객체 버전화를 중심으로 시간지원 개체-관계 모델의 시간지원 객체 지향 모델로 변환

Transforming an Entity-Relationship Model into a Temporal Object Oriented Model Based on Object Versioning

이 홍 로*
Hong-Ro Lee

요 약

개체-관계 모델은 데이터베이스 설계와 시스템 분석을 위해서 시간 응용 분야에서 사용되고 있는 개념적 모델로서, 현실세계의 시간 문제 영역을 논리적 모델로 표현하기 위한 기반이 된다. 객체 지향 모델은 실세계의 이력 자료에 대한 객체와 관계성을 데이터베이스 시스템에 표현하는데 적합한 논리적 모델로서, 이력 자료의 병행적 지원과 시공간 자료의 조작 등을 적절하게 표현하고자 하는 응용분야에서 이용되고 있다. 현실세계 개체의 표현을 정확하게 모델링하기 위해서는 아주 적은 제약조건을 갖고 E-R 모델을 객체 지향 모델로 변환하는 방법이 필요하다.

또한 시간지원 객체 지향 모델은 시간 의미를 객체 지향 모델에 추가하여 시간에 따라 변화된 정보를 처리할 수 있는 논리적 모델이다. 시간지원 객체 지향 모델의 두 부류는 시간을 결합하는 단위에 따라 속성 버전화와 객체 버전화로 나누어진다. 이 두 가지 중에서 객체 지향 모델은 이 상태와 행위의 재사용성을 증가시키고, 개체들 사이의 유기적 관계를 효율적으로 표현할 수 있기 때문에 객체 지향 모델의 연구가 필요하다. 그런데 개체-관계 모델을 객체 지향 모델로 변환하기 위한 기법과 제약조건이 수반된다. 그래서 시간 개념이 포함된 개체-관계 모델을 객체 지향 모델로 변환하는 연구가 필요하다.

따라서 현실세계의 객체 버전화를 위한 이력 개체를 정확하게 표현하기 위해서 본 논문은 E-R 모델을 객체 지향 데이터베이스 모델로 변환하기 위한 기법을 제시하고 있다. 즉, 현실세계의 개체와 관계를 데이터베이스에 표현하는데 적합한 시간지원 객체 지향 모델로 일반화, 집단화와 연관화에 대한 역할에 따라 변환하며, 그리고 다형성 관점에서 일반화 상속과 집단화 상속을 조합하여 기능을 확장한다. 이 변환과 확장은 데이터베이스 설계의 논리적 모델 설계 및 재사용성을 증가시키는데 기여할 것이다.

Abstract

Commonly to design a database system, a conceptual database has to be designed and then it is transformed into a logical database schema prior to building a target database system. This paper proposes a method which transforms a Temporal Entity-Relationship Model(TERM) into a Temporal Object-Oriented Model(TOOM) to build an efficient database schema.

I formalize the time concept in view of object versioning and specify the constraints required during transformation procedure. The proposed transformation method contributes to getting the logical temporal data from the conceptual temporal events without any loss of semantics. Compared to other approaches of supporting various properties, this approach is more general and efficient because it is the semantically seamless transformation method by using the orthogonality of types of objects, semantics of relationships and constraints over roles.

1. 서 론

실세계에서 시간에 따라 변화하는 개체와 개체 사이의 관계를 개념적으로 모델화 하기 위해서 사용자의 시간적인 요구 사항으로부터 이력 사항에 대한 개체, 개체 사이의 관계와 속성 등에 대해서 분석하는 것이 필요하다. 이 개념적 모델화가 시간지원 개체-관계 모델이다[1][5]. 이 시간원 개체-관계 모델을 설계하는 목적은 최종 사용자가 필요로 하는 시간 내용을 컴퓨터 시스템의 특

* 정회원 : 전북대학교 공과대학 컴퓨터공학과 강사
leehongro@orgio.net

성에 독립적이고, 여러 시스템(각종 H/W, DBMS, OS)에서 수행 가능하며, 사용자가 이해할 수 있는 형태로 표현이 가능해야 한다. 이 시간지원 개체-관계 모델은 자료 구조 자체만 규정하고, 응용 프로그램, H/W나 DBMS 등의 물리적 요소와 독립적이다. 또한 이 개념적 모델인 시간지원 개체-관계 모델을 물리적인 시스템에 표현하고 처리하기 위한 논리적 모델이 요구된다. 이 논리적 모델은 시간지원 관계 모델[28][31][33]과 시간지원 객체 지향 모델[9][10][24][26][35]이다. 그런데 이 개념적 모델인 개체-관계 모델을 논리적 모델로 변환하는 방법은 개체나 관계를 독립적으로 릴레이션화 하여 표현하는 관계 모델이 있으며, 개체와 관계의 상태와 행위를 함께 조합하여 클래스로 표현하는 객체-지향 모델이 있다. 이 두 가지 중에서 객체 지향 모델은 이 상태와 행위의 재사용성을 증가시키고, 개체들 사이의 유기적 관계를 효율적으로 표현할 수 있기 때문에 객체 지향 모델의 연구가 필요하다. 그런데 개체-관계 모델을 객체 지향 모델로 변환하기 위한 기법과 제약조건이 수반된다. 그래서 시간 개념이 포함된 개체-관계 모델을 객체 지향 모델로 변환하는 연구가 필요하다.

개체-관계 모델은 실세계를 구조적으로 분석하고 개념적으로 모델화 하는데 있어서 폭넓게 이용되어 왔고, 개체-관계 접근 방법은 실세계의 문제점들을 모델화하고 데이터베이스 스키마로 변환하는데 용이하다. 개체-관계 모델은 실세계가 개체(entity)들의 모임, 개체 사이의 관계 (relationship), 이 개체와 관계를 기술하기 위한 속성으로 구성되어 있다고 보는 것이다. 이에 덧붙여서 이 개체-관계 모델을 설계하기 위해서 농도 제약조건(cardinality constraint), 참여 제약조건(participation constraint), 일반화(generalization), 집단화(aggregation)와 연관화(association) 등의 구성요소가 필요하다. Chen[7]은 1976년에 처음 개체-관계 모델을 도입하였다. 이 연구는 참여 제약조건을 명시적으로 표기하였으며, 개체와 관계, 약 개체와 약 관계, 속성, 그리고 농도 제약조건을 규정하였다. 그러나 일반화와 참여 제약조건은 이용하지 않았다. Batini, Ceri와 Navathe[2]는 개체를 정규 개체와 약 개체로 구분하여 사용하고, 속성을 단일 속성과 주요키만 규정하며, 농도 제약조건을 (Min, Max)로 표기하며, 관계를 재귀적, 이항, 3항 속성을 명시적으로 표기하고 있다. 그리고 일반화를 분리성과 완벽성을 구분하지 않고 사용하였다. Elmasri와 Navathe[11][12]는 개체를 정규 개체, 약 개체와 Gerund 개체로 구분하여 사용하고, 속성을 주요키, 외래키, 선택적 속성, 합성 속성과 유도 속성으로 나누어 표기하며, 농도 제약조건을 Look Across, 참여 제약조건을 Look Here로 표기하였다. 또한 관계는 재귀적, 이항, 3항 속성을 명시적으로 표기하고 있고 일반화는 분리성과 완벽성을 구분하여 표기하였다.

시간지원 개체-관계 모델은 개체-관계 모델의 확장형이다. 이 시간지원 개체 관계 모델은 유효시간(valid time)과 거래시간(transaction time)이라는 직교적(orthogonal) 시간 특징을 폭넓게 사용하고 있다. 데이터베이스 사실의 유효시간은 실제로 사건이 발생한 시점을 의미한다. 유효시간과 직교하는 데이터베이스 사실에 대한 거래시간은 이 사실을 테이터베이스에 입력하고 검색하는 시간을 말한다. 유효시간과 다르게 거래시간은 데이터베이스에 저장된 구조 뿐만 아니라 그 사실과도 연관될 수 있다. 거래시간은 삽입시점으로부터 삭제시점까지의 기간을 말한다. 유효시간과 거래시간을 모두 지원하는 것을 이원시간(bitemporal time)이라 한다. 개체-관계 모델에서 이원시간을 수용할 수 있도록 내장된 수단을 제공한다면, 한 데이터 모델은 이원시간을 지원한다고 말한다. 타입 버전화(type versioning)는 해당 개체 타입 및 관계 타입의 비시간 속성에 시간 속성을 추가하여 이력사항을 표현하는 기법이다[29]. 이 버전화는 한 인스턴스에서 비시간 값에 시간 값을 결합한 것이다. Theodoulidis와 Loucopoulos[32]은 의미적 데이터 모델화와 요구 사양 관점에서 개념적 모델

화에 시간을 이용하고 규정하기 위해서 아홉 가지의 접근 방법을 비교, 기술하고 있다. 또 모델을 비교하기 위해서 시간 의미, 모델 의미, 시간 기능으로 모델을 분류하고도 있다. 이 논문의 주제는 시간의 특성과 용어를 예로 들어 연구한 것이다. McKenzie와 Snodgrass[23]는 관계 대수를 12가지의 다른 시간 개념으로 요약, 평가하고, 설계 기준에 따라 대수도 평가하였다. 그러나 시간지원 개체-관계 모델의 구조적인 면은 다루지 않았다. Klopprogge[19]는 개체-관계 모델에 시간 차원을 처음으로 도입하였고, Pascal 유형의 구분을 포함하고 있다. 데이터베이스 설계자가 속성 값과 관계 타입 역할로 이력구조를 이용하는 시간요소를 모델화 하도록 하였다. 덧붙여서 이력사항은 개체와 관계의 존재를 모델화하고 있다.

개체-관계 모델을 객체 지향 접근 방법과 조합하기 위한 많은 노력이 있어 왔다. 또한 스냅샷 개체-관계 모델을 객체 지향 접근 방법과 통합하기 위한 연구도 계속되고 있다. 이러한 연구들로는 객체 지향 개체-관계 모델(OOERM)[15], 행위 통합 개체-관계 접근(BIER Approach)[18], 개체-관계 언어(ERL)[3] 등이 있다. 관계 객체의 여부에 따라 객체 버전화와 속성 버전화를 지원하는 연구도 있다[12].

더욱이 개체 관계 모델을 또 다른 모델로 변환하기 위한 알고리즘이 제시되어 있는데, Kornatzky 등[20]은 E-R 모델의 변형인 이항-관계 모델을 객체 지향 모델로 변환하는 기법을 연구하였다. Poncelet 등[25]은 확장된 E-R 모델인 IFO 모델을 O2 객체 지향 모델로 변환하기 위한 알고리즘을 기술하고 있다. MOTAR[5]는 시간 속성과 관계를 가지는 객체를 위한 모델 개발의 동기는 객체-지향 데이터베이스, 지식 기반 시스템, 시간지원 데이터베이스와 같은 분야에 데이터베이스 연구를 통합하였다. 또한 이 시간지원 E-R 모델을 관계형 데이터 모델로 변환시켰다. Ben-Zvi[4]는 모델 성분을 개체 클래스, 값 클래스, 복합 개체 클래스, 복합 값 클래스와 관계 클래스로 구분하여 규정하였다. 또한 개체-관계 모델을 시간에 따라 변환하는 개체와 관계 클래스를 규정하였다. 그러나 개체-관계 모델을 시간지원 관계 모델이나 시간지원 객체 지향 모델로 변환하는 기법은 연구하지 않았다. Cardelli 등[6]은 일반적인 농도 제약조건을 스냅샷과 생존시간 농도 제약조건으로 대치시켰다. 또한 관계 타입의 부류를 정선하였다. 그러나 이 연구는 TER모델을 E-R 모델로 변환하는 것이다.

시간지원 자료를 개념적이고 논리적으로 규정하고, 이 개념적 모델인 개체-관계 모델을 객체 지향 모델로 변환하는데 다음과 같은 문제점을 지니고 있다. 유효시간과 거래시간 표현 방법, 사실과 이원시간을 결합하는 방법, 그리고 속성들을 표현하는 방법이 어렵다. 이 문제점을 해결하기 위해서는 모델 변환을 다음과 같은 조건으로 만족시킬 수 있다. 실세계에 맞는 개체-관계 모델에서 관계의미를 역할의 종류에 따라 명확히 기술해야 하며, 이에 기반한 객체 지향 모델을 설계해야 한다. 개체-관계 모델에서의 제약 조건을 일관성있게 객체 지향 모델로 변환해야 한다. 또한 실세계의 개체-관계 모델에서 시간 개념을 객체 지향 모델로 표현 가능토록 해야 한다.

그래서 본 연구는 시간 관점에서 실세계의 대상 및 관계 속성 등을 개념적인 수준에서 기술하고, 이에 대응하는 논리적 수준인 객체 지향 모델로 변환하는 방법도 제시하고자 한다. 본 논문의 연구와 내용은 실세계의 시간 정보구조를 개념적으로 모델화하기 위해 타입 버전화 개체-관계 모델을 제시한다. 제시한 모델은 기존의 모델과 비교해서 개체 사이의 관계 의미를 역할과 제약 조건으로 규정하고, 이 개념적 모델에 대응하는 객체 지향 모델을 규정하고, 이에 따른 변화 단계와 관계의 역할을 속성 및 메소드로 변환하는 방법을 제시한다. 시간 정보를 논리적으로 모델화하기 위해서 위의 변환기법을 이용하여 객체 버전화 객체 지향 모델을 제시한다. 이에 기반한 클래스 다이어그램과 클래스 정의어 구문을 규정한다.

본 논문의 구성에 대해서 제2장에서는 시간의 개요와 E-R 모델의 구성 요소에 대해 규정하고, 제3장에서는 객체 지향 모델의 구성 요소와 변환 기법에 대해 거론하며, 제4장에서는 시간지원 E-R 모델을 객체 지향 모델로 변환한 것에 대한 비교 및 분석하여 기술한다. 그리고 마지막으로 결론을 내리고 향후 연구에 대해 토의한다.

2. 시간지원 E-R 데이터 모델

이 절에서는 시간지원 모델에서 시간을 정의하고, 타입 버전화를 지원하는 E-R 모델과 객체 버전화를 지원하는 객체 지향 모델을 제시한다.

2.1 시간 개요

시간지원 데이터 모델에서 시간 표현 방법은 연속적인 시간을 표현하기 위해서 이산 시점, 시간 간격, 그리고 시간을 집합화하기 위한 시간 요소로 나눌 수 있다[17][29]. 구체적으로 설명하면, T는 전 순서 이산 시점(total order discrete time point)들의 셀 수 있는 무한 집합이며, t는 연속적으로 증가하는 시점을 표현한 것이다. 또한 임의의 두 시점들을 한 사건 사이의 간격으로 표현하는 방법과 이력 시간을 집합의 요소로 표현하는 방법이 있다. 시간은 전 순서(total order) 이산 시점 값들의 셀 수 있는 무한 집합으로 $T = \{ t_{-\infty}, ..., t_0, t_1, ..., t_{now}, t_{now+1}, ..., t_{\infty}\}$이다. 여기서, t_{now}는 현재 시점을 나타내는 시간의 단위(chronons: 초, 분, 시, 일, 주, 월, 분기, 반기, 년 등)이다. 또한 Tp = <T, < >, Tp는 시점 정의역이고, <는 전 순서를 나타낸다. 시간 간격(time interval)은 두 사건(event) 사이의 기간을 의미한다. 이 간격은 규정된 연속적인 두 시점 tis와 tie 순서쌍의 집합이다. 즉 시간 간격의 전체 집합은 $TI = \{ti_0, ti_1, ..., ti_n\}$이다. 여기서 ti_i는 (t_{is}, t_{ie}), $[t_{is}, t_{ie})$, $[t_{is}, t_{ie}]$ 또는 (t_{is}, t_{ie})이며, $1 \le i \le n$ 이다. 세부적으로 "("와 ")"는 인접 시점을 포함하고, "["와 "]"는 인접 시점을 포함하지 않는다. 또한 t_{is}는 시작 시점을 나타내고, t_{ie}는 종착 시점을 나타내며, 그리고 이들은 시간의 전체 집합 T에 존재하는 시점들이다. 시간 요소(time element)는 이력시간을 시점이나 시간 간격 단위로 집합화하여 표현하기 위한 기법이다. 그래서 시점들에 의한 요소들의 집합을 사건시간 스탬프 및 사건 생명주기라 한다. 또한 시간 간격들에 의한 요소들의 집합을 간격시간 스탬프 및 간격시간 사건 생명주기라 한다. 시간 요소 전체 집합은 $TE = \{te_1, ..., te_i, ..., te_n\} = T/TI$이다. 여기서 te_i는 시점 t_i 및 시간 간격 ti_i를 나타내며, $1 \le i \le n$ 이다.

시간지원 표현을 위한 시간 결합 방법은 현실 세계의 개체 및 관계에 대한 이력자료를 각 속성에 시간을 결합하는 속성 버전화와 시간속성을 추가하는 타입 버전화가 있다. 속성 버전화(attribute versioning)는 해당 타입 및 관계의 속성에 시간을 결합하여 이력사항을 나타내는 것을 의미한다[14][30][31]. 이 속성 버전화는 한 속성에 비시간 값과 시간 값을 동시에 같이 표현한 것이다. 속성 버전화는 av = <v, bt>이다. 여기서 v는 한 속성의 비시간 값이며, bt는 이원시간 값이며, 시점이나 시간 간격이 될 수 있다. 또한 타입 버전화(type versioning)는 해당 개체 타입 및 관계 타입의 비시간 속성에 시간 속성을 추가하여 이력사항을 표현하는 기법이다[28]. 이 버전화는 한 인스턴스에서 비시간 값에 시간 값을 결합한 것이며, 이 타입 버전화는 $tv = \{v_1, ..., v_i, ..., v_n, bt, ut\}$나 $tv = \{v_1, ..., v_i, ..., v_n, bt\}$이다. 여기서 vi는 한 인스턴스의 비시간 값들이며, $1 \le i \le n$ 이고, bt는 이원시간 값이며, ut는 사용자 정의 시간 값이다. 또한 이 시간 값들은 응용 방법에 따라 시점이나 시간 간격이 될 수 있다. 이 타입 버전화는 관계형 모델에서는 튜플 버전화라 하고, 객체 지향 모델에서는 객체 버전화라 한다.

시간지원 데이터베이스는 시간 속성과 그 자료의 이력사항을 제공하기 위하여 기존의 데이터베이스 시스템의 확장을 필요로 한다[28]. 이 시

간지원 시스템을 위한 시간차원(dimension)은 유효시간, 거래시간, 이원시간, 그리고 사용자 정의 시간으로 나누어 규정할 수 있다. 유효시간은 현실세계에 존재하는 실체에 대한 논리적인 시간을 의미하며, 실체가 발생하였거나 실체가 소멸된 시간을 가리킨다[30][33]. 유효시간은 일반적으로 사용자에 의해서 결정되는데 사용자가 위치한 시간 시스템에 의해서 상대적인 값으로 표현된다. 이 유효시간은 vt ⊆ T/TI이다. 여기서 T는 이산 시점의 전체 집합이며, TI는 시간 간격의 전체 집합이다. 거래시간(transaction time)은 현실세계에 존재하는 실체가 데이터베이스 관리시스템에 의해서 처리되어진 시간을 의미하며, 데이타베이스 관리시스템이 실체를 처리할 때 시스템 클럭(system clock)에 의해서 결정되어 자동적으로 삽입된다[17][28]. 이 거래시간은 tt ⊆ T/TI이다. 여기서 T는 이산 시점의 전체 집합이며, TI는 시간 간격의 전체 집합이다. 이원시간은 현실세계에 존재하는 실체에 대한 논리적인 시간을 의미하는 유효시간과 현실세계에 존재하는 실체가 데이터베이스 관리시스템에 의해서 처리되어진 시간을 의미하는 거래시간을 동시에 같이 표현하는 시간을 말한다[26]. 이 이원시간은 bt = <vt, tt > ⊆ T/TI이다. 여기서 vt는 유효시간이며, tt는 거래시간이다. T는 이산 시점의 전체 집합이며, TI는 시간 간격의 전체 집합이다. 사용자 정의 시간은 유효시간과 거래시간에 의해 처리되지 못하는 시간으로써 기존의 데이터베이스와 마찬가지로 사용자가 직접 정의한다. 사용자 정의 시간을 위한 내부적인 표현과 입출력 함수가 부가적으로 요구된다[28]. 이 사용자 정의 시간은 ut ⊆ T이다. 여기서 ut는 사용자 정의 시간이며, T는 이산 시점의 전체 집합이다.

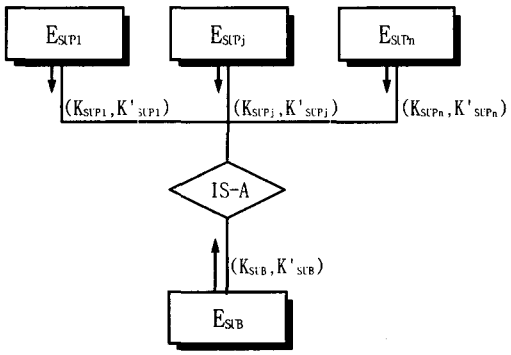
2.2 객체 버전화를 위한 E-R 데이터 모델

이 장에서는 객체 버전화를 위한 개념적 모델인 E-R 모델의 개체, 관계, 속성, 일반화 및 데이터베이스 스킴에 대해서 형식화한다. E-R 모델은 실세계의 개체와 그 관계에 대하여 구조적으로 분석하고 이 개체와 그 관계를 추상화하는 E-R 모델은 폭넓게 이용되어 왔다. 이 E-R 모델은 데이터베이스를 논리적으로 규정하는 스키마로 변화하는데 용이하다. E-R 모델은 개체들의 모임, 개체들 사이의 관계, 그리고 개체를 기술하기 위한 속성들로 구성된다. 또한 한 개체가 참여할 수 있는 관계 인스턴스의 수를 규정하는 농도 제약조건이 필요하다. 이 제약조건을 이항 관계에서는 1:1, 1:N과 M:N으로 나타내고, 3항 관계에서는 1:1:1, 1:1:N, 1:N:M과 M:N:P의 형식으로 표기된다. 이들은 최대 농도(maximum cardinality)라고도 표기한다. 이 개체-관계 모델은 실세계의 이력사건에 대한 개체와 그 관계에 대하여 구조적으로 분석하고 이 개체와 그 관계를 추상화하는 응용분야에서 폭넓게 이용되어 왔다[7][27]. 또한 이 E-R 모델은 데이터베이스를 논리적으로 규정하는 스키마로 변화하는데 용이하다. E-R 모델은 개체들의 모임, 개체들 사이의 관계, 그리고 개체를 기술하기 위한 속성들로 구성된다.

2.2.1 제약조건

개체들 간의 관계 의미에 따른 제약조건은 일반화, 세부화, 집단화와 연관화에 의해서 결정된다. 일반화와 세부화에 대한 농도 제약조건은 각각 상위 개체 타입과 하위 개체 타입의 개수를 표현하는 것이다. 그러나 집단화와 연관화는 기존의 방법과 같이 한 개체가 다른 개체들과 관계하는 수를 표현하는 것이다[3]12]. 일반화와 세부화의 참여 제약조건은 재사용성 여부를 결정하는 것이고, 집단화와 연관화의 참여 제약조건은 한 개체가 다른 개체들과 관계하는 지의 여부를 결정하는 것이다. 그래서 이들에 대한 세부적인 기술은 다음과 같다.

가. 일반화 제약조건



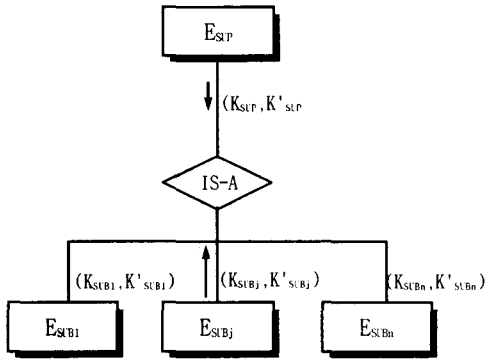
K_{SUB} = 0,1
- K_{SUB} = 0 : 부분
- K_{SUB} = 1 : 전체

K'_{SUB} = 0,N
- K'_{SUB} = 0 : binary relationship
- K'_{SUB} = N : (N+1)ary relationship

(그림 1) 일반화에 의한 제약조건

위 그림 1에서 일반화에 의한 제약조건 (K_{SUB}, K'_{SUB})에서 농도 제약조건은 K'_{SUB}에 의해서 결정되는데 K'_{SUB}가 1이면 개체 타입 E_{SUB}와 관계하는 상위 개체 타입 E_{SUP}의 개수가 하나임을 의미한다. 즉, 이항 관계에 의한 일반화이다. 또한 K'_{SUB}가 N이면 개체 타입 E_{SUB}와 관계하는 상위 개체 타입 E_{SUP}의 개수가 N임을 의미한다. 즉, (N+1)항 관계에 의한 일반화이다. N ≥ 2에 의한 농도 제약조건에서 상위 개체 타입들의 각 개체들의 합집합을 하위 클래스가 재사용하는 제약조건을 합집합 제약조건(union constraint)이라 하며, 이 일반화의 의미는 "is-in(u)"으로 표기한다. 또한 이 (N+1)항 관계에 의한 농도 제약조건에서 상위 개체 타입들의 각 개체들의 교집합을 하위 클래스가 재사용하는 제약조건을 교집합 제약조건(intersection constraint)이라 하며, 이 일반화의 의미는 "is-in(i)"으로 표기한다. 일반화에 의한 제약조건 (K_{SUB}, K'_{SUB})에서 참여 제약조건은 K_{SUB}에 의해서 결정되는데 K_{SUB}가 1이면 하위 개체 타입 E_{SUB}을 위해서 상위 개체 타입 E_{SUP}으로부터 필수적으로 개체들을 재사용함을 의미한다. 이를 일반화에 의한 전체 제약조건이라 한다. 또한 K_{SUB} = 0이면 하위 개체 타입 E_{SUB}을 위해 상위 개체 타입 E_{SUP}으로부터 NULL값을 상속할 수 있음을 의미한다. 이를 일반화 부분 제약조건이라 한다. 그래서 일반화에 의한 역할 기능의 개수가 N이면 상위 개체 타입의 개수가 N개임을 의미한다.

나. 세부화 제약조건



K_{SUP} = 0,1
- K_{SUP} = 0 : 부분
- K_{SUP} = 1 : 전체

K'_{SUP} = 0,N
- K'_{SUP} = 0 : binary relationship
- K'_{SUP} = N : (N+1)ary relationship

(그림 2) 세부화에 의한 제약조건

위 그림 2에서 세부화에 의한 제약조건 (K_{SUP}, K'_{SUP})에서 농도 제약조건은 K'_{SUP}에 의해서 결정되는데 K'_{SUP}가 1이면 상위 개체 타입 E_{SUP}와 관계하는 하위 개체 타입 E_{SUB}의 개수가 하나임을 의미한다. 즉, 이항 관계에 의한 세부화이다. 또한 K'_{SUP}가 N이면 개체 타입 E_{SUP}와 관계하는 하위 개체 타입 E_{SUB}의 갯수가 N임을 의미한다. 즉 (N+1)항 관계에 의한 세부화이다. 또한 N ≥ 2에 의한 농도 제약조건에서 부분 클래스들에 의한 각 개체 값들이 서로 중첩되는 제약조건을 분리 제약조건(disjoint constraint)이라 하며, 이 세부화의 의미는 is-a(d)로 표기한다. 부분 클래스들에 의한 각 객체들이 서로 중첩될 수 있는 제약 조건을 중첩 제약조건(overlapping constraint)이라 하며, 이 세부화에 의한 의미는 is-a(0)으로 표기한

다. 세부화 의한 제약조건(K_{SUP}, K'_{SUP})에서 참여 제약조건은 K_{SUP}에 의해서 결정되는데 K_{SUP}가 1 이면 상위 개체 타입 E_{SUP}의 값들이 하위 개체 타입 E_{SUB}으로 값들이 재사용됨을 의미한다. 이를 세부화에 의한 전체 제약조건이라 한다. 또한 K_{SUP} = 0이면 상위 개체 타입 E_{SUP}의 값들이 하위 개체 타입 E_{SUB}으로 N_{ULL}들이 재사용될 수 있음을 의미한다. 이를 세부화에 의한 부분 제약 조건이라 한다. 그래서 세부화에 의한 역할 기능의 개수가 N이면 하위 개체 타입의 개수가 N개임을 의미한다.

다. 집단화와 연관화 제약조건

집단화와 연관화에 대한 제약조건은 한 개체가 참여할 수 있는 관계 인스턴스의 수를 규정하는 농도 제약조건이 필요하다. 이 제약조건을 이항 관계에서는 1:1, 1:N과 M:N으로 나타내고, 3항 관계에서는 1:1:1, 1:1:N, 1:N:M과 M:N:P의 형식으로 표기된다. 이들은 최대 농도(maximum cardinality)라고도 표기한다. 참여 제약조건은 관계의 제약조건으로서 관계를 경유하여 개체와 개체 간에 연관된 최소 참여 개체 수이다. 이 제약조건을 최소 제약조건(minimum constraint)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 참여 없이 존재할 때 나타난다. 이 참여 제약조건과 농도 제약조건을 함께 사용한 형식으로 최소와 최대 제약조건(min, max)을 표현하기도 한다.

2.2.2 시간지원 개체-관계 모델

개체 타입과 관계 타입은 각각 E와 R로 표기한다. 개체 타입 E의 카아티션-곱(Cartesian Product)은 $E^{\times} = E_1 \times ... E_i \times ... \times E_n$이다. 값들의 전체 집합은 $U_V = \{V_1, V_2, ..., V_{nd}\}$이다. 각 V_i는 원자 값의 집합이다. 여기서 각 i에 대해서 $V_i \neq \varnothing$이며, $V = \bigcup_{i=1}^{nd} V_i$는 모든 값의 집합이다. 또한 2.1절의 시간 개요에 의해서 이산 시점의 집합은 T이고, 시간 간격의 집합은 TI이며, 그리고 시간 요소의 집합은 TE이다. 그래서 속성 버전화를 위한 개체 타입, 관계 타입, 속성, 역할, 제약조건, 및 값들에 대한 개념을 형식화하기 위해서 다음과 같이 정의할 수 있다.

[정의 2.1] 타입 버전화를 위한 개체-관계 스킴 $S = <E, R, б, \rho, V, A, F, AT>$이다. 각 성분은 다음과 같이 구성된다.

◎ 개체 타입의 유한 집합 E

$$E = \{E_1, ..., E_i, ..., E_n\}$$
$$= E_R \cup E_W = \{E_{R1}, ..., E_{Rp}, ..., E_{Rs}\} \cup \{E_{W1}, ..., E_{Wq}, ..., E_{Wt}\}$$

여기서 E_R은 자체 키 값을 가지고 있는 정규 개체 타입(regular entity type)이고, E_W는 자체 키 값을 가지지 않는 약 개체 타입(weak entity type)이다.

◎ 관계 타입의 유한 집합 R

$R = \{R_1, ..., R_i, ..., R_n\}$

관계 타입 이름의 요약(signature) б

$б: R \rightarrow E^{\times}$

여기서 $E^{\times} = E_1 \times ... E_i \times ... \times E_N$에서 N이 2이면 두 개의 개체 사이의 관계를 나타내는 이항 관계라 하고, N이 3이면 세 개의 개체 사이의 관계를 나타내는 3항 관계라 하며, 그리고 임의의 N개 사이의 관계를 N-항 관계(N-ary relationship)라 한다.

◎ 역할 할당 함수 ρ

$\rho: R \rightarrow (SE_i \rightarrow role_i)$

여기서 n-항 관계에서 SE_i는 관계 R의 의미를 나타내며, 그 관계 의미는 일반화(S_{ge}) 및 세부화(S_{sp}), 집단화(S_{ag}), 그리고 연관화(S_{as})로 구분된다. 각 의미 SE_i에 대해서 관계 R의 속성 역할을 하는 $role_i$은 $role_i$ = {$role_{ge}$/$role_{sp}$, $role_{ag}$, $role_{as}$}는 제 3장에서 다루고자 하는 객체의 속성과 그 영역을 연결하는 속성 이름을 규정하는 기반이 된다.

◎ 값들의 유한 집합 V

$V = \{V_1, \ldots, V_i, \ldots, V_n\}$

여기서 V는 기본 데이터 타입들(string, integer, real 등)의 값들이다.

◎ 속성의 전체 집합 A

$A = \{U_A, U_{AT}\}$

여기서 비시간 속성의 전체 집합은 $U_A = \{A_{sim}, A_{set}, A_{com}\}$이며, A_{sim}는 단순 속성(sinple attribute)을 나타내고, A_{set}는 합성 속성(composite attribute)을 나타내며, A_{com}는 집합 속성(set attribute)을 나타낸다. 또한 $U_{AT} = \{vtt, tta, uta\}$는 전체시간 속성을 나타내며, vta는 유효시간 속성을 나타내고, tta는 거래시간 속성을 나타내며, uta는 사용자 정의 시간 속성을 나타낸다.

◎ 속성 타입 함수 AT

$AT: F \rightarrow \{(E \cup R) \rightarrow \{V^{\times}\} \cup \{V^{\times} \times T \times T/TI \times T/TI\}\}$

여기서 $F = \{F_A, F_{TA}\}$는 전체 속성 A에 대한 속성이름이다. F_A는 전체 비시간 속성이름이고, 이에 대한 값들은 $V = V_1 \times \ldots \times V_i \times \ldots \times V_n$이다. F_{TA}는 시간 속성이름이고, 이에 대한 값들은 $T \times T/TI \times T/TI$이며, T는 사용자 정의 시간에 대한 값의 집합이며, 첫 번째 T/TI는 유효시간 값의 집합이며, 두 번째 T/TI는 거래시간에 대한 값의 집합이다. 그래서 타입 버전화를 위한 속성 이름은 $F = \langle F_A, F_{TA}\rangle$이고, $F_A \subseteq U_A$ 이며, $F_{TA} \subseteq U_{TA}$이다.

따라서 일반화 및 세부화에 대한 역할 의미에서 값들의 재사용성 여부를 고려하기 때문에 역할 이름을 명시적으로 나타내지 않으며, 제약조건만 명시적으로 나타낸다. 또한 타입 버전화를 위한 일반화 및 세부화에 대한 역할은 시간에 대한 재사용성만 고려하기 때문에 시간 값을 취급치 않는다.

집단화 및 연관화에 대한 역할 의미에서 역할 이름은 명시적으로 나타내며, 관계하는 개체 타입에 대한 값들은 원자 값만 고려하여 취급한다. 결국 이 특성에 의해서 다음의 정리가 성립한다.

[정리 2.1] 속성을 위해서 역할 이름과 값의 집합을 명시적으로 나타내는 것은 연관화와 집단화에 의한 관계 뿐이다.

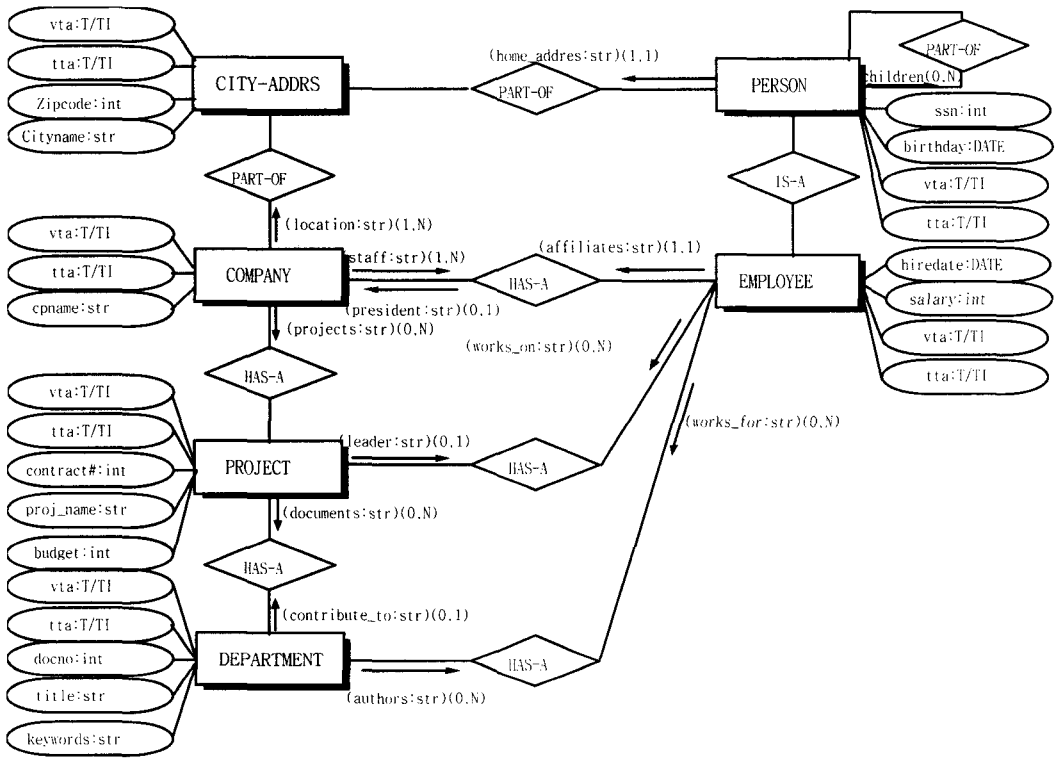
증명 : 정의 2.1의 역할 할당 함수에 의해서 속성 버전화에 대한 시간 값은 rolename : V로 표현한다. 이 식에서 rolename은 속성 값에 대한 역할을 표현한 것이며, V는 관계하는 개체 타입을 명시적으로 표현하기 위한 값의 집합이다. 그런데 일반화와 세부화는 값들을 재사용할 수 있도록 하는 기능만 하기 때문에 속성-영역 관계에 있는 속성 값을 표현하는 것은 연관화와 집단화 뿐이다.

[정리 2.2] 역할의 제약조건은 모든 관계 의미에서 명시적으로 나타난다.

(K, K') ..... ①

<rolename: V>(K, K') ..... ②

증명 : ①식은 정리 2.1에 의해서 역할명을 기입치 않은 일반화와 세부화에 대한 참여 제약조건과 농도 제약조건에 대한 제약조건이며, ②식은 정리 2.1에 의해서 역할명을 기입한 연관화와 집단화에 대한 참여 제약조건과 농도 제약조건을 나타낸 것이다. 관계 의미는 일반



(그림 3) 타입 버전화를 위한 시간지원 개체-관계 모델

화, 세부화, 연관화와 집단화 밖에 없기 때문에 역할에 의한 제약조건은 모든 관계의미에서 명시적으로 나타낸다.

그림 3에서 개체 타입들은 CITY_ADDRS, COMPANY, PERSON, EMPLOYEE, PROJECT, DEPARTMENT 등이며, 속성에 있어서 zipcode 는 비시간 속성을 나타내고, vta: T/TI와 tta: T/TI는 일률적으로 이산시간을 위해서 각 개체 타입에 나타내며, birthday : DATE는 사용자 정의 시간의 속성을 속성을 나타낸다. <home_address: str>>(1, 1)에서 home_address는 집단화 관계에 의한 역할이름이며, 이 역할에서 비시간 값의 집합 str은 관계하는 개체 타입을 원자값화 시킨 것이다. 또한 (1, 1)에서 첫 번째 1은 두 개체 타입 CITY_ADDRS와 PERSON이 전체 제약조건을 가짐을 의미하며, 두 번째 1은 CITY_ADDRS와 PERSON의 값이 1:1 대응 관계에 있음을 의미한다. <staff: str)(1, N)에서 staff는 연관화 관계에 의한 역할이름이며, 이 역할의 비시간 값의 집합 str은 관계하는 개체 타입을 원자값화 시킨 것이다. 또한 (1, N)에서 첫 번째 1은 두 개체 타입 COMPANY와 EMPLOYEE가 전체 제약조건을 가짐을 의미하며, 두 번째 N은 COMPANY와 EMPLOYEE의 값이 1:N 대응 관계에 있음을 의미한다. 각 개체 타입에 대한 시간 속성은 일률적으로 vta: T/TI, tta: T/TI이나 uat: T로 규정하고, 각 속성은 유효시간, 거래시간과 사용자 정의 시간을 위한 것이다.

3. 객체 버전화에 대한 객체 지향 변환

본 절에서는 객체, 메소드, 클래스와 상속성 등

을 규정하고, 개체-관계 모델을 객체 지향으로 변환하며, 이 변환한 기법을 이용하여 객체 버전화를 위한 객체 지향 모델을 제시한다.

3.1 객체 지향 모델 개요

제 2.2절의 개체-관계 모델에서 기술한 개념적 모델을 논리적 모델로 변환하는 방법으로 본 논문에서는 객체 지향 데이터 모델을 선택하였다. 객체는 식별자, 클래스와 값으로 기술한다. 각 객체는 실세계의 개체를 나타낸다. 객체 식별자는 객체의 존재를 나타낸다. 값은 주어진 클래스의 객체를 표현하는 것이다. 객체들은 응용할 수 있는 메소드의 집합인 인터페이스를 통해서만 접속할 수 있다. 그래서 속성과 메소드로 구성되는 클래스 스킴은 다음과 같이 정의한다.

[정의 3.1] 클래스 스킴 C = (A, M)
클래스의 속성을 나타내는 $A = \{A_c \cup A_s \cup A_d\}$는 표현 방법에 따른 속성들의 전체 집합이다. 세부적으로 속성 A_c는 객체의 값에 대한 데이터 타입의 복잡성에 의해서 원자 속성(simple attribute)과 합성 속성(composite attribute)으로 구분되고, A_s는 개체-관계 모델 제약조건인 1:1, N:1, N:1과 M:N의 다중성에 대응해서 단일값 속성(single valued attribute)과 다중값 속성(multi valued attribute)으로 구분되며, 그리고 A_d는 데이터베이스의 값 및 객체들의 저장성 여부에 의해서 저장 속성(stored attribute)과 유도 속성(derived attribute)으로 구분된다. 또한 데이터베이스에 저장된 객체 및 값을 변경하거나 생성하는 기능을 하는 메소드는 M = <mn, ms, mb>로 구성되며, mn은 메소드 이름, ms는 $f : S \times P_1 \times \cdots \times P_K \rightarrow RT$로 표기되는 메소드 요약(signature)이며, 여기서 S는 메소드 m이 정의되는 초기 입력인자의 클래스이고, P_i(i=1, ···, k)는 메소드 인자들이다. 또한 RT는 출력자료 클래스이고, mb는 메소드를 구현하는 본체이다. 개념적으로 메소드는 한 클래스와 연관된 함수이다.

정의 2.1의 역할 할당함수에서 관계의미가 일반화인 경우에 τ와 σ가 전체 클래스 C에 대한 한 클래스라면, $\tau \leq \sigma$로 표기되는 σ와 τ의 부분 클래스화는 다음처럼 재귀적으로 정의된다.

[정의 3.2] 부분 클래스화 및 일반화 상속
(i) τ_b가 기본 클래스라면 $\tau_b \leq \tau_b$
(ii) 집합이나 참조 구성자에 대한 클래스에서 $\tau \leq \sigma$이면 $\{\tau\} \leq \{\sigma\}$과 ref $\tau \leq$ref σ
(iii) 튜플 구성자에 대한 클래스 τ와 σ에 대해서 $\tau_i \leq \sigma_i$, i=1, n이면, $(A_1: \tau_1, \cdots, A_n: \tau_n) \leq (A_1: \sigma_1, \cdots, A_m: \sigma_m)(n \leq m)$

위 정의 3.2에 의해서 부분 클래스는 부분 순서(partial order)를 이룬다[6]. 루트 클래스로써 시스템 정의 클래스인 OBJECT를 가질 때, 임의의 두 클래스는 언제나 격자(lattice)에서 최소 상한 클래스(least upper class)를 가지므로 이 순서는 클래스의 반격자(semi-lattice)를 이룬다. 일반화 상속은 클래스화에 기반한 클래스 합성기법이다. 그래서 클래스 τ에 의해서 규정된 모든 속성들이 클래스 σ에서도 사용할 수 있다면($\tau \leq \sigma$), 클래스 τ로부터 클래스 σ로 속성 및 메소드를 상속한다고 한다. 하나 이상의 상위 클래스부터 한 부분 클래스가 규정된다면, 다중 상속이라 한다. 부분 클래스화에 기반하여 부분 클래스 관계를 정의할 수 있다. 두 클래스 (σ, X_σ)와 (τ, X_τ) 사이에 부분 클래스 관계가 존재한다면, $\sigma, \tau \in C$에 대해서 $\tau \leq \sigma$(내포성)와 $X_\tau \geq X_\sigma$(외연성)가 성립한다.

[정의 3.3] C가 클래스의 집합이며, O가 클래스 C에 연관된 객체의 집합이라 하자. O에 대한 집단화 클래스 계층(C, $\succeq$)은 C로 표현하고 집단화 참조관계는 $\succeq$로 표현되는 클래스 구조이다.

그래서 정의 3.3에 의해서 집단화는 성분 클래스들을 튜플 구성자들에 의해 합성 클래스로 추상화하는 방법이다. 이 방법은 공유 참조 여부에 따라 독점 참조와 공유 참조로 구분되며, 그 성분 클래스가 상위 클래스로부터 일반화 상속성 여부에 따라 의존 참조와 독립 참조로 구분할 수 있다[19].

[정의 3.4] C가 클래스의 집합이며, O가 클래스 C에 연관된 객체의 집합이라 하자. O에 대한 연관화 클래스 계층(C, ≒)은 C로 연관화 참조관계 ≒로 표현되는 클래스 구조이다.

그래서 정의 3.4에 의해 개체 E_i와 E_j가 서로 연관화 추상성의 관계를 가질 경우, 양방향성 속성-정의역 관계를 형성하므로 집단화 추상성에서 객체 지향 모델로 변환 시 역참조 관계가 표시되며, 원자-정의역에 대한 특성을 집단화 추상성과 같은 방법으로 변환된다.

임의의 두 클래스 사이에 집단화 관계가 성립하면 종속성, 추이성과 단방향성에 의해서 합성클래스와 부품 클래스 관계가 성립하므로 다음과 같이 집단화 상속을 정의할 수 있다.

[정의 3.5] 집단화 상속
클래스 τ와 σ 사이에 집단화 참조($\sigma \sqsubset \tau$) 관계는 정의 3.2에 의해서 부분 순서관계를 형성하고 시스템 정의 클래스인 OBJECT에 의해서 반격자 구조를 이룬다. 이 집단화 참조는 단방향으로 강한 결집력을 가진다. 그래서 집단화 참조에 기반한 클래스 합성 기법은 성분 클래스 σ의 속성들은 클래스 τ에 구문상(syntactic)으로 재사용할 수 있으며, 강제성(coercion)에 의한 다형성을 이룬다[22][21]. 이런 상속성을 집단화 상속성(aggregation inheritance)이라 한다. 두 클래스 (σ, X_σ)와 (τ, X_τ) 사이에 집단화 관계가 존재한다면, $\sigma, \tau \in T$에 대해서 $\tau \leq \sigma$ (내포성)와 $X_\sigma \leq X_\tau$(외연성)가 성립한다.

3.2 객체 변환 단계

E-R 모델을 객체 지향 모델로 변환할 때 본 논문에서는 E-R 모델의 각 성분을 변환하는 방법을 고려해야 한다. E-R 모델 측면에서는 개체, 개체 타입, 관계, 관계 타입, 역할, 속성, 값과 값의 집합을 거론하였다. 객체 지향 모델에서는 객체, 클래스, 일반화 관계, 집단화 관계, 연관화 관계, 메소드를 거론하였다. 그래서 본 절에서는 객체 지향 모델의 개념으로서 E-R 모델의 관계 클래스, 역할과 속성을 이용할 수 있다. 개체 클래스는 클래스 스킴으로 변환되며, 개체 클래스 간의 일반화는 클래스 계층으로 변환된다. E-R 모델의 관계는 의미에 따라 집단화와 연관화로 구분되며, 집단화 참조는 집단화 클래스 계층을 이루며, 연관화 참조는 연관화 클래스 계층을 이룬다. E-R 모델에서 개체 클래스 간의 관계를 객체 지향 모델에서 속성과 도메인으로 변환하기 위해서는 관계의 역할이 속성이 되며, 도메인은 참조되는 개체 클래스가 된다. 이에 대한 변환 단계는 다음과 같다.

3.2.1 변환 단계

단계 1. 개체-관계 모델에서 속성은 데이터 타입의 복잡성, 다중성과 저장성에 의해서 변환되고, 이 변환의 제약조건을 위해서 농도 제약조건, 참여 제약조건과 분리 제약조건은 각각 집합 구성자 및 튜플 구성자, "null" 키워드와 일반화 의미로 변환한다. 이에 대한 구체적인 변환 방법은 다음과 같다.

단계 1.1 데이터 타입의 표현에 의한 변환
정의 3.1의 클래스 스킴에서 속성의 종류에 따라 다음과 같이 변환된다.

단계 1.1.1 속성의 복잡성에 의한 방법
객체의 복잡성 여부에 따라 단순 속성과 복합 속성으로 변환하고, 단순 속성의 정의역은 기본

자료형을 이용하며, 복합 속성의 정의역은 튜플 구성자를 이용한다.

단계 1.1.2 속성의 다중성에 의한 방법

객체의 다중성 여부에 따라 단일 속성과 다중 속성으로 변환하고, 단일 속성은 집합 구성자를 이용치 않고, 다중 속성은 집합 구성자를 이용한다.

단계 1.1.3 속성의 저장성에 의한 방법

객체의 저장성 여부에 따라 저장 속성과 유도 속성으로 변환하고, 저장 속성은 이미 데이터베이스에 있는 자료를 이용하고, 유도 속성은 저장 자료를 계산하여 이용한다.

단계 1.2 제약조건에 의한 변환

위 단계 1.1의 변환 방법을 개체-관계 모델의 제약조건에 적용하여 다음과 같이 변환 할 수 있다.

단계 1.2.1 농도 제약조건

일반화에 의한 농도 제약조건은 E-R 모델에서의 하위 개체 타입과 상위 개체 타입을 각각 부분 클래스와 상위 클래스로 변환하고, 상위 클래스의 개수가 한 개일 경우에는 클래스 정의 구문에 IS-A를 명시한다. 상위 클래스의 개수가 두 개 이상일 경우에 합집합 제약조건은 IS-A^{-1}(u)를 명시하고, 교집합 제약조건은 IS-A^{-1}(i)를 명시한다. 세부화에 의한 농도 제약조건은 ER 모델에서의 하위 개체 타입과 상위 개체 타입을 각각 부분 클래스와 상위 클래스로 변환하고, 하위 클래스의 개수가 한 개일 경우에는 클래스 정의 구문에 IS-A를 명시한다. 하위 클래스의 개수가 두 개 이상일 경우에 분리 제약조건은 IS-A(d)를 명시하고, 중첩 제약조건은 IS-A(o)를 명시한다. 연관화와 집단화에 대한 농도 제약조건 1:1과 N:1 관계는 단계 1.1.1과 단계 1.1.2에 의해서 기본 데이터 타입과 튜플 구성자에 의해서 단일값 속성-영역으로 변환하며, 1:N과 M:N 관계는 단계 1.1.1과 단계 1.1.2에 의해서 집합 구성자에 기본 데이터 타입이나 튜플 구성자를 조합시켜 다중 값 속성-영역 관계로 변환한다.

단계 1.2.2 참여 제약조건

정리 3.2와 정리 3.4에 의해서 일반화, 세부화, 연관화와 집단화에 대한 참여 제약조건에서 제약조건이 k = 0이면, 클래스 정의 구문에서 명시적으로 변환하지 않고, 제약조건이 k = 1이면, 명시적으로 "total" 이라는 키워드로 변환한다.

단계 2. 정규 개체 타입(regular entity type) E_R는 개체 클래스 C_{RE}로 변환한다.

키가 존재하는 개체 타입으로써 단계 1.1의 데이터 타입에 의한 속성의 종류에 따라 기본 클래스, 합성 클래스와 집합 클래스로 변환한다.

단계 3. 약 개체 타입(weak entity type) E_W는 성분 클래스 CC_i로 변환한다.

키가 존재하는 개체 타입은 합성 클래스로 변환하고, 약 개체 타입은 성분 클래스로 변환한다.

단계 4. 관계 타입 R은 관계 클래스 C_R로 변환한다.

단계 1에 의해서 자체 속성들의 정의역을 규정하고, 관계하는 개체들을 지칭하기 위해서 특수 키워드를 사용한다.

단계 5. 개체-관계 모델에 없는 메소드를 추가한다.

단계 1의 유도 속성의 값을 계산하기 위해서 개체-관계 모델에 없는 메소드를 해당 클래스에 추가한다.

단계 6. 개체-관계모델의 관계 R에서 역할 role의 의미에 따라 속성과 영역 관계를 형성하기 위해서 일반화, 집단화와 연관화로 변환된다.

단계 6.1 두 클래스 사이에 재사용성 여부에

의해 관계 R의 의미를 일반화 및 세부화로 변환

정의 3.2에 의해서 관계 R의 의미를 참여 완벽성과 분리성의 제약조건 하에 일반화로 변환한다. 일반화의 역함수는 세부화이다. 두 클래스 사이에 계층을 이루기 때문에 클래스 계층을 형성한다. 이는 임의의 두 클래스 CO_i와 CO_j에 대해서 CO_i가 CO_j의 특성 및 메소드를 계승받음을 의미하며, CO_i Rge CO_j로 표기한다. 그래서 CO_i는 변환된 상위 클래스이고, CO_j는 변환된 하위 클래스이며, 관계 R의 의미는 Rge로 변환된다.

단계 6.2 두 클래스 사이에 부품과 합성 관계에 의해서 관계 R의 의미를 집단화로 변환

정의 3.3에 의해서 합성 클래스와 성분 클래스 관계를 형성한다. 임의의 두 클래스 CO_i와 CO_j에 대해서 CO_i가 CO_j에 집단화 관계성에 의해 CO_j가 강하게 결합되어 종속적으로 운행되며, CO_i Rag CO_j로 표기한다. 강한 결속력을 가진 합성 클래스로부터 속성과 메소드를 집단화 상속을 의미한다. 그래서 CO_i는 변환된 합성 클래스이고, CO_j는 변환된 부품 클래스이며, 관계 R의 의미는 Rag로 변환된다.

단계 6.3 독립적인 두 클래스의 연결성에 의해서 관계 R의 의미를 연관화로 변환

정의 3.4에 의해서 관계 R은 재귀적, 순환적, 그리고 생성적 연결성에 의해 두 클래스를 연결한다. 한 클래스가 다른 클래스를 서로 쌍방향적으로 연관성을 이루기 때문에 연관 클래스 계층을 형성한다. 이는 임의의 두 클래스 CO_i와 CO_j에 대해서 CO_i가 CO_j에 연관화 관계성에 의해 CO_j가 느슨하게 결합되어 운행되며, CO_i Ras CO_j로 표기한다. 그래서 CO_i는 변환된 속성에 해당하는 클래스이고, CO_j는 변환된 정의역 클래스이며, 순환성에 의해서 CO_i와 CO_j가 역참조가 가능하며, 그리고 관계 R의 의미는 Ras로 변환된다.

그래서 E-R 모델의 관계는 의미에 따라 집단화와 연관화로 구분되며, 집단화 참조는 집단화 클래스 계층을 이루며, 연관화 참조는 연관화 클래스 계층을 이룬다. E-R 모델에서 개체 클래스 간의 관계를 객체 지향 모델에서 속성과 정의역으로 변환하기 위해서는 관계의 역할이 속성이 되며, 정의역은 참조되는 개체 클래스가 된다. 또한 관계 의미에서 두 클래스 사이에 서로 결합되는 역할에 따라 속성-영역 관계를 다음과 같이 고찰할 수 있다.

3.2.2 관계의 역할을 속성 및 메소드로 변환

E-R 모델의 n-항 관계 $R(E_1, \ldots, E_i, \ldots, E_n)$을 변환 단계 2와 단계 3에 의해서 한 클래스 C_i와 다른 클래스 C_j 사이의 관계를 객체 지향 모델에서 일반화 관계 $R_{ge}(C_1, \ldots, C_i, \ldots, C_n)$, 집단화 관계 $R_{ag}(C_1, \ldots, C_i, \ldots, C_n)$와 연관화 관계 $R_{as}(C_1, \ldots, C_i, \ldots, C_n)$로 변환된다. 이 일반화 관계는 두 클래스 간의 상속 관계를 형성하고, 집단화 관계는 두 클래스를 합성 클래스와 성분 클래스로 구분하여 연결하는 역할을 하며, 연관화 관계는 두 클래스를 독립적으로 연결하는 역할을 한다. 그래서 한 클래스 C_i와 다른 클래스 C_j가 상속성 및 속성-도메인 관계를 이루기 위해서는 E-R 모델에서 정의 2.1의 역할 할당함수에 의해서 관계의미의 역할에 따라 규정할 수 있다. 이 역할을 객체 지향 모델로 변환하기 위해서 정의 3.1의 메소드 M으로부터 한 클래스 C_i와 다른 클래스 C_j가 속성-도메인 관계를 메소드 요약을 이용하여 표현하면 다음과 같이 된다.

C_i 속성 이름(C_j) -----> C_j (식 1)

여기서 수신 클래스(receive class) C_i는 속성의 클래스가 되고, selector인 속성 이름은 E-R 모델의 역할 이름(role name)이 되며, 입력 클래스도 수신 클래스와 동일하며, 도메인에 해당하는 반환 클래스는 C_j가 된다. 그래서 역할의 기능을 메소

드 요약으로 표현하면 식 1은 다음과 같이 된다.

$C_i\ role_k(C_i)$ -----> C_j, $1 \le i \le n$, $1 \le j \le n$, $k \ne i \ne j$ (식 2)

가. 일반화와 세분화 클래스 계층에 의한 메소드 일반화 관계 의미를 가지는 경우에 역할을 selector로 표현하면 다음과 같다.

C_i role-name$_k(C_i)$ ----> C_j, $1 \le i \le m$, $1 \le j \le n$, $1 \le k \le n-1$, $k \ne i \ne j$ (식 3)

여기서 m은 m항 관계의 수이고, j는 하위 클래스 C_i와 일반화 관계에 있는 상위 클래스 C_j의 수를 나타낸다. 또한 이 role-name$_k$은 하위 클래스와 상위 클래스가 일반화 관계의 의미를 나타내는 "is-a"를 형성한다. 그러나 역할 이름 role-name$_k$은 속성이름으로 변환하지 않고 상위 클래스의 속성 및 메소드를 하위 클래스로 상속시키는 역할을 한다. 그래서 역할 이름의 수는 (n-1)개다. 이 역할 이름은 객체의 상태를 나타내는 것이 아니고, 객체의 상속행위를 나타내므로 메소드 이름이다.

세부화 관계 의미를 가지는 경우에 역할을 selector로 표현하면 다음과 같다.

$C_j\ role-name_k^{-1}(C_j)$ -----> C_i, $1 \le i \le m$, $1 \le j \le n$, $1 \le k \le n-1$, $k \ne i \ne j$ (식 4)

여기서 m은 m항 관계의 수이고, j는 하위 클래스 C_i와 세부화 관계에 있는 상위 클래스 C_j의 수를 나타낸다. 또한 이 $role-name_k^{-1}$은 상위 클래스와 하위 클래스가 세부화 관계의 의미를 나타내는 "is-a-1"를 성형한다. 그러나 역할 이름 $role-name_k^{-1}$의 기능은 속성 이름으로 변환하지 않고, 하위 클래스의 속성 및 메소드를 상위 클래스로부터 상속받는 역할을 한다. 그래서 역할 이름의 수는 (n-1)개다. 이 역할 이름은 객체의 상태를 나타내는 것이 아니고, 객체의 상속 행위를 나타내므로 메소드 이름이다.

[정리 3.1] E-R 모델의 일반화와 세부화에 대한 역할 기능은 객체 지향 모델에서 메소드이다.

증명 : E-R 모델의 일반화와 세부화에 대한 역할 기능은 3.2.1절의 변환단계 1.2.1의 농도 제약 조건에 따라 객체 지향 모델에서 3.2.2절의 식 2, 식 3과 식 4에 의해서 재사용 기능을 메소드로 변환한다.

나. 연관화 클래스 계층에 의한 속성-도메인

연관화 참조의 특징은 정의 3.4에 의해서 양방향성, 재귀성, 생성성을 가진다. 양방향성에 의해서 수신 클래스가 임의의 한 클래스로 고정되지 않기 때문에 순환을 형성한다. 그래서 연관화 관계 $R_{as}(C_1, \ldots, C_i, \ldots, C_m)$에서 임의의 한 클래스 C_i에 대한 속성-도메인 관계의 일반식은 다음과 같이 된다.

C_i role-name$_{ijk}(C_i)$ ------> C_j, $1 \le i \le m$, $1 \le j \le m$, $1 \le k \le n$, $k \ne i \ne j$ (식 5)

위 식 5에서 i의 m은 m항 관계의 수를 나타내고, j의 m은 한 클래스 C_i와 연관 관계에 있는 클래스 C_j의 최대 수이다. k의 n은 한 클래스 C_i와 클래스 C_j가 연관화 관계에 있는 역할의 최대 수를 나타내며, 기존의 개체-관계 모델의 역할과 같은 의미이다[11][12]. 그래서 역할 이름 role-name$_{ijk}$은 관계하는 두 클래스 사이에 연관화 의미 "has-a"를 형성한다. 이 역할 이름 role-name$_{ijk}$은 m×m×n 개의 속성 이름으로 변환된다. 단계 5.3에 의해서 연관화는 역함수 관계가 다음과 같이 성립한다.

$C_j\ role-name_{ijk}^{-1}(C_j)$ ------> C_i, $1 \le i \le m$, $1 \le j \le m$, $1 \le k \le n$, $k \ne i \ne j$ (식 6)

위 식 6에서 i의 m은 m항 관계의 수를 나타내고, j의 m은 한 클래스 C_i와 역(inverse) 연관 관계에 있는 클래스 C_j의 최대 수이다. k의 n은 한 클래스 C_i와 클래스 C_j가 연관화 관계에 있는 역 역할(inverse role)의 최대 수를 나타낸다. 그래서 역 역할 이름 $role-name_{ijk}^{-1}$은 관계하는 두 클래스 사이에 연관화 의미 "have been"를 형성한다. 이 역 역할 이름 $role-name_{ijk}^{-1}$은 m×m×n개의 속성 이름으로 변환된다. 위 식 6과 같은 의미를 가지는 수식은 다음과 같다.

$$C_j \text{ role-name}_{jik}(C_j) \text{ ---> } C_i,\ 1 \le i \le m,\ 1 \le j \le n,\ 1 \le k \le o,\ k \ne i \ne j \qquad \text{(식 7)}$$

위 식 6과 식 7에서 $role-name_{ijk}^{-1}$와 role-name$_{jik}$는 같은 의미를 가진다. 그래서 전체 역할 이름이 속성 이름으로 변환하기 위한 수는 m×m×n이다.

다. 집단화 클래스 계층에 의한 속성-도메인

집단화 클래스인 경우에는 집단화 참조의 부분 순서성에 의해서 수신 클래스가 임의의 한 클래스로 고정되며, 재귀적 운행과 순환을 형성하지 못한다. 그래서 집단화 관계 $R_{ag}(C_1, \dots, C_i, \dots, C_m)$에서 합성 클래스가 $C_i(1 \le i \le m)$라면, 속성-도메인 관계의 일반식은 다음과 같이 된다.

$$C_i \text{ role-name}_{ijk}(C_i) \text{ ------> } C_j,\ 1 \le i \le m,\ 1 \le j \le m,\ 1 \le k \le n,\ k \ne i \ne j \qquad \text{(식 8)}$$

위 식 8에서 i의 m은 m항 관계의 수를 나타내고, j의 m은 한 클래스 C_i와 집단화 관계에 있는 클래스 C_j의 최대 수이다. k의 n은 한 클래스 C_i와 클래스 C_j가 집단화 관계에 있는 역할의 최대 수를 나타내며, 기존의 개체-관계 모델의 역할과 같은 의미이다[11][12]. 그래서 역할 이름 role-name$_{ijk}$은 관계하는 두 클래스 사이에 집단화 의미 "part-of"를 형성한다. 이 역할 이름 role-name$_{ijk}$은 m×m×n개의 속성 이름으로 변환된다. 그러나 변환 단계 5.2의 단방향 특성에 의해서 식 6과 식 7의 $role-name_{ijk}^{-1}$와 role-name$_{jik}$ 같은 역함수 관계가 존재하지 않으며, 객체 버전화에 의한 비재귀적 집단화 속성 개수는 m(n-1)이며, 재귀적 속성 개수는 m×n이다. 그래서 전체 역할 이름이 집단화 속성 이름으로 변환되었을 때의 개수는 (2n - 1)×m이다.

3.2.3 객체 버전화에 의한 변환

시간 차원을 객체 지향 모델과 통합하기 위해서 비시간지원 객체 지향 모델의 변환 과정은 변환 단계 1-6을 그대로 이용하고, 여기에 시간 속성을 추가하면 객체 버전화를 위한 객체 지향 데이터 모델이 된다. 그래서 3.2.1의 역할 의미에 따른 속성-영역 관계의 변환 방법을 이용하여 다음의 식을 고려하여 보자.

<vta: T/TI, tta: T/TI, uta: T>> ..... ①
a: V ..... ②
rolename: V ..... ③
bt: ATC ..... ④
uta: T ..... ⑤
ag: ADT ..... ⑥
as: ref ADT ..... ⑦

위 ①식은 ER 모델에서 시간 속성들을 나타낸 것이고, ②식은 ER 모델에 대한 비시간 속성을 나타내며, ③식은 집단화와 연관화에 대한 역할 기능과 이에 대한 값을 표현한 것이다. ④식의 ATC는 이원시간 속성에 대한 정의역인 추상화 시간 클래스이며, ⑤식은 사용자 정의 시간의 속성을 나타낸다. ⑥식과 ⑦식은 정리 2.1과 2.1에 의해서 ①식을 객체 지향 모델로 변환했을 때 각각 집단화 관계와 연관화 관계에 의한 속성-영역 관계를 나타낸 것이다. 여기서 ag는 역할 이름에

대응하는 집단화 속성이름이고, ADT는 집단화 참조되는 클래스를 나타내는 정의역이다. 또한 as는 역할 이름에 대응하는 연관화 속성 이름이다. 이 객체 지향 시간 속성에서 이원시간을 위해서 추상화 시간 클래스를 규정하면 다음과 같다.

```
class abstract time class(ATC) : Object {
        valid time T/TI ;
        transaction time T/TI ;
        return {ATC};}
```

[정리 3.2] 연관화와 집단화에 대한 역할 기능은 객체 지향 모델에서 속성이다.

증명 : E-R 모델의 연관화와 집단화에 대한 역할이름은 위 ③식이며, 객체 지향 모델에서는 3.2.1절의 식 2, 5, 6, 7과 식 8에 의해서 ④와 ⑤식으로 변환된다. 여기서 as와 ag는 객체의 상태이므로 정리가 증명된다.

[정리 3.3] 사용자 정의 시간은 추상시간 클래스의 속성으로 사용치 못한다.

증명 : 사용자 정의 시간은 고정적인 시간값이므로 독립적인 원자 객체이므로 이원시간과 함께 합성 클래스화 하지 못하기 때문에 합성 클래스의 속성이 되지 못한다.

기본 자료형에 대한 정의역의 전체 집합은 U D = (D_1, D_2, ... , D_{nd}}이며, 각 D_i는 원자 i체의 집합이다. 여기서는 값의 집합이다. 여기서 각 i에 대해서 $D_i \neq \varnothing$이고, $Đ = \cup_{i=1}^{nd} D_i$는 모든 값의 집합이다. 객체 식별자의 전체 집합은 ID = {i_1, i_2, ... , i_n}이고, 이산시간의 정의역은 정의 2.1에서 규정한 T를 이용한다. 비시간 속성의 전체 집합은 U_A = {A_1, A_2, ... , A_{nd}}이고, 시간 속성의 전체 집합은 U_{TA} = {bt, uta}이다. 여기서 bt는 유효시간에 대한 속성과 거래시간에 대한 속성의 합성이다. uta는 사용자 정의 시간에 대한 속성이다. 이 이원시간의 정의역은 ATC이며, 이 사용자 정의 시간에 대한 정의역은 T이다.

시간지원 객체 지향 객체 버전화를 위한 개체 클래스, 관계 클래스, 속성, 역할, 제약조건 및 정의역을 위한 스킴은 다음과 같다.

[정의 3.11] 객체 지향 객체 버전화를 위한 클래스 스킴 C = <C_E, C_R, б, ρ, D, F, AT>이다. 각 성분은 다음과 같이 구성된다.

◎ 개체 클래스의 유한 집합 C_E

$C_E = \{C_{E1}, \ldots, C_{Ei}, \ldots, C_{En}\}$
$= C_{RE} \cup C_W = \{E_{RE1}, \ldots, E_{REp}, \ldots, E_{REs}\} \cup \{E_{W1}, \ldots, E_{Wq}, \ldots, E_{Wt}\}$

여기서 C_{RE}는 변환단계 2에 의해서 정규 개체 타입에 대응하는 클래스이고, C_W는 변환단계 3에 의한 개체 타입에 대한 클래스이다.

◎ 관계 클래스 유한 집합 C_R

$C_R = \{C_{R1}, \ldots, C_{Ri}, \ldots C_{Rn}\}$

여기서 C_R은 변환단계 3에 의해서 관계 타입에 대한 클래스이다.

◎ 관계 타입 이름의 요약(signature) б

б: $C_R \rightarrow C_E^{\times}$

여기서 관계 클래스 C_R에 의해서 관계되는 개체 클래스 $C_E^{\times} = C_{E1} \times \ldots C_{Ei} \times \ldots \times C_{En}$이다.

◎ 기본 객체들의 유한 집합 D

$D = \{D_1, \ldots, D_i, \ldots, D_n\}$

여기서 기본 데이터 타입의 값에 대응하는 정의역이다.

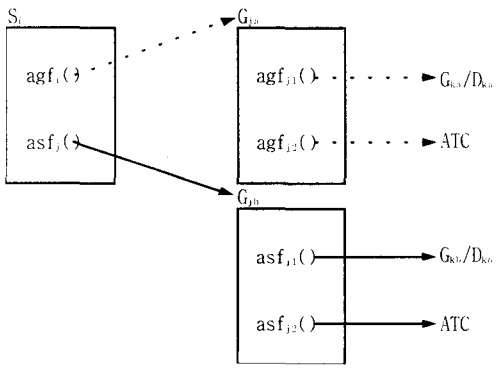
◎ 속성 타입 함수 AT

AT: $F \rightarrow \{(C_{RE} \cup C_W) \rightarrow D^{\times} \cup C^{\times} \cup <D \times T \times ATC>^{\times} \cup <C \times T \times ATC>^{\times}\}$

여기서 변환단계 2, 3, 그리고 정의 2.3에 의해서 $F = \{F_A, F_{TA}\}$는 전체 속성 A에 대한 속성 이름이다. F_A는 전체 비시간 속성 이름이고, 이에 대한 정의역은 이 비시간 속성의 정의역 ND는 원자 객체의 정의역 D와 합성 클래스 C의 조합이다. 즉 $ND = D^{\times} \cup C^{\times}$이다. 여기서 $D\times = D_1 \times ... \times D_i \times ... \times D_n$이며, $C^{\times} = C_1 \times ... \times C_i \times ... \times C_n$이다. 시간 속성 FTA는 시간 속성 이름이고, 이에 대한 정의역은 $T \times ATC$이며, T는 사용자 정의 시간에 대한 정의역이며, ATC는 이원시간에 대한 정의역이다. 또한 비시간 속성과 시간 속성을 혼합한 정의역은 원자 객체에 대한 시간 정의역 $<D \times T \times ATC>^{\times}$과 복합객체에 대한 정의역 $<C \times T \times ATC>^{\times}$의 조합이다. 그래서 타입 버전화를 위한 속성 이름은 $F = <F_A, F_{TA}>$이고, $F_A \subseteq U_A$이며, $F_{TA} \subseteq U_{TA}$이다.

그래서 일반화, 연관화와 집단화를 지원하는 속성 버전화는 재귀적으로 규정할 수 있다. 이러한 속성 버전화를 위한 운행 방법은 그림 4와 같다.

그림 4에서 S_i, G_{ja},와 G_{jb}는 개체 클래스를 나타내고, agf()는 집단화 속성함수를 나타내며, asf()는 연관화 속성함수를 나타내고, d는 원자 객체를 나타낸다. 이 속성함수의 정의역으로써 정의 2.1에 의해서 G_{ka}와 G_{kb}는 비시간 값들을 나타내며, D_{ka}와 D_{kb}는 원자 객체를 나타낸다. 그러나 이원시간 속성에 의한 정의역 A_{TC}는 나타내지 않는다. 이에 대한 구체적인 속성 영역의 함수를 유도하는 과정은 3.2.1절의 식 5, 식 6과 식 8에 의해서 다음과 같다.



(그림 4) 시간지원 객체 버전화

(a) 참조 영역이 G_{ka}와 G_{kb}인 경우의 시간지원 객체 버전화

ⓘ 집단화에 의한 경우

- $agf_i(S_i) \rightarrow G_{ja}$, 여기서 $G_{ja} = agf_{j1}()$
  그래서 집단화 추상화의 재귀적인 합성 객체의 클래스 G_{ja}는 다음과 같이 된다.
- $agf_{j1}(S_i) \rightarrow G_{ka}$

ⓙ 연관화에 의한 경우

- $asf_i(S_i) \rightarrow$ **ref** G_{jb}, 여기서 $G_{jb} = asf_{j1}()$
  그래서 연관화 추상화의 재귀적인 연관객체의 클래스 G_{jb}는 다음과 같이 된다.
- $asf_{j1}(S_i) \rightarrow$ **ref** G_{kb}

ⓘ와 ⓙ에서 G_{ka}와 G_{kb}는 정의역으로 원자 객체가 아닌 복합 객체를 나타내므로 재귀적으로 경로식을 운행하는 속성들이다. 만일에 경로식을 종결하기 위해서는 정의역이 원자 객체를 가지면 된다.

(b) 참조 영역이 d_{ka}와 d_{kb}인 경우의 시간지원 속성 버전화

ⓘ 집단화에 의한 경우

- $agf_i(S_i) \rightarrow G_{ja}$, 여기서 $G_{ja} = agf_{j1}()$
  그래서 집단화 추상화의 재귀적인 원자 객체의 클래스 G_{ja}는 다음과 같이 된다.
- $agf_{j1}(S_i) \rightarrow G_{ka}/d_{ka}$

ⓙ 연관화에 의한 경우

- $asf_i(S_i) \rightarrow$ **ref** G_{jb}, 여기서 $G_{jb} = asf_{j1}()$
  그래서 연관화 추상화의 재귀적인 원자

객체의 클래스 G_{jb}는 다음과 같이 된다.

- $asf_{j1}(S_i) \rightarrow$ (**ref** G_{kb})/d_{kb}

ⓘ와 ⓙ에서 d_{ka}와 d_{kb}는 정의역으로 원자 객체를 나타내므로 재귀적으로 경로식을 운행하지 못하며, 자체 객체로 객체 버전의 객체들을 가진다.

이 시간지원 객체 지향 모델의 객체 버전화를 위한 실체 타입을 변환 단계 1~6과 역할에 따라 클래스로 변화시켰을 때, 클래스 스킴 정의는 다음과 같다.

[정의 3.12] 실체 클래스 스킴 C_{OUE} = <F, OID, ≗, ≚, ≙>

◎ 속성들의 집합: F = {F_A, F_{TA}, uta}이다. 여기서 $F_A \subseteq U_A$, $F_{TA} \subseteq U_{TA}$, uta$\subseteq U_{TA}$이다.
◎ 식별자들의 집합: OID ∪ {Time}, 여기서 OID ⊆ ID 이다.
◎ 원자 객체 정의역의 집합: DOM: F_A ∪ uta --> U_D ∪ {T}
◎ 일반화 클래스 계층: ≗ ⊆ C×C
◎ 집단화 클래스 계층: ≚ ⊆ C×C
◎ 연관화 클래스 계층: ≙ ⊆ C×C

객체 버전화를 위한 관계성 타입을 클래스로 변환시켰을 때 관계성 스킴은 다음과 같다.

[정의 3.13] 관계성 클래스 스킴 C_{OUR} = <F, OID, EC, ATC, RAC>

◎ 속성들의 집합 :F = {F_A, F_{TA}, uta}이다. 여기서 $F_A \subseteq U_A$, $F_{TA} \subseteq U_{TA}$, uta$\subseteq U_{TA}$이다.
◎ 식별자들의 집합: OID ∪ {Time}, 여기서 OID ⊆ ID이다.
◎ 원자 객체 정의역의 집합: DOM: F_A ∪ uta --> U_D ∪ {T}
◎ EC: entity class
◎ ATC: abstract temporal class
◎ RAC: relationship atomic class

3.2절에서 제시한 변환단계, 정의 3.6, 정의 3.7과 정의 3.8에 의해서 제 2장의 시간지원 E-R 모델을 시간지원 객체 지향 모델로 변환한 스키마는 다음의 그림 5와 같다.

```
class PERSON  = (
   pname          : string;
   birthday       : date;
   home_address   : CITY_ADDRS,
                    independent & sharable, total;
   children       : {ref PERSON});
   bt             : ATC;);
end class

class EMPLOYEE = (
   IsA            : PERSON, total;
   ssn            : string;
   hiredate       : date;
   salary         : real;
   manages        : ref DEPARTMENT;
    //inverse of DEPARTMENT. manager
   belongs_to     : ref DEPARTMENT;
    //inverse of DEPARTMENT.Staff
    //inverse of DEPARTMENT.workers
   works_for      : {<dept_of,: ref DEPARTMENT,
                     Hours: real>};
   affiliates     : ref COMPANY, total;
   works_on       : {ref PROJECT};
   bt             : ATC;
   method void give-raise(percent: real););
end class EMPLOYEE;

class COMPANY = (
   cpname         : string;
   location       : CITY_ADDRS,
                    independent & sharable, total;
   president      : {ref EMPLOYEE},total;
   cpstaff        : {ref EMPLOYEE};
   projects       : {ref PROJECT};);
   bt             : ATC;);
end class COMPANY;
```

```
class CITY_ADDRS = (
   constituent_of : COMPANY, PERSON;
   cityname       : string;
   zipcode        : string;
   bt             : ATC;);
end class CITY_ADDRS;

class PROJECT = (
   contract#      : integer;
   proj_name      : string;
   leader         : ref EMPLOYEE;
   budget         : integer;
   documents      : {ref DOCUMENT};);
   bt             : ATC;);
end class PROJECT;

class INTERAL_PROJECT = (
   IsA(o)         : PROJECT, total;
   IPno           : integer;
   bt             : ATC;);
end class INTERAL_PROJECT;

class FUNDED_PROJECT = (
   IsA(o)         : PROJECT, total;
   FPno           : integer;
   bt             : ATC;);
end class FUNDED_PROJECT;

class FOUNDATION_PROJECT = (
   IsA(d)         : FUNDED_PROJECT, total;
   FOPno          : integer;
   bt             : ATC;);
end class FOUNDATION_PROJECT;

class CORPORATE_PROJECT = (
   IsA(d)         : FUNDED_PROJECT, total;
   CPno           : integer;
   bt             : ATC;);
end class CORPERATE_PROJECT;
```

(그림 5) 객체 버전화를 위한 시간지원 객체 지향 데이터베이스 스키마

그림 5에서 개체 클래스들은 CITY_ADDRS, COMPANY, PERSON, EMPLOYEE, PROJECT, DEPARTMENT 등이며, 속성에 있어서 zipcode는 비시간 속성을 나타내고, at는 시간 속성을 나타내며, 이에 대한 정의역은 ATC이다. <home_address : CITY_ADDRS> total에서 home_address는 집단화 관계에 의한 속성이름이며, 이 속성의 객체의 정의역인 CITY_ADDRS는 관계하는 개체 클래스를 나타낸다. 또한 total은 두 개체 클래스 CITY_ADDRS와 PESSON의 전체 제약조건을 나타낸다. <staff: {ref EMPLOYEE>}total에서 staff는 연관화 관계에 의한 속성 이름이며, 이 속성의 객체의 정의역 EMPLOYEE는 관계하는 개체 클래스를 참조하는 클래스이며, total은 두 개체 클래스 COMPANY와 EMPLOYEE가 전체 제약조건을 의미하며, "{}"은 ER 모델의 1:N 대응 관계를 객체 지향의 집합 구성자로 변환했을 때의 구성자이다.

4. 비교 및 검토

4.1절에서는 관계 의미에 따른 역할을 분석하고 객체 지향 특성 및 시간 속성을 비교 및 분석한다. 그리고 4.2절에서는 객체 버전화에 대해 검토한다.

4.1 객체 및 시간 특성 분석

시간지원 E-R 모델을 시간지원 객체 지향 모델로 변환했을 때 제시한 모델과 비교하기 위한 기준은 2-항 관계, n-항 관계, 속성의 유무에 의한 관계, 농도 제약조건, 참여 제약조건, 일반화 및 세부화, 집단화 그리고 연관화를 포함하고 있다.

농도 제약조건은 집단화 참조에 의한 제약조건과 연관화 참조에 의한 제약조건으로 구분한다. 우선 집단화 참조에서 합성 클래스에 있는 속성과 영역에 속하는 성분 클래스는 단방향으로 운행하기 때문에 1:1과 1:M만 수용한다. 또한 연관화 참조에서 해당 클래스들이 독립적으로 연관되어 쌍방향으로 운행되기 때문에 1:1, 1:N, 그리고 M:N의 농도 제약조건을 가진다.

(표 1) 관계 의미에 따른 역할 분석

관계의미 / 역할특성	일반화	세부화	집단화	연관화
역할 개수	n-1	n-1	(2n - 1)m	m×m×n
기 능	일반화 메소드	세부화 메소드	집단화 속성	연관화 속성

참여 제약조건은 연관화와 집단화 관계에서 전체와 부분 제약조건을 모두 운행하며, "total"이라는 키워드의 유무에 따라 구분하고 있으며, 시간 속성 표현은 시점, 시간 간격, 그리고 시간 요소의 정의역에 따라 유효시간 속성, 거래시간 속성, 그리고 사용자 정의 시간 속성으로 구분하여 시간지원 모델을 설계할 수 있다.

E-R 모델에서 관계 의미와 관계 기능에 따라 관계 타입과 역할을 각각 객체 지향 모델의 재사용성 의미 및 속성 기능으로 변환할 수 있다. 우선 관계 의미는 일반화 및 세부화, 집단화와 연관화로 표현하고, 역할은 관계 의미의 기능에 따라 재사용성, 집단화 속성과 연관화 속성으로 나누어진다. 그리고 다형성 관점에서 일반화 상속과 집단화 상속을 규정하여 속성 및 메소드의 재사용을 증대시켰다.

위 표 1에서 본 논문은 개체-관계 모델에 대한 관계 의미를 일반화, 세부화, 집단화와 연관화로 구분하여 사용하였으며, 개체 타입 간의 역할 기능을 규정함으로써 일반화 및 세부화의 재사용을 증가시키고, 개체 간을 연결하기 위해서 집단 속성과 연관 속성을 도입하였다. 또한 각 관계에 따른 역할 개수를 형식적으로 명시하였다.

4.2 검토

본 논문은 객체 버전화를 위한 클래스 수, 데이터베이스 수, 시간 단위와 제약조건에 대해 다음과 같이 검토할 수 있다.

기준 1. 규정된 시간지원 데이터베이스를 모델링하기 위해서 필요한 추가적인 클래스의 수

개체-관계 모델의 개체 타입과 관계 타입에 대응해서 객체 지향 모델에서 각각 개체 클래스와 관계 클래스로 변환된다. 객체 버전화에 의한 객체 지향 모델은 개체 클래스, 관계 클래스, 그리고 이 개체와 관계 클래스마다 이력사항을 요구하는 추상화 시간 클래스가 포함된다. 그래서 객체 버전화에 필요한 클래스의 수는 #E + #R + 2 × #ATC가 된다. 결국 관계하는 클래스의 수가 많을수록 객체 버전화 클래스의 수가 적다.

기준 2. 실세계 객체를 표현하기 위한 데이터베이스의 수

각 실세계 개체는 두 버전화에 같이 데이터베이스의 객체 사이에 일대일 대응한다. 그러나 개체의 속성에 값들이 데이터베이스화 했을 때 일반화와 집단화의 상속성에 의해 감소한다.

기준 3. 정보의 시간지원 단위의 granularity

객체 버전화는 이원시간을 일률적으로 각 개체 클래스와 관계 클래스에 단 한번만 포함시켜 사용한다.

기준 4. 시간지원 제약조건을 강화하기 위해서 도입된 추가 제약조건의 수

객체 버전화는 객체 상에서와 객체 사이에 제약조건들을 강화하기 위해서 시간지원 무결성 검토를 도입한다. 그 제약조건들이 다르게 보일지라도 둘 다 유사한 시간지원 규칙들을 지원한다.

그래서 객체 버전화 모델은 시간값과 비시간값을 개별적으로 표현한다. 집단화 참조에 의한 속성-영역 관계는 정의역을 성분 클래스만 표현하며, 연관화 참조에 의한 속성-영역 관계는 정의역을 연관된 클래스 만을 표현한다. 또한 시간 속성을 사용자 정의 시간과 이원시간으로 나누어 표현하며, 이원시간을 추상화 시간 클래스화하여

개체 클래스와 관계 클래스마다 시간 속성을 표기한다.

5. 결 론

본 논문은 개념적 모델인 개체-관계 모델에 시간 요소를 추가하여 실세계의 이력자료를 표현하고, 이에 대응하는 논리적인 모델인 객체 지향 모델로 변환하는 기법에 대해 연구하였다. 시간의 흐름에 따라 변경되는 이력자료를 표현하기 위해서 개체-관계 모델을 정의하였으며, 이를 컴퓨터 시스템에 구현하기 위한 모델인 객체 지향 모델을 규정하였다. 객체 지향 모델 하에 시간지원 개체-관계 모델을 변환 알고리즘에 의해 시간지원 객체 지향 모델로 변환하였으며, 변환에 따른 제약조건을 규정하였다.

시간지원 개체-관계 모델을 객체 지향 모델로 변환하기 위한 제약조건과 타입의 직교성을 이용한 객체 버전화에 따른 클래스 계층 구조를 설계하였다. 이러한 모델은 확장된 개체-관계 모델에 시간 개념을 추가하여 객체 지향 모델로 변환 했을 때 제약조건에 맞는 대응 관계를 고찰함으로써 개념적 이력자료를 논리적 이력자료로 변환하는 문제점을 해결하였다. 본 연구는 실세계의 이력자료를 개체-관계 모델로 표현하기 위해 일반화된 형태로 설계하고 도식화함으로써, 논리적 모델로 변환하는데 유용한 접근 방법이라 사료된다. 첫째, 시간 패러다임을 시간 표현 방법, 시간 결합 방법과 버전화에 적합한 수식으로 규정하는 것은 논리적 모델의 시간 정의역을 구현하는데 유용한 도구가 된다. 둘째, 개체-관계 모델에서 개체 사이의 관계를 관계 의미의 역할에 따라 일반화, 세부화, 집단화와 연관화로 구분하고, 객체 지향 모델의 클래스 계층 구조의 속성과 메소드를 규정하는데 편리한 방법이다. 셋째, 객체 변환 방법은 시간지원 객체 지향 모델의 속성 버전화와 객체 버전화의 선정에 있어서 결정적인 역할을 한다. 마지막으로 객체의 재사용성을 다형성 관점에서 고찰함으로써 객체의 검색 및 저장에 효율성을 기할 수 있다.

본 논문의 주요한 기여는 시간지원 개체-관계 모델의 형식화와 이 개념적 모델을 객체 지향 모델로 변환하는 방법을 제안하였다. 개체-관계 모델에서 관계 의미에 따른 역할을 객체 지향 모델의 속성과 메소드로 변환하는 방법을 도출하였으며, 이를 바탕으로 시간지원 클래스 정의어 구문을 설계하는데 효율성을 기할 수 있다. 이 접근 방법은 객체 지향 데이터베이스와 질의 처리 분야에 지대한 영향을 미친다. 그래서 시간지원 개체-관계 모델에서 관계 의미를 역할에 따라 규정함으로써 형식화를 이루었으며, 이를 기반으로 하여 객체 지향 모델로 변환하면 시간지원 클래스 스키마를 설계하는데 효율적인 방법이 된다. 개체-관계 모델의 구성 요소를 객체 지향 모델의 구성 요소로 변환하는 단계를 제안하였다. 또한 개체-관계 모델의 제약조건을 일반화, 세부화, 집단화와 연관화로 나누어 고려함으로써 객체 지향 모델로 변화하는데 효율적인 방법이다.

본 연구는 개념적 모델을 논리적 모델로 변환하는 기법에 대해 연구하였으며, 변환된 클래스 정의어 구문에 대한 검색 및 저장 비용에 대해서는 다루지 못하였다. 향후 연구로는 이 클래스 정의어 구문에 기발한 데이터베이스를 검색하는 인덱스 기법을 구현하여 효율성을 실제적으로 검토하는 연구 과제가 남아있다.

참 고 문 헌

[1] A. Ait-Braham, B. Theodoulidis, and G. Karvelis, "Conceptual modeling and manipulation of temporal databases," *Proc. Entity-Realtionship Conf.* 1994.

[2] Batini, C., Ceri, S. and Navathe, S., "Conceptual Database Design: Entity Relationship Approach," Redwood City: Benjamin/Cummings, 1992.

[3] 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.

[4] J. Ben-Zvi, "The Time Relational Model," Ph.D Thesis, Computer Science Dept., UCLA, 1982.

[5] P.A. Bernstein, V. Hadzilacos, and N. Goodman, *Concurrency Control and Recovery in Database Systems*, Addison-Wesley, 1987.

[6] Cardelli, L. and Wegner, P., "On Understanding types, Data Abstraction, and Polymorphism," ACM Computer Surv., Vol.17, No.4, December 1985, pp.471-522.

[7] Chen P.P., "The Entity-relationship Model-Toward a Unified View of Data," ACM TODS, Vol.1, No.1 May 1976, pp.9-35.

[8] James Clifford and Tomas Isakowitz. "On The Semantics of Transaction Time and Valid Time in Bitemporal Databases," Proceedings of International Workshop on an Infrastructure for Temporal Databases June 1993.

[9] Elmasri, R., El-Assal, I., Kouramajian, V., "Semantics of temporal data in an extended ER model," In 9th Entity-relationship Conference, 1990.

[10] Elmasri, R., and Wuu, G., "A temporal model and query language for ER database," In IEEE Data Engineering Conference, 1990.

[11] Elmasri, R., Kouramajian, V. and Fernando, S., "Temporal Database Modeling: An Object Oriented Approach," In international Conference on Information and Knowledge Management,, November 1993, pp.574-585.

[12] Elmasri, R., and Navathe, S., *Fundamentals of Database Systems*, 2nd E. D Benjamin/Cummings, 1993 [13] Gadia, S., and Yeung, C., "A generalized model for a temporal relational database," In ACM SIGMOD Conference, 1988.

[14] S.K. Gadia and C.S. Yeung, "A generalized model for a relational temporal database," *Proc. ACM Int'l Conf. Management of Data*, pp. 251-259, Chicago, June 1988.

[15] 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.

[16] I. Goralwalla and M.T. zsu, "Temporal extensions to a uniform behavioral object model," *Proc. Int'l Conf. Entity-Relationship Approach*, Dallas, June 1993.

[17] Jense, C., "Towards the realization of transaction time database system," Ph.D Dissertation, University of Maryland, 1990.

[18] 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.

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

[20] M.R. Klopprogge, "TERM: An approach to include the time dimension in the entity-relationship model," *Proc. Int'l Conf. Entity Relationship Approach*, pp. 477-512, Washington. D.C., Oct. 1981.

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

[22] Lee, H.R., et al., "Logical Optimization of Queries using a Complex Object Calculus Transformation," Journal of KISS(B): Software and Applications, Vo22, No.12, December 1995, pp.1601-1613.

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

[24] E. Mckenzie and R.T Snodgrass, "Supporting valid time in an historical relational algebra: Proofs and extensoins," Technical Report TR-91-15, Dept. of Computer Science, Univ. of Arizona, Tucson, Aug. 1991.

[25] Pissino, N., and Makki, K., "T-3DIS: an approach to temporal object databases," In international Conference on Information and Knowledge Management, 1992. pp.176-185.

[26] 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.

[27] E. Rose and A. Segev, "TOODM - A temporal object-orinted data model with temporal constraints," *Proc. Int'l Cong. Entity Relationship Approach*, Oct. 1991.

[28] Shipman P.P., "The Functional Data Model and the Data Language Daplex," ACM. Transaction on Database Systems, Vol.6, No.1, March, 1981, pp.140-173.

[29] Snodgrass, R., "The temporal query language TQUEL," In ACM TODS Vol. 12, No. 2, 1987.

[30] R. Snodgrass, Nick Kline, "Aggregate in TSQL2," The TSQL2 Language Design Committee, Mar. 21, 1994.

[31] Tansel, A. U., "Adding time dimension to relational model and extending relational algebra," Inf. Syst., Vol.11, No.4, pp.343-355, 1986.

[32] Tansel, Clifford, Gadia, Jajodia, Segev, and Snodgrass, "Temporal Databases," The Benjamin/ Cummings Publishing Company, Inc, 1993.

[33] C. I. Theodoulidis and P. Loucopoulos, "The time dimension in coneptual modeling," *Information Systems*, 16(3):273-300, 1988.

[34] Tuzhilin, A. and Clliford, J., "A temporal relational algebra as a basis for temporal relational completeness," In International Conference on VLDB, pp.13-23, 1990.

[35] G.T.J. Wuu, "SERQL: An ER query language supporting temporal data retrieval," *Proc. 10th Int'l Phoenix Conf. Computers and Comm.*, pp. 272-279, Mar. 1991.

[36] G.T.J. Wuu, and U. Dayal, "A uniform model for temporal object-oriented databases," *Proc. Int'l Conf. Data Eng.*, pp. 584-593, Tempe, Ariz, Feb. 1992.

◐ 저 자 소 개 ◑

이 홍 로

1984년 전북대학교 전기공학과 졸업(학사)
1986년 전북대학교 대학원 전자계산기 졸업(석사)
1994년 전북대학교 대학원 전산응용공학 졸업(박사)
1994년~현재 전북대학교 시간 강사
관심분야 : 객체지향 시스템, 객체지향 데이터모델링, 지리정보시스템, 시공간 데이터베이스
E-Mail : leehongro@orgio.net