「정보시스템연구」 제18권 제3호
한국정보시스템학회
2009년 9월, pp. 327~349

토픽맵의 다중역할 토픽 보존을 위한 관계형 데이터베이스 구조*

정윤수** · 이춘열*** · 김남규****

〈목 차〉

Ⅰ. 서 론

최근 정보 기술의 발달로 인해 정보의 생성, 배포, 저장 과정에 활용되는 매체의 디지털화가 급속하게 촉진되었다. 이로 인해 일반 사용자가 접근할 수 있는 정보의 양이 기하급수적으로 증가하였으며, 이들 대부분의 정보는 웹 상에서 검색 및 획득이 가능하다. 하지만 이처럼 접근 가능한 정보의 양이 방대해짐에 따라, 정보에 내재된 시멘틱(Semantic)을 고려하지 않은 기존의 단순한 검색 및 저장 방식은 사용자의 의도와 무관한 결과를 양산한다는 한계를 나타내게 되었다. 따라서 시멘틱 정보를 효율적으로 저장하기 위한 많은 연구(김하균, 2004)가 수행되었으며, 이를 위한 핵심 요소 중의 하나로 온톨로지(Ontology) 관련 기술이 사용되고 있다. 온톨로지란 특정한 영역에 속하는 속성들을 정의하고 그들 간의 관계를 추상화하여 나타낸 모델로, 개념과 관계, 속성을 표현하여 데이터를 지식화 함으로써 시맨틱 검색의 구현을 위한 도구로 사용될 수 있다. 온톨로지의 기술을 위한 언어(Mizoguchi, 2004)로는 W3C의 RDF(Resource

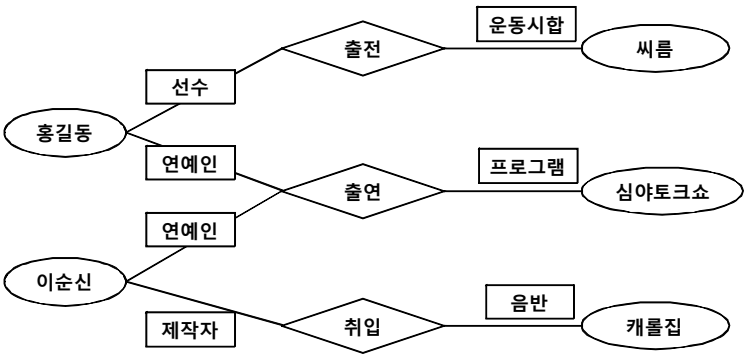
* 본 연구는 2009년도 국민대학교 신진교수 연구지원금으로 수행됨
** 국민대학교 비즈니스IT전문대학원 석사과정, maestrojung@naver.com
*** 국민대학교 비즈니스IT학부 교수, cylee@kookmin.ac.kr
**** 국민대학교 비즈니스IT학부 조교수(교신저자), ngkim@kookmin.ac.kr

Definition Framework), OWL(Web Ontology Language), DAML+OIL 등이 사용되고 있으며, 특히 최근에는 ISO의 토픽맵(Topic Maps)을 사용하여 온톨로지를 기술하기 위한 시도가 활발하게 이루어지고 있다.

토픽맵은 주제들의 집합을 모형화하는 하나의 도구로서 토픽(Topic), 토픽 타입(Topic Type), 관계(Association), 정보자원(Occurrences)의 요소를 사용하여 토픽, 토픽 간의 관계, 그리고 토픽에 대한 정보자원의 추론 규칙을 정의하고 정의된 구조와 지식 자원을 연계한다. 초기의 토픽맵은 서적의 인덱스, 용어집, 시소러스 등에만 한정적으로 적용되었지만, 최근 온톨로지 개념이 대두되면서 그 적용 영역이 크게 확장되었다. 또한 토픽맵은 특정 주제 영역의 용어 간 의미 관계를 표현함으로써 원하는 정보의 검색을 용이하게 하고, 정보관리 및 지식 표현 측면에서 각 응용분야 간 다리 역할을 한다는 측면에서 대용량 데이터의 네비게이션(Holger, 2000)을 위한 솔루션으로도 그 가치를 인정받고 있다.

토픽맵의 구성요소를 설명하기 위한 간단한 예가 <그림 1>에 나타나있다. <그림1>은 "홍길동"과 "이순신"이라는 사람이 "선수", "연예인", "제작자" 등의 역할로 활동하는 경우를 추상화하여 나타낸 토픽맵이다. 토픽맵은 각 토픽이 참여하는 역할의 의미를 토픽 타입으로 정의하여 관리한다. 즉 "출전"이라는 관계에는 "홍길동"이라는 토픽이 "선수"라는 토픽 타입으로 참여하고, "씨름"이라는 토픽이 "운동시합"이라는 토픽 타입으로 참여하고 있다. 그림에서 보는 바와 같이, "홍길동"이 "선수" 토픽 타입으로 참여하건 "연예인" 토픽 타입으로 참여하건, 각 토픽 타입의 실체인 토픽 자체는 "홍길동"으로 동일하다. 이와 같이 동일한 토픽이라고 하더라도, 임의의 응용에서 서로 다른 역할로 참여하는 경우를 충실하게 표현할 수 있다는 측면에서 토픽맵은 큰 장점을 갖는다.

토픽맵이 동일한 토픽이 다중역할(Multi-Role)로 참여하는 경우를 충실하게 표현하는 것과 달리, 기존의 다양한 개념적 설계 도구(정대율, 1998) 중 가장 널리 사용되어 온 개체관계도(ERD: Entity-Relationship Diagram)는 다중역할 표현의 측면에서 표현력의 한계를 갖는다. 비교를 위해 <그림 1>의 토픽맵과 동일한 현상을 표현하기 위한 ERD를 <그림2>에 제시하였다.



<그림 1> 토픽맵의 간단한 예

주민번호	이름	나이	우승회수	승률
123456	홍길동	37	3	75%

(a) 선수 토픽타입을 표현하기 위한 테이블

주민번호	이름	나이	데뷔작	소속사
123456	홍길동	37	시네마천국	H기획
654321	이순신	39	가요무대	L기획

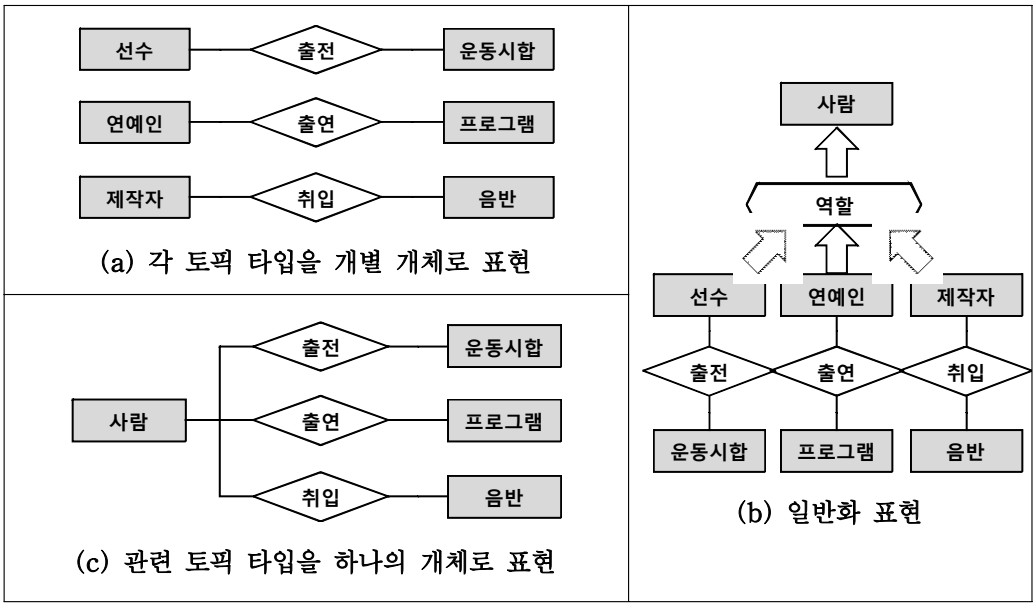
(b) 연예인 토픽타입을 표현하기 위한 테이블

주민번호	이름	나이	투자액	수입배분율
654321	이순신	39	1억원	40%

(c) 제작자 토픽타입을 표현하기 위한 테이블

<그림 3> <그림 2(a)>를 기준으로 도출된 테이블의 예

<그림2(a)>는 세 가지 토픽 타입인 “선수”, “연예인”, “제작자”를 모두 독립 개체로 표현한 ERD이다. 이 경우, 둘 이상의 토픽 타입에 연관된 토픽들은 둘 이상의 테이블에 중복 출현하게 된다. 예를 들어, <그림 3>에서 ″홍길동″은 ″선수″ 테이블과 ″연예인″ 테이블에, 그리고 ″이순신″은 ″연예인″ 테이블과 ″제작자″ 테이블에 각각 두 번씩 출현한다. 이는 곧 여러 토픽



<그림 2> <그림 1>의 정보를 개체관계도로 표현한 예

타입에 연관된 토픽에 대한 모든 정보를 조회하기 위해서는, 연관된 토픽 타입 수만큼의 조인 연산으로 인한 응답 시간 지연 현상이 초래됨을 의미한다. 또한 이 구조는, 각 개체에 공통으로 적용되는 속성들이 여러 테이블에서 중복 관리된다는 한계도 갖는다. 예를 들어 <그림 3>에서 "주민번호", "이름", "나이"는 세 테이블에서 모두 관리된다. 이러한 한계를 극복하기 위한 구조가 <그림 2(b)>에 제시된 일반화(Generalization)를 적용한 구조이다. 이 구조는 개체간 공통된 속성을 상위 개체로 일반화함으로써, 앞서 언급된 두 가지 한계점 중 후자를 극복할 수 있다. 하지만 여러 토픽 타입에 참여하는 토픽이 여러 테이블에 출현하므로, 토픽에 대한 모든 정보 조회 시 응답 시간 지연 현상이 초래된다는 한계점은 여전히 존재한다. 마지막으로 살펴보는 구조는 관련된 토픽 타입을 모두 합쳐서 하나의 개체로 표현하는 방법으로, <그림 2(c)>에 나타나있다. 이 구조는 앞에 언급한 두 가지 한계점은 갖지 않지만, 널(NULL) 값을 많이 포함할 우려가 있고(그림 4), 각 속성들이 어떤 토픽 타입에 연관되는지 여부가 명확하게 표현되지 않음으로써 의미상 손실이 있다는 한계를 갖게 된다.

토픽맵과 ERD는 데이터베이스 전체 설계 단계 중 개념적 설계에 해당되는 모델로서, 실제 서비스로 구현되기 위해서는 두 모델 모두 논리적 모델로의 변환이 이루어져야 한다. <그림 2>에서 살펴본 바와 같이 ERD는 토픽맵 상의 정보자원을 충분히 표현할 수 없다는 한계를 갖고 있음에도 불구하고 개념적 설계 단계의 대표적 도구로 사용되고 있는데, 그 이유는 ERD를 논리적 모델 중 가장 널리 사용되는 관계형 모델로 변환하는 과정이 매우 쉽고 정형화되어 있기 때문이다. 반면 토픽맵은 ERD에 비해 풍부한 표현력을 갖고 있지만 토픽맵의 실제 구현을 위한 논리적 단계의 자체 모델은 아직 사용되고 있지 않으며, 이 부분은 토픽맵 사용의 확장에 큰 저해 요소로 작용한다. 따라서 현 시점에서 토픽맵의 활용 가능성을 높이기 위해서는, 대표적 논리적 모델인 관계형 데이터베이스(RDB: Relational Database)의 구조 하에서 토픽맵을 저장하는 방법이 속히 고안되어야 할 것으로 사료된다. 즉, 토픽맵을 RDB로 연계하기 위한 저장 구조 및, 이를 운용하기 위한 질의 처리 방식의 모색이 반드시 필요하다.

본 연구의 핵심은 크게 두 가지로 요약될 수 있다. 첫 번째는 토픽맵의 표현력을 손상시키지 않으면서 RDB로 저장할 수 있는 매핑 구조를 제안하는 것이다. 그리고 두 번째는, 의미적 모델인 토픽맵과 실제 저장 구조인 RDB 간의 독립성을 제공하기 위해 질의 변환기를 설계하여

주민번호	이름	나이	우승회수	승률	데뷔작	소속사	투자액	수입배분율
123456	홍길동	37	3	75%	시네마천국	H기획	NULL	NULL
654321	이순신	39	NULL	NULL	가요무대	L기획	1억원	40%

<그림 4> <그림 2(c)>를 기준으로 도출된 테이블의 예

제안하는 것이다. 즉, 사용자가 토픽맵에서 표현된 의미에 입각해서 질의를 요청하면, 질의 변환기에서 이 질의를 실제 RDB 구조에 맞게 재구성하여 기존의 질의 처리기로 전달하는 것이다. 본 논문의 이후 구성은 다음과 같다. 2장에서는 본 연구와 관련된 기존의 연구 성과를 간략히 소개하고 3장에서는 토픽맵를 RDB로 매핑하기 위한 저장 구조 및 이 구조를 위한 질의 변환기의 핵심 알고리즘을 제시한다. 설계된 질의 변환기의 간단한 적용 사례가 4장에 소개되어 있으며, 마지막 장인 5장에서는 본 연구의 한계점과 향후 연구 방향을 제시한다.

Ⅱ. 관련연구

온톨로지는 해당 분야의 기본 개념에 대한 정의 및 그들 간의 관계에 대한 명세(고창택, 2007; Gruber, 1995)로 표현된다. 온톨로지는 물론 독립된 형태로 구축되어 있을 수도 있지만, 독립된 형태가 아닌 데이터베이스나 프로그램 코드에 내재된 형태로 존재할 수도 있다. 어휘사전이나 용어모음 등이 온톨로지의 가장 단순한 형태로 인식될 수 있지만, 컴퓨터가 처리할 수 있는 수준의 구조성과 구체성을 갖춰야 비로소 온톨로지로 인정받는 것이 일반적이다. 온톨로지의 기술을 위해 RDF, OWL, 토픽맵, DAML+OIL 등의 많은 언어들이 고안되어 왔으며, 이들 간의 비교 연구(Moore, 2000 이혜원, 2005)도 활발하게 수행되었다. W3C에서 제안한 RDF가 특정 자원의 메타데이터를 기술하는 언어(Lassila and Swick, 1999)인 것과 달리, 토픽맵은 주제를 중심으로 기술되는 언어(한국전자거래진흥원, 2003)라는 점에서 지식 구조의 표현에 있어서는 보다 바람직한 특성을 갖는 것으로 알려져 있다. 최근에는 서로 다른 언어인 RDF와 토픽맵으로 기술된 데이터를 통합하고 이를 상호 운용하기 위한 표준 지침(Pepper et al., 2006)이 제안되기도 하였다.

토픽맵은 다양한 토픽들을 분류하고, 토픽들 간의 관계를 설정하고, 그 결과를 온톨로지화하여 관리하기 위한 모형으로, 2000년에 세계 표준으로 채택되고 2002년에 그 개정판이 발표되었다. ISO는 토픽맵 기술을 위한 XML기반 표준 언어로 XTM(XML Topic Map) 1.0(Pepper and Moore, 2001)을 제안한 바 있다. 토픽맵은 정보의 주제나 개념을 나타내는 토픽, 토픽과 토픽의 연관성을 설정하는 관계, 그리고 실제 정보의 위치를 나타내는 정보자원으로 구성되어 있으며, 이들 3가지 구성요소의 타입(Type), 롤(Role), 범위(Scope)를 정의하는 형태로 모델링을 수행하게 된다. 각 토픽은 하나 이상의 관련 정보의 실체와 연결될 수 있으며, 이 실체를 정보자원이라고 명명한다. 정보자원은 시스템의 내부 또는 외부에 존재할 수 있으며, 인터넷 상의 정보를 포함한 모든 형태의 자원이 정보자원으로 사용될 수 있다(한국전자거래진흥원, 2003).

국내의 토픽맵 관련 연구는 주로 온톨로지 통합에 초점을 맞추고 진행되어 왔다. 지식관리 시스템의 단위 시스템에서 사용하는 지식맵을 토픽맵 기술 언어인 XTM을 사용하여 통합하기 위한 시도가 정호영 등(2003)에서 이루어졌으며, XML의 의미와 구조의 변화 없이 유사 스키마를 통합하기 위한 방법은 강혜란, 이경호(2008)에서 제시되었다. 한편 김정민 등(2006)

은 토픽맵을 사용하여 온톨로지를 통합할 때 발생할 수 있는 충돌의 유형을 분류하여 소개하였으며, 이를 탐지하고 해결하기 위한 기법을 제안하였다. 토픽맵을 이용하여 온톨로지를 구축하는 과정에서 용어 분류 작업을 자동화하기 위한 연구(구미숙 등, 2006)도 최근 수행되었다. 이 연구에서는 유사도메인의 XML문서 집합으로부터 데이터 마이닝 연관규칙 알고리즘을 이용하여 반자동으로 온톨로지를 구축하는 방법이 제안되었다. 그 외에도 다양한 토픽맵 매핑 방법들을 비교하기 위한 연구 및 무결성 검증에 관한 연구가 많이 시도되었는데, RDB와 RDF 간의 매핑 접근법에 대한 비교는 Sahoo et al.(2009)에서 이루어졌다. Rahm and Bernstein(2001)에서는 온톨로지의 스키마레벨, 인스턴스레벨, 엘레먼트레벨, 구조레벨, 언어레벨 별 제약기반 무결성 검사알고리즘을 구현하였으며, Dhanapalan and Chen(2007)는 RDF 스키마 병합 과정에서 데이터베이스 스키마의 의미적 불일치를 발견 및 해결하기 위한방안을 제시하였다.

기존 대부분의 토픽맵 관련 연구들은 토픽맵 설계 자체에는 많은 관심을 기울였지만, 이를 RDB로 저장하는 과정에서는 단순히 토픽맵의 모든 토픽 및 관계를 RDB의 테이블로 각각 대응시키는 방법을 사용하였다. 일례로 정현숙(2005)은 ERP 시스템을 위한 물품 온톨로지를 구축하는 과정에서 토픽맵을 RDB로 저장하는 예를 보였는데, 각 토픽은 부모 토픽으로부터 상속되며 모든 토픽과 관계는 각각 테이블로 대응되는 구조를 사용하였다. 이러한 변환은 곧 <그림 2(b)>의 ERD로부터 도출된 테이블로의 변환을 의미하며, 이 경우 <그림 3>에서 나타난 단점을 그대로 나타내게 된다. 즉 둘 이상의 토픽 타입에 연관된 토픽들은 둘 이상의 테이블에 중복출현하게 되므로, 이러한 토픽에 대한 모든 정보를 조회하기 위해서는 연관된 토픽 타입 수만큼의 조인 연산으로 인한 응답 시간 지연 현상이 초래된다. 토픽맵을 RDB로 변환하는 방안은 김정민 등(2004)의 후속 연구인 이한준 등(2007)에서 보다 심도 깊게 다루어졌는데, 이 연구는 토픽맵을 토픽과 관계들로 구성된 그래프로 파악하여 의미 정보의 무결성을 RDB 차원에서 검출하기 위한 기법을 제안하였다. 하지만 이 연구는 본 논문에서 다루고자 하는 요소인 다중역할 토픽을 처리하는 방안에 대해 충분히 논의하지 않았다. 또한 기본적으로 여러 역할로 참여하는 토픽들을 여러 테이블에서 관리함으로써, 앞서 언급한 <그림 3>에서 나타난 한계를 마찬가지로 내포하고 있다. 또한 박여삼 등(2008)은 GUI 기반으로 온톨로지를 관리하고, 토픽맵을 레거시 시스템 상의 RDB와 연계하기 위한 도구인 X-TOP을 개발하였다. 하지만 X-TOP의 구조를 나타낸 ERD에서 토픽 개체와 관계 개체간의 직접적인 관계가 설정되어 있지 않은 점에서 알 수 있듯이, 이 연구 역시 다중역할 토픽에 대해서는 구체적인 방안을 제시하지 않았다는 한계를 갖는다. 한편 Cerbah(2008)와 Cullot et al.(2007) 는 역으로 RDB를 온톨로지로 변환하기 위한 도구를 제안하였다. 이들 연구는 역방향 변환을 시도했다는 점에서 참신한 시도로 평가되며, 정방향 연구가 변환 과정에서 의미적 손실을 최소화하는 것에 초점을 두는 것에 반해, 이들은 테이블의 속성 및 데이터 값으로부터 내재된 의미를 추출하는 방안에 초점을 두고 있다.

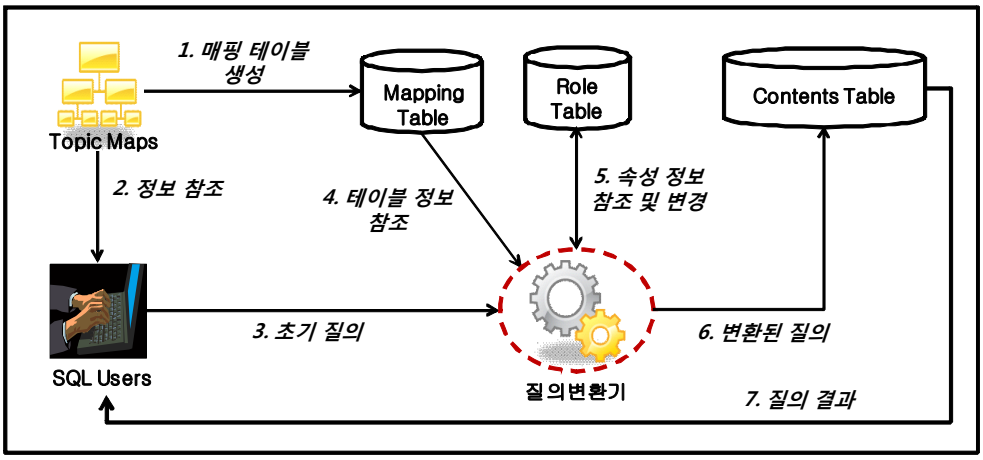
Ⅲ. 다중역할 토픽의 표현을 위한 RDB 저장 구조 및 질의 변환기

3.1. 시스템 모형 및 연구 범위

본 절에서는 다중역할 참여 토픽을 RDB에서 관리하기 위한 전체 모형을 소개한다. <그림5>는 토픽맵으로 표현된 정보를 기존의 RDB로 매핑하고 질의를 변환하기 위한 전체 과정을 도식화한 그림이다.

<그림 5>에서 데이터베이스 설계자는 사용자의 요구사항을 분석하여 토픽맵을 작성한 후, 토픽맵의 RDB 매핑에 관한 메타 정보를 포함하는 매핑 테이블(Mapping Table)을 생성한다. 데이터베이스 관리자(DBA)를 포함한 일반 사용자들은 RDB의 실제 저장 구조가 아닌 토픽맵을 참조하여 질의를 요청한다. 요청된 질의는 데이터가 실제로 저장된 RDB 상에서 수행 가능하도록 재구성되어야 하며, 이 과정은 그림에서 점선으로 표시된 질의 변환기에 의해 수행된다. 즉 사용자가 임의의 데이터에 접근하고자 하는 경우, 변환기가 매핑 테이블과 롤 테이블(Role Table)을 참조하여 실제 수행 가능한 형태의 질의로 자동 변환한 후 컨텐츠 테이블(Contents Table)에 접근하게 된다. 컨텐츠 테이블은 데이터 값이 실제로 저장되는 테이블이고, 매핑 테이블과 롤 테이블은 저장 구조에 대한 메타 정보를 담고 있는 테이블이다. 즉 본 논문에서 제안하는 구조는 1장의 <그림 2(c)>에 제시된 ERD에 기반한 것이다. <그림 2(a)>와 <그림 2(b)>의 구조를 채택하지 않은 이유는, 이미 언급한 바와 같이 이들 두 구조는 다중역할 토픽의 표현 과정에서 테이블간 속성의 중복 또는 인스턴스의 중복 현상을 야기하기 때문이다. 한편 <그림 2(c)>의 구조는 다양한 토픽 타입을 하나의 테이블로 통합하여 표현함으로써 의미 정보가 손실된다는 한계를 갖고 있음을 이미 지적한 바 있는데, 이러한 한계는 매핑 테이블과 롤 테이블을 도입함으로써 극복될 수 있다.

매핑 테이블은 각 토픽이 참여할 수 있는 토픽 타입들이 RDB에서 어떤 컨텐츠 테이블로 구현되었는지의 대응 관계를 명시한 테이블로



<그림 5> 토픽맵의 RDB 저장 및 질의 변환을 위한 모형

서, 최초 RDB 설계시 그 값이 기록되며 질의 처리 과정에서는 그 값이 변하지 않는다. 롤 테이블은 컨텐츠 테이블의 속성들이 각각 어떤 토픽 타입에 연관되었는지를 기록한 테이블로, 추후 스키마 관련 질의에 의해 그 값이 추가 또는 삭제된다. 매핑 테이블은 하나의 응용 전체에 대해 단 하나만 존재하며, 롤 테이블은 각 컨텐츠 테이블별로 하나씩 존재한다. <그림 6>은 매핑 테이블 "Mapping"과 컨텐츠 테이블 "사람", 그리고 이 컨텐츠 테이블을 위한 롤 테이블인 "사람_Role"에 대한 예를 보여준다.

<그림 6>은 1장에서 소개된 <그림 1>의 토픽맵을 RDB로 표현한 예이다. 그림의 RDB에서 "사람" 테이블은 "선수", "연예인", 그리고 "제작자"의 토픽 타입에 관한 의미를 모두 통합하여 관리하고 있다. 이로 인해 사용자가 토픽맵에 근거해서 작성한 질의를 그대로 수행할 경우, RDB로부터 의도하지 않은 정보가 불필요하게 제공될 수 있다. 예를 들어 사용자가 "연예인"으로서의 "홍길동"의 정보를 검색하고자 한다면, 주민번호, 이름, 나이, 데뷔작, 소속사 등의 정보를 반환 받을 것을 기대할 것이다. 하지만 실제로 "사람" 테이블은 위 정보와 함께 우승회수, 승률, 투자액, 수입배분율 등 "연예인" 역할이 아닌 "선수" 및 "제작자" 역할로서의 속성도 함께 반환한다. 따라서, 사용자의 검색에 대해 의미적 연관성이 있는 정보만을 제공하기 위해서는 사용자 질의를 제안하는 구조

사람

주민번호	이름	나이	우승회수	승률	데뷔작	소속사	투자액	수입배분율
123456	홍길동	37	3	75%	씨네마천국	H 기획	NULL	NULL
654321	이순신	39	NULL	NULL	가요무대	L 기획	1억원	40%

Mapping

Topic Type	Implemented Table
선수	사람
연예인	사람
제작자	사람

사람_Role

Attribute	Role
주민번호	선수
이름	선수
나이	선수
우승회수	선수
승률	선수
주민번호	연예인
이름	연예인
나이	연예인
데뷔작	연예인
소속사	연예인
주민번호	제작자
이름	제작자
나이	제작자
투자액	제작자
수입배분율	제작자

<그림 6> 매핑, 롤, 컨텐츠 테이블의 예

에 맞게 재구성하는 역할을 수행하는 질의 변환기가 반드시 필요하다. 즉, 사용자들은 토픽맵에 기반하여 SQL 문을 작성하고, 질의 변환기는 이를 RDB 구조에 맞는 질의로 변환한 후 기존의 질의 처리기에 전달하는 절차를 따른다. 이처럼 매핑 테이블, 롤 테이블, 컨텐츠 테이블로 형성된 구조와 질의 변환기를 통해, 기존 RDB 시스템의 변경을 최소화하면서도 토픽맵이 표현하고 있는 다중역할 토픽을 의미의 손실 없이 저장할 수 있다.

3.2. 질의 변환기 설계

본 절에서는 표준 SQL문으로 작성된 질의를 제안된 구조에 부합되게 변환하기 위한 질의 변환기를 설계하고, 이를 의사 알고리즘(Pseudo Algorithm)을 통해 소개한다. 변환 대상 SQL 질의로는 스키마 생성을 위한 CREATE 문, 인스턴스의 삽입을 위한 INSERT문, 그리고 인스턴스의 반환을 위한 SELECT문을 선택하였다. 이 외에도 대표적인 SQL 구문인 DROP 및 DELETE 등이 있으나, 이들은 변환 원리가 CREATE 및 INSERT와 유사하므로 별도로 소개하지 않는다.

질의 변환기의 메인 프로시져에 대한 의사 알고리즘이 <알고리즘 3.1>에 나타나 있다. 사용자가 작성한 SQL 문장들은 스크립트 형태로 입력되어 *SQL* 이라는 변수에 저장된다(Line 1). 각각의 SQL 문장은 CREATE, INSERT, SELECT 명령 중 한 가지를 수행하는 구문으로 구성되어 있으며, 구문의 종류에 따라 적합한 서브 프로시져를 호출하게 된다(Lines 3 ~ 8). 그 외의 구문이 사용된 경우는 고려하지 않는다(Lines 9 ~ 10).

CREATE 구문을 변환하기 위한 알고리즘은 <알고리즘 3.2>에 나타나 있다. CREATE 구문

<알고리즘 3.1> 메인 프로시져

```
BEGIN
1.    SQL ← a set of SQL sentences inputted by users;
2.    FOR EACH SQL_i in SQL
3.        IF SQL_i contains "CREATE" phrase THEN
4.            Call Tran_Create(SQL_i);
5.        ELSE IF SQL_i contains "INSERT" phrase THEN
6.            Call Tran_Insert(SQL_i);
7.        ELSE IF SQL_i contains "SELECT" phrase THEN
8.            Call Tran_Select(SQL_i);
9.        ELSE
10.           Show error message;
11.       END IF
12.   END FOR
END
```

의 경우 사용자는 토픽 타입(예: <그림 6>의 선수, 연예인, 제작자)을 테이블 명으로 지정하지만, 질의 변환기는 매핑 테이블을 참조하여 이들 토픽 타입의 실체명(예: <그림 6>의 사람)으로 테이블을 생성하게 된다. CREATE 구문은 동일한 테이블이 이미 존재하고 있는지의 여부에 따라 CREATE 또는 ALTER 구문으로 변환되는데, 그 과정이 각각 Lines 5 ~ 7과 Lines 9 ~ 13에 나타나 있다.

<알고리즘 3.2>의 첫 줄은 입력된 SQL 문으로부터 테이블 명(*Table_Name*)을 추출하는 과정을 나타낸다. 이 테이블 명은 사용자가 지정한 토픽 타입을 나타내며, 실제로 구현되는 테이블 명(*New_Name*)과는 차이가 있다. 또한 SQL 문에 포함된 속성의 리스트는 Line 2 에서 저장된다. 사용자가 지정한 토픽 타입이 실제로 구현된 테이블 명은 매핑 테이블을 참조함(Line 3)으로 얻을 수 있으며, 이 과정은 Mapping 프로시져(알고리즘 3.3)에 구현되어 있다. Mapping 프로시져는 토픽 타입을 매개변수로 입력 받아서, 이 토픽 타입이 실제로 구현되는 테이블의 이름을 반환하는 역할을 수행하는 것으로 간단

<알고리즘 3.2> CREATE 구문의 변환을 위한 프로시져

```
Procedure Tran_Create(SQL_i);
BEGIN
1.      Table_Name ← Table name in the SQL_i;
2.      Attr_Set ← A set of attributes in the SQL_i;
3.      New_Name ← Mapping(Table_Name);
4.      New_Name_Role ← Role table for the table New_Name;

5.      IF a table New_Name does not exist THEN
6.              Create table New_Name with attributes Attr_set;
7.              Create table New_Name_Role with attributes (Attribute, Role);
8.      ELSE
9.              FOR EACH Attr_i in Attr_Set
10.                     IF Attr_i does not exist in the table New_Name THEN
11.                             Alter table New_Name add Attr_i;
12.                     END IF
13.             END FOR
14.     END IF

15.     FOR EACH Attr_i in Attr_Set
16.             Insert a record (Attr_i, Table_Name) into New_Name_Role;
17.     END FOR
END
```

히 설명된다. 또한 Line 4에서 *New_Name_Role* 변수는 실제로 구현된 컨텐츠 테이블에 대한 롤 테이블의 이름을 저장한다.

만약 실제로 구현되는 테이블 *New_Name*이 아직 존재하지 않는 테이블이라면, 이에 해당되는 새로운 테이블이 생성되어야 하며, 이 부분은 Lines 5 ~ 7에서 담당한다. 즉 Line 6에서는 주어진 속성들로 구성된 컨텐츠 테이블을 새롭게 생성하며, Line 7에서는 이 컨텐츠 테이블을 위한 롤 테이블을 새롭게 생성한다. 만약 실제로 구현되는 테이블 *New_Name*이 이미 존재하는 테이블이라면, CREATE 구문은 ALTER 구문으로 변경되어 수행되어야 하며, 이 부분은 Lines 9 ~ 13에서 담당한다. 즉 주어진 SQL문에 포함된 속성 중 이미 *New_Name*의 이름으로 존재하고 있는 테이블에 포함되어 있지 않은 속성만을 추가로 생성하는 것이다. CREATE 구문의 변환을 위한 최종 단계는 새로 추가된 속성과 토픽 타입 간의 관계를 롤 테이블에 삽입하는 것으로서, 이 부분은 Lines 15 ~ 17에서 담당한다.

다음으로 INSERT 구문을 변환하기 위한 알고리즘인 <알고리즘 3.4>를 살펴보자. <알고리즘 3.2>와 마찬가지로, 사용자가 지정한 테이블 명이 *Table_Name*에(Line 1), 이를 실제로 구현한 테이블 명이 *New_Name*에(Line 4) 저장되어 있다. 또한 *Val_Set*에는 삽입하고자하는 값들의 리스트가 저장되어 있으며(Line 2), 이들 중 기본키 속성에 해당되는 값들은 *Key_Set*에도 그 값이 저장된다(Line 3). Line 5에서 호출되는 Role 프로시져(알고리즘 3.5)는 *New_Name_Role* 이름의 롤 테이블에서 *Table_Name*이름의 역할로 저장된 속성들의 집합을 반환한다. Line 6의 *Pairs* 변수는 UPDATE 문에 사용될 (속성, 값)의 리스트를 저장하기 위해 생성되었으며, 초기 값은 *NULL*로 할당한다. 만약 삽입되는 인스턴스가 이미 해당 테이블에 존재하는 경우, 즉 다른 토픽 타입에 의해 *New_Name_Role* 테이블에 이미 해당 인스턴스가 존재하는 경우 INSERT 문은 UPDATE 문으로 변경되어 수행되며 이 부분은 Lines 12 ~ 16에서 담당한다. Lines 12 ~ 14는 속성과 값의 조합을 Pairs 변수에 추가하는 과정을, Lines 15 ~ 16은 *Pairs* 변수를 이용하여 UPDATE 구문을 완성하는 과정

<알고리즘 3.3> Mapping Table의 참조를 위한 프로시져

Procedure Mapping(*T_Name*);
BEGIN
1. **IF** *T_Name* exists in the ith row of *Topic_Type* Field in the *Mapping Table* **THEN**
2. **RETURN** the ith value of *Implemented_Table* Field in the *Mapping Table*;
3. **ELSE**
4. Show error message;
5. **END IF**

END

<알고리즘 3.4> INSERT 구문의 변환을 위한 프로시져

```
Procedure Tran_Insert(SQL_i);
BEGIN
1.      Table_Name ← Table name in the SQL_i;
2.      Val_Set ← A set of values in the SQL_i;
3.      Key_Set ← A set of key attribute values in the SQL_i;
4.      New_Name ← Mapping(Table_Name);

5.      Attr_Set ← Role(New_Name_Role, Table_Name);
6.      Pairs ← NULL;

7.      IF no instance in the table New_Name has the same key attributes as Key_Set THEN
8.              Replace Table_Name with New_Name in the SQL_i;
9.              Add predicate Attr_Set to SQL_i;
10.             Perform SQL_i;
11.     ELSE
12.             FOR EACH Attr_Set_i in Attr_Set and Val_Set_i in Val_Set
13.                     Add Attr_Set_i = Val_Set_i to Pairs;
14.             END FOR
15.             SQL_i ← "UPDATE New_Name SET Pairs";
16.             Perform SQL_i;
17.     END IF
END
```

을 나타낸다. 한편 새로운 인스턴스, 즉 어떤 토픽 타입에 의해서건 삽입된 적이 없는 인스턴스의 경우는 Lines 7 ~ 10 에서 삽입한다. 인스턴스의 삽입은 사용자가 지정한 테이블의 이름을 실제 구현된 테이블 이름으로 변경(Line 8)한 뒤, 지정된 토픽 타입에 해당되는 속성들만을 SQL 문의 술어에 추가(Line 9) 시켜 줌으로써 수행된다.

마지막으로 <알고리즘 3.6>은 SELECT 구문의 변환을 다루고 있다. 사용자는 토픽의 특정 역할에 관심을 갖고 질의를 생성하게 되므로, 결과에 대한 사용자의 만족도를 높이기 위해서는 사용자가 원하는 검색 대상의 속성만을 추출해서 제공해야 한다. 사용자가 지정한 토픽 타입에 대한 검색 결과만을 반환하기 위해 질의 변환기는 다음과 같은 변환 과정을 수행한다. SELECT 구문 역시 사용자가 지정한 테이블 명과 실제로 구현된 테이블 명 간에는 차이가 있으므로, 이를 대체하는 과정(Lines 1 ~ 3)이 반드시 필요하다. 만약 SELECT 문이 '*' (SELECT ALL) 문자를 포함하고 있지 않다면, 테이블 명만 대체한 이후 곧바로 수행 가능하

<알고리즘 3.5> Role Table의 참조를 위한 프로시져

```
Procedure Role(Role_Table, Role_Name);
BEGIN
1.    Ret_Attr ← NULL;
2.    IF the value of i^th row of Role Field in the Role_Table is equivalent to Role_Name THEN
3.        Add the value of i^th row of Attribute Field in the Role_Table to Ret_Attr;
4.    END IF
5.    RETURN Ret_Attr;
END
```

다. 하지만 SELECT 문이 '*' 문자를 포함하고 있다면 테이블에 포함된 전체 속성 중 사용자가 지정한 토픽 타입에 해당된 속성만을 추출하는 작업(Lines 7 ~ 8)이 수행된 후 SQL 문이 실행되어야 한다.

Ⅳ. 사례 분석

4.1. 대상 업무 소개

본 장에서는 대학의 가상 업무를 단순화하여 토픽맵으로 표현하고 이를 RDB로 저장한 후, 이에 대해 질의를 요청하고 변환하는 과정을 소개한다. 이 대학의 업무를 구성하고 있는 주요 주체는 "학생", "교수", "학과", "과목" 등이며, 각 주체가 다양한 역할로 서로 관계를 맺고 있다. 주체가 참여하는 역할에 따라서 관리해야 할 정보의 종류가 달라지므로, 이를 의미적으로 구조화하기 위한 방법이 필요하다. 이를 위해

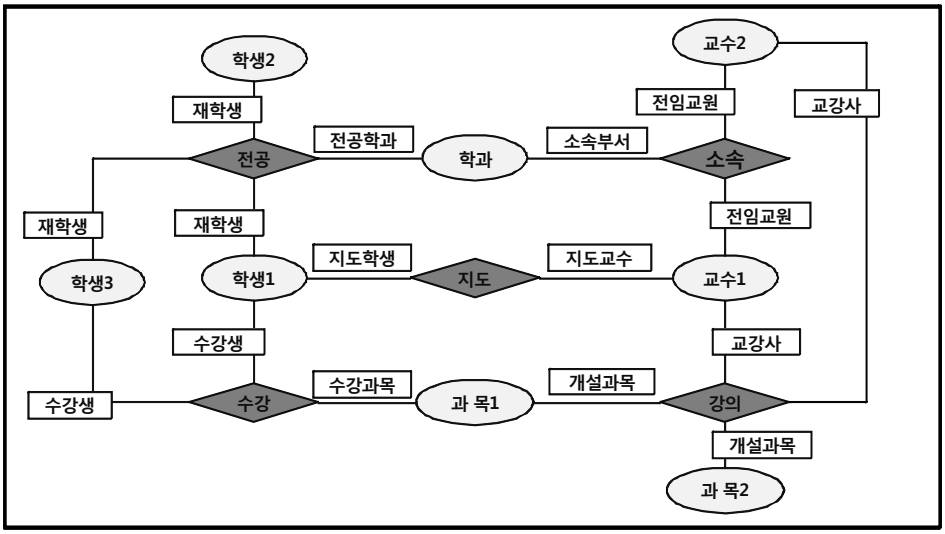
<알고리즘 3.6> SELECT 구문의 변환을 위한 프로시져

```
Procedure Tran_Select(SQL_i);
BEGIN
1.    Table_Name ← Table name in the SQL_i;
2.    New_Name ← Mapping(Table_Name);
3.    Replace Table_Name with New_Name in the SQL_i;
4.    IF SQL_i does not contain '*' phrase THEN
5.        Perform SQL_i;
6.    ELSE
7.        New_Attr ← Role(New_Name_Role, Table_Name);
8.        Replace predicate '*' with New_Attr in the SQL_i;
9.        Perform SQL_i;
10.   END IF
END
```

우선 요구사항 분석을 통해 토픽 및 관계를 추출하여 토픽맵을 구축해야 한다. “학생”은 “학과”와의 관계에서는 “재학생”이라는 역할로, “교수”와의 관계에서는 “지도학생”이라는 역할로, 그리고 “수업”과의 관계에서는 “수강생”이라는 역할로 참여할 수 있다. 또한 “교수”는 “학과”와의 관계에서는 “전임교원”이라는 역할로, “학생”과의 관계에서는 “지도교수”의 역할로, “과목”과의 관계에서는 “교강사”의 역할로 참여할 수 있다. 이러한 요구사항에 기반하여 도출된 토픽맵이 <그림 7>에 나타나 있다. <그림 7>은 8개의 토픽을 타원으로, 5개의 관계를 마름모로 표시하고 있다. 10개의 토픽 타입은 직사각형으로 표시되어 있으며, 이들은 “재학생”, “지도학생”, “수강생”, “전임교원”, “지도교수”, “교강사”, “전공학과”, “소속부서”, “개설과목”, “수강과목” 이다.

제안하는 구조는 각 토픽 타입이 실제로 어떤 테이블로 구현되었는지를 명시하는 매핑 테이블, 실제 데이터 값을 저장하고 있는 컨텐츠 테이블, 그리고 컨텐츠 테이블의 속성별 참여 역할을 정의한 롤 테이블로 구성된다. <그림 8>은 <그림 7>에 소개된 단순화된 대학의 가상 업무에 대한 컨텐츠 테이블의 스냅샷을 나타낸다. 총 4개의 테이블로 구성되며, 이들 각각은 <그림 7>의 토픽맵에 출현한 4 가지 주체인 “학생”, “교수”, “학과”, “과목”에 대한 정보를 담고 있다. 그림에서 각 테이블에는 여러 토픽 타입에 대한 정보가 혼재되어 있음을 알 수 있다. 예를 들어 “교수” 테이블에는 “전임교원”, “지도교수”, “교강사” 등의 역할에 대한 정보가 한꺼번에 나타나있다. 이에 대한 매핑 테이블은 <그림 9>에 나타나있다.

4.2. 다중역할 참여 토픽의 저장



<그림 7> 단순화된 대학의 가상 업무에 대한 토픽맵

학생

학번	이름	학년	출신전공	면담회수	총이수학점
S001	학생1	3	문과	5	96
S002	학생2	4	이과	Null	Null
S003	학생3	1	이과	Null	21

교수

교번	이름	나이	직급	주당면담시간	주당수업일수
P001	교수1	42	부교수	10	2
P002	교수2	39	조교수	Null	2

학과

학과번호	학과명	입학정원	전임교원수
D001	MIS	108	13

과목

과목번호	과목명	학점	수강인원	조교지원여부
C001	DataStructure	3	40	NO
C002	DataBase	3	NULL	YES

<그림 8> 단순화된 대학의 가상 업무에 대한 컨텐츠 테이블

Mapping

Topic Type	Implemented Table
지도학생	학생
재학생	학생
수강생	학생
전임교수	교수
지도교수	교수
강사	교수
전공학과	학과
소속부서	학과
수강과목	과목
개설과목	과목

<그림 9> 단순화된 대학의 가상 업무에 대한 매핑 테이블

학생_Role

Attribute	Role
학번	재학생
이름	재학생
학년	재학생
출신전공	재학생
학번	지도학생
이름	지도학생
학년	지도학생
면담회수	지도학생
학번	수강생
이름	수강생
학년	수강생
총이수학점	수강생

교수_Role

Attribute	Role
교번	전임교원
이름	전임교원
나이	전임교원
직급	전임교원
교번	지도교수
이름	지도교수
나이	지도교수
주당면담시간	지도교수
교번	교강사
이름	교강사
나이	교강사
주당수업일수	교강사

학과_Role

Attribute	Role
학과번호	전공학과
학과명	전공학과
입학정원	전공학과
학과번호	소속부서
학과명	소속부서
전임교원수	소속부서

과목_Role

Attribute	Role
과목번호	수강과목
과목명	수강과목
학점	수강과목
수강인원	수강과목
과목번호	개설과목
과목명	개설과목
학점	개설과목
조교지원여부	개설과목

<그림 10> 단순화된 대학의 가상 업무에 대한 롤 테이블

<그림 9>의 매핑 테이블의 역할을 간단히 살펴보면 다음과 같다. 예를 들어 사용자가 "학생 1"이라는 "지도학생"에 대한 모든 정보를 검색하는 경우를 가정하자. 이 경우 "지도학생" 토픽 타입에 대한 정보는 "학생" 테이블에서 관리됨을 매핑 테이블을 통해 알 수 있다. 따라서 "지도학생"에 대한 모든 질의는 "학생" 테이블에 대한 질의로 변환되어야 한다. 하지만 "학생" 테이블에는 "지도학생" 토픽 타입과는 무관한 여타 정보들이 포함되어 있으므로 이들을 걸러내기 위한 별도의 과정이 필요하며, 이 과정에서 롤 테이블(그림 10)이 활용된다. 즉 "학생" 테이블의 여러 속성 중 "지도학생" 토픽 타입에 관련된 속성은 "학번", "이름", "학년", "면담회수" 라는 사실이 롤 테이블을 통해 파악된다.

4.3. 질의 변환기의 동작 과정

4.3.1. CREATE 구문의 변환

다음 질의 A1은 사용자가 "지도학생" 역할로 참여하는 토픽에 대한 테이블을 생성하기 위한 질의이다. 단, 현재 상태는 어떠한 컨텐츠 테이블과 롤 테이블도 존재하지 않고 매핑 테이블만 존재하는 초기 상태인 것으로 가정하였으며, 기본키는 간략하게 밑줄로 표시하였다. 질의 A1에 대해 질의 변환기는 <그림 9>의 매핑 테이블을 참조하여 변환을 수행하며, 그 결과로 질의 A2 ~ A7을 생성하게 된다. <그림 11>은 질의 A2 ~ A6의 수행 결과로 "학생" 컨텐츠 테이블이 생성되고 "학생_Role" 롤 테이블에 새로운 값이 추가된 결과를 보여준다.

A1: CREATE TABLE 지도학생 (학번, 이름, 학년, 면담회수);

A2: CREATE TABLE 학생 (학번, 이름, 학년, 면담회수);

A3: CREATE TABLE 학생_Role (Attribute, Role);

A4: INSERT INTO 학생_Role VALUES ('학번', '지도학생');

A5: INSERT INTO 학생_Role VALUES ('이름', '지도학생');

A6: INSERT INTO 학생_Role VALUES ('학년', '지도학생');

A7: INSERT INTO 학생_Role VALUES ('면담회수', '지도학생');

<그림 11>의 상황에서 다음의 사용자 질의 B1이 요청된 경우를 살펴보자. B1의 경우 생성하고자 하는 테이블 "학생"이 이미 존재하고 있기 때문에, 새로운 테이블을 생성하는 것이 아

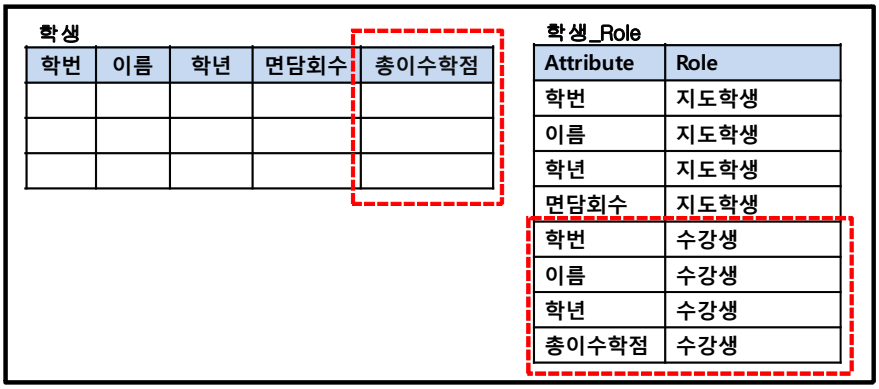
학생

학번	이름	학년	면담회수

학생_Role

Attribute	Role
학번	지도학생
이름	지도학생
학년	지도학생
면담회수	지도학생

<그림 11> "지도학생" 토픽 타입을 위한 테이블 생성 결과



<그림 12> “수강생” 토픽 타입을 위한 테이블 생성 결과

니라 기존의 테이블을 변경하기 위한 질의를 수행해야 한다. 이를 위해 질의 B1은 다음의 질의 B2 ~ B6으로 변환되며, <그림 12>는 질의 B2 ~ B6의 수행 결과로 “학생” 컨텐츠 테이블이 변경되고 “학생_Role” 롤 테이블에 새로운 값이 추가된 결과를 보여준다.

B1: CREATE TABLE 수강생 (학번, 이름, 학년, 총이수학점);

B2: ALTER TABLE 학생 ADD 이수학점

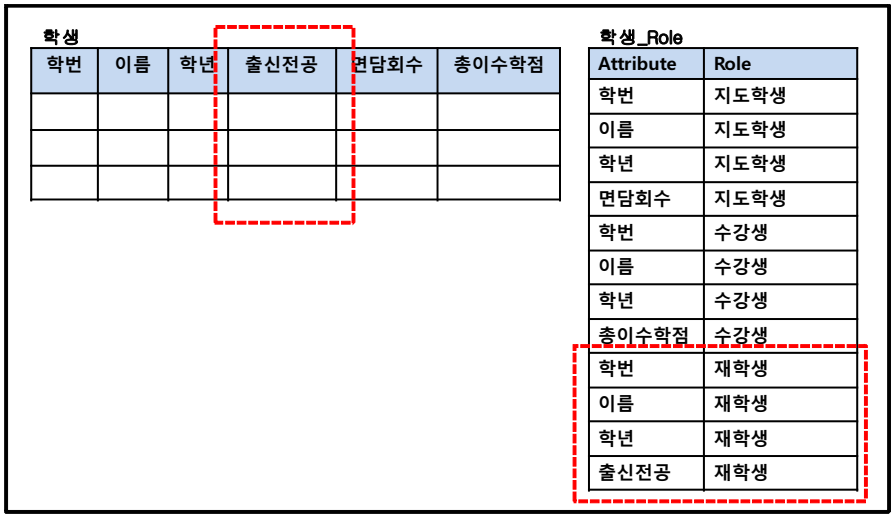
B3: INSERT INTO 학생_Role VALUES (‘학번’, ‘수강생’);

B4: INSERT INTO 학생_Role VALUES (‘이름’, ‘수강생’);

B5: INSERT INTO 학생_Role VALUES (‘학년’, ‘수강생’);

B6: INSERT INTO 학생_Role VALUES (‘총이수학점’, ‘수강생’);

마지막으로 사용자가 “재학생” 토픽 타입에 대응되는 테이블을 생성하고자 할 때의 질의인



<그림 13> “재학생” 토픽 타입을 위한 테이블 생성 결과

C1을 살펴보면 다음과 같다. "재학생" 토픽 타입은 "학생" 테이블로 구현되어야 하는데 이미 "학생" 테이블이 존재하고 있으므로, C1의 질의는 B1과 유사한 방식으로 변환된다. 즉 C1의 질의는 C2 ~ C6으로 변환되며, 변환된 질의를 수행한 결과가 <그림 13>에 나타나있다.

C1: CREATE TABLE 재학생 (학번, 이름, 학년, 출신전공);

C2: ALTER TABLE 학생 ADD 출신전공

C3: INSERT INTO 학생_Role VALUES ('학번', '재학생');

C4: INSERT INTO 학생_Role VALUES ('이름', '재학생');

C5: INSERT INTO 학생_Role VALUES ('학년', '재학생');

C6: INSERT INTO 학생_Role VALUES ('출신전공', '재학생');

4.3.2. INSERT 구문의 변환.

다음의 질의 D1은 사용자가 토픽 타입 "지도학생"에 참여하는 토픽 "학생1"의 인스턴스를 추가하기 위한 질의이다. D1 질의에 대해 질의 변환기는 매핑 테이블과 "학생_Role" 테이블을 참조하여 질의 D2를 작성하며, 변환된 질의의 수행 결과가 <그림 14>에 나타나 있다.

D1: INSERT INTO 지도학생 VALUES ('S001', '학생1', 3, 5);

D2: INSERT INTO 학생 (학번, 이름, 학년, 면담회수) VALUES ('S001', '학생1', 3, 5)

<그림 14>의 상황에서 토픽 타입 "재학생"에 참여하는 토픽 "학생2"의 인스턴스를 추가하기 위한 질의 E1을 변환하는 경우를 살펴보자. E1은 D1과 동일한 과정을 통해 E2로 변환되며, 변환된 질의의 수행 결과가 <그림 15>에 나타나있다.

E1: INSERT INTO 재학생 VALUES ('S002', '학생2', 4, '이과');

E2: INSERT INTO 학생 (학번, 이름, 학년, 출신전공) VALUES ('S002', '학생2', 4, '이과');

INSERT 구문이 항상 INSERT 구문으로 변환되는 것은 아니며, 그 예는 다음 질의 F1에서

학생

학번	이름	학년	출신전공	면담회수	총이수학점
S001	학생1	3	Null	5	Null

<그림 14> "지도학생" 토픽 타입의 "학생1" 인스턴스 추가

학생

학번	이름	학년	출신전공	면담회수	총이수학점
S001	학생1	3	Null	5	Null
S002	학생2	4	이과	Null	Null

<그림 15> "재학생" 토픽 타입의 "학생2" 인스턴스 추가

학생

학번	이름	학년	출신전공	면담회수	총이수학점
S001	학생1	3	Null	5	78
S002	학생2	4	이과	Null	Null

<그림 16> "수강생" 토픽 타입의 "학생1" 인스턴스 추가

살펴볼 수 있다. F1은 토픽 타입 "수강생"에 참여하는 토픽 "학생1" 인스턴스를 추가하기 위한 질의이다. "학생1"은 "수강생" 토픽 타입으로서는 처음 삽입되지만, 동일한 토픽 "학생1"이 이미 "지도학생" 토픽 타입으로 "학생" 테이블에 존재하고 있기 때문에, 해당 인스턴스가 새로 추가되는 대신 기존 인스턴스의 값을 변경하는 형태로 질의가 변경되어야 한다. 따라서 질의 F1은 UPDATE 구문을 사용한 F2로 변경되며, 그 결과 "학생1"의 "총이수학점" 은 NULL에서 78로 갱신된다(그림 16).

F1: INSERT INTO 수강생 ('S001', '학생1', 3, 78);
F2: UPDATE 학생 SET 총이수학점=78 WHERE 학번='S001

4.3.3. SELECT 구문의 변환

질의 변환기가 SELECT 구문을 처리하는 과정에서의 핵심은 '*' (SELECT ALL) 문자를 특정 속성 리스트로 한정하는 것이다. 즉 토픽 타입 "지도교수"의 속성만을 조회하고자 하는 질의 G1을 G2로 변경하는 것이 질의 변환기의 역할이며, 이 과정에서 "Mapping", "교수_Role" 테이블의 참조가 이루어진다. <그림 17>은 질의 수행 결과를 나타낸다.

*G1: SELECT * FROM 지도교수;*
G2: SELECT 교번, 이름, 나이, 주당면담시간 FROM 교수;

교수

교번	이름	나이	주당면담시간
P001	교수1	42	10
P002	교수2	39	Null

<그림 17> "지도교수" 역할에 해당하는 모든 속성의 조회

본 장에서는 토픽맵의 다중역할 참여 토픽을 RDB로 효과적으로 저장하기 위한 구조 및 이러한 저장구조 상에서 동작하는 질의 변환기의 동작 원리를 간단한 시나리오를 통해 소개하였다. 제안된 모델은 기존 RDB의 구조 변경을 최소화하면서도, 토픽맵에 표현된 의미 정보를 손실 없이 관리할 수 있을 것으로 사료된다.

V. 결 론

정보의 양이 방대해짐에 따라 다양한 의미 정보를 구조화하여 저장하기 위한 많은 방안이 모색되었으며, 그 중 토픽맵은 주제 중심의 모델로서 지식 구조의 표현에 있어 많은 장점을 갖는 것으로 인식되고 있다. 특히 토픽맵은 하나의 토픽이 여러 토픽 타입으로서의 역할을 수행하는, 즉 다중역할(Multi Role)을 수행하는

상황의 모델링에 있어서 매우 유용하다. 최근 토픽맵에 표현된 정보를 기존의 RDB 시스템에서 표현하기 위한 많은 시도가 있었음에도 불구하고, 이 과정에서 다중역할 토픽을 심도 깊게 다룬 연구는 찾아보기 어려운 실정이다. 본 논문에서는 토픽맵의 다중역할 토픽을 의미 손실 없이 RDB로 저장하기 위한 구조를 제안하고, 토픽맵과 RDB 간의 독립성을 보장하기 위한 질의 변환기를 설계하여 제시하였다. 제안된 모델은 기존 RDB의 구조 변경을 최소화하면서도, 토픽맵에 표현된 의미 정보를 손실 없이 관리할 수 있을 것으로 기대된다.

하지만 본 연구의 다음과 같은 측면은 후속 연구에서 면밀하게 분석될 필요가 있다. 질의 변환 과정에서 많은 경우 추가 질의가 생성되므로, 질의 변환 및 처리 시 응답 지연과 같은 성능상의 저하가 초래될 수 있다는 부작용이 있다. 물론 본 연구는 시간 측면의 성능 개선에 초점을 둔 연구가 아닌 논리적 모델에서 의미 정보의 표현력을 증대시키기 위한 연구이지만, 지나친 응답 지연은 제안하는 구조의 실제 적용에 분명한 저해 요소가 될 수 있다. 따라서 제안하는 모델의 활용 가능성을 평가하기 위해서는 다양한 환경 하에서의 성능 테스트가 반드시 수행되어야 할 것으로 사료된다. 또한 본 논문에서는 질의 변환기의 동작 과정을 살피기 위해 완전히 구현된 시스템이 아닌 사용자 인터페이스 중심의 간단한 프로토타입을 사용하였는데, 향후 연구에서는 이를 실제로 구현하여 보다 심도 깊은 테스트를 거치는 과정 역시 반드시 다루어져야 할 것이다.

〈참고문헌〉

강혜란, 이경호, "도메인 온톨로지에 기반한 XML 스키마의 통합," 멀티미디어학회 논문지, 제11권 제7호, pp. 940-955, 2008.

고창택, "정보체계 탐구 평가의 철학적 분석 모델과 그 방법론적 활용: 비판 실재론적 접근," 정보시스템연구, 제16권 제4호, pp. 131-155, 2007.

구미숙, 황정희, 류근호, 홍장의, "데이터 마이닝 기법을 이용한 XML 문서의 온톨로지 반자동 생성," 정보처리학회논문지 D, 제13권-D권 제3호, pp. 299-308, 2006.

김정민, 박철만, 정준원, 이한준, 민경섭, 김형주, "K-Box : 토픽맵 기반의 온톨로지 관리 시스템," 정보과학회논문지: 컴퓨팅의 실제, 제10권 제1호, pp. 1-13, 2004.

김정민, 신효필, 김형주, "T-MERGE 연산자에 기반한 분산 토픽맵의 자동 통합," 정보과학회논문지: 소프트웨어 및 응용, 제33권 제9호, pp. 787-801, 2006.

김하균, "XBC(XML Based Catalog)의 설계에 관한 연구'" 정보시스템연구, 제13권 제1호, pp. 161-177, 2004.

박여삼, 장옥배, 한성국, "X-TOP: 레거시 시스템 상에서 온톨로지 구축을 위한 토픽맵 플랫폼의 설계와 구현," 정보과학회논문지: 컴퓨터의 실제 및 레터, 제14권 제2호, pp. 130-142, 2008.

이한준, 민경섭, 김형주, "RDBMS 기반의 토픽맵 무결성 검사 기법," 정보과학회논문지: 데이타베이스, 제34권 제6호, pp. 493-502, 2007.

이혜원, "정보를 표현하는 기법으로서의 RDF와 토픽맵과의 비교," 제12회 한국정보관리학회 학술대회 논문집, pp. 99-106, 2005.

정대율, "모델베이스 설계를 위한 개념적 모델링 도구에 관한 연구," 정보시스템연구, 제7권 제1호, pp. 181-208, 1998.

정호영, 김정민, 정준원, 김형주, "XTM 기반의 토픽맵," 데이터베이스연구, 제19권 제1호, pp. 38-47, 2003.

정현숙, "ERP 시스템의 데이터 정확성 유지를 위한 물품 온톨로지의 설계 및 구현," 한국산업정보학회논문지, 제10권 제1호, pp. 38-48, 2005.

한국전자거래진흥원, "Topic Maps 응용 표준 및 활용 가이드라인 개발," 한국전자거래진흥원, 2003.

Cerbah, F., "Learning Highly Structured Semantic Repositories from Relational Databases: The RDBToOnto Tool," Lecture Notes in Computer Science, Vol. 5021, pp. 777-781, 2008.

Cullot, N., Ghawi, R., and Yetongnon, K., "DB2OWL: A Tool for Automatic Database-to-Ontology Mapping," in Proceedings of the 15th Italian Symposium on Advanced Database Systems, Torre Canne di Fasano (BR), Italy, 491-494, 2007.

Dhanapalan, L. and Chen, J. Y., "A Case Study of Integrating Protein Interaction Data using Semantic Web Technology," International Journal of Bioinformatics Research and Applications, Vol. 3, pp. 286-302, 2007.

Gruber, T. R., "Toward Principles for the Design of Ontologies Used for Knowledge Sharing," International Journal of Human Computer Studies, Vol. 43, pp. 907-928, 1995.

Holger, R. H., "Topic Maps : Template, Topology, and Hierarchies," Markup Language Theory and Practice, Vol. 2, No. 1, pp. 45-64, 2000.

Lassila, O. and Swick, R. R., "Resource Description Framework(RDF) Model and Syntax Specification," W3C Recommendation, 1999.

Mizoguchi, R., "Tutorial on Ontological Engineering: Part2: Ontology Development, Tools, and Languages," New Generation Computing, Vol. 22, No.1, pp. 61-96, 2004.

Moore, G., "Topic Map Technology - The State of the Art," XML 2000 Conference & Exposition, Washington, USA, 2000.

Pepper, S. and Moore, G., "XML Topic Maps(XTM) 1.0," TopicMaps.org, 2001.

Pepper, S., Vitali, F., Garshol, L. M., Gessa, N., and Presutti, V., "A Survey of RDF/Topic Maps Interoperability Proposals," W3C Working Group Note, 2006.

Rahm, E. and Bernstein, P. A., "A Survey of Approaches to Automatic Schema Matching," The VLDB Journal, pp. 334-350, 2001.

Sahoo, S. S., Halb, W., Hellmann, S., Idehen, K., Thibodeau, T. Jr., Auer, S., Sequeda, J., and Ezzat, A., "A Survey of Current Approaches for Mapping of Relational Databases to RDF," W3C RDB2RDF Incubator Group, 2009.

정윤수(Yoonsoo Jung)

수원과학대학에서 전자계산학을 전공하였으며, 국민대학교 비즈니스IT전문대학원에서 2009년 정보시스템전공 석사학위를 취득하였다. 주요 관심분야는 지식 모델링, 데이터베이스 설계 등이다.

이춘열(Choon Y. Lee)

1979년 서울대학교 산업공학과를 졸업하고, 1983년 서울대학교 대학원 경영학과에서 경영학석사 학위를 취득하였으며, 1990년 미시간대학교에서 경영정보학박사 (Computer and Information Systems) 학위를 취득하였다. 또한, 1779년부터 1984년까지 국방정보체계연구원에서 연구원으로 근무하였으며, 1991년부터 1993년까지 한국통신 소프트웨어연구소에서 근무하였다. 국민대학교 비즈니스 IT 전문대학원 원장을 역임하였으며, 현재 국민대학교 비즈니스IT학부 교수로 재직하고 있다. 한국경영정보학회, 학국경영학회, 대한산업공학회, 데이타베이스학회 등의 정회원이며, 주요 관심분야는 데이터베이스, 데이터웨어하우징, 정보 자원 계획 및 관리 등이다.

김남규(Kim, Namgyu)

현재 국민대학교 비즈니스IT학부에서 조교수로 재직 중이다. 서울대학교 컴퓨터공학과에서 학사 학위를 취득하고, KAIST 테크노경영대학원에서 Database와 MIS를 전공하여 경영공학 석사 및 박사학위를 취득하였다. 한국정보시스템학회, 한국지능정보시스템학회, 한국경영정보학회 등의 종신회원으로 활동 중이며, 주요 관심분야는 데이터베이스 설계 및 데이터 마이닝 등이다.

<Abstract>

Relational Database Structure for Preserving Multi-role Topics in Topic Map

Yoonsoo Jung · Choon Y. Lee · Namgyu Kim

Traditional keyword-based searching methods suffer from low accuracy and high complexity due to the rapid growth in the amount of information. Accordingly, many researchers attempt to implement a so-called semantic search which is based on the semantics of the user's query. Semantic information can be described using a semantic modeling language, such as Topic Map. In this paper, we propose a new method to map a topic map to a traditional Relational Database (RDB) without any information loss. Although there have been a few attempts to map topic maps to RDB, they have paid scant attention to handling multi-role topics. In this paper, we propose a new storage structure to map multi-role topics to traditional RDB. The proposed structure consists of a mapping table, role tables, and content tables. Additionally, we devise a query translator to convert a user's query to one appropriate to the proposed structure.

***Keywords*:** Topic Map, Semantic Model, Relational Database, Query Translator

* 이 논문은 2009년 6월 24일 접수하여 2차 수정을 거쳐 2009년 9월 3일 게재 확정되었습니다.