공학 연구논문집 제27권 제1호 pp. 1~14, 울산대학교
Journal of Enginering Research Vol. 27, No. 1, pp. 1~14, 1996
University of Ulsan.

설계 데이터베이스의 구성과 응용에 관한 연구

고 재 진
전자계산학과

<요 약>

설계 데이터베이스 모델링 시스템(DDMS)의 개념적 모델에 대해서 기술한다.

DDMS는 복잡한 설계 분야를 위한 의미적 객체와 관계들을 제공한다. 그리고 스키마 설계자가 객체와 관계 타입에 대한 이행규약을 지정할 수 있도록 한다. 의미적 그리고 객체지향적 모델에서 관계의 해석과 DDMS가 지원하는 혼합 모델에서 관계의 해석을 비교한다. 그리고 건축 설계 지원 시스템의 모델링 요구사항을 사용해서 DDMS의 성능을 예시한다.

A Study on the Organization and Application of Design Database

Jae Jin Koh
Dept. of Computer Science

<Abstract>

We describe the conceptual model of a Design Database Modelling System(DDMS). DDMS supports both semantic objects and relationships for complex design domains. It permits the schema designer to specify enforcement rules on the object and relationship types. We compare the interpretations of relationships within the semantic and object- oriented models to the mixed model that DDMS supports.

We then use the modelling requirements of an architectural design support system to demonstrate the capabilities of DDMS.

* 본 연구는 1995년도 울산대학교 연구비 지원에 의해 수행되었음.

1. 서 론

대형 설계 과제를 관리하는 것은 다단계의 절차가 필요하고 다양한 양상을 갖는다. 이러한 일을 계획하고, 일정을 관리하고, 조정하고, 분석하는 것은 컴퓨터의 도움이 필요하다.

여기서는 설계 시스템을 지원하는 의미적 및 객체 지향 설계 데이터베이스의 의미적 모델링에 대해서 기술한다. 표준적인 의미적 모델은 설계 시스템을 지원하기에는 상세 기능면에서 부족하기 때문에 많은 특수 관계 타입을 갖춘 데이터 모델이 필요하다.

예를 들면 IS-A, 집합체(aggregation), 연관(association), 분류(classification)같은 표준 모델에 있는 관계 이외에 is-part-of, role, alternate, procedure 같은 관계가 설계시스템에 더 필요하다[6]. 즉 data encapsulation과 연관된 동적 구조(associated dynamic structure)가 더 필요하다.

설계 데이터베이스를 모델링하는 하나의 접근방법은 객체지향적 데이터베이스를 이용하는 것이다. 객체지향적 데이터모델[5]은 object class의 procedural knowledge를 표현할 수 있기 때문에 설계 데이터베이스를 모델링하는데 많이 이용되어 왔다. 그러나 IS-A 관계가 없기 때문에 객체지향적 데이터모델은 본래의 의미를 표현하는 관계를 제공하지 못한다. 따라서 설계 데이터베이스 개발자는 특수한 관계를 구성하여야 한다.

복잡한 관계의 의미를 분석하기 위해서는 본래의 관계를 사용해서 데이터베이스의 의미를 신중하게 분석할 수 있어야 한다. 본래의 의미란 intra-object 제한조건과 inter object 제한조건의 형태로 나타난다. 설계데이터베이스를 모델링하는데 일어나는 다른 문제점은 관계의 구조와 의미가 표준적이지 못하다는 것이다. 즉 응용분야마다 다르다는 것이다. Sathi등[7]은 몇개의 표현 layers 구조를 제안했다. 이 layers는 특정 응용의 설계 구조를 제공하는 설계 인터페이스를 더 일반적인 구현 layers로의 매핑으로 나타낸다.

예를 들면, 건축분야에서 part 관계는 물건과 그것의 부품관계를 나타낸다. 이것은 특정 응용 에 국한되지 않는 일반적인 모델로 확장할 수 있다. 따라서 HAS-PART라는 관계를 일반화 할 수 있고, 이것에 1-N 링크의 관계수 제한조건을 줄 수 있다. 이런 일반화를 통해서 객체지향적인 데이터베이스의 구현을 실현할 수 있다. Eastman등[3]은 공학제품 데이터베이스를 위한 데이터모델을 개발했다. 이것은 논리식 형태의 제한조건으로 표현된 의미를 갖는 몇개의 설계 지향적인 본래의 관계들을 채택했다. 이런 구조를 통해서 상위의 의미가 몇개의 관계위에 표현될 수 있고, 지엽적 그리고 총체적 스키마 설계와 설계 통합을 이룰 수 있다. ADAM[2]이란 객체 지향 데이터베이스에서는 관계는 객체로 정의되고, 구조적인 동적 양상으로 간주되는 관계 의미의 encapsulation을 허용한다.

관계가 객체로 정의되면 관계는 IS-A 계층 구조로 들어오고, 사용자는 특수한 관계를 생성할 수 있다.

이 논문은 설계 데이터베이스의 복잡한 객체와 관계의 정의를 위한 데이터모델링 방법을 제시한다. 2절에서는 설계 데이터베이스를 위한 의미적 모델과 객체 지향적 모델의 관계 모델들을 비교한다. 3절에서는 설계 데이터베이스에 많이 사용되는 관계 타입들의 예로서 건축 설계 지원 시스템을 제시한다. 4절에서는 설계 데이터베이스 모델링 시스템에 대해서 기술한다. 5절에서는 건축 설계 시스템에의 적용예를 제시한다. 6절에서는 설계 데이터베이스 모델링 시스템의 구현에 대해서 기술한다.

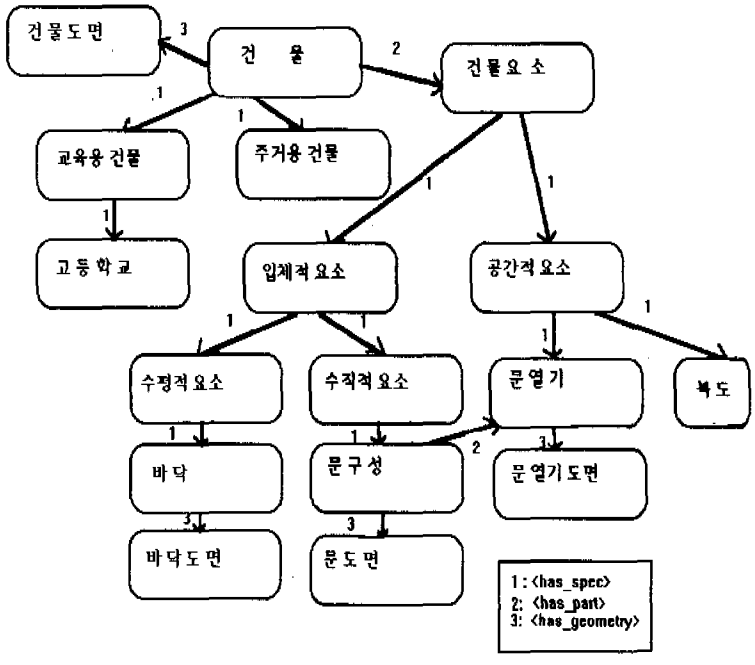
2. 관계(relationships)

데이터베이스 스키마의 객체들 사이의 관계를 규정하는 방법에는 여러가지 관점이 있을 수 있다. 예를 들면 관계를 정보 흐름의 통로로 생각할 수 있다. 여기서는 관계를 객체들 사이의 연관 관계를 제한하는 구조로 생각한다. 예를 들면 존재 제한과 관계수 제한은 다른 객체에 연관되는 객체들의 존재와 관계수의 제한을 표현한다.

관계는 데이터의 공유와 구성을 표현하고, 데이터베이스 스키마의 객체들 사이의 연산들의 공유와 연관을 촉진시킨다. 이것은 객체지향 모델의 IS-A 표현에 의해서 예시된다.

그림 1. 은 여러종류의 계층을 사용해서 데이터베이스 스키마 설계자가 설계 객체들을 구성한 예이다.

그림1. 건물 구성요소 계층도



몇명의 저자들이 의미적 모델과 객체지향 모델을 비교 검토하였다[4,5]. 객체지향 모델은 구조와 연산의 통합 모델링 능력이 있는 반면에, 의미적 모델은 구조적 모델링 능력에 모아 졌다. 의미적 데이터 모델링은 객체지향 모델보다 구체적 구조를 사용해서 객체들 사이의 관계를 표현하는 경향이 있다. ER(Entity Relationship)모델[1]에서 관계는 객체와 구분되지만, 관계도 데이터를 가질 수 있기 때문에 객체와 비슷하다.

의미적 표현은 객체간 제한과 객체간 활동을 제공하기 때문에 편리하다. 더불어 양방향 관계도 쉽게 표현할 수 있다. 그리고 계층적 구조와 깊게 중첩된 구조도 표현할 수 있다.

예를 들면 그림 1.에서 여러종류의 객체 타입과 관계 타입이 표현되고 있다. 각 관계의 의미가 매우 다르기 때문에 분석하기에 매우 혼란스러운 구조를 하고 있다. 관계를 별도의 형을 갖는 구조로 생각하는 것은 데이터베이스의 수정 활동을 규정하고, 분석하고, 이해하는 데 많은 도움을 준다. 객체지향 모델에서는 객체 타입과 연관된 메소드가 데이터베이스 객체의 구조와 활동을 정의하는 방식의 기본 구조로 되어 있다.

관계 표현의 의미적 모델과 객체지향 모델을 분석해 보면 각 모델이 장단점을 각각 가지고 있음을 알 수 있다. 의미적 모델에서 객체와 관계를 구분하는 점은 깊이 중첩된 구조와 이 구조에 바탕을 둔 객체간 활동을 표현하고 분석하는 데 편리하다.

관계의 객체지향 표현은 복잡한 객체 타입의 규정뿐만 아니라 메소드를 정의하는데 편리하다.

3. 건축 설계 지원시스템의 예

이 절에서는 통합적인 객체지향 데이터베이스의 정의를 위해서 필요한 데이터 모델링 기능들을 건축 설계 지원 시스템의 예를 통해서 제시한다. 여기서는 관계의 의미를 규정하기 위해서 미리 규정된 관계를 지원한다. 이 시스템에서는 각 건축물 객체들의 의미적 표현을 저장해서 사용한다. 설계 영역에서의 하나의 문제점은 비슷하다고 생각되는 객체들의 표현이 상당히 서로 다를 수 있다는 것이다. 객체 실재는 그것이 속하는 객체 타입보다 더 복잡하고 구체적이다. 따라서 설계 객체 실재들 사이의 관계 의미를 정의할 필요가 있고 이 의미들을 데이터 모델에서 지원해야 한다.

건축 설계 지원 시스템에서의 관계들의 집합은 다음과 같다.

- <has_instance> : 객체 실재는 이 링크를 통해서 그것의 타입에 연결된다. 객체 실재는 타입으로부터 속성과 메소드를 상속받는다.
- <has_spec> : 두개의 객체 타입사이의 관계를 나타내고, IS_A와 같은 뜻이다.
- <has_attr> : 특정한 값이 특정한 객체에 연관되는 것을 나타낸다. 속성값은 원자적이다.
- <has_geometry> : 설계 객체와 설계 도면과의 이진관계를 나타낸다.
- <has_member> : 객체 집합과 그것을 구성하는 개별 객체사이의 관계를 나타낸다.
- <has_part> : 설계 객체들의 기본적인 계층관계를 나타낸다. 예를들면 한 층은 여러개의 방들로 구성되고, 각 방은 벽들과 출입구로 구성된다. 출입구는 문짝을 포함하고, 문짝은 문틀과 문으로 구성된다.
- <has_role> : 설계 객체와 그의 역할간의 계층적 관계를 나타낸다. 객체는 기본타입을 갖음과 동시에 역할 타입으로부터 상속된 특정한 역할을 갖는다.
  예를들면, 모든 문이 비상구의 역할을 수행하는 것이 아니다. 어떤 경우에는 창문이 비상구의 역할을 수행하는 경우도 있다. 따라서 기능에 따른 역할 타입 계층이 지식표현의 부분이 된다.
- <adjacent_to> : 설계에서 객체와 그것에 이웃하는 부품간의 관계를 나타낸다.

두개의 객체가 <has_part> 관계에 참여하지 않고, 그들간의 거리가 특정값 이내이면 이웃관계라 한다.

- <connected_to> : 두 객체간에 겹치는 부피가 있을 때의 관계를 나타낸다.

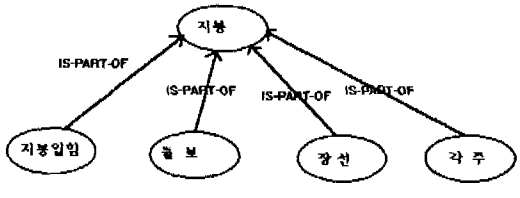
도면 클래스와 기호적 클래스는 별개의 클래스 계층으로 존재한다. 각 객체 실재는 도면적인 면과 기호적인 면을 둘다 갖는다. 도면 모델에서의 부품 계층과 의미적 모델에서의 부품 계층은 1 대 1 대응관계를 갖는다. 도면 모델과 의미적 모델로 구분하는 것은 설계를 단순화하고, 일관적인 시스템을 구성할 수 있도록 한다. 관계가 주요한 개념이기 때문에 데이터베이스 수정 활동은 객체사이에 관계들의 삽입 또는 삭제로서 가장 낮은 레벨에서 이루어 진다. 예를들면 새로운 타입의 객체 실재를 생성하는 메소드를 부르면 일련의 낮은 레벨의 데이터베이스 수정활동이 생성되어서, 방 실재를 삽입하고 벽 실재를 방 실재에 연결하고 최대 높이 속성값을 세트한다.

4. 설계 데이터베이스 모델링 시스템 (Design Database Modelling System)

여기서 논하는 설계 데이터베이스 모델링 시스템은 의미적 객체/관계 모델링 시스템이다.

의미적 객체/관계 모델링 시스템은 의미적 객체와 의미적 관계의 정의를 받아서, 그것을 지정된 제한조건과 수정 의미를 유지하는 데이터베이스를 구현하는 객체지향 언어로 자동적으로 변환하는 모델링 시스템이다. 예를 들면 part 관계는 설계 응용에서 자주 응용되므로, 이 관계의 구조와 의미를 갖는 미리 정해진 관계 타입이 제공된다.

그림 2. 지붕 객체 클래스의 정의



4.1 관계의 의미

관계의 의미는 구조, 관계수, 존재, 통고, 선택규칙, 상속으로 정의한다.

- 구조 : 관계에 참여하는 타입들간의 관계 연결을 규정한다. 예를들면 IS-PART-OF는 한개의 소유타입과 한개 이상의 part타입간의 관계를 나타낸다.
- 관계수 : 주어진 타입이 관계연결을 통해서 다른 타입에 연결될 수 있는 최대의 객체수를 규정한다. 예를 들면 IS-A는 관계수가 1-1이다. 왜냐하면 하나의 supertype 객체가 IS-A 연결을 통해서 하나의 subtype 객체에 연결되기 때문이다.
- 존재 : 관계 연결을 통한 객체들의 상대적인 존재를 규정한다. 예를 들면 들보와 지붕 타입간의 IS-PART-OF 관계의 존재 의미는 모든 지붕 객체는 IS-PART-OF를 통해서 어떤 들보 객체에 연결되어야 함을 규정한다.
- 통고 : 이행규칙을 유지하기 위해서 시스템에 의해서 취해진 모든 행위를 사용자에게 통고하도록 한다. 예를 들면 사용자가 지붕 객체를 삭제해서, 시스템이 그것의 part를 전부 삭제했다면 사용자에게 그것을 통고한다.
- 선택 : 한 객체에 대해서 질의를 했을 때 사용자에게 제시되는 관련된 객체들을 규정한다. 예를들면 지붕 객체에 대해서 질의되었을 때 IS-PART-OF 관계에 의해서 연결되는 관련된 part객체들도 표시될 수 있을 것이다.
- 상속 : 객체들이 관련된 타입들로부터 정의들을 재사용할 수 있는 방법을 규정한다.

각 관계에 따라서 이런 의미들중에 어떤 것은 필수이고 어떤 것은 선택사항이 된다.

모든 관계에 대해서 두개의 관계 이행규칙이 있는데, 관계 삽입규칙은 관계가 정확히 삽입되기 위해서는 참여하는 객체들이 이미 삽입되어 있어야 한다는 것이고, 관계 삭제규칙은 어떤 참여하는 객체가 삭제되었다면 그 관계 실재도 역시 삭제되어야 한다는 것이다.

(1) HAS-ATTRIBUTE 의미

HAS-ATTRIBUTE관계는 관계 및 의미적 모델에서 집합체(aggregate)의 형태로 제공된다. 속성이 부분 범위 타입이나 시스템 고유 타입인 경우에 그 의미는 자명하다.

속성값이 범위를 벗어날 때의 시스템의 반응을 규정하면 된다. 속성값이 다른 객체를 지칭할 때의 의미를 규정할 필요도 있다. 역관계가 유지될 때 양방향 관계를 정의할 필요도 있다. 예를 들면 DOOR타입이 OPENING-MECHANISM타입의 Opener속성을 갖는다고 가정하자. OPENING-MECHANISM은 객체타입이다. 그러면 door와 그것의 opener사이에 opens의 역관계를 다음과 같이 정의할 수 있다.

DOOR $Opens^{-1}$: HAS-ATTRIBUTE Opener : OPENING-MECHANISM (2) IS-A 관계 의미 IS-A 관계는 의미적 그리고 객체지향 데이터모델에서 중심적인 역할을 한다.

그렇지만 IS-A의 의미는 고정되어 있지 않다. 여러 문제들 중 하나가 상속에서 발생한다.

예를 들면 IS-A의 어떤 의미적 해석에서 만약 "회전문 IS-A 문"이고, 문은 속성으로 색을 갖고 있다면, 이 속성은 회전문에게 상속된다. 따라서 문의 색이 청색이라면 회전문의 색도 청색이라고 생각할 수 있다. 어떤 타입이 여러개의 상위타입을 갖는다면 다중 상

속이 발생한다. 이런 경우 상속 속성들의 충돌을 해결하는 기법이 개발되어야 한다.

여기서는 IS-A 관계에서 필수적인 핵심 의미를 갖는 일반적인 IS-A를 정의한다. 그리고 선택에 의해서 부가적인 의미를 부과할 수 있도록 한다. 다음과 같은 의미를 추가적으로 부과할 수 있다.

1) 상속 의미
   (a) 다중 상속 : 속성 충돌을 해결하는 기능을 갖는 다중 상속이 제공된다.
   (b) 상속 억제 : 상속을 억제한다.
2) 관계수/존재 의미
   (a) Strict mandatory subtype membership : 상위 타입의 실재인 모든 객체는 반드시 부분 타입의 하나의 실재이어야 한다.
   (b) Mandatory subtype membership : 상위 타입의 실재인 모든 객체는 한개 이상의 부분 타입 의 실재이어야 한다.

(3) IS-PART-OF 관계 의미

IS-PART-OF관계는 설계 데이터베이스를 위해서 필요하다. 그러나 이 관계에 대한 의미는 범용적으로 설정되어 있지 않다. 따라서 IS-PART-OF 관계의 구조와 관계수를 기술하는 의미를 갖는 일반 관계를 정의한다. 다른 의미는 선택사항으로 주어진다.

IS-PART-OF 관계의 구조는 그림2. 에서 보여준다. 지붕은 소유 객체타입이고 그것을 구성하는 part타입의 지붕입힘, 들보, 장선, 각주 객체가 있다. 지붕과 각 part타입 간의 관계수는 Ni이고 그것의 의미는 각 part 객체는 최대로 Ni 소유 객체들의 part-of가 될 수 있다. IS-PART-OF 관계는 설계 객체의 부품들의 타입과 수가 알려져 있을 때 사용된다. 어떤 부품 관계가 설정되어야 하는지를 결정하는 규칙은 존재와 통고 의미에 의해서 주어진다.

1) 존재 의미
   (a) 하나의 부품 객체는 적어도 하나의 소유 객체에 소속되어야 한다.
   (b) 하나의 소유 객체는 모든 그것의 부품 객체가 없으면 존재할 수 없다.
2) 통고 의미
   소유나 부품 객체의 삽입과 삭제에 있어서 그것으로부터 파생되는 활동에 대해서 사용자에게 통고되어야 한다.

(4) COLLECTION 관계 의미

COLLECTION 관계는 어떤 타입의 객체들의 집합으로 구성되는 객체들을 규정하기 위해서 사용된다. COLLECTION 관계의 의미는 특정 응용의 필요에 따라서 다르게 된다.

COLLECTION 관계의 관계수는 항상 M:N 이다. 이것은 각 collection 객체에 대해서 M개 까지의 member 객체들이 있을 수 있고, 각 member 객체는 N 개의 collection에 있을 수 있다.

1) 존재 의미
   (a) Member 삭제 : collection 객체는 collection안에 적어도 한개의 member가 없으

면 존재할 수 없다.

(b) Collection 삭제 : member 객체들은 관련된 collection 객체가 없으면 존재할 수 없다.

2) 통고 의미 : collection 안의 마지막 객체가 삭제될 때는 그 사실을 사용자에게 통고한다.

(5) 유도 의미

유도 의미는 한 객체의 속성이 다른 객체들의 속성들로부터 유도될 때 사용된다.

유도 의미는 객체에 수정이 일어날 때 유도되는 속성들이 어떻게 정확히 계산되는 지를 규정하는 데 있다.

5. 건축 설계 지원시스템에의 적용예

이 절에서는 건축 설계 지원시스템에의 적용예를 제시한다.

5.1 <has_spec> 관계

<has_spec> 관계는 특수화를 표현한다. 예를 들면 "A <has_spec> B" 라면 B는 A의 특수화를 말한다. <has_spec> 관계는 타입들 사이에 정의되며 1대다이고 이행적인 관계를 나타낸다. 이 관계를 통해서 상속이 일어나고 A타입의 각 객체는 이 연결을 통해서 B타입의 한 객체에 연결된다. 이 관계를 모델링하기 위해서 역IS-A 관계타입을 사용한다.

예를들면 COMPONENT 와 FLOOR, CEILING, WALL 사이에 <has_spec> 관계를 설정할 수 있다.

COMPONENT <has_spec>: IS-A^{-1} FLOOR, CEILING, WALL

5.2 <has_part> 관계

여기에 Room class의 part들을 규정하는 관계 class의 집합이 있다.

Room <has_part> Wallset
Room <has_part> Floor
Room <has_part> Ceiling
Room <has_part> Openingset
Room <has_part> Elementset

만약 Wall, Floor, Room, Ceiling, Element가 미리 객체타입으로 정의되어 있다면 이 구조의 규정은 다음과 같다.

ELEMENTSET COLLECTION ELEMENT

WALLSET COLLECTION WALL
OPENINGSET COLLECTION OPENING

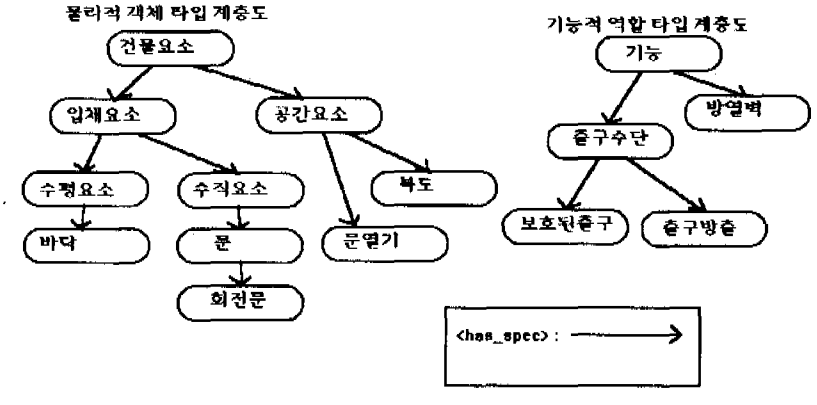
ROOM <has_part> : IS-PART-OF^{-1} WALLSET
ROOM <has_part> : IS-PART-OF^{-1} FLOOR
ROOM <has_part> : IS-PART-OF^{-1} CEILING
ROOM <has_part> : IS-PART-OF^{-1} OPENINGSET
ROOM <has_part> : IS-PART-OF^{-1} ELEMENTSET

여기서 <has_part> 관계를 구성하기 위해서 역IS-PART-OF를 사용했다.

5.3 <has_role> 관계

설계 객체와 그들의 기능적 역할을 표현하는 두개의 서로 다른 IS-A 계층이 있다.
예를 들면 칫수와 위치와 같은 설계 정보는 CORRIDOR형의 한 실재를 구성한다. 그러나 corridor 객체는 출구의 수단으로도 사용될 수 있다. 따라서 출구의 수단으로서의 역할에 관한 정보가 연관되어 진다. 그리고 객체와 역할을 기술하는 두개의 계층을 가질 수 있다.

그림3. 물리적 객체 타입과 역할 타입의 계층도



객체 타입과 역할 타입간에 수평적 연관을 생각할 수 있으나, 그것의 복잡도 때문에 collection과 속성관계 타입의 결합된 형태를 사용한다. 주어진 객체 타입에 대해서 그 타입의 객체가 참여할 수 있는 모든 가능한 역할을 고려한다. 그러면 이런 역할을 특징짓는 가장 특화된 역할 상위 타입을 사용하는 HAS-ATTRIBUTE 관계를 정의한다.

DOOR <has_role> : HAS_ATTRIBUTE Role: MEANS-OF-EGRESS-COMPONENT

여기서 DOOR 타입을 door가 참여할 수 있는 역할 타입에 연관시켰다.

5.4 <has_geometry> 관계

설계의 물리적 구성요소를 표현하는 객체 타입들과 그 객체들의 기하 도형적 배치를 표현하는 객체들 사이에는 특별한 관계가 있다. 이런 경우를 모델링하기 위해서 물리적 구성요소를 표현하는 모든 객체는 상위타입 PHYSICAL-COMPONENT의 부분타입으로 정의된다. 그 상위타입은 SOLID-MODEL의 집합인 속성 Solid-Model-Set를 갖는다.

SOLID-MODEL-SET Collection : COLLECTION SOLID-MODEL
PHYSICAL-COMPONENT <has_geometry>: HAS-ATTRIBUTE Solid-Model-Set: SOLID-MODEL-SET

다음은 <has_geometry> 관계의 의미에 관한 것이다.

- PHYSICAL-COMPONENT 객체는 0개 이상의 geometry를 가질 수 있다. 그러나 0개의 geometry인 경우 집합 객체는 존재할 수 없다.
- 해당하는 PHYSICAL-COMPONENT객체 없이 SOLID-MODEL-SET은 존재할 수 없다.

6. 설계 데이터베이스 모델링 시스템의 구현

설계 데이터베이스는 객체 인터페이스 언어에 의해서 구현된다. 객체 인터페이스 언어는 객체지향 데이터모델에서 의미적 관계를 프로그램할 수 있도록 한다. 이것은 고수준의 객체와 관계 의미를 정의해서, 이 의미를 객체지향 코드로 변환함으로써 가능하다.

두개의 응용 스키마 설계 인터페이스가 있는데, 하나는 구조적 관계 및 제한이고, 다른 하나는 데이터베이스 스키마 설계 도구이다. 각 스키마 설계 인터페이스는 객체와 관계에 대한 고수준의 설계를 해서, 그것을 자동적으로 객체 인터페이스 언어로 출력한다.

설계 데이터베이스 모델링 시스템의 구현에 있어서 다음과 같은 기준이 고려되었다.

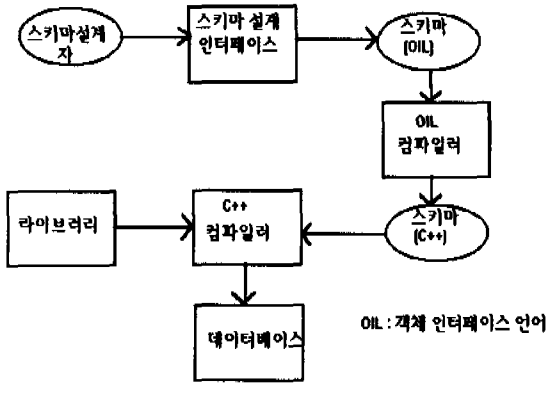
- 설계자의 스키마에 있는 개념적 구조는 데이터베이스에 주소가 부여된 개체로 나타난다. 특히 관계들은 모델링 구조와 데이터베이스 객체로 표현되어야 한다.
- 시스템이 정의하는 관계와 사용자가 정의하는 관계의 의미는 동등하게 제공되어야 한다.
- 객체간 제한을 유지하는 이행규칙의 구현은 객체지향의 encapsulation을 위반하지 않도록 해야 한다. 데이터 객체의 구현은 그들이 참여하는 관계에 의해서 영향을 받아서는 안된다.
- 개념적 스키마로부터 구현으로의 매핑은 자동적이어야 한다.
- 스키마 의미를 구현하기 전에 그것들의 정확성,완전성,일관성을 검증할 도구가 있어야 한다.

이런 기능을 제공하기 위해서 객체 인터페이스 언어는 관계 기술을 모니터 구조를 구현하는 코드로 매핑한다. 이렇게 함으로써 객체들은 관련된 객체들의 활동을 조절하고 관계의 의미들을 유지할 수 있다.

6.1 설계 데이터베이스 모델링 시스템 구현의 개요

설계 데이터베이스 모델링 시스템 구현의 개요는 그림 4. 에 도시되어 있다.

그림 4. 설계 데이터베이스 모델링 시스템 구성도



스키마 설계 인터페이스는 특정 데이터베이스 영역의 특정 의미를 지원하는 교환 가능한 모듈로 생각할 수 있다. 스키마 설계 인터페이스는 객체 인터페이스 언어의 객체 타입 선언의 집합으로서의 스키마 정의를 출력한다. 객체 인터페이스 언어 컴파일러는 객체 지향 DBMS에 해당하는 C++ 코드를 생성한다. 이 코드는 유틸리티와 C++ 컴파일러에 의해서 처리되어서 필요한 데이터베이스를 생성한다. 정확하고 일관성있는 스키마를 만들기 위해서 설계자와 대화하는 의미적 정보를 사용하는 스키마 체커가 개발되어야 한다.

여러 가지 스키마 설계의 잘못을 찾기 위해서 객체, 관계, 이행규칙을 그래프 이론적으로 표현해서 처리하였다.

6.2 객체 인터페이스 언어

객체 인터페이스 언어는 관계를 대화적인 객체로 간주하는 설계 데이터 모델을 지원한다. 관계의 의미는 관계가 참여하는 객체들의 활동에 부과하는 제한에 의해서 정의된다. 객체 인터페이스 언어는 참여자와 모니터를 추가한 C++언어에 기초하고 있다.

참여자 리스트와 모니터 집합이 관계를 데이터 객체와 구별하는 중요한 근거가 된다.

참여자 리스트는 관계에 참여하는 객체 타입들과 그 객체 타입들이 관계에서 수행하는 역할들을 정의한다. 모니터는 관계의 의미를 정의하는 제한들을 구현한다.

6.3 이행규칙의 구현

(구조적 관계 및 제한을 표현하는 DoorHasLatch 관계의 구현)

명세
Relationship: DoorHasLatch
Relationship Type: Has_Part
Composite Object: Door
Part Object: Latch
Enforcement Rules for Deletion of Composite Object:
 Mandatory Deletion

객체 인터페이스 언어 명세:

```
object DoorHasLatch
{participants
 {object Door composite;
  object Latch part;}
monitor (composite.delete) // Mandatory Deletion constraint
 {updateif (part != NULL) delete part;}}
```

(데이터베이스 스키마 설계 도구로 DoorHasOpener 관계의 구현)

명세:
Relationship: DoorHasOpener
Participants:
 Source: Door
 Destination: OpeningMechanism
Constraints:
 Existence: OpeningMechanism depends on Door
 Cardinality: 1_to_1

객체 인터페이스 언어 명세:

```
object DoorHasOpener
{participants
 {object Door source;
  object OpeningMechanism destination;}
```

```
monitor (source.delete) // existence constraint
 {delete destination;}
monitor (destination.delete) // existence constraint
 {reject;}
monitor (selt.insert) // cardinality constraint
 {if (source.ParticipatesIn("DoorHasOpener")) reject;
  if (destination.ParticipatesIn("DoorHasOpener")) reject;}}
```

7. 결 론

이 논문에서는 기본 의미적 관계들을 갖는 객체지향 모델을 확장한 설계 데이터모델을 제시했다. 기본 의미적 관계들로부터 영역에 특수한 관계들을 유도할 수 있다. 이 모델은 건축 분야에서 사용될 수 있는 설계 데이터베이스에 적용하였다. 설계 데이터베이스는 의미적 관계들을 도입할 필요가 있는 좋은 예이다. 사실 설계 모델의 많은 의미가 관계의미의 이해를 통해서 전달될 수 있다. 따라서 관계 의미를 명확히 할 필요가 있다.

어떤 관계는 설계 분야에서 기본적일 수가 있다. 따라서 매번 재사용해서 능률을 올릴 수 있다. <has_part> 관계가 좋은 예이다.

관계간의 관계 의미를 추가하는 일을 연구하는 것은 앞으로 연구할 과제의 하나이다.

주어진 데이터베이스에 정의된 관계들에 작용하는 질의어의 기본틀을 개발하는 것도 또 하나의 앞으로 연구할 과제이다.

설계 분야는 매우 상세한 의미와 활동을 갖는 관계들에 집중되어 있다. 따라서 관계는 매우 중요한 구조적 규범이 된다.

참 고 문 헌

1. Chen, P., The entity-relationship model: Towards a unified view of data, TODS, Vol.1, No.1, P.9-36, 1976.
2. Diaz, O. and Gray, P.M.D., Semantic-rich user-defined relationships as a main constructor In Meersman, R. etc eds., Object Oriented Databases: Analysis, Design, and Construction, North-Holland, New York, P.207-224, 1991.
3. Eastman, C., bond, A., and Chase, S., A data model for design databases, Proceedings of the First International Conference on Artificial Intelligence and Design, Edinburgh, Scotland, 1991.
4. Hull, R. and King, R., Semantic database modelling: Survey, applications, and research issues, ACM Computing Surveys, Vol.19, No.3, P.201-260, 1987.
5. Kim, W., Introduction to Object-Oriented Databases, Computer Systems Series, Cambridge, MA, MIT Press, 1990.
6. MacKellar, B. and Peckham, J., Representing design objects in SORAC: A data

model with semantic objects, relationships and constraints In Gero, J.S., ed.1 Artificial Intelligence in Design, Academic Publishers, Netherlands, pp.201 - 219, 1992.

7. Sathi, A., Fox, M.S., and Greenberg, M., Representation of activity knowledge for project management, IEEE Transaction on Pattern Analysis and Machine Intelligence, 1985.