객체지향 질의처리를 위한 객체관리기 인터페이스

이 연 식† 전 병 실†† 류 근 호†††

요 약

현실세계의 복잡한 데이타모델을 표현하고 관리하는 객체지향 데이타베이스 관리시스템에서는 모든 객체들에 대한 접근과 조작이 객체관리기에 의해 처리된다. 본 논문에서는 객체지향 질의처리를 위한 객체관리기의 호출함수와 의미를 규정하는 객체관리기 인터페이스의 설계 원칙을 제안하고, 이에 따라 객체관리기 인터페이스를 구현한다. 구현된 객체관리기 인터페이스는 질의처리부와 객체관리부를 서로 독립적으로 개발할 수 있는 환경을 제공할 뿐만 아니라, 다양한 응용의 적용을 가능하게 하며, 사용자에게 효율적 접근 방법을 제공한다.

An Object Manager Interface for Object-Oriented Query Processing

Yon Sik Lee,† Byoung Sil Chon†† and Keun Ho Ryu†††

ABSTRACT

Object-oriented database systems represent the complex data model of real-world and manage the complex data. An object manager handles the manipulation and access of all objects in object-oriented database management systems. In this paper, we not only propose the design rules of an object manager interface(OMI) which is specified the calling function and the semantics of an object manager for object-oriented query processing, but also implement the OMI based on the rules. The OMI implemented supports the independently developing environment of query processing and object management modules, and can be applicable to various applications, and also provides the efficient access method to users.

1. 서 론

객체지향 시스템에 대한 기법은 최근 10여년간 많은 발전을 가져왔는데, 이것은 기존 데이타베이스의 단점인 의미적 속성 표현의 한계성, 기능성의 부족, 응용 프로그램에서 데이타의 직접 검색 제한 등의 많은 문제의 해결과 함께 CAD/CAM, OIS 등과 같이 대단히 다양하면서 크고 복잡한 현실 세계의 정보를 처리할 수 있는 데이타모델을 지원하고 있다. 객체지향 데이타모델에서는 실세계의 모든 개념적 개체(정보, 연산자 및 제약 조건 등)를 하나의 객체로 간주하고 같은 성격을 갖는 객체들을 묶어 하나의 클래스를 형성한다. 객체는 속성과 속성-영역 값으로 구성되는 상태 표현 부분과 객체의 행위를 규정하는 메소드로 구성된다. 객체는 원시 객체와 튜플과 집합에 의해 재귀적으로 구성되는 복합 객체로 구분된다[6, 7]. 각 객체의 참조는 객체 식별자에 의해 이루어지며, 객체의 내부 구현 상태는 은닉되어 있고 메세지에 의해서만 반응한다. 클래스는 각 객체들의 중복되는 기술을 방지하여 저장의 효율성을 증가시킨다. 클래스 계층 구조상의 각 클래스는 IS-A 관계로써, 상위 클래스에서 기

†정 회 원 : 군산대학교 컴퓨터과학과 교수
††정 회 원 : 전북대학교 전자공학과 교수
†††종신회원 : 충북대학교 컴퓨터과학과 교수
논문접수 : 1994년 6월 20일, 심사완료 : 1995년 1월 7일

술된 상태와 메소드들은 하위 클래스로 계승되며, 하위 클래스에서는 계승받은 특징들 외에 자신만의 특징들이 추가된다[4, 7].

객체지향 데이타베이스 시스템은 객체지향 데이타모델의 특징들을 지원해야하고, 이 밖에도 디스크 관리, 데이타의 무결성, 동시성 제어, 보안, 확장성 및 질의 언어 등을 포함해야 한다[1]. 1980년대 초에 시작된 객체지향 데이타베이스 분야는 최근 연구가 급속히 발전되어, 데이타베이스의 기본 기술인 동시성 제어, 회복 기법, 저장 기술, 질의 처리 및 최적화 등 기존의 데이타베이스와 프로그래밍 언어를 그대로 재사용하게 되었을 뿐만 아니라, 새로운 데이타모델, 질의 언어, 색인 기법, 클래스 수정, 사용자 인터페이스, 성능 기준, 클라이언트/서버 구조, 메모리 상주객체관리와 같은 새로운 객체지향 데이타베이스를 위한 기법들이 개발되었다. 클라이언트/서버 구조로 이루어진 객체지향 데이타베이스 시스템은 클라이언트 부분을 클래스 정의처리부, 응용 프로그램 처리부 및 질의 처리부와 이들에게 객체관리기 인터페이스를 통하여 제공되는 객체를 관리하는 객체관리부로 구분된다[9, 12, 13, 15].

객체지향 데이타베이스는 클래스 정의에 의해서 설계되며, 이러한 객체지향 데이타베이스 설계와 설계된 데이타베이스의 모든 객체에 대한 접근 및 관리가 객체관리기를 통하여 이루어진다. 사용자는 응용 프로그램과 질의어를 통해서 데이타베이스에 접근하게 되는데[4, 6, 7], 이때 객체관리기는 하부저장구조나 서버로 연결하여 데이타에 접근한다. 객체관리기는 객체지향 테이타베이스의 주체가 되는 객체를 효율적으로 저장하고 관리하는 주요 부분으로써, 개념적으로 추상화된 객체 모델을 위한 객체지향의 기본적인 특성들을 지원하고[7, 13], 물리적인 하부 저장구조를 이용하여 객체에 접근하며, 내부적으로 객체의 저장을 제어 할 때 유사한 객체를 클래스터링하여 색인하므로써 저장과 접근을 보다 효과적으로 할 수 있도록 한다[2, 8, 16].

객체관리기와 그 접속에 관한 연구로서 ANNEVELINK와 DEWILDE[1]는 추상자료형을 이용하여 응용 프로그래머에게 투명한 객체관리를 제공할 수 있는 객체관리기를 설계하였고, STRAW, MELLENDER와 RIEGEL[10]은 smalltalk 언어 시스템과 데이타베이스 시스템의 결합시 발생하는 불일치(impedance mismatch)를 해결하여 응용 프로그래머가 두 시스템의 객체들에 대한 접근을 동일한 의미(semantics)로 수행할 수 있는 객체관리기에 관한 연구를 수행하였다. 그러나 이 연구들은 프로그래머에게 투명한 객체관리를 제공하여 객체지향 시스템들의 구축을 용이하게는 하지만, 자신의 시스템만을 지원하는 질의처리부와 객체관리부의 종속성에 의하여 다양한 객체지향 시스템의 응용을 지원하지 못한다. 그러므로, 보다나은 객체관리시스템 개발환경 및 다양한 응용에의 적용을 위하여, 질의처리부와 객체관리부를 독립적으로 유지하며 이들을 접속시키는 객체관리기 인터페이스의 구현이 필요하다. 따라서 본 논문은 질의 처리부와 객체관리부가 독립된 객체지향의 일반화된 개념을 지원할 수 있는 객체 시스템은 물론, 응용 프로그램에서도 쉽게 접근할 수 있는 사용자를 위한 상부 구조인 객체관리 시스템의 인터페이스를 설계 구현하는데 그 목적이 있다.

본 논문에서는 구현된 객체관리기 인터페이스에 의하여 실제 객체지향 질의가 처리됨을 보이기 위해, 질의어는 기존 관계 데이타베이스의 표준 질의어인 SQL을 객체지향 특성을 수용하도록 확장하여 이용하고, 객체관리기는 객체의 물리적인 하부구조의 접근을 위해서 Mneme 객체 저장관리기[18]를 객체지향 데이타베이스 시스템에 적합하도록 확장하여 이용한다.

연구의 수행 과정을 순조롭게 설명하기 위하여 본 논문의 서론에서는 전체적인 개요와 연구 수행 목적 등을 설명하고, 제 2 장에서는 객체관리기와 인터페이스 설계 원칙에 대해서 기술한다. 제 3 장에서는 인터페이스 설계 원칙에 따라 객체관리기 인터페이스를 구현하고, 제4장에서는 객체지향 질의 처리과정을 설명하고 설계된 인터

페이스를 이용하여 질의가 수행되는 과정을 보인다. 마지막 제 5 장에서는 현재 수행된 연구를 마무리하며 결론을 맺는다.

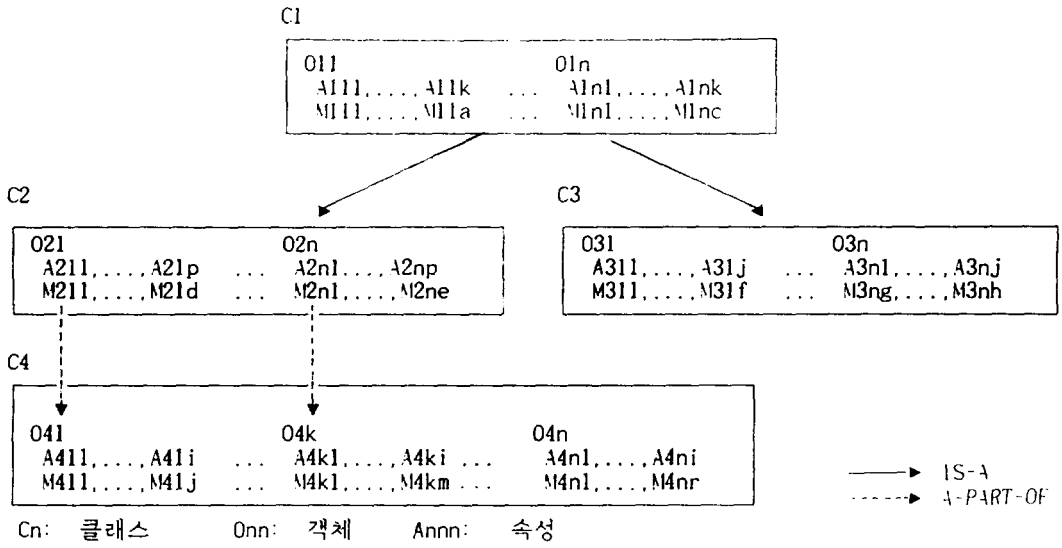
2. 객체관리기 인터페이스

2.1 객체관리기 개요

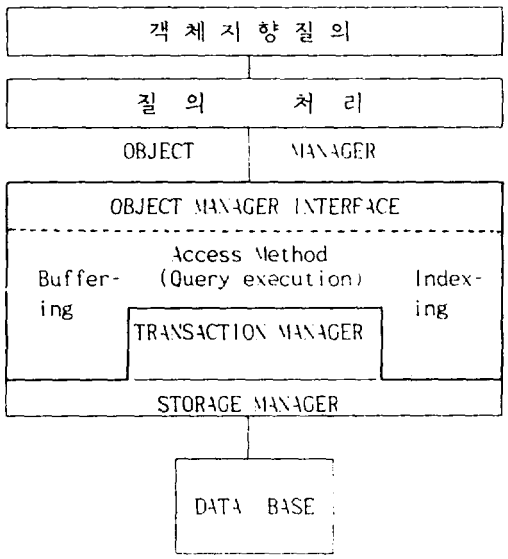
객체관리기는 객체지향 데이타모델을 수용하는 클래스에 따라 생성되는 객체들을 관리하므로 객체지향의 기본 개념과 데이타베이스의 특성을 지원해야 한다. 따라서, 본 논문에서는 일반화된 객체지향 개념들을 지원하도록 제안[5, 7, 13]된 데이타모델을 기반으로 한다. (그림 1)은 클래스의 계층과 객체 사이의 관계를 보인 것으로, 클래스(Cn)는 객체(Onn)의 상태부분(Annn)과 객체의 행위를 규정하는 메소드(Mnnn)로 구성됨을 보여주며, 상위 클래스의 특성에 대한 상속을 표현((그림 1)의 → : IS-A)하고, 속성이 원자 객체가 아닌 다른 객체를 참조할 수 있음을 표현((그림 1)의 ⇢ : A-PART-OF)하고 있다. 이와 같은 내용을 포함하는 클래스는 객체를 물리적 하부구조에 저장하기 위해 필요한 정보를 생성할 뿐만 아니라, 객체관리기 인터페이스를 이용한 질의 수행의 정보로 이용된다.

데이타베이스 관리 시스템은 질의 처리기,버퍼링, 색인 및 접근을 위한 객체관리기, 복구와 일관성을 유지하기 위한 트랜잭션 관리기와 효율적인 저장 관리를 위한 저장 관리기 등으로 구성되는데, (그림 2)의 Object Manager는 데이타베이스 생성과 운영에 관한 전반적인 사항를 규정하는 객체관리기로써, 클래스 생성, 삭제, 갱신, 클래스를 이용한 객체의 생성, 삭제 및 갱신과 함께 객체를 검색하는 임무를 수행한다. 또한, 객체관리기는 객체 식별자를 통하여 객체에 메세지를 전송하므로써 객체를 운영한다. 객체관리기는 이러한 기능을 수행하는 물리적 객체 테이블, 클래스 계승에 관한 정보, 객체의 집합에 관한 정보와 객체 사이의 관계를 규정하는 정보 등을 제공하고 관리한다.

객체관리기는 영속적으로 저장된 객체를 관리함에 있어서 저장 관리기를 이용하여 더이상 참조되지 않는 객체를 수집하고, 객체 접근 효율의 향상을 위해서 객체의 집합에 근거하여 객체를 색인하는 클러스터링 전략을 구현하며, 트랜잭션



(그림 1) 클래스 계층과 객체
(Fig. 1) Class-hierarchy and objects



(그림 2) 객체관리기
(Fig. 2) Object manager

단위 등을 제공한다. 객체는 영속적으로 저장되기 위한 물리적인 식별자가 요구되며, 객체관리자에게 제공되는 논리적인 객체 식별자가 필요하다. 따라서 항존성을 갖는 객체는 물리적인 식별자가 논리적 식별자로의 변환이 필요하며, 이는 저장 관리기에 의해서 수행된다.

2.2 객체관리기 인터페이스 설계방식

클래스는 현실 세계 객체 구조를 나타내며, 객체에 필요한 연산을 표현하고, 객체가 갖는 체계적 관계성을 표현한다. 클래스 정의어(Class Definition Language)를 통해서 규정된 클래스는 데이타베이스 구축의 모델이 된다. 클래스를 통해서 생성된 객체는 데이타베이스에 저장되며, 저장된 객체는 유일한 식별자를 가지며, 이를 이용해 객체가 편집되고 검색 된다[3, 17]. 객체의 조작 관리는 객체관리기 인터페이스를 이용하여 이루어진다.

사용자는 데이타베이스 내의 원하는 객체에 접근을 위해 질의어를 사용하고, 질의 처리에 의해서 질의 수행을 위한 계획이 생성되며, 이 수행계획에 의해서 질의가 실행된다. 질의 처리를 위한 연산자는 주어진 피연산자를 매개 변수로하여 연산을 실행한다. 이러한 연산은 주어진 피연산자에 의해서 원하는 객체에 접근하여야 하며, 이를 위해서 객체관리기가 제공하는 인터페이스를 사용한다.

객체관리기 인터페이스는 객체에 대한 효과적인 접근을 제공하기 위해 다음과 같은 원칙하에서 설계하도록 한다. 첫째, 객체는 메세지에 의한 접근만을 허용하고 복합 객체의 특성을 포함한 일반적인 객체의 특성과 질의 실행 요구를 수용하여야 한다.

둘째, 하나의 객체는 기본적인 객체와 여러 다른 객체가 집합적으로 이루어질 수 있으므로, 집합적 의미의 객체 운영(삽입, 삭제, 갱신)을 위한 내용이 요구되어야 한다.

셋째, 객체는 여러 가지 응용에 의해서 접근이 요구되므로 새롭고 다양한 응용의 요구에 적용될 수 있도록 확장성이 제공 되어야 한다.

넷째, 대량의 다양한 객체지향 데이타베이스의 성능 향상에 적합하도록 한다. 다량의 객체가 관리될 때 효율적인 저장공간의 이용과 객체에 대한 신속한 접근이 절실히 요구되고, 불의의 사태에 대한 자료의 손실 및 일관된 정보의 제공 등에 따른 성능 개선에 적합하도록 해야 한다.

다섯째, 복잡한 질의에 있어서 한 질의의 결과는 다른 질의의 매개체로 사용되며, 여러 단계의 질의가 복합적으로 발생할 때도 중간적인 결과뿐만이 아니라 최종적인 질의 결과는 영속적으로 저장될 수 있어야 한다.

객체관리기 인터페이스는 필수적으로 위의 다섯 가지 원칙하에서 설계되었으나, 네번째와 다섯번째의 내용은 실제 객체관리기의 버퍼 관리와 새로운 클래스 생성에 따른 유동적인 클래스 설정 관리가 충분히 이루어져야만이 보다 효율적인 인터페이스 설계가 가능하다.

3. 인터페이스 구현

객체관리기 인터페이스는 데이타베이스의 질

의 처리 뿐만이 아니라, 보편적 객체지향 시스템의 프로그래밍에서도 이용이 가능하도록 'C'언어의 라이브러리 형태로 구현한다. 함수에 표현된 'IN'은 입력되는, 'OUT'은 반환되는 인자를 나타내는 것이다. 함수명은 함수가 가지는 의미를 충분히 수용하도록 한다.

3.1 클래스 관련 인터페이스

클래스 정의어를 이용해 데이타베이스가 설계되고 객체관리기 인터페이스를 이용해서 클래스가 생성되고 수정되어 진다.

(1) 클래스 생성

다음은 새로운 클래스 생성을 위한 객체관리기 인터페이스를 설계한 것이다. 클래스의 생성은 클래스 이름(className), 하위 클래스 수(subClass-Num), 참조되는 클래스 수(referClass-Num), 클래스 요소, 속성 타입(type) 및 길이(attributeInfor) 그리고 메소드(methodInfor) 등에 관한 정보를 요구하게 된다. 이것은 객체의 운영에 필요한 정보를 제공하기 위한 것이다. 상위 객체 이름들은 하위 객체가 상속 관계가 있는 클래스의 범위를 결정하게 한다.

```
createClass( IN *className
             *superClassName
              subClassNum
              referClassNum
             *attributeInfor
             *methodInfor )
```

(2) 클래스 수정

데이타베이스는 생성된 클래스에 대한 삽입(add), 삭제(delete) 및 갱신(alter) 등을 처리 할 수 있는 인터페이스를 필요로 한다.

다음의 인터페이스들은 이미 생성된 클래스에 대하여 새로운 속성을 추가(addClassAttribute) 할 때 관련된 클래스 이름, 삽입되는 속성이름(attributeName), 속성 타입(attributeType), 속성 길이(attributeLength) 등의 인자에 대한 정보가 필요하다. 속성-영역를 새롭게 삽입하고자 하면 클래스 이름, 클래스의 속성명, 속성에 대응하는 속성-영역의 타입(attributeDomainType)과 참조되는 객체의 식별자(domainOid)에 대한 내용이 주어져야 한다. 클래스에 새로운 메세지를 처리하는 메소드가 필요할 때는 클래스 이름, 메소드 이름과 메소드 실제 내용(methodBodyPtr)이 있어야 한다. 그리고 상위 클래스(superClass) 등이 새롭게 삽입될 수 있도록 설계하였다.

```
addClassAttribute( IN *className
        *attributeName *attributeType
        *attributeLength)

addClassAttributeDomain( IN *className
        *attributeName
        *attributeDomainType domainOid)

addClassMethod( IN *className
        *methodName *methodBodyPtr( ))

addClassHierarchy( IN *className
        *superClassName)
```

또한, 클래스는 필요에 따라서 클래스 자신 삭제(deleteClass)뿐만 아니라, 속성과 메소드가 삭제(deleteClassAttribute, deleteClassMethod) 될 수 있으며, 이를 위해 다음과 같은 인터페이스가 제공된다.

```
deleteClass( IN *className)
deleteClassAttribute( IN *className *attributeName)
deleteClassMethod( IN *className *methodName)
```

객체관리기는 생성된 클래스의 클래스 이름, 속성 또는 메소드 등의 기존의 것(old)에서 새로운 것(new)으로 변경을 위해서 alterClassName(클래스이름 변경), alterClassAttributeName(속성 이름 변경)과 alterClassMethodName(메소드 이름 변경) 등의 인터페이스를 설계하였다. 또한, 상위 객체의 변경에는 alterClassHierarchy 인터페이스가 이용되며, 속성의 타입과 크기 변경에는 alterClassAttributeType과 alterClassAttribute-Length 인터페이스가 이용되고, 속성-영역의 변

경에는 alterClassAttributeDomain 인터페이스가 이용된다. 아울러, 객체의 운영을 담당하는 메소드의 내부 동작을 수정하기 위해서는 새로운 메소드로 교체되어야 하는데, 이는 alterClass-Method 인터페이스를 이용하여 수행되도록 하였다.

```
alterClassName( IN *oldClassName
        *newClassName )

alterClassAttributeName( IN *className
        *oldAttributeName
        *newAttributeName )

alterClassMethodName( IN
        *className *oldMethodName
        *newMethodName )

alterClassHierarchy( IN
        *className *oldSuperClassName
        *newSuperClassName )

alterClassAttributeType( IN
        *className *attributeName
        *oldAttributeType *newAttributeType )

alterClassAttributeLength( IN
        *className *attributeName
        *oldAttributeLength
        *newAttributeLength )

alterClassAttributeDomain( IN
        *className *attributeName
        *oldAttributeDomain
        *newAttributeDomain )

alterClassMethod( IN *className
        *methodName *newMethodNBody( ) )
```

3.2 객체 관련 인터페이스

객체 인터페이스는 객체 식별자를 이용하여 객체를 운영하기 위한 질의 수행 순서 결정에 사용되어 진다.

(1) 객체 생성

새로운 객체의 생성에는 관련된 클래스의 이름과 생성하고자 하는 클래스의 내용(상위 객체 식별자, 참조 객체 식별자, 속성 값 : objectInformTable)으로 객체가 생성되며, 새롭게 생성된 객체의 식별자를 반환하도록 하였다.

```
insertObject( IN *className *objectInformTable
        OUT newObjectID )
```

(2) 객체 삭제

객체의 삭제는 객체의 식별자(eraseObjectID)를 이용하여 이루어 진다.

```
eraseObject( IN eraseObjectID )
```

(3) 객체 검색

객체의 모든 조작에 가장 중요한 것이 검색이며, 검색을 위해서 한정되는 클래스(ClassName)와 객체의 정보(getInformTable: 클래스의 속성, 연산자, 값) 등이 포함되어야 한다.

```
getObject( IN ClassName *getInformTable OUT
        objectID )
```

(4) 객체 변경(modification)

객체의 내용을 변경하기 위해서 필요한 정보인 클래스 이름, 변경하고자 하는 객체 식별자(modifyobjectID), 그리고 관련된 속성과 값(modifyInformTable) 등을 가진다.

InsertObjectMethod와 eraseObjectMethod는 객체가 가지는 고유의 메소드에 관한 인터페이스는 여기서는 구체적으로 고려하지 않았다.

```
modifyObjectAttribute( IN *className
                       modifyobjectID
                       *modifyInformTable )

insertObjectMethod(   )

eraseObjectMethod(   )
```

3.3 사용자 뷰 인터페이스

다음 인터페이스는 검색된 내용이 어떤한 형태로 사용자에게 보여질 것인가에 대한 양식에 관련된 것이다. 출력을 위한 인터페이스는 클래스와 검색된 객체 식별자, 속성값이나 객체 식별자 출력 양식(viewInformTable) 등의 인자를 가지도록 하였다.

viewObject(IN *className objectID *view-InformTable)

현재는 사용자의 질의 인터페이스의 검색을 위한 내용만을 단순히 출력하고 있으며, 앞으로는 필요한 양식에 맞는 출력을 고려할 계획이다.

4. 질의 처리를 위한 객체관리기 인터페이스 적용

4.1 객체지향 질의 처리

데이타베이스의 객체에 접근을 위해서 사용자는 질의어를 이용하며, 질의어는 데이타베이스에서 객체들의 모임에 대해 의미있게 명시된 질의를 의미한다. 질의어의 수행은 "데이타베이스 클래스에 대한 데이타베이스 인스탄스들의 대응"으로 볼 수 있다. 질의 연산자의 의미는 클래스의 변화와 함께, 데이타베이스의 외연의 변화로서 묘사된다. 질의 처리를 위해서는 질의를 선언적으로 규정하는 복합 객체 해석(complex object calculus) 변환과 절차적 처리 및 최적화를 위한 복합 객체 대수로의 변환이 필요하고, 생성된 질의 처리 계획을 수행함으로써 마치게 된다. 질의 처리 계획은 객체 대수 표현 트리로 나타나고, 접근 계획 실행기(access plan generator)가 객체관리기 인터페이스를 이용해서 연산하게 된다.

다음 (그림 3)은 객체지향 형식 질의의 모델을 위하여 정의된 클래스 구조를 나타낸것이다.

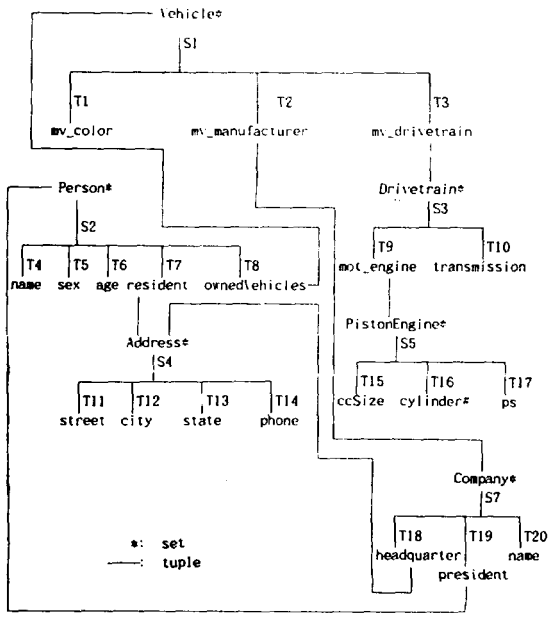
객체지향 질의에서 술어는 복합객체의 구조가 가시적이어야 하는데, 정의된 질의 언어에서는 이를 위하여 속성영역의 개체 변수로써 경로식을 사용하고, 추상자료형을 지원하기 위해서 객체 내용이 아니라 행위에 기반하여 질의할 수 있도록한다. 즉, 이는 클래스의 메소드를 이용하여 질의를 처리할 수 있음을 의미한다. 다음 [질의 1]은 질의 처리과정을 명확히 보이기 위해 사용한 질의의 예이다.

[질의 1] "홍길동이 살고있는 도시에 본사(headquarter)가 있는 자동차회사의 사장 이름을 찾아라."

위의 [질의 1]을 질의 구문을 이용하여 표현하면 다음과 같다.

SELECT C.president.name
FROM Vehicle: V, Person: P, V.mv_manufacturer: C
WHERE P.name="홍길동" AND C.headquar
ter.city=P.resident.city

위 예는 C.president.name을 찾기 위하여 Vehicle.mv_manufacturer, Company.president.name, Company.headquarter.city 및 person.resident.city와 같은 속성에 의한 경로를 표현한 것이다. 또한



(그림 3) 객체지향 질의의 클래스 예
(Fig. 3) The example of class of object-oriented query

행위 기반 질의를 위하여 SELECT절을 SELECT Display(Retrieve(C.president.name))와 같이 작성하므로써 Display와 Retrieve와 같은 연산 메소드에 의한 경로를 표현할 수도 있다.

[질의 1]을 질의 처리의 첫번째 단계인 복합 객체의 해석 변환을 Straube[11, 12, 13, 14]가 사용한 방법으로 수행하면 다음과 같다.

{0 | (∃Person(P) ∧ "홍길동"=P.name ∧ (∃ Vehicle.mv_manufacturer(C) ∧ C.headquarter.city = P.resident.city ∧ (O∈C.president.name))) }각 요소를 일원화하여 순서화하면 다음과 같다.

a1 : Person(P)
a2 : "홍길동"= P.name
a3 : Vehicle.mv manufacturer(C)
a4 : C.headquarter.city=P.resident.city
a5 : O∈C.president.name

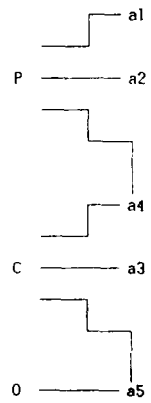
이것은 객체대수로 변환되며, 객체 대수 연산자는 Union(PUQ), Difference(P−Q), Select($P\sigma_{<Q_1, \cdots, Q_n>}$), Generate($Q_i\ \gamma_{<Q_1, \cdots, Q_n>}$) 및 Map($Q_i \rightarrow$mlist$<Q_1, \cdots, Q_n>$)을 이용하고, 변환 알고리즘은 원자의 축약표기, 해석과 대수의 등가식 및 원자배치 방법을 이용한 Straube 등[11, 12, 13, 14]의 알고리즘을 이용한다. 다음은 [질의 1]을 대수식으로 변환한 경우이다.

(Vehicle.mv_manufacturer σ_{a_4} < (Person σ_{a_2} < >) >) γ_{a_5} < >

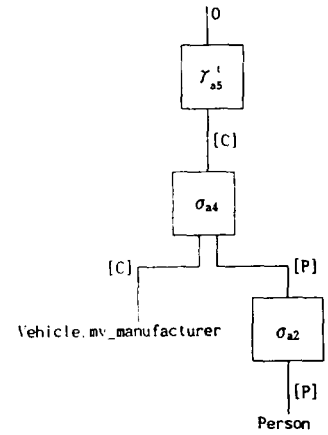
다음 (그림 4)는 대수식으로의 변환 과정 중에서 생성되는 질의 그래프로서, 순환을 형성하지 않고 해당원자가 다른 원자와 조합될 필요가 없는 이분 그래프이다. 또한, (그림 5)는 이러한 질의 그래프를 객체 대수 연산자를 이용하여 객체 대수 트리로 표현한 것이다.

4.2 객체관리기 인터페이스 적용 예

객체지향 데이타베이스에 대한 질의는 질의 처리에 의해서 (그림 4)의 질의 그래프가 생성되며, 실행 계획의 생성은 이 질의 그래프가 (그림 5)의 객체 대수 트리로 표현된 것을 객체관리기 연산의 트리로 대응 시키는 것이다. (그림 5)의 객체 대수 트리에서 링크는 객체의 집합이 개별화된 객체로써 참조되는 것을 나타낸다. 즉, mv manufacturer로 표시된 객체 집합은 [C] ∈ mv_manufacturer인 하나의 객체 [C] 로써 생각할 수 있다. 하나의 링크는 유일한 하나의 객체를 나타낸다. 이것은 연산의 결과가 다른 연산의 입력으로 사용될 수 있는 대수의 폐포성(closure property)을 만족한다.

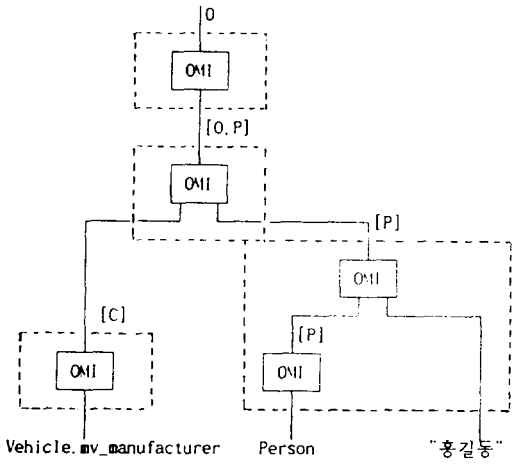


(그림 4) 질의 그래프
(Fig. 4) Query graph



(그림 5) 질의 그래프에 대한 객체 대수 트리
(Fig. 5) Object algebra tree of the query graph

(그림 6)은 대수 연산 트리에 상응하는 실행 계획을 표현하고 있다. 실행계획 그래프의 노드는 객체관리기 연산이 되며, 객체 식별자를 반환한다. 객체관리기의 연산은 객체관리기 인터페이스로써 기술되며, 실행 계획은 객체 인터페이스로 구성된 객체 그래프의 끝(leaf)에서 루트(root) 방향으로 진행된다. 실행 그래프에서 점선의 내부는 객체 대수 연산에 대한 개별적인 관리기 연산의 수행을 기술한 것이다.



(그림 6) 질의 수행 트리
(Fig. 6) Query execution tree

질의 수행 트리에 대한 객체관리기 연산은 (그림 7)과같이 객체관리기 인터페이스에 의하여 순차적 처리 프로그램으로 표현된다.

```
getObject( IN Person *getObjectInformTableP
    OUT ObjcetIDPid )
getObject( IN AutoCompany *getObjectInformTableA
    OUT ObjectIDCid )
viewObject( IN President
    *viewInformTable )
```

(그림 7) 객체관리기 인터페이스에 의한 질의 수행 모듈 표현
(Fig. 7) The representation of query execution module using OMI

객체관리기는 객체의 저장 및 검색 관리를 위해서 객체관리기의 하부 구조로써 Mneme 객체 저장 관리기를 데이타베이스에 적합하도록 확장하여 이용한다. Mneme 객체 저장 관리기는 디스크상의 객체와 사용자가 보는 메모리상의 객체간의 매핑을 위한 포인터 연결(pointer swizzling)을 포함한 객체관리기의 대부분의 기능을 수행하고 있다[13]. 또한, 이용자를 위한 편리한 인터페이스를 제공하므로써 Mneme를 이용한 객체관리 시스템의 구축을 용이하게 하고 있다. 그러나 클래스의 인스탄스에 대한 의미를 부과하는 것과 객체 검색에 대한 정보(요소 사전, 메세지 관리)를 위한 추가적인 부분및 효과적인 객체 접근을 위한 색인 관리에 필요한 자료구조 등이 요구된다.

5. 결 론

객체는 객체지향 데이타베이스의 주체로써 객체지향 데이타베이스 관리 시스템의 주요 부분인 객체관리기에 의해서 접근 및 조작된다. 객체관리기는 객체를 데이타베이스에 적절하게 저장하거나, 접근하게 하기 위해서 데이타베이스 이용자에게 자신의 객체관리기 인터페이스를 제공한다.

데이타베이스 관리자와 일반 사용자는 객체지향 형식 질의어로써 객체지향 데이타베이스에 접근하게 되는데, 비절차적 질의는 질의 처리기에 의해서 질의 수행에 필요한 순차적 계획을 생성하게 되고, 이 계획에 따라 질의 처리 순서가 결정된다.

질의 수행 계획은 객체관리기의 호출함수와 의미를 규정하는 객체관리기 인터페이스에 의하여 구체적으로 표현되어 순차적으로 수행된다. 본 논문에서는 클래스관리기 및 질의처리기와 객체를 관리하는 객체관리기를 접속시키는 객체관리기 인터페이스 시스템을 설계하고 구현하므로써, 질의 처리부와 객체관리부를 독립적으로 유지시켜 보다나은 객체관리 시스템의 개발 환경 및 다

양한 응용에의 적용을 가능하게 하였다. 이 객체 관리기 인터페이스들은 제 2 장에서 제안한 설계 원칙에 의하여 구현되었으며, 이들은 데이타베이스 뿐만 아니라 응용 프로그램에서도 사용이 편리 하도록 하기 위해서 "C" 언어의 함수로 구성하였다. 또한, 객체지향 질의언어에 따른 질의의 예와 질의 처리과정을 통하여, 객체지향 질의가 구현된 객체관리기 인터페이스에 의하여 실질적으로 순차적 실행될 수 있음을 증명하였다. 본 논문에서 객체 저장 및 관리를 위한 하부 구조로는 Mneme 객체 저장 관리기를 객체지향 데이타베이스 시스템에 적합하도록 확장하여 이용하였다.

본 논문에서 구현된 객체관리기 인터페이스는 객체지향 데이타베이스 시스템의 프로토타입으로써 객체 접근을 보다 효율적으로 하기 위한 색인기법과 동적인 클래스의 관리 등을 제공하지 않지만, 이러한 기능은 관련된 인터페이스들을 추가하므로써 확장이 가능하다. 향후 이와 같은 추가적인 성능향상의 요소들은 객체지향 데이타베이스의 보다 많은 이론적 연구와 이를 뒷받침할 수 있는 기술들의 개발과 더불어 이루어질 수 있으리라 본다.

참 고 문 헌

[1] J. Annevelink and P. Dewilde, "Object-Oriented Data Management Based on Abstract Data types," Software-Practice and Experience, Vol. 17(1), pp. 757-781, Nov. 1987.

[2] E. Bertino and W. Kim, "Indexing Techniques for Queries on Nested Object," IEEE Transaction on Knowledge and Data Engineering, Vol. 1(2), Jun. 1989.

[3] G. Copeland, M. Franklin and G. Weikun, "Uniform Object Management, Lecture Notes in Computer Science," Advances in Database Technology-EDBT, Springer-Verlag, pp. 253-268, 1990.

[4] K. E. Gorlen, "A Object-Oriented Class Library for C++ Programs," Software-Practice and Experience, Vol. 17(12), pp. 899-922, Dec. 1997.

[5] J. G. Hughes, Object-Oriented Databases, Prentice Hall international series in computer science. 1991.

[6] S. Khoshafian, "A persistent complex object database lnaguage," Data & Knowledge Engineering, North-Holland, Vol. 3(1), pp. 225-243, 1988/89.

[7] W. Kim, Introduction to Object Oriented Databases, The MIT Press, 1990.

[8] C. C. Low, H. Lu, B. C. Ooi and J. Han, "Efficient Access Methodes in Deductive and Object-Oriented Databases," DOOD., pp. 68-84, 1991.

[9] R.G.G. Cattell, Object Data Management, Sun Microsystems , Inc., Addison-Wesley Publishing Company, Inc.

[10] A. Straw, F. Mellender and S. Riegel, "Object Management in a Persistent Smalltalk System," Software-Practice and Experience, Vol. 19(8), pp. 719-737, Aug. 1989.

[11] D. D. Straube and M.T Ozsu, "Query transformation rules for an object algebra," TR. 89-23, Department of Computing Science, University of Alberta, Sep. 1989.

[12] D. D. Straube and M.T Ozsu, "Access plan generation for an object algebra," TR. 90-20, Department of Computing Science, University of Alberta, June 1990.

[13] D. D. Straube and M.T Ozsu, "Queries and Query Processing in Object-Oriented Database System," ACM Transaction on Information Systems, Vol. 8(4), pp. 387-430, Oct. 1990.

[14] D. D. Straube, Queries and query Processing in Object-Oriented Database Systems,

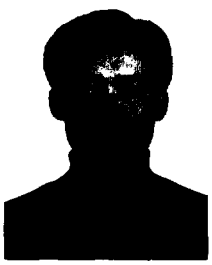
PhD thesis, University of Alberta, 1991.

[15] Velez, Bernard, and Darnis, "The O2 Object manager: An overview, The history of O2," pp. 333-366.

[16] C. Vogt, "A Buffer-Based Method for Storage Allocation in an Object-Oriented System," IEEE Transactions on Computers. Vol, 39(3), Mar. 1990.

[17] S. Khoshafian and G.P. Copeland, Object Ideintity, Readings in Object-Oriented Database System , Morgan Kaufmann Publishers, Inc. 1990.

[18] J. Eliot, B. Moss and S. Simofsky, Managing Persistent Data with Mneme : User's Guide the Client Interface, Object-Oriented System Laboratory Department of Computer and Information Science University of Massachusetts, Mar. 1989.

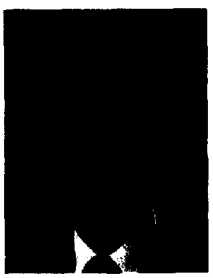


이 연 식

1982년 전남대학교 전자계산학과 졸업.
1984년 전남대학교 대학원 전자계산학 전공(이학 석사).
1994년 전북대학교 대학원 전산응용공학 전공(공학박사).
1986년~현재 군산대학교 컴퓨터과학과 부교수.
1994년~현재 군산대학교 전자계산소장.
관심분야 : 프로그래밍언어론, 객체지향 시스템, 질의처리.

전 병 실

1967년 전북대학교 졸업.
1969년 전북대학교 전자공학 전공(공학석사).
1974년 전북대학교 대학원 전자공학 전공(공학박사).
1979년~1980년 Univ.of Notre-Dame객원 교수.
1984년~1986년 전북대학교 전자산업개발연구소장.
1971년~현재 전북대학교 전자공학과 교수.
1994년~현재 전북대학교 도서관장.
관심분야 : 컴퓨터구조, 병렬처리, GUi, 객체지향응용.



류 근 호

1976년 숭실대 전산학과 졸업.
1980년 연세대 산업대학원 전산전공(공학석사).
1988년 연세대 대학원 전산전공(공학박사).
1976년~1986년 육군군수 지원사 전산실(ROTC장교), 한국전자 통신 연구소(연구원), 한국 방송통신대 전산학과(조교수) 근무.
1989년~1991년 Univ. of Arizona 연구원.
1986~현재 충북대학교 컴퓨터과학과 부교수겸 컴퓨터과학연구소장.
관심분야 : 시간지원 데이타 베이스, 시공간 데이타베이스, DBMS 및 OS, 객체 및 지식베이스 시스템.