論文93-30B-4-2

객체 중심 데이타베이스를 위한 관계성 관리 기법 및 질의어

(A Query Language and Relationship Management Techniques for Object-Oriented Databases)

黃 壽 贊*, 李 錫 浩**

(Soo Chan Hwang and Suk ho Lee)

要 約

사무 정보 시스템, CAD/CAM, 인공지능 등의 새로운 데이타베이스 응용 분야에서는 정적이고 고정된 Is-A나 Part-Of 관계성 뿐만아니라 복잡한 제약조건을 포함하고 동적으로 정의되는 다양한 관계성의 표현및 조작도 필수적인 요건이다. 그러나 기존 객체 중심 데이타베이스 시스템들은 이러한 복잡한 관계성을 표현하고 조작하는데에 많은 문제점들을 가지고 있다. 따라서 본 논문에서는 현실 세계의 복잡한 관계성들을 추상화(abstraction) 개념을 이용하여 객체 중심 데이타베이스에 표현할 수 있으며, 관계성에 관련된 다양한 무결성 및 일관성 제약조건을 표현할 수 있도록 하는 객체 중심 관계성 데이타 모델과 이 모델의 연산을 지원하며 SQL과 유사한 구조를 가지는 비절차적 질의어인 ORSQL(Object Relationship SQL)을 제시하고자 한다.

Abstract

In the new database applications such as office information systems, CAD/CAM, and AI, it is required to support not only fixed Is-A and Part-Of relationships but also various user-defined dynamic relationships including complicate constraints. However, existing object-oriented systems have many weaknesses in managing those complex relationships. This paper presents the Object-Oriented Relationship data Model(OORM) which is designed to provide facilities for modeling complex relationships into object-oriented databases using abstraction concept. In the model, various integrity and consistency constraints related to the relationships can be also represented. And this paper presents a query language, ORSQL(Object Relationship SQL). ORSQL is a nonprocedural query language having similiar syntax to the standard SQL and supporting OORM 's operations.

I. 서론

현재 객체 중심 프로그래밍(Object-Oriented Programming) 개념은 사무 정보 시스템[1], 인공지능[2], CAD/CAM[3,4] 등 여러 다양한 데이타베이스 응용 분야에서 활발히 연구되어 이미 많은 객체 중심 데이타베이스 시스템들이 개발되었다.[5,6,7] 이러한 객체 중심 데이타베이스 시스템들은 데이타베이

* 正會員, 韓國航空大學 電子計算學科
(Dept. of Computer Science, Hankuk Aviation Univ.)
**正會員, 서울 大學校 컴퓨터 工學科
(Dept. of Computer Eng., Seoul Nat'l Univ.)
接受日字: 1992年 6月 10日

스의 모델링 능력을 향상시키며, 기존 레코드 중심 데이타베이스 시스템들의 의미적 표현 능력의 부족을 보완하고 있다. 하지만 현실 세계의 다양하고 복잡한 관계성을 표현하고 조작하는 데에는 아직 많은 부족함이 있다.

일반적으로 엔티티 간의 관계는 일반화(generalization) 관계인 Is-A 관계, 집단화(aggregation) 관계인 Part-Of 관계, 그리고 연관성(association) 관계 등으로 분류될 수 있다.[8] 현재까지 객체 중심 데이타베이스 시스템 분야에서는 정적이고 고정된 Is-A나 Part-Of 관계성을 지원하기 위한 많은 연구 결과가 발표되어 왔지만, 새로운 데이타베이스 응용 분야에서는 동적으로 정의되는 다양하고 복잡한 관계성의 표현및 조작도 필수적인 요건이다. 그러나 기존 객체 중심 데이타베이스는 관계성을 직접 표현할 수 있는 도구로써의 구문(syntax)과 의미(semantics)를 제공하지 않는다.[9] 그러므로 관계성은 애트리뷰트로만 취급되므로 하나의 단위로 조작되어야 하는 관계성에 관한 정보가 관계성에 참가하는 여러 객체에 중복, 분산되게 된다. 관계성에 대한 연산도 관련된 모든 객체에 대해 적용되기 보다는 메소드의 대상이 되는 개개의 객체에 대해서 부분적으로 적용되어 잘못된 결과를 가져올 수도 있다.[9,10] 또한 데이타베이스에는 데이타의 구조와 데이타 뿐만 아니라 다양한 무결성이나 일관성 제약조건들도 표현되고 유지되어야 한다.[4] 그러나 기존 객체 중심 시스템들은 이러한 제약조건들을 표현하고 유지할 수 있는 기능도 제대로 제공되지 않고 있는 실정이다.

관계성에 관련된 객체 중심 데이타베이스 시스템들의 또 다른 문제로는 복잡한 관계성을 통해 정보를 탐색할 수 있는 기능이 제한되어 있으며, 관계 데이타베이스의 SQL과 같은 비절차적인 고급의 질의어가 없다는 것이다. 많은 클라스들 간의 복잡한 관계성들을 포함하는 방대한 데이타베이스를 조작하는데에는 기존 객체 중심 시스템들이 제공하는 단순한 개개의 인스턴스 객체를 조작하는 연산만으로는 매우 어렵다. 따라서 관련된 여러 객체들 간의 다단계 참조 관계를 통해 원하는 정보를 쉽게 탐색할 수 있는 연산이 필요하며, 이를 기반으로 원하는 바를 쉽게 표현할 수 있도록 하는 고급의 비절차적 질의어가 객체 중심 데이타베이스 시스템에 제공되어야만 한다.

따라서 본 논문에서는 현실 세계의 복잡한 관계성들을 추상화(abstraction) 개념을 이용하여 객체 중심 데이타베이스에 표현할 수 있으며, 관계성에 관련된 다양한 무결성 및 일관성 제약조건을 표현할 수 있도록 하는 객체 중심 관계성 데이타 모델(Object-Oriented Relationship Data Model, OORM)과 OORM의 연산을 지원하며 SQL과 유사한 구조를 가지는 비절차적 질의어 ORSQL(Object Relationship SQL)을 제시하고자한다. 본 논문의 2장에서는 객체 중심 관계성 모델의 기본 개념과 관계성을 표현하는 방법및 기본 연산들을 기술하며, 3장에서는 객체 중심 관계성 모델이 제공하는 연산을 지원하는 질의어 ORSQL을 설명한다. 마지막으로 4장에서 결론을 맺는다.

Ⅱ. 객체 중심 관계성 모델

객체 중심 관계성 모델은 객체 중심 개념이 제공하는 데이타 추상화, 특성 계승, 복합 객체 등의 개념과 현실 세계의 엔티티 사이에 존재하는 다양하고 복잡한 관계성과 제약조건을 정확히 표현하고 관리하는 방법을 제공한다. 또한 연산으로써 다양한 집합 연산과 다른 객체를 참조하거나 관계성을 통한 탐색을 지원하는 연산을 제공하고 있다.[11] 이 장에서는 객체 중심 관계성 모델의 기본적인 구성요소들을 한 제조회사를 위한 데이타베이스를 예로 설명하기로 한다. 그림 1은 제조회사의 업무를 모델링한 ER(Entity Relationship) 다이아그램이며, 그림 2는 이를 OORM으로 정의한 것이다.

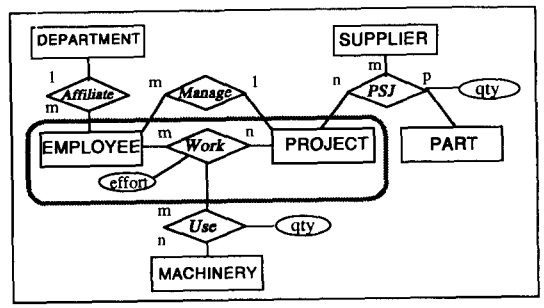

그림 1. 제조회사의 ER 다이아그램
Fig. 1. ER diagram for a manufacturing company.

```
Class Employee super Person
  (birthdate : date, salary : integer with
   [:x| x between: 10000 and: 100000] ,
   friends:{Person}, affiliate : Department)
Class Supplier super Rclass
  (sname : string, city : string)
Class Part super Rclass
```

```
  (pname : string, city : string)
Class Project super Rclass
  (jname : string, city : string, manager :
   Employee)
Class Department super Rclass
  (dname : string, deptStaus : string, chair
   : Employee)
Class Work for Employee(*), Project(*)
      super Rclass
  (effort : integer with effortConst)
   instanceMethod
    effortConst
     (Work where: [:w| w Employee = (self
      Employee)] ) sum: effort < 100.
Class PSJ for Project(*), Supplier(*), Part
      (*) super Rclass
  constraint Colocation
   [(self Part city = (self Supplier city)) and:
    [self Supplier city =(self Project city)] ] .
   (qty : integer)
Class Use for Work(*), Machinery(*) super
      Rclass
  (qty : integer)
```

그림 2. 제조회사의 스키마 정의

Fig. 2. Schema definition for the manufacturing company.

1. 객체 및 클라스

객체 중심 관계성 모델에서는 현실 세계의 모든 엔티티와 개념들이 객체로 표현되고 객체는 그들의 객체식별자(oid)에 의해 유일하게 식별된다. OORM의 객체는 실제 데이타 값이 저장되는 애트리뷰트와 객체에 대한 연산을 정의하는 메소드, 무결성 유지를 위한 제약조건으로 구성된다. 모든 객체는 클라스에 속하고 같은 클라스에 속하는 객체들은 공통 특성인 애트리뷰트와 메소드를 공유하며 동일한 제약조건을 만족해야 된다. OORM의 클라스는 애트리뷰트의 도메인, 엔티티, 관계성을 표현하며 Is-A 관계에 따라 클라스 계층을 구성한다.

OORM에서는 내용식 질의(associative query)와 집합 연산을 지원하기 위해 각 클라스마다 하나의 집합클라스가 내부적으로 정의된다. 이 집합 클라스는 하나의 인스턴스 객체를 가지며 그 객체는 해당 클라스의 모든 인스턴스들을 원소로 가지는 집합 객체이다. 이 집합 객체를 인스턴스 집합(instance set)이라 한다.

OORM의 애트리뷰트는 객체의 내부 상태를 나타내며 애트리뷰트 이름, 도메인 및 관련 제약조건으로 명세된다. 애트리뷰트의 값은 선언된 도메인 뿐만 아니라 그 서브클라스들의 인스턴스및 그들의 집합을 가질 수 있다. 그리고 도메인으로 선언된 클라스가 엔티티 또는 관계성을 표현한 경우는 그 애트리뷰트를 포함하는 클라스와 애트리뷰트를 통해 참조되는 클라스와의 관계성을 나타내므로, 참조 애트리뷰트(reference attribute)라고 한다. 또 애트리뷰트가 만족해야 하는 애트리뷰트 제약조건(attribute constraint)은 with절에 명세하는데 이 제약조건은 애트리뷰트의 값을 제한하거나 또는 다른 애트리뷰트와의 관련성을 표현하여 애트리뷰트의 값이 항상 무결성을 유지하도록 지원한다. 예로 그림 2에서 클라스 Employee의 애트리뷰트 salary는 그 값이 10000에서 100000 사이여야 된다는 제약조건을 표현한것이다.

OORM에서 객체의 행위는 메소드에 의해 표현된다. 메소드는 Smalltalk[13]와 유사하게 메소드 명, 파라미터, 메소드 본체로 구성되며 Smalltalk에서 제공하는 기본적인 메소드들과 OORM에서 정의한 연산들을 이용하여 정의한다. 한편, 각 애트리뷰트에는 그 값의 검색과 갱신을 위하여 각 사용자 정의 애트리뷰트에 대해서 애트리뷰트 이름과 같은 이름을 가지는 두개의 인스턴스 메소드가 자동적으로 정의되는데, 이를 애트리뷰트 메소드(attribute method)라 한다. 예로 그림 2의 클라스 Employee의 애트리뷰트 salary에도 검색용 메소드 salary와 갱신용 메소드 salary:가 정의된다.

2. 접합 객체

OORM의 특징 중의 하나는 객체 중심 개념이 제공하는 데이타 인캡슐레이션(encapsulation)의 장점을 잃지 않고 여러 종류의 객체들을 하나의 객체에 결집시킬 수 있다는 것이다. OORM에서 결집은 객체 접합(object concatenation) 연산을 통하여 수행되는데 그 결과로 생성된 접합 객체(concatenated object)는 접합되는 객체들의 모든 특성을 공유하게 된다. 예로 객체 aSupplier와 aPart가 각각 클라스 Supplier와 Part의 인스턴스일때 이 두 객체의 접합은 aSupplier||aPart로 표현되며, 이 접합 객체는 논리적으로 두 객체의 모든 특성을 포함하는 객체가 된다. 그러므로 두 객체에 관련된 모든 메세지들은 정확한 메세지 수신자를 명세할 필요없이 이 접합 객체에 전달하면 처리가 된다. 그러나 접합 객체는 두

객체가 물리적으로 결합된 것이 아니고 이들 두 객체 간에 일종의 메세지 프로토콜을 설정하여 특성을 공유할 수 있도록 한 것이다.

OORM에서는 객체 접합 연산을 기본으로한 카티션 프러덕트(Cartesian product) 연산을 제공한다. 카티션 프러덕트는 지정된 클라스의 각 인스턴스 객체들로 구성되는 모든 가능한 접합 객체의 집합을 생성한다. 이 연산은 여러 클라스의 인스턴스 객체들을 결합하여 한 클라스의 인스턴스 객체처럼 취급하고 특성을 공유하도록 하여, 관련된 여러 클라스들의 모든 인스턴스 객체에 대해 연산을 적용할 수 있는 방법을 제공한다. 두 클라스 S와 R의 카티션 프러덕트는 다음과 같이 정의된다.

$$S \times R = \{ s||r \mid s \in S \text{ and } r \in R \}$$

3. 관계성 객체

OORM에서 관계성은 객체로 표현되며, 이 관계성 객체는 관계성에 관련된 정보및 제약조건들을 포함하는 하나의 논리적 엔티티로 취급된다. 만약 관계성 R이 클라스 E1, E2, …, En 사이의 연관성을 표현한다면 관계성을 나타내는 관계성 클라스 R은 관계성에 참가하는 클라스들의 카티션 프러덕트인 $E1 \times E2 \times \cdots \times En$의 부분 집합과 관계성 자체의 특성 P 및 관계성 R에 관련된 제약조건 C로 정의할 수 있다. OORM에서 관계성 클라스 R은 다음과 같이 정의된다. 여기서 특성 P는 실제 클라스 정의시에 관계성의 애트리뷰트로 표현되며 제약조건은 불리언 값을 가지는 메소드 형태로 표현된다.

$$R = (E1 \times E2 \times \cdots \times En, P, C)$$

관계성 클라스의 인스턴스인 관계성 객체는 관계성 자체의 특성 뿐만 아니라 관계성에 참여하는 객체에 관한 정보를 접합 객체 형태로 포함하는데 이는 클라스들의 카티션 프러덕트는 각 클라스의 인스턴스로 구성되는 접합 객체를 생성하기 때문이다. 따라서 접합 객체의 특성 공유 기능에 의해 관계성 객체는 참가 객체의 모든 특성도 함께 공유할 수 있다. 그러므로 관계성 객체는 관계성에 관련된 모든 정보를 포함하고, 이 관계성에 관한 연산은 모든 참가 객체들에 동등하게 적용될 수 있다.

OORM에서 실제 클라스를 정의할때는 관계성과 엔티티가 모두 클라스로 표현되는데 클라스의 명세시 for 절은 정의하는 클라스가 관계성을 표현하는 경우 명세하며, 관계성에 참가하는 클라스와 카디날리티 등을 표현한다. 참가 클라스의 카디날리티는 일(1) 또는 다(*)로 명세된다. 예로 그림 2에서 관계성 클라스 Work는 Employee와 Project 사이의 다대다 관계성을 표현하며 관계성 애트리뷰트 참여율(effort)과 애트리뷰트 제약조건을 포함하고 있다. 이 제약조건은 한 고용인의 전체 참여율은 100%를 넘지 못한다는 것을 나타낸다. Machinery와 Work 사이의 이항 관계성 역시 관계성 클라스 Use로 표현되는데, 명세에서 관계성 클라스 Work는 집단화 개념에 의해 Employee와 Project, 또 그 사이의 관계성인 Work에 관한 모든 정보를 포함하는 상위 레벨의 집단 객체로 취급된다.

그림 2의 관계성 클라스 PSJ의 constraint 절은 그 클라스의 모든 객체들이 만족해야 되는 클라스 제약조건(class constraint)을 명세한다. 여기서 클라스 제약조건은 앞의 관계성 클라스 R의 정의에 있는 제약조건 C를 표현하는 것으로써 제약조건의 이름과 그 결과가 불리언 값인 프로그램 블럭으로 명세되며 여러 개의 제약조건을 한 클라스에 대해 정의할 수 있다. 예로 관계성 클라스 PSJ의 무결성 제약조건 Colocation은 "각 참가 객체는 모두 같은 도시에 존재하여야 한다"는 제약조건을 의미한다. 이 제약조건은 관계성이 생성되고 수정될 때 적용되며 이 제약조건을 만족하지 않는 관계성 객체는 존재할수 없게 된다.

4. 연산

객체 중심 관계성 모델은 일반 객체 중심 프로그래밍 언어가 제공하는 기본적인 연산에 데이타베이스 시스템에 필요한 집합 연산 및 객체 간의 관계성을 통하여 탐색을 가능토록 하는 연산 등을 추가로 제공하고 있다. 연산에 관한 공식적 정의와 예는[11,12] 을 참고하기 바란다.

1) 기본 집합 연산 : OORM은 일반적인 집합 연산으로서 합집합, 교집합, 차집합 등을 제공한다. 이러한 연산들은 일반적인 집합 객체, 인스턴스 객체의 집합, 클라스 등에 적용될 수 있으며 클라스가 연산의 대상인 경우는 해당 클라스의 인스턴스 집합에 대한 연산을 의미한다.

2) 카티션 프러덕트 : 카티션 프러덕트는 앞에서 정의된 것과 같이 지정된 클라스들의 각 객체로 구성된 모든 가능한 접합 객체의 집합을 생성한다.

3) 객체 선택 : 객체 선택(object selection) 연산은 관계 데이타 모델의 선택(selection) 연산과 유사하게 지정된 클라스의 인스턴스 집합으로부터 주어진 조건을 만족하는 객체들을 추출하는데 이용된다. 객

체 선택 연산으로는 주어진 조건을 만족하는 객체들의 집합을 반환하는 다중 객체 선택연산과 하나의 객체만을 반환하는 단일 객체 선택 연산이 제공된다. 두 연산의 대상은 클라스 또는 객체의 집합이 되며 클라스가 연산의 대상이면 해당 클라스의 인스턴스 집합에 대해 연산이 적용된다.

4) 프로젝션 : OORM에서 애트리뷰트 메소드는 단일 객체의 애트리뷰트에 대한 검색및 갱신 기능을 제공한다. 그리고 한 클라스의 모든 인스턴스 또는 인스턴스들의 집합으로부터 애트리뷰트 값을 검색하기위해서는 프로젝션(projection) 연산을 제공한다. 이 프로젝션 연산은 접합 객체들의 집합으로부터 원소 객체를 추출하기 위해서도 이용되는데 그 결과는 접합 객체의 구성원소인 한 객체 또는 접합 객체들의 집합이 반환된다. 그리고 접합 객체의 집합으로부터 원소 객체들을 검색하기 위해서는 원소의 클라스명만을 단순히 표현하면 된다.

5) 참조 조인 : 참조 조인(referential join)의 목적은 참조하는 객체와 참조되는 객체들을 결합함으로써 관련된 클라스들의 모든 객체에 연산을 적용할 수 있게 하려는 것이다. 만약 클라스 S로부터 클라스 R로 S의 애트리뷰트 a에 의한 참조 관계가 있다면, S와 R의 참조 조인은 S의 참조하는 객체와 이 객체에 의해 참조되는 R의 객체로 구성되는 접합 객체들의 집합을 생성한다. 참조 조인의 정의는 다음과 같다.

클라스 S의 애트리뷰트 a의 도메인이 클라스 R이고,
s [a] 는 객체 s의 애트리뷰트 a의 값을 의미한다면,
S join(a) R={s||r | s∈ S and r ∈ R and s [a] =r}

Ⅲ. 질의어

이 장에서는 OORM의 연산을 지원하는 비절차적 질의어 ORSQL에 대해 설명하기로한다. ORSQL은 현재 관계 데이타베이스 시스템의 표준 데이타 언어인 SQL[14]을 객체 중심 관계성 데이타베이스 시스템에 사용할 수 있도록 확장한 것으로서 데이타베이스를 검색하고 갱신하는데에 이용되는 데이타 조작어 기능을 가진다. 질의 예들은 그림 2의 스키마에 대한 것이다.

1. 검색문

검색용 질의어의 기본 구조는 기존 SQL과 동일하게 SELECT-FROM-WHERE의 형태를 가진다. ORSQL의 기본검색문 구조는 다음과 같다.

SELECT [+] 목표리스트
FROM 질의 대상
WHERE 조건

목표리스트에는 검색하고자하는 대상이 명세되는데 클라스의 특성(애트리뷰트, 메소드) 이름 또는 산술연산자나 함수 등을 포함하는 수식(expression)이 명세될 수 있다. 특성은 한정된(qualified) 형태로 사용될 수 있으며, 특성 이름이 질의 대상 클라스 사이에서 유일할 때에는 생략할 수도 있다.

FROM 절의 질의 대상은 일반 객체 중심 언어에서의 메세지 수신자를 의미하는데 하나, 또는 다수의 클라스, 객체의 집합, 클라스 간의 참조 조인 형태 등이 명세될 수 있다. 질의 대상으로 단일 클라스가 명세되면 그 클라스의 인스턴스 집합이 질의 대상이 되며, 다수 클라스가 명세되면 각 클라스의 인스턴스 집합들 간의 카티션 프러덕트가 질의 대상이 된다. 그리고 클라스 간의 참조 조인이 명세되면 참조 조인 연산의 결과가 질의 대상이 되는데 이는 SQL에서 조인 조건이 조건식에서 표현되는 것과는 구분이 된다. 조인 명세가 질의 대상에 명세되어야 하는 이유는 객체 간의 관계성이 객체 중심 데이타베이스에서는 참조(reference) 관계를 통해 데이타베이스에 표현되어 있으므로 관계 모델에서처럼 질의시에 조인 조건을 표현할 필요없이 이미 명세된 관계성만을 질의 검색 대상으로 지정하기만 하면된다.

ORSQL에도 범위 변수를 사용할 수 있다. 이 변수는 하나의 객체를 나타내는 객체 변수(object variable)로서 이 변수를 이용하면 목표리스트나 WHERE 절의 조건을 간략히 기술할 수 있다. 객체 변수는 특성 이름 등의 한정자(qualifier)로 이용되거나 객체의 객체식별자(oid) 자체가 검색 대상이 될 때 이용한다. oid를 검색하는 경우는 부속질의어및 갱신 연산에만 사용할 수 있으며, 객체 변수 선언이 생략되면 클라스 이름으로 대체된다. 다음은 FROM 절의 질의 대상 구문에대한 BNF 양식을 보이고 있다. FROM 절의 실제 예는 질의문 예에서 설명하기로 한다.

FROM 〈질의 대상〉
〈질의 대상〉 ::= 〈클라스선언식〉 | 〈클라스선언식〉, 〈클라스선언식〉
〈클라스선언식〉 ::= 〈클라스범위식1〉 | (〈참조조인식〉)
〈참조조인식〉 ::= 〈클라스범위식2〉 ! 〈참조애트리뷰트식〉 〈객체변수〉
| (〈참조조인식〉) 〈객체변수〉
〈참조애트리뷰트식〉 ::= 〈애트리뷰트명〉 | (〈애트리뷰트명〉 〈객체변수〉)

〈클라스범위식2〉 ::= 〈클라스명〉 | (〈클라스명〉 〈객체변수〉)

〈클라스범위식1〉 ::= 〈클라스명〉 | 〈클라스명〉 〈객체변수〉

WHERE 절의 조건은 SQL과 유사하게 "한정 애트리뷰트 θ 한정 애트리뷰트" 또는 "한정 애트리뷰트 θ 수식" 등의 프레디킷(predicate)과 불리언 연산자인 AND, OR, NOT 등으로 구성된다. 여기서 θ 는 =, ~=, 〉, 〉=, 〈, 〈= 등의 비교 연산자를 의미하며 수식은 직접 값을 명세하거나 클라스에 명세된 메소드를 이용하는 메세지 수식, 산술식 등을 명세할 수 있다. 또한, 조건식에는 멤버십 조건인 [NOT] IN과 존재 정량자(existential quantifier)인 [NOT] EXISTS 등을 이용하여 부속질의어를 명세할 수 있다.

연산의 결과는 사용자에게 테이블 형태로 디스플레이되어 지는데 부속 질의어에서 oid가 검색 대상인 경우 그 결과는 oid의 집합이 된다. 그러나 단일 객체만을 검색해야 하는 경우는 SELECT 다음에 +를 명세하여 SELECT+와 같이 기술한다. 이때 하나 이상의 객체가 검색된 경우는 경고 메세지를 주며 검색된 객체중 첫번째 객체를 선택한다. 그러면 질의의 예들을 보이기로 한다.

(예 1) 30세 이상인 남자 고용인에 관한 정보를 모두 검색하라.

```
SELECT *
FROM Employee e
WHERE e.age 〉 30 and e.sex = 'Male'
```

예 1의 질의는 조건식을 만족하는 객체의 모든 애트리뷰트를 검색하는 경우로 모든 애트리뷰트의 명세 대신 '*'를 기술하면 된다. 그리고 FROM 절의 e는 객체 변수로서 Employee 클라스의 한 인스턴스 객체를 나타내며 생략할 수도 있다. 다음 예 2는 관계 데이타베이스의 조인과 같이 값에 의한 조인(value join)을 행하는 질의이다.

(예 2) 동일한 도시에 있는 공급자와 프로젝트의 이름을 검색하라.

```
SELECT s.sname, j.jname
FROM Supplier s, Project j
WHERE s.city = j.city
```

(예 3) 30세 이상의 여자 친구를 가진 고용인을 검색하라.

```
SELECT *
FROM Employee e
WHERE EXISTS ( SELECT f
               FROM e.friends f
               WHERE f.sex = 'Female' and f.age 〉 30 )
```

예 3은 부속질의어의 사용예를 보인것으로 Employee의 집합 애트리뷰트인 friends가 참조하고 있는Employee 객체들중 성별이 여자이며 나이가 30세 이상인 객체가 존재하면 그 고용인에 관한 정보를 검색하게 된다. Employee 클라스의 애트리뷰트 friends는 그 자체가 집합 값을 가지므로 이 집합의 개개원소에 대한 조건을 명세하기 위해 부속질의어를 이용하였다. 부속질의어에서 FROM절의 질의 대상에 명세된 'e.friends f'에서 객체 변수 f의 값은 클라스 Employee의 한 객체 e가 친구로 참조하고 있는 다른 Employee의 객체 하나를 나타낸다. 위의 질의는 다음과 같이 참조 조인을 이용하여 동일하게 표현할 수 있다.

```
SELECT e
FROM (Employee e)!(friends f)
WHERE f.sex = 'Female' and f.age 〉 30
```

위의 질의 대상은 Employee 클라스를 집합 애트리뷰트인 friends의 참조 관계를 이용하여 참조 조인한 결과가 된다. 참조 조인은 ORSQL에서 '클라스명!참조애트리뷰트'의 형태로 표현된다. 위 참조 조인의 결과는 친구 관계인 두 Employee 객체가 접합된 접합 객체의 집합이다. 참조 조인이 FROM 절에 명세될 경우에 조인 결과 생성되는 접합 객체의 일부만을 나타내는 객체 변수를 별도로 명세할 수도 있다. 위의 질의에서 객체 변수 e는 접합된 두 Employee 객체중 첫번째 객체를, f는 두번째 객체를 나타낸다.

다음은 복잡한 질의의 예로 조건을 명세하기 위한 참조 경로(referential path)와 목표리스트를 검색하기 위한 참조 경로가 서로 다른 예를 보이기로 한다. 참조 경로란 질의의 목표리스트 또는 조건을 표현하는데 사용되는 클라스와 그들 간의 참조 관계를 그래프 형태로 표현한 것을 말한다. 참조 경로의 명세는 그림 3에 있으며 이러한 참조 경로를 ORSQL의 검색문으로 표현한 것이 예 4에 있다.

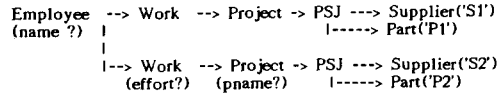

그림 3. 참조 경로

Fig. 3. Referential path.

(예 4) 공급자 S1 이 부품 P1 을 공급하는 프로젝트에서 일을 하는 고용인의 이름과 그 고용인이 공 급자 S2 가 부품 P2 를 공급하는

프로젝트에 참여하는 참여율 및 그 프로젝트명을 검색하라.

```
SELECT w.name, w.effort, w.jname
FROM Work w
WHERE w.Project IN ( SELECT j.Project
                     FROM PSJ j
                     WHERE j.pname = 'P2'
                     and j.sname = 'S2' )
and w.Employee IN ( SELECT j1.Employee
                    FROM PSJ j1
                    WHERE j1.pname = 'P1'
                    and j1.sname = 'S1' )
```

질의 4에서는 부속질의어를 멤버십 조건인 IN을 통하여 연결하고 있다. 만약 그림 3의 클라스들이 관계 데이타베이스의 테이블들로 표현되어 있다면 위의 질의를 SQL 형태로 바꾸면 다음과 같이 매우 많은 수의 조인 조건을 포함하는 복잡한 질의가 될 것이다. 아래 질의에 있는 애트리뷰트 이름중에 문자열 'id'가 붙은 것은 기본키로서 조인에 사용되는 애트리뷰트라 가정한다.

```
SELECT Employee.name, Work.effort,
       Project.jname
FROM Employee, Work, Project
WHERE Employee.eid = Work.eid and
       Work.jid = Project.jid
  and Employee.eid in (
    SELECT Employee.eid
    FROM Employee, Work, Project, PSJ,
     Supplier, Part
    WHERE Employee.eid = Work.eid and
     Work.pid = Project.jid
      and Project.jid = PSJ.jid and PSJ.sid
          = Suppier.sid
       and PSJ.pid = Part.pid and Supplier.
          sname='S1' and Part.pname = 'P1' )
and Project.jid in (
    SELECT Project.jid
    FROM Employee, Work, Project, PSJ,
     Supplier, Part
    WHERE Employee.eid = Work.eid and
     Work.pid = Project.jid
      and Project.jid = PSJ.jid and PSJ.sid
          = Suppier.sid
      and Part.pid = PSJ.pid and Supplier.
          sname = 'S2' and Part.pname =
          'P2' )
```

2. 갱신문

ORSQL의 갱신문으로는 SQL처럼 기존 객체의 애트리뷰트 값을 변경하기 위한 UPDATE 문과, 기존 객체를 삭제하기 위한 DELETE 문, 새로운 인스턴스 객체의 생성을 위한 INSERT 문이 제공된다. ORSQL의 UPDATE문의 기본 구조는 다음과 같다.

```
UPDATE 갱신대상
SET 애트리뷰트 θ 수식 | 검색문
WHERE 조건
```

UPDATE문에서 갱신 대상절에는 일반적으로 갱신하고자하는 클라스 명이 지정되지만 참조 조인 형태도 나타날 수도 있으며, 검색문과 마찬가지로 객체변수를 사용할 수 있다. SET 절은 SQL과 마찬가지로 갱신할 애트리뷰트가 지정되지만 애트리뷰트의 유형이 관계 모델보다 다양하므로 연산자 θ 는 치환 기호(:=) 뿐만아니라 집합 연산자인 UNION(합집합), MINUS(차집합), INTERSECT(교집합) 등을 사용한다. 그리고 애트리뷰트의 갱신할 값은 SQL과 같은 수식이나, 참조 애트리뷰트와 같이 oid가 그 값인 경우는 검색문을 이용한다. 이때 검색문의 SELECT 절에는 갱신할 애트리뷰트의 도메인에 적합한 목표리스트가 지정되어야 한다.

(예 5) 30세이상 고용인의 월급을 10% 인상하라

```
UPDATE Employee e
SET e.salary = e.salary * 1.1
WHERE e.age > 30
```

(예 6) 이름이 Kim'인 고용인의 부서를 ABC 로 바꾸어라.

```
UPDATE Employee e
SET e.affiliate := (SELECT+ d
                    FROM Department d
                    WHERE d.dname = 'ABC' )
WHERE e.name = 'Kim'
```

예 6의 갱신에서 클라스 Employee의 애트리뷰트 affiliate는 단순 객체를 참조한다. 즉 애트리뷰트 값으로 클라스 Department의 한 인스턴스 객체의 oid를 그 값으로 가지므로 객체의 집합을 결과로 만드는 SELECT 대신 SELECT+를 이용한다.

ORSQL에서는 관련된 여러 클라스들의 객체들을 하나의 갱신문을 이용하여 한꺼번에 갱신할 수 있다. 이런 경우 관련된 여러 객체들을 지정하기 위해 검색문의 FROM 절과 마찬가지로 참조 조인 형태가 UPDATE 절에 나타날 수 있다. 다음 예 7은 참조 조인을 이용하여 한꺼번에 클라스 Employee의 집합 애트리뷰트 friends에 원소들을 첨가하며, Department의 애트리뷰트 deptStatus를 갱신하는 경우이다.

(예 7) 'ABC' 부서에 근무하는 고용인 모두를 서로 친구 관계로 설정하고 그 부서의 상태를 very good '으로 변경하라.

```
UPDATE Employee!affiliate c
SET c.friends UNION ( SELECT p
                      FROM Employee p
                      WHERE p.affiliate = c.affiliate)
    c.deptStatus := 'very good'
WHERE c.dname = 'ABC'
```

ORSQL에서는 새로운 객체를 생성하기 위한 삽입문으로 두 종류를 제공한다. 첫번째는 각 애트리뷰트의 값을 수식이나 검색문 등을 이용하여 직접 지정하는 방법으로써 한번에 하나의 객체가 생성된다. 두번째는 다른 클라스의 기존 인스턴스들로부터 새로운 객체의 각 애트리뷰트 값을 지정받는 경우로써 이 경우는 한번에 다수의 객체가 생성된다. 다음은 앞에서 설명한 두가지 종류의 삽입문 형태이다.

```
INSERT INTO 삽입 대상
VALUES (애트리뷰트명 : 수식 | SELECT 문:
        {,애트리뷰트명 : 수식 | SELECT 문:} )
INSERT INTO 삽입 대상 클라스(애트리뷰트 리스트)
 SELECT 문
```

(예 8) 고용인 ' Kim '이 프로젝트 ' J1 '에 참여율 30%로 새로 참여한다.

```
INSERT INTO Work
  VALUES( Employee : SELECT+ e
                     FROM Employee e
                     WHERE e.name = 'Kim' ;
          Project : SELECT+ j
                    FROM Project j
                    WHERE j.jname = 'J1' ;
          effort : 30 )
```

관계성 객체 Work에는 관계성에 참가하는 Employee와 Project의 참여 객체를 각각 지정해야 하므로 검색문을 이용하여 oid를 검색하며, 참여율 애트리뷰트는 직접 값을 지정했다. 한번에 다수의 관계성 객체를 명세하는 경우의 삽입문이 예 9에 있다.

(예 9) 고용인 'Kim'이 'Seoul'에 위치하는 모든 프로젝트에 참여율 10%로 새로이 참가한다

```
INSERT INTO Work (Employee, Project,
     effort)
   SELECT e, j, 10
   FROM Employee e, Project j
WHERE e.name = 'Kim' and j.city =
'Seoul'
```

위의 삽입문에 나타나는 검색문은 Employee 클라스와 Project 클라스를 카티젼 프러덕트한 결과중에서 조건을 만족하는 접합 객체들을 선택한후 각각 고용인과 프로젝트 객체의 oid를 검색하며 새로 삽입되는 객체의 참여율은 모두 10의 값을 갖도록 한다.

삭제문의 기본 구조는 다음과 같으며 FROM 절에는 객체 변수 선언을 포함한 삭제 대상 객체의 클라스가 명세되며 WHERE 절의 조건은 삭제될 객체를 선택하기 위한 조건을 명세한다. 예 10은 삭제문의 예를 보인것이다.

```
DELETE
FROM 삭제 대상
WHERE 조건식
```

(예 10) 공급자 S1'이 부품 'P1'을 더이상 공급하지 않는다.

```
DELETE
FROM PSJ j
WHERE j.pname= 'P1' and j.sname = 'S1'
```

Ⅳ. 결 론

본 논문에서는 객체 중심 데이타베이스 시스템을 위한 데이타 모델로써 객체 중심 관계성 모델을 제시하고 이를 지원하는 비절차적 질의어인 ORSQL을 제시하였다.

객체 중심 관계성 모델에서 관계성은 추상화 개념을 지원하는 관계성 객체로 표현된다. 이 관계성 객체는 데이타베이스의 의미적 구조물로써 관계성의 의미와 그 제약조건들을 명확히 표현할 수 있도록 한다. 또한 관계성 간의 Is-A 관계에 따라 일반화 계층을 구성할 수 있도록 함으로써 현실 세계의 복잡한 관계성이 그 의미에 따라 동적이며 체계적으로 데이타베이스에 표현될 수 있도록 하였다. 그리고 객체 중심 데이타베이스의 객체 및 관계성을 조작하기 위한 연산을 정의하여 관계성에 관련된 연산은 관계성 객체에만 적용토록하여 관계성을 통한 데이타베이스의 효율적인 검색과 조작이 가능하다.

한편 ORSQL은 관계 데이타베이스의 SQL과 유사한 고급의 비절차적인 질의어로서 복잡한 관계성을 포함한 데이타베이스에 대한 질의를 쉽게 표현할 수 있도록 하였다. 또한 ORSQL은 기존 SQL의 구문을 그대로 이용하면서 확장하였으므로 SQL에 익숙한 관계 데이타베이스의 사용자들도 새로운 객체 중심 데이타베이스를 쉽게 사용할 수 있을 것이다.

본 논문에서 제시한 객체 중심 관계성 모델은 IBM PC상에 Smalltalk/V [13]를 이용하여 프로토타입

시스템으로 구현되었다.[11] 앞으로는 ORSQL의 효율적인 질의 처리기 구현과 인덱싱과 클러스터링 등의 저장 기법을 위한 연구가 필요하다고 본다.

參 考 文 獻

[1] Ahlsen, M., et al., "An Architecture for Object Management in OIS," *ACM Trans. on on Office Info. Syst.*, 2(3), 1984.

[2] Shephard, A. and Kerschberg, L., "PRISM : A Knowledge Based System for Semantic Integrity Specification and Enforcement in Database Systems," *Proc. of ACM SIGMOD Conf.*, 1984.

[3] Afsarmanesh, H., et al., "An Object-Oriented Approach to VLSI/CAD," *Proc. of 11th Int. Conf. on VLDB*, 1985.

[4] Cammarata, S.J. and Melkanoff, M. A., "An Interactive Data Dictionary Facilities for CAD/CAM Data Bases," *Expert Database Systems, Proc. of 1st Int. Workshop*, 1986.

[5] Banerjee, J., et al., "Data Model Issues for Object-Oriented Applications," *ACM Trans. on Office Info. Syst.*, 5(1), 1987.

[6] Copeland, G. and Maier, D., "Making Smalltalk a Database System," *Proc. of ACM SIGMOD Conf.*, 1984.

[7] Lecluse, C., Richard, P., and Velez, F., "O2, an Object-Oriented Data Model," *Proc. of ACM SIGMOD Conf.*, 1988.

[8] Korth, H.F. and Silberschatz, A., *Database System Concepts*, McGraw Hill, 1986.

[9] Rumbaugh, J., "Relations as Semantic Constructs in an Object-Oriented Language," *Proc. of ACM OOPSLA Conf.*, 1987.

[10] Cattell, R.G.G. and Rogers, T.R., "Combining Object-Oriented and Relational Models of Data," *Proc. of Int. Workshop on Object-Oriented Database Systems*, 1986.

[11] Hwang, S. and Lee, S., "The Object-Oriented Relationship System for Managing Complex Relationships," *Proc. of Int. Symp. on Database Systems for Advanced Applications*, 1991.

[12] Hwang, S. and Lee, S., "An Object-Oriented Approach to Modeling Relationships and Constraints based on Abstraction Concept," *Proc. of Int. Conf. on Database and Expert Systems Applications*, 1990.

[13] Digitalk Inc., *Smalltalk/V Tutorial and Programming Handbook*, 1987.

[14] Date, C.J., *A Guide to The SQL Standard*, 2nd ed., Addison-Wesley, 1989.

[15] Kifer, M., Kim, W., and Sagiv, Y., "*Querying Object-Oriented Databases.*" *Proc. of Int. Conf. on Management of Data*, 1992.

著 者 紹 介

黃 壽 贊 (正會員)

1984年 서울대학교 컴퓨터공학과 졸업. 1986年 서울대학교 컴퓨터공학과 석사학위 취득. 1991年 서울대학교 컴퓨터공학과 박사학위 취득 1991年 ~ 현재 한국항공대학교 전자계산학과 조교수로 재직중. 주관심분야는 데이타베이스, 객체 지향 시스템, 멀티미디어 데이타베이스 시스템 등임.

李 錫 浩 (正會員)

1964年 연세대학교 정치외교학과 졸업. 1975年 ~ 1979年 미국 텍사스대학교 전산학 석사와 박사학위 취득. 1979年~1982年 한국과학원 전산학과 조교수. 1986年~1988年 한국정보과학회 부회장. 1988年~1989年 IBM Watson 연구소 객원교수. 1989年~1991年 서울대학교 중앙교육연구전산원 원장. 1982年~현재 서울대학교 컴퓨터공학과 교수로 재직중. 데이타베이스, 화일처리, 자료구조 등을 강의.