객체 중심 측면 모델에 의한 KB/DB 통합 방법론

오선영* · 백두권**

A KB/DB Coupling Methodology based on the Object-Oriented Entity Aspect Model

Sun-Young Oh, Doo-Kwon Baik

요 약

기존의 데이타 모델 및 설계 방법론들은 실세계의 데이타 객체에 대해 고정된 한 측면의 모델 표현만을 허용하기 때문에 여러 측면으로 관측이 가능한 실세계 객체들의 표현에 어려움을 갖는다.

제시한 객체 중심 측면 모델(OOAM : Object-Oriented Entity Aspect Model)은 실세계의 각 객체에 대해 다수의 측면 표현을 가능하게 한 객체 중심의 데이타 모델로 데이타와 지식 표현에 유용한 모델이다.

데이타베이스 시스템과 지식베이스 시스템 중 어느 하나의 시스템이 다른 시스템의 특징을 빌리거나 통합할 수 있다면 두 시스템에게 서로 이득이 될 수 있다. 이러한 KB/DB(Knowledge Base/Data Base)의 통합은 최근에 객체 지향 개념과 연역 개념에 의해 연구가 활발히 진행되고 있다.

본 논문에서는 객체의 측면 개념을 제공하는 OOAM의 기본 개념을 보여주고 OOAM에 의해 구축되는 데이타베이스 스키마의 시맨틱을 분석하고 서술하기 위해 OOAM을 형식적으로 정의하였다. 그리고 KB/DB 통합에 관련된 연구들을 분석하고 데이타베이스에 관련된 지식의 종류를 서술한 후 OOAM을 사용하여 KB/DB 통합을 위한 지식베이스와 데이타베이스의 개발 방법론을 제시하였다.

1. 序 論

과거 2세기 동안 데이타베이스는 정보 시스템에서 주요 요소로 많은 발전을 했다. 그러나 전형적인 데이타베이스에서는 예외 처리, 질의 최적화, 변경 제약조건 등을 위한 시맨틱을 제공하는데 미약했다. 또한 데이타베이스에서는

*충남전문대학 전자계산학과
**고려대학교 전산과학과

지식 정보를 표현하는 방법과 추론, 판단 기능이 없다.

지식베이스가 데이타베이스에 없는 이러한 기능들을 제공한다. 인공 지능의 특성인 지식 표현, 탐색, 논리적인 사고, 추론 등을 사용하여 데이타베이스를 크게 향상시킬 수 있다. 특히 추상적인 지식은 지식베이스로, 사실은 데이타베이스로 구분시킴으로써 기능을 향상시킬 수 있다. 그러므로 KB와 DB의 두 시스템의 장점을 결합, 보완한 시스템이 필요하게 되고 자연스럽고 효율적인 결합방법이 요구된다.

인공지능과 데이타베이스 기술의 통합은 각 연구 분야의 초기 단계에서부터 시작되었다. 처음에는 주로 데이타/지식 표현에 대한 문제들이 연구되어져 왔고 그에 따라 의미적 데이타 모델(semantic data model), ER 모델 등이 개발되었다. 최근에는 객체 개념과 연역 개념을 도입한 통합 시스템의 개발이 활발히 진행되고 있다.

결합 방법에는 인터페이스 방법(interface approach), 내포적 방법(intensional approach), 통합방법(integrated approach) 등이 있으며 결합시 여러가지 문제점들이 발생된다. 이러한 문제점들은 두 시스템의 인터페이스에서 발생하는 임피던스 부정합으로 표현력(expressibility), 기능(functionality), 성능(performance) 면에서 문제점이 발생된다[박종태 1991]. 이를 해결하는 방법 중 하나는 객체 지향 개념을 사용하는 것이다. 즉 객체 지향 데이타 모델의 개발과 객체 지향적 방법론을 사용하여 표현의 불일치에서 발생하는 문제점을 해결할 수 있다.

객체 지향 기법의 이용과 추상적인 지식을 지식베이스로 사실을 데이타베이스로 구분시킴으로써 양쪽 모두의 유지보수와 portability를 향상시킬 수 있다[Higa 1992]. 즉 KB/DB의 통합을 위한 데이타와 지식을 함께 표현해 줄 수 있는 모델이 필요하며 KB에 DB에 나타난 지식을 표현하는 자연스럽고 효율적이며 일반적인 통합 방법론의 개발이 필요하다.

이러한 요구에 의해 오선영, 백두권에 의해 제시된 객체 중심 측면 모델(OOAM)[오선영 1992, 오선영 1994, 오선영 1994B]은 의미 데이타 모델(semantic data model)인 MAO 모델[조동영 1991]을 객체 중심 데이타 모델로 확장시킨 것으로 메소드, N-항 연관성 등 새로운 개념을 첨가하여 데이타베이스의 모델링 능력을 증가시킨 모델이다. OOAM은 실세계를 관점에 따라 여러 측면으로 모델링하므로 모델링 접근을 용이하게 하며 보다 더 융통성 있는 모델링 방법을 제공한다. 또한 OOAM은 데이타베이스 설계와 지식베이스 설계에도 유용하게 사용될 수 있으며 DB 스키마가 트리구조이므로 KB/DB 통합 시스템 설계에도 사용될 수 있다.

데이타베이스의 개념적 스키마에서 시맨틱을 취하는 것은 데이타베이스 설계 과정에서 가장 중요한 행동이다[김형주 1989, Siebes 1987]. 개념적 스키마 설계의 가장 중요한 점은 어떻게 정보들을 효율적으로 구분하여 주어진 개념적 모델로 변형시키는 가이다. 그러나 이것은 설계 단계에서 수집된 정보들이 모호하고 부정확해서 일관성있고 정확하게 이루어지지 않고 있다. 그리고 기존의 제안된 여러개의 모델들은 정형화의 부족으로 개념적 스키마의 분석을 다소 번거롭게 만든다.

데이타베이스 설계의 정형적인 접근은 데이타베이스 이론 분야에서 많이 연구되고 있

다. 관계 데이타 모델은 현재까지 광범위하게 연구됨과 동시에 많은 시제품으로 사용되어 이상적인 진보(이론에서 구현까지)를 해왔다 [김형주 1989]. 관계 데이타 모델 이론은 1970년에 처음으로 소개되어 SQL/DS와 INGRES 등에서 구현되었다. 그러나 관계 데이타베이스 스키마에 대한 이론(데이타 종속, 정규화 등)은 많은 연구가 되어 왔으나 [Ullman 1982], 객체중심데이타 모델에 대한 이론적인 접근은 김형주[1989], Weddell [1989]과 같이 소수에 불과하다.

관계 데이타베이스 모델과 달리 객체 지향 데이타베이스 모델은 이론적인 뒷받침 대신에 SMALLTALK, COMMON Objects, C++과 같은 프로그래밍 환경으로 개발되어왔다. 특히 대부분의 연구들이 객체지향 특성, 예를 들면 객체 식별, 복합 객체, 상속, 메소드 등에 대한 정형화를 프로그래밍 관점으로 보고 logic에 대해 중점을 두고 있다[Chen 1989, Masahiko 1991, Kifer 1989, Kifer 1989B, Clyde 1992].

본 논문에서는 OOAM의 기본 개념을 서술하고 OOAM의 형식론을 보여준다. 그리고 KB/DB 통합을 위한 개발 방법론을 보여준다. 먼저 2장에서는 KB/DB 통합에 관련된 연구들을 분석하고 3장에서는 OOAM의 기본적인 개념과 OOAM의 형식론을 간략하게 보여준다. 4장에서는 KB/DB 통합을 위한 개발 방법론으로 데이타베이스와 지식베이스를 개발하기 위한 단계가 자세히 설명된다. 마지막으로 5장에서는 본 논문에 대한 결론과 추후 연구과제를 제시한다.

2. 관련 연구

먼저 데이타베이스와 관련된 지식들을 구분해 보고 KB/DB 통합에 관련된 연구를 분석한다.

2.1 데이타베이스와 관련된 지식의 종류

(1) Weiderhold에 따른 지식의 종류

Weiderhold는 데이타베이스에 관련된 지식을 9개로 구분하여 정의하였다[Weiderhold 1984]. 이 구 분은 intensional(semantic)과 extensional(episodic) 지식으로 느슨하게 나누어질 수 있으며 그림 1과 같다.

extensional 지식은 조직체의 객체들과 사건들의 인스턴스들로 구성되며 보통 데이타베이스에 저장된다. intensional 지식은 데이타베이스의 구조를 서술한 것으로 지식베이스에 저장된다. 이는 데이타베이스의 사실적인 내용보다 위에 있는 지식으로 extensional 지식보다 더 추상적이다.

intensional과 extensional 지식 사이의 차이는 항상 명확하지 않다. 그러나 데이타베이스는 지식베이스 내에 포함된 extensional 지식보다 더 추상적이고 안정된 지식을 포함해야만 한다.

KB/DB 통합의 가장 자연스러운 부분은 derived와 structural 지식과 같은 intensional과 extensional 지식사이의 경계 부분이다.

(2) 뷰에 따른 지식의 종류

intensional 지식을 뷰관점에서 좀 더 세분화시켜보면 general semantic 지식과 task-

specific semantic 지식으로 나눌 수 있다 [Higa 1992]. general 지식은 데이타베이스의 전체뷰와 유사하며 task-specific 지식은 외부뷰와 유사하다.

1) general semantic knowledge

general semantic 지식은 전체 조직체의 객체들과 그들 사이의 관계에 대한 지식을 나타낸다. 이는 보안, 데이타 검색정책 등에 관련된 지식으로 확장될 수 있다. 전체 데이타베이스 스키마 구조에 대한 정보를 가지므로 스키마 evolution시 사용할 수 있다. general semantic 지식의 설계는 다음에 설명될 task-specific 지식보다 선행되어야 한다.

2) task-specific semantic knowledge

정보검색은 실제로 전체 객체들 중 몇개의 객체들에만 관련된다. 이들 객체들은 요구되는 작업을 수행하기 위해 특정 구조로 조직된 후 탐색, 추론 등이 수행된다.

INTENSIONAL(semantic) Knowledge

KB
- Enterprise Directing
  추상화의 가장 높은 단계
  예 : 여러 대안의 투자 평가
- Focus Management
  모호성을 제거하기 위한 사용자의 의도 정의
  예 : shipping DB에서 구출 임무냐 수송이냐?
- Application specific
  예 : N 객체들 사이의 N*N 거리는 DB에 저장되지 않고 계산에 의해 얻음
- General Procedure
  예 : 데이타가 이산적으로 분포되어 있으면 회귀분석보다 요인분석을 함
- structural
  데이타간의 종속성과 제약조건들
  예 : 개체무결성과 참조무결성

EXTENSIONAL(episodic) Knowledge

DB
- Derived
  예 : 여러 개의 컬럼들의 합
- Data domain
  데이타 타입
  예 : int, char, float, money, date, etc.
- 시스템
  예 : 분산시스템에서 접근패스, 인덱스최적화
- Resiliency & Recovery
  시스템 실패의 관리

그림 1. 지식의 종류

general 지식 구조가 구축된 후에 특정 작업에 대한 특수화된 지식 구조를 표현하기 위해 부구조가 형성되어야 한다. 각 특수화된 지식 구조는 지식베이스의 부분집합이다. 이렇게 새로이 형성된 부구조와 그에 관련된 탐색 기법, 추론 패턴 등이 task-specific 지식이다. task-specific 지식은 주어진 작업을 위해 general 지식으로부터 요구되는 객체들이 수집된 후 그들 사이의 관계가 설정됨으로써 구축된다. 그리고 탐색 기법과 추론 패턴이 정의된다.

2.2 KB/DB 통합

KB/DB 통합에 관련된 많은 연구가 지식 모델링에 중점을 두고 있다. 이 중 predicate logic, production rule, semantic nets, frame 등이 대표적인 예이다.

SOM(Structured Object Model)은 Higa [Higa 1989]에 의해 개발된 모델로 계층적 관계에 의해 데이타베이스 스키마를 설계한다. 이 방법은 2개의 다이어그램(초기 SOM과 최종 SOM)을 생성한다. 초기 SOM은 엔티티들

간의 논리 관계를 표현하고 최종 SOM은 논리화일 구조를 보여준다. 규칙을 사용하여 초기 SOM을 최종 SOM으로 변경한다. 스키마 설계가 복잡해질수록 더욱 용이하게 사용할 수 있다. 또한 통합된 KB/DB 설계에 유용한 모델로 DB 스키마를 표현하는 트리 구조 자체가 KB 탐색과 추론 전략을 개발하는데 용이하다. 그러나 추상화 개념이 부족하며 표현이 중복되어 부가 작업인 변환 방법이 필요하다. 또한 task- specific semantic knowledge의 표현 방법론만 언급하였다.

A-Object(다중 측면 지식 객체)[김일도 1992]는 지식을 객체화하고 상속 개념만을 고려하였으며 구체적인 모델과 방법론을 제시하지 않았다.

KDL-advisor[Higa 1992]는 Potter가 제시한 통합 시스템으로 지식 모델링에 중점을 두었다. 지식과 데이타 시멘틱을 나타내기 위해 의미적 데이타 모델(semantic data model) 상에 객체 지향 특성을 이용하여 지식베이스 개발 모델을 제시하였다. 그러나 개발 방법론에 대해서는 언급하고 있지 않다.

Ram, Hayne, Carlson[Higa 1992] 등은 계층 트리 구조를 사용하여 객체지향 지식베이스 구조를 가진 통합 시스템을 개발하였다. 그러나 특정 도메인에만 적합하고 일반적인 구현 절차를 제공하지 않았다.

위에서 살펴본 바와 같이 구조적 DB 지식을 개발하고 표현하는 일반적인 방법론이 없다. 따라서 객체 지향 전략을 사용한 KB/DB 통합 방법론 개발이 요구된다[박종태 1991, Higa 1992].

3. 객체 중심 측면 모델

OOAM은 객체 지향 개념과 계층 구조를 가지므로 데이타와 지식 표현에 유용하게 사용될 수 있다. 또한 OOAM은 여러 측면으로 관측이 가능한 실세계의 데이타 객체들을 직접 데이타베이스의 다중 측면 객체로 모델링한다. 이를 위해 OOAM의 스키마는 엔티티를 나타내는 객체 클래스와 각 엔티티를 이해하는 관점인 측면 클래스로 구성된다. 모든 엔티티는 객체로 표현되고 이러한 객체는 엔티티 타입에 해당하는 클래스에 속하며 속성과 메소드를 공유하게 된다. 객체에 대한 연산은 해당 객체의 메소드로 표현되고 연산의 실행은 이 객체에 메시지를 보냄으로써 처리된다.

3.1 클래스

OOAM에서는 클래스를 객체 클래스(object class)와 측면 클래스(aspect class)로 구분한다. 객체 클래스는 모델링 과정에서 표현하고자 하는 실세계의 물리적 대상체들이나 개념, 사건, 혹은 추상화된 사고 등을 나타낸다.

측면 클래스는 객체 클래스에 대한 한 이해 관점을 나타낸다. 한 객체 클래스는 여러 측면을 가질 수 있으며, 측면을 통해 다른 여러 객체 클래스들과 연계된다. 즉, 객체 클래스들은 측면개념을 통해 상호간의 암시적, 혹은 명시적 관계성을 갖는다. 전통적인 객체 중심 시스템에서 측면은 추상화된 관계성으로 볼 수 있다. OOAM에서 측면 클래스는 P-타입, G-타입, A-타입의 세가지로 나누어진다. OOAM의 기본 개념은 그림 2와 같다.

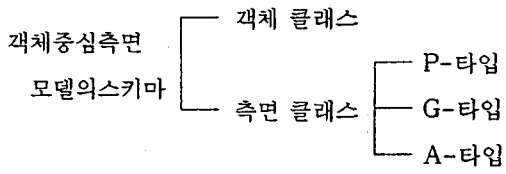

그림 2. 객체중심측면모델의 기본개념

OOAM의 스키마는 OOAM 다이어그램을 이용하여 그림으로 표현할 수 있다[오선영 1992]. OOAM 다이어그램은 객체 클래스을 나타내는 객체 클래스 노드와 객체 클래스 노드에 대한 이해 관점을 나타내는 측면 클래스, 객체 클래스 노드와 측면 클래스들을 연결하는 링크로 구성되는 그래픽 구조의 다이어그램이다.

클래스에는 유사한 객체들의 공통되는 성질, 즉 이들의 스키마 정보와 이 객체들에 적용 가능한 연산에 대한 정보가 포함되며 인스턴스는 해당 클래스의 스키마 정보에 따라 실제값을 갖는 객체이다.

3.2 관계성

측면 클래스는 객체 클래스에 대한 한 이해 관점을 나타낸다. 즉, 객체 클래스들은 측면 클래스를 통해 상호간의 암시적 혹은 명시적 관계성을 갖는다. OOAM에서 지원하는 추상화 관계성은 측면 클래스로 IS_A 관계를 나타내는 G-타입과 PART_OF 관계를 나타내는 P-타입, 연관성 관계를 나타내는 A-타입이 있다.

(1) G-타입 측면

G-타입 측면은 객체 클래스들의 유사한 성질들을 공유하기 위한 추상화로 일반 객체 클래스와 더욱 세분된 클래스 사이의 IS-A 관계성을 나타낸다. IS-A 관계의 하위 클래스는 상위 클래스들로부터 특성을 계승받는다[Kim 1990].

G-타입 측면은 2가지의 종류로 분류될 수 있는데[Banerjee 1987] 하나는 배타적 일반화(disjoint generalization)로 하위 객체 클래스들이 상위 객체 클래스의 배타적 부분집합으로 이루어진 것이다. 즉, 객체타입의 객체들이 다수의 상호배재적인 객체타입들로 분할되어 이해될 수 있다[Rumbaugh 1991]. 다른 하나는 하위객체 클래스들이 상위 객체 클래스들의 중첩된 부분 집합으로 이루어진 중첩 일반화(overlapping generalization)로서 주어진 객체타입이 여러 중복 가능한 객체타입들로 나누어질 수 있음을 나타낸다.

그림 3은 객체타입 'student'가 G-타입 측면인 'graduate'에 의해 객체타입 'gradstudent'로 특수화된 예를 보여준다.

G-타입 측면에서는 존재종속성[Banerjee 1987, Kim 1990]이 존재한다.

(2) P-타입 측면

P-타입 측면은 상위의 객체 클래스들을 하위의 객체 클래스들의 모임으로 나타내는 추상화로 PART-WHOLE 또는 A-PART-OF의 의미를 가진다[Banerjee 1987]. 즉, P-타입 측면은 주어진 객체타입이 다수의 상호배제적인 객체 타입들의 합성에 의해 인식될 수 있음을 나타내는 것이다.

P-타입 측면에 의해 주어진 객체 타입과 관련되는 각 하위 객체 타입을 주어진 객체에 대한 성분 객체(component object)라고 한다. 반대로 성분객체 타입을 합성한 객체타입을 복합객체(composite object)라고 한다.

P-타입 측면은 existence 속성과 exclusive 속성에 의해 특정지워질 수 있다. 그림 3에서 성분객체 'arms', 'legs', 'body'는 복합객체 'student'를 구성한다.

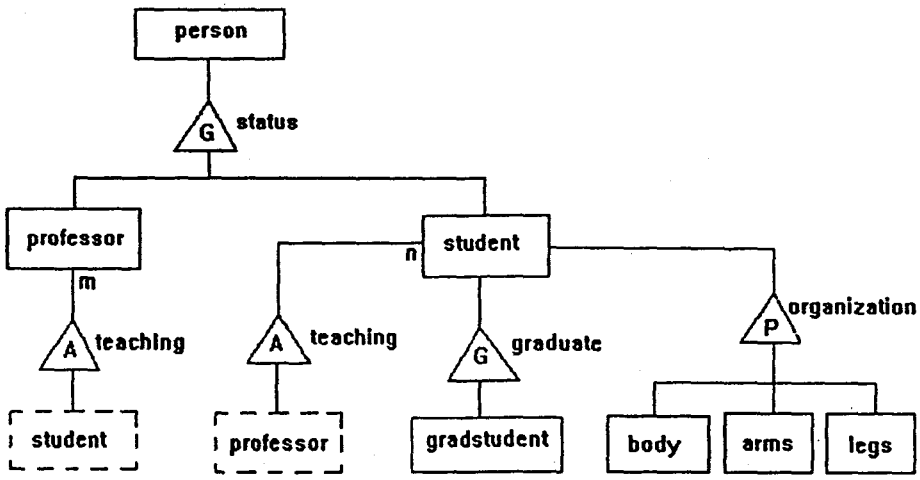

그림 3. 객체중심측면 모델의 예

(3) A-타입 측면

A-타입 측면은 주어진 객체타입의 객체들이 다른 객체타입의 객체들과의 연관성(association)에 의해 인식되는 측면이다[Rumbaugh 1991]. MAO 모델에서는 R-타입 측면과 연관성 타입에 의해 연관성 추상화 개념을 표현하였으나 OOAM에서는 A-타입 측면 하나로 표현한다.

실세계의 엔티티 사이에는 많은 종류의 연관성이 존재하며 일반적으로 연관성은 참여하는 엔티티의 수에 따라 이항연관성(binary associatioin)과 N항 연관성(N-ary association)으로 분류되고, 엔티티간의 함수성에 따라 일대일, 일대다, 다대다 등으로 구분된다[Smith 1991, Rumbaugh 1991]. 기존의 객체 중심 데이타베이스 시스템에서는 연관성을 직접적으로 표현할 수 있는 구문과 시맨틱을 제공하지 못하고 있다. 즉, 기존 방법은 특정 연관성을 특별한 방법으로 표현하거나 또는 연관성의 서로 다른 면을 특정 클래스의 메소드 또는 속성들로 구현하고 있다[Banerjee 1987, Smith 1991].

OOAM에서는 연관성을 클래스로 표현한다. 연관성에서 관련된 객체들은 OOAM 다이어그램에 N-항 관계에 의해 N번 나타난다. 이렇게 하므로써 OOAM 스키마는 계층구조를 갖게 된다. 그림 3에서 객체 타입 'professor'와 'student'은 각각 A-타입 측면을 갖고 있다.

A-타입에 의해 스키마는 비계층적으로 구성되어 다이어그램 표현이 복잡하게 된다.

3.3 객체 중심 측면 모델의 형식론

형식론은 설계과정에서 필요한 공리를 정의하고 데이타베이스의 intension과 extension을 구분하여 정의한 후 intesion과 extension 사이의 관계를 서술하였다[오선영 1994]. 이 절에서는 공리에 의해 얻을 수 있는 데이타베이

스의 설계과정을 간략하게 설명하고 OOAM의 intension과 extension 정의에 의해 유도되는 규칙들을 서술한다. 형식론에 대한 자세한 정의는 참고논문[오선영 1994B]에 있다.

OOAM에 근거한 공리들은 속성, 클래스, 클래스타입, 관계성, 측면 등이다. 유도된 공리들은 데이타베이스 설계 과정에서 다음과 같은 데이타베이스의 간략한 기술을 얻는데 사용될 수 있다.

① 설계하고자 하는 데이타베이스에서 발견할 수 있는 모든 클래스 타입들을 선정한다. 클래스 공리에 따라 두개의 클래스 타입은 동일한 property를 가질 수 없다. 만약에 두개의 클래스 타입이 동일한 것으로 간주된다면 다중 역할로 간주하여 역할에 대한 property를 첨가한다. 즉 각 클래스 타입들은 다른 클래스 타입에 대해 독립적으로 인식되는 모델링 대상의 주요 개념, 사물 등이 된다. 클래스 공리를 만족하도록 클래스 타입의 개념들은 상호 배제적이어야 한다.

② 각 클래스 타입에 대한 property 이름 집합, 원자 값 집합, 그에 따른 속성들을 선정한다. 속성 공리에 따라 각 속성에 대한 원자 값 집합이 모호하지 않도록 한다.

③ 클래스 타입에 대해 측면을 설계한다. 이는 측면 공리에 따라 클래스 타입에 대해서만 정의되며 한 클래스 타입에 대해 여러개의 측면이 존재할 수 있다. 측면이 속성을 가질 때 이 속성들은 측면에 대한 클래스에 표기된다.

④ 결과는 데이타베이스의 개념적 스키마이다.

설계 과정에서의 공리를 기반으로 하여 데이타베이스의 intension, 즉 OOAM의 데이타베이스 스키마의 정의는 아래와 같다.

정의 다음과 같은 형식의 클래스 스킴들의 집합을 데이타베이스 스키마 S라 한다.

$\{C_1, C_2, C_3, ..., C_n\}$ KOA $\{C_1', C_2', C_3', ..., C_m'\}$
$C_n\{P_1, P_2, P_3, ..., P_i\}$
$C_m'\{P_1', P_2', P_3', ..., P_j'\}$
$KOA\{P_1'', P_2'', P_3'', ..., P_k''\}$
여기서, $i,j,k \in N$,
$m,n \in N$, C_n과 C_m' 는 클래스 이름,
$i,j \in N$, P_i와 P_j'는 클래스 C_n과 C_m'이 가지고 있는 속성,
$k \in N$, P_k''는 측면이 가지고 있는 속성

예. 그림 3에 대한 데이타베이스 스키마 S는 아래와 같다.

{person} status_typeG {professor, student}
{student} graduate_typeG {gradstudent}
{professor} teaching_typeA {student}
{student} teaching_typeA {professor}
{arms, legs, body} organization_typeP {student}
person{snumber}
student{id, department}
professor{major}
gradstudent{term}
arms{}
legs{}
body{}
teaching_typeA{course, room}
organization_typeP{}
status_typeG{}
gradstudent_typeG{}

클래스 타입에 대한 데이타베이스 extension의 정의는 다음과 같다.

정의 스키마 S에 속해 있는 각 클래스 타입 C를 $\pi_C(I)$로 프로젝션 시키는 함수를 E_C라 하고 다음과 같이 표기한다.

E_C : Classes(S)→P(D_C)

데이타베이스의 extensions은 위의 정의에 의한 각 클래스 타입의 extension의 집합이다.

위의 데이타베이스 intension과 extension에 의해 유도되는 규칙들은 아래와 같다.

규칙 1. C(p) & typeG(B,C) ⇒ B(p)
이는 G-측면에서 나타나는 property 상속에 대한 메카니즘이다.

규칙 2. typeG(A,B) & typeG(B,A)
⇔ A=B

규칙 3. typeG(A,B) & typeG(B,C)
⇒ typeG(A,C)

규칙 4. typeG(A,B) & typeG(A,C)
⇒ ∃D(typeG(B,D) & typeG(C,D))

규칙 5. typeP(A,B) ⇒ ~typeP(B,A)

규칙 6. typeP(A,B) & typeP(B,C)
⇒ typeP(A,C)

규칙 7. typeG(A,B) & typeP(B,C)
⇒ typeP(A,C)
규칙7은 항상 성립하나 그의 반대 즉,
typeP(A,B) & typeG(B,C)
⇒ typeP(A,C)는 항상 참이 아니다.

규칙 8. typeG(A,B) & typeP(A,C)
⇒ typeP(B,C)

규칙 9. typeG(B,C) & typeP(A,C)
⇒ typeP(A,B)

규칙 10. typeG(A,B) & typeA(A,C)
⇒ typeA(B,C)

규칙 11. instance(e,C) & typeG(C,D)
⇒ instance(d,D)
인스턴스 d는 인스턴스 e의 부분 집합이다.

규칙 12. C(p,x,y) & typeG(B,C)
⇒ B(p,x,y)
이는 property값이 G-측면에서 상속됨을 의미한다.

제시한 규칙들은 클래스들간의 관계에 의해 명확한 시맨틱을 나타낸다. 이 시맨틱은 실세계를 정확하게 모델링할 수 있는 기본이 되고 지식베이스에 구축되어 질의처리에 도움을 줄 수 있다.

4. KB/DB 통합 방법론

객체 중심 측면 모델(OOAM)은 객체 지향 표현과 계층 구조를 제공하므로 이를 사용하여 데이타와 지식 표현을 자연스럽게 할 수 있다. 또한 OOAM은 데이타베이스의 개념적 설계, 지식베이스 설계, KB/DB 통합 시스템 설계에도 사용될 수 있다.

OOAM을 이용하여 통합된 KB/DB 개발을 하는 전체적인 방법은 먼저 데이타베이스의 개념스키마와 목표 시스템에 적합한 논리 스키마를 개발한다. 다음으로 데이타베이스의 설계 방법을 사용하여 지식 집합, 탐색 패스, 추론 패턴 등을 구별하여 객체 관계를 정의하고 이를 지식베이스에 나타낸다.

본 장에서는 OOAM을 이용하여 데이타베이스를 개발하는 단계를 서술한 다음 지식베이스 설계 단계를 서술한다.

4.1 데이타베이스 개발

OOAM을 사용하여 데이타베이스의 개념적 스키마를 모델링[오선영 1992]하고 이를 목표 시스템에 구현하기 위한 설계단계를 설명한다. 예는 대학의 학사업무를 위한 데이타베이스로 간략하게 보인다.

(1) 개념적 스키마의 개발

단계 1. 가장 일반적인 객체(시스템 엔티티)[조동영 1991]를 정의하고 그와 관련된 모든 클래스들과 속성들을 정의한다.

가장 일반적인 객체는 모델링의 시작을 위해 도입한 특수한 타입으로 속성을 가지지 않으며 측면에 의해 의미를 갖게 된다. 이 단계에서 설정되는 객체들은 다른 어떠한 객체 타입들에 대해서도 독립적으로 인식되는 모델링 대상의 주요 개념, 사물이 된다.

예에서 가장 일반적인 객체는 UNIVERSITY가 되고 이와 관련된 클래스들과 속성들은 아래와 같다.

① PERSON
UNIVERSITY의 인적 객체를 나타낸다. 속성은 name이 있다.

② PROJECT
연구 프로젝트를 표현한 것으로 속성들은 id, title이 있다.

③ STUDENT
학생 객체를 나타내며 number 속성을 가진다.

④ EMPLOYEE
대학내의 교직원들로 속성들은 ssn과 salary가 있다.

⑤ STAFF
교직원중 사무직원들을 나타내며 속성은 status가 있다.

⑥ PROFESSOR
교수들을 표현하며 속성은 tel, address가 있다.

⑦ COURSE
교과과정의 과목들로 cname, day, hour, room 속성들을 가진다.

⑧ DEPARTMENT
대학내의 모든 과들을 표현하며 dname 속성을 가진다.

⑨ AREA
대학내에서 연구되는 분야로 속성은 aname, description 등이 있다.

⑩ STRATEGY
프로젝트에서의 다양한 기법들을 나타내며 속성은 sname이 있다.

⑪ TOOL
프로젝트에서 사용되는 H/W와 S/W 연구도구들을 가르키며 tname, keyword 속성을 가진다.

⑫ HEAD
교직원중 어떤 부서의 장들을 표현한다.

단계 2. 측면을 사용하여 클래스들간의 관계를 정의한다.

단계 1에서 정의한 객체들에 대해 먼저 G-타입, P-타입 측면 관계를 설정한다. 다음으로 A-타입 측면 관계를 설정한다. A-타입 측면에서 3-항 관계성을 나타낼 수 있다.

보통 A-타입 측면은 G-타입, P-타입 측면과 달리 측면클래스 내에 속성들을 가지므로 정확히 속성들을 표시한다. A-타입에 의해 구축된 개념적 스키마는 네트워크 형태가 된다. 네트워크 형태의 모델은 시맨틱 관계를 표현하

기는 용이하나 구조적 지식들을 나타내거나 추론 패턴등을 인식하기에는 적당하지 않다. 이를 해결하기 위해 A-타입 관계에 의해 연결된 객체 클래스는 만약 클래스들이 서로 다른 부구조에 존재하면 OOAM 스키마에 두번 나타나게 한다. 이렇게 하므로써 OOAM 스키마는 계층구조인 트리형태를 유지하게 된다.

보통 위의 단계 1과 단계 2는 개별적인 것이 아니라 서로 반복적으로 수행되며 완전한 스키마가 될 때까지 반복한다. 단계 1과 단계 2의 반복 수행 결과는 그림 4와 같다.

① UNIVERSITY는 P-타입 측면 HAS에 의해 PERSON과 PROJECT 클래스를 가진다.

② PERSON은 G-타입 측면 STATUS에 의해 STUDENT와 EMPLOYEE로 세분화된다.

③ EMPLOYEE는 G-타입 측면 POSITION에 의해 STAFF와 PROFESSOR로 세분화된다.

④ STAFF와 PROFEESOR 클래스는 G-타입 측면 BOSS에 의해 HEAD로 세분화된다. HEAD 클래스는 양쪽 클래스들로부터 속성들을 상속받는다(다중상속).

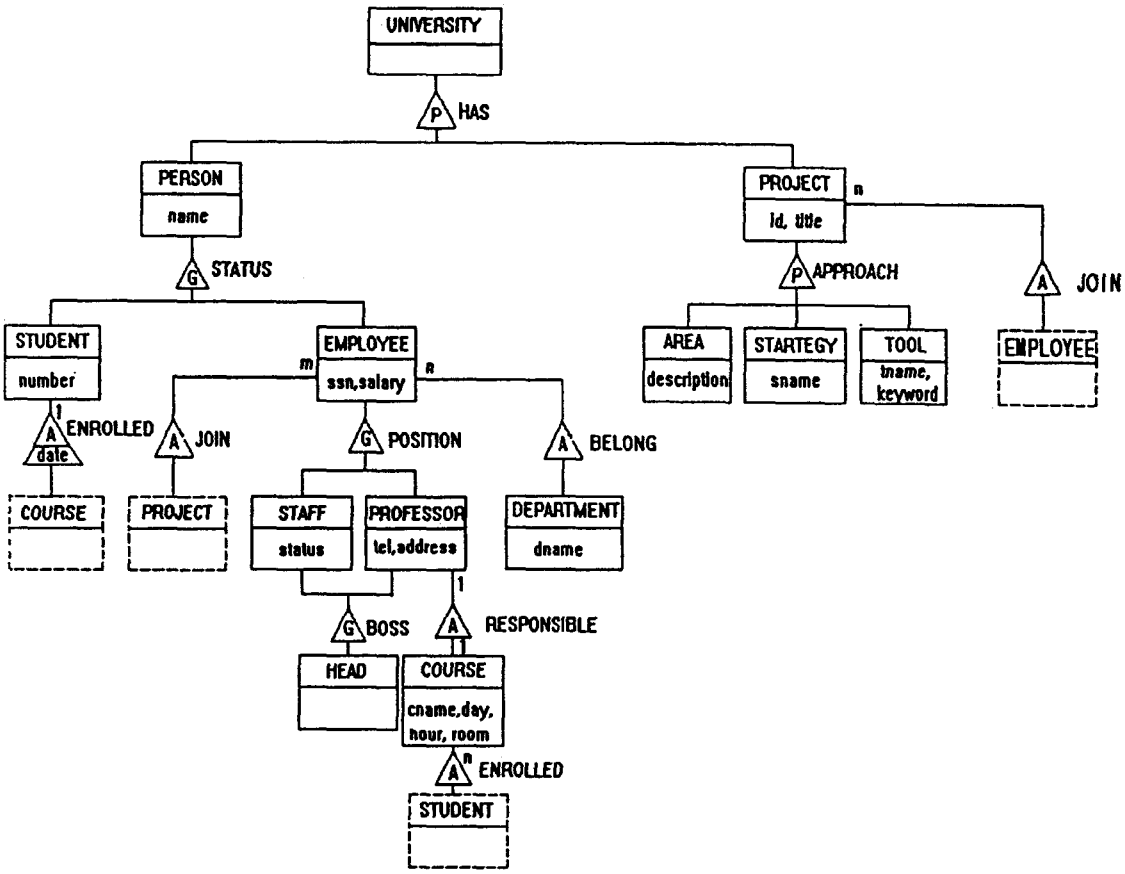

그림 4. 대학 학사 업무에 대한 예

⑤ PROJECT는 P-타입 측면 APPROACH에 의해 AREA, STRATEGY, TOOL 클래스들을 가진다.

⑥ PROJECT는 A-타입 측면 JOIN에 의해 EMPLOYEE 클래스와 연관된다.

⑦ EMPLOYEE는 A-타입 측면 BELONG에 의해 DEPARTMENT 클래스와 연관된다.

⑧ PROFESSOR는 A-타입 측면 RESPONSIBLE에 의해 COURSE 클래스와 연관된다.

⑨ STUDENT는 A-타입 측면 ENROLLED에 의해 COURSE 클래스와 연관된다.

단계 3. 관계성이 완전한가를 검사한다.

측면 타입에 의해 모든 클래스가 완전히 관계성을 갖는지를 검사한다. 특히 P-타입과 G-타입 측면에 대한 existence와 exclusive 값을 명세하고 A-타입에 연결된 각 객체 클래스의 대응수(cardinality)를 정확히 결정한다. A-타입에 연결된 객체 클래스들이 스키마 트리의 서로 다른 부구조에 놓여 있으면 스키마에 두번 나타나 있나를 검사한다.

(2) 논리적 스키마의 개발

목표 시스템에 맞게 앞에서 개발된 개념적 스키마를 논리적 스키마로 변경한다. OOAM의 객체 특성을 살리기 위해 목표 시스템은 객체 지향 데이타베이스 시스템인 VERSANT로 이를 이용하여 데이타베이스를 구축한다. 그러나 VERSANT는 is-A 관계에 의해 스키마가 구축되므로 OOAM의 측면들을 직접적으로 스키마에 표현할 수 없다. 이를 위해 측면을 VERANT 스키마로 변경하기 위한 규칙들을 개발한다.

① 가장 일반적인 객체(시스템 엔티티)

가장 일반적인 객체는 모델링의 시작을 위해 도입된 특수한 객체로 속성들을 가지지 않는다. 이 객체는 VERSANT 스키마의 최상위 클래스인 PObject로 대치된다.

② 객체 클래스

OOAM의 객체 클래스는 VERSANT의 객체 클래스로 변환된다.

③ 속성

가. 단일값 속성

단일값 속성은 객체 클래스의 단순 속성으로 변환된다.

나. 다중값 속성

다중값 속성에서 값들의 순서가 중요하면 List 데이타 타입으로 변환하고 순서가 중요하지 않으면 Set 데이타 타입으로 변환한다. List와 Set은 VERSANT가 제공하는 collection 클래스들이다.

VVSet⟨element-type⟩ attribute
VVList⟨element-type⟩ attribute

다. 1:1 관계에서의 속성

연결된 두 개의 클래스 중 보다 더 확고한 클래스의 속성으로 첨가한다. 보다 더 확고한 클래스란 영구적인 클래스를 의미한다.

라. 1:n 관계에서의 속성

이 속성은 대응수가 N인 클래스의 속성으로 첨가한다.

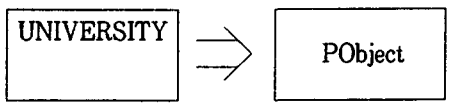

그림 5. 시스템 엔티티의 변환

마. m:n 관계에서의 속성

이 속성은 새로 생성된 링크 객체 클래스의 속성으로 첨가한다(⑥-다-ⓒ).

④ G-타입 측면

G-타입 측면에서는 존재 종속성이 존재하므로 VERSANT의 is-A 관계로 직접 변경할 수 있다. 그러나 속성 exclusive에 의해 구분되는 배타적 일반화와 중첩 일반화에 대해서는 서로 다른 변경 방법이 필요하다.

가. 배타적 일반화

배타적 일반화는 하위 객체 클래스들이 상위 객체 클래스의 배타적 부분집합으로 이루어진 것이다. 이는 수정없이 VERSANT의 is-A 관계로 직접 표현된다. 예에서 클래스 PERSON은 클래스 STUDENT와 EMPLOYEE에 대해 배타적 일반화로 VERSANT 스키마로 변경없이 표현될 수 있다.

OOAM 다이어그램에서 G-타입 측면 클래스 노드를 삭제한다.

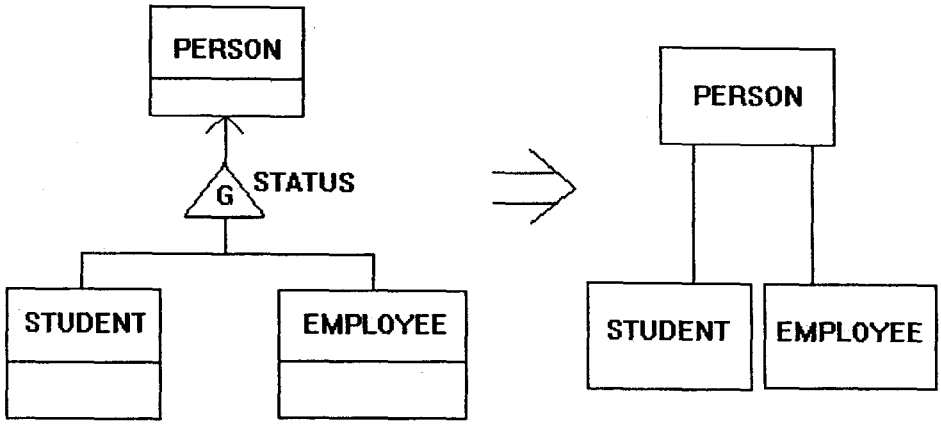

그림 6. 배타적 일반화의 변환

나. 중첩 일반화

중첩 일반화는 하위 객체 클래스들이 상위 객체 클래스의 중첩된 부분 집합으로 이루어진 것이다. 즉 주어진 객체 타입이 여러 중복 가능한 객체 타입들로 나누어지므로 VERSANT의 is-A관계로 직접 표현할 수 없다.

중첩 일반화는 하위 객체 클래스들이 상위 객체 클래스에 G-타입 측면 이름을 갖는 속성으로 변형된다. 이 속성은 Set 클래스(VERSANT에서 제공하는 collection 클래스들 중의 하나) 타입으로 하위 객체 클래스들의 객체를 값으로 가진다.

VVSet<subclass> attribute

OOAM 다이어그램에서 G-타입 측면 이름을 상위 클래스의 속성으로 표시하고 G-타입 측면 클래스 노드를 삭제한다.

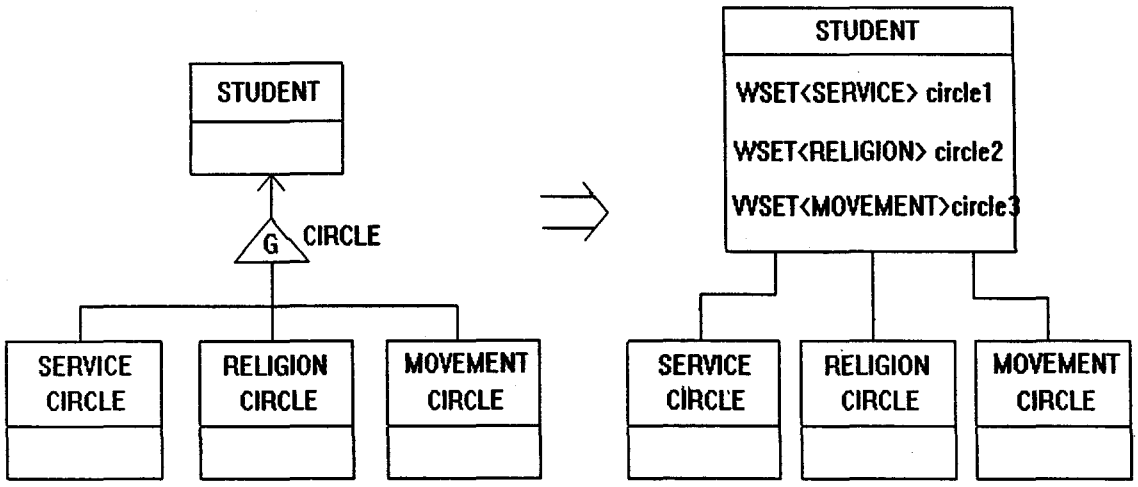

그림 7. 중첩 일반화의 변환

⑤ P-타입 측면

P-타입 측면을 변경시키기 위해서는 성분 객체와 복합 객체를 서로 분리시켜야 한다. 복합 객체는 스키마의 그 위치에 두고 성분 객체들은 VERSANT의 최상위의 클래스인 PObject에 연결시킨다. 그리고 복합 객체에 성분 객체들과 연결시키기 위한 링크 속성들을 만들어 준다.

복합 객체는 각 성분 객체와의 함수 관계에 의해 1:1 또는 1:n의 관계를 갖는다. 그러므로 링크 속성을 만들어줄 때 서로 다른 방법을 사용해야 한다.

가. 1:1 관계성분 객체를 VERSANT가 제공하는 Link 클래스 타입으로 변경시킨 후 복합 객체의 속성으로 첨가한다.

Link〈component〉 attribute

나. 1:n 관계

성분 객체를 VERSANT가 제공하는 LinkVstr 클래스 타입으로 변경시킨 후 복합 객체의 속성으로 첨가한다.

LinkVstr〈component〉 attribute

exclusive와 existence 속성은 스키마 변환에 어떠한 영향을 주지 못하며 데이타 입력시에 규칙으로써 작용한다.

OOAM 다이어그램에서 P-타입 측면 노드를 삭제하고 성분객체를 VERSANT의 최상위 클래스와 연결시킨다. 그리고 복합 객체에 각 성분 객체에 대한 링크 속성을 첨가한다.

⑥ A-타입 측면

A-타입 측면은 참여하는 클래스의 수에 따라 이항 연관성과 3-항 연관성으로 분류되고 클래스의 객체간의 다중성에 따라 1:1, 1:n, m:n 등으로 구분된다. 각각의 종류에 따라 VERSANT 스키마로 변경하기 위한 방법은 다르다.

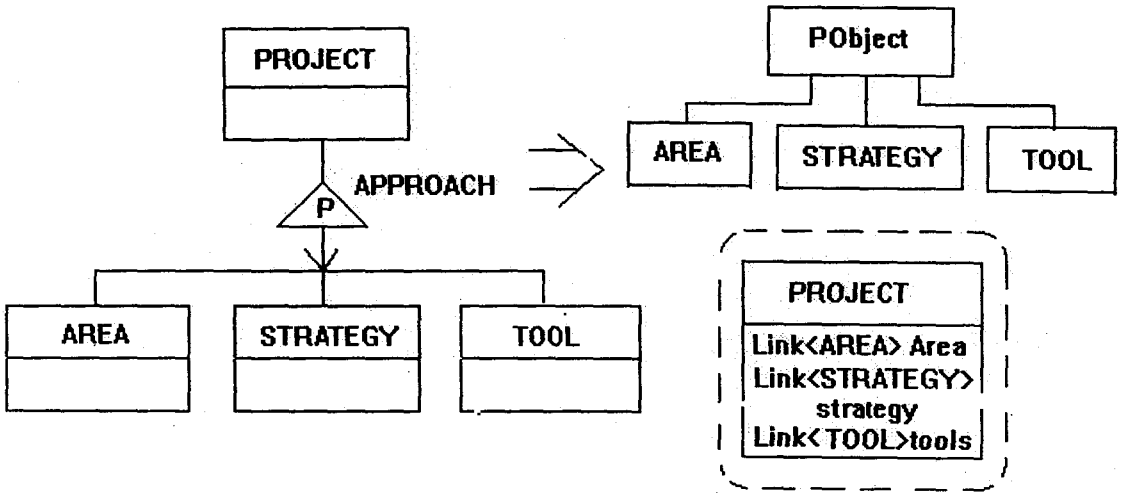

그림 8. P-타입 측면의 변환

가. 클래스 위치에 따른 클래스 이동

A-타입에 참여하는 클래스가 이미 G-타입이나 P-타입에 의해 유도된 클래스이면 그 스키마의 위치에 두나 독립적으로 참여하는 클래스이면 VERSANT의 최상위 클래스인 PObject에 연결시킨다.

나. 참여 클래스에 따른 변환

㉠ 이항 연관성

VERSANT에서 제공하는 양방향 링크 클래스 타입을 사용하여 각 클래스에 다른 클래스에 대한 링크를 속성으로 나타낸다.

㉡ 3-항 연관성

3-항 연관성의 측면 클래스를 VERSANT의 객체 클래스로 만들어준다. 생성된 클래스를 링크 객체 클래스라 하며 여기에 각 클래스와 연결시키기 위한 링크 속성을 첨가한다. 그리고 측면 클래스 노드가 가지는 속성들을 첨가한다.

다. 다중성에 따른 변환

다중성에 따라 연관성은 1:1, 1:n, m:n 등으로 구분되는데 각각의 변환방법이 다르다.

㉠ 1:1 관계

각 클래스에 다른 클래스에 대한 양방향 링크 속성을 준다.

BiLink〈type, a_attribute〉 b_attribute

BiLink〈type, b_attribute〉 a_attribute

㉡ 1:n 관계

대응수가 1인 클래스에는 다중 양방향 링크 속성을 주고 대응수가 n인 클래스에는 양방향 링크 속성을 준다.

- 대응수가 1인 클래스 : BiLinkVstr〈type, a_attribute〉 b_attribute
- 대응수가 n인 클래스 : BiLink〈type, b_attribute〉 a_attribute

㉢ m:n 관계

새로운 링크 객체 클래스를 생성한다. 그리고 이 링크 객체 클래스에 각 객체 클래스와 연결시키기 위한 양방향 링크 속성을 첨가한다.

BiLink〈type, attribute〉 c_attribute

또한 각 객체 클래스들은 다중성에 따라 링크 객체 클래스에 대한 다중 양방향 링크 속성 또는 양방향 링크 속성을 가진다.

BiLinkVstr〈type, c_attribute〉 attribute

BiLink〈type, c_attribute〉 attribute

라. A-타입이 가지는 속성

A-타입은 P-타입, G-타입과 달리 속성들을 가질 수 있다. 이 속성들의 변환 방법은 앞의 ③과 같다.

A-타입 측면은 위의 나,다,라의 종류에 의해 결정되므로 VERSANT 스키마로 변경할 때에도 위의 변경 방법을 조합하여 사용한다.

OOAM 다이어그램에서 A-타입 측면 노드를 삭제하고 독립적으로 참여하는 클래스는 VERSANT의 최상위 클래스 PObject에 연결시킨다. 새로 생성된 링크 객체 클래스들도 PObject에 연결시킨다.

그리고 위의 방법들을 조합하여 각 클래스에 속성과 링크속성들을 첨가한다.

위의 규칙들을 적용하여 그림 4의 OOAM 스키마는 그림 10의 VERSANT 스키마로 변경된다.

예의 데이타베이스는 일관성이 있으므로 다음 절에서 개발될 지식 베이스와 자연스러운 결합을 할 수 있게 된다.

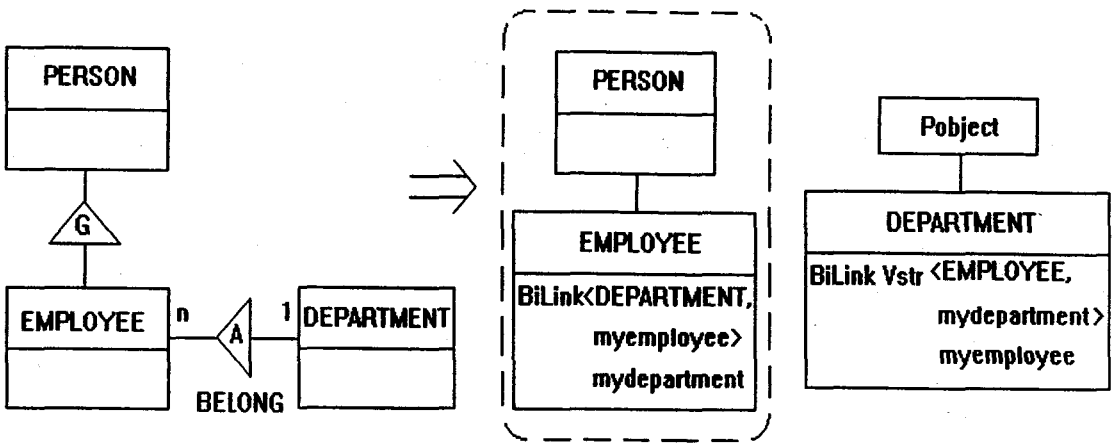

그림 9. A-타입 측면의 변환

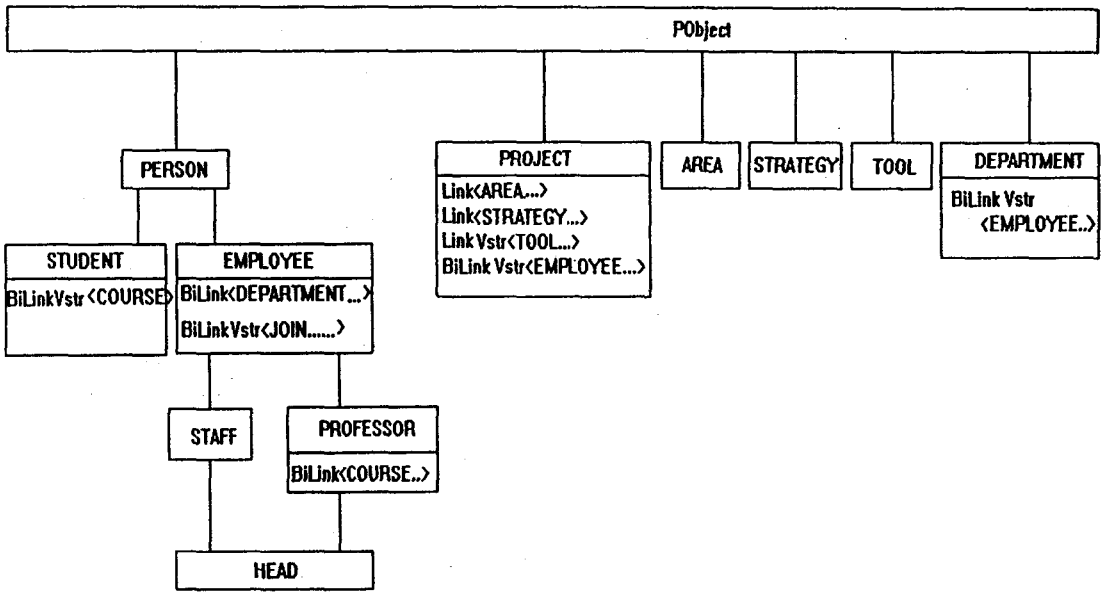

그림 10. 예의 VERSANT 스키마

4.2 지식베이스 개발

2.1에서 살펴본 바와 같이 일반적으로 지식의 종류는 크게 extensional과 intensional로 나눈다. DB 측면에서 볼 때 extensional 지식은 조직체의 객체들과 사건들의 인스턴스들로 보통 DB에 저장되어 episodic 지식이라고도 한다.

intensional 지식은 semantic 지식으로서 데이타베이스의 구조를 나타낸다. 이는 general 지식과 task-specific 지식으로 더욱 세분화될 수 있다.

general semantic 지식은 데이타베이스의 개념 스키마를 나타내며 보안 측정, 자료 검색/갱신 정책 등에 관련된 지식으로 확장될 수 있다. task-specific 지식은 데이타베이스의 외부 스키마에 관련된 지식을 말한다. 그러므로 general semantic 지식은 task-specific 지식보다 먼저 수행되어야 한다. 즉 task-specific 지식은 사용자에게 뷰를 제공하기 위한 관련된 지식이라 볼 수 있다.

현재 관계 데이타베이스 시스템에서는 뷰 개

념을 지원한다. 그러나 객체 중심 데이타베이스 시스템에서는 뷰 개념과 유사한 개념을 제공하는 시스템은 몇개 있으나 완전한 뷰 기능을 제공하고 있는 객체 중심 시스템은 없다[안상현 1990]. 따라서 task-specific 지식에 대한 개발 방법은 추후 연구 과제로 설정하고 본 논문에서는 general 시맨틱 지식에 대해서만 개발 방법론을 서술한다.

먼저 객체 지향 원리를 이용하여 OOAM에 의해 개발된 데이타베이스의 스키마를 표현한다. 그리고 객체들에 대한 탐색 방법과 전략들을 정의한다. 자세한 general 시맨틱 지식을 개발하기 위한 설계 단계는 아래와 같다.

단계 1. 객체 지향 원리를 사용한 데이타베이스 구조의 표현 방법을 개발한다.

OOAM 다이어그램에서의 각 객체 클래스와 측면 클래스를 그림 11과 같이 지식 객체로 표현한다. 객체 지향 원리를 이용하므로 지식 객체는 클래스 이름, 속성들을 가지고 객체 클래스 지식 객체에서는 이 클래스와 연결된 측면들을 aspects에서 나타낸다.

후에 다른 클래스의 속성들을 탐색할 때 용이하게 하기 위해 각 지식 객체에서는 super에 그 클래스와 연결된 상위 클래스를 나타내준다.

```
(c-object class-name
  (super class-name)
  (aspects (list aspect-name))
  (attributes (list attr-name)) )
(a-object aspect-name
  (super class-name)
  (type KOA)
  (attributes (list attr-name)) )
```

그림 11. 지식베이스 객체(OOAM)

OOAM의 메타 모델은 데이타베이스의 구조를 이해하는데 많은 도움을 준다. 그림 12는 OOAM의 메타모델을 ER 다이어그램으로 표현한 것으로 그에 대한 의미는 아래와 같다.

- CLASS는 ATTRIBUTE를 여러개 가질 수 있다.
- ASPECT는 ATTRIBUTE를 여러개 가질 수 있다.
- CLASS는 ASPECT를 가지고 ASPECT는 CLASS를 가진다. 이와같이 재귀적으로 반복되어 CLASS는 ASPECT를 통해 여러개의 관계성을 가질 수 있다.

CLASS와 ASPECT가 m:n의 관계이므로 한 CLASS는 여러개의 ASPECT를 가지며 한 ASPECT는 여러개의 CLASS를 가져 다중 상속(multiple inheritance)을 제공한다.

단계 2. 탐색 방법과 전략을 정의한다.

정보 검색 작업은 여러개의 객체와 그들 사이의 관계에 의해 수행된다. 이를 위해 객체들은 탐색 또는 추론을 위한 구조를 형성하게 된다. 단계 1에서 구축된 데이타베이스의 구조적인 지식을 이용하여 주어진 입력에 대해 타당한 출력이 나올 수 있도록 탐색 기법과 추론 방법을 정의해 주어야 한다.

OOAM에 의해 구축된 데이타베이스는 VERSANT 스키마로 변경되므로 OOAM에서의 탐색 방법과 전략을 정의하려면 VERSANT 스키마를 이해해야 한다.

VERSANT 스키마에 대한 메타모델은 그림 13과 같다.

이 모델은 다음과 같은 의미를 지닌다.

- CLASS는 ATTRIBUTE를 가진다.
- PROPERTY는 DOMAIN을 가진다.

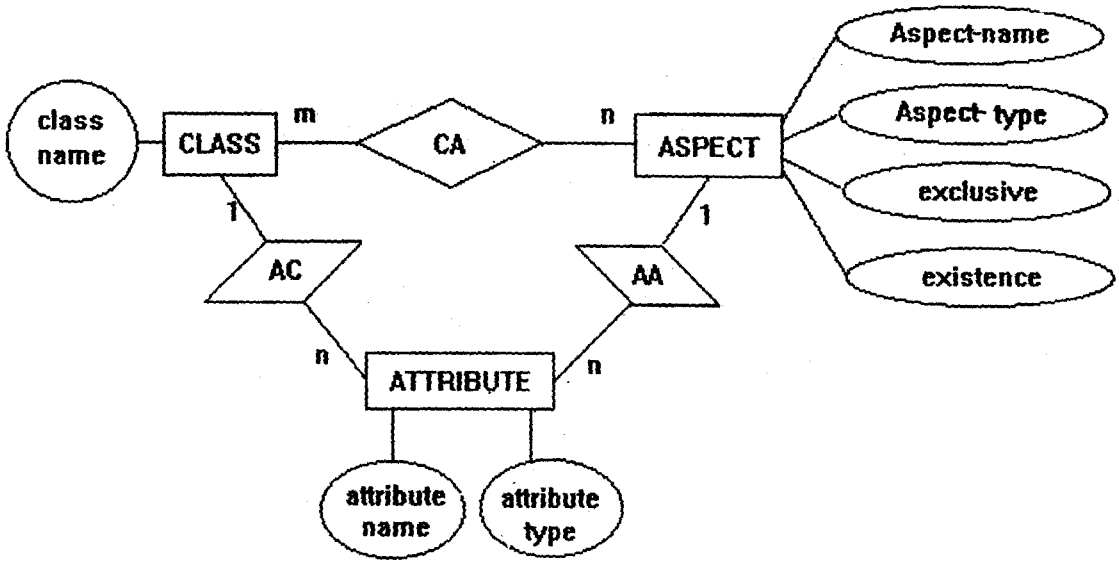

그림 12. VERSANT의 메타모델

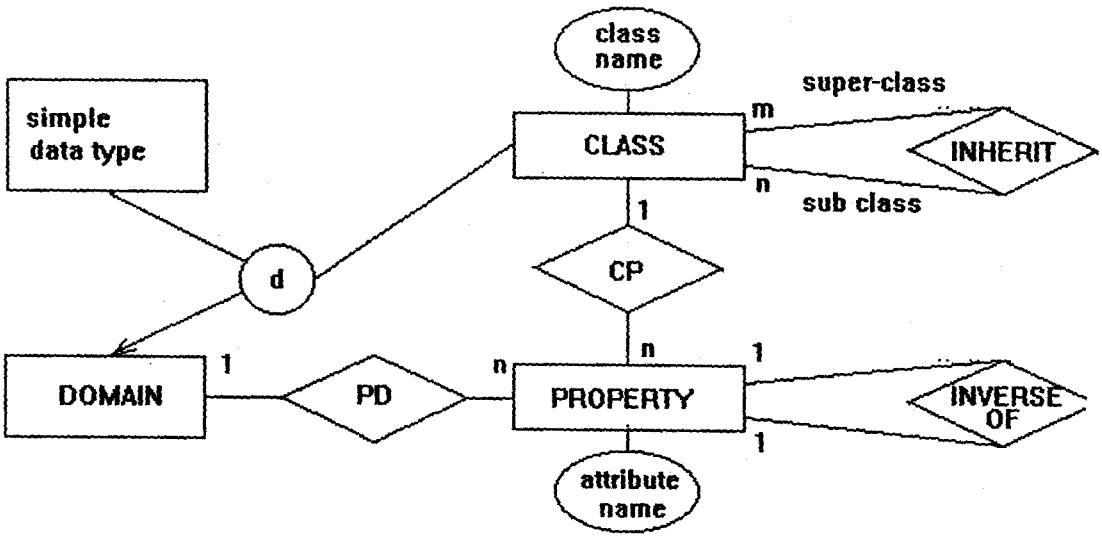

그림 13. OOAM의 메타모델

- DOMAIN은 단순 데이타 타입이거나 CLASS 타입이다.
- 일반화 관계는 INHERIT 관계에 의해 다루어진다.
  SUPER-CLASS가 m이므로 다중 상속을 제공한다.
- 집단화 관계는 CP와 PD 관계의 재귀적 정의에 의해 다루어진다. 집단화된 객체는 CP관계를 통해 속성들의 집합을 가지며 속성들은 다시 PD 관계에 의해 DOMAIN 클래스로 정의된다. 이와같이 재귀적으로 반복정의되어 집단화 관계가 설정된다.

VERSANT의 클래스에 대한 지식베이스 객체는 그림 14와 같다.

탐색 방법과 전략은 주어진 입력, 요구되는 출력, 구조적인 지식들 사이의 관계에 의해 개발되어질 수 있다. 자세한 개발 방법은 아래와 같다.

① 입력에서 주어진 클래스에 대해 OOAM 지식베이스 객체를 사용하여 그 객체가 접근할 수 있는 모든 속성들을 찾는다.

② 선택된 속성들에 대해 VERSANT 지식베이스 객체를 사용하여 그 속성이 위치한 클래스를 찾는다.

```
(v-object class-name
  (inherit superclass)
  (attributes (list (name attr-name)
                    (location c-a-object))) )
```

그림 14. 지식베이스 객체(VERSANT)

예를 통한 설계 방법과 구현은 설계 및 구현 단계에서 자세히 설명한다.

5. 결 론

현재 관심이 고조되고 있는 객체 지향 설계와 구축 환경은 통합된 KB/DB 시스템 개발에 있어서 발생하는 많은 문제점들을 해결할 수 있는 방안을 제시하고 있다. 특히 클래스, 계승, 데이타 은닉과 같은 객체 지향 개념은 데이타와 지식의 분리나 통합을 자연스럽게 해준다. 그러나 KB/DB 시스템 구축을 위한 객체 지향 모델과 이를 이용한 일반적인 방법론이 제시되지 않았다.

객체중심측면 모델(OOAM)은 데이타베이스의 개념적 모델링에서 실세계의 각 객체에 대해 다수의 측면 표현을 가능하게 한 MAO 모델에 추상화 개념을 강조하여 확장시킨 객체 중심의 데이타 모델이다. OOAM은 실세계를 관점에 따라 여러 측면으로 모델링하므로 모델링 접근을 용이하게 하며 보다 더 융통성 있는 모델링 방법을 제공한다.

본 논문에서는 객체 중심 측면 모델의 기본 개념에 대해 설명한 뒤 OOAM에 의해 구축되는 데이타베이스 스키마의 시맨틱을 분석하고 서술하기 위해 OOAM을 형식적으로 정의하였다. OOAM에 관련된 공리들(속성, 클래스, 클래스 타입, 측면 등)은 데이타베이스 설계 과정에서 데이타베이스 설계의 간략한 기술을 얻는데 사용된다. 유도된 규칙들은 클래스들간의 관계에 의해 명확한 시맨틱을 가지며 지식베이스에 구축될 수 있다.

또한 OOAM을 이용하여 KB/DB 통합 중 intensional 지식중 구조적 지식을 위한 개발 방법론을 제시하였다. OOAM은 실세계의 각 객체에 대해 다수의 측면 표현을 가능하게 하

여 KB와 DB 모델링에 모두 적합하므로 KB/DB의 통합에 매우 유용하게 사용될 수 있다. 먼저 데이타베이스를 개발하기 위한 방법론을 단계별로 자세히 보여준 후 VERSANT 시스템으로 구현하기 위한 변환 방법을 서술하였다. 그리고 이 스키마에 대한 지식베이스 개발 방법론을 자세히 서술했다. 제시한 방법론은 기존의 여러 연구보다 자연스럽고 효율적이며 일반적인 방법론으로 보다 더 추상적인 intensional 지식을 표현하기 위한 확장이 용이하다.

앞으로의 연구과제는 효율적인 질의 처리을 위한 스키마의 관리방법, 버젼관리, DB로부터 지식을 추출하고 DB내의 데이타에 지식을 적용하기 위한 통합된 KB/DB 시스템의 구축이다. 그리고 DB 스키마 변경시 이를 용이하게 할 수 있는 고수준의 도구, 데이타베이스와 지식베이스를 개발하기 위한 전문가 시스템의 도입 등이다.

참고논문

김일도, 다중측면 지식객체에 의한 데이타베이스 설계시스템, 고려대학교, 박사학위논문

김형주, "성질 계승 그래프:객체 중심 데이타베이스 스키마를 위한 형식 모델", 한국정보과학회 논문지, Vol.16, No.5, 1989.

박종태, 성종진, 여상록, "지식베이스 시스템과 데이타베이스 관리 시스템의 통합", 데이타베이스 연구회지, 7권, 1호, 1991.

안상현, 황수찬, 이석호, "객체중심 데이타 모델에서 뷰 개념의 지원", 한국정보과학회 논문지, Vol.17, No.6, November, 1990.

오선영, 백두권, 객체 중심 측면 모델에 의한 개념적 데이타베이스 모델링, 한국정보과학회 학술발표 논문집, Vol. 19, No.2, 1992.

오선영, 백두권, "객체 중심 측면 모델에 의한 KB/DB의 설계", 한국정보처리응용학회 춘계 학술발표 논문집, 제1권, 제1호, 1994.

오선영, 백두권, "객체 중심 측면 모델의 형식론", DB 산업 기술 활성화를 위한 학술대회 및 기술 심포지움, 1994B.

조동영, 다중측면을 고려한 객체 중심 데이타베이스 모델링, 고려대학교, 박사학위논문, 1991.

Banerjee, J., Chou, H. T., Garza, J. F., Kim, W., Woelk, D., and Ballou, N., "Data Model Issues for Object-Oriented Appicatoins", ACM Trans. on Office Information Systems, 5(1), January, 1987.

Blaha, M. R., W. J. Premerlani, and J. E. Rumbaugh, "Relational Database Design using an Object-Oriented Methodology", Comm. of the ACM, Vol.31, No.4, April, 1988.

Cattell, R. G. G., Object Data Management: Object-Oriented and Extended Relational Database Systems, Addison-Wesley, Reading, MA.

Chen Weidong, David S. Warren, "C-Logic of Complex Objects", Proceedings of the 8th ACM'S SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, March 1989.

Clyde, Stephen W., David W. Embley, Scott N. Woodfield, "Tunable Formalism in Object-Oriented Systems Analysis:Meeting

the Needs of Both the Theoreticians and Practitioners", OOPSLA'92.

Fishman, D. H., et. al., "Iris : An Object Oriented Database Management System", ACM Trans. on Office Information Systems, 5(1), January, 1987.

Higa, K. and Lie Sheng, O. R., An Object-Oriented Methodology for End-User Logical Database Design : The Structured Entity Model Approach, In Proc. of Compsac '89, September 1989.

Higa, K., Mike Morrison, Joline Morrison, Olivia R. Lie Sheng, An Object-Oriented Methodology for Knowledge Base/Database Coupling, Comm. ACM, Vol.35, No.6, 1992.

Hughes, J.G., Object-Oriented Databases, Prentice-Hall, 1991.

Kifer, M., Georg Lausen, "F-Logic:A Higher-Order Language for Reasoning about Objects, Inheritance, and Scheme", SIGMOD RECORD, 1989.

Kifer, M., James Wa, "A Logic for Object-Oriented Logic Programming", SIGMOD RECORD, 1989B.

Kim, W., Introduction to Object-oriented Databases, The MIT Press, 1990.

Masahiko Tsukamoto, "DOT:A Term Representation Using DOT Algebra for Knowledge-Bases", Second Internatonal Conference on Deductive and Object-Oriented Databases, 1991.

Rumbaugh, J. E., et. al., Object-Oriented Modeling and Design, Prentice-Hall, 1991.

Schiel, U., "Abstractions in Semantic Networks", ACM SIGART Newsletter, No. 107, 1989.

Siebes, A., "Using Design Axioms and Topology to Model Database Semantics", Proceedings of the 13th VLDB Conference, 1987.

Smith, D. N., Concepts of Object-Oriented Programmong, McGraw-Hill, 1991.

Song, Il-Yeol, H. M. Godsey, "A Knowledge Based System Converting ER Model into an Object-Oriented Database Schem, DASFAA '93, 1993.

Ullman, Principles of Database Systems, Computer Science Press, 1982.

Weddell, Grant E., "A theory of Functional Dependencies for Object-Oriented Data Models", The first Internationsal Conference on Deductive and Object-Oriented Databases, 1989.

Weiderhold, G., Knowledge and Database Management, IEEE Software, Jan. 1984.