# 관계 역할에 따른 개체-관계 모델의 객체지향 데이터베이스 모델로 변환

김 삼 남[†] · 이 홍 로[††] · 류 근 호[†††]


## 요 약

다양한 응용 분야의 데이터베이스 설계시에 시스템 사용자와 설계자 간의 이해를 증진시키고 올바른 모델링을 위해 개체-관계 모델이 많이 사용되고 있다. 그렇지만 효과적인 구현을 위해 이 개체-관계 모델로 표현된 데이터베이스는 데이터 조작과 표현이 효율적인 객체지향 데이터베이스 모델로 변환되어야 한다. 이를 위해 개체-관계 모델의 모든 개념을 객체지향 모델의 개념으로 의미의 손실없이 변환시키는 방법이 연구되어야 한다.

이 논문은 일반화 상속과 집단화 상속의 개념을 도입하여 개체-관계 모델의 기능을 확장시켰고, 개체-관계 모델의 각 성분을 객체지향 데이터베이스 모델의 성분으로 변환시키는 규칙을 제시하고자 한다. 이 변환 규칙에서 역할에 따른 변환 기법을 제시하고, 이 제시된 변환 규칙을 이용하여 개체-관계 모델에서 객체지향 모델의 스키마로 의미의 손실 없이 변환됨을 예로써 보인다. 제안된 변환 기법은 데이터베이스 설계의 논리적 모델 설계에 활용될 수 있다.


# Transforming an Entity-Relationship Model into an Object-Oriented Database Model Depending on the Role of Relationship

Sam Nam Kim[†] · Hong Ro Lee[††] · Keun Ho Ryu[†††]


## ABSTRACT

The Entity-Relationship (E-R) model is widely used not only to increase understanding between user and designer, but also to model the relationship of real world data appropriately when designing database system in many application areas. It should be then transformed into an Object-Oriented database model which gives good merits to represent and manipulate data efficiently. Therefore, a method of transforming an E-R model into an Object-Oriented database model should be studied, but without losing any semantics of concept for the E-R model.

This paper not only deals with transformation rules taking as input the elements of E-R model and delivering the elements of an Object-Oriented database model, but also improves the concept of generalization and aggregation inheritance. The paper also presents a method of transformation of relationship depending on these rules. The proposed method that obtains Object-Oriented database schema from an E-R model with preserving the properties of the E-R model is shown with examples. The method presented is able to be used to the logical database design.


† 정 회 원:육군 교육사령부 C4I 개발단
†† 정 회 원:충북대학교 컴퓨터정보통신연구소 연구원
††† 종신회원:충북대학교 컴퓨터과학과 교수
논문접수:1997년 2월 4일, 심사완료:1997년 7월 23일

## 1. 서 론

개체-관계(Entity-Relationship : E-R) 모델은 데이터베이스의 개념적 설계시에 폭넓게 사용되는 모델로서 사용자의 요구 사항으로부터 그 조직의 개체, 사건, 활동 및 그들 사이의 관계를 쉽게 표현한다. 이러한 E-R 모델은 최종 사용자가 필요로 하는 내용을 시스템의 특성에는 독립적이면서, 시스템 설계자와 시스템 사용자 모두가 이해할 수 있는 형태로 표현하는 것이다. 또한 데이터베이스의 논리적 설계시에 사용되는 모델로는 관계 모델[17, 19, 21], E-R 모델의 확장형인 의미망 모델[8, 12] 및 객체지향 모델[2, 12] 등이 있다. 이들 모델 중 객체지향 모델은 실세계의 의미를 풍부하게 시스템에 표현할 수 있고 복잡한 객체들 간의 관계성을 표현하는데 적절하다. 그래서 데이터베이스의 설계시에 시스템 사용자와 설계자 간의 이해를 돕기 위하여 E-R 모델을 객체 지향 모델로 변환하는 것이 필요하다.

최근의 E-R 모델에 대한 연구는 객체지향 E-R 모델[7], 행위 통합 E-R 접근방법[9], 객체 모델링 기법[19], E-R 언어[1], 객체지향 모델링을 위한 형식화[16] 등이 있다. E-R 모델을 또 다른 모델로 변환하기 위한 방안으로서 Kornatzky[11]는 E-R 모델의 변형인 이항-관계 모델을 객체지향 모델로 변환하는 기법을 연구하였다. 또한 Poncelet[18]는 확장된 E-R 모델인 IFO 모델을 $O_2$ 객체지향 모델로 변환하기 위한 방안을 제시하였다. 그러나 이러한 모델 변환은 E-R 모델의 모든 개념을 객체지향 모델로 의미의 손실 없이 보존하지는 못하는데, 특히 함수 무결성(functional integrity)이라 부르는 E-R 모델에서 제약조건을 객체지향 모델로 표현하기가 어렵다. 또한 변환된 객체 지향 모델 관점에서 ODMG-93[3]는 객체 정의어 구문을 속성-영역(attribute-domain) 관계를 나타내는 속성, 여러 객체들의 운행 경로(traversal path)를 규정하는 관계(relationship)와 메소드를 규정하는 연산으로 구성하였다. 그래서 이 관계성를 명시적으로 표현하지 않고 속성과 메소드만 규정하는 간략화를 위해서 E-R 모델에서 관계성의 의미 부여에 따라 의미를 집단화(aggregation)와 연관화(association)로 구분하는 것이 필요한데, 일반적인 객체지향 모델은 개체 사이의 관계성을 모델화하기 위해서 두 가지 구성자만을 제공하고 있다. 이것은 객체의 합성화와 연관화 사이에 애매 모호성을 증가시키고, 합성 객체와 성분 객체 사이에 구조와 행위 면에서 상호 작용의 오류를 발생시킬 수 있다. 이 문제점은 집단화와 연관화를 의미적으로 분리함으로써 해결할 수 있다.

이 논문에서는 E-R 모델 중에서 Elmasri와 Navath [6]의 특성을 기반으로 하고, 객체지향 모델 중에서 Ling 등[13, 14 15]의 특성을 기반으로 설계하고자 한다. E-R 모델에서 일반화, 연관화와 집단화 관계의 의미를 객체지향 모델에서 클래스 계층, 연관화 계층, 집단화 계층으로 변환하는 기법을 제시한다. 아울러 관계에 대한 변환에 추상화를 도입하고, 집단화 및 연관화 타입 계층의 속성-영역 관계를 형성하기 위한 역할(role)의 형식화도 제시한다. 이러한 변환은 집단화 참조와 연관화 참조 사이의 일반적 차이에 기반을 두고 있으며 세분화와 집단화 추상성 둘 다에 상속성을 도입함으로써 기능 면에서 재사용성을 증가시킬 수 있다.

이 논문은 다음과 같이 구성한다. 2장에서는 E-R 모델과 객체지향 모델의 구성요소에 대해 기술한다. 그리고 3장에서는 E-R 모델을 객체지향 모델로 변환하는 기법을 기술하고 비교분석하며, 4장에서는 결론을 논의하고자 한다.

## 2. 개체-관계 모델과 객체지향 모델

이 장에서는 개체-관계(E-R) 모델과 객체지향 모델을 간단히 기술한다.

### 2.1 개체-관계 모델

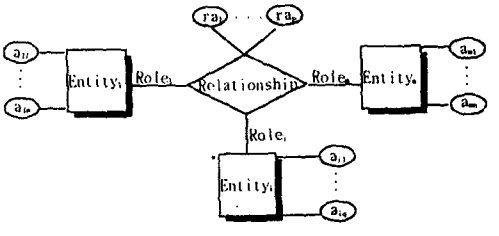
E-R 모델은 개체들의 모임, 개체들 사이의 관계, 그리고 개체를 기술하기 위한 속성들로 구성된다. 또한 한 개체가 참여할 수 있는 관계 인스턴스의 수를 규정하는 대응수 제약조건이 필요하다. 이 제약조건을 이항관계에서는 1:1, 1:N과 M:N로 나타내고, 3항 관계에서는 1:1:1, 1:1:N, 1:N:M과 M:N:P의 형식으로 표기된다. 이들은 최대 대응수(maximum cardinality)라고도 표기한다.

참여 제약조건은 관계의 제약조건으로서 관계를 경유하여 개체와 개체간에 연관된 최소 참여 개체 수이다. 이 제약조건을 최소 제약조건(minimum con-

straint)이라고도 한다. 전체(total)와 부분(partial) 참여는 개체와 개체 사이의 관계에 의한 참여 제약조건의 두 가지 타입이다. 전체 참여는 한 개체 인스턴스가 다른 개체 인스턴스와 관계없이 존재할 수 없음을 나타낸다. 부분 참여는 개체 인스턴스가 다른 개체 인스턴스와의 관계에 참여없이 존재할 때 나타난다. 이 참여 제약조건과 대응수 제약조건을 함께 사용한 형식으로 최소와 최대 제약조건(*min, max*)을 표현하기도 한다. 일반화와 세분화는 개체타입들 사이에 상위개체타입과 하위개체타입의 관계를 규정하는 것이다.

일반화와 세분화 계층(generalization/specialization hierarchy)에는 두 가지 제약조건 즉, 분해와 완전성 제약조건이 있다[6]. 첫째, 분해 제약조건은 하나의 개체가 하위타입에 속하는 개체의 한 개 이상에 나타날 수 있는지(중첩 존재(overlapping))와 나타낼 수 없는지(분해(disjoint))를 규정한다. 만일 상위 타입에서 하나의 개체 인스턴스가 다중 하위타입의 개체에 나타날 수 있을 경우, 세분화 계층은 중첩을 허용한다고 말하고, 만일 그렇지 않다면 하위타입은 분해라고 한다. 두 번째 제약조건은 완전성 제약조건으로 상위타입의 개체 인스턴스가 적어도 하나의 하위타입에 속함이 없이 존재할 수 있는지(부분 세분화(partial specialization))와 없는지(전체 세분화(total specialization))를 규정한다. 개념적 모델인 E-R 모델의 개체, 관계, 속성, 일반화 및 데이터베이스 스킴에 대해서 *Elmasri*[6] 등을 참조바란다.

아래 (그림 2.1)의 $Entity_1$, ..., $Entity_i$, ..., $Entity_m$은 개체 타입들을 나타내고, Relationship은 m개의 개체에 대한 m항 관계(m-ary relationship)를 표시하고 있다. 그리고 $a_{11}$, ..., $a_{1m}$, $a_{i1}$, ..., $a_{iq}$, $a_{m1}$, ..., $a_{mn}$과 $ra_1$, .., $ra_p$은 각각 $Entity_1$, ..., $Entity_i$, ..., $Entity_m$과 Relationship에 대한 원자 영역을 가지는 속성들을 나타낸다. Relationship에 참가하는 각 개체 타입은 Relationship내에서 특별한 역할을 한다. 이를 Role이라 하며, 그림에서 $role_1$, ..., $role_i$, ..., $role_m$은 각각 $Entity_1$, ..., $Entity_i$, ..., $Entity_m$과 Relationship의 관계에 의한 역할들을 나타낸다.




(그림 2.1) E-R 모델 스킴

### 2.2 객체지향 데이터 모델

인스턴스 수준과 클래스 수준에서 집단화와 연관화 의미를 연구하기 위하여 객체, 메소드, 타입, 클래스와 부분 타입과 같은 기본 개념을 간략히 소개한다.

#### 2.2.1 객체와 메소드

객체는 실세계의 개체를 나타내며 식별자, 타입과 값으로 기술한다. 객체 식별자는 객체의 존재를 나타낸다. 값은 주어진 타입의 객체를 표현하는 것이다. 메소드 m은 ⟨mn, ms, mb⟩로 구성되며, mn은 메소드 이름, ms는 $f: S \times P_1 \times \cdots \times P_K \rightarrow T$로 표기되는 메소드 요약(signature)이며, 여기서 S는 메소드 m이 정의되는 초기 입력 인자의 타입이고, $P_i(i=1, \cdots, k)$는 메소드 인자들이다. 또한 T는 출력 타입이고, mb는 메소드 m의 구현과 의미를 규정하는 본체부(body)이다. 개념적으로 메소드는 한 타입과 연관된 함수로서 외부 사용자가 객체를 이용 시에는 인터페이스, 즉 메소드 요약을 통해서만 접속할 수 있다.

#### 2.2.2 타입과 클래스

타입은 그 용도와 행위에 따라서 객체들을 분류하기 위한 영역으로 사용된다. 타입의 네 가지 종류가 다음과 같이 규정된다.

(i) 기본타입 : string, integer, boolean, real, date, 등.
(ii) $\tau_1, \cdots, \tau_n$이 타입이고, $A_1, A_2, \cdots, A_n$이 구분이름이면, $(A_1:\tau_1, \cdots, A_n:\tau_n)$은 튜플타입이다.
(iii) $\tau$가 타입이면, $\{\tau\}$은 집합타입이다.
(iv) $\tau$가 타입이면, ref$\tau$는 참조타입이다.

D와 Oidset이 셀 수 있는 집합이라 하자. 각각 기본타입의 자체 값과 모든 객체 식별자들의 값을 나타낸다. T는 타입들의 집합이며, P(S)는 집합 S의 멱집합(power set)이다. T에 있는 $\tau$의 영역을 dom($\tau$)로 표기

한다.

(i) $\tau$가 기본타입 중 하나라면, dom($\tau$)는 그 타입의 원자 값의 집합이다. 즉 dom($\tau$)$\subseteq$D.
(ii) $\tau$가 집합 값($\tau=\{\sigma\}$)이면, dom($\tau$)는 P(dom($\sigma$))이며, dom($\tau$)$=\{v/v\subseteq dom(\sigma)\}$이다.
(iii) $\tau$가 튜플 값($A_1:\tau_1, \cdots, A_n:\tau_n$)이면, dom($\tau$)$=\{(A_1:o_1, \cdots, A_n:o_n)|o_i\in dom(\tau_i), i=1, \cdots, n\}$이다.
(iv) $\tau$가 참조타입(ref $\sigma$)이면, dom($\tau$)$\subseteq$Oidset은 타입 $\sigma$의 인스턴스 객체를 참조하는 각 객체 식별자의 집합이다. $\tau$가 ref $\{\sigma\}$이면, dom($\tau$)$\subseteq$(Oidset)이다.

O가 객체들의 전체집합이며 T가 타입들의 전체집합이라 하자. 일반적으로 $\tau\in$T에 대해서 O의 부분집합 $X_\tau$가 존재한다면, ($\tau$, $X_\tau$)은 $\tau$로 구현된 객체들의 클래스라 규정한다. $X_\tau$는 타입 $\tau$의 객체들을 명명한 집합이다. 그래서 한 타입은 하나 이상의 클래스를 가질 수 있다.

【정의 2.1】 객체지향 데이터베이스 스키마($d_o$)

$d_o=\{(\tau, X_\tau)|\tau\in T, X_\tau\subseteq O\}$ □

2.2.3 부분타입화와 부분타입 상속성

$\tau$와 $\sigma$가 T에 있는 타입이라면, $\tau\leq\sigma$로 표기되는 $\sigma$와 $\tau$의 부분타입화는 다음처럼 재귀적으로 정의된다.

(i) $\tau$가 기본타입이라면 $\tau\leq\tau$
(ii) $\tau\leq\sigma$이면 $\{\tau\}\leq\{\sigma\}$과 ref $\tau\leq$ref $\sigma$
(iii) $\tau_i\leq\sigma_i$, i=1, ..., n이면, $(A_1:\tau_1, \cdots, A_n:\tau_n)\leq(A_1:\sigma_1, \cdots, A_n:\sigma_m)(n\leq m)$

부분타입화는 부분 순서(partial order)를 이룬다. 루트 타입으로써 시스템 정의 타입인 *OBJECT*를 가질 때, 임의의 두 타입은 언제나 격자(lattice)에서 최소 상한 타입(least upper type)을 가지므로 이 순서는 타입의 반격자(semi-lattice)를 이룬다. 부분타입 상속은 타입화에 기반한 타입 합성기법이다. 그래서 타입 $\tau$에 의해서 규정된 모든 속성들이 타입 $\sigma$에서도 사용할 수 있다면($\tau\leq\sigma$), 타입 $\tau$로부터 타입 $\sigma$로 속성 및 메소드를 상속한다고 한다. 하나 이상의 상위 타입으로부터 한 서브타입이 규정된다면, 다중 상속이라 한다. 부분 타입화에 기반하여 부분 클래스 관계를 정의할 수 있다. 두 클래스 ($\sigma$, $X_\sigma$)와 ($\tau$, $X_\tau$) 사이에 부분 클래스 관계가 존재한다면, $\sigma$, $\tau\in$T에 대해서 $\tau\leq\sigma$(내포성)와 $X_\tau\leq X_\sigma$(외연성)가 성립한다.

2.2.4 집단화 타입 계층과 연관화 타입 계층

객체들은 "PART-OF" 관계에 의해서 다른 객체들과 합성될 수 있다. 이 "PART-OF" 관계에 의해서 결합된 객체의 집합을 합성객체라 한다. 합성객체는 원자 객체들로 부터 기본타입의 집합에 귀납적으로 적용하여 튜플과 집합 구성자들을 구성할 수 있다. 객체 $o_1, \cdots, o_n$과 식별가능 이름 $A_1, \cdots, A_n$이 주어진다면, 다음이 성립한다.

(i) $o=[A_1:o_1, \cdots, A_n:o_n]$은 튜플 객체이며, $o.A_j$는 이름 $A_j$에 대한 객체 O의 투사이다.
(ii) $o=\{o_1, \cdots, o_n\}$은 집합 객체이다.

객체의 여러 모델[11, 14, 15]들이 유사한 방법으로 제시되었다.

【전제조건 2.1】 O가 객체들의 집합이라면, 다음과 같은 이항 연산 $\diamond$가 존재한다.

(i) $x\diamond y$가 유효하다면, $\forall x, y\in O$, $x\diamond y\in O$이고, O는 $\diamond$에 닫혀있다.
(ii) $x\diamond x=x$, O에 있는 객체들은 $\diamond$하에서 멱등이 성립한다.
(iii) $x\diamond y=y\diamond x$이 성립한다(교환법칙).
(iv) $x\diamond(y\diamond z)=(x\diamond y)\diamond z$이 성립한다(결합법칙). □

위의 전제에서 세 가지 경우를 고려할 필요가 있다. (i) $x\diamond y=x$ (ii) $x\diamond y=y$, (iii) $x\diamond y=z$, 여기서 $z\neq x$, $z\neq y$이다. (i)와 (ii)의 경우에 y가 x사이의 결합은 "PART-OF" 관계에 의한 집단화를 의미하며 집단화 타입 계층(aggregation type hierarchy)을 이룬다. (iii)의 경우는 x와 y사이에 "HAS-A" 관계에 의한 연관화를 의미하며 연관화 타입 계층(association type hierarchy)을 이룬다.

【정의 2.2】 O가 객체들의 집합이며 x∈O이라 하자. y≠z 이고 x=y◇z가 되도록 하는 객체 y, z∈O가 존재한다면 x는 합성객체라 한다. 그렇지 않다면 x는 O에 대해서 단순 객체라 한다. □

【정의 2.3】 모든 x, y∈O에 대해서 x◇y=y가 성립하면, x는 y와 "PART-OF" 관계를 이룬다. □

【정의 2.4】 모든 x, y∈O에 대해서 x◇y=z가 성립하면, x는 y와 "HAS-A" 관계를 이룬다. □

【전제조건 2.2】 "PART-OF" 관계는 부분 순서 관계와 비순환 관계를 이룬다.

증명 : 정의 2.3에 의해서 "PART-OF" 관계는 각 객체에 대해서 반사성(x◇x=x), 반대칭성(x◇y=y와 x◇y=x이면, y=y◇x=y◇x=x)과 추이성(x◇y=y와 y◇z=z이면, x◇z=x◇(y◇z)=(x◇y)◇z=y◇z=z)이 존재한다. □

【전제조건 2.3】 "HAS-A" 관계는 비순서 관계와 순환 관계를 이룬다.

증명 : 정의 2.4에 의해서 "HAS-A" 관계는 각 객체에 대해서 반사성(x◇x=x)과 생성성(y◇x=z)이 존재하며, x와 y가 서로 연관 참조관계를 형성하기 때문에 순환한다. □

【정의 2.5】 Σ가 O에 대한 타입 시스템이고 T가 연관된 타입들의 집합이라 하자. 모든 $\tau$, $\sigma$∈T와 dom($\sigma$)에 y가 존재할 수 있는 모든 x∈dom($\tau$)에 대해서 x≠y이고 y◇x=x가 성립하도록 하는 $\sigma \sqsubset \tau$가 존재한다면, T에 대한 부분 순서는 집단화 참조가 된다. 그래서 $\tau$는 집단화 타입이라 하고 $\sigma$는 성분 타입이라 한다. □

【정의 2.6】 T가 타입의 집합이며, O가 타입 T에 연관된 객체의 집합이라 하자. O에 대한 집단화 타입 계층(T, ⊏)은 T로 표현하고 집단화 참조관계는 ⊏로 표현되는 타입구조이다. □

【정의 2.7】 T가 타입의 집합이며, O가 타입 T에 연관된 객체의 집합이라 하자. O에 대한 연관화 타입 계층(T, ⊏)은 T로 연관화 참조관계 ⊏로 표현되는 타입구조이다. □

타입 $\tau$와 $\sigma$ 사이에 집단화 참조($\sigma \sqsubset \tau$) 관계는 전제조건 2.2에 의해서 부분 순서관계를 형성하고 시스템 정의 타입인 OBJECT에 의해서 반격자 구조를 이룬다. 이 집단화 참조는 단방향으로 강한 결집력을 가진다. 그래서 집단화 참조에 기반한 타입 합성 기법은 성분 타입 $\sigma$의 속성들은 타입 $\tau$에 구문상(syntactic)으로 재사용할 수 있으며, 강제성(coercion)에 의한 다형성을 이룬다[15, 17]. 이런 상속성을 집단화 상속성(aggregation inheritance)이라 한다. 두 클래스 ($\sigma$, $X_\sigma$)와 ($\tau$, $X_\tau$) 사이에 집단화 관계가 존재한다면, $\sigma$, $\tau$∈T에 대해서 $\tau \le \sigma$(내포성)와 $X_\sigma \le X_\tau$(외연성)가 성립한다.

## 3. E-R 모델을 객체지향 모델로 변환

E-R 모델을 객체지향 모델로 변환할 때 E-R 모델의 각 성분을 변환하는 방법을 고려해야 한다. E-R 모델 측면에서는 개체, 개체 타입, 관계, 관계 타입, 역할, 속성, 값, 값의 집합을 거론하였으며, 객체지향 모델에서는 객체, 타입, 클래스, 일반화 관계, 집단화 관계, 연관화 관계, 메소드를 기술하였다.

E-R 모델은 정의역이 모두 원자 자료형만을 가지고, 집단화, 연관화와 일반화의 관계를 가지지 못하며, 단지 외래키에 의해서 정규화될 뿐이다. 또한 구성자(constructor)는 집합뿐이다. 객체지향 모델에서 관계는 객체의 인스턴스들 사이에 의미를 부여한 것이다. 이 의미는 2장에서 기술한 것처럼 일반화 및 세분화, 집단화, 연관화와 인스턴스화로 구분되며 관계의 특성은 〈표 3.1〉과 같다.

따라서 이러한 성분과 특성을 고려하여 각각의 변환을 기술하여 보면, E-R 모델의 개체 타입과 관계 타입은 개체 클래스 스킴과 관계 클래스 스킴으로 변환되며, 개체와 관계를 기술하는 속성들은 객체지향 모델의 클래스에서 원자 영역을 가지는 속성으로 변환되며, 속성-영역 관계가 집단화 타입 계층에 의해 형성되면, 속성은 그 관계의 역할에 따라 속성 이름이 부여되며, 영역은 성분 타입이 된다. 또한 속성-영

〈표 3.1〉 관계의 특성

| 특성 \ 종류 | | | 일반화/세분화 | 연관화 | 집단화 |
|---|---|---|---|---|---|
| 방향성 | | | 양방향: 상위클래스에서 하위클래스 로의 방향 (세분화), 하위 클래스에서 상위 클래스로의 방향 (일반화) | 양방향 : 독립적, 재귀적, 대칭적, 추이적, 순환적 | 단방향 : 종속적, 추이적, 반대칭적, 비순환적 |
| 제약조건 | 대응수 | | 1:1, 1:N, M:N, M:1 | 1:1, 1:N, M:N, M:1 | 1:1, 1:N |
| | 참여 완벽성 | total | $U_{i=1}^{n} S_i = G$ 여기서 $S_i$: 하위클래스 G: 상위클래스 $G=\{S_1,S_2,.....,S_n\}$ | domain: not null | domain: not null |
| | | partial | $U_{i=1}^{n} S_i \neq G$ | domain: null가능 | domain: null가능 |
| | 분리성 | disjoint | $S_i \cap S_j = \emptyset$, $i \neq j$ | (관련 없음) | (관련 없음) |
| | | over-lapping | $S_i \cap S_j \neq \emptyset$, $i \neq j$ | (관련 없음) | (관련 없음) |
| 속성과 영역의 관계 | 배타성 | 공유 | (관련 없음) | 한 정의역을 여러 속성이 참조 | 좌동 |
| | | 배타 | (관련 없음) | 한 정의역을 한 속성만 참조 | 좌동 |
| | 종속성 | 독립 | (관련 없음) | 상위 클래스로부터 속성/메소드 상속 없음 | 좌동 |
| | | 종속 | (관련 없음) | 상위클래스로부터 속성/메소드 상속 있음 | 좌동 |
| 다형성 | | | 전체다형성 | (관련 없음) | 강제다형성 |

역 관계가 연관화 타입 계층에 의해 형성되면, 속성은 그 관계의 역할에 따라 속성 이름이 부여되고, 연관화 타입이 영역이 된다. 이에 대한 변환 규칙은 다음과 같다.

【변환 규칙 3.1】

ⓐ E-R 모델에서 개체와 관계의 특성을 기술하는 속성들은 객체의 원자속성으로 변환된다.

ⓑ 개체간의 관계 의미가 일반화일 때는 포함 다형성(inclusion polymorphism)과 대치 다형성(substitution polymorphism)[14, 15]에 의해서 메소드로 변환된다.

ⓒ 개체간의 관계 의미가 집단화일 때는 합성객체(composite object)로 변환되고, 강제 다형성(coercion polymorphism)[14, 15]에 의하여 집단화 상속성을 지원하며 메소드로 변환된다.

ⓓ 개체간의 관계 의미가 연관화일 때는 연결 속성(link attribute)으로 변환된다.

ⓔ 약 개체(weak entity) 타입은 성분 클래스(component class)로 변환된다.

ⓕ E-R 모델에 없는 메소드를 추가한다.

ⓖ 관계의 역할이 다른 객체를 참조할 때는 속성의 영역이 복합객체가 되도록 하는 튜플과 집합 구성자를 추가한다.

이 논문에서는 객체지향 모델의 개념으로서 E-R 모델의 관계 타입, 역할과 속성을 이용할 수 있다. 변환 규칙에 따라 개체 타입은 클래스 스킴으로 변환되며, 개체 타입간의 일반화는 타입 계층으로 변환된다. E-R 모델의 관계는 의미에 따라 집단화와 연관화로

구분되며, 집단화 참조는 집단화 타입 계층을 이루며, 연관화 참조는 연관화 타입 계층을 이룬다. E-R 모델에서 개체 타입간의 관계를 객체지향 모델에서 속성과 영역으로 변환하기 위해서는 관계의 역할이 속성이 되며, 영역은 참조되는 개체 타입이 된다.

변환 규칙 3.1 내용에서 관계의 의미(semantics)는 임의의 개체들 사이의 결합성에 의해 집단화 추상화와 연관화 추상화로 구분할 수 있다. 일반화는 상위 타입과 하위 타입의 관계에서 상위 타입의 속성들을 포함 다형성과 대치 다형성에 의해서 재사용할 수 있는 일반화 상속성을 도입할 수 있다. 집단화 추상화는 관계의 의미가 두 개체 사이에 합성-성분 관계에 있을 때를 의미하며, 두 개체 사이에는 단방향성 관계, 종속적 관계와 추이적 관계만을 가지기 때문에 강한 결집력을 가진다. 이 강한 결집력으로 성분 타입의 속성들을 합성 타입에서 재사용할 수 있는 집단화 상속성을 도입할 수 있다. 개체 사이의 관계에 대한 의미가 독립적이며, 두 개체 사이에는 양방향성 관계, 반사적, 생성적 관계를 가지기 때문에 약한 결집력을 가진다. 그래서 관계에 대한 변환에 추상화를 도입하고, 집단화 타입 계층과 연관화 타입 계층의 속성-영역 관계를 형성하기 위한 역할의 형식화를 제시한다.

### 3.1 역할의 형식화

E-R 모델의 n-항 관계 $R(E_1, ..., E_i, ..., E_n)$에서 한 객체 타입 $E_i$와 다른 개체 타입 $E_j$ 사이의 관계를 객체지향 모델의 집단화 관계 $R_{ag}(C_1, ..., C_i, ..., C_n)$와 연관화 관계 $R_{as}(C_1, ..., C_i, ..., C_n)$로 변환된다. 이 집단화 관계와 연관화 관계에서 해당 추상화 자료 타입에 대한 한 클래스 $C_i$와 다른 클래스 $C_j$가 속성-영역 관계를 이루기 위해서는 E-R 모델에서 관계의 역할에 따라 규정된다. 이 역할을 객체지향 모델로 변환하기 위해서 메소드로 구현할 수 있다. 메소드로 변환하기 위해서는 역할을 객체지향 모델의 속성과 영역 관계에 도입하여 일반식을 유도하여야만 가능하다. 이를 위해 한 클래스 $C_i$와 다른 클래스 $C_j$가 속성-영역 관계를 2.2.1 절에서 규정된 메소드 요약을 이용하여 표현하면 다음과 같이 된다.

$$C_i \text{ 속성이름}(C_i) \rightarrow C_j \quad (1)$$

여기서 수신 클래스 $C_i$는 속성의 클래스가 되고, selector인 속성이름은 E-R 모델의 역할이 되며, 입력 클래스도 수신 클래스와 동일하며, 그리고 영역에 해당하는 반환 클래스는 $C_j$가 된다. 그래서 역할을 selector로 표현하면 위 식(1)은 다음과 같이 된다.

$$C_i \text{ role-name}_k(C_i) \rightarrow C_j,\ 1 \le i \le m,\ 1 \le j \le n,\ k \ne i \ne j \quad (2)$$

#### 3.1.1 일반화/세분화 타입 계층에 의한 속성-영역

일반화 클래스인 경우에 역할을 selector로 표현하면 다음과 같다.

$C_i \text{ role-name}_k(C_i) \rightarrow C_j$
($C_i$는 하위클래스, $C_j$는 상위클래스, $1 \le i \le m$, $1 \le j \le n$, $1 \le k \le n-1$, $k \ne i \ne j$)

여기서 역할의 추상화에 의거 role-name$_k$는 "IS-A"가 된다. 또한 세분화 클래스인 경우에 역할을 selector로 표현하면 다음과 같다.

$C_i \text{ role-name}_k^{-1}(C_i) \rightarrow C_j$
($C_i$는 하위클래스, $C_j$는 상위클래스, $1 \le i \le m$, $1 \le j \le n$, $1 \le k \le n-1$, $k \ne i \ne j$)

여기서 역할의 추상화에 의거 role-name$_k$는 IS-A$^{-1}$가 된다

#### 3.1.2 연관화 타입 계층에 의한 속성-영역

연관화 참조의 특징은 전제조건 2.3에 의해서 양방향성, 재귀성, 생성성을 가진다. 양방향성에 의해서 수신 클래스가 임의의 한 클래스로 고정되지 않기 때문에 순환을 형성한다. 그래서 연관화 관계 $R_{as}(C_1, ..., C_i, ..., C_n)$에서 임의의 한 클래스 $C_i$에 대한 속성-영역 관계의 일반식은 다음과 같다. 연관화 경우 수의 총합은 $n((k_{11}+k_{12}+...+k_{1n})+(k_{21}+k_{22}+...+k_{2n})+...+(k_{n1}+k_{n2}+...+k_{nn}))$이다.

$$\begin{aligned}
&C_1 \text{ role-name}_{11o1}(C_1) \rightarrow C_1, \text{ 재귀적}, 1 \le o_1 \le k_{11}\\
&C_1 \text{ role-name}_{11p1}(C_1) \rightarrow C_2, \quad 1 \le p_1 \le k_{12}\\
&\quad\vdots\\
&C_1 \text{ role-name}_{11q1}(C_1) \rightarrow Cn, \text{ 재귀적}, 1 \le q_1 \le k_{1n}
\end{aligned}$$

연관화 경우의 수는 $n(k_{11}+k_{12}+\ldots+k_{1n})$이고

$C2\ role\text{-}name_{21o2}(C2) \rightarrow C_1, \quad 1 \le o_2 \le k_{21}$
$C2\ role\text{-}name_{21p2}(C2) \rightarrow C2$, 재귀적, $1 \le p_2 \le k_{21}$
:
$C2\ role\text{-}name_{21q2}(C2) \rightarrow Cn, \quad 1 \le q_2 \le k_{2n}$
연관화 경우의 수는 $n(k_{21}+k_{22}+\ldots+k_{2n})$이고
:
$Cn\ role\text{-}name_{n1o1}(Cn) \rightarrow C1, \quad 1 \le o_n \le k_{n1}$
$Cn\ role\text{-}name_{n2p2}(Cn) \rightarrow C2, \quad 1 \le p_n \le k_{n1}$
:
$Cn\ role\text{-}name_{nnqn}(Cn) \rightarrow Cn$, 재귀적, $1 \le q_n \le k_{nn}$
연관화 경우의 수는 $n(k_{n1}+k_{n2}+\ldots+k_{nn})$이다.

연관화는 역함수가 존재하기 때문에 다음과 같이 위의 식이 변환될 수 있다.

$C1\ role\text{-}name_{11o1}{}^{-1}(C1) \rightarrow C1$, 재귀적, $1 \le o_1 \le k_{11}$
$C1\ role\text{-}name_{11p1}{}^{-1}(C2) \rightarrow C1, \quad 1 \le p_1 \le k_{12}$
:
$C1\ role\text{-}name_{11q1}{}^{-1}(Cn) \rightarrow C1$, 재귀적, $1 \le q_1 \le k_{1n}$
연관화 경우의 수는 $n(k_{11}+k_{12}+\ldots+k_{1n})$이고
:
$C2\ role\text{-}name_{21o2}{}^{-1}(C1) \rightarrow C2, \quad 1 \le o_1 \le k_{21}$
$C2\ role\text{-}name_{21p2}{}^{-1}(C2) \rightarrow C2$, 재귀적, $1 \le p_2 \le k_{21}$
:
$C2\ role\text{-}name_{21q2}{}^{-1}(Cn) \rightarrow C2, \quad 1 \le q_2 \le k_{2n}$
연관화 경우의 수는 $n(k_{21}+k_{22}+\ldots+k_{2n})$이고
:
$Cn\ role\text{-}name_{n1o1}{}^{-1}(C1) \rightarrow Cn, \quad 1 \le o_n \le k_{n1}$
$Cn\ role\text{-}name_{n2p2}{}^{-1}(C2) \rightarrow Cn, \quad 1 \le p_n \le k_{n1}$
:
$Cn\ role\text{-}name_{nnqn}{}^{-1}(Cn) \rightarrow Cn$, 재귀적, $1 \le q_n \le k_{nn}$
연관화 경우의 수는 $n(k_{n1}+k_{n2}+\ldots+k_{nn})$이다.

### 3.1.3. 집단화 타입 계층에 의한 속성-영역

집단화 클래스인 경우에는 집단화 참조의 부분 순서성에 의해서 수신 클래스가 임의의 한 클래스로 고정되며, 재귀적 운행과 순환을 형성하지 못한다. 그래서 집단화 관계 $R_{ag}(C_1, \ldots, C_i, \ldots, C_n)$에서 합성 클래스가 $C_i(1 \le i \le n)$라면, 속성-영역 관계의 일반식은 다음과 같다.

$C1\ role\text{-}name_{11o1}(C1) \rightarrow C1$, 불가, $1 \le o_1 \le k_{11}$
$C1\ role\text{-}name_{11p1}(C1) \rightarrow C2, \quad 1 \le p_1 \le k_{12}$
:
$C1\ role\text{-}name_{11q1}(C1) \rightarrow Cn$, 재귀적, $1 \le q_1 \le k_{1n}$
집단화 경우의 수는 $(n-1)(k_{12}+k_{13}+\ldots+k_{1n})$이고, 단 $k_{11}$은 재귀적 참조가 불가하므로 제외된다.

$C2\ role\text{-}name_{21o2}(C2) \rightarrow C2, \quad 1 \le o_1 \le k_{21}$
$C2\ role\text{-}name_{21p2}(C2) \rightarrow C2$, 불가, $1 \le p_2 \le k_{21}$
:
$C2\ role\text{-}name_{21q2}(C2) \rightarrow Cn, \quad 1 \le q_2 \le k_{2n}$
집단화 경우의 수는 $(n-1)(k_{21}+k_{23}+\ldots+k_{2n})$이고, 단 $k_{22}$은 재귀적 참조가 불가하므로 제외된다.
:
$Cn\ role\text{-}name_{n1On}(Cn) \rightarrow C1, 1 \le o_n \le k_{n1}$
$Cn\ role\text{-}name_{n2pn}(Cn) \rightarrow C2, 1 \le p_n \le k_{n1}$
:
$Cn\ role\text{-}name_{nnqn}(Cn) \rightarrow Cn$, 불가, $1 \le q_n \le k_{nn}$
집단화 경우의 수는 $(n-1)(k_{n1}+k_{n2}+\ldots+k_{nn-1})$이고, 단 $k_{nn}$은 재귀적 참조가 불가하므로 제외된다. 또한 집단화는 역함수가 존재하지 않는다.

### 3.2 관계 변환

객체지향 모델에서 데이터 추상화란 새로운 타입의 객체를 데이터 구조와 같은 구현 세부 사항을 고려하지 않고 추상적이고 개념적인 행위의 기술만으로 정의할 수 있는 기법을 말하며, 대부분의 객체지향 언어는 객체가 외부적으로 정의된 연산에 의해서만 조작되게 함으로서 데이터 추상화를 지원한다[15]. 그래서 의미에 따라 E-R 모델에서 개체들 사이의 관계 타입을 객체지향 모델에서는 다음과 같이 표현할 수 있다.

【정의 3.1】 관계 추상성(R)

$R = (Rge, Rag, Ras, Ris)$

여기서 관계는 해당 추상 자료 타입에 대한 클래스 수준이나 한 클래스의 객체 인스턴스에 기반한 객체 수준 사이에 연결되는 의미에 따라 추상성이 부여되

는 것으로써 일반화 관계성(Rge), 집단화 관계성(Rag), 연관화 관계성(Ras) 그리고 인스턴스화 관계성(Ris)으로 구분된다. □

위 관계 추상성을 구체적으로 기술하면 다음과 같이 된다.

가. 일반화 관계성

객체 수준에서의 관계성은 이미 상속을 받았기 때문에 관계성을 가지지 않는다. 클래스 수준 관계성은 클래스 사이에 계층을 이루기 때문에 클래스 계층(class hierarchy: Rgec)을 형성한다. 이는 임의의 두 클래스 $CO_i$와 $CO_j$에 대해서 $CO_i$가 $CO_j$의 특성 및 메소드를 계승받음을 의미하며, $CO_i$ Rgeo $CO_j$로 표기한다. 역으로 상위클래스에서 하위 클래스로 세분화하는 과정을 세분화 관계성이라 한다.

나. 집단화 관계성

객체 수준에서의 관계성(Rago)은 임의의 두 객체 $o_i$와 $o_j$에 대해서 $o_i$의 성분 관계에 의해서 $o_j$가 강하게 결합되어 종속적으로 운행되며, $o_i$ Rago $o_j$로 표기한다. Rago는 비대칭성과 이행성을 가진다. 클래스 수준에서의 관계성(Ragc)은 합성 클래스가 성분 클래스를 집단적으로 참조하는 합성 클래스 계층(aggregation class hierarchy)을 형성한다. 이는 임의의 두 클래스 $CO_i$와 $CO_j$에 대해서 $CO_i$가 $CO_j$에 집단화 관계성에 의해 $CO_j$가 강하게 결합되어 종속적으로 운행되며, $CO_i$ Rgeo $CO_j$로 표기한다. 강한 결속력을 가진 합성 클래스로부터 속성과 메소드를 상속받을 수 있기 때문에 이를 집단화 상속성이라 규정한다[14, 15].

다. 연관화 관계성

객체 수준에서의 관계성(Raso)은 임의의 두 객체 $o_i$와 $o_j$에 대해서 $o_i$의 연관 관계에 의해서 $o_j$가 느슨하게 결합되어 운행되며, $o_i$ Raso $o_j$로 표기한다. Raso는 반사적이며, 대칭성과 이행성을 가진다. 클래스 수준에서의 관계성(Rasc)은 한 클래스가 다른 클래스를 서로 양방향으로 연관성을 이루기 때문에 연관 클래스 계층(association class hierarchy)을 형성한다. 이는 임의의 두 클래스 $CO_i$와 $CO_j$에 대해서 $CO_i$가 $CO_j$에 연관화 관계성에 의해 $CO_j$가 느슨하게 결합되어 운행되며, $CO_i$ Rasc $CO_j$로 표기한다.

라. 인스턴스화 관계성

임의의 객체 o와 클래스 C에 대해서 o가 C의 인스턴스임을 의미하며, o Ris C로 표기한다. 만일 이 관계성이 임의의 두 클래스 $CO_i$와 $CO_j$에 대해서 성립할 경우, $CO_i$ Ris $CO_j$로 표기한다. $CO_i$는 $CO_j$의 메타 클래스가 된다.

3.2.1 집단화 추상성

집단화는 성분 클래스들을 튜플 구성자들에 의해 합성 클래스(composite class)로 추상화하는 방법이다. 객체 사이에 속성과 영역 관계를 가질 때 이를 합성 참조(composite reference)라 한다. 이 방법은 공유 참조 여부에 따라 독점 참조와 공유 참조로 구분되며, 그 성분 클래스가 상위 클래스로부터 상속성 여부에 따라 의존 참조와 독립 참조로 구분할 수 있다[10]. 이에 대한 합성 참조에 의한 구분은 다음과 같이 4가지이다. 배타 독립 합성 참조(exclusive independent composite reference)는 하나의 객체 x만이 객체 y를 독점적으로 참조하며, y가 상위 객체 y′로부터 속성 및 메소드를 상속받지 않는다. 공유 독립 합성 참조(shared independent composite reference)는 여러 개의 객체들 $x_i(1 \le i \le n)$이 객체 y를 공유하면서 참조하며, y가 상위 객체 y′로부터 속성 및 메소드를 상속받지 않는다. 배타 종속 합성 참조(exclusive dependent composite reference)는 하나의 객체 x만이 객체 y를 독립적으로 참조하며, y가 상위 객체 y′로부터 속성 및 메소드를 상속받는다. 또한 공유 종속 합성 참조(shared dependent composite reference)는 여러 개의 객체들 $x_i(1 \le i \le n)$이 객체 y를 공유하면서 참조하며, y가 상위 객체 y′로부터 속성 및 메소드를 상속받는다. 이 집단화 참조에 의한 특징은 2.2.4절에서 규정한 것처럼 개체와 개체 사이에 성분(부품)과 합성(조립) 관계를 가지기 때문에 단방향성 속성 영역 관계를 형성한다. $E_i$가 합성 개체이고, $E_j$가 성분 개체이며, $R_i$가 집단화 추상성을 가지는 관계라면, 클래스 수준에서의 클래스 정의어와 인스턴스 수준에서의 각 객체의 구성에 대해서 다음과 같이 기술할 수 있다.

가. 클래스 수준

$E_i$가 합성 개체이고 $E_j$가 성분 개체라면, 관계의 역할은 집단화 관계를 의미하므로 단방향성 속성-영역 관계를 형성한다. 변환 절차 3.1에 의해 개체 $E_i$와 $E_j$의 속성들은 원자-영역이기 때문에 객체지향 모델의 해당 클래스에서 그대로 사용되며, 더불어 메소드가 추가된다. 이와 같은 변환 결과를 클래스 수준 관계성과 인스턴스 수준 관계성에 따라 각각 (그림 3.1)과 (그림 3.2)와 같이 표현한다.

```
class CEi
{  ai1 : {atomic domain(D)}
   :
   aim: {atomic domain(D)}
   mi1(): {C/D}
   :
   min():{C/D}
   ag: {[exclusive dependant]|[exclusive independent]|
        [shared dependent]|[shared independent]|component domain(CEj)}}
```

(a) 합성-클래스

```
class CEj
{  aj1: {D}
   :
   ajo: {D}
   mj1(): {C/D}
   :
   mjp():{C/D}}
```

(b) 성분-클래스

```
class CR
{   ra1: {D}
    :
    rar: {D}
    rm1(): {C/D}
    :
    rms(): {C/D}
    composi: {ref CEi}
    compona: {ref CEj}}
```

(c) 관계- 클래스

(그림 3.1) 집단화 추상성을 가지는 클래스 구조

위 (그림 3.1)에서 (a)는 합성 클래스의 구문을 나타내고, 여기서 $a_{i1}$, ..., $a_{im}$은 원자 정의역을 참조하는 원자속성들이며, $m_{i1}$, ..., $m_{in}$은 원자 정의역이나 추상 클래스를 반환하는 메소드들이며, ag는 상위 클래스로부터 상속성과 공유성의 여부에 따라 4 가지 구분에 의해 성분 클래스를 참조하는 집단화 속성이다. (b)는 성분 클래스의 구문을 나타내고, $a_{j1}$, ..., $a_{jm}$은 원자 정의역을 참조하는 원자속성들이며, $m_{j1}$, ..., $m_{jn}$은 원자 정의역이나 추상 클래스를 반환하는 메소드들이다. 그러나 성분 클래스가 성분 클래스가 될 수 없는 단방향성으로 인해서 성분 클래스는 집단화 속성을 가지지 않는다. (c)는 관계 클래스의 구문을 나타내고, $ra_1$, ..., $ra_r$은 원자 정의역을 참조하는 원자속성들이며, $rm_1$, ..., $rm_s$는 원자 정의역이나 추상 클래스를 반환하는 메소드들이며, composi는 합성 클래스를 간접 참조하는 합성 속성이며, compona는 성분 클래스를 간접 참조하는 성분 속성이다. 여기서 **"ref"**는 간접 참조를 나타내기 위한 키워드이며, "{}"는 참조되는 값이나 객체가 집합임을 나타낸다.

### 나. 인스턴스 수준

집단화 추상성에 의한 클래스 수준에 기반해서 생성되는 객체들은 다음과 같이 표현될 수 있다.

$id_i(CE_i)$

| $a_{i1}$ | .. | $a_{im}$ | $m_{i1}()$ | .. | $m_{in}()$ | ag |
|---|---|---|---|---|---|---|
| $\{v_{i1}\}$ | .. | $\{v_{im}\}$ | $\{mv_{i1}/o_{i1}\}$ | .. | $\{mv_{in}/o_{in}\}$ | $\{id_j\}$ |

(a) 합성클래스에 의한 외연

$id_j(CE_j)$

| $a_{j1}$ | .. | $a_{jn}$ | $m_{j1}()$ | .. | $m_{jp}()$ |
|---|---|---|---|---|---|
| $\{v_{j1}\}$ | .. | $\{v_{jn}\}$ | $\{mv_{j1}/o_{j1}\}$ | .. | $\{mv_{jp}/o_{jp}\}$ |

(b) 성분클래스에 대한 외연

$id_r(CR)$

| $ra_1$ | .. | $ra_r$ | $rm_1()$ | .. | $rm_s()$ | composi | compona |
|---|---|---|---|---|---|---|---|
| $\{rv_1\}$ | .. | $\{rv_r\}$ | $\{rmv_1/o_1\}$ | .. | $\{rmv_s/0_s\}$ | {ref $id_i$} | {ref $id_j$} |

(c) 관계클래스에 대한 외연

(그림 3.2) 집단화 추상화에 의한 클래스 외연

위 (그림 3.2)는 (그림 3.1)의 각 클래스 구문에 의해 생성되는 객체를 표현하기 위한 것으로서 (a)는 합성 객체를 나타내고, 여기서 원자 속성 $a_{i1}$, ..., $a_{im}$ 의해 참조되는 값들은 각각 $v_{i1}$, ..., $v_{im}$이며, 메소드 $m_{i1}$, ..., $m_{in}$에 의해 반환되는 값 및 객체는 각각 $mv_{i1}/o_{i1}$, ..., $mv_{in}/o_{in}$이며, 집단화 속성 ag에 의해 참조되는 성분 객체의 식별자는 $id_j$이다. (b)는 성분 객체를 나타내고, 여기서 원자 속성 $a_{j1}$, ..., $a_{jm}$ 의해 참조되는 값들은 각각 $v_{j1}$, ... $v_{jm}$이며, 메소드 $m_{j1}$, ..., $m_{jn}$에 의해 반환되는 값 및 객체는 각각 $mv_{j1}/o_{j1}$, ..., $mv_{jn}/o_{jn}$이다. (c)는 관계 객체를 나타내고, 여기서 원자 속성

$ra_1$, ..., $ra_r$에 참조되는 값들은 각각 $rv_{i1}$, ..., $rv_{ir}$이며, 메소드 $rm_1$, ..., $rm_s$에 의해 반환되는 값 및 객체는 각각 $rmv_{i1}/o_1$, ..., $mv_{in}/o_s$이다. 또한 합성 속성 composi에 의해 간접 참조되는 표현은 **ref** $id_i$이며, 성분 속성 componа에 의해 간접 참조되는 표현은 **ref** $id_j$이다.

### 3.2.2 연관화 추상성

2.2절에서 규정한 것처럼 한 클래스가 다른 클래스를 서로 쌍방향으로 연관성을 이루기 때문에 느슨하게 결합되어 있으며, 두 객체 및 클래스는 반사적, 대칭성과 이행성을 가진다. 그래서 클래스 수준과 객체 수준에 대한 내포와 외연은 다음과 같다.

#### 가. 클래스 수준

개체 $E_i$와 $E_j$가 서로 연관화 추상성의 관계를 가질 경우, 양방향성 속성-영역 관계를 형성하므로 집단화 추상성에서 객체지향 모델로 변환할 때 역참조 관계가 표시되며, 원자-영역에 대한 특성을 집단화 추상성과 같은 방법으로 변환된다. 그 결과는 (그림 3.3)(클래스 수준)과 (그림 3.4)(인스턴스 수준)와 같이 된다.

```
class CEi
{  ai1 : {D}
   :
   aim: {D}
   mi1():{D/C}
   :
   min():{D/C}
   asi: {ref CEj}}
```

(a) 연관-클래스

```
class CEj
{  aj1: {D}
   :
   ajo: {D}
   mj1():{D/C}
   :
   mjo():{D/C}
   asj: {ref CEi}  // asi 와 asj는 역참조 관계 // }
```

(b) 연관-클래스

```
class CR
{   ra1: {D}
    :
    rar: {D}
    m1f():{D/C}
    :
    mjo():{D/C}
    assoi: ref {CEi}
    assoj: ref {CEj}}
```

(c) 관계-클래스

**(그림 3.3) 연관화 추상성을 가지는 클래스 구조**

위 (그림 3.3)에서 (a)는 $CE_i$를 연관화 클래스라 하고 $CE_j$를 연관화 참조되는 클래스라 할 경우의 클래스 구문을 나타내고, 여기서 $a_{i1}$, ..., $a_{im}$은 원자 정의역을 참조하는 원자속성들이며, $m_{i1}$, ..., $m_{in}$은 원자 정의역이나 추상 클래스를 반환하는 메소드들이며, $as_i$는 연관화되는 클래스 $CE_j$를 간접 참조하는 연관화 속성이다. (b)는 (a)와 역참조 관계를 형성하는 클래스의 구문을 나타내고, 여기서 $a_{j1}$, ..., $a_{jm}$은 원자 정의역을 참조하는 원자속성들이며, $m_{j1}$, ..., $m_{jn}$은 원자 정의역이나 추상 클래스를 반환하는 메소드들이다. $as_j$는 연관화되는 클래스 $CE_i$를 간접 참조하는 연관화 속성이다. (c)는 관계 클래스의 구문을 나타내고, 여기서 $ra_1$, ..., $ra_r$는 원자 정의역을 참조하는 원자속성들이며, $rm_1$, ..., $rm_s$는 원자 정의역이나 추상 클래스를 반환하는 메소드들이며, assoi와 assoj는 각각 연관화 클래스 $CE_i$와 $CE_j$를 간접 참조하는 연관화 속성들이다.

#### 나. 인스턴스 수준

인스턴스 수준은 (그림 3.3)의 연관화 추상성 클래스의 각 객체를 집합화하여 표현한 것으로 각 값은 정의역과 참조하는 클래스의 식별자를 나타낸다.

$id_i(CE_i)$

| $a_{i1}$ | .. | $a_{im}$ | $m_{i1}()$ | .. | $m_{in}()$ | $as_i$ |
|---|---|---|---|---|---|---|
| $\{v_{i1}\}$ | .. | $\{v_{im}\}$ | $\{mv_{i1}/o_{i1}\}$ | .. | $\{mv_{in}/o_{in}\}$ | {ref $id_j$} |

(a) 연관-클래스 $CE_i$에 의한 외연

$id_j(CE_j)$

| $a_{j1}$ | .. | $a_{jm}$ | $m_{j1}()$ | .. | $m_{jn}()$ | $as_j$ |
|---|---|---|---|---|---|---|
| $\{v_{j1}\}$ | .. | $\{v_{jm}\}$ | $\{mv_{j1}/o_{j1}\}$ | .. | $\{mv_{jn}/o_{jn}\}$ | {ref $id_j$} |

(b) 연관-클래스 $CE_j$에 대한 외연

$id_r(CR)$

| $ra_1$ | ... | $ra_r$ | $rm_1()$ | ... | $rm_s()$ | $asso_i$ | $asso_i$ |
|---|---|---|---|---|---|---|---|
| $\{rv_1\}$ | ... | $\{rv_r\}$ | $\{rmv_1/o_1\}$ | ... | $\{rmv_s/0_s\}$ | {ref $id_i$} | {ref $id_j$} |

(c) 관계-클래스 CR에 대한 외연

**(그림 3.4) 연관 추상화에 의한 클래스 외연**

위 (그림 3.4)는 (그림 3.3)의 각 클래스 구문에 의해 생성되는 객체를 표현하기 위한 것으로서 (a)는 클래스 $CE_i$에 대응하는 연관화 객체를 나타내고, 여기서

원자 속성 $a_{i1}$, ..., $a_{im}$ 의해 참조되는 값들은 각각 $v_{i1}$, ..., $v_{im}$이며, 메소드 $m_{i1}$, ..., $m_{in}$에 의해 반환되는 값 및 객체는 각각 $mv_{i1}/o_{i1}$, ..., $mv_{in}/o_{in}$이며, 연관화 $as_i$에 의해 간접 참조되는 연관화 객체의 표현은 **ref** $id_j$이다. (b)는 클래스 $CE_j$에 대응하는 연관화 객체를 나타내고, 여기서 원자 속성 $a_{j1}$, ..., $a_{jm}$ 의해 참조되는 값들은 각각 $v_{j1}$, ... $v_{jm}$이며, 메소드 $m_{j1}$, ..., $m_{jn}$에 의해 반환되는 값 및 객체는 각각 $mv_{j1}/o_{j1}$, ..., $mv_{jn}/o_{jn}$이다. 또한 연관화 $as_j$에 의해 간접 참조되는 연관화 객체의 표현은 **ref** $id_j$이다. (c)는 관계 객체를 나타내고, 여기서 원자 속성 $ra_1$, ..., $ra_r$에 참조되는 값들은 각각 $rv_{i1}$, ..., $rv_{ir}$이며, 메소드 $rm_1$, ..., $rm_s$에 의해 반환되는 값 및 객체는 각각 $rmv_{i1}/o_1$, ..., $rmv_{in}/o_s$이며, 연관화 속성 $asso_i$와 $asso_j$에 의해 연관화 객체들을 간접 참조하는 표현은 각각 **ref** $id_i$와 **ref** $id_j$이다. 따라서 집단화 추상성의 클래스 (그림 3.1)과 (그림 3.2)와의 차이는 (그림 3.3(b))와 (그림 3.4(b))의 연관 클래스에 양방향성을 표현하는 참조 속성 asj를 추가하는 것이다.

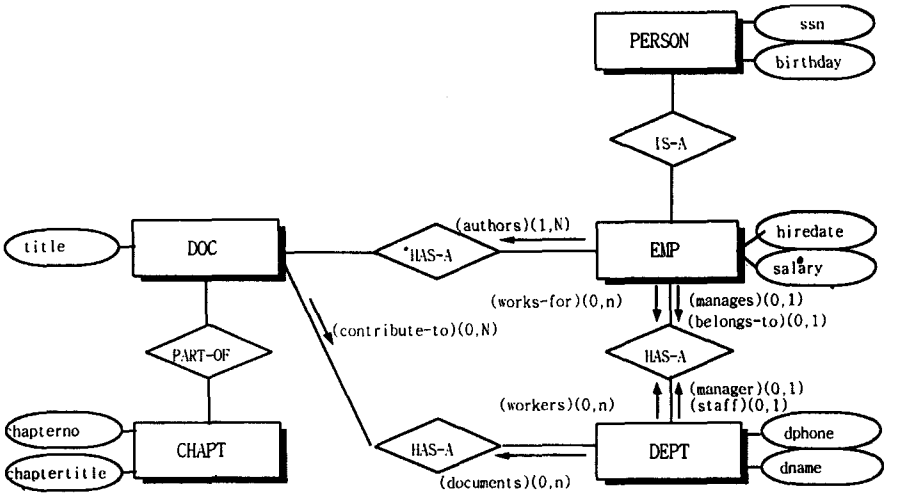
### 3.3 변환 예

두 개체가 어떤 사건에 의하여 연결되었을 때, 연결되는 의미에 따라 일반화, 집단화와 연관화에 분류되며, 그 참조하는 방향에 따라 순환성과 비순환성을 이룬다. 그래서 이 절에서는 의미망 모델의 특성을 도입하여 객체지향 모델로 변환하기 위한 E-R 모델을 제시하고(예 3.1), 변환된 이후의 객체지향 모델은 객체지향 질의 언어[13, 14, 16]를 이용하여 스키마를 작성한다(예 3.2).

【예 3.1】객체지향 모델에 대한 변환 기법의 실제 예를 보이기 위해서 E-R 모델의 표현은 다음과 같다.

(그림 3.5)에서 객체지향 모델에 1:1 대응 관계를 용이하게 하기 위해 KUPER[12]가 제안한 개체간의 관계 타입의 의미를 관계의 이름과 화살표로 표기하여 각 개체 타입의 속성을 기술하였으며, 이 논문에서 제안한 개체 타입의 속성과 개체간의 관계에 대한 의미를 일반화(IS-A), 집단화(PART-OF)와 연관화(HAS-A) 관계로 구분하였다. 또한 관계의 제약조건인 대응수제약조건과 참여제약조건을 관계 이름의 후미에 괄호(예로(1, N) 등)로 표기하였다.

이 논문에서 제안한 방법은 관계 타입에서 속성들을 가지지 않을 경우를 고려하지 못하였으므로 관계 클래스를 생성하지 못한다. 그러나 관계 클래스를 생성치 않으면 E-R 모델과 객체지향 모델이 서로 1대1 대응 관계가 성립하지 않기 때문에 완벽하게 변환할 수 없다. 따라서 이를 보완하여 완벽한 변환을 이루기 위해서 관계 타입에 일반화, 집단화와 연관화라는 의미망 모델의 특성을 도입하여 E-R모델을 구성하면 객체지향 모델로 변환할 수 있다.




(그림 3.5) E-R모델 스키마

다음 예 3.2는 E-R 모델을 객체지향 모델로 변환된 결과이다.

【예 3.2】 클래스 계층, 집단화 클래스 계층과 연관화 클래스 계층을 포함하는 스키마를 기술하기 위해서 예를 들어 설명한 것이다. tuple, set과 list는 여러 타입 구성자들을 규정하기 위해서 사용되고 있다. 주석은 양방향으로 관계를 기술하므로 서로 역(inverse)인 참조 속성(reference attribute)을 구분하고 있다.

(그림 3.6)의 객체지향 스키마에서 클래스 Person과 클래스 Emp는 일반화 상속 관계(IsA)를 이루고, 클래스 Doc와 클래스 Chapt는 서로 집단화 상속 관계(Constituent_of)를 이루며 이 계층에서 집단화 참조는 단방향성 특징을 가진다. 또한 클래스 Emp와 클래스 Dept는 연관화 참조 관계(ref)를 형성하며, 관계의 역할에 따라 속성이름을 부여하여 양방향성을 나타내고 있다.

```
class Person
  string pname;
  birthday;
end class Person;

class Emp
  IsA: Person;
  string ssn;
  string name;
  real salary;
  ref Dept manages; //inverse of dept. manager
  ref Dept belongs-to; //inverse of Dept.Staff
                       //inverse of Dept.workers
  set<tuple(ref Dept dept-of, real Hours)> works-for;
  method void give-raise(percent:real)
end class Emp;

class Doc
   integer docno;
   string title;
   set< ref Emp >authors;
   ref Dept contribute_to; //inverse of Dept.documents
   set<Chapt>;
end class Doc;

class Chapt
   constituent_of: Doc;
   integer chapterno;
   string chaptertitle;
end class Chapt;

class Dept
   string dname;
   string dphone;
   ref Emp manager; //inverse of Emp. manages
   set< ref Doc > documents;
   set< ref Emp > staff; // inverse of Emp.belongs-to
                         // inverse of Emp.works-for
   set< tuple (ref Emp emp-of, real hours) > workers
end class
```

(그림 3.6) 객체지향 스키마

### 3.4 제약조건 및 비교

E-R 모델을 객체지향 모델로 변환했을 때 제시한 모델과 비교하기 위한 기준은 이항 관계, n항 관계, 속성의 유무에 의한 관계, 대응수와 참여 제약조건, 일반화 및 세분화에서 분해와 완벽 제약조건을 포함한다.

대응수제약조건은 집단화 참조에 의한 제약조건과 연관화 참조에 의한 제약조건으로 구분한다. 우선, 집단화 참조에서 합성클래스에 있는 속성과 영역에 속하는 성분 클래스는 단방향으로 운행하기 때문에 1:1과 1:N만 수용한다. 또한 연관화 참조에서는 클래스

〈표 3.2〉 대응수 제약조건과 참여 제약조건의 비교

| 모델 / 제약조건 | E-R 모델 (Elmasri & Navathe[5]) | 제안모델 |
|---|---|---|
| 대응수 제약조건 | 연관화와 집단화 구분 없이 사용. 1:1, 1:N, M:N, 양방향 | 연관화: 1:1, 1:N, M:N, 양방향<br>집단화: 1:1, 1:N, 단방향 |
| 참여 제약조건 | 연관화와 집단화 구분 없이 사용. total, partial | 연관화: total, partial<br>집단화: total, partial |

〈표 3.3〉 E-R 모델과 제시한 모델비교

| 모델 | E-R 모델 (Elmasri & Navathe[5]) | | 제안모델 |
|---|---|---|---|
| 특성 | 개체 | 주요키 있는 개체 | 객체 |
| | | 주요키 없는 개체 | 성분객체 |
| | | 제런드(Gerand) | 연관화 |
| | 속성 | 주요키 | 객체 식별자 |
| | | 다중값 | 집합 구성자 |
| | | 합성 | 집단화 |
| | 관계 | 재귀 | 연관화 |
| | | 이항 | 관계의미(IS-A, HAS-A, PART-OF)에 따른 역할 |
| | | 다항 | 관계의미(IS-A, HAS-A, PART-OF)에 따른 역할 |
| | 일반화/세분화 | 참여 완전성 | 지원 |
| | | 분리성 | 지원 |
| | 집단화 | | 합성객체 |
| | 범주 | | 다중 상속 |

들이 독립적으로 연관되어 양방향으로 운행되기 때문에 1:1, 1:N과 M:N 대응수 제약조건을 가진다.

참여 제약조건은 연관화와 집단화 관계에서 전체와 부분 제약조건을 모두 운행하며, "total" 키워드 유무에 의해서 구분하고 있으며 비교는 〈표 3.2〉와 같다. E-R 모델에서 개체, 속성, 대응수, 관계 및 연관화 등에 대한 비교는 〈표 3.3〉와 같다. 또한 〈표 3.2〉에서는 제시한 개체-관계 모델과의 비교 기준은 Elmasri와 Nevathe[6]의 특성에 기준하여 분석하였으며, 〈표 3.3〉에서는 Elmasri와 Nevathe의 특성에 기반하여 객체 지향 모델로 변환했을 때의 특성을 대응해서 보여주고 있다.

## 4. 결 론

현실 세계의 문제 영역을 논리적 모델로 표현하기 위하여 E-R 모델을 사용한다. 객체지향 모델은 실세계의 개체와 관계성을 데이터베이스 시스템에 표현하는데 적합한 논리적 모델로서, 다중 사용자의 병행적 지원, 대용량의 데이터 조작 등을 적절하게 표현하고자하는 응용분야에서 이용되고 있다. 현실세계 개체의 표현을 정확하게 모델링하기 위해서는 E-R 모델을 객체지향 모델로 변환하는 효율적인 방법이 필요하다.

따라서 이 논문은 현실세계의 개체를 정확하게 표현하기 위해서 E-R 모델을 객체지향 데이터베이스 모델로 변환하기 위한 기법을 제안하였다. 이 제안된 방법은 개체들 사이의 관계를 일반화, 집단화와 연관화를 구분하기 위해서 관계 추상성을 규정하였으며, 이 개체 사이의 관계에 대한 역할을 객체지향 모델의 속성-영역을 표현하는데 중점을 두어 연구하였다.

그리고 이 논문은 E-R 모델을 객체지향 모델로 변환하기 위해서 E-R 모델의 구성 요소인 개체와 개체 스킴, 관계와 관계 스킴, 데이터베이스와 데이터베이스 스킴, 그리고 개체 타입들과 관계의 역할 등을 형식화하고, 이들에 대응하는 객체지향 모델의 구성 요소인 객체와 메소드, 타입과 클래스, 클래스 계층, 집단화 타입 계층, 그리고 연관화 타입 계층 등을 규정하였다. 이 집단화 타입 계층에서 두 객체 사이에 단방향성, 종속성과 추이성 관계에 의한 강한 결집력으로 인해서 성분 타입의 속성으로 재사용할 수 있도록 하는 변환 규칙을 규정하였고, 이 연관화 타입 계층에서 두 객체 사이에 양방향성, 독립성과 생산성 관계에 의한 약한 결집력으로 인해서 독립적인 타입 사이에 연결 참조 속성을 도입하여 변환 규칙을 규정하였다.

이 객체지향 모델 구성요소와 변환 규칙 등을 규정함으로서 E-R모델에서 개체들 사이의 관계 의미에 대한 역할을 객체지향 모델의 속성-영역 관계에 도입하여 일반식을 유도할 수 있음을 보였다. 또한 규정된 변환 규칙에서 E-R모델의 대응수 제약조건과 참여 제약조건으로 구분하고, 개체, 속성, 관계, 일반화 및 집단화 등의 특성에 따라 각각 E-R 모델과 제시한 모델을 비교 분석하였다.

이 연구는 실세계에서 개체들 사이의 관계를 일반화, 집단화와 연관화로 구분하여 이 관계들의 역할을 객체지향 모델의 속성-영역 관계에 도입하여 논리적 모델을 설계하는데 적용할 수 있다. 현재 이 연구에서 제안한 변환 방법을 확장하여 시간개념의 적용과 객체버전 및 속성버전에 따른 의미 부여에 관한 연구가 진행 중에 있다.

## 참 고 문 헌

[1] Battista G. D. and Lenzerini M., "Deductive entity relationship modeling," IEEE Transactions on knowledge and data Engineering, Vol. 5, No. 3, June 1993, pp. 439-450.

[2] Bertino, E., and Marrino, L., "Object-oriented database management systems:concepts and issues," IEEE Computer, 1991.

[3] Cartell, R. G. G., *The Object Database Standard: ODMG-93(Release 1.2)*, Morgan Kaufmann Publishers, Inc. San Fracisco, California, 1996, Chapter 2-3, pp. 11-52.

[4] Chen P. P., "The entity-relationship model-toward a unified view of data," ACM TODS, Vol. 1, No. 1 May 1976, pp. 9-35.

[5] Deux, O., et al., *The $O_2$ system.*, ACM, October 1991.

[6] Elmasri, R., and Navathe, S., *Fundamentals of database systems*, 2nd Ed. benjamin/Cummings,

1993.

[7] Gorman K. and Choobineh. J., "The object-oriented entity-relationship model(OOERM)," Journal of Management Information Systems, Vol. 7, No. 3, Winter 1990-91, pp. 41-65.

[8] Hull, R. and Yap, C. "The format model : a theory of database organization," Journal of the ACM, Vol. 31, No. 3, July 1984, pp. 518-537.

[9] Kappel G. and Schrefl M., "A behavior integrated entity-relationship approach for the design of object-oriented databases," C. Batini, editor, Proceedings of the 7th International Conference on Entity-Relationship Approach, Rome, Italy, Nov. 1988, pp. 311-328.

[10] Kim W., *Introduction to object-oriented databases*, The MIT Press, 1991.

[11] Kornatzky Y. and Shoval. P., "Conceptual design of object-oriented schemes using binary-relationship model," Technical Report FC 93-08, BenGurion University the Negev, P.O.B. 653, Beer-Sheva 84105, Israel, June 1993.

[12] Kuper, G. and Vard, M., "A new approach to database logic," Proc. PODS, 1984, pp. 86-96.

[13] Lee, H. R., "A logical optimization of queries in object oriented database system," PhD Thesis, the Chonbuk National Univ. at Chonbuk in Korea 1994.

[14] Lee, H. R., et al., "Logical optimization of queries using a complex object calculus transformation," Journal of KISS(B) : Software and Applications, VoL. 22, No. 12, December 1995, pp. 1601-1613.

[15] Ling L., "A recursive object algebra based on aggregation for manipulating complex objects," Data & Knowledge engineering, Vol. 11, North-Holland, 1993, pp. 21-60.

[16] Lieberherr K. J. and Xiao C., "Formal foundations for object-oriented data modeling," IEEE Transactions on Knowledge and Data Engineering, Vol. 5, No. 3, June 1993, pp. 462-478.

[17] Osborn, S. and Treleaven. T., "The design of a relational database system with abstract types as domains," ACM TODS, Vol. 11, No3, Sep. 1986, pp. 357-373.

[18] Poncelet, P. M., Teisseire, Cicchetti, R. and Lakhal. L., "Towards a formal approach for object database," Proc. of the 19th International Conference on Very Large Databases, Dublin, Ireland, 1993, pp. 278-289.

[19] Rowe, L.A. and Stonebraker, M.R., "The POSTGRES data model," Proc. PLOB, 1987, pp. 93-96.

[20] Rumbaugh J., Blaha M., Premerlani W., Eddy F., Lorensen W., *Object-oriented modeling and design*, Prentice-Hall, 1991.

[21] Wilms, P., Schwara, P., Schek, H. and Hass, L., "In corporating data types in as extensible database architecture," Proc. 3rd Int'l. Conf. on Data and Knowledge Bases : Improving Usability and Responsiveness, 1988, pp. 180-192.

**김 삼 남**

1977년 육군사관학교 토목공학과졸업(이학사)
1987년 미해군 대학원 전자계산학과 졸업(이학석사)
1997년 8월 충북대학교 전자계산학과졸업예정(이학박사)
1995년~현재 육군 교육사령부 C4I 개발단
관심분야 : 객체지향 데이터베이스, C4I 시스템 및 데이터 마이닝

**이 홍 로**

1984년 전북대학교 전기공학과 졸업(공학사)
1986년 전북대학교 대학원 전자계산기 전공(공학석사)
1994년 전북대학교 대학원 전산응용공학 전공(공학박사)
1994년~현재 충북대학교 컴퓨터정보통신연구소 연구원
관심분야 : 객체지향 언어, 객체지향 데이터베이스 질의 처리 및 시간지원 데이터베이스 시스템


**류 근 호**

1976년 숭실대학교 전산학과 졸업(이학사)

1980년 연세대학교 산업대학원 전산전공(공학석사)

1988년 연세대학교 대학원 전산전공(공학박사)

1976년~1986년 육군군수 지원사 전산실 (ROTC장교). 한국전자통신연구소(연구원). 한국방송통신대 전산학과(조교수) 근무

1989년~1991년 Univ. of Arozona. Research Staff (TempIS 연구원, Temporal DB)

1986년~현재 충북대학교 컴퓨터과학과 교수 겸 컴퓨터정보통신연구소장

관심분야 : 시간지원 데이터베이스, 시공간 데이터베이스, DBMS 및 OS, 객체 및 지식베이스 시스템