경영정보학연구
제11권 제2호
2001년 6월

시각적 객체지향 데이터베이스 질의어의 설계 및 질의처리기의 구현*

이 석 균**, 나 연 묵***, 서 용 무****

Design of Visual Object-Oriented Database Query Language and Implementation of the Query Processor

Lee, Suk Kyoon, Nah, Yunmook, Suh, Yongmoo

VOQL* query language, recently proposed, is a visual language for object-oriented databases. It is based on Ven Diagram and graph, so that the underlying schema structure can be naturally implied in query expressions. In VOQL*, structural relationship among the objects used in a query expression is represented graphically and thus it has formal semantics that can be inductively defined, as well as it can be used with ease.

In this paper, we proposed revised VOQL* and introduced its query processor, InQs (Intelligent Querying System). While retaining the merit of VOQL* that it allows the structural relationship among the objects to be represented visually, the revised VOQL* has another merit that users can formulate a query interactively using various forms supplied by InQs. As a query processor that translates queries in revised VOQL* into those in ODMG OQL, InQs provides an environment in which users express queries in revised VOQL* and then the system automatically translates them into those in ODMG OQL. Translation algorithm of InQs is much simpler and intuitive than other algorithms used in QUIVER and other systems, since it reflects the formal semantics of VOQL*, which is defined inductively.

* 이 논문은 1997년 한국학술진흥재단의 공모과제 연구비에 의해 연구되었음.
** 단국대학교 자연과학부 부교수
*** 단국대학교 컴퓨터공학과 부교수
**** 고려대학교 경영대학 부교수

Ⅰ. 서 론

최근에 등장하는 멀티미디어 시스템 등 다양한 응용분야는 복잡한 구조의 데이터의 표현을 필요로 하는데 이를 위한 데이터 모델의 하나로 객체 지향 데이터 모델이 등장하였다[8]. 객체 지향 데이터 모델은 상속 계층(inheritance hierarchy)과 집계화 계층(aggregation hiearchy)을 통해 데이터를 표현하는데 데이터 구조의 복잡성 때문에 효과적인 질의어의 개발이 매우 중요하다. 그동안 객체 지향 데이터 모델을 위한 질의어에 대한 연구는 주로 텍스트 기반의 질의어에 집중되어 있다. 그러나, 객체 지향 모델에서는 상속 계층과 집계화 계층을 통해 표현되는 데이터 구조의 복잡성 때문에 관계형 모델에서보다 시각질의어가 더욱 중요한 역할을 담당하나, 객체 지향 모델에서의 시각질의어에 대한 연구는 아직 초보적인 상태이다[1, 3, 5, 6, 9, 13].

객체 지향 데이터베이스를 위한 시각 질의어들의 대부분은 관계형 질의어 QBE[11]를 확장하거나 그래프에 기초하고 있다. QBE에 기초한 객체 지향 질의어는 VQL[15], PESTO[3]를 들 수 있다. VQL[15]은 QBE와 같은 템플레이트 기반의 언어로 VQL에서 경로식의 표현은 복수의 템플레이트들과 이들 사이를 연결하는 레이블과 텍스트 변수를 통해 이루어진다. 이를 좀더 발전시킨 것이 PESTO인데, 이는 복합 객체 관계를 복수의 폼과 연결 에지를 사용하여 표현한다. PESTO에서 에지로 연결된 폼 사이의 관계는 단지 클래스간의 관계만을 반영하고 있어 복합 객체들 간의 접근 경로를 의미하는 경로식[7]의 시맨틱을 제대로 전달하지 못한다[9].

그래프에 기초한 객체지향 질의어로는 VQL[15], QUIVER[4], VOQL[9], VOQL*[10]등이 있는데, VQL은 다양한 그래프 기반의 구성 요소를 제공하며 이들을 통해 부정, 전체정량자, 재귀적 질의문, 스키마 질의 등을 처리할 수 있다. 그러나, 경로식 및 조인 연산 등의 표현에 텍스트 변수들이 과다하게 사용되며 관계형 모델을 기본으로 하는 등 객체 지향 질의어로는 적합하지 않다. 이에 대해 QUIVER는 클래스 이름, 속성명과 메소드 명을 제외하고는 텍스트를 거의 사용하지 않으며 풍부한 메뉴와 아이콘들을 통하여 편리한 사용자 인터페이스를 제공한다. 그러나 QUIVER는 형식 시맨틱이 정의되고 있지 않다. 이로 인해 번역 알고리즘이 직관적이지 않을 뿐 아니라 전체 정량자, 부정 등의 개념적 확장에 한계가 있다. 특히 변수의 개념을 제공하고 있지 못해 경로식과 같은 주요 개념을 제대로 시각화하지 못하고 있다.

VOQL은 Harel의 Higraph[14]에 기초한 시각 질의어로 OOPC(Object-Oriented Predicate Calculus)[2]에 기초한 형식 시맨틱을 갖는다. VOQL에서는 경로식(path expression)을 객체 레벨에서 시각화함으로써 경로식의 표현이 명확하며, 집합 관련 조건 연산을 벤다이어그램(Ven Diagram)에 기초하여 표현함으로 집합 관련 조건 연산의 표현이 자연스럽고 간결하다. 그러나 VOQL에는 형식 시맨틱의 정의의 편이를 위해 모든 속성을 다중치 속성(multi-valued attribute)으로 가정하여 항상 집합으로 표현하고 있어 실용 언어로는 적합하지 않았다. 이를 개선한 것이 VOQL*로 속성의 다중 값뿐만 아니라 단일 값을 시각적으로 표현할 수 있도록 하고 시각 변수의 개념을 도입하였다. VOQL*의 가장 큰 장점은 질의어의 구조가 그 형식 시맨틱을 정의하는 OOPC의 귀납적인 문법 구조를 그대로 반영하므로 질의문의 시맨틱이 명확하다는 점이다.

본 논문에서는 그래프 기반의 VOQL*에 QBE 또는 MS ACCESS와 같은 폼 기반의 질의어의 특성을 첨가한 수정된 VOQL*를 소개하고 이를 구현한 프로토타입 지능형 질의 시스템(InQs: Intelligent Querying System)의 구조와 특징을 설명한다. 본 논문의 구성은 다음과 같다. 2장은

연구 배경으로 ODMG 데이터 모델과 질의어 ODL (Object Definition Language)과 OQL(Object Query Language)을 소개하며 VOQL*의 설명을 위해 예제 스키마를 설명한다. 3장에서는 VOQL*에 대한 소개를 하였고, 4장에서는 수정된 VOQL*와 이를 구현한 프로토타입 시스템 InQs의 구조를 설명하고, InQs에서 사용되는 VOQL* 질의문의 OQL로의 번역 알고리즘을 소개한다. 끝으로 5장에서는 결론을 내리고 향후 연구방향을 제시한다.

Ⅱ. 연구 배경 및 예제 스키마

본 논문에서 제안한 프로토타입 질의 시스템(InQs)에서는 수정된 VOQL*로 표현된 질의문을 ODMG 2.0[12]에 정의된 객체지향 질의어인 OQL로 변환한다. 따라서 본 절에서는 우선 ODMG 2.0의 주요 구성요소인 객체모델, ODL 그리고 OQL에 대하여 소개한다.

2.1 ODMG 2.0의 객체 모델과 ODL

ODMG 2.0 객체 모델의 기본적인 개념은 객체로 이는 실세계를 구성하는 요소들을 나타내며 상태(state)와 행위(behavior)로 구성된다. 객체의 상태는 성질(property)들의 집합으로, 객체의 행위는 연산(operation)들의 집합으로 정의된다. 객체는 고유 식별자(oid)를 갖고 있으며 타입에 의해 분류된다. 즉 하나의 타입에 속하는 모든 객체는 공통적인 성질들과 공통적인 연산들을 가지고 있다.

객체의 성질은 속성(attribute)과 관계(relationship)으로 구성되며 속성은 객체의 정적인 상태를 나타내는 값을 의미한다. 한편 관계는 두 개의 객체들 사이에 정의되며 객체들 사이의 연관관계를 나타낸다. 관계는 두 객체들 사이에 접근할 수 있는 항해경로(traversal path)를 제공하며 역방향 항해경로는 'inverse'라는 키워드를 통해 표현한다. 이와 같은 관계는 집계화 계층을 구성하며 이를 통해 복합객체가 표현된다.

ODMG 객체 모델에서는 타입도 객체로 간주되며 속성을 가지고 있는데, 이를 타입속성이라고 한다. 상위타입, 익스텐트(extent) 그리고 키 등이 타입속성이 된다. 상위타입과 하위타입과의 사이에는 ISA 관계가 성립하며, 타입들이 ISA 관계에 의해 계층구조를 이룬다. 이때, 상위타입의 특성과 연산은 계층구조를 따라 하위타입에 상속된다. 하나의 타입에 속하는 모든 인스탄스 객체들의 집합을 익스텐트라고 하며 이 타입의 여러 속성들 중에서 모든 인스탄스 객체들을 식별하는데 사용될 수 있는 하나 이상의 속성을 키라고 한다.

타입의 정의에는 타입특성 외에도 인스탄스 속성과 인스탄스 연산이 있다. 타입에 속하는 모든 객체는 인스탄스 속성에 해당하는 값을 가지고 있으며, 이들 각 객체에 인스탄스 연산을 적용할 수 있다.

ODL은 OMG CORBA의 IDL을 확장한 객체 데이터 정의어로 타입들의 특성과 연산들을 정의하는데 사용된다. 이 때, 연산의 구현에 대한 구체적인 내용은 언급을 하지 않고 다만 연산의 시그니춰(signature)만 정의한다.

<그림 2-1>는 Person, Instructor, Department, Course 그리고, Student 타입으로 구성된 데이터베이스 스키마를 ODL로 기술한 것이다. 이는 제3절에서 VOQL*의 개요를 설명할 때 사용된다.

2.2 OQL

OQL은 ODMG의 데이터 모델을 지원하는 선언적 질의어로 기본적인 문법구조는 SQL과 마찬가지로 select-from-where로 구성되며 질의문 내에서 메소드의 사용이 가능하다. OQL에서의 질의문들은 단일 객체 또는 객체들의 컬렉션, 리터럴(literal) 또는 리터럴의 컬렉션을 질의 결과로 반환한다. 이외에도 OQL에서는 select 절,

from 절, 그리고 where 절 어디에서나 내포질의(nested query)가 가능하며 클래스들 사이의 복합 객체 관계가 relationship으로 정의되어 있는 경우 join연산 대신 경로식을 통해 접근할 수 있으며 질의 결과를 다른 질의에서 참조할 수 있다. (자세한 내용은 ODMG 2.0을 참조바람)

<그림 2-2>는 <그림 2-1>의 스키마를 다이어그램으로 표현한 것이다. 굵은 화살표는 클래스

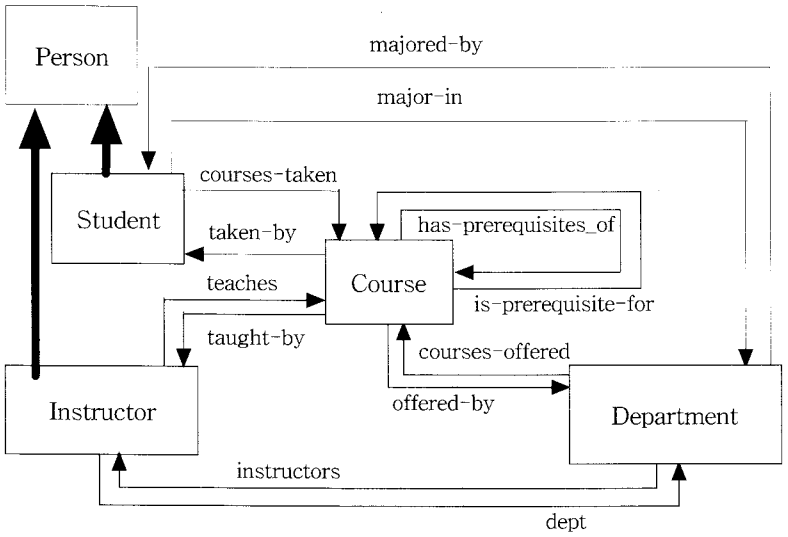
```
class Person (extent Persons key ssn)
{ attribute string ssn;
  attribute string name;
  attribute string address:};

class Instructor extends Person (extent Instructors)
{ attribute long salary;
  attribute string rank;
  attribute set<string> degrees;
  relationship Department dept          inverse Department :: instructors;
  relationship set<Course> teaches      inverse Course :: taught_by;
 };

 class Department (extent Departments keys dno, name)
 { attribute long dno;
   attribute string name;
   relationship set<Instructor> instructors          inverse Instructor :: dept;
   relationship set<Courses> courses_offered         inverse Course :: offered_by;
   relationship set<Students> majored_by             inverse Student :: major_in;
 }
 class Course (extent Courses keys code, name)
 { attribute string code;
   attribute string name;
   relationship Department offered_by          inverse Department :: courses_offered;
   relationship Instructor taught_by           inverse Instructor :: teaches;
   relationship set<Course> is_prerequisite_for
         inverse Course :: has_prerequisites_of;
   relationship set<Course> has_prerequisites_of inverse Course :: is_prerequisite_for;
   relationship set<Student> taken_by      inverse Course :: courses_taken;
 }

 class Student extends Person (extent Students keys id)
 { attribute string id;
   relationship set<Course> courses_taken inverse Course :: taken_by;
   relationship Department major_in inverse Department :: majored_by;
   }
```

<그림 2-1> Class들의 정의 예



<그림 2-2> <그림 2-1>의 스키마 다이어그램

들간의 ISA 관계를 나타내고 레이블이 있는 보통 화살표는 두 클래스간의 relationship 관계를 의미하는 속성을 나타낸다. <그림 2-3>은 OQL로 표현된 질의문들을 나타내며 이들은 경로식, 집합연산자, 내포질의를 예시하고 있다.

<그림 2-3>의 예제 질의 (a)와 (f)는 관계형 질의어 SQL 질의와 아무런 차이가 없으나. 질의 (b)는 x.dept.dno, x.dept.name과 같은 경로식을 사용하고 있다. 질의 (c)는 Course의 relationship has_prerequisites_of는 집합을 나타내는 속

```
(a) select x.name, x.address from x in Person where x.ssn = 12345;
(b) select x.name, x.dept.name from x in Instructor
        where x.salary > 30000 and x.dept.dno = 312;
(c) select y.code, y.name from x in Course, y in x.has_prerequisites_of
        where x.name = "전산학개론" and y.taught_by = "홍길동"
(d) select y.name from x in Department, y in x.majored_by
        where x.name = "전산학";
(e) select y.name from x in Department, y in Student
        where y.courses_taken < x.courses-offered
(f) select y.name from x in Instructor, y in Student
        where x.name = "홍길동" and x.dept = y.major_in ;
(g) select x.name from x in Course, y1 in Instructor, y2 in Instructor
        where y1.name = "이교수" and y2.name = "정교수"
        and x in (y1.dept.courses-offered intersect y2.dept.courses-offered);
(h) select x.name from x in Instructor
        where x.teaches  < (select y.courses_offered
                            from y in Department
                            where y.dno = "100");
```

<그림 2-3> 예제 질의

성으로 변수 y가 x.has_prerequisites_of에 바인딩됨을 예시하고 있다. 질의 (e), (g), (h)는 집합 관련 조건을 사용되고 있다. 이들 질의문들의 일부가 VOQL*의 질의 예제에 사용된다.

Ⅲ. 객체지향 시각 질의어(VOQL*)

VOQL*는 객체 지향 데이타베이스를 위한 시각 질의어로 그래프와 벤 다이어그램에 기초한 문법과 귀납적으로 정의되는 형식 시맨틱을 갖는 특징이 있다[10]. 데이타의 중첩된 구조는 그래프로 표현되고 집합 관련 조건은 벤 다이어그램을 통하여 표현하므로 VOQL*로 표현된 질의는 다른 질의어에 비해 직관적이다. VOQL*의 형식 정의는 [10]에서 찾아볼 수 있다. 본 절에서는 VOQL*의 기본 구성 요소와 관련 예제를 설명한다.

3.1 VOQL*의 개요

3.1.1 VOQL*의 기본 구성 요소

VOQL*의 문법은 벤다이어그램과 그래프에 기초한 **시각적 구성 요소**와 텍스트에 기초한 **텍스트 구성 요소**로 정의된다. 시각적 구성 요소에는 블랍(blob), 서브블랍(subblob), 시각 변수(visual variable), 시각 요소(visual element), 에지(edge)와 스텀프 블랍(stump blob)이 있으며 이들은 질의의 구조를 시각화하는데 사용된다. 텍스트 구성 요소는 블랍, 에지, 스텀프 블랍과 같은 시각적 구성 요소의 의미를 전달하는 **레이블**과 조건식의 표현에 필요한 **비교 연산자**(>, >=, =, <, <=)와 **상수 값** 등으로 구성된다. 시각적 구성 요소들의 표기와 의미는 다음과 같이 정의된다.

블랍 : 블랍은 사각형으로 나타내며 클래스에 속한 모든 객체들을 의미하는 익스텐트의 시각적 표현이다. 익스텐트를 구성하는 객체들의 컬렉션은 집합, 다중집합, 리스트 및 배열 등으로 표현할 수 있는데, VOQL*에서는 이들을 추상화하여 블랍으로 표현한다.

서브블랍 : 블랍보다 작은 사각형으로 표현되는 서브블랍은 속성의 결과로 반환되는 객체들의 컬렉션 또는 리터럴들의 컬렉션을 시각적으로 표현할 때 사용된다. 이때 객체들의 컬렉션은 그 객체들이 속한 익스텐트를 의미하는 블랍 안에 내포되어 표시되나 리터럴 컬렉션은 블랍에 내포되지 않는다.

시각변수 : 검은색의 작은 원으로 표현되며 OQL의 객체 변수에 대한 시각적 표기이다. 항상 블랍 또는 서브블랍과 함께 사용되어 객체 변수의 바인딩을 표현하게 된다.

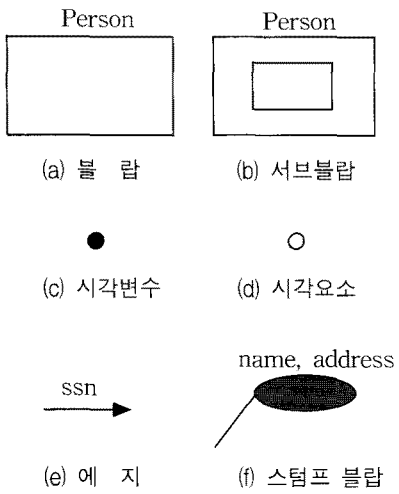
시각요소 : 투명한 작은 원으로 표현되며 하나의 객체 또는 리터럴을 의미한다. OQL의 경로식의 중간 결과 또는 최종 결과를 시각화한 것이다.

에지 : 에지는 방향성이 있는 화살표로 객체의 속성에 대한 시각적 표현이다. 객체를 시각 변수(또는 시각 요소)로 나타낼 때 이 객체의 속성은 시각 변수(또는 시각 요소)로부터의 에지로 나타낸다. 에지의 위쪽에 있는 레이블은 속성 이름을 의미한다.

스텀프 블랍 : 스텀프 블랍은 시각 변수 또는 시각 요소로부터 무방향 에지로 연결된 타원형으로 표현되며 프로젝트될 속성들을 나타낼 때 사용한다. 프로젝트될 속성들의 이름은 레이블로 표현된다.

시각적 구성 요소들의 예는 <그림 3-1>과 같이 표현된다. <그림 3-1>의 (a), (b), (e), (f)의 레이블들은 각각 익스텐트 이름, 익스텐트 이름,

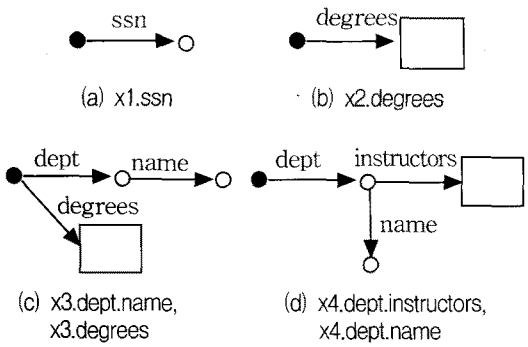
속성 이름과 프로젝트될 속성 이름 리스트를 의미한다. (b)의 서브블랍은 익스텐트 Persons의 부분 컬렉션을 나타낸다.



<그림 3-1> 시각적 구성 요소

3.1.2 VOQL*의 경로식의 표현

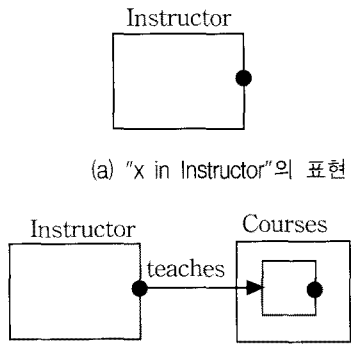
객체 변수와 마침표로 연결된 일련의 속성으로 표현되는 OQL의 경로식은 VOQL*에서 시각 변수, 시각 요소, 서브블랍과 에지에 기초하여 표현된다. VOQL*의 경로식은, 시각 변수를 루트노드(root node)로, 시각요소를 중간노드로, 그리고 시각요소나 서브블랍을 리프노드(leaf node)로 하는 트리구조를 이룬다. 그리고, 노드들을 연결하는 에지는 속성을 나타내는 레이블을 갖는다(<그림 3-2> 참조). <그림 3-2>의 (c), (d)의 예들은 동일한 객체 변수가 사용되는 경우, 다수의 OQL 경로식들이 하나의 트리 구조의 VOQL* 경로식으로 표현됨을 보이고 있다.



<그림 3-2> VOQL*의 경로식 표현의 예

3.1.3 VOQL*의 시각 변수의 바인딩의 표현

OQL에서 객체 변수가 컬렉션에 바인딩됨과 마찬가지로, VOQL*에서 시각 변수는 익스텐트를 의미하는 블랍 또는 익스텐트의 부분 집합을 의미하는 서브블랍에 바인딩된다. <그림 3-3> (a)에서는 시각 변수가 익스텐트 Instructors에 바인딩하는 것을, (b)에서는 경로식의 반환 결과인 서브블랍에 시각 변수가 바인딩됨을 보여준다. 이들 시각 변수들은 OQL로의 번역시 각각 서로 다른 객체 변수들로 해석된다. 또한 서브블랍이 익스텐트 Courses에 내포되어 있어 서브블랍은 Courses의 부분 컬렉션임을 나타내고 있다.

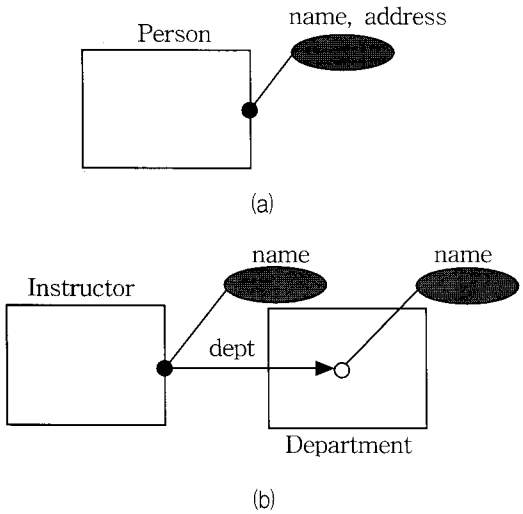


<그림 3-3> 시각 변수 바인딩의 예

3.1.4 VOQL*의 프로젝션의 표현

VOQL*의 프로젝션은 스텀프 블랍을 시각 변수 또는 시각 요소에 연결하고 프로젝션시킬 속성들의 이름을 스텀프 블랍의 상단에 레이블로 표현함으로 이루어진다. 이때 시각 변수나 시각 요소는 프로젝션시키고자 하는 객체를 의미하며

레이블로 표현되는 속성들은 이 객체의 속성들이어야 한다. <그림 3-4> (a)는 시각변수에 대한 프로젝션이고 <그림 3-4> (b)는 시각 요소에 대한 프로젝션의 예이다. 특히 <그림 3-4> (b)의 경우, 시각 요소는 블랍에 내포되어 있는데, 이는 dept라는 속성이 Department 클래스에 정의되어 있어 시각 요소가 나타내는 객체는 Department라는 익스텐트에 포함됨을 의미한다. 이의 자세한 내용은 시각화된 조건식에서 다시 설명한다.



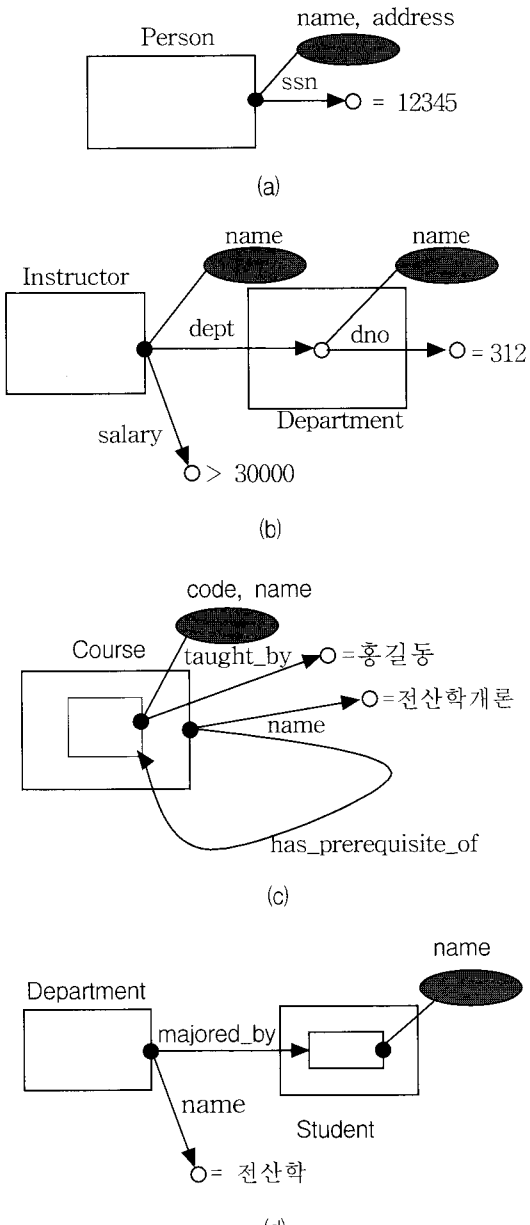
<그림 3-4> 프로젝션의 예

<그림 3-4>의 VOQL* 질의들을 OQL로 번역하면 각각 다음과 같다.

-- select x.name, x.address from x in Person;
-- select x.name, x.dept.name from x in Instructor;

3.1.5 VOQL*의 조건식의 표현

VOQL*에서 조건식의 표현은 벤 다이어그램의 집합 표기에 기초한 *시각화된 조건식*과 *텍스트 기반 조건식*으로 구분된다. 텍스트 기반 조건식은, 시각 요소나 서브블랍을 *vt*로, 상수를 *c*라 하고 텍스트 기반 조건 연산자를 θ로 표기할 때, *vt* θ *c* 또는 *vt* θ *vt*의 형식으로 표현된다. 이때 시각 요소나 서브블랍은 리터럴 또는 리터럴의 컬렉션을 의미한다. <그림 3-5>의 (a)와 (b)는 <그림 3-4>의 예에 텍스트 기반 조건을 추가한 것이다. *vt* θ *vt* 형식의 조건식 표현은 [10]에 자세히 설명되어 있다.

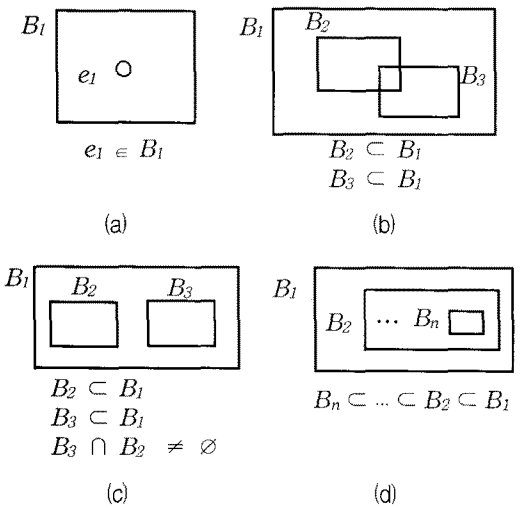


<그림 3-5> 텍스트 기반 조건식을 사용한 VOQL* 질의문의 예.

<그림 3-5>의 VOQL* 질의문들은 텍스트 기반 조건식을 적용한 것으로 이들을 OQL로 번역하면 다음과 같다.

```
(a) select  x.name, x.address
    from    x in Person
    where x.ssn = 12345;
(b) select  x.name, x.dept.name
    from    x in Instructor
    where x.salary > 30000 and
            x.dept.dno = 312;
(c) select  y.code, y.name
    from    x in Course,
            y in x.has_prerequisites_of
            where x.name = "전산학개론" and
            y.taught_by = "홍길동";
(d) select  y.name
    from    x in Department,
            y in x.majored_by
    where x.name = "전산학";
```

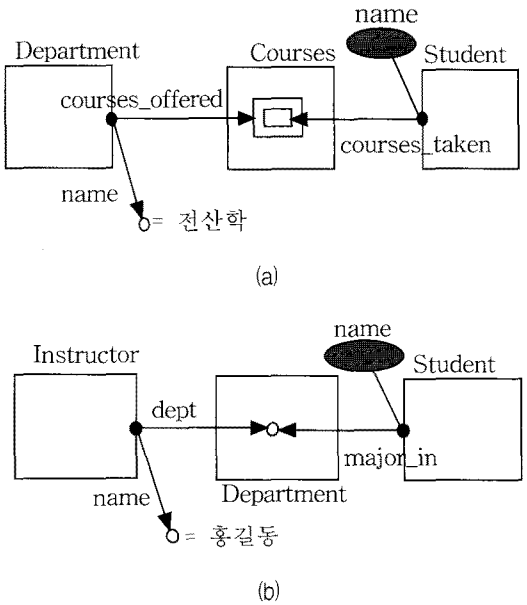
벤다이어그램에 기초한 시각화된 조건식은 객체 식별자를 가지고 있는 객체 또는 객체들의 컬렉션들 사이의 조건식을 나타낼 때 사용된다.



<그림 3-6> 집합 관련 시각화된 조건식의 정의

시각화된 조건식의 정의는 <그림 3-6>에 주어져 있다. <그림 3-6>의 레이블 B_i, e_i은 조건식의 의미를 정의하기 위해 사용되는데 B_i는 블랍 또는 서브블랍이 의미하는 컬렉션을, e_i는 시각 요소가 의미하는 객체를 나타낸다. <그림 3-6> (a)는 객체와 컬렉션과의 멤버쉽 조건을 나타내고 있고 (b)-(d)는 블랍 또는 서브블랍과의 다양한 부분 집합 조건을 표현한 것이다.

<그림 3-6>은 기본적인 조건식의 정의를 표현하고 있다. 보다 다양한 시각화된 조건식의 정의는 [10]에서 찾아볼 수 있다. <그림 3-6>의 조건식에 나타나는 시각 요소, 서브블랍은 VOQL* 경로식들의 일부분이 될 수 있어 다양한 조건을 생성할 수 있다. <그림 3-7> (a)에서 다수의 경로식들이 시각화된 조건식을 구성하는 예를 제시한다. 뿐만아니라 서로 다른 경로식들이 시각 요소나 서브블랍을 공유할 수 있는데 공유된 시각 요소나 서브블랍의 존재는 객체 식별자 또는 객체식별자들의 컬렉션의 동등 비교를 의미한다 (<그림 3-7> (b)와 그 OQL 번역을 참조바람).



<그림 3-7> 시각화된 조건문을 포함한 VOQL* 질의의 예

```
(a) select  y.name
    from    x in Department,
            y in Student
    where y.courses_taken
            < x.courses-offered ;
(b) select  y.name
    from    x in Instructor,
            y in Student
    where x.name = "홍길동" and
            x.dept = y.major_in ;
```

<그림 3-5>의 (b)와 (c)는 각각 <그림 3-6>의 (a)와 (d)의 조건식을 포함하고 있으나 포함하고 있는 블랍이 속성이 정의된 클래스의 익스텐트이므로 이들 조건식으로 인하여 where절에 추가되는 조건이 없다.

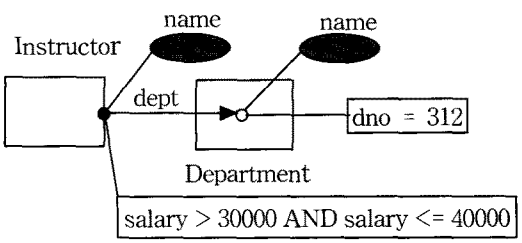
Ⅳ. 프로토타입 질의처리 시스템: InQs

InQs(Intelligent Quering System)는 VOQL*를 구현한 프로토타입 시스템으로 메뉴 구동 방식을 통한 지능형 질의 시스템이다. InQs는 대화형 시스템으로 사용자가 쉽게 질의를 생성하도록 하며, 생성된 질의문은 시각화된 표기를 사용하므로 직관적이며 질의문에 자동적으로 스키마 정보가 반영되는 특성이 있어 그 의미를 쉽게 이해할 수 있다.

4.1 VOQL*의 수정

InQs에서는 사용자의 편의와 직관성을 위해 그래프 기반의 VOQL*의 장점을 살릴 뿐 아니라 텍스트 기반의 QBE의 장점도 적절히 보완하기 위해 VOQL*의 시각요소의 정의와 텍스트 기반 조건식의 정의를 다음과 같이 수정하였다.

시각요소는 리터럴이 아닌 객체만을 표현하는 것으로 정의하며 따라서 텍스트 기반의 조건식은 더 이상 시각요소와 상수와의 비교가 아니라 <그림 4-1>의 예와 같이 표현된다.



<그림 4-1> 수정된 VOQL*의 질의문

이는 시각요소를 객체의 표현으로만 제한하여 과도하게 많은 노드가 생성되어 화면이 복잡하게 됨을 방지하며 AND나 OR연산과 같은 관계 연산의 경우 하나의 식으로 표현하게 하는 장점이 있다.

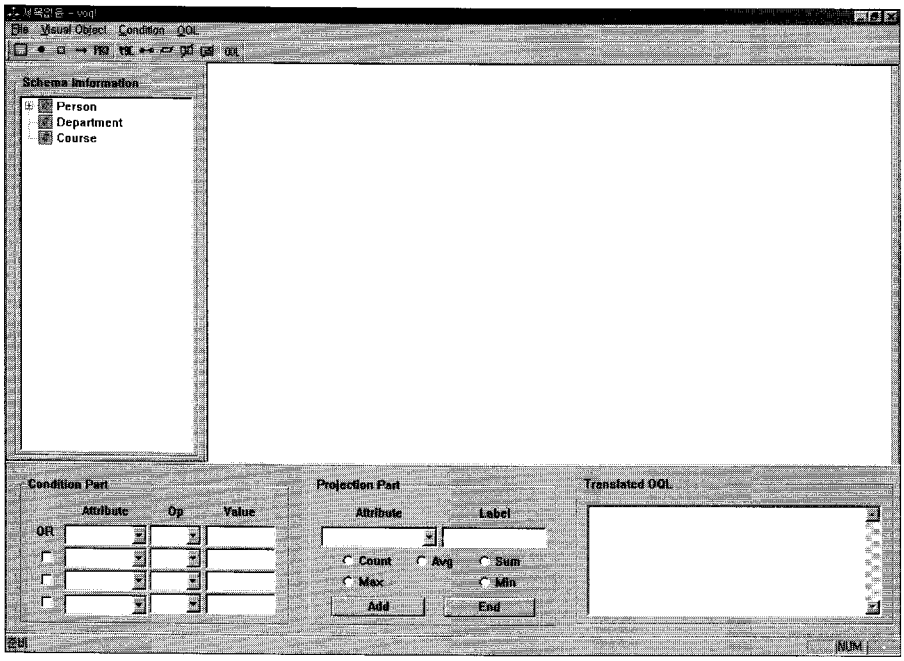
VOQL*에서 에지는 객체의 속성을 나타낼 때 사용되었으나, 수정된 VOQL*에서는 복합객체의 관계(relationship) 즉 객체들간의 관계를 표현할 때 사용된다.

4.2 InQs의 시스템 구성

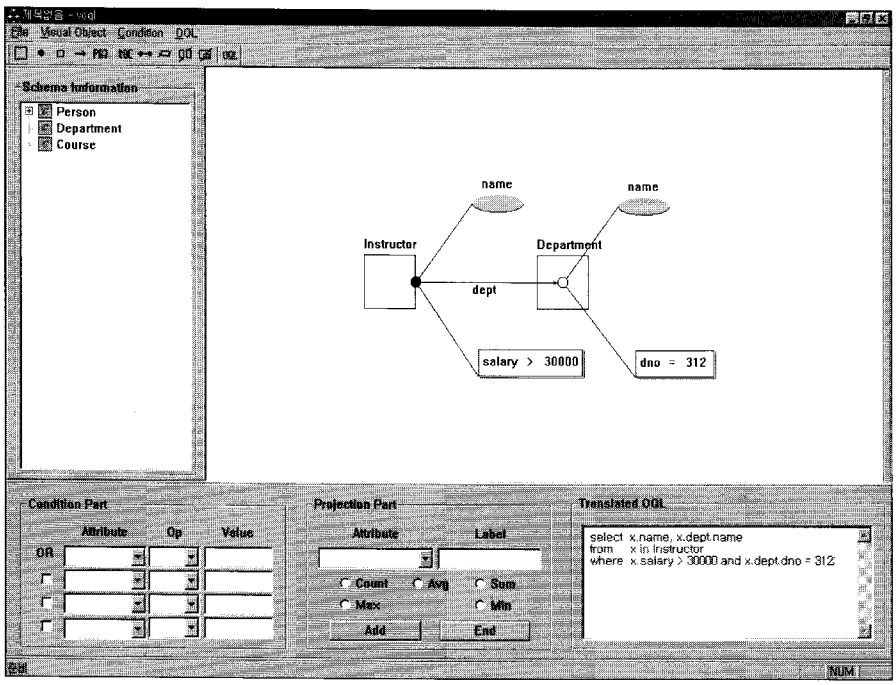
InQs의 화면 구성은 크게 메뉴, 스키마 정보 윈도, 질의 생성 윈도, 조건식 명세 윈도, 프로젝션 명세 윈도와 OQL 질의문 윈도로 구성된다. 메뉴는 객체 생성 메뉴, 프로젝션 생성버튼, 조건식 생성 메뉴, OQL 번역버튼으로 구성되어 있다(<그림 4-2>을 참조).

메뉴는 화면의 상단에 위치하여 블랍, 에지, 시각변수의 생성버튼들로 구성된 객체 생성메뉴, 시각 조건의 생성버튼들로 구성된 조건식 생성 메뉴 그리고 VOQL*의 질의문이 완성되었을 때 OQL로 번역을 수행하는 OQL번역메뉴로 구성된다.

객체 생성메뉴는 사용자로 하여금 다양한 객체 생성버튼을 사용하여 질의 생성 윈도에 VOQL* 질의문을 생성할 수 있도록 한다. 블랍 생성버튼을 눌러 블랍을 생성하는데, 이때 익스텐트



<그림 4-2> InQs 시스템 화면 구성



<그림 4-3> InQs 질의 처리기의 예제 질의 표현

이름을 선택하여 블랍을 생성한다. 그리고 시각변수 생성버튼을 눌러 블랍에 바인딩할 시각변수를 생성한다. 생성된 시각변수로부터 에지 생성버튼을 눌러 경로식을 생성하는데 에지 생성버튼을 누르면 스키마 정보에 기초한 속성 리스트를 화면에 보여준다. 속성 리스트 중 하나의 속성을 선택하면 선택된 속성의 타입에 따라 시각요소 또는 서브블랍이 생성된다. 생성된 시각요소로부터 경로식의 생성을 계속해 나갈 수 있다. 시각변수의 생성버튼을 눌러 서브블랍에 대해 시각변수를 바인딩시킬 수도 있다.

조건식 생성메뉴는 3.2절에서 설명한 텍스트 기반 조건식 생성버튼과 다양한 집합 관련 시각화된 조건식들의 생성버튼들로 구성되어 있다. 텍스트 기반 조건식을 생성하기 위해서는 질의 윈도에서 조건에 참여하는 시각변수 또는 시각요소를 선택한 후 텍스트 기반 조건식 생성버튼을 누르면 된다. 이때 화면 하단에 있는 조건식 명세 윈도가 활성화되며 대화식으로 조건을 표현하여 조건식을 완성한다. 시각화된 조건식을 표현하는 경우, 참여하는 시각변수 또는 서브블랍들을 선택한 후 해당하는 조건 생성버튼을 눌러 조건을 표현한다.

프로젝션 생성버튼은 프로젝션될 속성을 선택하게 한다. 우선 질의 윈도에서 시각변수나 시각요소를 선택한 후 프로젝션 생성버튼을 눌러 화면 하단의 프로젝션 상세 명세 윈도를 활성화하여 속성 리스트로부터 프로젝션될 속성들을 선택한다.

OQL 번역메뉴는 OQL 번역버튼으로 구성되는데, 객체 생성메뉴와 조건식 생성메뉴를 사용하여 질의 윈도에 질의문을 완성한 후 OQL 번역버튼을 누르면 OQL 질의문 윈도에 OQL에 질의문을 생성하게 한다.

본 시스템을 구성하기 위해 기반 환경으로는 MS Window98를, 구현 프로그래밍 언어는 C++로 MS Visual Studio(visual C++)를 사용하였다.

4.3 예제 질의

본 절에서는 3.3절에서 보였던 예제 질의문을 사용하여 InQs 질의 처리기를 이용하여 질의문을 생성하고 번역하는 부분까지 실행하여 보겠다.

예제 질의문 :

```
select x.name, x.dept.name
from x in Instructor
where x.salary > 30000 and x.dept.dno = 312;
```

위의 예제 질의문을 다음 <그림 4-3>과 같이 표현할 수 있다. 번역된 부분은 오른쪽 하단에서 보여진다.

4.4 번역 알고리즘

지금까지는 VOQL* 질의어를 통해 어떻게 질의문을 표현하는가에 대하여 설명하였다. 본 절에서는 VOQL*로 표현된 질의문을 OQL로 변환하는 알고리즘을 설명한다. 우선 번역 알고리즘에서 사용되는 주요 변수 및 함수들에 대해 설명한다.

(1) 질의문의 내부 표현은 query라는 전역 변수를 통해 나타내며 알고리즘에서 직접 접근 가능하도록 한다.

VOQL*_query 타입의 전역 변수 query는 VOQL* 질의문을 표현하는 자료 구조로 시각변수, 시각요소, 조건식, 스텀프블랍, 블랍, 서브블랍 등 VOQL* 구성 요소들을 그래프 형태로 표현한 자료 구조이다. 이들 VOQL* 구성 요소들에는 id와 label이라는 필드가 있다. id 필드는 항상 유일한 값을 가지며 각 구성 요소의 식별자로 쓰인다. label 필드는 각 구성 요소에 대한 OQL 번역에 해당되는 문자열을 나타낸다. id 필드의 값은 VOQL* 질의문의 생성시 질의문

편집기에 의해 자동 부여되며 label 필드의 값은 generate_labels_of_all_components 함수 호출 시 생성된다. 시각변수에는 visited라는 불리언 타입의 필드가 있어 모든 시각변수는 visited가 거짓(false)으로 초기화되는데, 시각변수가 from-clause에 포함될 때 참(true)으로 바뀐다.

(2) generate_labels_of_all_components()

generate_labels_of_all_components 함수는 VOQL* 질의문의 모든 구성 요소에 다음과 같이 label을 생성한다.

① 모든 블랍에 대응하는 익스텐트 이름의 label을 생성한다.
② 모든 시각변수에 변수 명을 할당하고 변수 명에 해당하는 label을 생성한다. 같은 변수 명이 서로 다른 시각 변수에 사용될 수 없도록 변수 명을 할당해야 한다.
③ 모든 시각변수에 대해, 시각변수로부터 시작되는 VOQL* 경로식에 속한 모든 시각요소와 서브블랍에 대응하는 OQL 경로식에 해당하는 label을 생성한다.
④ 시각변수, 시각요소, 서브블랍의 label들을 사용하여 모든 조건식의 label을 생성한다.
⑤ 시각변수, 시각요소, 서브블랍의 label들을 사용하여 모든 프로젝션의 label을 생성한다.

(3) make_var_id_list()

vid_list는 시각변수의 식별자의 리스트를 위한 전역변수로 시각변수의 식별자의 리스트는 다음의 함수 make_var_id_list에서 생성된다.

<visual_var_id> make_var_id_list(VOQL*_ query query)

함수 make_var_id_list는 query로부터 블랍에 바인딩된 모든 시각 변수의 식별자(id)를 식별자 리스트 리스트(vid_list)에 삽입한다.

(4) str1 || str2

문자열 str1의 끝에 문자열 str2를 연결한다 (string concatenation).

(5) label(id)

식별자 id가 나타내는 VOQL*의 구성 요소에 대한 레이블(label)을 반환한다.

(6) binding_blob_id(vid)

식별자 vid가 나타내는 시각변수가 바인딩하는 블랍 또는 서브블랍의 식별자를 반환한다.

(7) remove_last_comma(clause)

clause의 문자열로부터 마지막 ","가 있으면 이를 삭제한다.

(8) remove_last_and(clause)

clause의 문자열로부터 마지막 "AND"를 삭제한다.

(9) find_condition_id_list_related_to(vid)

식별자 vid가 나타내는 시각변수와 관련된 각 조건식의 식별자 리스트들을 반환하다.

(10) find_proj_id_list(vid)

식별자 vid의 시각변수가 관련된 프로젝션들의 식별자 리스트들을 반환한다.

위에서 설명한 함수와 변수를 기반으로 다음에 번역 알고리즘을 설명한다.

이제, 이상의 알고리즘을 <그림 4-3>의 VOQL* query의 예를 통해 설명한다. <그림 4-3>의 예제 질의의 생성시 주어진 시각변수의 id가 v1으로

```
VOQL*_query query; /* 전역변수로 VOQL* 질의문의 내부 표현 */
string generate_OQL_query(void) {
 visual_var_id vid;
 list<visual_var_id> vid_list;
 /* 시각변수들을 위한 식별자 리스트의 선언 */
 generate_labels_of_all_components(query);
 /* query에 속한 모든 블랍, 시각변수, 시각요소, 서브블랍, 조건식, 프로젝션들의 레이블을 생성
*/
 vid_list = make_var_id_list(query);
 /* 블랍에 바인딩된 시각변수들의 식별자 리스트 생성 */
 return (generate_OQL_sub_query(vid_list));
}

string generate_OQL_sub_query(list<visual_var_id> vid_list) {
/* 시각변수의 식별자 리스트(vid_list)에 기초하여 번역이 이루어지며 그 결과(OQL string)를
   반환한다. */
visual_var_id vid;                    /* 시각변수 식별자의 선언 */
condition_id cid;                     /* 조건식 식별자의 선언 */
projection_id pid;                    /* 프로젝션 식별자의 선언 */
list<visual_var_id> temp_list;
list<visual_var_id> sel_dep_var_list;
/* 선택 종속 변수들의 id 리스트 */
/* temp_list와 sel_dep_var_list는 empty list로 초기화됨 */

 /* OQL문의 구성 요소들의 초기화 */
 string select_clause = "SELECT ";
 string from_clause = "FROM ";
 string where_clause = "WHERE ";

 for vid ∈ vid_list {
    remove(vid_list, vid);
    set_visited(vid);
    from_clause = from_clause || label(vid) || " IN " || label(binding_blob_id(vid)) || ", ";
    for pid ∈ find_proj_id_list(vid)
       select_clause = select_clause || label(pid) || ", ";
    for cid ∈ find_condition_id_list_related_to(vid) {
      if (cid가 "Ei θ c" 형태의 조건을 나타냄)
         where_clause = where_clause || label(cid) || " AND ";
      if (cid가 "Ei θ Ej" 형태의 조건을 나타냄) AND
        (Ei과 Ej의 시각변수들이 다 방문(visited)되었을 때)
          where_clause = where_clause || label(cid) || " AND ";
    }
  }
  from_clause = remove_last_comma(from_clause);
      where_clause = remove_last_and(where_clause);
      select_clause = remove_last_comma(select_clause);
      return select_clause || from_clause || where_clause;
}
```

<번역 알고리즘>

정해졌다고 가정하자. generate_OQL_query( )의 함수 내에서는 generate_labels_of_all_components(query)에 의해 질의의 구성 요소들에 대해 레이블들을 생성한다. 질의 번역 과정을 설명하기 위해 다음과 같이 레이블이 생성되었다고 가정하자.

Instructor, Department, v1, v1.salary > 30000,
v1.dept.dno = 312, v1.name, v1.dept.name

다음은 make_var_id_list(query) 함수를 통해 블랍 또는 서브블랍에 바인딩되는 시각변수들의 식별자 리스트 vid_list를 생성한다. 현재 사용 예제에서 vid_list는 하나의 시각변수 식별자로 구성된 리스트, 즉 <v1>이다. 실제 질의번역 과정은 generate_OQL_sub_query(vid_list)를 통해 이루어진다.

generate_OQL_sub_query(vid_list)는 우선 스트링 변수 select_clause, from_clause, where_clause들을 각각 "SELECT ", "FROM ", "WHERE "로 초기화 해 놓고, from절, where절, select절의 순서로 번역해 간다. 번역 과정은 vid_list에 속한 시각변수들을 중심으로 이루어지는데, vid_ list로부터 하나의 시각변수를 선택하여 이에 대해 from절, select절, where절을 순차적으로 생성한다. 선택된 시각변수에 대해 from절은 label ( ) 함수와 binding_blob_id( ) 함수를 이용하여 생성되는데, 위에서 초기화된 from_clause는 다음과 같이 변경된다.

from_clause = "from v1 in Instructor, "

다음은 위에서 선택된 시각변수에 대해 프로젝션 리스트를 생성한다. find_proj_id_list (vid) 함수는 프로젝션 요소의 id 리스트를 반환하는데 각 프로젝션 요소의 id에 대해 이의 레이블(v1.name)을 select절에 추가한다. v1 시각변수 식별자에 대해 다음과 같은 select절이 생성된다.

select_clause = "select v1.name, v1.dept.name,"

다음은 선택된 시각변수에 대해 where절을 생성한다. find_condition_id_list_related_to(vid) 함수는 vid에 관련된 모든 조건들의 id 리스트를 반환하는데 이 경우, 하나의 조건절 "v1.salary > 30000"의 id를 반환된다. 반환된 각 조건절의 타입이 상수 값과의 비교인 경우에는 즉시 where절에 삽입하고 다른 경로식과의 비교인 경우, 해당 경로식의 시각변수가 이미 방문되었는지를 확인한다. 이는 필요한 시각변수들이 from 절에 포함되어 있는지 여부를 확인하여 from절에 삽입된 시각변수들에 대한 조건식을 생성하는 것이다. 현재 다루는 예제의 경우에는 다음과 같은 where절이 생성된다.

where_clause = "where v1.salary > 30000 and"

다음 조건 또한 위와 같이 "Ei θ c"의 형태이므로 where절은

where_clause = where_clause || "v1.dept.dno
= 312 and"

로 표현할 수 있다.

위의 절차를 vid_list에 있는 모든 시각변수들에 대해 반복 적용하여 OQL 질의문을 만들어간다. 모든 시각변수에 대해 위의 내용을 적용한 후, from절의 마지막 쉼표(,), where절의 마지막 AND(and) 그리고 select절의 마지막 쉼표(,)를 제거하고 이들을 통합하여 다음과 같은 OQL 질의문을 완성한다.

```
select v1.name, v1.dept.name
from v1 in Instructor
where v1.salary > 30000 and v1.dept.dno = 312
```

다수의 시각변수들이 사용되는 질의 예제의 번역을 <그림 4-4>의 예를 통해 설명한다. VOQL* 질의문에서 시각변수의 id들을 오른쪽부터 v1, v2, v3라고 하고 make_var_id_list()에 의해서 생성된 시각변수 id 리스트가 <v1, v2, v3>라고 가정하자. 이때 generate_labels_of_all_components(query)에 의해서 각 요소의 label들이 다음과 같이 생성된다.

Department, Instructor, Course, v1,
v1.instructors, v2, v2.teaches, v3, v3.name,
v3.name = "OS", v1.name

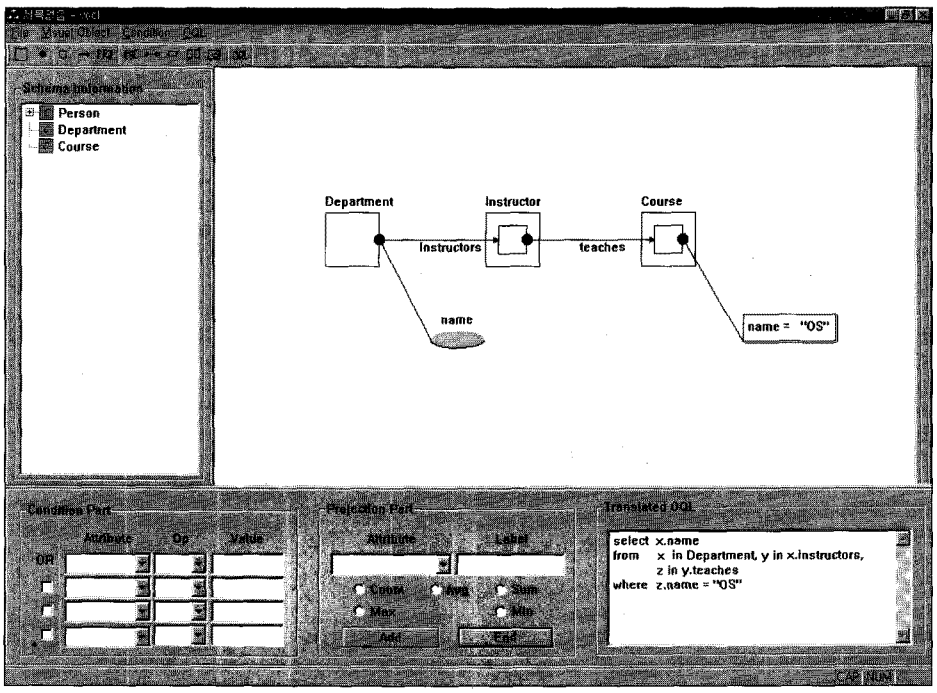
함수 generate_OQL_sub_query(<v1, v2, v3>)의 호출을 통해 OQL 질의문이 완성된다. generate_OQL_sub_query함수의 바디 안에서는 각 시각변수에 대해 select절, from절, where절을 생성해나가는데 그 과정을 다음에 간단히 설명한다. v1의 경우, from절, select절과 where절은 다음과 같이 구성된다.

from_clause = "from v1 in Department, "
select_clause = "select v1.name,"
where_clause = "where "

v1의 처리 후 시각변수 v2에 대해 from절, select절과 where절은 다음과 같이 갱신된다.

from_clause = "from v1 in Department,
v2 in v1.instructors,"
select_clause = "select v1.name,"
where_clause = "where "

그리고, 시각변수 v3의 경우, from절, select절과 where절은 다음과 같이 갱신된다.



<그림 4-4> 번역 알고리즘 예제

```
from_clause = "from v1 in Department,
              v2 in v1.instructors, v3 in
              v2.teaches, "
select_clause = "select v1.name,"
where_clause = "where v3.name = "OS""
```

마지막으로 from절, select절과 where절의 내용을 정리하고 통합하면 다음과 같은 OQL 질의문으로 번역된다.

```
select v1.name
from v1 in Department, v2 in v1.instructors,
     v3 in v2.teaches
where v3.name = "OS"
```

V. 결 론

본 연구에서는 객체지향 데이타베이스의 시각 질의어 VOQL*와 이를 OQL문으로 번역하여 주는 질의 처리 시스템(InQs)를 소개하였다. VOQL*는 객체 지향 데이타베이스를 위한 시각 질의어로 그래프와 벤 다이어그램에 기초한 문법을 통해 질의문이 표현된다. 즉, 데이타의 중첩된 구조는 그래프에 기초하여 표현되고 집합 관련 조건은 벤 다이어그램을 통하여 표현되므로 VOQL*로 표현된 질의는 다른 질의어에 비해 직관적이다. 한편, VOQL*는 OOPC라는 형식 질의어를 통해 시맨틱이 귀납적으로 정의되는 특징이 있다[10].

InQs에서는 제한된 화면의 적절한 사용을 위해 리터럴의 시각적 표현을 제한하는 수정된 VOQL*을 사용하여 화면 사용의 효율성을 높이며, 스키마 정보 윈도를 통해 사용자에게 현재 사용되는 데이타베이스의 스키마 정보를 제공한다. 사용자는 이를 참조하여 대화식 방법으로 다양한 메뉴와 버튼 그리고 윈도를 통해 질의문을 생성하고 이를 OQL로 변환한다. 이와 더불어 InQs는 다음같은 특징들을 갖는다. 첫째, 사용자는 기존 범용 질의어인 SQL과 같은 텍스트 기반 언어의 문법을 기억할 필요없이 쉽게 질의문을 표현할 수 있으며, 둘째, 그래프 기반의 VOQL*의 특성 때문에 중첩 질의문의 표현이 비 중첩 질의문과의 차이없이 자연스럽게 작성할 수 있으며 OQL의 번역시 프로젝션되는 대상의 타입에 따라 중첩 질의문이 자동 생성된다.

본 논문에서 제안한 VOQL*와 이를 구현한 InQs시스템은 아직 많은 개선점을 가지고 있다. 그러나 VOQL*는 형식 시맨틱에 기초한 언어로 로직을 그 근간으로 하고 있어 이론적 확장이 용이하며 그 의미가 명확하다. InQs 또한 보다 편리한 사용자 인터페이스의 개발이 요구된다. 이러한 개선 사항은 앞으로의 추후 연구 과제로 남겨 둔다.

〈참 고 문 헌〉

[1] Angelaccio, M., Catarci, T., and Santucci, G., "QBD*; A Graphical Query Language With Recursion," *IEEE Trans. on Software Engineering*, Vol. 16, No. 10, October 1990, pp. 1150-1163.

[2] Bertino, E., et al., "Object - Oriented Query Language: The Notion and the Issues," *IEEE Trans. on Knowledge and Data Engineering*, Vol. 1, No. 3, June 1992, pp. 223-237.

[3] Carey, M., Haas, L., Maganty, V., and Williams, J., "PESTO: An Integrated Que-

ry/Browser for Object Databases," in *Proc. Intl. Conf. on Very Large Data Bases*, 1996, pp. 203-214.

[4] Chavda, M., and Wood, P., "Towards an ODMG-Compliant Visual Object Query Language," In *Proc. Intl. Conf. on Very Large Data Bases*, Athens, Greece 1997, pp. 456-465.

[5] Cruz, I., Mendelzon, A., and Wood, P., "Graphical Query Language supporting Recursion," In *Proc. ACM SIGMOD Intl. Conf. on Management of Data*, 1987, pp. 323-330.

[6] Czejdo, B., Elmasri, R., and Rusinkiewicz., M., "A Graphical Data Manipulation Language for an Extended Entitiy-Relationship Model," *IEEE Computer*, Vol. 23, Mar. 1990, pp. 26-36.

[7] Frohn, J., Lausen, G., and Uphoff, H., "Access to Objects by Path Expressions and Rules," In *Proc. the 20th VLDB Conference*, 1994, pp. 273-294.

[8] Kifer, M., Kim, W., and Sagiv, Y., "Querying Object-Oriented Databases," In *Proc. ACM SIGMOD Intl. Conf. on Management of Data*, San Diego, CA, 1992, pp. 393-402.

[9] Jeonghee, Kim, Tae-sook, Han, Suk Kyoon, Lee, "VOQL: A Visual Object-Oriented Database Query Language For Visualizing Expressions," accepted for publication in International Journal of Computer Systems Science and Engineering, 1998.

[10] 이석균, "VOQL*: 귀납적으로 정의된 형식 시맨틱을 지닌 시각 객체 질의어," 한국정보과학회 논문지: 데이터베이스, 27권 2호, pp. 151-164, 2000.

[11] Zloof, M., "Query By Example," *IBM Systems Journal*, Vol. 16, 1977, pp. 324-343.

[12] Cattell, R.G., editor, The Object Database Standard: ODMG 2.0 Morgan Kaufmann Publishers, San Francisco 1997.

[13] Gyssens, M., Paredaens, J., Van den Bussche, J., and Van Gucht, D., "A Graph-Oriented Object Database Model," *IEEE Trans. on Knowledge and Data Engineering*, Vol. 6, No. 4, August 1994, pp. 572-586.

[14] Harel, D., "On Visual Formalisms," In *Comm. of the ACM*, Vol. 31, No. 5, 1988, pp. 514-530.

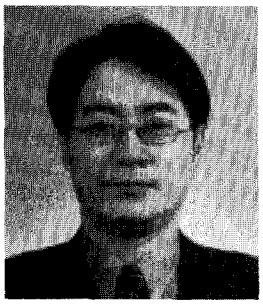
[15] Mohan, L. and Kashyap, R.L., "A Visual Query Language for Graphical Interaction With Schema-Intensive Databases," *IEEE Trans. on Knowledge and Data Engineering*, Vol. 5, No. 5, 1993, pp. 843-858.

[16] Vadaparty, K., Aslandogan, Y.A., and Ozsoyoglu, G., "Towards a Unified Visual Database Access," In *Proc. ACM SIGMOD Intl. Conf. on Management of Data*, 1993, pp. 357-366.

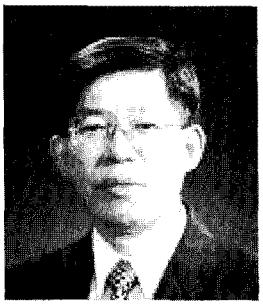
◈ 이 논문은 2001년 3월 7일 접수하여 1차 수정을 거쳐 2001년 5월 10일 게재확정되었습니다.

◆ 저자소개 ◆

이석균 (Lee, Suk Kyoon)
서울대학교 경제학과에서 학사, University of Iowa에서 전산학 석사 및 박사를 취득하였다. IEEE 8th International Conference on Data Engineering에서 최우수 논문상을 수상하였으며 세종대학교 정보처리학과에서 전임 강사로 근무하였다. 현재 단국대학교 자연과학부 부교수이다. 주요 관심 분야는 데이타베이스에서 불완전 정보관리, 데이타베이스 시각 질의어 설계, 다중처리기에서의 실시간 스케줄링, 데이터웨어하우스, 데이터 마이닝, XML 등이다.



나연묵 (Nah, Yunmook)
서울대학교 컴퓨터공학과에서 학사, 석사, 박사학위를 취득하였다. 현재 단국대학교 컴퓨터공학과에 부교수로 재직중이다. 또한 한국정보과학회 논문지 편집위원과 데이타베이스연구회 운영위원, 서울특별시 정보화추진위원회 위원 등으로 활동중이다. 주요 관심분야는 데이타베이스, 객체지향 데이타베이스, 멀티미디어 데이터베이스 등이다.



서용무 (Suh, Yongmoo)
서울대학교 사범대학 수학과, 한국과학원 전산학과를 졸업하고, 한국과학기술 연구소 전산센터에서 연구원으로 재직시 도미하여, University of Texas (at Austin)에서 전산학석사, 경영정보학박사를 취득한 후, 세종대학교, 건국대학교를 거쳐, 현재 고려대학교 경영대학에 재직하고 있다. 주요 관심분야는 web-based organizational computing, database, data warehouse, data mining, XML 등이다.