그래픽 객체 질의어에서 집합 속성과 메소드를 포함한 경로식의 시각화

(Visualization of Path Expressions with Set Attributes and Methods in Graphical Object Query Languages)

조 완 섭 †

(Wan-Sup Cho)

요 약 대부분의 상용 관계 DBMS(Database Management System)에서는 사용자 편의성을 위하여 SQL과 함께 그래픽 질의어를 제공하고 있으나, 객체 DBMS의 경우에는 그래픽 질의어에 관한 연구와 개발이 미흡한 실정이다. 그래픽 질의어에서는 복잡한 질의 조건을 간결하고도 직관적인 방법으로 표현하는 것이 중요한 이슈이다. 특히, 객체 DBMS는 관계 DBMS 보다 복잡한 데이타 모델과 객체 질의어를 제공하므로 그래픽 객체 질의어를 설계하고 구현할 때 간결성과 직관성을 유지하는 것이 더욱 중요하다. 본 논문에서는 인터넷 환경에서 원격지 객체 데이타베이스에 접근하여 자료를 검색하고 관리하는 그래픽 객체 질의어인 GOQL(Graphical Object Query Language)을 제안한다. GOQL은 그래픽 관계 질의어에서 다루지 않은 집합값 속성과 한정어 및 메소드를 포함한 길이가 2 이상인 경로를 간단한 그래픽 요소들로 시각화함으로써 간결성과 직관성을 높인다. 그리고, 대표적인 객체 질의어인 XSQL[1,2]에서 사용하는 경로는 GOQL에서 제공하는 간단한 시각적 도구로 표현할 수 있음을 보인다. 또한, 논문에서는 GOQL의 그래픽 질의어를 텍스트 객체 질의어로 변환하는 알고리즘을 제안하고, 실제로 인터넷 환경에서 동작하도록 구현한 결과를 소개한다.

키워드 : 객체 데이타베이스, 객체 질의어, 그래픽 질의어, 그래픽 객체 질의어

Abstract Although most commercial relational DBMSs provide a graphical query language for the user friendly interfaces of the databases, few research has been done for graphical query languages in object databases. Expressing complex query conditions in a concise and intuitive way has been an important issue in the design of graphical query languages. Since the object data model and object query languages are more complex than those of the relational ones, the graphical object query language should have a concise and intuitive representation method. We propose a graphical object query language called GOQL (Graphical Object Query Language) for object databases. By employing simple graphical notations, advanced features of the object queries such as path expressions including set attributes, quantifiers, and/or methods can be represented in a simple graphical notation. GOQL has an excellent expressive power compared with previous graphical object query languages. We show that path expressions in XSQL[1,2] can be represented by the simple graphical notations in GOQL. We also propose an algorithm that translates a graphical query in GOQL into the textual object query with the same semantics. We finally describe implementation results of GOQL in the Internet environments.

Key words : Object Databases, Object Query Language, Graphical Object Query Language

1. 서 론

SQL과 같은 텍스트 기반 질의어에 비하여 그래픽 질의어 (graphic query language)는 사용자에게 편리한 데이타베이스 인터페이스를 제공하며, 질의 작성시 오류 발생 빈도를 줄여준다[3,4,5,6]. 텍스트 기반 질의어의 사용자는 질의를 할 때 질의어 구문 뿐 아니라 데이타베이스 스키마 정보와 객체간의 복잡한 관련성을 기억

· 본 연구는 첨단정보기술연구센터 과제 "인터넷 환경에서 객체-관계 DBMS를 위한 그래픽 질의어"를 통하여 과학재단의 지원을 받았음.

† 정 회 원 : 충북대학교 경영정보학과 교수
wscho@chungbuk.ac.kr

논문접수 : 2002년 8월 8일
심사완료 : 2003년 2월 7일

해야 하지만, 그래픽 질의어의 사용자는 이러한 정보를 기억할 필요가 없기 때문이다. 일반적으로 그래픽 질의어의 사용자는 다이어그램 형태로 표시된 데이타베이스 스키마 정보를 보면서 질의에 필요한 시각적 요소들을 선택한 후 간단한 형태의 질의 조건만을 부과하여 질의를 작성하게 된다.

그래픽 질의어에 관한 연구는 지금까지 관계 데이타베이스와 개체-관계(entity-relationship) 데이타베이스에 집중되어 왔으나[7,8,3,5,9], 최근들어 객체 데이타베이스를 위한 그래픽 질의어의 연구가 활발하게 이루어지고 있다[10,11,12,13,3,4,14,2,6]. 그러나, 기존의 그래픽 객체 질의어들은 그 표현력이 간단한 질의를 작성하는 수준으로 한정되어 있으므로 객체 모델의 다양한 특성을 모델링하는데 충분하지 못하다. 예를들어, 기존의 그래픽 객체 질의어에서는 조건을 표시하는데 중요한 요소인 경로 (path expression)[1]를 표현하는데 한계가 있다. 특히, 객체 질의어에서는 집합 속성이나 메소드 혹은 한정어 (for all, exist)가 포함된 경로가 널리 사용되고 있으나 대부분의 그래픽 객체 질의어에서는 이들이 포함된 경로를 다루지 않고 있다. 실제로 21SQL을 비롯한 대부분의 상용 객체 데이타베이스 시스템에서 제공하는 그래픽 질의어는 하나의 클래스에 대하여 간단한 조건을 제시하는 정도이다[2]. 참고문헌[12,3] 등에서도 그래프를 사용한 객체 질의어를 제안하였지만 경로나 집합 속성 및 한정어 등은 다루지 않고 있다. 참고문헌 [10,4]에서는 시각적 관계 질의어인 QBE(Query By Example)[15]를 확장한 아이콘 기반(icon-based)의 시각적 객체 질의어를 제안하였으며, 특히 전체 한정어(universal quantifier)를 시각적으로 표현할 수 있도록 하였다. 그러나 여기서는 그래프 형태의 객체 스키마를 보면서 그래프와는 다른 시각적 도구인 아이콘을 사용하여 질의를 표현하므로 사용자 입장에서 두 가지 서로 다른 시각적 도구(그래프와 아이콘)를 사용하게 되어 복잡성이 증대된다. 참고문헌 [13,14]에서는 스키마 그래프와는 다른 도구인 벤다이어그램을 사용하여 경로 사이의 관계를 명시하고 있다. 그러나, 여기서도 참고문헌 [10,4]과 마찬가지로 두가지 서로 다른 시각적 도구로 인한 복잡성이 존재하며, 집합 속성에 관한 시각적 표현은 구체적으로 다루지 않고 있다.

본 논문에서는 객체 데이타베이스에서 GOQL (Graphic Object Query Language)이라는 새로운 그래픽 질의어를 제안한다. GOQL 사용자는 그래프 형태의 객체 데이타베이스 스키마로부터 질의에 필요한 클래스(들)와 속성(들)을 선택하여 간단한 질의 조건을 추가하는 방식으로 질의를 작성한다. GOQL에서는 질의어 자체가 스키마 그래프와 동일한 그래프 형태이므로 사용자는 한가지 시각적 도구로 스키마와 질의를 모두 인식할 수 있다. 객체 질의에서 집합 속성을 포함하는 경우 집합 전체에 대한 조건과 원소에 대한 조건을 구분하여 연산자를 선택해야 하므로 텍스트 질의어 조차도 복잡성을 가진다[1,16]. 특히, 길이가 2 이상인 경로에서 다수의 집합 속성이나 한정어가 포함된 경우에 질의 작성의 복잡성은 더욱 증가한다. GOQL에서는 이러한 경우에도 간단한 그래픽 요소를 사용하여 복잡한 질의를 시각적으로 표현하는 방안을 제시한다. 또한, 경로에 속성 뿐 아니라 메소드가 포함된 경우에도 이를 간단히 시각화하고, 관계 질의에서 널리 사용하는 명시적 조인 (explicit join)도 간단한 그래픽 요소들로 표현할 수 있도록 한다. 논문에서는 제안된 GOQL의 표현력(expressive power)을 분석하기 위하여 객체 질의어인 XSQL[1,2]과 비교한다. 사용자가 작성한 GOQL의 그래픽 질의는 실제로 특정 객체 DBMS의 텍스트 질의어로 변환되어 처리된다. 논문에서는 GOQL을 텍스트 질의로 변환하는 알고리즘을 제시한다.

제안된 GOQL은 인터넷 환경에서 객체 DBMS의 클라이언트 시스템으로 개발되어 객체 데이타베이스의 검색과 관리에 이용되고 있다. GOQL 클라이언트 시스템은 인터넷 환경에서 쉽게 동작하도록 하기 위하여 JAVA와 JDBC를 이용하여 구현하였으며, 객체 DBMS인 Tachyon[17]과 Odysseus[18]의 사용자 인터페이스로 사용되고 있다. GOQL 사용자는 인터넷으로 연결된 원격지 객체 데이타베이스를 간단한 그래픽 요소들을 사용하여 연결한 후, 데이타베이스를 검색하고 관리할 수 있다. 사용자는 연결된 원격지 객체 데이타베이스의 카탈로그 정보를 대화식으로 제공받으면서 그래픽 질의를 작성하고, 객체를 삽입/삭제/변경할 수 있다. 작성된 그래픽 질의어는 동일한 의미의 텍스트 질의어로 변환된 후, 원격지 데이타베이스 서버로 전송되어 처리된다.

논문의 구조는 다음과 같다. 제 2 장에서는 관련 연구로써 객체 데이타베이스와 객체 질의어에 관한 기존 결과를 요약하여 소개한다. 제 3 장에서는 그래픽 객체 질의어인 GOQL을 제안하고, GOQL의 표현력을 XSQL[1,2]과 비교 분석한다. 그리고, GOQL의 그래픽 질의를 동일한 의미의 텍스트 질의로 변환하는 알고리즘을 제안한다. 제 4 장에서는 GOQL의 구현 결과를 소개하고, 제 5 장에서 결론을 맺는다.

2. 관련 연구

본 장에서는 GOQL의 특징을 기술하는데 필요한 객체 데이타 모델을 설명하고, 이를 지원하는 텍스트 기반 객체 질의어의 특징들을 기술한다. 그리고, 기존의 그래픽 객체 질의어에 관하여 검토 분석한다.

2.1 객체 데이타 모델과 텍스트 객체 질의어

그림 1은 본 논문에서 예제로 사용하는 객체 데이타베이스 스키마이다. 그림에서 사각형 노드는 사용자 정의 클래스를 의미하며, 원형 노드는 시스템 정의 클래스 (예를들어, Integer나 String 등의 타입)를 나타낸다. 링크는 속성이 정의된 클래스 노드에서 속성의 도메인이 되는 클래스를 연결하는 속성-도메인 관계를 나타낸다. 속성은 사용자 정의 클래스를 도메인으로 하는 복합 속성(composit attribute)과 시스템 정의 클래스를 도메인으로 하는 단순 속성(simple attribute)으로 나누어진다[19]. 속성의 이름 뒤에 * 표시는 그 속성이 집합값을 가짐을 의미한다.

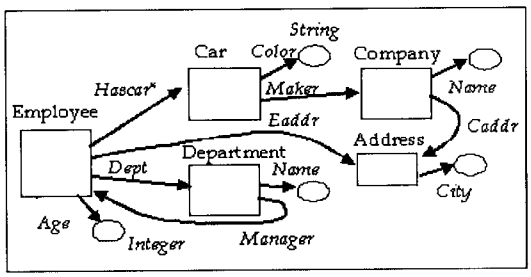


그림 1 객체 데이타베이스 스키마

객체 스키마는 다음과 같이 그래프로 정의되며, 이를 스키마 그래프(SG: Schema Graph)라고 한다. 스키마 그래프 SG는 다음과 같은 노드와 링크의 집합으로 각 노드는 클래스 이름을 레이블로 가지며, 속성-도메인 관계를 나타내는 링크는 속성 이름을 레이블로 가진다.

SG = (N, N', E, E')

- N : 사용자 정의 클래스를 나타내는 사각형 노드들의 집합
- N' : 시스템 정의 클래스 (타입)를 나타내는 원형 노드들의 집합
- E : 속성-도메인 관계를 나타내는 레이블을 가진 실선 링크 집합
- E' : 클래스-서브 클래스 관계를 나타내는 점선 링크 집합

객체 질의어에서 가장 큰 특징은 경로를 사용한다는 점이다[1]. *경로(path expression)*는 스키마 그래프에서 속성-도메인 관계(링크)를 따라서 클래스와 속성들을 나열한 것으로 객체 질의어에서 조건을 표시하는 중요한 요소이다. 예를들어, 그림 1에서 *Employee.Age*는 "직원의 나이"를 의미하는 경로이고, *Employee.Hascar.Maker.Name*은 "직원이 가진 자동차들의 제조회사 이름"을 의미하는 경로이다. 경로에서 속성 다음에 괄호[ ]를 사용하여 그 속성의 도메인이 되는 클래스를 표시할 수도 있으나[1], 본 논문에서는 혼란이 없는 한 클래스 이름을 생략한다. 예를들어, 경로 *Employee.Hascar[Car].Color[String]*은 간단히 *Employee.Hascar.Color*로 표시한다. *경로의 길이(length)*는 경로에 포함된 속성의 개수로 정의한다. 예를들어, *Employee.Hascar.Color*는 길이가 2이다. 경로는 *집합 경로*와 *스칼라 경로*로 구분된다. 집합 경로는 집합 속성을 포함한 경로이며, 계산 결과가 집합값이다. 스칼라 경로는 단일값 속성들만으로 구성된 경로이며, 계산 결과는 스칼라 값이다. 예를들어, *Employee.Hascar.Color*는 속성 Hascar가 집합 속성이므로 집합 경로이며, *Employee.Dept.Name*은 단일값 속성만 포함하므로 스칼라 경로이다. 관계 질의어에서는 길이가 1인 경로만을 사용하지만, 객체 질의어에서는 길이가 2 이상인 경로가 사용될 수 있다.

질의 1은 그림 1의 객체 데이타베이스에 대하여 XSQL[1]로 작성한 객체 질의이다. XSQL에서는 질의 1과 같이 집합 경로(e.Hascar.Color)와 스칼라 값("Red")을 비교하기 위하여 한정어(all, exist)가 부착된 스칼라 연산자 (all= 혹은 exist=)를 사용한다.

(질의 1) Employee 중에서 나이가 40세 이상이며 빨간색 자동차 만을 가진 사람은?

```
SELECT    e.Name
FROM      Employee e
WHERE     e.Age >= 40
AND       e.Hascar.Color all= "Red" ;
```

WHERE 절에서 *e.Hascar.Color*는 직원 e가 가진 자동차들의 색깔의 집합으로 계산되는 집합 경로이다. 질의 조건의 의미는 직원 e의 나이가 40세 이상이고, 그가 소유한 모든 자동차의 Color가 Red이면 참이 된다.

2.2 그래픽 객체 질의어

최근 객체 DBMS가 널리 보급되면서 객체 데이타

1) XSQL[1]에서는 이를 셀렉터(selector)라고 하며, 도메인을 지칭하거나 한정하는 용도로 사용함.

베이스를 위한 그래픽 질의어에 대한 연구가 활발해 지고 있다[10,11,13,4,2,6]. [10]에서는 아이콘 기반의 그래픽 객체 질의어인 VISUAL을 제안하고 있다. VISUAL은 과학 데이타베이스(scientific database)를 위한 그래픽 사용자 인터페이스를 주목표로 하고 있으며, 아이콘 기반 그래픽 질의어와 그에 대한 질의 처리 기법을 제안하고 있다. 특히, VISUAL을 ODMG OQL[16]과 객체 대수로 변환하는 알고리즘을 제안함으로써 다양한 객체 DBMS 플랫폼에서 VISUAL이 사용될 수 있도록 하였다. 그러나, VISUAL은 아이콘 기반 그래픽 질의어로써 본 논문의 그래프 기반의 질의어와는 접근 방식이 다르다. 일반적으로 그래픽 질의어는 테이블 폼을 이용하는 폼-기반, 아이콘을 이용하는 아이콘-기반, 그래프 형태의 그래프-기반으로 구분한다. 객체 데이타베이스의 경우 스키마 자체가 자연스럽게 그래프 형태로 표현되므로 본 연구에서는 그래프 기반의 그래픽 질의어를 제안한다. 사용자가 그래프라는 통일된 하나의 시각적 도구로 스키마와 질의어를 작성하는 것이 사용자 인터페이스 측면에서 더욱 단순하다는 판단에서이다. [11]에서도 ODMG의 OQL을 위한 그래픽 질의어를 제안하였으나 다양한 형태의 아이콘을 사용한다는 점과 전체 한정자(universal quantifier)를 제외한다는 점에서 제안된 기법과 차이가 있다. [13]에서는 벤다이어그램 이론을 기반으로 하여 객체 데이타베이스를 위한 그래픽 질의어를 제안하고 있으나 실제 객체 DBMS를 이용하여 구현하지는 않았다. 벤다이어그램의 원리는 수학적으로 명확하기는 하지만 실제 구현하는데 어려움이 있기 때문이다. 본 논문에서는 [13]에서 제안한 벤다이어그램 개념을 시각화하는 대신에 집합 연산자로 단순화하여 구현하고 있다. [2]에서는 UniSQL 객체 DBMS를 위한 그래픽 사용자 인터페이스를 제공하고 있다. 여기서는 하나의 클래스에 대한 단순 질의를 그래픽 형태로 작성할 수 있으며, 길이가 2 이상인 경로식에 대한 그래픽 질의 표현은 지원하고 있지 않다. [6]에서는 관계 데이타베이스, 중첩-관계 데이타베이스, 객체지향 데이타베이스를 위한 하나의 통일된 (unified) 그래픽 질의어를 제안하고 있다. 여기서는 테이블 폼을 사용하는 QBE (Query-By-Example) 형태를 취하고 있으며, 이를 확장하여 중첩-관계 모델과 객체지향 모델의 데이타베이스에 대한 그래픽 질의어로 사용될 수 있도록 하였다. [6]의 경우도 아이콘 기반의 그래픽 질의와 마찬가지로 그래프 형태의 객체 데이타베이스 스키마와는 다른 유형의 시각적 도구인 폼을 사용하여 질의를 한다는 점에세 제안된 그래픽 질의와는 차이가 있다.

3. 그래픽 객체 질의어 GOQL의 설계

GOQL은 객체 데이타베이스에 관한 그래픽 인터페이스로서 스키마 관리, 질의 검색, 데이타 변경, 데이타베이스 로더/언로드 등과 같은 유용한 기능을 제공한다. 본 논문에서는 GOQL의 핵심 기능인 그래픽 질의어에 관하여 설명한다.

그래픽 객체 질의어를 설계할 때 가장 중요한 사항은 "경로에 대한 조건을 어떻게 간단한 그래픽 요소로 표현할 것인가?" 라는 문제이다. 본 장에서는 길이가 1인 경로부터 길이가 2 이상인 경로에 대한 조건을 그래픽 요소로 간단히 표시하는 방법을 차례로 제안한다. 특히, 집합 속성(들)이 포함된 경로를 그래픽 요소로 표현하는 경우 복잡한 문제가 발생함을 지적하고, 이를 해결하는 방안을 제시한다.

3.1 GOQL 정의

GOQL은 스키마 그래프 SG로부터 노드와 링크를 선택하여 생성한 서브 그래프에 조건 레이블을 추가한 것으로 다음과 같이 정의된다.

GOQL = (N, E, C)

- N : 클래스를 나타내는 노드들의 집합
- E : 속성-도메인 관계를 나타내는 링크들의 집합
- C : "*속성 θ Value*" 혹은 "*속성 θ 속성*" 형태의 조건 레이블의 집합

단순 속성 (도메인이 시스템 정의 클래스인 속성)을 나타내는 원형 노드와 상속 관계를 나타내는 점선 링크는 스키마 그래프에 포함되지만 GOQL에서는 복잡성을 줄이기 위하여 제외한다. GOQL에서 단순 속성까지 그래픽 요소인 노드로 표시하면 그래픽 질의 자체가 복잡해지므로 텍스트 형태의 속성 이름을 사용하도록 한다. 이 경우에도 GOQL 사용자는 속성과 타입 정보를 텍스트 형태로 제공받을 수 있으므로 질의 작성 과정에서 텍스트 속성 이름을 사용하는데 따른 어려움은 없을 것이다. 다음으로, 상속 관계를 나타내는 점선 링크를 GOQL에서 제외한 이유는 사용자가 질의의 대상이 되는 클래스를 일단 선택하면 계층 구조 전체를 GOQL에 표시할 필요가 없이 선택된 노드만 표시하면 되기 때문이다. GOQL에서는 상위 클래스가 선택되면 자동으로 그의 서브 클래스들도 질의 대상으로 간주하고, 그래픽 질의에서는 상위 클래스에 대한 노드만 표시하게 된다.

다음에는 GOQL에 대하여 단순한 질의에서 복잡한 질의까지 차례로 설명한다. 중요한 사항은 사용자가 생성한 GOQL의 의미를 규정하는 것이며, 본 논문에서는 각 GOQL에 대하여 그 의미를 텍스트 객체 질의어를 사용하여 기술한다.

3.2 단일 경로에 대한 조건

여기서는 단일 경로(single path)에 대한 조건(들)을 GOQL에서 어떻게 표현하는지 살펴본다. 단일 경로라 함은 스키마 그래프 상에서 들어오고/나가는 차수(in/out degree)가 1인 노드와 링크로 구성된 경로를 의미한다. 이 절에서는 편의상 길이가 1인 단순 조건에 대한 GOQL 표현을 살펴본 다음에, 이를 길이가 2 이상인 경로에 대한 조건의 표현으로 확장한다. 그리고, 경로에 집합 속성과 메소드가 포함된 경우도 살펴본다.

3.2.1 길이가 1인 경로의 표현

길이가 1인 경로에 대한 조건은 하나의 클래스에 대한 조건으로 *단순 조건(simple predicate)*이라 부른다. 예를들어, *Employee.name = "Smith"*는 단순 조건이다. 단순 조건은 관계 질의에서도 널리 사용되고 있다. GOQL에서 단순 조건은 클래스를 나타내는 클래스 노드에 "속성 θ Value" 형태의 조건 레이블을 첨가하는 형태로 표시한다. 그림 2는 조건 *C1.A1 ∋ 10* 에 대한 그래픽 질의어이다. 여기서 관계 질의와 다른 점은 집합 속성과 메소드에 관한 조건의 작성이다. 집합 속성의 경우 그림 2와 같이 조건 레이블을 작성할 때 집합 연산자를 사용하면 된다. 메소드의 경우는 제 3.2.4 절에서 별도로 다룬다.

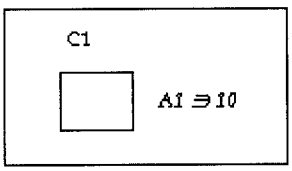


그림 2 길이가 1인 경로에 대한 조건의 GOQL 표현

3.2.2 길이가 2인 경로의 표현

길이가 2인 경로에 대한 조건은 속성-도메인 관계에 있는 두 클래스에 대한 조건이다. 예를들어, 그림 1에서 조건 *Employee.Dept.Name = "R&D"*는 속성-도메인 관계에 있는 두 클래스 *Employee*와 *Department*에 대한 조건으로 길이가 2인 경로에 대한 조건이다. 이 경우에 GOQL은 경로에 포함된 두 클래스를 노드로 표시하고, 이들을 속성 링크로 연결한 후, 노드에 조건을 부과하는 방식으로 질의를 표현한다. 그림 3은 경로에 대한 조건 *C1.A1.A2=20*을 그래픽 질의로 표시한 것이다. 그림에서 속성 A1과 A2는 단일값 속성으로 가정하였다.

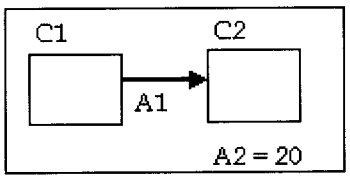


그림 3 길이가 2인 경로에 대한 조건의 GOQL 표현

그림 3에서 경로의 마지막 속성[2]인 A2가 집합 속성이라면 마지막 노드의 조건 레이블에서 A2 ∋ 20과 같이 집합 연산자를 사용하면 된다. 그러나, 복합 속성[3]인 A1이 집합값을 가진다면 GOQL 설계에서 주의가 필요하다. 그림 3에서 A1이 집합 속성이면 A2의 집합 속성 여부와 무관하게 경로 C1.A1.A2를 계산하면 집합값이 리턴되므로 A2에 대한 연산자의 선택을 주의해야 한다. GOQL에서는 경로 C1.A1.A2의 계산 결과로 리턴되는 집합에 대하여 원소 조건 혹은 집합 조건 두가지 중 하나를 선택하여 조건을 부과할 수 있도록 하였다. 다음에서 원소 조건과 집합 조건을 자세히 살펴본다.

(1) 원소 조건

원소 조건은 집합 속성의 원소 하나 하나에 대하여 스칼라 조건을 부과하는 조건이다. 예를들어, 그림 1의 예제 스키마에 대하여 질의 2를 생각하자.

(질의 2) Select e.Name, c.Maker.Name
From Employee e, e.Hascar c
Where c.Color = "Red";

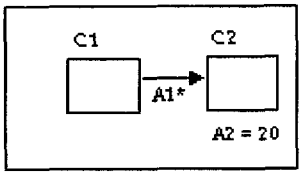
그림 1의 스키마에서 경로 Employee.Hascar.Color의 계산 결과는 직원이 가진 자동차들의 색깔의 집합이다. 이 집합 경로에 대한 원소 조건이란 색깔 집합의 각 원소에 대하여 조건을 부과한다는 의미이다. 질의 2에서는 객체 변수 c를 사용하여 경로 Employee.Hascar에 대한 각 원소를 지칭하고, Where 절에서 각 원소에 대한 조건을 표시하고 있다. 즉, 직원 e가 가진 각 자동차들을 객체 변수 c와 바인딩시킨 후, c의 색깔을 Where 절에서 검사하고 있다. 여기서 유의

2) 이는 조건이 부과되는 속성으로 대개 정수나 실수 등과 같은 프리미티브 타입을 의미함.

3) 도메인이 사용자 정의 클래스인 속성을 의미함[19].

할 점은 텍스트 객체 질의어에서 집합 경로에 대한 원소 조건은 객체 변수를 사용하여 From 절과 Where 절에 분할되어 작성된다는 점이다. 질의 2에서는 경로 Employee.Hascar.Color에 대한 원소 조건은 From 절과 Where 절에서 각각 e.Hascar c와 c.Color="Red"로 분할되어 나타나고 있다.

객체 질의에 대한 그래픽 질의어에서도 집합 경로에 대한 원소 조건을 간단한 그래픽 도구로 표현하는 것이 중요한 이슈이다. 집합 경로에 대한 원소 조건은 텍스트 객체 질의어에서 널리 사용됨에도 불구하고 기존의 그래픽 질의에서 다루지 않은 문제이기 때문이다. GOQL에서는 집합 경로에 대한 원소 조건을 "집합 경로식 임에도 불구하고 사용자가 스칼라 연산자를 선택하여 조건을 명시하는 방식"으로 간단히 표시한다. 이 방식은 사용자에게 객체 변수라는 복잡한 개념을 숨기면서도 집합 경로에 대한 원소 조건을 그래픽 형태로 간단히 명시할 수 있게 하므로 GOQL의 단순화 측면에서 바람직한 아이디어이다. 예를들어, 그림 4의 그래픽 질의에서 C1.A1.A2가 집합 경로이지만 조건 레이블을 작성할 때 사용자가 스칼라 연산자(=)를 선택하였으므로 이는 집합 경로에 대한 원소 조건으로 해석된다. 그래서 GOQL은 대응하는 텍스트 질의를 생성할 때 그림 4(b)와 같이 집합 속성 A1에 대하여 객체 변수 t2를 도입하여 집합 경로 C1.A1.A2에 대한 조건을 C1.A1 t2와 t2.A2 = 20으로 분할하고, 이를 Form 절과 Where 절에 나누어 작성한다. 그림 4(b)에서 프로젝션 리스트는 논문의 주요 사항이 아니므로 임의로 나열하였으며, 제 4 장에서 구현 방법을 구체적으로 설명한다.



(a) 원소 조건을 포함한 그래픽 질의

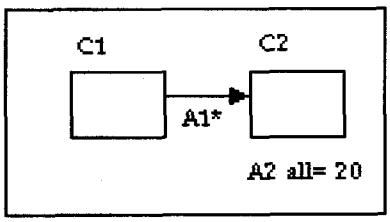
```
Select      c.A3, t2.A4
From        C1 c, c.A1 t2
Where       t2.A2 = 20 ;
```

(b) 대응하는 텍스트 질의

그림 4 집합 경로에 대한 원소 조건의 GOQL 표현

(2) 집합 조건

집합 조건은 집합 경로의 계산 결과로 리턴되는 집합값에 대한 조건이다. 사용자는 집합 경로에 대하여 집합 연산자(∈, ⊂ 등)나 한정어를 가진 스칼라 연산자(all=, all>, exits=, exist> 등)를 선택함으로써 집합 조건을 표현한다. 한정어를 가진 스칼라 연산자는 참고문헌 [1]에서 소개되었으며, 집합값과 스칼라 값을 서로 비교하는데 사용된다. 제 2 장의 질의 1에서 한정어를 가진 스칼라 연산자 (all=)의 사용 예를 살펴보았다. 그림 5(a)는 속성 A1이 집합값을 가지는 경우를 가정하고, 집합 경로 C1.A1.A2에 대하여 집합 조건을 부과한 예를 보여주고 있다. 사용자는 한정어를 가진 스칼라 연산자 (all=)를 사용하여 집합 경로 C1.A1.A2와 스칼라 값 20을 서로 비교하고 있다. 여기서 사용자는 all= 대신에 집합 연산자 (예를들어, ∋)를 사용할 수도 있다. 그림 5(b)는 대응하는 텍스트 질의어이다.



(a) 집합 조건을 가진 그래픽 질의

```
Select      c.A4
From        C1 c
Where       c.A1.A2 all=20;
```

(b) 대응하는 텍스트 질의

그림 5 집합 경로에 대한 집합 조건의 GOQL 표현

3.2.3 길이가 3 이상인 경로의 표현

길이가 i (>=3)인 경로에 대한 조건은 길이가 i-1인 경로에 대한 조건으로부터 쉽게 확장된다. 길이가 i 인 경로는 길이가 i-1인 경로에서 노드가 하나 더 추가된 형태이다. 그림 6은 길이가 i인 경로 C1.A1.A2....Ai에 대한 조건을 그래픽 질의로 보여주고 있다. 그림에서 길이가 j (=1,2,...,i)인 경로 C1.A1...Aj를 간단히 Pj라고 표시한다.

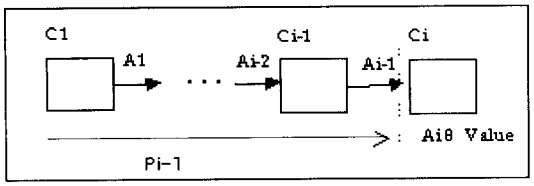
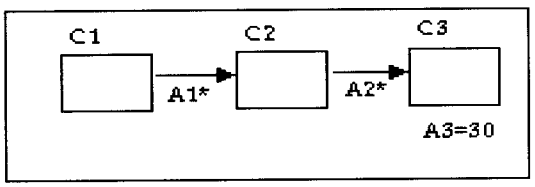


그림 6 길이가 i인 경로에 대한 조건의 GOQL 표현

경로 Pi에 대한 조건 (즉, 클래스 Ci의 속성에 대한 조건)은 경로 Pi-1에 대한 조건 (즉, 클래스 Ci-1의 속성에 대한 조건)의 그래픽 표현으로부터 다음과 같이 확장된다. 먼저, 경로 Pi-1이 스칼라 경로이면 조건 레이블 Ai θ Value를 작성할 때 Ai의 집합 속성 여부에 따라서 스칼라 연산자나 집합 연산자를 사용한다. 다음으로, Pi-1이 집합 경로이면[4] Ai의 집합 속성 여부와 무관하게 전체 경로 C1.A1....Ai는 집합값을 리턴한다. 따라서 사용자는 경로 C1.A1....Ai에 대하여 원소 조건이나 집합 조건을 부과할 수 있다. 즉, 원소 조건인 경우 조건 레이블 Ai θ Value에서 스칼라 연산자를 선택하면 되고, 집합 조건인 경우 집합 연산자를 사용하거나 한정어를 가진 스칼라 연산자를 선택하면 된다.

그림 7은 길이가 3인 경로에 대한 조건을 보여주고 있다. 여기서 속성 A1과 A2가 집합값을 가진다고 가정하였다. 왼쪽 GOQL에서는 집합 경로 C1.A1.A2.A3에 대하여 사용자가 C3에서 스칼라 연산자 =를 선택하였으므로 원소 조건을 표현한 것이다. 오른쪽 GOQL에서는 집합 경로 C1.A1.A2.A3에 대하여 사용자가 한정어를 가진 스칼라 연산자 (all=)를 선택하였으므로 집합 조건을 표현한 것이다. 그림 7의 좌측 그래픽 질의는 GOQL에 의하여 원소 조건으로 해석하여 집합 속성 A1과 A2에 대하여 객체 변수 t2와 t3을 도입하여 텍스트 객체 질의로 변환한다 (질의 3 참고). 반면에, 그림 7의 우측 그래픽 질의는 집합 조건 (한정어를 가진 스칼라 연산자)이므로 객체 변수를 도입하지 않고 질의 4와 같이 텍스트 질의를 변환된다.



4) 이 경우는 속성 A1,...,Ai-1 중에서 하나 이상이 집합 속성인 경우를 의미함.

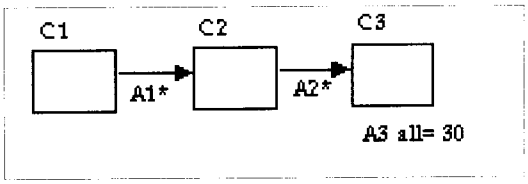


그림 7 길이가 3인 경로에 대한 조건의 표현

3.2.4 메소드를 포함한 그래픽 질의

그래픽 질의어 수준에서 볼 때 메소드는 인자(들)를 가질 수 있다는 점을 제외하고는 속성과 동일하게 취급될 수 있다[1]. 예를들어, *Employee*에 정의된 메소드 *Bonus(year, month)*는 년과 월을 인자로 입력하면 특정 직원의 보너스(정수)를 리턴한다. 즉, 메소드 Bonus는 질의에서 정수형 속성과 동일하게 취급될 수 있다. 이러한 유사성을 바탕으로 GOQL에서는 메소드를 속성과 동일한 그래픽 요소로 표현한다. 즉, 메소드가 프리미티브 타입의 값을 리턴하면 프리미티브 속성으로, 사용자 정의 타입의 객체를 리턴하면 복합 속성으로 간주한다. 다만, 인자가 있는 경우 GOQL에서 인자 값을 대화식으로 입력 받을 수 있도록 그래픽 요소를 제공한다. 그림 8에서 클래스 C1에 정의된 메소드 m()은 정수를 인자로 받아서 C2 타입의 객체를 리턴한다. 사용자가 클래스 C1에서 메소드 m()을 선택하면 GOQL은 인자의 값을 입력하도록 요구하며, 사용자가 인자의 값으로 10을 입력하면, 메소드 m()이 리턴하는 객체의 도메인인 C2 클래스에 대한 노드가 생성된다. 그림 8의 그래픽 질의는 m()이 리턴하는 C2 객체의 속성 A3 값이 30 이라는 조건을 시각적으로 표현한 것이다. 만일, 메소드 m()이 집합값을 리턴한다면 C1.m(10).A3은 집합 경로가 되고, 사용자는 이 집합 경로에 대하여 원소 조건이나 집합 조건을 명시할 수 있다.

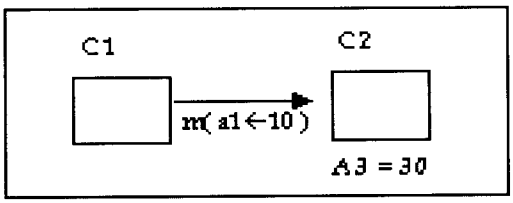


그림 8 GOQL에서 메소드 호출

3.3 경로 조건들의 AND 표현

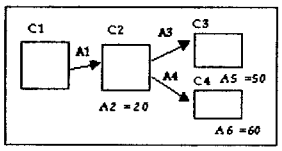
지금까지 단일 경로에 대한 조건의 시각적 표현에 관하여 살펴보았다. 여기서는 여러 조건들이 AND로

결합되는 경우에 그래픽 질의로 표현하는 방안을 살펴본다. OR 연산자를 포함한 질의는 불리언 연산의 정규화를 통하여 AND 연산자 만을 포함한 질의들로 변환하여 처리할 수 있으므로 GOQL에서는 제외한다.

경로 조건들의 AND 표현은 그래픽 질의의 작성 순서를 살펴봄으로써 쉽게 이해할 수 있다. GOQL 사용자는 질의를 작성하기 위하여 먼저 질의의 대상이 되는 클래스를 하나 선택하고, 이 클래스로부터 복합 속성(들)을 선택해 나가면서 노드에 조건 레이블을 부과함으로써 질의를 완성하게 된다. 예를들어, 그림 9 (a)의 그래픽 질의는 사용자가 다음의 과정을 거쳐서 작성한다.

- 질의 대상으로 클래스 C1을 선택함
- 속성 A1을 선택하면 C2 노드가 생성되고, 여기서 조건 A2=20을 첨가함
- C2에서 속성 A3과 A4를 선택하면 노드 C3과 C4가 생성됨
- C3과 C4에 조건 A5=50과 A6=60을 각각 부과하여 질의 작성을 완성함

질의 작성 과정에서 사용자가 조건 레이블을 다수개 부과하면 GOQL은 이들을 AND로 연결한다. 그림 9(a)의 그래픽 질의에서는 클래스 C2에 조건 A2=20, C3에 A5=50, C4에 A6=60을 부과하고 있으며, 이들은 모두 AND로 연결된 조건이다. GOQL 시스템은 그림 9(a)의 그래픽 질의를 그림 9(b)와 같은 동일한 의미를 가지는 텍스트 질의로 변환하여 객체 DBMS로 보내서 질의를 처리하게 된다. 변환된 텍스트 질의에서는 세개의 조건이 AND로 연결되어 있음을 볼 수 있다. 임의의 GOQL을 텍스트 질의로 변환하는 알고리즘은 제 3.4 절에서 소개한다.



(a) AND로 연결된 질의 조건들

C1.A1[C2].A2=20 And
C1.A1[C2].A3[C3].A5=50 And
C1.A1[C2].A4[C4].A6=60

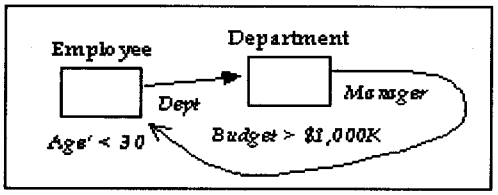
(b) 대응하는 텍스트 조건

그림 9 다수의 경로 조건들이 AND로 연결된 경우 GOQL 표현

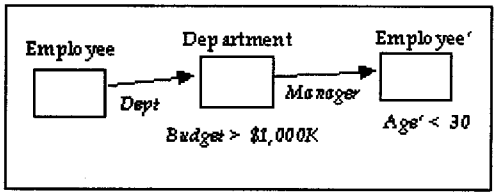
이와 같이 AND로 연결된 조건들을 GOQL로 표현할 때 한가지 유의할 사항이 있다. 사용자가 도메인이 동일한 두개 이상의 복합 속성을 질의 작성 과정에서 선택하는 경우 도메인에 해당하는 클래스를 GOQL에서 몇 번이나 표시하는가 ? 라는 문제이다. 예들들어, 그림 1의 스키마에 대하여 다음의 경로 조건과 그에 대응하는 GOQL 표현을 생각하자.

Employee.Dept[Department].Budget > $1,000K
And
Employee.Dept[Department].Manager[Employee].Age < 30

사용자는 질의 대상으로 Employee 클래스를 선택하여 Employee 노드를 생성하고, 속성 Dept를 선택하여 Department 노드를 생성하며, 조건 Budget > $1,000K를 레이블로 첨가한다. 그리고, Department 노드에서 Manager 속성을 선택하면 도메인인 Employee 클래스가 이미 GOQL의 노드로 존재하므로 이를 활용하는 방안(선택-1)과 별도의 노드를 생성하는 방안(선택-2)을 생각할 수 있다.



(a) 선택-1



(b) 선택-2

그림 10 경로 조건에서 두 번 이상 선택되는 클래스의 GOQL 표현

GOQL에서는 하나의 클래스가 서로 다른 두 개 이상의 복합 속성으로 인하여 두 번 이상 선택되는 경우에는 별도의 노드를 생성하고, 클래스/속성 이름 뒤에 기호 ', '', ...를 붙여서 각 노드를 서로 구분하도록 한다 (그림 10(b) 방식). 하나의 클래스가 서로 다른 두 속성의 도메인 자격으로 질의에 나타나면 그 클래스는 질의에서 두가지의 다른 의미를 가지기 때문에 서로 다른 노드로 표시하여 구분해야 모호성이 없어진다. 예를들어, 그림 10에서 복합 속성 Manager에

의하여 선택되는 Employee 클래스는 질의에서 "직원"이 아니라 "관리자"의 의미를 가지게 되므로 그림 10(b)와 같이 별도의 노드 Employee'로 표시한다. 만일, 그림 10(a)와 같이 이들을 서로 다른 노드로 표시하지 않고 하나의 노드로 표시하면 싸이클이 형성되어 대응하는 텍스트 질의를 생성하는데 어려움이 있게 된다. 즉, 그림 10(a)에서는 조건 Age<30이 Employee에 대한 조건인지 혹은 Employee가 근무하는 Department의 Manager에 대한 조건인지 그래픽 자체로서는 구분하기가 불가능하다. 그러나, 그림 10(b)에서는 이 조건이 Manager에 대한 조건임이 분명해진다.

반면, 동일한 속성으로 인하여 하나의 클래스가 두 번 이상 선택되거나 경로의 시작 노드에 해당하는 클래스는 질의 조건에서 두 번 이상 나타나면 GOQL에는 한 번만 표시한다. 예를들어 위의 두 조건에서 경로의 시작 노드인 Employee는 두 번 나타나지만 하나의 노드로 표시하였으며, Department도 속성 Dept에 의하여 두 번 선택되었으나 GOQL에서는 한 번만 표시되었다.

3.4 GOQL을 텍스트 객체 질의어로 변환하는 알고리즘

여기서는 GOQL의 그래픽 질의를 동일한 의미를 가지는 텍스트 객체 질의로 변환하는 알고리즘을 설명한다. 질의 변환 알고리즘은 Tachyon ODBMS[17]와 Java 및 JDBC를 이용하여 실제로 구현되어 동작하고 있으며, 메소드는 Tachyon에서 지원되지 않기 때문에 제외되었다.

질의 변환 알고리즘의 동작 과정은 다음과 같다. 먼저 GOQL로부터 소스 노드(들어오는 차수가 0인 노드)를 찾아서 이를 From 절에 지정한다. 소스 노드는 경로의 시작 노드가 되기 때문에 From 절에 지정하며, 나머지 노드들은 시작 노드로부터 경로를 따라서 나타나므로 From 절에 나타날 필요가 없다. 다음으로 GOQL에 포함된 각 조건 A θ B에 대하여 완전한 경로를 생성한다. 완전한 경로를 생성한다는 것은 조건에 나타난 각 속성에 대하여 소스 노드로부터 그 속성이 정의된 클래스까지의 경로를 속성 이름 앞에 붙여주는 것이다. 즉, 그래프 형태로 나타난 경로를 텍스트 경로로 변환하여 텍스트 질의를 생성하기 위함이다. 다음에는 경로에 집합 속성이 포함됨에도 불구

TransGOQLtoOQL() :

```
(1) Input: GOQL = <N, E, C>
(2) Output: Object SQL having equivalent semantics
(3) F-clause=Null;                  // From clause
(4) W-clause=Null;                        // Where clause
(5) for each node M ∈ N                   // find the source node
       if (indegree(M) == 0) {
             F-clause = F-clause || "M m," ;                    // From clause
(6) F-clause=removeLastComma(F-clause);// remove the last comma from F-clause
(7) for each condition A θ B ∈ C {          // make conditions for the Where clause
(7-1)  if (A is an attribute)
                    pathL=expandPath(A);   // make a complete path expression
(7-2)  if (B is an attribute)
                    pathR=expandPath(B); // make a complete path expression
(7-3)  if ( ∃set attributes s1,..., sk ∈ pathL and θ is a scalar operator) {
             // employee the object variable ti for each set attribute si
             // in the From-clause; element comparison
             F-clause = F-clause || ", m.s1 t1, t1.s2 t2, ..., tk-1.sk tk" ;
             pathL = replaceSetPathToObjectVar(pathL);
                         // replace the set path into the object variable
       }
(7-4)  if ( ∃set attributes s1,...,sk ∈ pathR and θ is a scalar operator) {
            F-clause = From-clause || "m.s1 t1, t1.s2 t2, ..., tk-1.sk tk" ;
            pathL = replaceSetPathToObjectVar(pathR);
                         // replace the set path into the object variable
      }
(7-5)  W-clause = W-clause || "pathL θ pathR And" ;
    } // for
(8) W-caluse = removeLastAnd(W-clause);
(9) P-clause = "*";
(9) return("Select" || P-clause || "From" || F-clause || "Where" || W-clause);
```

그림 11 GOQL을 텍스트 객체 질의로 변환하는 알고리즘

하고 사용자가 조건 A θ B의 연산자로 스칼라 연산자를 사용한 경우를 검사한다. 이는 집합 경로에 대하여 원소 조건을 명시한 경우를 인식하여 객체 변수로 대응시켜 텍스트 질의로 생성하기 위함이다. 각 조건에 대하여 완전한 경로가 생성되면 이들을 AND 연산자로 연결하여 완전한 Where 절을 생성한다. 다음으로, 사용자가 프로젝션 리스트로 지정한 속성(들)을 Select 리스트에 나열하여 프로젝션을 완성한다.

그림 11은 질의 변환 알고리즘을 보여준다. 질의 변환 알고리즘에 포함된 함수들의 의미는 다음과 같다.

- expandPath(A) : 소스 노드(들어오는 차수가 0인 노드)에서 A가 정의된 노드까지의 경로를 속성 A 앞에 첨가함.
- replaceSetPathToObjectVar() : 집합값을 가지는 서브 경로를 객체 변수로 대치함.
- removeLastComma() : From 절에서 마지막 컴마(,)를 삭제함.
- removeLastAnd() : Where 절의 조건 스트링에서 마지막에 나타나는 "And"를 제거함.

(예제) 그림 12의 그래픽 질의에 대하여 텍스트 질의로 변환하는 과정을 살펴보자.

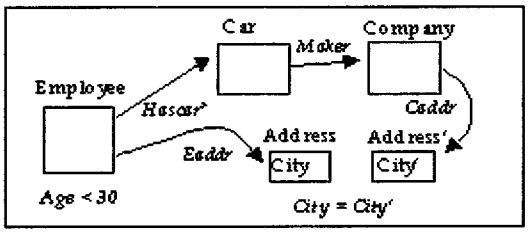


그림 12 GOQL의 예

질의 변환 알고리즘의 입력으로 사용되는 GOQL은 다음과 같은 그래프이다.

N = {Employee, Car, Company, Address, Address'}

E = {Hascar, Eaddr, Maker, Caddr}

C = {Age < 30, City=City'}

먼저, 알고리즘의 step 5에서 F-clause = "From Employee e"를 생성한다. 다음으로, 조건 레이블 L의 각 조건에 대하여 step 7과 그의 내부 루프(7-1~7-5)에서 다음과 같이 속성 앞에 완전한 경로를 첨가한다. 먼저 Age > 30에 대하여 경로의 시작 노드인 Employee에서 Age가 속한 클래스인 Employee까지의 경로를 Age 앞에 추가한다. 이 경우 클래스 이름 대신에 동일한 의미를 갖는 객체 변수 e를 사용하여 e.Age>30으로 조건을 생성한다. 다음으로 조건 City=City'에 대하여 속성 이름 앞에 경로를 추가한다. 먼저, City에 대하여 PathL= Employee.Eaddr.City를 생성하고, City'에 대하여 PathR=Employee.Hascar.Maker.Caddr.City를 생성한다. 그런데 PathR = Employee.Hascar.Maker.Caddr.City에서 속성 Hascar가 집합값을 가짐에도 불구하고 스칼라 연산자 =를 사용하므로 step 7-3에서 From 절에 객체 변수 c를 도입하고, 객체 변수 c를 이용하여 조건에 포함된 집합 경로(Employee.Hascar)를 대치한다. 따라서 다음의 변수 값을 생성한다.

F-clause : *From Employee e, e.Hascar c*

PathR : *c.Maker.Caddr.City*

W-clause : e.Age > 30 And *e.Eaddr.City = c.Maker.Caddr.City*

마지막으로 step 10에서 리턴하는 결과는 다음과 같다. 여기서 프로젝션 리스트는 복잡성을 피하기 위하여 *로 표시하였으며, 제 4 장의 구현에서 다시 설명한다.

*Select **

From Employee e, e.Hascar c

Where e.Age >30 And c.Maker.Caddr.City = e.Eaddr.City;

3.5 GOQL의 표현력

관계 데이타베이스에서는 관계 대수가 관계적으로 완전한 질의임이 증명되어 있으므로 새로운 질의어를 제안할 때 관계 대수와 비교하여 그 표현력의 정도를 측정할 수 있다. 그러나, 객체 질의어의 경우는 질의어 자체가 연구 단계에 있으며, 대다수가 공감하는 표준 질의어도 아직 제정되지 못한 상태이므로 이러한 비교가 어려운 실정이다. 여기서는 GOQL의 표현력을 기존의 여러 객체 질의어 중에서 표현력이 비교적 뛰어난 XSQL[1]와 비교한다. XSQL의 가장 중요한 특징은 확장된 경로에 대한 조건을 사용한다는 점이다. 여기서는 XSQL의 경로에 대한 조건을 GOQL의 노드와 링크를 사용하여 시각적으로 표시할 수 있음을 입증한다.

먼저, 길이가 1인 경로(하나의 클래스와 하나의 속성으로 이루어짐)에 대한 조건을 살펴보자. 이 경우 경로에 포함된 클래스는 GOQL에서 사각형 노드로 표시하며, 속성과 그에 대한 조건은 노드의 조건-레이블로 표현된다. 메소드가 포함된 경로는 제 3.2.4에서 살펴본 바와 같이 속성과 유사하게 취급하며, 집합 속성이 포함된 경로는 집합 연산자를 사용하여 표시할 수 있다. 다음으로 길이가 i (>=2)인 경로는 경로

가 i-1인 경로의 표현으로부터 쉽게 확장하여 표현할 수 있다. 제 3.2.2 절과 제 3.2.3 절에서 길이가 2인 경로와 길이가 3 이상인 경로에 대한 조건의 그래픽 표현을 제안하였다. 마지막으로 다수의 경로 조건들이 AND로 연결된 경우에도 GOQL에서는 트리 형태로 이들을 표현할 수 있다. 본문에서는 그래픽 질의에서 하나의 클래스가 두 번 이상 선택되는 경우에도 동일한 속성에 의하여 선택된 클래스들을 하나의 노드로 표시함으로써 복잡성을 줄이고, 그렇지 않는 경우에는 노드를 중복하여 표시하는 방식으로 모호성이 발생하지 않도록 하였다. 다만, OR로 연결된 경우는 두개 이상의 AND로 연결된 질의로 변환하여 처리할 수 있으므로 여기서는 다루지 않았다. 따라서 GOQL은 XSQL에서 사용되는 경로에 대한 조건 중에서 OR로 연결된 경우를 제외하고는 모두 표현하고 있음을 알 수 있다. 다만, 뷰를 포함한 질의나 부질의어 에 관한 그래픽 질의어 표현은 별도의 연구 과제이므로 본 논문에서는 다루지 않는다.

4. GOQL의 구현

GOQL은 제 3장에서 소개한 그래픽 요소를 이용한 검색 뿐 아니라 사용자 편의를 증진시키기 위하여 스키마 관리, 데이타 변경 등의 편리한 기능들을 가지고 있다. 여기서는 GOQL의 구현에 관하여 플랫폼과 실험 과정을 기술한다.

4.1 실험 플랫폼

GOQL은 그림 13과 같은 플랫폼에서 개발되었다. GOQL은 Java/JDBC 프로그램으로 작성되어 PC 상에서 인터넷을 통하여 원격지 객체 데이타베이스와 연결된다. 원격지 DBMS로는 Sun 엔트프라이즈 250에서 운용되는 Tachyon ODBMS[17]가 사용되며, 현재 Odysseus ODBMS [18]와의 연결 작업도 진행 중에 있다. 그림 13에서 사용자는 GOQL을 사용하여 질의의 결과 뿐 아니라 데이타베이스 카탈로그 정보를 제공받을 수 있으며, 이 정보는 사용자가 질의를 작성하는데 이용된다.

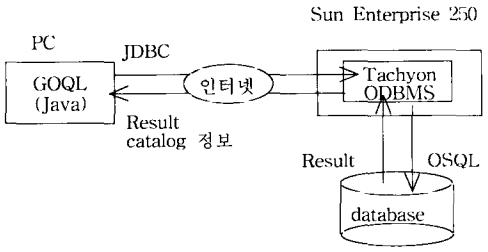


그림 13 실험 환경

GOQL의 사용 과정은 그림 14와 같은 시퀀스 다이어그램으로 표시된다. 그림에서 점선 화살표는 선택적임을 의미하고, * 표시는 반복 수행될 수 있다는 의미이다. 또한, GUI에서 박스는 GOQL 사용자가 수행하는 작업들을 의미한다. 그림에서 사용자가 GUI를 초기화하면 GOQL은 사용자로부터 IP를 입력받아서 원격지 DBMS를 연결한다. 사용자는 원격지 DBMS에서 관리하는 데이타베이스 리스트를 전달받아서 그 중 하나를 선택한다. DBMS는 선택된 데이타베이스를 오픈하여 질의의 작성과 실행이 가능하도록 하고, 사용자에게 데이타베이스에 포함된 테이블 리스트를 전달한다. 사용자가 질의 대상이 되는 테이블을 선택하면 GOQL에서는 그 테이블을 노드 형태의 그래픽 질의로 만들어 GUI에 제공하고, DBMS로부터 테이블 정보 (속성과 타입)를 받아서 GUI로 전달한다. 사용자는 테이블 정보를 보면서 질의 조건을 작성하게 되고, GOQL은 이를 그래픽 질의 형태로 GUI에 전달하며, 사용자가 추가로 테이블을 선택하는 경우에 그 테이블에 관한 정보를 DBMS로부터 받아서 GUI로 전달하는 과정을 반복 수행한다. 그래픽 질의가 완성되면 사용자는 Execute 메시지를 보내고, GOQL은 이를 동일한 의미를 가지는 텍스트 질의로 변환하여 DBMS로 보내며, DBMS는 이를 처리하여 질의 결과를 GUI로 보낸다.

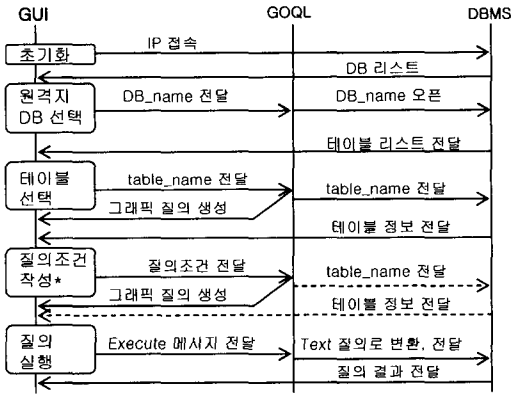


그림 14 그래픽 질의어 작성과 실행 과정

4.2 GOQL의 실행 예

다음에는 GOQL의 사용 방법을 예를 통하여 설명한다. GOQL에는 그림 15 상단과 같이 Connection Manager, Schema Manager, Query Manager, Util-

ities 메뉴가 있다. Connection Manager는 인터넷으로 연결된 원격지 데이타베이스들 중에서 사용자가 관심이 있는 데이타베이스를 선택할 수 있도록 한다. Schema Manager는 선택된 데이타베이스에 대하여 스키마 변경 작업(예를들어, 클래스의 생성과 삭제 등)을 지원하는 윈도우이다. Query Manager는 그래픽 질의를 작성할 수 있도록 지원하는 윈도우이며, Utilities는 데이타베이스 관리에 유용한 도구(로더/언로드 등)들을 포함하고 있다. 여기서는 이들 중에서 Query Manager에 관하여 살펴본다.

GOQL/Query Manager의 메인 윈도는 그림 15와 같이 스키마 정보 영역과 그래픽 질의 영역 및 텍스트 질의 영역으로 구성된다. 스키마 정보 영역은 선택된 데이타베이스의 스키마 정보 (클래스와 속성 정보)를 보여주고 있다. 그림에서는 "전자상거래" 데이타베이스에 car, company, department, employee (서브클래스로 engineer) 클래스가 포함되어 있으며, 각 클래스에는 속성 정보가 보여지고 있다. 그래픽 질의 영역은 사용자가 그래픽 질의어를 작성하는 영역이다. 사용자가 스키마 정보 영역에서 질의 대상이 되는 클래스를 클릭하면 그 클래스에 대한 노드가 그래픽 질의 영역에 나타나게 된다. 또한, 사용자가 선택된 클래스에서 복합 속성을 클릭하게 되면 도메인에 해당하는 클래스의 노드가 그래픽 질의 영역에 나타나는 방식으로 경로가 시각적으로 표시된다. 사용자는 그래픽 질의 영역에 나타난 노드에 대하여 조건 레이블을 부과함으로써 질의 작성을 마치게 된다. 텍스트 질의 영역에는 그래픽 질의어와 동등한 텍스트 질의가 생성되는 곳이다. 사용자는 그래픽 질의 영역에서 그래픽 질의를 작성한 다음에 텍스트 질의 생성 버튼을 클릭하면 상단의 텍스트 질의 영역에 동일한 의미의 텍스트 질의가 생성된다. 현재 Tachyon ODBMS의 문법에 맞는 OSQL이 생성된다. 숙달된 사용자는 그래픽 질의 작성 없이 바로 텍스트 질의 영역에 OSQL

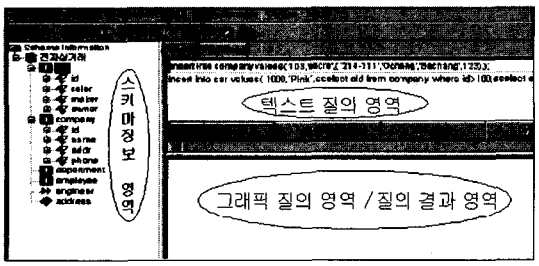
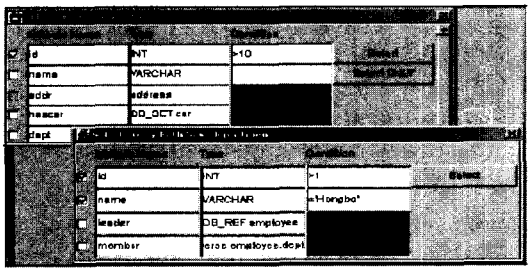


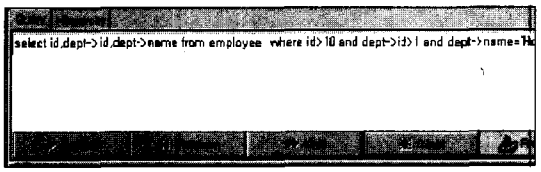
그림 15 GOQL의 메인 윈도우

을 입력하여 실행할 수도 있다.

예를 통하여 GOQL의 동작 과정을 살펴본다. 그림 16(a)는 그림 15의 스키마 정보 영역으로부터 employee 클래스를 선택하여 생성한 그래픽 질의어를 보여주고 있다.



(a) 경로 조건의 표현 : id >10 and dept.id > 1 and dept.name="Hongbo";



(b) 대응하는 텍스트 객체 질의의 생성

그림 16 그래픽 질의어 작성과 대응하는 텍스트 질의 생성

사용자는 employee 노드에서 단순 속성에 대한 조건을 기술하거나 혹은 복합 속성을 더블 클릭하여 복합 속성의 도메인을 클래스 노드 형태로 생성할 수 있다. 그림 16(a)는 사용자가 employee.id > 10이란 조건을 부과한 후, 복합 속성 dept를 클릭하여 department 노드를 생성한 모습이다. 그리고, 사용자는 노드 department에 대하여 조건 department.id > 1과 department.name = "Hongbo"를 부과하고 있다. 클래스 department에서 조건 작성이 완성되면 사용자는 완료 버튼을 클릭하여 텍스트 질의를 생성한다. 그림 16(b)는 그림 16(a)의 그래픽 질의와 동일한 텍스트 질의를 보여주고 있다. GOQL에서는 그림 16(b)의 텍스트 질의를 원격지 객체 데이타베이스에 전달하여 실행하고, 그 결과를 그림 15의 그래픽 질의 영역에 테이블 형태로 보여주게 된다. 그림 16에서 좌측의 체크 박스는 프로젝션 리스트를 지정한 모습을 보여준다. 현재 질의의 프로젝션 리스트는 employee.id, employee.dept.id, employee.dept.name이며, 질의 결과

에는 이들 속성의 값이 저장된다. 그림 17은 그림 16의 질의 결과를 보여주고 있다. 질의 결과는 그래픽 질의 영역에 테이블 형태로 나타난다.

사용자의 질의 작성과 처리 과정은 그림 18과 같다. 사용자는 GOQL을 실행하여 원격지 데이타베이스를 연결한 후, 그림 15의 스키마 정보 영역에 나타난 클래스 중에서 질의 대상이 되는 클래스를 선택한다. GOQL은 선택된 클래스 정보 (속성과 그 타입)를 시스템 카탈로그로부터 가져와서 그림 15과 같이 그래픽 질의 영역에 노드 형태로 보여준다.

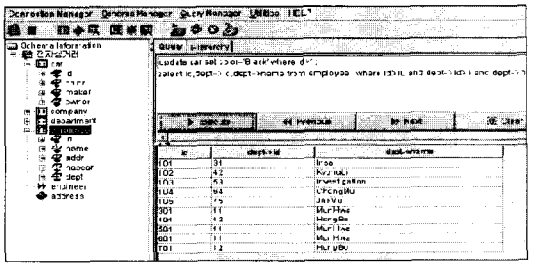


그림 17 질의 결과

사용자는 클래스 정보를 보면서 속성에 질의 조건을 명시하거나 혹은 속성의 도메인이 되는 클래스를 질의 대상으로 추가할 수 있다. 그림 16과 같이 그래픽 질의 영역에서 그래픽 질의를 작성한 후에 Execute 버튼을 누르면 텍스트 질의 영역에 그래픽 질의와 동일한 의미를 가지는 텍스트 질의가 생성되고 (그림 16(b) 참고), 이것을 원격지 DBMS로 보내서 처리하게 된다. 질의의 결과는 그림 17과 같이 질의 결과 영역에 테이블 형태로 나타나게 된다.

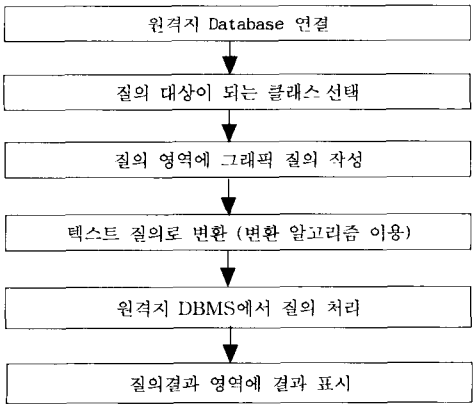


그림 18 GOQL 시스템의 질의 작성과 처리 과정

5. GOQL 시스템의 평가

이 절에서는 그래픽 객체 질의어에 대한 몇가지 평가 요소를 설정하고, 이에 대하여 GOQL과 기존의 대표적인 그래픽 객체 질의어를 평가한다. 기존의 그래픽 질의어는 대부분 관계 데이타베이스와 개체-관계 데이타베이스를 대상으로 한 것이며, 객체 데이타베이스를 위한 것은 소수에 불과한 상황이다[10,20,11,13,4,2,6]. 여기서는 기존의 그래픽 질의어 중에서 객체 데이타베이스를 위한 그래픽 질의어로 한정하여 제안된 그래픽 객체 질의어와 비교 평가한다.

그래픽 객체 질의어를 설계하기 위하여 첫째로는 스키마의 표현 방식과 그래픽 객체 질의어의 표현 방식을 결정하는 것이 중요하다. 그래픽 질의어의 표현 방식은 폼기반(form-based), 아이콘 기반(icon-based), 다이어그램 혹은 그래프 기반 (diagram-based), 하이브리드 방식(hybrid-based) 등으로 구분된다[21]. 객체 데이타베이스의 스키마가 그래프 형태로 표시되므로(이를 **스키마 그래프**라고 부름)[19] 그래픽 객체 질의어도 스키마 그래프와 유사한 형태를 취하는 것이 사용자 편의성 측면에서 바람직하다. 이렇게 함으로써 사용자는 하나의 개념인 그래프로 스키마와 질의어를 모두 인식할 수 있게 된다. 두 번째 고려할 사항으로는 그래픽 질의어의 표현력을 최대화 해야 한다는 점이다. 관계 질의어와 비교하여 객체 질의어의 가장 중요한 특성은 다수의 조인과 셀렉션 연산을 표현하기 위하여 길이가 2 이상인 경로식을 사용한다는 점이다. 특히, 경로식에는 집합값을 가지는 속성과 메소드가 포함될 수 있다는 점에서 그래픽 관계 질의어보다 더욱 복잡한 시각화 기법이 요구된다. 경로에 집합값 속성이 포함되면 집합 연산자가 사용될 뿐 아니라 전체 및 존재 한정어(quantifier)가 사용될 수 있으며, 메소드가 포함되면 인자(들)가 필요하므로 이들을 시각화하는데 복잡성이 증대되기 때문이다.

여기서는 이러한 객체 질의어의 특성을 감안하여 다음의 관점에서 GOQL과 기존의 그래픽 객체 질의어를 비교 평가한다.

(1) 그래픽 질의어의 표현 방식

객체 데이타베이스의 스키마가 그래프 형태로 자연스럽게 표현되므로[19] 참고문헌 [4,2,6]을 제외한 대부분의 기존 그래픽 객체 질의어[10,20,11,13]가 그래프 표현방식을 취하고 있다. 그래프 기반의 경우 하나의 시각적 도구인 그래프를 이용하여 스키마와 질의를 모두 표현하므로 사용자 측면에서 복잡성이 줄어든다. 특히, 최근에는 사용자 편의성을 높이기 위하여

그래프 기반에 폼을 추가하거나[10,20] 아이콘을 추가한[11] 혼합 방식이 채택되는 추세이다. 제안된 GOQL은 스키마 그래프의 서브 그래프에 노드 자체가 특정한 폼을 가지도록 확장한 혼합 방식에 해당한다.

(2) 길이가 2 이상인 경로식의 시각화

대부분의 그래픽 객체 질의어에서 경로식은 그래프에서의 경로와 같이 노드와 에지의 반복으로 시각화하고 있다[20,11,13]. 그러나, 관계형 그래픽 질의어[15]나 폼 기반의 그래픽 질의어[4,2,6]에서는 두 노드 사이의 조인 관계를 QBE[15]에서와 마찬가지로 동일한 변수명을 사용하여 표현한다. GOQL은 스키마 그래프의 서브 그래프로써 경로가 노드와 에지의 반복으로 표현된다. 특히, 사용자가 경로를 생성할 때 이미 생성된 노드에서 복합 속성을 클릭하면 그 속성의 도메인에 해당하는 클래스 노드가 생성되는 방식이므로 길이가 긴 경로를 간단하게 시각화 한다는 장점을 가진다.

(3) 집합 속성과 집합 연산자의 시각화

GOQL을 포함한 대부분의 그래픽 객체 질의어에서 집합 속성과 집합 연산자를 제공하고 있다[10,20,11,13,4,6]. 그러나, 제안된 GOQL과 참고문헌 [20,6]에서는 확장된 경로식(extended path expression)을 사용하는 XSQL[1]에 적합하도록 시각화한다는 점에서 차이가 있다. ODMG의 객체 모델[16]을 지원하는 [10,11]에서는 경로식의 중간에 집합 속성이 포함되면 객체 변수를 이용하여 원소에 대한 조건으로 표현하지만, XSQL[1]에서는 집합값 전체에 대하여 조건을 부과하는 **집합 조건**과 집합의 각 원소에 대하여 조건을 부과하는 **원소 조건**으로 구분하여 더욱 세분화된 조건을 제시할 수 있으므로 이러한 점을 감안하여 시각화 기법이 제시되어야 한다. GOQL에서는 [20,6]과 달리 원소 조건과 집합 조건을 사용자가 별도로 의식하지 않고서도 올바르게 질의를 작성할 수 있도록 함으로써 편의성을 높이고 있다.

(4) 한정어(restricted quantifier)의 시각화

한정어의 경우 [11,4,2]를 제외한 대부분의 그래픽 질의어[10,20,13,6]에서 제공하고 있다. [10]의 경우 한정어를 집합 연산자로 변환하여 처리하는 방식이며, [20]에서는 별도의 ALL/SOME/NOT 버튼을 사용하여 한정어를 표현하고 있고, [13]의 경우 벤다이어그램 방식을 제안하고 있으나 실제 구현되지는 않았으며, [6]의 경우 컴포넌트 스켈러톤(component skeleton)이라고 하는 특별한 박스를 사용하여 한정어를 표현하고 있다. 제안된 GOQL에서는 [20]과 유사한 방식이나 버튼을 선택하는 대신 사용자가 텍스트로 All이라는 한정어를 기입하도록 한다는 점에서 차이가 있다.

(5) 메소드의 시각화

기존의 그래픽 질의어들은 메소드를 시각화하는 경우[10,20,11]와 그렇지 않은 경우[8,16,18,19]로 나누어진다. GOQL에서는 메소드도 속성과 동일하게 취급하여 시각화하는 방안을 제시하였다. 특히, 인자를 가지는 메소드의 경우 질의에서 사용자가 인자의 값을 입력할 수 있도록 하는 방식으로 그래픽 질의를 제안하였다.

(6) Java/JDBC로 구현

대부분의 그래픽 질의어는 특정한 객체 DBMS를 가정하여 설계 제작되고 있으나 Java/JDBC를 이용하여 제작된 경우는 거의 없다. VISUAL[10]은 C++로 제작되어 O2 시스템과 연결되며, PESTO[20]은 C++/Tcl로 제작되어 DB2, ObjectStore 등과 연결된며, Visual-SQL[11]은 Delphi로 구현되어 UniSQL과 연결된다. 그러나, [13,6]은 실제로 구현되지 않았다. 제안된 GOQL은 JAVA/JDBC를 사용하여 구현하였으며 Tachyon[17]과 Odysseus[18]과 연결된다. 또한, GOQL은 JDBC를 지원하는 다양한 DBMS로 쉽게 이

표 1 평가 요소에 대한 그래픽 질의어들의 비교 분석

	[10]	[20]	[11]	[13]	[4]	[6]	GOQL
GQL 표현방식	Graph +Form	Graph +Form	Graph +Icon	Graph	Form	Form	Graph +Form
길이가 2이상인 경로식 표현	path	path	path	path	variable	variable	path
집합속성	D	O	O	O	O	O	O
한정어	O	O	X	O	X	O	O
메소드	O	O	O	X	X	X	O
구현언어 (DBMS)	C++ (O2)	C++/Tcl (DB2)	C++/Tcl (O2)	X	X	X	Java/JDBC (Tachyon)
Subquery	O	X	O	X	X	X	X
Schema Query	O	X	X	X	X	O	X

식될 수 있다는 점과 하부 시스템의 플랫폼에 영향을 받지 않는다는 점에서 장점이 있다.

(7) 부 질의어와 스키마 질의

부 질의어는 질의어 안에 질의어가 포함된 형태로써 대부분의 그래픽 질의어서는 허용되지 않는다. 스키마 질의어는 XSQL[1]에서 제안된 것으로 스키마 구조를 정확하게 모르는 사용자가 질의를 할 수 있도록 하는 기능이며, 이것도 대부분의 그래픽 질의어에서 제공되지 않고 있다.

이상의 평가 요소들에 대한 비교는 표 1과 같이 요약된다. 테이블에서 길이가 2 이상인 경로식의 표현으로 path는 노드와 에지로 경로식을 표현한다는 의미이고, variable이란 QBE[15]처럼 동일한 변수명을 사용하여 조인 관계를 표시하는 방식을 채택한다는 의미이다. 표에서 Visual-SQL[2]는 그래픽 질의어라기보다 텍스트 질의를 편리하게 작성하기 위한 도구이므로 평가요소의 대부분을 지원하지 않으며, 따라서 비교 대상에서 제외하였다.

6. 결 론

본 논문에서는 인터넷 환경에서 객체 데이타베이스를 위한 그래픽 질의어인 GOQL을 제안하였다. GOQL 사용자는 GUI 형태로 제공되는 데이타베이스 카탈로그 정보를 보면서 질의 대상이 되는 클래스를 클릭하여 그래픽 노드를 생성하고, 이로부터 복합 속성을 클릭하여 새로운 클래스 노드를 생성하거나 노드에 조건을 부과함으로써 경로에 대한 조건을 간단히 작성할 수 있다.

GOQL의 특징은 객체 질의어에서 널리 사용되는 경로에 대한 조건을 간단한 그래픽 요소들로 시각화한다는 점이다. 특히, GOQL에서는 집합 속성이 포함된 경로에 대한 조건을 원소 조건과 집합 조건으로 구분하여 간단히 시각화하는 방안을 제시하였다. 원소 조건의 경우 집합 경로에 대하여 스칼라 연산자를 사용하여 그래픽 질의 조건을 작성하는 방식으로 시각화하며, 객체 변수를 도입하여 텍스트 질의로 변환한다. 집합 조건의 경우 집합 연산자나 한정어를 가진 스칼라 연산자를 사용하여 그래픽 질의 조건을 작성하며, 객체 변수를 도입할 필요 없이 이들 연산자를 그대로 사용하여 텍스트 질의로 변환한다. 경로의 길이가 2이상인 일반적인 경우에도 사용자는 그래픽 요소들을 클릭하고, 노드에 대하여 조건을 부과하는 방식으로 간단히 그래픽 질의를 작성할 수 있다. 그리고, 메소드를 포함한 질의에서 메소드는 속성과 유사하게 취급하여 그래픽 질의를 작성할 수 있도록 하였다. 논문에서는 GOQL을 텍스트 질의로 변환하는 알고리즘을 제안하였으며, XSQL[1]에서의 경로 조건은 GOQL의 그래픽 질의 조건으로 표현할 수 있음을 보였다. 제안된 GOQL은 인터넷 상의 원격지 객체 데이타베이스 시스템과 연결되어 사용되도록 JAVA와 JDBC를 이용하여 구현하였다. 또한, GOQL에서는 질의 기능뿐 아니라 사용자 편의를 위하여 스키마 관리 기능과 데이타 삽입/삭제/변경 기능 및 데이타베이스 로딩/언로딩 기능을 GUI 환경에서 간단히 수행할 수 있도록 하였다.

향후, 부 질의어(subquery)와 스키마 질의 및 뷰(view)를 포함한 질의 기능을 GOQL에 추가할 예정이며, GOQL과 연결되는 서버 객체 DBMS의 종류도 현재의 Odysseus[18]과 Tachyon 객체 DBMS[17]에서 지속적으로 확장해 나갈 예정이다.

참 고 문 헌

[1] M. Kifer et al., "Querying Object-Oriented Databases," In *Proc. ACM SIGMOD Int'l Conf. on Management of Data*, pages 393-402, 1990.

[2] UniSQL/Visual-SQL, http://www.kcom.co.kr, 2002.

[3] J. Paredaens et al., "A-Gog: A Graph-Based Query Language," *IEEE Trans. on Knowledge and Data Engineering*, Vol. 7, No. 3, June 1995.

[4] F. Staes et al., "A Graphical Query Language for Object-Oriented Databases," In *Proc. IEEE Workshop on Visual Languages*, pages 205-210, 1991.

[5] G. H. Sockut et al., "GRAQULA: A Graphical Query Language for Entity-Relationship or Relational Databases," *Data and Knowledge Engineering*, Vol.11, No. 2, pages 171-202, 1993.

[6] K. Vadaparty et al., "Towards a Unified Visual Database Access," In *Proc. ACM SIGMOD Int'l Conf. on Management of Data*, pages 357-366, 1993.

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

[8] G. Ozsoyoglu and H. Wang, "A Relational Calculus with Set Operators, Its Safety, and Equivalent Graphical Languages," *IEEE Trans. on Software Engineering*, Vol. 15, No. 9, Sept. 1989.

[9] K. Y. Whang et al., "Two-Dimensional Specifica-

tion of Universal Quantification in a Graphical Database Query Language," *IEEE Trans. on Knowledge and Data Engineering*, Vol. 18, No. 3, Mar. 1992.

[10] N. H. Balkir et al., "A Graphical Query Language: VISUAL and its Query Processing," *IEEE Trans. on Knowledge and Data Engineering*, Vol. 14, No. pages 5955-978, Sept. 2002.

[11] M. Chauda and P. T. Wood, "Towards an ODMG-Compliant Visual Object Query language," In Proc. Very Large Data Bases, pp. 456-465, 1997.

[12] M. Gyssens, et al., "A Graph-Oriented Object Model for Database End-User Interfaces," In *Proc. ACM SIGMOD Int'l Conf. on Management of Data*, pages 24-33, 1990.

[13] J. Kim et al., "VOQL: A Visual Object-oriented Database Query Language for Visualizing Path Expressions," *Journal of Computer Systems Science and Engineering*, 2000.

[14] A. J. Szabo et al., "Graphic User Interface for Database System," U. S. Patent, #5966126, Oct. 12, 1999

[15] M. Zloof, "Query-By-Example: A Data Base Language," *IBM Systems Journal*, Vol. 16, No. 4, pages 324-343, 1977.

[16] R. G. G. Cattell et al., *The Object Data Standard ODMG 3.0*, 2000.

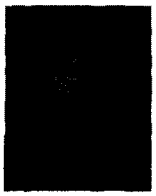
[17] Tachyon 사용자 메뉴얼, 한국전자통신연구원, 2001.

[18] 오디세우스/OOSQL Version 3.0, Reference Manual, 한국과학기술원 전산학과, 2000.

[19] W. Kim, *Introduction to Object-Oriented Databases*, MIT Press, 1990.

[20] M. Carey et al., "PESTO: An Integrated Query/Browser for Object Databases," In *Proc. Int'l Conf. on Very Large Data Bases*, pp. 203-214. 1996.

[21] T. Cataric, et al., "Visual Query Systems for Databases ; A Survey," *Journal of Visual Languages and Computing*, Vol. 8, pp. 215-260, 1997.



조 완 섭
1985 경북대학교 (학사). 1987 한국과학기술원 (석사). 1996 한국과학기술원 (박사, 전산학). 2002~2003 미국 프로리다대학 Post. Doc. 1987~1990 한국전자통신연구원. 1997~현재 충북대학교 경영정보학과 교수. 관심분야는 데이타베이스, Data Warehouse & OLAP, 데이타 마이닝, 바이오정보시스템