객체지향 데이타베이스 상에서 참조 구조의 시각화를 지원하는 시각적 검색 시스템

(A Visual Retrieval System supporting Visualization of Reference Structure on Object-Oriented Databases)

장 성 우 † 김 형 주 ††
(Sung-Woo Chang) (Hyoung-Joo Kim)

요 약 본 논문에서는 객체지향 데이타베이스 상에서 앵커 개념에 기반하여 전진 및 후진 항해 기능을 지원하는 시각적 객체 검색 모델을 제시한다. 본 모델 상에서 사용자는 관련성을 통해 연관되어 있는 두 클래스 상에서 항해의 기준으로 삼고 싶은 클래스에 앵커를 위치시킴으로써 관련성을 따라 전진 및 후진 항해를 자유롭게 수행할 수 있다. 이 기능에 기반하여 확장된 동기화 열람 기능도 지원된다. 또한, 항해 시 참조 객체들은 관련성의 카디널러티에 따라 단일 객체 노드와 군집 노드로 구분되어 표현된다. 이 시각화 검색 모델을 통해 사용자는 원하는 클래스를 기준점으로 하여 자유로이 전진 및 후진 참조 구조를 살펴볼 수 있게 된다.

Abstract In this paper, we present the visual retrieval model for object-oriented databases supporting forward and backward navigation using the concept of anchor. Users can freely navigate forward or backward on the relationship which associates two related classes in the database only by moving anchor to the class which users want to make as a base of navigation. Extended synchronized browsing is provided based on this mechanism. In addition, referenced objects are presented in a single object node or a collection node with respect to the cardinality(one or multiple) of relationship during navigation. With this visual retrieval model, users can move the point of their view and navigate the reference structure in forward and backward direction.

1. 서 론

최근 객체지향 데이타베이스가 많은 각광을 받고 있다. 객체지향 데이타베이스는 풍부한 세만틱(semantics), 예를 들면, 클래스(class), 상속성(inheritance), 관련성(relationship) 및 집합 속성(set attribute)을 표현할 수 있는 데이타 모델을 제공하여, 멀티미디어, CAD(Computer-Aided Design), GIS(Geographic Information System) 등의 응용 분야에 쉽게 적용될 수 있다[1]. 객체지향 데이타베이스에서 정보 저장 및 검색의 단위가 되는 것은 객체(object)이다[2]. 객체는 일반적으로 많은 관련성 정보를 포함하고 있다. 하지만, 객체들 사이에 다양한 관련성 정보들이 혼재하게 되면 정보 구조가 복잡하게 되어 사용자들의 이해에 어려움을 주는 요인이 된다. 때문에, 사용자의 정보 이해를 지원하기 위해서는 체계적인 시각화 검색 기법이 필요되어지게 된다.

1.1 배경

현재까지 다양한 객체 검색 시스템들이 개발되었다[7,8,9,10,11,12,13,14]. 이러한 관련 연구를 통해 객체 검색 기법에 있어 몇 가지 문제점들이 알려져 있는데, 이 중 중요한 것으로 반복적인 항해 문제(repetitious navigation problem)[3]와 화면 배치 문제(screen layout problem)[4] 등이 있다. 반복적인 항해 문제는 하나의 객체 집합에 속한 서로 다른 객체에 대해서 이들 객체가 참조하는 참조 경로를 매번 똑같이 사용자가 선택해서 항해해 나가야 하는 데에서 비롯된다. KIVIEW[7]와 OdeView[8] 시스템은 이 문제를 동기화 열람

· 본 연구는 통상 산업부의 공업 기반 기술 과제 943-20-4, "객체지향 데이타베이스를 위한 설계 도구의 구현"의 지원을 받아 작성된 것입니다.
† 비 회 원 : 서울대학교 컴퓨터공학과
†† 종신회원 : 서울대학교 컴퓨터공학과 교수
논문접수 : 1996년 8월 29일
심사완료 : 1997년 5월 28일

(synchronized browsing)이라고 하는 방법을 사용하여 해결하였다. 이 방법은 현재 출력된 객체가 참조하는 객체들을 자동적으로 출력하여 준다. 이를 통해 사용자는 같은 참조 경로를 매번 항해할 필요가 없게 된다. 화면 배치 문제는 화면 크기의 제약으로 인해 한 번에 많은 객체가 출력될 수 없는 데에서 기인한다. 이를 해결하기 위해 IconicBrowser 시스템[9]과 Rosengren et al.[10]은 아이콘에 기반한 열람 기법을 제공하였다. 아이콘은 객체의 그림 표현을 나타내는데, 작은 크기로 인해 한 번에 많은 객체들을 표현할 수 있게 된다. 하지만, 정교하거나 의미있는 정보는 보이지 못하는 단점을 가진다.

이 밖에 또 중요한 문제점은 대부분의 검색 시스템들이 전진 항해(forward navigation)만을 지원할 뿐, 후진 항해(backward navigation)는 지원하지 못한다는 점이다[5]. 즉, 어떤 객체가 관련성을 통해 참조하는 객체들의 집합만을 출력할 뿐, 특정 객체를 참조하는 객체들의 집합은 출력하지 못한다. 이로 인해 객체 검색시 사용자의 관점은 항상 고정되어지게 된다. 하지만, 사용자의 자유로운 객체 검색을 지원하기 위해서는 사용자 인터페이스 상에서 간단한 조작만으로 전진 및 후진 항해가 지원될 수 있어야 한다.

1.2 연구 목표

본 논문에서는 이러한 필요성에 기반하여 객체들 사이의 관련성을 통한 참조 관계를 사용자가 충분히 이해할 수 있도록 시각화하여 표현하여 주고, 전진 및 후진 항해 기능을 관련성을 따라 자유롭게 수행할 수 있도록 지원하여 주는 시각적 객체 검색 모델을 설계 및 개발하였다. 이를 위해 본 연구는 앞서 제시한 후진 항해 기능의 지원 미비의 문제점에 대한 해결책으로서 앵커에 기반한 객체 검색 기법을 제시한다. 또, 기존의 반복적인 항해 문제에 대한 해결책으로 제시된 동기화 열람 기능을 후진 항해에도 적용하여 확장된 동기화 열람 기능으로 발전시켰다. 그리고, 화면 배치 문제에 대한 해결책으로 제시된 아이콘 기반 표현 방법을 수용하였다.

본 시각적 객체 검색 모델은 다음과 같은 특징들을 제공한다 :

* 시각적 검색 구조 : 객체들 사이에 존재하는 관련성과 이를 통한 참조 객체의 다수성(단일 혹은 군집)을 사용자에게 명확히 표현하기 위한 시각화 구조를 제공한다. 이 시각화 구조 상에서 개별 객체들은 기본적으로 아이콘으로 표현되며, 단일 객체와 군집 객체는 각각 단일 객체 노드와 군집 노드로 구분되어 표시된다.

* 앵커를 이용한 전진 및 후진 항해 기능 : 관련성으로 연결된 두 클래스 상에서 관련성을 따라 사용자가 쉽게 전진 및 후진 항해를 실행할 수 있도록 도와준다. 전진 및 후진 항해의 선택은 앵커를 항해의 시작점 클래스에 위치시킴으로써 이루어진다. 사용자는 단지 앵커의 위치만을 옮김으로써 관련성을 통해 전진 및 후진 항해를 수행할 수 있게 된다. 이 때, 후진 항해는 역 관련성의 존재 여부와 상관없이 지원되어진다.

* 확장된 동기화 열람 기능 : 기존의 동기화 열람 기법을 확장한 확장된 동기화 열람(extended synchronized browsing) 기법을 제공한다. 즉, 기존의 동기화 열람 기법은 전진 항해를 통한 객체 검색시의 동기화 기능만을 제공하였지만, 확장된 동기화 열람 기법은 전진 및 후진 항해 모두를 통한 동기화 출력 기능을 제공한다.

본 논문의 나머지 부분에서는 객체 검색의 기본 개념과 시각적 객체 검색 모델에 대하여 설명한다. 또, 이를 구현한 프로토타입 시스템인 SOPView의 구조와 객체 검색 예제에 대해서 설명한다. SOPView 시스템은 객체 지향 데이타베이스 관리 시스템인 SOP(SNU QODB Platform)[1] 상에서 객체 검색을 위한 사용자 인터페이스로서 구현되었다. SOP는 ODMG- 93 표준안 Release 1.1[6]을 지원한다. 따라서, 본 논문에서 설명하는 객체 지향 데이타베이스의 기저 데이타 모델은 ODMG 객체 모델(object model)[6]이 된다. 하지만, 본 연구에서 제안한 기법들은 다른 데이타 모델을 지원하는 객체지향 데이타베이스 상에서도 활용될 수 있다.

1.3 논문의 구성

본 논문의 구성은 다음과 같다. 먼저, 2장에서는 본 연구의 개관으로서 데이타 모델 및 예제 스키마, 그리고, 주요 개념들의 정의 등에 대해서 설명한다. 3장에서는 본 연구가 제안하는 시각적 객체 검색 모델의 구성 요소들에 대해서 살펴 보기로 한다. 4장에서는 앵커 기법을 지원하기 위한 내부 메카니즘에 대해서 설명한다. 5장에서는 제안 모델을 구현한 프로토타입 시스템인 SOPView 시스템의 구조 및 실행 예제를 보인다. 6장에서는 객체 검색 기법을 지원하여 주는 관련 시스템들을 통해 기존의 관련 연구 상황에 대해 고찰하여 본다. 그런 후, 7장에서 결론을 맺기로 한다.

2. 객체 검색의 개관

본 장에서는 본 시각적 검색 모델의 기반 데이타 모델인 ODMG 객체 모델에 대해 간략히 설명한 후, 예제

1) SOP는 서울대학교 객체지향 연구실에서 1993년부터 1995년까지 3년간 개발하여 완성한 객체 지향 데이타베이스 관리 시스템입니다.

스키마를 보인다. 그런 후, 본 모델의 핵심이 되는 중요 개념들에 대한 정의를 내리기로 한다.

2.1 데이타 모델: ODMG

ODMG 표준안[6]의 객체 모델에 있어 가장 기본이 되는 개념은 객체이다. 객체는 실세계를 구성하는 요소들을 나타내며, 상태(state)와 행위(behavior)로 구성된다. 객체의 상태는 성질(property)들의 집합으로, 객체의 행위는 연산(operation)들의 집합으로 정의된다. 객체의 성질은 속성(attribute)과 관련성(relationship)으로 구성된다. 속성은 타입(type)과 리터럴(literal)의 쌍으로 나타내어지며, 객체의 정적인 속성을 나타내게 된다. 관련성은 두 개의 객체들 사이에 정의되어지며, 객체들 사이의 연관 관계를 나타내게 된다. 이러한 두 객체 간에는 항해할 수 있는 순회 경로(traversal path)가 존재하게 된다. 관련성은 'inverse' 키워드의 지정을 통해 역방향 경로(inverse path)가 제공된다. 즉, 한쪽 방향에서 해당 관련성에 대해 변경 및 삭제 등의 연산을 수행하면 이것이 반대편 객체들의 역 관련성에도 반영되어, 참조 무결성(referential integrity)이 보장받을 수 있게 된다. 객체의 연산은 객체가 실행할 수 있는 기능들을 나타낸다.

같은 성질 및 행위를 갖는 객체들은 타입(type)으로 무리지어진다. 타입에는 하나의 인터페이스(interface)와 하나 이상의 구현(implementation)이 존재할 수 있다. 타입들 간에는 상속 관계가 존재할 수 있으며, 이 관계들은 계층 구조(hierarchy)를 이룬다. 하나의 타입은 타입 성질로서 익스텐트(extent)와 키(key)를 가진다. 익스텐트는 그 타입에 속한 모든 객체들의 집합을 나타낸다. 키는 그 클래스에 속한 각각의 객체들을 유일하게 구별할 수 있는 속성을 나타낸다.

2.2 예제 스키마

그림 1은 ODMG 모델 상에서의 대학 스키마의 구조를 보이고 있다. 이 스키마에서 'Professor', 'Graduate', 'Student' 및 'Employee' 클래스는 대학의 구성원들을 모델링하고 있다. 'Person'과 'Univ-Person' 클래스는 이들의 수퍼클래스(superclass)이다. 'Department' 클래스는 대학 구성원들이 속한 학과를, 'Course' 클래스는 대학에서 개설된 강의를 모델링한다. 'Person' 클래스의 colleague 관련성은 자기 자신을 도메인으로 하는 재귀적(recursive) 관련성이다. 이 관련성은 같은 'Person' 클래스에 속한 다른 객체들의 집합을 참조한다. 이는 한 사람이 다른 여러 사람을 동료로 가질 수 있음을 나타낸다. 또, 'Univ-Person' 클래스의 dept 관련성은 'Department' 클래스를 도메인으로 가지는데, 이 관련성은 단지 하나의 객체만을 참조하는 구조이다. 즉, 한 대학 사람은 하나의 과에만 속함을 나타낸다. 이 두 관련성은 다른 속성들과 함께 모두 'Professor', 'Graduate', 'Student' 및 'Employee' 클래스들로 상속된다. 'Professor' 클래스의 객체들은 supervises 관련성을 통해 다수의 'Graduate' 클래스의 객체들을 참조하며, teaches 관련성을 통해 다수의 'Course' 객체들을 참조한다. 'Student' 클래스의 객체들은 enrolls 관련성을 통해 다수의 'Course' 객체들을 참조한다. 'Person' 클래스는 ssn 속성을, 'Department' 클래스는 code 속성을, 그리고, 'Course' 클래스는 code 속성을 키로서 갖는다. 특히, 'Person' 클래스의 키는 서브클래스(subclass)들로 상속되어지게 된다.

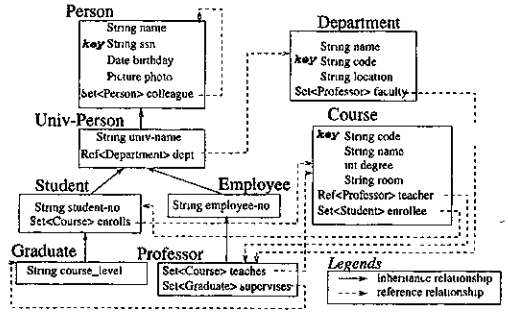


그림 1 예제 스키마 구조(대학 데이타베이스)

2.3 기본 정의

이 절에서는 본 논문을 통해 많이 사용될 주요 개념들에 대한 정의를 내리도록 한다.

정의 1 (단일 객체 노드)

객체 검색시 하나의 단일 객체로 구성된 시각적 참조 구조를 단일 객체 노드(single object node)라고 한다.□

단일 객체 노드는 관련성을 통해 하나의 객체만을 참조하게 됨을 나타낸다. 예를 들어, 앞의 스키마에서 'Univ-Person' 클래스의 객체가 dept 관련성을 통해 참조하는 'Department' 클래스의 객체는 단일 객체 노드로서 표현되어지게 된다.

정의 2 (군집 노드)

객체 검색시 객체들의 군집으로 구성된 시각적 참조 구조를 군집 노드(collection node)라고 한다. □

군집 노드는 관련성을 통해 참조되는 객체가 여러 객체들이 모여 이루어진 군집임을 나타낸다. 앞의 스키마에서 'Person' 클래스의 객체가 colleague 관련성을 통

해 참조하는 'Person' 클래스의 다른 객체들의 집합은 군집 노드로서 표현되어지게 된다.

정의 3 (전진 항해)

어떤 클래스 A가 다른 클래스 B를 도메인으로 하는 관련성 r을 가지고 있을 때, 클래스 A의 현재 객체 a가 관련성 r을 통해 참조하는 클래스 r의 객체 혹은 객체들의 집합을 검색하는 작업을 클래스 A로부터 클래스 B로의 전진 항해라고 한다. □

이 정의에서 전진 항해의 시작점이 되는 기준 클래스는 클래스 A이다.

정의 4 (후진 항해)

어떤 클래스 A가 다른 클래스 B를 도메인으로 하는 관련성 r을 가지고 있을 때, 관련성 r을 통해 클래스 B의 현재 객체 b를 참조하는 클래스 A의 객체 혹은 객체들의 집합을 검색하는 작업을 클래스 B로부터 클래스 A로의 후진 항해라고 한다. □

이 정의에서 후진 항해의 시작점이 되는 기준 클래스는 클래스 B이다.

정의 5 (앵커)

특정 클래스가 관련성을 통해 다른 클래스를 참조할 때, 이 관련성을 통한 전진 및 후진 항해의 시작점이 되는 기준 클래스를 지정하는 기호를 앵커(anchor)라고 한다. □

정의 6 (앵커 노드, 앵커 객체 및 앵커 클래스)

앵커가 위치한 노드를 앵커 노드(anchor node)라고 한다. 또, 앵커 노드 상의 현재 객체를 앵커 객체(anchor object)라고 한다. 그리고, 앵커 객체가 속한 클래스를 앵커 클래스(anchor class)라고 한다. □

3. 시각적 객체 검색 모델

본 장에서는 본 논문이 제안하는 시각적 검색 모델의 기본적인 시각화 구조와 이것의 실제 데이타베이스와의 관계에 대해서 설명한다.

3.1 시각화 구조

이 절에서는 시각화 구조를 구성하는 기본적인 시각화 요소들과 이들을 이용한 관련성 참조 구조의 표현 방법, 그리고, 이들 상에서의 객체 검색 기능에 대해서 설명하도록 한다.

3.1.1 객체의 표현

검색된 객체의 표현을 위한 기본적인 시각화 요소는 다음의 그림 2와 같다. 단일 객체 노드는 하나의 단일 객체에 대한 참조 구조를 나타내며, 하나의 객체를 나타내는 아이콘과 이 객체가 가지는 관련성들로 표현된다. 군집 노드는 객체들의 군집에 대한 참조 구조를 나타내는 단위로서, 아이콘의 집합과 관련성들로 표현된다. 또, 군집 노드는 군집의 타입에 따라 'Set'과 'List' 타입을 구별하여 표현하여 준다. 이 시각화 요소들을 이용하면 객체들 사이의 참조 계층 구조를 구성하는 각 참조 관련성이 단일 객체에 대한 참조인지 혹은 객체들의 군집에 대한 참조인지가, 또, 군집의 경우 어떤 군집 타입에 대한 참조인지가 명확히 구분되어지게 된다.

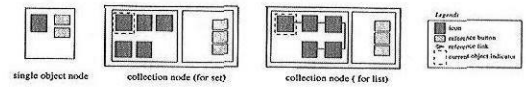


그림 2 객체 열람을 위한 기본적인 시각화 구조

객체를 나타내기 위한 아이콘으로는 클래스 상에 정의된 사진이나 그림 속성 중에서 하나가 선택될 수 있다. 아이콘 지정이 없을 경우에는 클래스의 키로 지정된 속성을 아이콘으로 나타낸다. 키 지정이 없을 경우, 사용자에게 하나의 속성을 반드시 선택할 것을 요구한다. 그러면, 지정된 속성 값이 출력되어지게 된다. 키 속성이나 사용자가 지정한 속성이 아이콘으로는 적합하지 않은 형태, 즉, 숫자나 문자열이면 속성값은 그냥 문자열의 형태로 화면에 출력된다.

3.1.2 앵커에 기반한 관련성의 표현

클래스 A로부터 클래스 B로 관련성이 설정되어 있을 때, 앵커에 기반한 관련성의 표현 방법은 각각 다음의 그림 3의 (a), (b)와 같다. 그림 3의 (a)에서, 클래스 A는 앵커 클래스이고, 항해의 기준 클래스가 된다. 따라서, 클래스의 A의 객체 집합 $a_1, a_2, \ldots, a_m$ 중에서 현재 객체(앵커 객체)가 a_1일 경우, 관련성 $r_1, r_2, \ldots, r_n$ 중에서 클래스 B를 도메인으로 하는 관련성 r_1을 선택하면

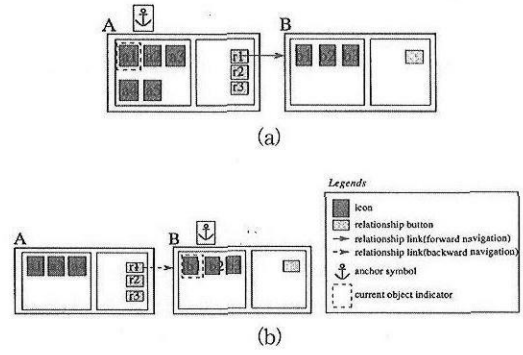


그림 3 관련성의 표현 방법(관련성의 카디날리티가 m 일 경우)

클래스 B에는 현재 객체 a_1이 관련성 r_1을 통해 참조하고 있는 객체들의 집합이 나타난다. 이것은 전진 항해의 표현이 된다. 이 전진 항해는 참조 관련성을 따라 계속적으로 진행될 수 있다.

이제, 앵커를 클래스 A에서 클래스 B로 옮겨 보자. 그러면, 그림 3의 (b)와 같이 클래스 B가 앵커 클래스가 된다. 이제, 객체 참조의 기준이 되는 앵커 객체는 클래스 B의 현재 객체 b_1이 된다. 따라서, 후진 항해가 수행되며, 클래스 A에는 b_1 객체를 참조하는 객체들의 집합들로 바뀌어 나타나게 된다. 이 때, 관련성 링크는 후진 항해를 의미하는 점선의 링크로 바뀌어 표현된다. 또한, 클래스 A 뿐만이 아니라 b_1 객체를 참조하는 다른 클래스의 객체들도 표현할 수 있다.

3.2 시각화 구조와 데이타베이스 사이의 관계

시각화 구조에 나타난 요소들과 실제 데이타베이스 객체들 사이의 관계는 그림 4와 같다.

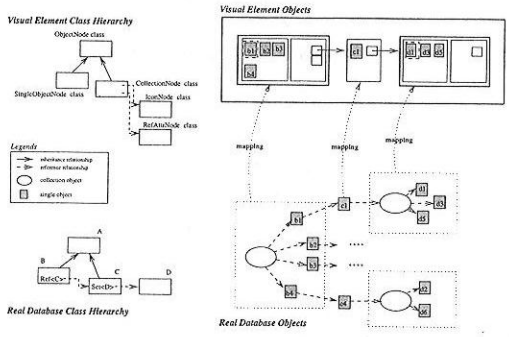


그림 4 시각화 구조와 데이타베이스 사이의 관계

실제 데이타베이스의 스키마는 클래스들의 계층 구조로서 구성된다. 클래스와 클래스 사이에는 두 가지 형태의 관계, 즉, 상속 관계와 참조 관계가 존재할 수 있다. 이 클래스들의 인스턴스로서 여러 객체들이 데이타베이스 상에 존재하게 되며, 이들은 시스템 전체를 통해 객체 식별자(object identifier:OID)를 통해 고유하게 식별되어진다. 객체들은 다른 객체들의 OID를 특정 속성 혹은 관련성의 값으로 가짐으로써 다른 객체를 참조하게 된다. 객체들의 군집을 나타내기 위해 여러 객체들의 OID를 포함하는 군집 객체가 존재하며, 객체들의 집합에 대한 참조는 이 군집 객체의 OID를 값으로 가짐으로써 가능하게 된다. 위의 그림에 실제 데이타베이스 객체들 사이의 참조 관계가 잘 나타나 있다.

본 시각화 시스템 역시 시각화 구조상의 객체 노드들을 모델링하기 위한 클래스 계층 구조를 가진다. 이 계층 구조는 크게 'SingleObjectNode' 클래스와 'CollectionNode' 클래스로 구성된다. 이 두 클래스는 각각 시각화 구조 상의 단일 객체 노드와 군집 노드에 해당한다. 이들은 모두 'ObjectNode' 클래스의 자식 클래스들인데, 이 클래스는 이 두 클래스의 공통 특징을 정의한 추상 클래스(abstract class)이다. 화면에 나타난 시각화 요소들은 시각화 클래스들의 객체들이며, 이들 객체들은 실제 데이타베이스 클래스의 인스턴스인 데이타베이스 객체들과 사상(mapping)되어 데이타베이스 객체들을 화면에 표현하여 주는 역할을 수행한다. 이 때, 데이타베이스 객체가 단일 객체이면 단일 객체 노드로, 군집 객체이면 군집 노드로 사상되어진다.

3.3 수행 기능

앞의 시각화 요소를 이용하여 객체를 검색하는 작업은 질의의 실행 결과로 리턴되어진 하나의 객체 혹은 여러 객체들을 포함하는 군집 객체의 화면 표현으로부터 시작된다(질의 실행 및 결과 객체 검색). 이 때, 질의 결과가 하나의 객체이면 단일 객체 노드를 통해, 객체들의 군집이면 군집 노드를 통해 표현된다. 이 객체 혹은 군집에 속한 객체들은 속성 a_1, a_2,..., a_m과 관련성 r_1, r_2,..., r_n을 갖는다. 한 객체가 가지는 속성값들은 객체 정보 노드를 통해 출력될 수 있다(객체 정보의 검색). 또, 항해를 원하는 관련성을 선택함으로써 현재 객체가 참조하는 객체를 살펴볼 수 있게 된다(관련성의 전진 항해). 이 때, 관련성이 참조하는 객체 역시 관련성의 카디날리티에 따라 단일 객체 노드 혹은 군집 노드의 형태로 표현되어지게 된다. 이 참조 객체 역시 속성들과 관련성들을 포함하게 되므로, 속성값들은 화면에 출력되고 관련성들은 계속적으로 참조되어질 수 있게 된다. 이 때, 만일 사용자가 앵커 기호를 결과 객체 집합에서 참조되어지는 객체 집합으로 옮긴다면, 이 참조 객체 집합 상의 현재 객체를 참조하는 객체들의 집합이 원래의 객체 집합 상에 출력되어지게 된다(관련성의 후진 항해).

4. 항해 지원 메카니즘

4.1 전진 및 후진 항해 지원 방법

본 연구가 기반으로 하고 있는 ODMG-93 표준안 Release 1.1[6]은 참조 무결성을 지원하기 위해 'inverse' 지정을 통한 역 관련성의 설정을 지원하고 있다. 하지만, 참조 무결성의 지원이 필요없는 경우에는 스키마 설계자가 역 관련성을 설정하지 않을 수도 있다.[2]

2) 최근에 공개된 ODMG-93 Release 2.0 draft에서는 역 관련성을 반드시 설정하도록 되어 있다.

본 연구에서의 전진 및 후진 항해는 기본적으로 관련성이 포함하는 OID에 의한 객체 검색을 통해 지원된다. 하지만, 역 관련성이 설정되어 있지 않은 경우에 대해서는 질의 작성 방법을 통해 후진 항해를 지원한다.

관련성을 통한 전진 항해의 경우

두 클래스 A, B가 클래스 A로부터 클래스 B로의 관련성 r을 통해 연관되어 있다고 하자. 이 때, 클래스 A가 앵커 클래스가 되어 클래스 A의 현재 객체 a가 참조하는 클래스 B의 모든 객체들의 집합을 검색하기 위해서는 관련성 r이 가지는 참조 객체(들)의 OID를 이용하여 해당 객체를 검색하여 클래스 B의 객체 노드에 출력하면 된다.

역 관련성을 통한 후진 항해의 경우

다음으로 역 관련성을 통한 후진 항해의 경우를 살펴보자. 두 클래스 A, B가 앞의 경우와 동일한 구조를 가지고 있고, 클래스 B로부터 클래스 A로의 역 관련성 r'가 정의되어 있다면, 클래스 B가 앵커 클래스가 되어 클래스 B의 현재 객체 b를 참조하는 클래스 A의 모든 객체들의 집합을 검색하기 위해서는 관련성 r'가 가지는 참조 객체(들)의 OID를 이용하여 해당 객체를 검색하여 클래스 A의 객체 노드에 출력하면 된다.

역 관련성이 설정되어 있지 않았을 때의 후진 항해의 경우

만약, 클래스 B로부터 클래스 A로의 역 관련성이 설정되어 있지 않다면, 클래스 B를 앵커 클래스로 하는 후진 항해를 지원하기 위한 OQL 질의어를 생성하여 수행한 후, 이 결과를 클래스 A의 객체 노드에 출력하면 된다. 해당 질의어는 다음과 같이 작성될 수 있다 :

```
select a                          select a
from a in A                       from a in A, b in a.r
where a.r.kb = key_value          where b.kb = key_value
```

(관련성 r의 카디날리티가 1인 경우) (관련성 r의 카디날리티가 m인 경우)

여기서, A, a는 각각 클래스 이름과 클래스 변수가 된다. 또, 위의 두 질의어의 where 절은 key_value를 키 속성 값으로 갖는 클래스 B의 객체를 관련성 r을 통해 참조하는 클래스 A의 객체를 지정하는 조건절이 된다. 여기서, 관련성의 카디날리티에 따라 다른 질의를 작성한 이유는 OQL 질의에서 처리시의 모호성(ambiguity) 때문에 경로 표현 상에 복수의 군집을 명세하는 것을 금지하고 있기 때문이다. 하지만, 이 방법은 앵커 클래스의 정의상에 키 속성이 지정되지 않은 경우에는 문제가 된다. 현재까지로는 특정 클래스에 속한 모든 객체에 대해서 유일한 하나의 객체를 찾아 이를 질의상에서 지정하는 방법으로는 키 속성이 유일한 방법이다. 물론, 데이타베이스 상에서 모든 객체를 유일하게 식별하기 위해 OID가 존재하지만, OID를 이용하여 질의상에서 객체를 지정할 수는 없다. 이는 OID는 시스템 내부의 구현 메카니즘이기 때문에 사용자가 이를 직접 이용하는 것은 객체지향 데이타베이스에서 제한하고 있기 때문이다. 따라서, 역 관련성이 설정되지 않는 상태에서의 후진 항해가 지원되기 위해서는 스키마에 속한 모든 클래스는 반드시 키 속성을 가지고 있어야만 한다.

4.2 앵커 지정 방법

관련성을 통한 전진 및 후진 항해의 선택은 앵커 기호를 나타내는 버튼을 'Drag & Drop'을 이용하여 항해의 기준점으로 삼고 싶은 노드 위로 옮겨 앵커 노드로서 지정함으로써 수행된다. 이제 앵커 객체가 가지는 관련성들 중에서 사용자가 참조 구조를 검색하기로 선택한 관련성에 대해서는 전진 항해가 수행된다. 이를 위해 앵커 클래스에 속한 관련성의 리스트를 유지하고, 이 리스트 상에 사용자의 검색 선택 여부를 기록하여 유지하면 된다. 그런 후, 사용자에 의해 선택된 관련성들에 대해서만 전진 항해를 실행한다. 반대로, 앵커 클래스를 도메인으로 하는 관련성들 중에서 사용자가 선택한 관련성에 대해서는 후진 항해가 수행된다. 이를 위해서는 앵커 클래스를 도메인으로 하는 관련성과 이 관련성이 속한 클래스의 이름을 갖는 구조체의 리스트를 유지하고, 이 리스트 상에 사용자의 검색 선택 여부를 기록하여 유지하면 된다. 그런 후, 사용자에 의해 선택된 관련성들에 대해서만 후진 항해를 실행한다.

4.3 성능 문제

이 절에서는 전진 및 후진 항해를 지원하는 각각의 방법의 수행에 따른 성능 문제에 대해 살펴 본다.

관련성을 통한 전진 항해와 역 관련성을 통한 후진 항해의 경우

이 경우 앵커 클래스에서 현재 객체를 찾아내어 이 객체가 가지는 관련성 혹은 역 관련성의 값, 즉, 현재 객체가 참조하는 객체들의 OID 값만을 리턴하면 되기 때문에 검색 연산의 복잡도는 O(1)이 된다.

역 관련성이 설정되어 있지 않았을 때의 후진 항해의 경우

객체지향 데이타 모델에서는 두 객체에 걸친 정보 검색이 필요할 때에는 관계형 모델과는 달리 무조건 조인(join) 연산을 수행하는 것이 아니라, 두 객체 사이에 관련성이 설정되어 있으면 이 관련성을 따라 항해하여 참조 객체를 알아내어 정보를 검색하는데 이를 묵시적(implicit) 조인이라 한다. 이 경우 작성된 질의어에서는

묵시적 조인 연산이 일어나게 된다. 클래스 A에 존재하는 모든 객체들에 대해서 이 객체가 관련성 r을 참조 경로로 하여 참조하는 클래스 B의 객체가 현재 객체 b인가를 검사하게 된다. 따라서, 참조 경로를 거치는 과정에서 묵시적 조인이 수행된다. 이 때, 클래스 A의 객체의 갯수를 n이라고 하고, 각각의 객체들이 관련성을 통해 참조하는 참조 객체들의 평균 갯수가 s라고 할 때, 질의 수행시의 복잡도는 O(ns)가 된다. 따라서, 역 관련성이 설정되지 않은 경우를 위해 질의를 실행하는 것은 시스템에 엄청난 부담을 주게 된다. 이는 위의 경우에서 클래스 A에 속한 모든 객체들을 검색해야 하기 때문인데, 이 클래스에 속한 객체들의 갯수가 엄청나게 많으면 시스템 성능에 절대적 영향을 미치게 된다. 따라서, 역 관련성이 없는 경우에는 후진 항해에 앞서 사용자에게 이를 주지시키는 메카니즘이 지원되어야 한다. 또한, 이러한 질의의 횟수가 많을 경우에는 이를 시스템 설계자에게 알리고, 스키마 진화를 통해 새로이 역 관련성을 설정할 수 있도록 하여야 할 것이다. 이러한 방법에 대해서는 차후 구현할 계획이다.

5. 프로토타입 시스템 : SOPView

본 장에서는 앞서 설명한 시각화 객체 검색 기능을 구현한 프로토타입 시스템인 SOPView에 대하여 설명한다. SOPView 시스템은 객체지향 데이타베이스 관리 시스템인 SOP 상에서의 시각적 질의 및 객체 검색을 실행할 수 있도록 도와 주는 GUI(Graphical User Interface) 프로그램이다[15]. 본 장에서는 [15]에 기술된 내용 이외에 본 연구에 의해 새로이 추가된 앵커 기반의 전진 및 후진 항해 기법에 대하여서만 설명하기로 한다. 먼저 시스템의 구조에 대해서 설명한 후, 앵커 기반 객체 검색의 예를 데모 실행을 통해 보이기로 한다.

5.1 시스템 구조

SOPView 시스템 중 시각적 객체 검색 모듈의 구조는 다음의 그림 5와 같다. 객체 검색기(object retriever)는 사용자의 질의 실행 및 참조 객체 항해에 의해 검색되어진 객체를 데이타베이스로부터 가져오는 기능을 수행한다. 질의 생성기(query generator)는 전진 및 후진 항해의 지원을 위해 질의어를 생성하여 질의 처리기로 넘겨 주는 기능을 수행한다. 객체 시각화 도구(object visualizer)는 검색되어진 객체들 사이의 관계를 시각화 구조를 통해 사용자에게 출력하여 주고, 또한, 객체 검색과 관련된 사용자의 입력을 처리하는 기능을 수행한다.

객체 관리자 및 질의 처리기는 모두 SOP OODBMS의 요소들로서, 객체 관리자(object manager)는 객체의 생성 및 삭제, 수정 등 객체 저장 및 검색과 관련된 모든 작업을, 그리고, 질의 처리기(query processor)는 입력된 질의의 최적화 및 실행 기능을 수행한다.

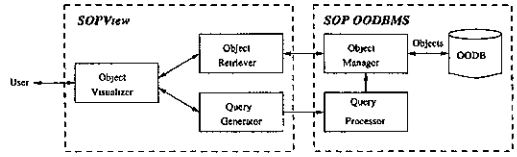


그림 5 객체 검색 시스템의 구조

본 시스템을 구현하기 위해 기반 환경으로는 Unix와 X 윈도우 시스템을, 프로그래밍 언어로는 GNU C++[16]를, 그래픽 사용자 인터페이스 라이브러리로는 Motif[17]를 사용하였다. 시스템은 현재 약 22,000 라인 정도의 C++ 코드로 구성되어 있다. 이 중에서 본 논문에서 제시한 앵커 개념과 전진 및 후진 항해, 확장된 동기화 기능을 구현한 코드는 약 7,000 라인 정도가 된다.

5.2 객체 검색 예제

이 장에서는 객체 검색의 실제 수행 예를 앞서의 그림 1의 예제 데이타베이스 상에서 살펴보기로 한다.

5.2.1 질의 실행 및 전진 항해를 통한 객체 검색

객체 검색의 첫단계는 질의의 실행이 된다. ODMG 모델의 질의어인 OQL[6]은 객체 이름을 이용한 객체 검색과 조건 탐색에 의한 객체 검색을 모두 지원하여 준다. 이제 이 데이타베이스에 대해 "컴퓨터공학과에 속한 모든 교수들을 보여라" 라고 하는 질의를 수행하기로 하자. 이 질의는 OQL로 다음과 같이 표현되며, 질의 다이어로그를 통해 입력되거나 시각 질의[15]를 통해 명세될 수 있다 :

```
select x
from x in Professor
where x.dept.name = 'Computer Engineering'
```

이 질의의 수행 결과로 컴퓨터공학과 교수들을 나타내는 객체들이 화면에 출력된다. 이 결과 객체들에 대해서 이들이 지도하는 대학원생들이 누구인지를 살펴 보기 위해서는 'Professor'클래스의 supervise 관련성을 따라 전진 항해를 수행하면 된다. 그러면, 교수들이 지도하는 대학원생을 나타내는 객체들이 화면에 출력되어진다.

또, 이 대학원생들의 동료들로는 누가 있는지 살펴보고자 하면 'Graduate' 클래스의 colleague 관련성(이 관련성은 'Person' 클래스로부터 상속받은 것임)을 따라 전진 항해를 수행하면 된다. 이와 같이 질의 수행 결과에 대해 두 번의 전진 항해를 실행한 후의 결과는 다음의 그림 6과 같다.

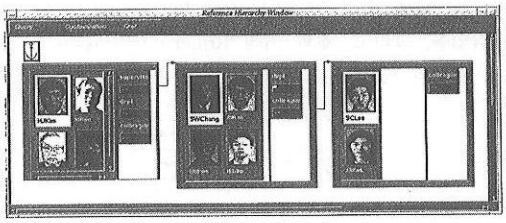


그림 6 전진 항해를 통한 참조 객체 검색의 결과

5.2.2 후진 항해를 통한 객체 검색

이제 대학원생 객체 중에서 'SWChang' 객체를 지도하는 교수로는 누가 있는지를 살펴 보자. 이를 위해서는 'Professor' 클래스에 위치한 앵커를 'Graduate' 클래스로 옮기기만 하면 된다. 이와 같이 후진 항해를 실행한 후의 결과는 다음의 그림 7과 같다.

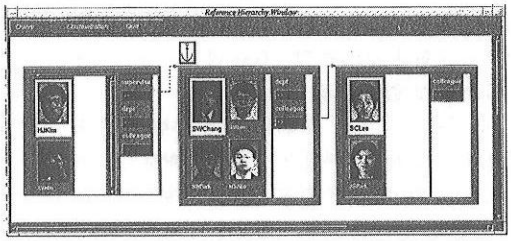


그림 7 후진 항해를 통한 참조 객체 검색의 결과

5.2.3 객체 정보의 검색

이제 교수 객체들 중에서 'HJK' 객체와 대학원생 객체 중에서 'SWChang' 객체의 내용을 살펴 보자. 이들 객체의 내용을 살펴보기 위해서는 해당 아이콘 버튼을 더블 클릭하면 된다. 이 결과 화면은 다음의 그림 8과 같다.

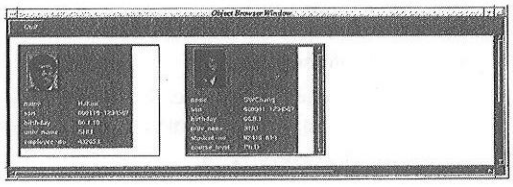


그림 8 객체 내용의 검색

5.2.4 확장된 동기화 열람

이제 대학원생 객체 중에서 현재 객체를 'SWChang' 객체에서 'SWLee' 객체로 바꾸어 보자. 그러면, 전진 항해를 수행하고 있는 'Person' 클래스에는 'SWLee'의 동료 객체들이, 후진 항해를 수행하고 있는 'Professor' 클래스에는 'SWLee'를 지도하는 교수 객체들이 화면에 출력되어진다. 이는 기존의 전진 항해를 통한 동기화 열람을 후진 항해로까지 확장한 확장된 동기화 열람 기능이다. 확장된 동기화 열람 기능을 통해 사용자가 전진 항해 및 후진 항해시 같은 경로를 따라 반복적 항해를 수행할 때 일일이 참조 객체를 지정하는 문제를 해결하게 된다. 다음의 그림 9는 확장된 동기화 열람의 실행 결과를 보이고 있다.

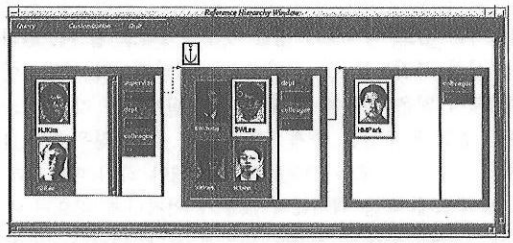


그림 9 확장된 동기화 열람의 수행 결과

5.2.5 후진 항해 경로의 추가

이제 대학원생 객체 집합에 대해서 후진 항해 경로를 추가해 보자. 그림 9 상에서 대학원생들을 수강자로서 가지는 수업 객체들을 살펴 보자. 이를 위해서는 '주문(Customization)' 메뉴 상에서 '후진 항해 경로 추가' 부메뉴를 선택한다. 그러면, 현재 앵커 클래스인 'Graduate' 클래스를 참조하고 있는 클래스와 참조 관련성의 리스트가 출력된다. 이 중에서 'Course' 클래스의 'enrollee' 참조 관련성을 선택한다. 그러면, 'SWLee' 객체를 수강자로 갖고 있는 수업들, 즉, 'SWLee' 객체가 수강하고 있는 수업 객체들의 군집 노드가 다음의 그림 10과 같이 나타난다.

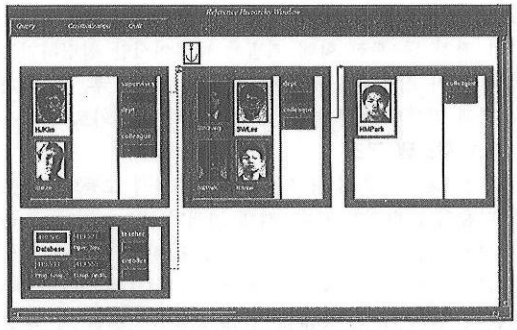


그림 10 새로운 후진 항해 경로 추가 예제

6. 관련 연구

현재까지 많은 시스템들이 객체지향 데이타 모델 상에서의 객체 검색을 위해 구현되었다.

KIVIEW 시스템[7]은 반복적인 항해 검색을 피할 수 있도록 동기화 열람 기법을 처음 제공했다. 이 시스템은

클래스에 속한 객체들의 열람, 이 객체들로부터 참조되어지는 참조 객체들에 대한 열람 및 동기화 열람 기능을 제공한다. 하지만, KIVIEW에서 참조 객체는 기존 객체의 내부에서만 표현되도록 구성되어 있어, 참조 객체의 표현에 제한이 가해지는 단점을 가지고 있다.

본 시스템에 많은 영향을 준 시스템은 OdeView[8]이다. OdeView는 Ode OODBMS를 위한 사용자 인터페이스로서 스키마 열람, 데이타베이스 검색 및 시각 질의 기능을 제공한다. 특히, KIVIEW의 동기화 열람 기법이 가지는 제한된 참조 객체 표현의 단점을 극복하여, 참조 객체를 별도의 윈도우로서 표현하여 주고 있으며, 이 기법의 고찰이 본 연구의 태동에 영향을 주었다. 하지만, 참조 객체가 군집으로 구성되어있을 경우를 위한 시각화 방법이 없는 단점을 가진다.

IconicBrowser 시스템[9]은 객체 지향 데이타베이스를 구성하는 클래스 및 객체들을 아이콘으로 표시하고, 이 아이콘을 이용하여 시각 질의 및 객체 열람을 수행할 수 있도록 지원하여 준다. 질의의 결과 역시 아이콘으로 표현되어 지지만, 질의 결과 객체들로부터 다른 참조 객체를 살펴볼 수 있는 방법은 제시하지 못하고 있다.

MoodView 시스템[11]은 MOOD OODBMS를 위한 GUI이다. 사용자는 MoodView를 통해 클래스 정의 및 변경, 객체 열람, 객체 관리 및 시각 질의를 수행할 수 있다. 객체들은 SID(Single Instance Display) 및 MID(Multiple Instance Display)의 두 가지 모드로 표현될 수 있다. 참조 정보도 표현되어지지만 동기화 열람 기능은 제공되지 않고 있다.

GOMI 시스템[12]은 객체지향 데이타베이스 시스템을 위한 GUI 프로그램이다. 이 시스템은 객체의 정의, 스키마 구조에 기반한 표현 및 객체 관리 기능과 시각 질의를 제공한다. GOMI 시스템에서 객체들은 스키마 구조와 함께 표현되어지며, 참조 관계 및 참조 객체 역시 표현되어진다. 하지만, 동기화 열람 기능은 제공되지 않는다.

SUPER 시스템[13]은 ERC+ 데이타모델을 위한 GUI이다. 이 시스템은 두 가지 형태의 객체 표현 모드를 가지는데, 하나는 형태 기반 모드이고, 다른 하나는 그래픽 모드이다. 형태 기반 모드에서 객체 및 관련성은 서식 구조로서 표현된다. 그래픽 모드에서 객체들은 스키마 형태의 그래픽 상에 개별 객체들의 집합으로서 표현되어진다. 이 시스템은 그래픽 모드에서 참조 관계가 표현되어지며, 동기화 열람 기능을 제공한다.

COMIB[14]에서는 복합 아이콘 브라우징 기법을 제안하고 있다. 이 시스템은 질의 수행 결과 검색되어진 여러 객체들을 아이콘을 이용하여 동시에 표현하여 준다. 그리고, 이 결과 객체들이 참조하는 참조 경로 상의 참조 객체에 속한 속성을 선택하여 결과 객체 아이콘에 함께 출력하여 줌으로써 여러 결과 객체들을 동시에 보면서 참조 속성도 함께 비교할 수 있도록 하여 주는 특징을 가진다. 하지만, 질의 결과 객체들에서의 참조 속성 비교만이 가능하고, 참조 객체가 집합일 경우를 위한 처리 방법이 없는 등의 단점을 가진다.

본 연구와 기존 연구와의 비교는 다음의 표 1에 나와 있다.

7. 결론

본 연구에서는 객체지향 데이타베이스 상에서의 시각적 객체 검색 모델을 구성하는 두 가지 형태의 기법, 즉,참조 관련성의 카디날리티에 기반한 객체 표현 방법과 앵커를 이용한 전진 및 후진 항해 기법을 제시하였다. 관련성의 카디날리티에 기반한 객체 표현 방법을 통해 데이타베이스에 있는 모든 객체들은 관련성을 통한 객체 참조시 그 관련성이 가지는 카디날리티(하나 혹은 다수)에 따라 각각 단일 객체 노드와 군집 노드를 통해 구분되어 표현된다. 이를 통해 사용자는 객체들 사이에 존재하는 카디날리티 세만틱을 쉽게 알 수 있게 되었다. 또한, 앵커를 이용한 전진 및 후진 항해 기법을 통해 사용자는 하나의 관련성과 연관되어 있는 두 클래스 상에서 단지 앵커를 움직임을 통해서 전진 항해 및 후진 항해를 수행할 수 있었다. 이를 통해 사용자는 원하는 클

표 1 기존 연구와의 비교표

	SOPView	KIVIEW	IB	OdeView	MoodView	SMARTIE	SUPER	GOMI	COMIB
군집 표현	O	△	O	X	O	O	O	△	△
객체 정보의 출력	O	O	O	O	O	O	O	O	O
관련성의 전진 항해	O	O	X	△	△	O	O	O	△
관련성의 후진 항해	O	X	X	X	X	X	X	X	X
동기화 열람	O	O	X	O	X	X	O	X	X

O : 지원, X : 지원 않음, △ : 부분적으로 지원, IB: IconicBrowser

래스를 기준점으로 하여 자유로이 전진 및 후진 참조 구조를 살펴볼 수 있게 되었다. 이 기능에 기반하여 확장된 동기화 열람 기능도 지원하였다. 기존의 동기화 열람 기법은 전진 항해를 통한 동기화만을 제공하였지만, 확장된 동기화 열람 기법은 전진 및 후진 항해 모두를 통한 동기화 출력 기능을 제공하였다. 그리고, 이러한 검색 기법을 설계 및 구현한 프로토타입 시스템인 SOPView 시스템에 구조 및 검색 예제의 데모를 보였다.

하지만, 이러한 특징들에도 불구하고 본 시스템은 다음과 같은 몇 가지 미비점을 가지고 있다. 첫째, 사용자가 자유로이 앵커를 이동하여 전진 및 후진 항해를 복잡하게 실행할 경우 발생할 수 있는 미아 문제(getting lost problem)[18]에 대한 지원 방법,. 즉, 사용자가 현재에까지 이르게 된 과정을 알려주고, 사용자가 원할 경우 원래의 상태로 복구할 수 있도록 도와주는 기능이 마련되어 있지 않다. 둘째, 현재에는 질의의 실행 결과가 특정 클래스에 속한 객체들을 리턴하는 경우만으로 한정하고 있다. 여러 클래스를 조인(join)하여 이들 클래스에 속한 속성들을 조합하여 생성된 새로운 타입의 객체에 대한 검색 기능은 제공되지 않는다. 이러한 기능들은 차후 계속적으로 연구하여 지속적으로 보완할 계획이다.

이러한 미비점의 보완 이외에도 향후 연구 계획으로 대규모 참조 구조의 효율적 지원을 위한 시스템 성능 향상에 대한 연구를 수행할 계획이다.

참 고 문 헌

[1] Won Kim, *Introduction to Object-Oriented Database*, p.10, The MIT Press, Cambridge, Massachusetts, 1990

[2] R. G. G. Cattell, *Object Data Management*, p.94, Addison-Wesley, 1992

[3] C. Batini *et al.*, "Visual Strategies for Querying Database," *Proc. of 1991 IEEE Workshop on Visual Languages*, pp.183-189, 1991

[4] Ben Shneiderman, *Designing the User Interface*, 2nd Ed., Addison-Wesley, 1992

[5] D. Lucarella and A. Zanzi, "A Visual Retrieval Environment for Hypermedia Information Systems," *ACM Transaction of Information System*, Vol. 14, No. 1, January 1996, pp.3-29

[6] R. G. G. Cattell, *The Object Database Standard: ODMG-93, Release 1.1*, Morgan Kaufman Publishers, San Mateo, California, 1994

[7] A. Motro, A. D'Atri, and L. Tarantino, "The Design of KIVIEW: an Object-Oriented Browser," *Proc. of 2nd Int'l Conf. on Expert Database Systems*, pp. 17-31, 1988

[8] R. Agrawal, N.H. Gehani, and J. Srinivasan, "OdeView: The Graphical Interface to Ode," *Proc. of Int'l Conf. on Management of Data*, San Diego, California, 1990

[9] K. Tsuda, M. Hirakawa, and T. Ichikawa, "IconicBrowser: an Iconic Retrieval System for Object-Oriented Databases," *Journal of Visual Languages and Computing*, Vol.1, No.1, pp.59-76, 1990

[10] P. Rosengren et al., "A Tools-Oriented Visual Interface for Multimedia Databases," *Proc. of Int'l Symp. on Next Generation Database Systems and Their Applications*, pp.219-226, September, 1993

[11] Ismailcem Budak Arpinar, Asuman Dogac, and Cem Evrendilek, "MoodView: An Advanced Graphical User Interface for OODBMSs," SIGMOD RECORD, Vol.22, No.4, Dec., 1993

[12] Y. Jun and S. Yoo, "GOMI: A Graphical User Interface for Object-Oriented Databases," *Proc. of Int'l Conf. on Object-Oriented Information Systems*, 1994

[13] A. Auddino, E. Amiel, Y. Dennebouy, Y. Dupont, E. Fontana, S. Spaccapierta, and Z. Tari, "Database Visual Environments Based on Advanced Data Models," *Proc. of Work. on Advanced Visual Interfaces*, pp. 156-172, 1992

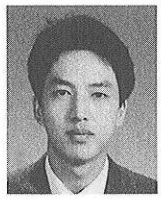
[14] J. Cha and S. Lee, "Browsing Multimedia Objects via Composite Icons," *Proc. of IEEE Conf. on Systems, Man, and Cybernetics*, Vancouver, Canada, 1995

[15] S-W. Chang, S. Lee, and H-J. Kim, "SOPView: A Visual Query and Object Browsing Environment for SOP OODBMS," *Proc. of 20th Int'l Computer Software and Application Conf.*, pp.354-360, 1996

[16] Bjarne Stroustrup, *The C++ Programming Language*, 2nd Ed., Addison Wesley, 1992

[17] Open Software Foundation, *OSF/Motif Programmer's Guide*, Prentice Hall, 1993

[18] R. Akscyn, D. McDracken, and E. Yoder, "KMS: A Distributed Hypermedia System for Managing Knowledge in Organizations," *Proc. of Hypertext 87*, ACM Press, Baltimore, 1987



장 성 우

1990년 2월 서강대학교 전자계산학과 학사. 1992년 2월 서울대학교 컴퓨터공학과 석사. 1992년 3월 ~ 현재 서울대학교 컴퓨터공학과 박사과정 재학중. 관심분야는 객체지향 시스템, 데이타베이스, 인간-컴퓨터 인터페이스.

김 형 주

제 3 권 제 2 호(C) 참조