그램의 데이터베이스 스키마로부터 생성된 온톨로지에 대한 결과 분석을 수행한다. 마지막으로 7장에서 결론에 대해 언급하기로 한다.

2. 관련 연구

관계형 데이터베이스로부터 온톨로지를 생성하는 방법에 대해 여러 연구들이 진행되어왔다. 이러한 연구들은 크게 자동으로 생성하는 방법과, 수동 혹은 반-자동(semi-automatic)으로 생성하는 방법으로 구분된다[8]. 자동 생성 연구들은 데이터베이스 스키마와 저장된 데이터에 대한 분석을 통해 온톨로지를 생성한다. 자동 생성의 기반이 되는 방법은 팀 버너스 리가 제안한 테이

블-클래스(Table to Class) 방법[9]으로, 테이블의 레코드는 RDF의 노드로, 테이블의 컬럼은 RDF의 프로퍼티로, 테이블의 셀은 값으로 생성한다. 대부분의 자동 생성 연구들은 이 방법의 영향을 받았으며, 일반적으로 테이블로부터 클래스가 생성된다. D2RQ[3]에서는 테이블-클래스 방법을 사용하여 테이블과 뷰로부터 클래스를 생성하고 이를 바탕으로 온토롤지를 생성한다. 이때 D2RQ는 뷰의 특성을 고려하지 않고, 뷰를 테이블과 동일한 방법으로 다룬다. SOAM[7]에서는 먼저 데이터베이스 스키마와 저장된 데이터를 분석하여 온톨로지를 생성한 후, WordNet과 같은 다른 분류체계를 이용하여 자동 생성된 온톨로지에 대해 도메인 전문가가 평가 및 수정을 할 수 있도록 한다. [4,5]에서는 데이터베이스 스키마에서 테이블뿐만 아니라 주키(primary key)와 외부키(foreign key) 제약조건을 이용하여, 테이블들로부터 생성된 클래스들 사이의 상하위 관계를 생성한다. 이와 같은 방법은 관계형 데이터베이스 스키마로부터 객체지향 데이터베이스 스키마를 얻어내는 연구[10]와 같이, 과거 객체지향 데이터베이스 분야에서도 연구된 바가 있다. [6]에서는 테이블의 컬럼들 중 특정 컬럼이 테이블의 레코드들을 분류하는 특성이 있다는 사실에 기반하여, 저장된 데이터에 대한 마이닝을 통해 해당 컬럼을 찾아내고 그에 따른 하위 클래스들을 추가적으로 생성한다.

한편, 수동 혹은 반-자동 생성 연구들은 자동으로 생성하는 방법만으로는 얻을 수 없는 도메인에 특화된 의미 정보들을 반영하여 온톨로지를 생성한다. [2]에서는 도메인 특성을 고려하여 도메인에 적합한 온톨로지 스키마를 만든 후, 이를 바탕으로 온톨로지를 생성한다. 그리고 D2RQ에서는 도메인 전문가가 선언적 맵핑 언어(declarative mapping language)를 사용해서 생성 방법을 기술할 수 있도록 하여, 자동 생성뿐만 아니라 수동 혹은 반-자동으로 온톨로지를 생성할 수 있다.

3. 모델 정의 및 예제 스키마

이 장에서는 본 논문에서 사용하는 관계형 데이터베이스 스키마 모델과 온톨로지 스키마 모델을 정의하고, 앞으로 사용할 예제 데이터베이스 스키마를 보인다.

3.1 관계형 데이터베이스 스키마 모델

관계형 데이터베이스 스키마 D는 $D=(R, V, A, B)$로 정의한다. R은 테이블 전체집합인 $R=\{R_1, ..., R_n\}$, V는 뷰 전체집합인 $V=\{V_1, ..., V_m\}$로 정의하며 A는 테이블의 컬럼 전체집합인 $A=\{a_{11}, ..., a_{nl}\}$, B는 뷰의 결과 컬럼 전체집합인 $B=\{b_{11}, ..., b_{mk}\}$로 정의한다. 테이블 R_i는 해당 테이블에 속한 컬럼들의 집합인 $R_i=\{a_{i1}, ..., a_{ij}\}$로 정의하며, 테이블 R_i에 속한 컬럼들은 a_{ij}의 형태를 갖는다. 또한 뷰 V_i는 해당 뷰에 속한 컬럼들의 집합인 $V_i=\{b_{i1}, ..., b_{ij}\}$로 정의하며, 뷰 V_i에 속한 컬럼들은 b_{ij}의 형태를 갖는다. 단, 뷰의 결과 컬럼 중 숫자나 문자열과 같이 테이블이나 뷰의 컬럼을 참조하지 않는 결과 컬럼은 고려하지 않는다. 그리고 테이블 R_i의 주키는 $pkey(R_i)=\{a_{ij} \in R_i \mid a_{ij}$는 R_i의 주키$\}$로 컬럼들의 집합이며, 테이블 R_i에 속한 컬럼 중 테이블 R_j를 참조하는 외부키는 $fkey(R_i, R_j)=\{F \subseteq R_i \mid F$는 외부키로 $pkey(R_j)$를 참조$\}$다. 테이블은 외부키 여러 개를 가질 수 있으므로 $fkey$의 결과는 컬럼들의 집합을 원소로 갖는 집합이다. 또한 뷰의 주키를 다음과 같이 정의한다.

정의 3.1 뷰의 주키: $Y \in R$, $Z \in V$에 대해 뷰 $X \in V$의 주키는 다음과 같다.

$$vpkey(X)=\{P \subseteq X \mid parent(P)=pkey(Y) \wedge Y \in source(X)\} \cup \{P \subseteq X \mid parent(P) \in vpkey(Z) \wedge Z \in source(X)-\{X\}\}$$

정의 3.1에서 뷰 X의 주키는 뷰 정의에 사용된 테이블 Y나 뷰 Z의 주키 집합이다. 여기서 $source(X)$는 뷰 X를 정의하는데 사용된 테이블이나 뷰들의 집합이며, $parent(P)$는 뷰 결과 컬럼의 부분집합 P안의 모든 컬럼을 자신의 부모 컬럼으로 변환한다. 예를 들어, 그림 1에서 $source(V_9)=\{R_3, V_7\}$이며 $parent(V_9)=\{a_{31}, b_{71}\}$이다. 만약 $pkey(R_3)=\{a_{31}\}$이고 $vpkey(V_7)=\{\{b_{71}\}\}$라면, 정의 3.1에 의해 $vpkey(V_9)=\{\{b_{91}\}, \{b_{92}\}\}$가 된다. 또한 정의 3.1에서 뷰 Z는 X가 될 수 없다. 즉, 재귀적으로 정의된 뷰는 고려하지 않는다.

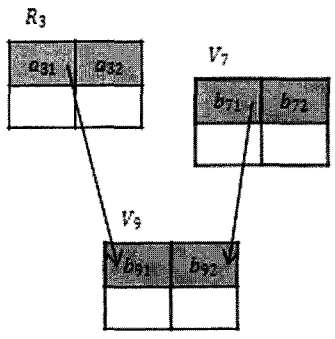

그림 1 뷰 컬럼의 부모 컬럼

3.2 온톨로지 스키마 모델

본 논문에서는 온톨로지 스키마를 표현하기 위해 RDFS[11]를 사용한다. RDFS는 온톨로지에서 사용하는 클래스와 프로퍼티 및 그들 사이의 관계를 정의하기 위해 사용되며, 그래프-기반 데이터 모델인 RDF로 표현

가능하다. RDFS에서 클래스는 온톨로지에서 표현하고자 하는 정보 자원인 리소스(resource)를 그룹으로 나누는 역할을 하며, 리소스는 자신만의 유일한 식별자인 URI(Uniform Resource Identifier)를 가지거나 URI가 없는 공노드(blank node)가 된다. 우리가 제안하는 방법은 클래스와 그들 사이의 상하위 관계 생성에 초점을 두었으므로 온톨로지 스키마 모델을 아래와 같이 정의한다.

온톨로지 스키마 O는 $O=(C,S)$인 방향성 그래프로 정의한다. C는 클래스 전체집합 $C=\{c_1,...,c_n\}$로 정의하며 S는 간선들의 집합 $S=\{(c_i,c_j),...\}$로 정의한다. 간선 (c_i,c_j)는 RDF 트리플 (c_i *subClassOf* c_j.)를 의미한다. 그리고 테이블 R_i나 뷰 V_i로부터 생성된 클래스 c_i는 각각 $class(R_i)$, $class(V_i)$로 정의한다.

3.3 예제 스키마

앞으로는 예제 데이터베이스 스키마로 표 1을 사용한다. 예제 데이터베이스 스키마는 [5]에서 사용한 대학교 데이터베이스 스키마를 기반으로, 우리가 제안한 방법을 설명하기 위해 뷰 정의를 추가했다.

예제 데이터베이스 스키마에서 *Person* 테이블은 사람에 대한 정보를 가지고 있으며, 사람은 *Professor* 테이블이 가지고 있는 교수 혹은 *Ustudent*, *Gstudent* 테이블이 가지고 있는 학생이다. *Dept* 테이블은 학과 정보를 가지고 있으며, *Course* 테이블은 강의 정보를 가지고 있다. 그리고 *Takes* 테이블은 어떤 학생이 어떤 강의를 수강하는지에 대한 정보를 가지고 있다. 또한 *Ustudent_male* 뷰는 학부 학생들 중 남학생, *Ustudent_grade_cnt* 뷰는 학년별 학부 학생수, *Student* 뷰는 전체 학생을 의미한다.

표 1 예제 데이터베이스 스키마

테이블	컬럼		
	주키	외부키	
PERSON	ID, SSN, NAME, SEX, EMAIL		
	ID		
PROFESSOR	ID, TITLE, DEPT		
	ID	ID	→ PERSON.ID
		DEPT	→ DEPT.CODE
USTUDENT	STUDENTID, PERSONID, DEPT, ADVISER, GRADE, RESULT		
	STUDENTID	PERSONID	→ PERSON.ID
		DEPT	→ DEPT.CODE
		ADVISER	→ PROFESSOR.ID
GSTUDENT	STUDENTID, PERSONID, DEPT, ADVISER, RESULT, DEGREE		
	STUDENTID	PERSONID	→ PERSON.ID
		DEPT	→ DEPT.CODE
		ADVISER	→ PROFESSOR.ID
DEPT	CODE, NAME		
	CODE		
COURSE	CID, TITLE, DEPT		
	CID	DEPT	→ DEPT.CODE
TAKES	STUDENTID, CID		
	STUDENTID, CID	STUDENTID	→ USTUDENT.STUDENTID
		STUDENTID	→ GSTUDENT.STUDENTID
		CID	→ COURSE.CID

뷰 정의
create view USTUDENT_MALE as select * from USTUDENT where SEX='M'
create view USTUDENT_GRADE_CNT as select GRADE, COUNT(*) as CNT from USTUDENT group by GRADE
create view STUDENT as select STUDENTID, NAME, DEPT, ADVISER, RESULT from USTUDENT, PERSON where PERSONID=ID union select STUDENTID, NAME, DEPT, ADVISER, RESULT from GSTUDENT, PERSON where PERSONID=ID

예제 데이터베이스 스키마 중 일부를 위에서 정의한 데이터베이스 스키마 모델로 표현하면 다음과 같다.

- *Course={cid,title,dept}*
- *pkey(Course)={cid}*
- *fkey(Course)={{dept}}*
- *Ustudent_male={studentid,personid,dept,adviser, grade, result}*
- *vpkey(Ustudent_male)={{studentid}}*

4. 기존의 방법

이 장에서는 데이터베이스 스키마의 테이블과 주키·외부키 제약조건을 이용하여 클래스와 상하위 관계를 생성하는 기존 연구들의 방법에 대해서 분석하고, 이를 정형화 한다.

4.1 클래스 생성

기존의 연구들은 데이터베이스 스키마의 테이블들을 기반으로 클래스들을 생성한다. 클래스 생성 과정 중 대부분의 테이블로부터 클래스를 생성하지만, 이진관계(binary relationship) 테이블의 경우 클래스로 생성하지 않는다 [4,5]. 이는 이진관계 테이블이 2개의 외부키를 이용하여 두 테이블 사이의 관계를 표현한 테이블이며, 온톨로지에서 두 클래스 사이의 관계는 프로퍼티로 표현하기 때문이다. 그러므로 우리는 이진관계 테이블을 아래와 같이 정의하며, 이진관계 테이블들의 집합을 B라고 정의한다.

정의 4.1 이진관계 테이블: $X, Y, Z \in R$에 대해 다음 조건을 만족하는 X는 이진관계 테이블이다.

$$\exists F_{XY}, F_{XZ}\,(F_{XY} \in fkey\,(X, Y) \wedge F_{XZ} \in fkey\,(X, Z) \wedge F_{XY} \neq F_{XZ} \wedge F_{XY} \cup F_{XZ} = X)$$

정의 4.1에서 테이블 X는 테이블 Y와 테이블 Z사이의 관계 테이블이며, 외부키 F_{XY}와 F_{XZ} 외의 컬럼은 가지고 있지 않다. 또한 외부키 F_{XY}와 F_{XZ}는 서로 다르다. 여기서 테이블 X는 재귀적 이진관계(recursive binary relationship)를 표현할 수도 있으므로 Y와 Z가 다르다는 조건은 없다. 예를 들어, 예제 데이터베이스 스키마에서 정의 4.1에 의해 이진관계 테이블 집합 B의 원소가 되는 테이블은 *Takes*이다. *Takes* 테이블은 그림 2와 같이 2개의 외부키인 *studentid* 컬럼과 *cid* 컬럼을 이용하여 *Ustudent* 테이블과 *Course* 테이블 사이의 관계를 표현한다. 그리고 외부키 외의 다른 컬럼은 가지고 있지 않다.

한편, 테이블로부터 생성된 클래스와 해당 테이블의

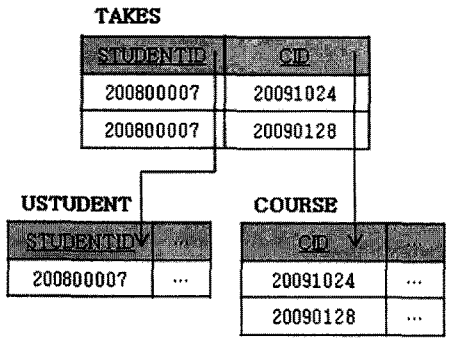

그림 2 이진 관계 테이블

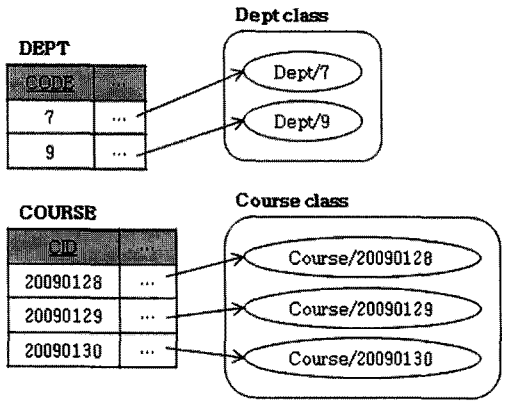

그림 3 테이블과 클래스, 레코드와 리소스

관계는 그림 3과 같으며, 그림에서 타원은 리소스를, 타원을 감싸고 있는 사각형은 해당 리소스의 클래스를 의미한다. 그림 3에서 확인할 수 있듯이 테이블의 레코드들은 생성된 클래스의 리소스가 된다. 기존의 테이블-클래스 방법에서는 레코드를 리소스 URI가 없는 공노드로 생성함으로써 URI를 통해 특정 리소스를 지칭할 수 없으며, 그로 인해 리소스들 사이의 관계를 생성할 수 없다. [2]에서는 이러한 방법의 단점을 지적하고 주키를 이용하여 리소스의 URI를 생성하며, D2RQ에서도 [2]와 마찬가지로 주키를 이용하여 리소스 URI를 생성한다. 그리고 [5,7]에서는 데이터베이스 스키마가 최소 3NF (third normal form) 이상임을 가정한다. 이와 같은 연구들을 바탕으로, 테이블로부터 클래스를 생성하는데 있어서 주키가 중요한 역할을 함을 알 수 있다.

우리는 정의 4.1과 위에서 살펴본 주키의 중요성을 바탕으로, 클래스로 생성되는 테이블을 아래와 같이 정의하며, 클래스로 생성되는 테이블들의 집합을 T라고 정의한다.

정의 4.2 클래스로 생성되는 테이블: 다음 조건을 만족하는 $X \in R$는 클래스로 생성되는 테이블이다.

$$pkey(X) \neq \varnothing \wedge X \notin B$$

즉, 주키를 가지고 있으면서 이진관계 테이블이 아닌 테이블이 클래스로 생성되는 테이블이다. 예를 들어, 정의 4.2에 의해 클래스로 생성되는 테이블들의 집합 T의 원소가 되는 테이블들은 *Person*, *Professor*, *Ustudent*, *Gstudent*, *Dept*, *Course*이다.

4.2 상하위 관계 생성

기존의 연구들[4,5]에서는 데이터베이스 스키마의 주키와 외부키 제약조건을 분석하여 테이블로부터 생성된 클래스들 사이의 상하위 관계를 생성한다. 우리는 기존의 방법을 사용하여 생성되는 상하위 관계를 아래와 같이 정의한다.

정의 4.3 테이블 상하위 관계: $X, Y \in T$에 대해 다음 조건을 만족하는 X, Y로부터 생성된 클래스 $c_i = class(X)$, $c_j = class(Y)$는 상하위 관계 (c_i, c_j)이다.

$$pkey(X) \in fkey(X, Y) \wedge pkey(X) \neq \varnothing$$

정의 4.3에서 상하위 관계 (c_i, c_j)는 주키이면서 외부키인 컬럼을 가지고 있는 테이블 X로부터 생성된 클래스 c_i가 클래스 c_j의 하위 클래스임을 의미하며, 테이블 X의 외부키가 참조하는 테이블 Y로부터 생성된 클래스 c_j가 클래스 c_i의 상위 클래스임을 의미한다. 예를 들어, 정의 4.3에 의해 생성되는 상하위 관계는 *Professor* 테이블로부터 생성된 클래스가 *Person* 테이블로부터 생성된 클래스의 하위 클래스가 되는 관계로 그림 4와 같다.

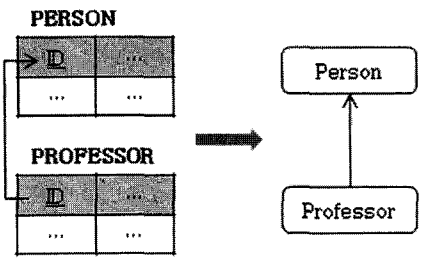

그림 4 테이블 상하위 관계

5. 뷰 정의를 이용한 클래스와 상하위 관계 생성

이 장에서는 뷰 정의로부터 클래스를 생성하는 방법과, 이렇게 생성된 클래스의 상하위 관계를 생성하는 방법에 대해서 설명한다.

5.1 클래스 생성

뷰는 가상 테이블(virtual table)이며, 뷰의 결과는 테이블과 마찬가지로 컬럼과 레코드를 가지고 있다. 그러므로 뷰로부터 클래스를 생성할 수 있다고 직관적으로 생각할 수 있다. 그러나 테이블과 달리, 뷰 정의는 주키 제약조건을 가지고 있지 않으므로 정의 4.2를 그대로 이

용할 수 없다. 그러므로 본 논문에서는 아래와 같은 과정을 통해 클래스로 생성될 수 있는 뷰의 조건에 대해서 알아보았다.

- 테이블의 레코드로부터 리소스 생성시, 주키를 이용하여 리소스의 URI를 생성한다.
- 뷰는 주키를 가지고 있지 않으므로, 그림 5의 (a)와 같이 결과 레코드가 어느 리소스에 대한 정보인지 알 수 없다.
- 뷰의 결과 레코드가 어느 리소스에 대한 정보인지 알려면, 결과 레코드로부터 리소스의 URI를 생성할 수 있어야 한다.
- 뷰의 결과 컬럼은 그림 5의 (b)와 같이 기존 테이블의 주키를 가질 수 있다.
- 정의 4.2에서는 주키가 있는 테이블로부터 클래스를 생성한다.

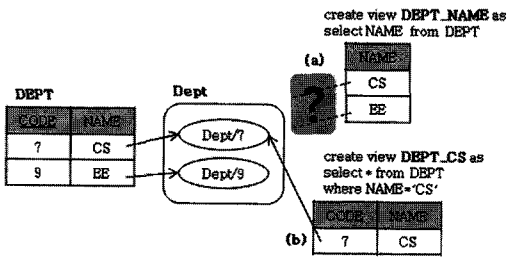

그림 5 뷰와 클래스

위의 과정을 통해 우리는 클래스로 생성될 수 있는 뷰를 아래와 같이 정의하며, 클래스로 생성되는 뷰들의 집합을 U라고 정의한다.

정의 5.1 클래스로 생성되는 뷰: 다음 조건을 만족하는 $X \in V$는 클래스로 생성되는 뷰이다.

$$| vpkey(X) | = 1$$

즉, 뷰 X가 결과 컬럼에 오직 하나의 주키만 가지고 있는 경우, X는 클래스로 생성될 수 있다. 뷰의 결과 컬럼에 주키가 없는 경우 해당 뷰는 클래스로 생성하지 않는다. 그리고 뷰의 결과 컬럼에 2개 이상의 주키를 가지고 있는 경우 해당 뷰는 그림 2와 같이 관계를 표현한 뷰라고 보고 클래스로 생성하지 않는다. 예를 들어, 뷰 *Ustudent_male*과 *Student*는 결과 컬럼에 하나의 주키를 가지고 있으므로 클래스로 생성되며, *Ustudent_grade_cnt*는 결과 컬럼에 주키를 가지고 있지 않으므로 클래스로 생성되지 않는다.

뷰 정의에는 그림 6과 같이 서브쿼리가 포함될 수 있다. 이 논문에서는 서브쿼리 또한 뷰 *Dept_cs*={*code*, *name*}로 보고, 정의 5.1을 이용하여 클래스로 생성한다.

5.2 상하위 관계 생성

```
create view ··· as
select ··· from ···,
select * from DEPT where NAME='CS' as DEPT_CS
···
```

그림 6 서브쿼리

정의 4.3에 따르면, 테이블의 경우 주키와 외부키 조건을 이용하여 상하위 관계를 생성하였다. 그러나 뷰 정의는 주키와 외부키 제약조건을 가지고 있지 않으므로, 정의 4.3을 이용해서 뷰로부터 생성된 클래스에 대한 상하위 관계를 생성할 수 없다. 그러므로 우리는 뷰 정의를 분석하여 뷰로부터 생성된 클래스에 대한 상하위 관계를 생성한다. 이를 위해 우리는 먼저 뷰를 그림 7과 같이 FROM 절과 집합연산의 두 가지 형태로 분류한다.

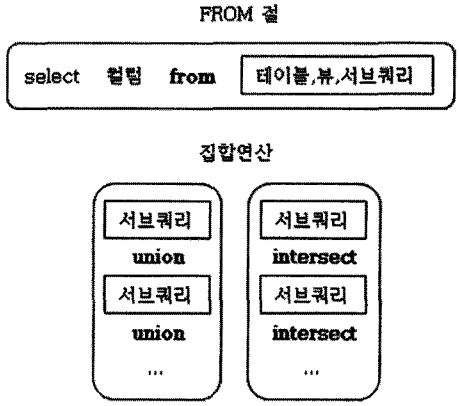

그림 7 뷰의 형태

먼저 뷰 정의의 FROM 절로부터 생성할 수 있는 상하위 관계는 아래와 같이 정의한다.

정의 5.2 FROM 절 상하위 관계: 다음 조건을 만족하는 $X \in U, Y$로부터 생성된 클래스 $c_i = class(X)$, $c_j = class(Y)$는 상하위 관계 (c_i, c_j)이다.

$$Y \in source(X) \wedge (Y \in T \wedge pkey(Y) \in parent(vpkey(X)) \vee Y \in U - \{X\} \wedge vpkey(Y) = parent(vpkey(X)))$$

정의 5.2에서 상하위 관계 (c_i, c_j)는 분석중인 뷰 X로부터 생성된 클래스 c_i가 FROM 절의 테이블이나 뷰 Y로부터 생성된 클래스 c_j의 하위 클래스임을 의미한다. 이때, X의 주키가 FROM 절에 있는 테이블이나 뷰인 Y의 주키임을 만족해야 한다. 여기서 $X \in U$이므로 정의 5.1에 의해 X의 결과 컬럼은 하나의 주키만 가지고 있다.

예를 들어, 그림 8과 같이 뷰 *Ustudent_male*의 결과 컬럼은 FROM 절에 있는 테이블 *Ustudent*의 주키를

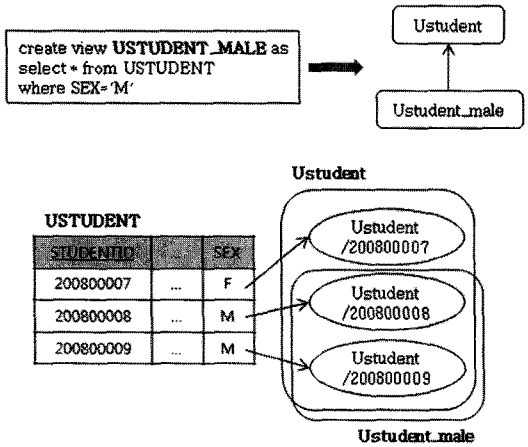

그림 8 FROM 절의 상하위 관계

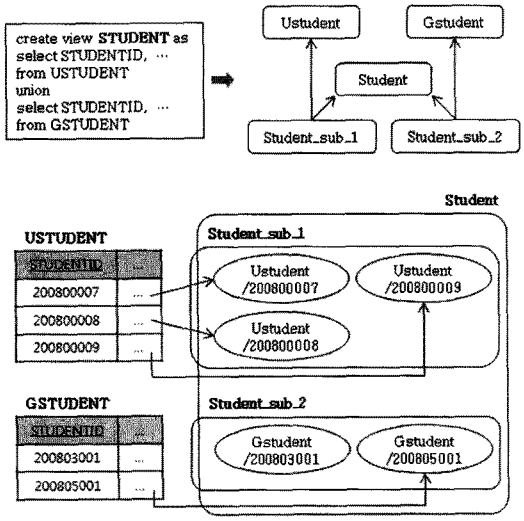

그림 9 집합연산 UNION의 상하위 관계

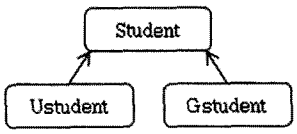

그림 10 집합연산 UNION의 예상되는 상하위 관계

가지고 있으며 그 외의 주키는 가지고 있지 않다. 그러므로 뷰 *Ustudent_male*은 정의 5.1에 의해 클래스로 생성되며, 정의 5.2에 의해 테이블 *Ustudent*로부터 생성된 클래스의 하위 클래스가 된다.

다음으로 뷰 정의의 집합연산으로부터 생성할 수 있는 상하위 관계는 아래와 같이 정의한다.

정의 5.3 집합연산 상하위 관계: $X \in U$, $Y \in U - \{X\}$ 에 대해 다음 조건을 만족하는 X, Y 로부터 생성된 클래스 $c_i = class(X)$, $c_j = class(Y)$ 는 집합연산 UNION/INTERSECT의 경우 상하위 관계 $(c_j, c_i)/(c_i, c_j)$이다.

$$\forall Y \in source(X)(vpkey(Y) = parent(vpkey(X)))$$

정의 5.3에서 상하위 관계 $(c_j, c_i)/(c_i, c_j)$는, 분석중인 뷰 X로부터 생성된 클래스 c_i가, 집합연산 UNION/INTERSECT에 참여하는 서브쿼리 Y로부터 생성된 클래스 c_j의 상/하위 클래스임을 의미한다. 정의 5.3은 정의 5.2와 마찬가지로 X의 주키가 집합연산에 참여하는 서브쿼리의 주키임을 만족해야 한다. 한 가지 차이점은, 집합연산에 참여하는 모든 서브쿼리에 대해서 조건을 만족해야 한다는 점이다.

예를 들어, 그림 9와 같이 뷰 *Student*의 결과 컬럼은 집합연산 UNION에 참여하는 서브쿼리 *Student_sub_1*, *Student_sub_2*의 공통된 주키를 가지고 있으며 그 외의 주키는 가지고 있지 않다. 그러므로 뷰 *Student*는 정의 5.1에 의해 클래스로 생성되며, 정의 5.3에 의해 서브쿼리 *Student_sub_1*, *Student_sub_2*의 상위 클래스가 된다.

그러나 그림 9의 결과는 이상해 보인다. 뷰 정의를 보면, 예상되는 결과는 그림 10과 같아야 하기 때문이다. 결과가 이와 같이 나온 이유는 서브쿼리가 참조하는 테이블 *Ustudent*와 서브쿼리 *Student_sub_1*의 결과가 동일한지 아닌지 판별할 수 없기 때문이다. 이와 관련하여 쿼리 포함(query containment)이나 쿼리 동등(query equivalence), 그리고 [12]와 같은 연구들이 있으며, 이러한 연구들의 결과를 이용하여 위의 문제를 해결한다. 이러한 연구들은 뷰의 모든 결과 컬럼을 이용하여 쿼리 동등 여부를 판별하는 방법을 제안하지만, 우리의 연구에서는 뷰로부터 생성된 클래스의 상하위 관계 생성시 주키를 이용하므로 주키에 대해서만 해당 방법을 적용하면 된다.

5.3 알고리즘

지금까지 살펴본 정의들을 토대로 표 2의 알고리즘을 통해 관계형 데이터베이스 스키마로부터 온톨로지를 생성한다. 여기서 정의 4.2를 통해 클래스로 생성되는 테이블들의 집합 T, 그리고 그로부터 생성된 클래스들의 집합 C_T는 주어졌다고 가정하며, 정의 4.3을 통해 생성된 상하위 관계들의 집합 S_T도 주어졌다고 가정한다. 또한 뷰 정의들 중 재귀적인 뷰 정의는 없다고 가정한다.

알고리즘은 뷰 전체집합 V가 주어진 경우 뷰로부터 생성되는 클래스와 상하위 관계를 추가적으로 생성한다. 그리고 알고리즘의 결과는 뷰로부터 생성되는 클래스들의 집합 C_U와 상하위 관계들의 집합 S_U이다.

알고리즘에서 L은 분석 완료된 테이블이나 뷰들의 집합이며 알고리즘 수행 전에 T의 원소들로 초기화된다.

표 2 뷰로부터 클래스와 상하위 관계 생성

```
알고리즘 VIEW2RDF(Source)
    Input: V
    Output: C_U, S_U
    U ← ∅, L ← T, C_U ← ∅, S_U ← ∅
01  for each v ∈ Source
02    if v ∉ L then
03      l ← source(v)
04      VIEW2RDF(l)
05      if |vpkey(v)| = 1 then
          if 뷰 v가 집합연산 UNION이고
06            ∀t ∈ l (t ∈ T ∪ U ∧ vpkey(t) = parent(vpkey(v))) then
07          c ← class(v)
08          S_U ← S_U ∪ {(class(t), c)|t ∈ l}
          else if 뷰 v가 집합연산 INTERSECT이고
09            ∀t ∈ l (t ∈ T ∪ U ∧ vpkey(t) = parent(vpkey(v))) then
10          c ← class(v)
11          S_U ← S_U ∪ {(c, class(t))|t ∈ l}
          else if t ∈ l ∧ (t ∈ T ∧ pkey(t) ∈ parent(vpkey(v))
12            ∨ t ∈ U ∧ vpkey(t) = parent(vpkey(v)))이고
              뷰 v가 t ∈ l와 같지 않다면 then
13          c ← class(v)
14          S_U ← S_U ∪ {(c, class(t))|t ∈ l}
15      U ← U ∪ {v}
16      C_U ← C_U ∪ {c}
17    L ← L ∪ {v}
```

또한 알고리즘은 뷰 정의에 사용된 다른 테이블이나 뷰와의 의존 관계를 고려하여 만들어졌다. 그러므로 뷰 정의를 분석하기 전에 뷰 정의에 사용된 다른 테이블이나 뷰를 먼저 분석한다(04). 그리고 만약 뷰로부터 클래스를 생성할 수 있다면(05), 해당 뷰가 집합연산(06, 09)인지 FROM 절(12)인지 판별한 후, 클래스와 상하위 관계를 생성한다. FROM 절의 경우 뷰 v가 상위 클래스가 될 테이블이나 뷰 t와 같은지 판별하고, 같지 않은 경우 해당 클래스와 상하위 관계를 생성한다.

우리가 제안한 알고리즘과 정의들을 통해 최종 생성된 온톨로지는 클래스들의 집합 $C = C_T \cup C_U$와 상하위 관계들의 집합 $S = S_T \cup S_U$이다.

6. 결과 분석

이 장에서는 뷰 정의를 분석하여 추가적인 클래스와 상하위 관계를 생성하는 방법의 결과를 분석하기 위해 표 1의 예제 데이터베이스 스키마에 표 3의 뷰 정의들을 추가한 스키마에 대한 결과 분석을 수행한다. 또한 실제 응용프로그램의 데이터베이스 스키마에 우리가 제안한 방법을 적용하여 생성된 온톨로지에 대한 결과 분석을 수행하고, 제안한 방법이 추가적인 클래스와 상하위 관계를 생성함을 보인다.

6.1 예제 데이터베이스 스키마

예제 데이터베이스 스키마는 7개의 테이블과 13개의 뷰 정의로 구성되어있다. 우리가 제안하는 방법을 통해 예제 데이터베이스 스키마로부터 생성된 클래스와 상하위 관계는 그림 11과 같다. 먼저 정의 4.2에 의해 7개의 테이블 중 6개의 테이블로부터 클래스가 생성되었으며(*Person, Professor, Ustudent, Gstudent, Dept, Course*), 정의 4.3에 의해 상하위 관계 1개(*Professor subClassOf Person.*)가 생성되었다. 나머지 하나의 테이블 *Takes*는 정의 1에 의해 이진관계 테이블이므로 클래스로 생성되지 않았다. 다음으로 정의 5.1에 의해 13개의 뷰중 결과 컬럼에 주키 하나를 가지고 있는 11개의 뷰로부터 클래스가 생성되었다(*Ustudent_male, Ustudent_female, Ustudent_grade_1, Ustudent_grade_2, Ustudent_grade_3, Ustudent_grade_4, Ustudent_cs, Student, Student_cs, Gstudent_master, Student_dept_cnt*). 클래스로 생성된 뷰들은 성별 학부 학생, 학년별 학부 학생, 학과별 학부 학생, 전체 학생, 학

표 3 뷰 정의

뷰 정의
create view **USTUDENT_GRADE_1** as select * from USTUDENT where GRADE=1
create view **USTUDENT_GRADE_2** as select * from USTUDENT where GRADE=2
create view **USTUDENT_GRADE_3** as select * from USTUDENT where GRADE=3
create view **USTUDENT_GRADE_4** as select * from USTUDENT where GRADE=4
create view **USTUDENT_FEMALE** as select * from USTUDENT where SEX='F'
create view **USTUDENT_CS** as select STUDENTID, GRADE from USTUDENT, DEPT where USTUDENT.DEPT=CODE and NAME='CS'
create view **STUDENT_CS** as select STUDENTID, NAME from STUDENT, DEPT where STUDENT.DEPT=DEPT.CODE and DEPT.NAME='CS'
create view **STUDENT_DEPT_CNT** as select CODE, NAME, COUNT(*) as CNT from DEPT, STUDENT where DEPT.CODE=STUDENT.DEPT group by CODE
create view **GSTUDENT_MASTER** as select STUDENTID, NAME, RESULT, ADVISER from GSTUDENT, PERSON where PERSONID = ID and DEGREE='Master'
create view *GSTUDENT_PHD* as select * from GSTUDENT, PERSON where PERSONID = ID and DEGREE='PhD'

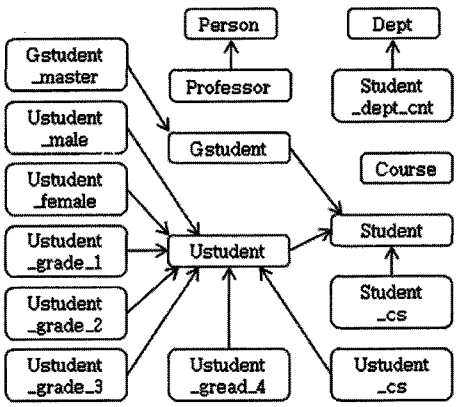

그림 11 예제 데이터베이스 스키마로부터 생성된 온톨로지 스키마

과별 전체 학생, 석사과정 학생 등 클래스로써 의미가 있는 뷰들이다. 그러나 *Student_dept_cnt*의 경우 학과별 전체 학생 수를 결과로 가지는 뷰로, 클래스로써 의미는 없으나 결과 컬럼에 Dept 테이블의 주키 컬럼을 가지고 있으므로 클래스로 생성되었다.

클래스로 생성되지 않은 나머지 2개의 뷰중 *Ustudent_grade_cnt*는 학년별 학부 학생수를 결과로 가지는 뷰로, 클래스로써 의미가 없으며 결과 컬럼에 주키를 가지고 있지 않으므로 클래스로 생성되지 않았다. 그러나 *Gstudent_phd*의 경우 박사과정 학생을 결과로 가지는 뷰로, 클래스로써 의미가 있지만 결과 컬럼에 주키를 2개 가지므로 이진관계를 표현하는 것으로 판별되어 클래스로 생성되지 않았다.

우리가 제안한 방법은 뷰의 결과 컬럼에 주키 정보가 제대로 되어있는 경우 올바른 결과가 나오지만, 주키 정보만을 이용하여 뷰 정의를 분석하므로 *Student_dept_cnt*, *Gstudent_phd*와 같이 올바르지 않은 결과가 나왔다.

6.2 응용프로그램 데이터베이스 스키마

결과 분석을 위해 사용된 응용프로그램은 Open ERP[13]이며, 기본 설치 과정을 통해 설치하였다. 해당 응용프로그램을 설치 한 후 생성된 데이터베이스 스키마는 모두 249개의 테이블과 32개의 뷰 정의로 구성되어 있다.

우리가 제안한 방법과 D2RQ를 통해 생성된 클래스와 상하위 관계 결과는 표 4와 같다.

표 4에서 확인할 수 있듯이 우리가 제안한 방법과 D2RQ를 사용하여 클래스로 생성된 테이블은 전체 249개 중 206개로 동일하다. 클래스로 생성되지 않은 나머지 테이블들은 주키가 없는 테이블들이며, 테이블들 사이의 상하위 관계는 생성되지 않았다. 이는 주키이면서 외부키인 컬럼을 가지고 있는 테이블이 없기 때문이다.

우리가 제안한 방법과 D2RQ 사이의 차이점은 뷰를 분석하는 과정에서 드러난다. 먼저 우리가 제안한 방법을 사용하여 32개의 뷰를 분석하는 과정에서 14개의 서브쿼리가 뷰로 추가되었다. 그로 인해 총 46개의 뷰로부터 16개의 클래스가 생성되었으며 11개의 상하위 관계가 생성되었다. 생성된 클래스와 상하위 관계의 의미를 분석한 결과는 다음과 같다.

먼저 16개의 클래스 중 클래스로서 의미를 가진 뷰나 서브쿼리는 5개이며, 다른 11개는 집계 함수를 사용한 뷰에서 생성된 클래스이다. 이 11개의 클래스들은 결과 컬럼에 하나의 주키를 가지고 있어서 클래스로 생성되었으며, 클래스로서의 의미는 없다. 그리고 나머지 30개의 뷰는 결과 컬럼에 주키가 없거나 여러 개인 경우, 또는 집계 함수를 사용한 경우로, 이러한 뷰는 클래스로 생성되지 않았다. 또한 앞서 생성된 11개의 상하위 관계 중 6개는 클래스로써 의미가 없는 뷰와 관련된 상하위 관계이다.

표 5 의미있는 클래스로 생성되는 뷰 정의

```
create view REPORT_CLOSED_TASK
select TSK.ID, TSK.SEQUENCE, TSK.NAME, TSK.PROJECT_ID,
TSK.USER_ID, TSK.DATE_DEADLINE, TSK.PLANNED_HOURS,
TSK.DELAY_HOURS, TSK.PROGRESS, TSK.PRIORITY,
TSK.STATE, TSK.REMAINING_HOURS, TSK.DATE_CLOSE
from PROJECT_TASK TSK
where ((TSK.DATE_CLOSE <= ('now'::text)::date)
AND (TSK.DATE_CLOSE > (('now'::text)::date - 15)))
```

의미있는 클래스로 생성되는 뷰와, 그와 관련된 상하위 관계의 예로 표 5의 뷰 정의를 들 수 있다.

표 5의 뷰 *Report_closed_task*는 마감 된 작업들 중 마감일로부터 15일 이내의 작업들에 대한 정보를 결과로 가지는 뷰이다. 여기서 뷰 *Report_closed_task*는 결과 컬럼에 하나의 주키를 가지고 있으므로 클래스로 생성되었으며, 테이블 *Project_task*의 하위 클래스가 되었다.

D2RQ의 경우 단순히 32개의 뷰를 모두 클래스로 생성하였으며 상하위 관계는 생성하지 않았다. D2RQ에서는 뷰에 대한 주키를 정의하고 있지 않으므로 리소스가 될 레코드들의 URI를 생성하기 위해 각각의 뷰 마다 특정 컬럼을 직접 지정해줄 것을 요구한다. D2RQ를 사용하는 경우 우리가 제안한 방법에 비해 더 많은 클래스를 생성하는 것처럼 보이나 이들 중 대부분은 의미없는 클래스였으며 URI 생성을 위한 컬럼을 지정하는 추가적인 작업을 필요로 한다. 또한 상하위 관계 역시 직접 생성해야 한다.

따라서 비교 결과, 우리가 제안한 방법이 뷰로부터 클래스 생성시 의미있는 클래스들을 생성함을 알 수 있으며, 또한 상하위 관계와 클래스에 속한 리소스들의 URI를 자동으로 생성함을 알 수 있다.

7. 결론 및 향후 연구

관계형 데이터베이스로부터 온톨로지를 자동으로 생성하는 기존의 연구들은 데이터베이스 스키마와 저장된 데이터 분석을 통한 생성에 대한 연구들이 주를 이룬다. 그러나 기존의 연구들은 데이터베이스 스키마에 테이블

표 4 응용프로그램 데이터베이스 스키마로부터 생성된 온톨로지 결과

	테이블			뷰		
	전체	클래스	상하위 관계	전체	클래스	상하위 관계
제안한 방법	249	206	0	46	16	11
D2RQ	249	206	0	32	32	0

과 제약조건뿐만 아니라 뷰 정의가 있음에도 불구하고 이를 활용하지 않는다. 그러므로 관계형 데이터베이스로부터 온톨로지를 자동으로 생성하는 분야에서 뷰 정의를 분석하여 추가적인 클래스와 상하위 관계를 생성하는 연구는 충분히 의미를 가진다.

본 논문에서는 기존의 방법들에 대한 분석 및 정형화를 통해 클래스로 생성되는 테이블의 특징을 알아내고, 주키 정보를 이용하여 뷰 정의로부터 클래스를 생성한다. 또한 뷰 정의 분석을 통해 뷰로부터 생성된 클래스의 상하위 관계를 생성한다. 그리고 우리가 제안한 방법을 이용하여 데이터베이스 스키마로부터 온톨로지를 생성하고, 이에 대한 결과 분석을 통해 우리가 제안한 방법이 뷰 정의로부터 의미 있는 클래스와 상하위 관계를 추가로 생성함을 보였다.

우리가 제안하는 방법은 주키 정보만을 이용하므로 주키 이외의 다른 정보들을 이용하는 뷰 정의 분석 방법에 대한 연구가 필요하며, 이를 바탕으로 RDFS에서 OWL로의 확장에 대한 연구가 필요하다. 그리고 뷰 정의는 SQL 쿼리이므로 뷰 정의나 쿼리를 이용하여 동적으로 온톨로지를 생성하는 연구가 필요하다.

참 고 문 헌

[1] Bin He, M.P., Zhen Zhang, Kevin Chen-Chuan Chang, *Accessing the deep web.* Communications of the ACM, 2007. 50(5): pp.94-101.

[2] Byrne, K. *Having Triplets - Holding Cultural Data as RDF.* in *Proceedings of the ECDL 2008 Workshop on Information Access to Cultural Heritage.* 2008. Aarhus, Denmark.

[3] Christian Bizer, R.C., *D2RQ - Lessons Learned.* 2007: Position paper for the W3C Workshop on RDF Access to Relational Databases.

[4] Nadine Cullot, R.G., Kokou Yétongnon, *DB2OWL: A Tool for Automatic Database-to-Ontology Mapping*, in *SEDB.* 2007.

[5] Syed Hamid Tirmizi, J.S., Daniel Miranker, *Translating SQL Applications to the Semantic Web*, in *DEXA.* 2008.

[6] Cerbah, F., *Learning Highly Structured Semantic Repositories from Relational Databases: The RDBToOnto Tool*, in *ESWC.* 2008.

[7] Man Li, X.D., Shan Wang, *A Semi-automatic Ontology Acquisition Methods for the Semantic Web*, in *WAIM.* 2005.

[8] Satya S. Sahoo, W.H., Sebastian Hellmann, Kingsley Idehen, Ted Thibodeau Jr, Sören Auer, Juan Sequeda, Ahmed Ezzat, *A Survey of Current Approaches for Mapping of Relational Databases to RDF.* 2009, W3C RDB2RDF Incubator Group.

[9] Berners-Lee, T. *Relational Databases on the Semantic Web.* 1998; Available from: http://www.w3.org/DesignIssues/RDB-RDF.html.

[10] Shekar Ramanathan, J.H., *Extraction of Object-Oriented Structures from Existing Relational Databases.* ACM SIGMOD, 1997.

[11] Dan Brickley, R.V.G. *RDF Vocabulary Description Language 1.0: RDF Schema.* 2004 10 February 2004; Available from: http://www.w3.org/TR/rdf-schema/.

[12] Halevy, A.Y., *Answering queries using views: A survey.* The VLDB Journal, 2001. 10(4): pp.270-294.

[13] *Open ERP.* Available from: http://www.openerp.com/.

양 준 석

2008년 전북대학교 전자정보공학부 학사 2010년 서울대학교 전기컴퓨터공학부 석사. 관심분야는 데이터베이스, 시맨틱 웹, 온톨로지

김 기 성

2003년 서울대학교 응용화학부 학사. 2006년~2009년 티맥스소프트. 2003년~현재 서울대학교 컴퓨터공학부 박사과정 재학 중. 관심분야는 데이터베이스, 시맨틱 웹, 분산 병렬 데이터 처리

김 형 주

정보과학회논문지 : 데이타베이스 제 37 권 제 4 호 참조