관계형 데이타베이스의 객체지향적 인터페이스를 위한 게이트웨이의 설계 및 구현

(Design and Implementation of a Gateway for Object-Oriented Interface on Relational Database)

박 상 원 † 김 형 주 ††
(Sang-Won Park) (Hyoung-Joo Kim)

요 약 본 논문은 객체지향적 인터페이스를 통한 관계형 데이타베이스 접근을 지원하는 스키마 게이트웨이의 설계와 구현내용을 다룬다. 스키마 게이트웨이는 관계형 데이타베이스 스키마의 객체지향 스키마로의 변환, 양 스키마사이의 사상정보 저장, 객체지향 질의어의 SQL로의 변환 기능을 제공한다. 특히 스키마의 변환은 관계형 데이타베이스에서 계승, 참조, 관계, 집합 애트리뷰트 등으로 중첩되어 있는 외래키(foreign key)의 의미(semantics)를 분석하여 이루어진다. 이를 통해 사용자 인터페이스 면에서 관계형 데이타베이스가 가지는 문제점을 제거함과 동시에, 멀티 데이타베이스 환경에서의 사용자에게 단일한 데이타 모델을 제공하는 장점을 얻었다.

Abstract This paper describes the design and implementation of schema gateway which allows accesses to RDBMS through the object-oriented interface The proposed schema gateway supports translation of RDB schemas to OODB schemas, mapping informations of both schemas, and translation of queries written in OQL to SQL queries. Schema translation is based on the analysis of foreign key semantics which can be classified into inheritance, reference, relationship, and set attributes in relational databases. The major advantage of the proposed framework is that it may be served as the uniform data model in the multi or heterogeneous database environment.

1. 서 론

1970년도 Codd가 제안한 관계형 데이타 모델[3]은 80년대 이후 가장 널리 쓰이는 데이타베이스 모델로서, 수학적으로 잘 정의되어 있고 모델이 간단하다는 장점이 있다. 이 관계형 모델을 바탕으로 만들어진 관계형 데이타베이스는 기업이나 은행에서 발생하는 레코드 형태의 데이타 처리에는 적합한 시스템이지만 CAD/CASE, 멀티미디어와 같은 실생활에서 볼 수 있는 여러 가지 형태를 모델링하기에는 그 모델링 능력이 떨어진다. 그래서 모델링 능력을 향상하기 위한 여러 노력들이 있었다[16]. 이런 시맨틱(semantic) 데이타 모델이 발전하여 객체지향 데이타베이스가 등장하게 되었다. 90년대에는 두 데이타베이스 시스템이 공존할 것이다. 본 논문에서 제안하는 스키마 게이트웨이는 관계형 데이타베이스에 대한 객체지향적 인터페이스를 제공함으로써 멀티 데이타베이스 접근을 가능하게 하였고 관계형 데이타베이스가 가지는 문제점을 해결하였다.

관계형 데이타베이스는 사용자의 관점에서 여러 문제점이 발생한다. 첫째, 관계형 데이타베이스 스키마의 이해가 어렵다. 관계형 데이타베이스 스키마 설계 방법으로 가장 널리 사용되는 ER 모델[2]은 현실세계를 개체(entity)와 관계(relationship)로 표현한다. 이와 같은 ER 모델을 관계형 데이타베이스의 스키마로 매핑[10]시키면 ER 모델에서 개념적으로 사용한 많은 의미들이 사라지게 된다. 테이블간의 관계를 이용하여 필요한 정보를 얻는 방법인 조인은 대부분 기본키(primary key)와 외래키(foreign key)간에 일어난다[19]. 즉, 관계의 다양한

· 이 논문은 통상산업부 공업기반기술과제인 No 943204 "객체지향 데이타베이스 설계도구 개발에 관한 연구" 프로젝트의 일환으로 진행되었던 과제이다
† 학생회원 : 서울대학교 컴퓨터공학과
†† 종신회원 : 서울대학교 컴퓨터공학과 교수
논문접수 1996년 3월 11일
심사완료 · 1997년 4월 17일

의미가 외래키라는 한가지 방법으로 표현되기 때문에 스키마 정보만으로 스키마의 의미를 파악하는 것이 힘들다. 둘째, 사용자가 복잡한 의미의 질의어를 쉽게 생성하는 것이 쉽지 않다. 대부분의 SQL 사용자는 한 개의 릴레이션에 서너 개의 애트리뷰트를 이용한 질의어를 사용한다[13]. 즉, 조인을 이용하는 것이 힘든데 이는 외래키의 의미를 파악하여 의미 있는 SQL 문장을 만들기 힘든 때문이다. 하지만 많은 부분에서 SQL의 조인은 OQL(object query language)에서 경로식(path expression)으로 바뀔 수 있기 때문에[22] OQL을 사용하면 보다 쉽게 원하는 질의어를 생성할 수 있다.

새로운 응용을 위해 객체지향 데이타베이스가 제안되었지만 기존의 모든 데이타를 객체지향 데이타베이스로 옮겨서 사용하기에는 위험이 따른다[8]. 또한 기존의 데이타를 모두 버리고 새로 구축할 수도 없다. 이와 같은 문제점을 해결하기 위해 여러 가지 방법이 제시되었는데[12] 그 대표적인 것들을 살펴보면 다음과 같다. 첫째, 두 가지의 시스템, 즉 관계형 데이타베이스와 객체지향 데이타베이스를 모두 사용하는 것이다. 이것은 새로운 응용에서 발생하는 데이타는 객체지향 데이타베이스에 저장하고, 기존의 응용에서 발생하는 데이타는 관계형 데이타베이스에 저장하는 것이다. 이때 발생할 수 있는 문제점은 사용자가 두 시스템을 모두 알아야 한다는 점이다. 둘째, 관계형 데이타베이스에 객체지향 인터페이스를 제공하는 것이다. 이렇게 객체지향 인터페이스를 제공하면 관계형 데이타베이스가 가지는 모델링 능력의 부족을 해소할 수 있고 사용자는 새로운 관점으로 데이타베이스를 바라볼 수 있다. 이런 방법을 제공하는 것에는 Penguin[6, 17, 18], Persistence[7]와 같이 외부 인터페이스 부분인 스키마와 질의어 등은 객체지향 데이타베이스이지만 내부적인 저장 시스템은 관계형 데이타베이스를 이용하는 것과 [5, 21]과 같이 OQL(object query language)을 사용하여 관계형 데이타베이스를 접근하는 방법이 있다. 본 논문에서는 관계형 데이타베이스의 외래키의 의미를 분석하여 관계형 데이타베이스의 스키마를 객체지향 데이타베이스의 표준인 ODMG[1] 데이타 모델[1]로 변환하므로서 ER 모델에서 테이블로의 매핑 시 잃어버린 정보를 복원하여 사용자에게 스키마의 의미를 쉽게 알 수 있도록 하였다. 또한 ODMG OQL을 이용한 질의를 SQL 문장으로 변환하여 결과를 객체의 형태로 사용자에게 보여줌으로서 관계형 데이타베이스에 대한 복잡한 의미의 질의어를 쉽게 생성할 수 있도록 하였다. 이것을 통하여 사용자에게 관계형 데이타베이스와 객체지향 데이타베이스에 대한 통일된 접근방법을 제공하였다.

본 논문의 구성은 다음과 같다. 2장에서는 시스템 전반에 대하여 설명하고 3장에서는 ODMG 데이타 모델에 대해 논문의 전개에 필요한 범위에서 설명한다. 관계형 스키마에서 객체지향 스키마로의 변환과 객체지향 질의어의 SQL로의 변환에 대해서 각각 4, 5장에서 다루고 마지막으로 시스템의 응용과 결론 및 향후 연구 방향에 대해 설명한다.

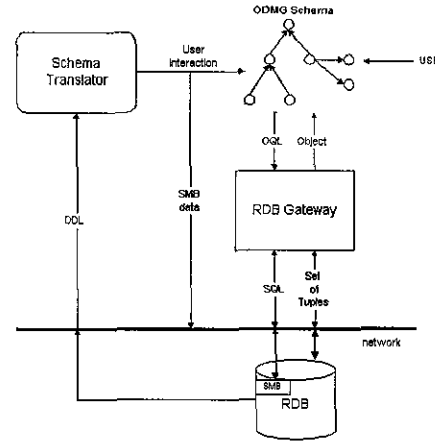

그림 1 시스템 구조

2. 시스템 구성

그림 1은 본 논문에서 제안하는 시스템의 전체구성을 보여주고 있는데, 크게 관계형 데이타베이스 스키마를 읽어 객체지향 데이타베이스 스키마로 변환하는 스키마 변환 도구(schema translator), 양 스키마 사이의 사상정보를 저장하는 스키마 매핑 베이스(schema mapping base), 그리고 OQL을 SQL로 변환하는 질의어 변환 라이브러리(RDB gateway)로 구성되어 있다.

2.1 스키마 변환 도구

그림 1에서 스키마 변환 도구는 관계형 스키마를 읽어들여 사용자의 도움을 받아 객체지향 스키마로의 변환을 담당하는 모듈이다. 스키마 변환에 관한 자세한 내용은 4장을 참조하기 바란다. 스키마 변환은 사용자 편의를 위해 Windows와 X/Motif 용 GUI 도구를 이용한다.

1) Object Database Management Group에서 제정한 객체지향 데이타베이스 표준안

그림 2는 관계형 데이타베이스 스키마를 읽어들인 스키마 변환 도구 화면이다.

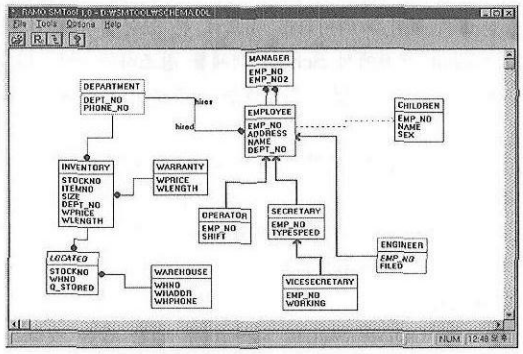

그림 2 스키마 변환 도구

2.2 스키마 매핑 베이스

스키마를 변환하면 변환된 스키마 정보를 저장하기 위해 스키마 매핑 베이스[2]가 구축되고 이것은 데이타베이스에 저장된다. 이 SMB를 통하여 사용자에게 관계형 데이타베이스의 스키마를 ODMG 스키마로 보여 준다. 여기에는 테이블과 클래스 사이의 매핑과 테이블의 애트리뷰트와 클래스 애트리뷰트간의 매핑, 외래키와 클래스 참조(reference)의 매핑을 저장한다. 5장에서 설명할 질의어 변환은 이 SMB를 이용하여 수행된다. 그리고 다른 객체지향 응용 프로그램에서의 스키마 정보에 대한 요청도 SMB를 이용한다.

예제 2.1 (SMB 생성) 아래와 같은 관계형 데이타베이스 스키마를 스키마 매핑 도구에서 읽어들이면 그림 2와 같다.

EMPLOYEE (EMP_NO, ADDRESS, NAME, DEPT_NO)
OPERATOR (EMP_NO, FIELD)
CHILDREN (EMP_NO, NAME, SEX)
DEPARTMENT (DEPT_NO, PHONE_NO)
INVENTORY (STOCKNO, ITEMNO, SIZE, DEPT_NO, WPRICE, WLENGTH)
LOCATED (STOCKNO, WHNO), Q_STORED)
WAREHOUSE (WHNO, WHADDR, WHPHONE)

그림 2에서 사각형은 테이블을 나타내고 사각형과 사각형을 연결하는 링크는 한쪽 테이블에서 다른 쪽 테이블을 가리키는 외래키가 존재함을 나타낸다. 링크 양단의 작은 원 중에서 검게 칠해진 쪽 테이블에 외래키가 존재한다. 예를 들어 EMPLOYEE의 EPT_NO는 DEPARTMENT를 가리키는 외래키이므로 두 테이블은 관계를 가지고 있다. 또한 계승 관계는 화살표로 표시한다. 예를 들어 EMPLOYEE와 ENGINEER 사이의 화살표는 IS-A 관계를 표시한다. 또한 집합 애트리뷰트는 점선 화살표로 나타낸다. 이를 변환하면 그림 3과 같은 SMB가 생성된다. SMB에는 각 클래스가 어느 클래스로부터 계승되었는지 각 애트리뷰트의 타입은 어떤 것이고 어떤 의미가 있는지가 나타나 있다. □

Employee	EMPLOYEE	
Address	ADDRESS	
Name	NAME	
hired	*Ref<Department>*	
Engineer	ENGINEER	*inherit from Employee*
Field	*FIELD*	
Inventory	INVENTORY	
Stockno	STOCKNO	
Itemno	ITEMNO	
Size	SIZE	
manages	*Ref<Department>* DEPT_NO	
Wprice	WPRICE	
Wlength	WLENGTH	
Department	DEPARTMENT	
Phone_no	PHONE_NO	
hires	*Set<Employee>*	
managed	*Set<Inventory>*	

그림 3 SMB 예제

2.3 질의어 변환 라이브러리

사용자가 변환된 스키마를 이용하여 원하는 바를 ODMG OQL로 질의하면 이 질의는 질의어 변환 라이브러리를 통해 SQL로 변환된 다음, 관계형 데이타베이스에서 결과를 가져와 사용자에게 객체로서 보여지는 것이다. 이에 대한 자세한 내용은 5장을 참고하기 바란다. 이것은 입력으로 받아들인 ODMG OQL을 내부적으로 SMB를 이용하여 SQL 문장으로 변환한 다음 관계형 데이타베이스에 질의한다. 그 결과 튜플의 집합을 결과로 얻는데 이것을 사용자에게 객체의 형태로 전달한다.

질의어 변환 라이브러리에 질의어를 이용한 것으로 6장에서 설명할 OOViewer[3]와 OQL을 그래픽을 이용하여 생성하고 질의하는 SOPView[4]가 있다. 이와 같이 여러 응용 프로그램에서 질의어 변환 라이브러리를 이용하여 ODMG OQL로 관계형 데이타베이스를 접근할 수 있다.

2) Schema Mapping Base, 이하 SMB라 한다.

3) 본 시스템을 이용하여 만든 대화형 질의어 처리기이다.

3. ODMG 데이타 모델

본 논문에서는 ODMG-93을 객체지향 데이타베이스 모델로 선택했다. 현재 산업체의 객체지향 데이타베이스 표준이 정해지지 않아 완전한 모델의 명세가 가능하고 많은 상용 객체지향 데이타베이스가 따르고 있는 ODMG-93을 선택하였다. 하지만 대부분의 객체지향 모델은 기본적으로 3.1, 3.2절의 명세를 모두 가지므로 타 객체지향 데이타 모델로의 적용 또한 가능하다. 여기서는 스키마 변환 및 질의어 변환과 관련하여 관심 있는 ODL(object definition language)과 OQL에 대해서만 다루기로 한다.

3.1 ODL

ODL은 객체지향 데이타베이스에서 스키마를 기술하는 언어로서 계승(inheritance), 집합 애트리뷰트와 같이 관계형 데이타 모델로 직접 표현할 수 없는 것과 List, Bag 등과 같이 관계형 데이타베이스에서는 표현 불가능한 것도 포함하고 있다. 이 절에서는 관계형 스키마를 객체 지향형 스키마로 변환하였을 때 나타나는 ODMG 모델만을 예제 3.1을 이용하여 설명한다. 각각에 대한 자세한 설명은 [1]을 참조하기 바란다.

예제 3.1 이름과 주소, 취미, 학교, 친구 등의 애트리뷰트를 가지는 학생을 ODL로 모델링 하면 다음과 같다.

```
interface Student : Person
( keys (soc_no))
{
    attribute String name;
    attribute String address;
    attribute Set<String> hobby;
    relationship School school
            inverse School::students;
    relationship Set<Person> friend;
} □
```

계승 예제 3.1의 Student 객체는 Person 객체로부터 계승받는다. 즉, Person 객체의 모든 인자를 자신의 인자로 가진다.

집합 애트리뷰트 예제 3.1의 hobby는 집합 애트리뷰트이다. 관계형 데이타베이스의 경우 데이타 모델의 특성상 레코드의 한 필드가 원자값(atomic value)을 가져야 하기 때문에 두개의 테이블로 스키마를 만들어야 하지만, ODMG 모델에서는 집합 애트리뷰트를 지원하므로 객체내의 한 애트리뷰트로 생성한다.

relationship 한 객체에서 다른 객체로 참조하는 경우이다. 예제 3.1에서 school과 같은 경우이다. 이것은 School 객체를 참조하는 애트리뷰트이다. 이때 inverse는 참조 무결성을 의미하는 것으로 본 논문에서는 단순히 Student 객체에서 School 객체를 참조하는 애트리뷰트 students가 존재함을 나타내는 것으로 사용한다.

relationship Set 한 객체가 다른 객체와 맺고 있는 관계가 1:n 혹은 n:m 일 경우이다. 예제 3.1에서 Student 객체는 Person 객체와 1:n 관계를 가지며 이 관계의 이름은 friend이다.

3.2 OQL

ODMG OQL은 SQL로 나타낼 수 없는 많은 것들을 표현할 수 있다. 예를 들어 SELECT, FROM 절에 다시 중첩된 질의어가 올 수 있으며, 사용자 정의 타입을 둘 수 있는데 이와 같은 모든 OQL 문장을 SQL 문장으로 변환할 수는 없다. 다음은 본 논문에서 변환 가능한 OQL 문장들이다.

SELECT FROM WHERE SELECT FROM WHERE 절을 변환한다. 또한 SELECT, FROM, WHERE 각 절에 다시 중첩된 SELECT FROM WHERE 절이 오는 것도 변환한다. 단 여기서는 중첩된 질의어를 SELECT FROM WHERE로 제한하였다. 원래 OQL 문장은 모든 OQL이 SELECT, FROM, WHERE 절에 중첩되어 올 수 있지만 이 모두를 SQL 문장으로 변환할 수 없으므로 SELECT FROM WHERE 절만 중첩되는 것으로 제한한다.

집단화 함수(aggregate function) SUM, AVG, COUNT, MAX, MIN과 같은 집단화 함수를 허용한다. 하지만 이것 또한 함수의 인자를 SELECT FROM WHERE 절로 제한하였다.

경로식(path expression) 한 객체에서 다른 객체를 경로식을 통해 참조할 수 있다. 예제 3.1에서 학교의 주소는 Person.school.Address로 나타낼 수 있다.

4. 스키마 변환

이 장에서는 스키마 변환에 관하여 설명한다. 먼저 관계형 데이타베이스에서 외래키가 가지는 의미를 각각 살펴본다. 이와 같은 외래키 의미의 분석을 통하여 관계형 데이타베이스 스키마를 객체지향 데이타베이스 스키마로 변환하는 변환규칙을 제시한다.

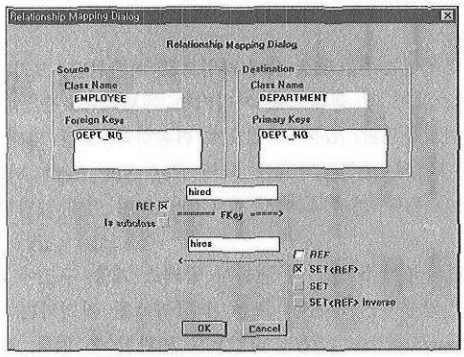

그림 4 링크에 의미 부여

4.1 변환 방법

관계형 스키마를 객체지향 스키마로 변환하는 방법은 [15, 20]등에 나와 있는데 여기서는 확장된 ER 모델을 자동적으로 변환하는 방법을 취하고 있다. 이 방법의 단점은 먼저 스키마가 [15, 20]에서 제안한 방법을 이용하여 잘 정의되어 있어야 한다는 것이다. 잘 정의된 스키마만 변환할 수 있다는 것은 많은 관계형 스키마를 변환할 수 없다는 것을 말한다. 두 번째 단점은 사용자가 스키마 변환에 관여하여 원하는 형태의 스키마로 만들 수 없다는 점이다. 그리고 객체지향 데이타베이스 표준인 ODMG를 따르지 않았기 때문에 상용 객체지향 데이타베이스에서의 적용이 고려되지 않았다.

이런 단점을 극복하기 위해 본 연구에서는 중첩되어 있는 외래키의 의미를 분석하여 스키마를 변환한다. 그림 2에서 보는 바와 같이 관계형 데이타베이스 스키마의 테이블은 사각형으로 표시하고, 테이블간에 관계가 있으면 즉, 외래키를 가지는 테이블과 참조되는 테이블은 링크로 연결한다. 각 링크의 의미는 그림 4를 이용하여 사용자가 정의하는데 이것의 의미는 3.1절의 각 타입으로 분해된다. 이때 의미가 없는 일을 사용자가 할 경우, 이를 방지하기 위해 스키마 매핑 도구에서 이를 감지하여 규칙에 어긋나는 변환은 거부한다. 예를 들어 그림 4에서 EMPLOYEE 테이블과 DEPARTMENT 테이블간에 상위-하위 계층구조를 가지면 다른 선택사항은 모두 무시된다. 또한 왼쪽에 있는 테이블에 외래키가 있으므로 Set<Ref> 관계가 왼쪽에는 없다. 또한 상위-하위 클래스 관계는 반드시 외래키가 있는 곳이 하위 클래스가 되므로 이를 반영하였다. 즉, 스키마를 변환할 때 모순되는 변환을 할 수 없도록 스키마 매핑 도구에서 모순되는 행위를 방지하여 항상 의미가 있는 변환만 가능하도록 하였다. 예를 들어 클래스 계층 구조에 사이클이 발생하면 사용자에게 경고를 주고 그 변환은 무시된다. 이때 외래키가 가지는 의미로는 계승, 집합 애트리뷰트, Ref, Set<Ref>가 있고 이에 대한 변환 방법은 4.2절에서 설명한다.

4.2 변환규칙

본 논문의 스키마 변환은 사용자가 자유롭게 원하는 형태로 변환할 수 있게 하는 방법을 택하였다. 그 이유는 관계형 스키마를 객체지향형 스키마로 완전 자동으로 변환할 수 없기 때문이다[15, 20]. 또한 사용자는 자신의 관점으로 스키마를 보기를 원하기 때문에 사용자가 원하는 형태의 스키마로 변환할 수 있도록 허용하였다. 하지만 일정규칙에 의거하여 스키마가 자동 변환한 다음 사용자가 변환을 하는 것이 사용자의 수고를 줄일 수 있을 것이다. 관계형 스키마는 아래의 변환규칙 1, 2, 3, 4에 의해 스키마 매핑 도구에서 자동적으로 객체지향형 스키마로 변환되어 스키마 변환의 정확성을 높인다. 이때 사용자의 변환이 변환규칙 5에 어긋나지 않아야 한다.

정의 1 (P, F, PF) $P(A)$는 A 테이블의 기본키 집합을 의미하는 것으로 예제 2.1에서

$$P(LOCATED) = \{STOCKNO, WHNO\}$$

이다. $F_A(B)$는 B 테이블에서 A 테이블을 가리키는 외래키의 집합을 의미하는 것으로 예제 2.1에서

$$F_{DEPARTMENT}(EMPLOYEE) = \{DEPT\_NO\}$$

이다. 즉 *EMPLOYEE* 테이블에서 *DEPARTMENT* 테이블을 가리키는 외래키 집합은 {DEPT_NO}이다. 또한 $PF(A)$는 A 테이블의 기본키들이 외래키의 역할도 할 때 그 외래키들이 가리키는 테이블들의 집합으로서 예제 2.1을 보면

$$PF(LOCATED)=\{INVENTORY, WAREHOUSE\}$$

이다. 즉 *LOCATED* 테이블의 기본키 중 외래키로는 *INVENTORY* 테이블을 가리키는 *STOCKNO*와 *WAREHOUSE* 테이블을 가리키는 *WHNO*가 있다. 그러므로 *LOCATED*의 외래키들이 가리키는 테이블로는 *INVENTORY*와 *WAREHOUSE*가 있다. □

예제 4.1 다음은 예제 2.1의 스키마에서 *P, F, PF* 값들을 구한 것이다.

$$
\begin{aligned}
P(EMPLOYEE) &= \{EMP\_NO\} && (1)\\
P(CHILDREN) &= \{EMP\_NO, NAME\} && (2)\\
P(OPERATOR) &= \{EMP\_NO\} && (3)\\
P(DEPARTMENT) &= \{DEPT\_NO\} && (4)\\
P(INVENTORY) &= \{STOCKNO\} && (5)\\
P(LOCATED) &= \{STOCKNO, WHINO\} && (6)\\
P(LOCATED) &= \{STOCKNO, WHINO\} && (7)\\
P(WAREHOUSE) &= \{WHINO\} && (8)\\
F_{EMPLOYEE}(CHILDREN) &= \{EMP\_NO\} && (9)\\
F_{EMPLOYEE}(OPERATOR) &= \{EMP\_NO\} && (10)\\
F_{DEPARTMENT}(EMPLOYEE) &= \{DEPT\_NO\} && (11)\\
F_{DEPARTMENT}(INVENTORY) &= \{DEPT\_NO\} && (12)\\
F_{INVENTORY}(LOCATED) &= \{STOCKNO\} && (13)\\
F_{WAREHOUSE}(LOCATED) &= \{WHINO\} && (14)\\
PF(CHILDREN) &= \{EMPLOYEE\} && (15)\\
PF(OPERATOR) &= \{EMPLOYEE\} && (16)\\
PF(EMPLOYEE) &= \{DEPARTMENT\} && (17)\\
PF(INVENTORY) &= \{DEPARTMENT, LOCATED\} && (18)
\end{aligned}
$$

□

변환규칙 1 각 테이블을 클래스로 매핑하고 클래스 리스트에 삽입한다. 테이블의 애트리뷰트는 클래스의 애트리뷰트로 매핑된다.

변환규칙 2 (집합 애트리뷰트) 1:n 관계에서 외래키로 참조되는 클래스가 집합 애트리뷰트일 경우, 즉

$$
\begin{aligned}
P(A) &\equiv F_A(B)\\
P(A) &\subset P(B)\\
PF(B) &\equiv \{A\}
\end{aligned}
$$

와 같을 때 B 클래스는 A 클래스의 집합 애트리뷰트가 되고 A 클래스에 *Set* 타입의 애트리뷰트가 생성된다. 그리고 B 클래스는 클래스 리스트에서 삭제된다. 예제 4.1에서 식 (1), (2), (8), (14)에 의해 *CHILDREN*이 집합 애트리뷰트로 변한다.

변환규칙 3 (계승) 1:n 관계에서 외래키로 참조되는 테이블이 상위 클래스가 되는 경우가 있다. 즉

$$
\begin{aligned}
P(A) &\equiv P_B(A)\\
P(A) &\equiv P(B)
\end{aligned}
$$

와 같이 A 테이블의 기본키 집합 $P(A)$가 B 테이블의 기본키 집합 $P(B)$와 같고, A 테이블의 기본키 집합 P(A)가 A 테이블의 외래키 집합 $F_B(A)$와 같을 때 A는 B의 하위 클래스(subclass)로 매핑된다. 이때 하위 클래스는 상위 클래스의 애트리뷰트들을 모두 계승한다. 예제 4.1에서 식 (1), (3), (9)에 의해 *OPERATOR*가 *EMPLOYEE*의 하위 클래스로 매핑된다.

변환규칙 4 (Ref, Set<Ref>) A 클래스의 기본키 집합 P(A)가 B 클래스의 A 클래스를 가리키는 외래키 집합 $F_A(B)$와 같을 때, 즉

$$P(A) \equiv F_A(B)$$

일 경우에는 $F_A(B)$를 Ref 타입으로 변환한다. 그리고 A 클래스에서 B 클래스를 가리키는 애트리뷰트를 생성하고 이것을 Set<Ref> 타입으로 변환한다. 모든 외래키는 기본적으로 변환규칙 4에 의해 *Ref, Set<Ref>* 타입으로 변환된다. 예제 4.1에서 식 (10)-(13)이 이에 해당한다.

변환규칙 5 스키마 변환시 클래스 계층 구조에 사이클이 발생하면 안된다. 또한 A 클래스를 가리키는 외래키 $F_A(B)$를 *Set<Ref>* 타입으로 변환할 수 없다. 그 이유는 1:n 관계에서 외래키가 있는 쪽이 항상 1 이기 때문이다. 변환규칙 2와 같은 조건에서 A 클래스를 B 클래스의 집합 애트리뷰트로 둘 수 없으며 변환규칙 3과 같은 조건에서 A 클래스를 하위 클래스로 둘 수 없다.

사용자가 위의 변환규칙 1, 2, 3, 4 순서에 의해 변환된 스키마를 사용자가 다시 변환할 수 있다. 즉, 변환규칙 4의 경우 사용자가 *Set<Ref>* 타입으로 변환된 것을 Ref 타입으로 변환할 수 있다. 예를 들어 두 클래스가 1:1의 관계를 가질 때도 외래키의 의미는 *Ref, Set<Ref>*로 매핑된다. 하지만 1:1인 경우에는 사용자가 *Set<Ref>* 타입을 *Ref* 타입으로 변환할 수 있다. 또한 하위 클래스나 집합 애트리뷰트로 매핑된 것을 단순히 *Ref, Set<Ref>* 타입으로 변환할 수 있다. 이와 같이 사용자가 관계형 스키마를 적절히 디자인하지 못하여 변환 과정에서 원하지 않는 형태의 객체지향형 스키마로 변환되었더라도 사용자가 다시 원하는 형태의 스키마로 변환할 수 있기 때문에 기존의 많은 관계형 스키마를 객체지향형 스키마로 변환할 수 있다. 하지만 사용자 스키마 변환 시에도 반드시 변환규칙 5를 위배하지 않아야 한다. 이와 같이 스키마 변환이 완료되면 스키마 변환 도구는 2.2절에서 설명한 SMB를 생성한다. 이것을 이용하여 5장에서 설명할 질의어 변환을 한다.

5. 질의어 변환

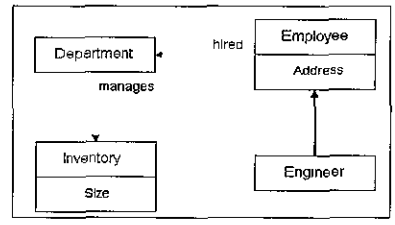

그림 5 예제 스키마

이 장에서는 질의어 변환에 관하여 설명한다. 먼저 OQL 질의어가 들어오면 질의어 그래프를 생성하고 OQL의 각 심볼을 SQL 심볼로 치환하고 중첩 질의어 변환을 위해 질의어 그래프 변환을 한 다음 SQL 문장을 생성한다. 설명의 편의를 위하여 예제 5.1의 간단한 OQL 문장의 변환을 통하여 질의어 변환을 설명한다.

예제 5.1 예제 2.1에서 관계형 스키마를 객체지향 데이타베이스 스키마로 변환하였다. 그림 5는 이중 다음 질의에 참여하는 것만 그린 것이다. Engineer 클래스는 Employee 클래스의 하위 클래스이고 Address와 Size는 애트리뷰트이다. 점선으로 표시된 hired와 manages는 다른 클래스를 참조하는 애트리뷰트로서 hired는 *Ref* 타입이고 manages는 *Set* 타입이다. 이 스키마에 대한 예제 질의어는 다음과 같다.

```
select x.Size
from x in (select y.hired.manages
    from y in Engineer where y.Address = "Seoul")
where x.Size > 500;
```

이것은 "주소가 서울인 엔지니어가 속한 부서가 관리하는 창고의 크기 중에 500 보다 큰 것들의 창고 크기를 나열하라"는 것이다. □

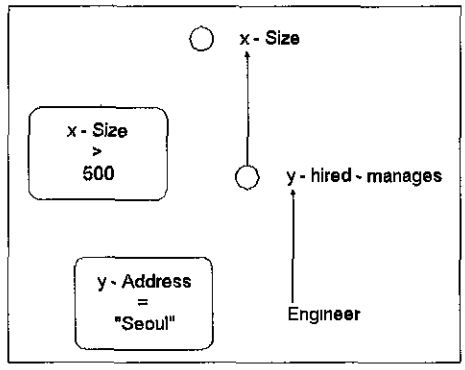

그림 6 예제 질의어 그래프

5.1 질의어 그래프 생성

먼저 입력된 질의어는 그림 6과 같이 변환된다. 원은 SELECT 절을 의미하고 화살표는 심볼을 결정하는 인자를 의미한다. 심볼 x는 y.hired.manages로 결정되고 심볼 y는 Engineer로 결정된다는 것을 알 수 있다. 점선은 조건 즉, WHERE 절에 있는 것을 의미한다.

5.2 심볼 치환

그림 6에서 보는 바와 같이 각 심볼은 다른 심볼로부터 정의된다. 이때 각 심볼들을 SQL 심볼로 바꾸어야 한다. 예제의 질의어에 나온 심볼들을 변환하면 다음과 같다.

```
x c4 Inventory
        Size c4 SIZE
<c2 Employee>
y c1 Engineer
        [c1.EMP_NO=c2.EMP_NO]
        [c2.DEPT_NO=c3.DEPT_NO]
        hired c3 Department
            [c3.DEPT_NO=c4.DEPT_NO]
            manages c4 Inventory
        Address c2 ADDRESS
Engineer c1 Engineer
```

위의 심볼 변환을 살펴보면 심볼 y는 c1 으로 변환되었다. y.hired 심볼을 결정하기 위해 두번의 죠인이 발생하였다. 하나는 하위 클래스와 상위 클래스간의 죠인이다(변환규칙 3). 즉 Engineer 클래스에는 실제 hired가 없으나 상위 클래스간의 죠인으로 구할 수 있다. 또한 hired는 *EMPLOYEE* 테이블과 *DEPARTMENT* 테이블간의 죠인으로 구해진다(변환규칙 4). 그러므로 y.hired는 두번의 죠인과 심볼 c3으로 대치된다. 심볼 y.hired.manages는 INVENTORY 테이블과 DEPARTMENT 테이블간의 죠인으로 구해지고 심볼 c4로 대치된다(변환규칙 4). y.Address는 c2.ADDRESS로 대치된다. 또한 x.Size는 c4.SIZE로 대치된다. 모든 죠인은 변환규칙에 의해서 변환된 정보를 이용하여 만들어진다.

이렇게 대치된 심볼은 SQL 문장으로 변환될 때 사용될 심볼들이다. 즉 OQL의 각 심볼들이 그에 상응하는 SQL 심볼들로 변환된 것이다. 이 변환된 심볼은 다음의 질의어 그래프 변환과정을 거친 후 마지막으로 SQL 문장으로 변환될 때 사용된다.

5.3 질의어 그래프 변환

ODMG OQL은 SELECT, FROM, WHERE 절 어느 곳에서든지 다시 중첩된 질의어가 올 수 있다. 예제는 FROM 절에 다시 SELECT 절이 중첩된 경우이다. 각 부분에서 중첩된 질의어의 의미를 살펴보면 다음과 같다. 첫째, SELECT 절에 중첩된 SELECT 절은 카티션 프러덕트(Cartesian product)의 의미를 나타낸다. 둘째, FROM 절의 SELECT 절은 심볼을 결정하는 의미이다. 즉 예제의 그림 6에서 x를 가리키는 화살표와 같이 중첩된 SELECT는 x 심볼을 결정하는 역할을 한다.

셋째, WHERE 절의 SELECT 절은 SQL 에서와 의미가 같다. 그러므로 첫 번째와 두 번째 경우에 중첩되지 않은 질의어로 변환하기 위해 질의어 그래프를 변환한다. 이때 중첩된 SELECT 절의 WHERE 절은 바깥 절로 나온다.

5.4 SQL 문장 생성

심볼 테이블에서 심볼을 결정하기 위해 참조된 테이블은 FROM 절로, SELECT 절에 나오는 심볼을 변환한 것은 SELECT 절로 나타나게 한다. 예제에서 SELECT 절에는 c4.SIZE 심볼이 FROM 절에는 ENGINEER c1, EMPLOYEE c2, DEPARTMENT c3, INVENTORY c4 심볼이 등록된다. 그리고 WHERE 절은 심볼을 만들기 위해 생성된 죠인들과 OQL에서 사용된 조건을 AND 로 연결한 결과가 된다. 그 결과는 다음과 같다.

```
SELECT c4.SIZE
FROM   ENGINEER c1, EMPLOYEE c2,
       DEPARTMENT c3, INVENTORY c4
WHERE (((c4.SIZE>500) AND (c2.ADDRESS='Seoul'))
       AND (c1.EMP_NO=c2.EMP_NO)
       AND (c2.DEPT_NO=c3.DEPT_NO)
       AND (c3.DEPT_NO=c4.DEPT_NO));
```

6. 시스템 적용 예

본 논문의 시스템에서는 객체지향 데이타베이스로 SOP[4]를 관계형 데이타베이스로 SRP[5]을 이용하였다. 그러나 외래키는 모든 관계형 데이타베이스에서 표현할 수 있으므로 어떠한 관계형 데이타베이스 시스템이라도 관계없이 모두 적용할 수 있다.

6.1 OOViewer

OOViewer는 본 논문에서 제안한 시스템을 이용하여 사용자가 터미널에서 ODMG OQL을 작성하여 질의하면 내부적으로 질의어를 SMB를 이용하여 변환한 다음 그 결과를 화면에 보여주는 시스템이다. 즉, 관계형 데이타베이스를 SQL이 아닌 OQL로 질의하는 시스템이다. 다음은 예제 6을 OOViewer를 이용하여 OQL로 관계형 데이타베이스를 검색한 것이다.

```
1> select distinct(x.Size)
2> from x in (select y.hired.manages
3> from y in Engineer
4> where y.Address = "Seoul")
5> where x.Size > 500;
------
 SIZE
 ----
 720
 829
------
```

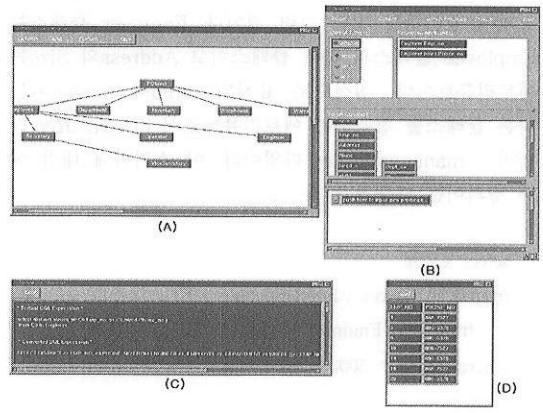

그림 7 SOPView

6.2 SOPView

SOPView는 [4]에서 제안한 시스템이다. 이것은 객체지향 데이타베이스에 대한 질의어를 쉽게 생성하기 위하여 그래픽을 이용하였다. 사용자가 스키마 그래프를 보고 질의어를 생성할 수 있기 때문에 쉽게 원하는 결과를 얻을 수 있는 시스템이다. SOPView에 본 시스템을 합쳐 객체지향 데이타베이스 뿐만 아니라 관계형 데이타베이스도 이 시스템을 통하여 질의할 수 있게 하였다. 그림 7에서와 같이 SOPView는 관계형 데이타베이스 스키마를 클래스 계층 구조로 보여주고(A), OQL로 질의어를 생성한 다음(B), 내부적으로 질의어 변환기를 거쳐(C), 관계형 데이타베이스를 접근하여 그 결과를 보여주는(D) 시스템임을 알 수 있다. 이때 스키마를 클래스 계층 구조로 보여주는 것은 SMB 데이타를 통해 이루어진다.

7. 관련연구

[8]에서 관계형 데이타베이스를 객체지향 데이타베이

4) SNU OODB Platform, 서울대 컴퓨터공학과에서 개발한 객체지향 데이타베이스 시스템

5) SNU RDB Platform, 서울대 컴퓨터공학과에서 개발한 관계형 데이타베이스 시스템

스로 이동해야 되는 이유와 이동했을 때의 문제점을 제시하고 이에 대한 대안으로 관계형 데이타베이스에 객체지향 인터페이스를 제안하였다. [12]에서는 객체지향 인터페이스 제공 방법들에 관하여 언급하고 있다.

관계형 데이타베이스의 스키마와 데이타를 C++ 응용프로그램에서 접근하는 것으로 Penguin[6, 17, 18], Persistence[7] 등이 있다. 이는 저장장치만 관계형 데이타베이스를 사용하고 사용자 인터페이스는 객체지향 데이타베이스를 이용하는 것이다. 여기서는 세단계의 스키마 단계로서 사용자 단계의 뷰와 개념적 및 시스템 단계의 뷰를 두어 C++ 인터페이스로 관계형 데이타베이스의 데이타를 접근한다. 이때 사용하는 질의어는 SQL을 확장하여 경로식(path expression) 등을 표현할 수 있게 한 것을 사용한다. 또한 관계형 데이타베이스 스키마에서 객체지향 데이타베이스 스키마 변환에 관해 [5, 14, 15, 21]에서 언급하고 있다. 이것들은 모두 관계형 데이타베이스 스키마의 의미를 해석, 확장하여 그에 대응되는 객체지향 데이타베이스 스키마로 변환한다. [15, 21]은 EER 스키마를 객체지향 스키마로 변환하는데 잘 정의된 EER 스키마만을 변환할 수 있는 것이 단점이다. [15]는 Sybase에서 기본키와 외래키의 관계를 이용하여 O_2 스키마로 변환하는데 [21]에서와 같은 문제가 발생한다. [5]는 지식 베이스(knowledge base)를 두어 변환된 스키마를 저장한다. 질의어 변환은 지식 베이스(knowledge base)를 이용한 [5]와 질의어 그래프를 변환한 [21]등이 있다. [21]은 양방향으로 질의어를 변환할 수 있는 장점이 있다.

[11]에서는 관계형 데이타베이스에 대한 객체지향적 질의어의 결과에 대한 문제점을 제기하고 외부 죠인(outer join)이 필요한 이유에 대해 언급하고 있다. 그리고 [9]에서는 UniSQL/M에서 멀티 데이타베이스 스키마에 대한 구현에 대해서 언급하고 있다.

8. 결론 및 향후 연구 계획

본 논문에서 제안한 스키마 게이트웨이는 사용자로 하여금 관계형 데이타베이스를 객체지향 데이타베이스 인터페이스, 즉 ODMG 스키마와 OQL로 접근할 수 있게 해준다. 이렇게 함으로서 관계형 및 객체지향 데이타베이스로 구성된 멀티 데이타베이스 환경에서 사용자가 하나의 데이타 모델에 관한 뷰를 가질 수 있는 장점을 제공한다. 또한 ER 모델에서 관계형 데이타베이스 스키마로 변환하면서 잃어버린 스키마 정보를 스키마 변환 과정을 통하여 다시 얻을 수 있어 스키마 이해를 증진시킴과 동시에 OQL을 이용하여 의미 있는 복잡한 질의어를 쉽게 생성할 수 있게 한다.

현재 구현된 시스템은 관계형 데이타베이스에 대한 C++ 바인딩을 지원하지 않고 질의어 검색으로만 관계형 데이타베이스를 검색할 수 있다. 앞으로 객체지향 데이타베이스의 객체 관리자를 확장하여 C++ 바인딩을 지원함으로서 객체지향 응용프로그램 내에서 객체지향 데이타베이스에 객체를 요청하듯 관계형 데이타베이스의 데이타도 객체 관리자를 통한 엑세스를 가능하게 할 계획이다. 이는 모든 환경에서 관계형 데이타베이스를 객체지향적 방법으로 접근하는 것이므로 사용자 편리성을 더욱 증진시킬 것이다. 또한 변환된 스키마에는 스키마의 의미가 많이 있으므로 의미적 질의어 최적화(semantic query optimization)를 가능하게 할 것이다.

참 고 문 헌

[1] R.G.G. Cattel, editor, *"The Object Database Standard: ODMG-93"*, Morgan Kaufmann Publishers, San Francisco, California, 1.1 edition, 1994

[2] Peter P. Chen, "The Entity-relationship model-toward a unified view of data", *ACM Transaction on Database Systems*, 1(1):9-36, 1976

[3] E.F. Codd, "A Relational Model of Data for Large Shared Data Banks", *Communication of the ACM*, 13(6):377-387, June 1970

[4] Sungwoo Jang, Suk-Ho Lee, and Hyoung-Joo Kim, "SOPView : A Visual Query and Object Browsing Environment for SOP OODBMS", *COMPSAC*, 1996

[5] Daniel A. Keim, Hans-Peter, and Andreas Miethsam, "Object-Oriented Querying of Existing Relational Databases", *International Conference on Database and Expert Systems Applications*, September 1993

[6] Arthur M. Keller, "Penguin: Object for Programs, Relations for Persistence", *submitted for publication*, April 1993

[7] Arthur M. Keller, Richard Jenson, and Shailesh Agarwal, "Persistence Software: Bridging Object-Oriented Programming and Relational Databases", *Proceedings of the ACM SIGMOD International Conference on the Management of Data*, May 1993

[8] Arthur M. Keller and Paul Turner, "Migrating to Object Data Management", *OOPSLA Workshop on Legacy Systems and Object Technology*, October 1995

[9] William Kelley, Sunit Dala, Won Kim, Tom Reyes, and Bruce Graham, *"Modern Database System"*, chapter 30, pages 621-648, ACM Press, 1995

[10] Henry F. Korth and Abraham Silberschatz,

"*Database System Concepts*", chapter 2, McGraw-Hill, 2nd edition, 1991

[11] B.S. Lee and Gio Wiederhold, "Outer Joins and Filters for Insanitation Objects from Relational Databases Through Views", *IEEE Transaction on Knowledge and Data Engineering*, 6(1), February 1994

[12] Mary E.S. Loomis, "Objects and SQL: Accessing relational databases", *Object Magazine*, September 1991

[13] Hongjun Lu, Hock Chuan Chan, and Kwok Kee Wei, "A Survey on Usage of SQL", *SIGMOD RECORD*, 22(4), December 1993

[14] Victor M. Markowitz and Johann A. Makowsky, "Identifying Extended Entity-Relationship Object Structures in Relational Schemas", *ACM Transaction on Database Systems*, 16(8):777-790, August 1990

[15] Weiyi Meng, Aqueo Kamada, and Yu-Hsi Chang, "Transformation of Relational Schema to Object-Oriented Schema", *SUNY Binghamton Tech. Rep.*, 1993

[16] Joan Peckham and Fred Maryanski, "Semantic Data Models", *Computing Surveys*, 20(3):153-189, September 1988

[17] Tetsuya Takahashi and Arthur M. Keller, "Querying Heterogeneous Object Views of a Relational Database", *Int'l Symp. on Next Generation DBMS and Their Appl. (NDA) 93*, September 1993

[18] Tetsuya Takahashi and Arthur M. Keller, "implementation of Object View Query on a Relational Database", *Data and Knowledge Systems for Manufacturing and Engineering*, May 1994

[19] P. Valduriez, "Join Indices", *ACM Transaction on Database Systems*, 12(2):218-246, 1987

[20] Ling-Ling Yan and Tok-Wang Ling, "Translating Relational Schema With Constraints Into OODB Schema", In D. Haiso et at., editors, *Interoperable Database Systems*, Elsevier Science, 1993

[21] Clement Yu, Yi Zhang, Weiyi Meng, Won Kim, et al., "Translation of Object-Oriented Queries to Relational Queries", *IEEE International Conference on Data Engineering*, March 1995

[22] Carlo Zaniolo, "The Database Language GEM", *Proceedings of the ACM SIGMOD International Conference on the Management of Data*, 1983

박 상 원

1992년 서울대학교 농토목과 졸업. 1994년 서울대학교 컴퓨터공학과 졸업. 1997년 서울대학교 컴퓨터공학과 석사. 1997년 ~ 현재 서울대학교 컴퓨터공학과 박사과정. 관심분야는 객체지향 시스템, 데이타베이스, 미들웨어.

김 형 주

1982년 2월 서울대학교 컴퓨터공학과 졸업. 1985년 8월 Univ. of Texas at Austin 전자계산학 석사. 1988년 5월 Univ. of Texas at Austin 전자계산학 박사. 1988년 5월 ~ 9월 Univ. of Texas at Austin Post-Doc. 1988년 9월 ~ 1990년 12월 Georgia Institue of Technology 조교수. 1991년 1월 ~ 현재 서울대학교 컴퓨터공학과 부교수. 관심분야는 객체지향 시스템, 사용자 인터페이스, 데이타베이스.