데이터베이스 스키마 전환규칙 통합을 위한 속성 및 is-a관계

강병영*, 박철제*
*동의대학교 경영정보학과
e-mail:{bykang, cjpark}@dongeui.ac.kr

Attribute and is-a Relationship for Unification of Converting Rule on Database Schema

Byung-Young Kang*, Chul-Jae Park*
*Dept of MIS, Dong-Eui University

요약

본 논문에서는 객체지향 데이터베이스 스키마 전환 규칙 통합을 위한 속성, is-a 관계 및 예외처리를 정의하였다. 개념적모델(ER모델)에서 객체지향 스키마로 전환하는 규칙을 확장하여 개발하였으며 다중상속 문제나 예외처리와 같은 스키마의 변경에 대한 규칙도 제시한다. 객체지향 데이터베이스의 개념적모델을 전환하는 규칙을 정립함으로서 이미 생성된 정보를 활용할 수 있으며, 데이터베이스와 지식베이스를 결합하려는 시도에도 활용할 수 있다.

1. 서론

객체지향데이타베이스(OODB)에 대한 많은 연구가 있지만 객체지향데이타베이스 스키마 설계 방법론에 대한 연구는 만족스럽지 못하고 있다. 데이타베이스 개념적 모델(OOER 모델)을 객체지향 데이타베이스 시스템으로 전환하는 규칙을 정의하는 것은 의미적 접근법에 근거를 두고 있는 기존(예: 관계형 DB)의 설계 방법론들을 이용함으로써 가치 있는 정보의 손실없이 완벽한 객체지향 데이타베이스 설계가 가능하도록 할 수 있다. 또한 이러한 일련의 규칙을 정의함으로써 쉽게 기존 시스템을 객체지향 시스템으로 전환할 수 있다.

개념적모델(ER모델)에서 객체지향 스키마로 전환하는 규칙에 대한 연구로는 Cattell[1]과 Hughes[4]가 제시한 규칙이 있으나 Cattell은 집단화를 어떻게 다루는지를 보이지 않으며, 또한 Cattell과 Hughes 둘 다 키(key) 속성이 아닌 M:M 관계의 속성은 보이지 않고 기본적인 사항들만 제시하고 있다. 또한, 이들 규칙을 확장하여 M:M 관계와 집단화를 다루는 규칙을 개발한 Song과 Godsey의 연구도 있다[10]. 이 연구에서는 ER 모델을 객체지향 데이타베이스의 스키마로 확장하기 위하여 추상 클래스 개념과 메소드, 제약조건 모델링을 확장하여 삼진 관계, 순환 관계, 일반화/특수화, 상속, 집단화와 같은 의미 구조의 전환 규칙을 제시하였다. ER 개념 스키마로부터 객체지향 뷰를 생성하기 위한 매핑 규칙은 Ling 등이 제시한 연구가 있다[7]. 이 연구에서는 OOER 모델에 메소드, 트리거, 유도 속성, 연역 규칙 등을 추가하여 표준 형태 OOER 다이어그램으로부터 객체지향 외부 스키마가 구축되는 매핑 규칙을 보인다.

그러나 위의 연구에서는 이행적 관계와 같은 다양한 관계를 보이지 않았고 관계 사이에서 발생하는 상속 등을 처리하는 문제를 다루지 않았다. 또한 다중 상속의 문제나 예외 처리와 같은 스키마의 변경에 대한 규칙도 제시하지 않고 있으며, 사건과 오퍼

레이션에 대한 규칙도 완전하지 못하다.

본 논문에서는 이들 연구를 바탕으로 완전한 객체 지향 스키마로 전환하는 규칙 중 속성(attribute), is-a관계 및 예외처리를 개발하여 스키마 전환규칙으로 통합하였다.

2. 속성(attribute)

속성은 단일-값 속성, 다중-값 속성, 유도된 속성, 복합 속성, 튜플 속성, 1:1관계의 속성, 1:N관계의 속성 및 M:N관계의 속성으로 구분 할 수 있다.

<규칙 2.1> 기초적 엔터티 타입이 A인 외부클래스 A′는 A′의 속성으로 최소한 A의 한 키(key)를 포함해야 한다. 이는 A′의 갱신 가능성을 보장하는 엔터티-유지 조건이다. 그러나, 관계 타입으로 생성된 외부클래스 A′는 속성으로 기초적(근원적) 관계 타입의 키를 가지지 않을 수 있다. 이 규칙은 어느 엔터티 타입으로도 생성될 수 없는 의미없는 뷰를 보장한다.

<규칙 2.2> 개념적 스키마의 정규 엔터티 타입 A를 고려한다. 근원적 엔터티 타입이 A인 외부 클래스 A′는 A의 모든 단일-값 속성이나 그것의 일부인 데이타-값 속성(또는 A의 복합 속성의 단일 요소로부터)에 기초할 수 있다. A′는 그것의 튜플-값 속성의 어느 것, A의 단일-값 속성의 일부, 복합 속성, 또는 A의 복합 속성의 부분(part)으로 구축될 수 있다. 필요하다면 각 A의 다중 값 속성은 A′의 셀-값 속성으로 매핑될 수 있다.

<규칙 2.3> 정규 엔터티 타입 B가 존재종속이나 식별자 종속 관계 타입 R에 의해 약 엔터티 타입 A와 연결되었다고 생각하자. B는 쉽게 외부 클래스로 매핑된다(B′).

<규칙 2.2>에 의해 A의 외부 클래스(A′)로 매핑은 R에 의존한다. R이 존재 종속관계 타입이라면, <규칙 2.2>를 적용하여 정규 엔터티 타입과 정확히 같은 방법으로 A′는 A로부터 유도될 수 있다. R이 식별자 종속 관계 타입이라면, A′는 그것의 속성의 하나로 B의 식별자를 포함해야 한다. B′에 대한 A′의 존재 종속 제약조건을 부여하는 것이 바람직하다면, B′는 그것의 도메인 엔터티-값 속성이 A′(R에서 A의 카디널리티가 1이면)이거나 도메인의 셀-값 속성이 셀(A′) (R에서 A의 카디널리티가 n, n>1) 중 하나로 열거된다. 속성은 B′에서 A′의 역할을 반영하는 명칭이 주어진다.

3. is-a 관계

<규칙 3.1> 도메인 B의 A′의 속성을 가진 외부 클래스 세분화(specification) A에서, A′의 속성 값은 B나 B의 서브클래스에서 얻어낼 수 있다.

<규칙 3.2> 클래스와 서브클래스 사이에 포함(inclusion) 종속이 유지되는 한, 상속 격자(lattice)의 구조는 다양한 방법으로 뷰 될수 있다.

<규칙 3.3> 두 엔터티 타입 A와 B가 A is-a B와 같은 is-a 관계로 연결되었다면, A는 B의 속성/메소드를 상속하거나 간과할 수 있다.

<규칙 3.4> A is-a B와 같은 두 엔터티 타입 A와 B가 주어지면, B가 관계타입 R과 관계되고, A에 의해 관계타입 R에 있는 관계자 B를 교체함으로써 ER 다이어그램에 있는 R로부터 가상 관계 타입 R′를 구축할 수 있다.

<규칙 3.3>과 <규칙 3.4>는 UNION, INTERSECT, DECOMPOSE와 같은 특수 관계에 적용된다. 예를 들면, 엔터티 타입 A, B_1, …, B_n이 A=UNION(B_1, B_2, …, B_n)과 같은 특수관계 UNION에 의해 연결되면, 모든 i=1, 2, …, n에 대해 B_i is-a A 이다. 외부 클래스 명세화는 <규칙 3.3>과 B를 사용하여 구축될 수 있다.

<규칙 3.5> 외부클래스의 속성값 범주를 규정할 수 있다. 이것은 속성값에 근거한 외부 클래스의 서브클래스를 유도한다. 다른 속성의 값을 기술함으로써 다른 서브클래스들이 정의될 수 있다.

상속과 관련해서는 <공리 1> 부터 <공리 8>까지를 정의할 수 있다.

<공리 1> 모든 클래스들은 이진 is-a 관계로 is-a 계층으로 조직화된다.

<공리 2> Incremental Addition

P = (P_1, ... , P_N)가 C의 특성 리스트이고, C′ is-a C 이면, P′= (P_1′.....,P_N′)는 다음 조건을 가진 C′의 특성 리스트이다.

(a) P′ = P up, where p = (p_1, ... , p_N)는 클래스 C′에서 세분화된 특성 리스트.

(b) P_1, P′와 p′는 is-a계층에서 속성, 관계와 사용자 정의 오퍼레이션이다.

(1) 클래스와 인스턴스

클래스의 확장은 공통 특성을 가지는 인스턴스 집합에 의해 표현된다. 상위클래스와 서브클래스 계층에서, 서브 클래스의 확장은 다음 두 방법에서 상위 클래스의 확장과 관련된다.

<공리 3> 클래스 C'와 C에서 C' is-a C 이면, C'의 모든 인스턴스 또는 C의 인스턴스이다.

<공리 4> 클래스 C가 클래스 {E,F,G}로 구성되는 C' is-a C 이면, C'의 모든 인스턴스는 E' is-a E, F' is-a F와 G' is-a G의 클래스 {E',F',G'}의 인스턴스로 구성될 것이다.

<공리 3>은 클래스의 전체 확장이 상위 클래스 확장의 부분집합임을 나타낸다.

<공리 4>는 복합 클래스를 지원하는 시스템에 관련된다. 몇 개의 요소 클래스로 구성되는 복합 상위 클래스의 is-a 계층에서의 서브클래스를 고려하자. 서브클래스의 각 인스턴스는 상위 클래스를 구성하는 요소 클래스의 서브클래스 인스턴스로 구성된다.

(2) 속성

서브클래스의 속성들을 가지며, 또한 자신의 부가적 속성들은 가질 수 있다. 서브클래스 속성의 도메인은 서브클래스 속성에 도메인의 부분집합이 될 수 있다. 속성 상속은 각 속성에 관련된 명칭, 제약조건, 타입 외 상속을 포함한다.

class c (a=A, ... , X : X)
| is-a U
U: 상위클래스 도메인의 부분집합

class c' [a : A', ... , x : X', y : Y, z : Z]

<공리 5>는 도메인 상속을 정의한다.

<공리 5> A가 도메인 D를 가진 클래스 C의 속성이며 C' is-a C이면, 도메인 D'를 가진 C'의 속성이 존재한다. (D' is-a D) 속성을 상속받는 서브클래스를 상위 클래스 속성과 연관된 디폴트 값을 상속받는다.

<공리 6>은 디폴트 값과 상속을 정의한다.

<공리 6> C' is-a C 에서 A가 도메인 D를 가진 값 V 클래스 C의 속성이라면, 도메인 D'을 가진 클래스 C'의 속성 A'가 존재한다. 값 V'는 다음 두 경우 중 하나이다.

(a) V' = v' (값 v'가 A'에서 지정된 경우)

(b) V' = v (v는 도메인 D'안에서 V'을 A'에 지정하지 않은 경우)

(3) 관계

관계의 상속과 관련 명칭, 카디넬리티 제약조건, 관계와 관련된 클래스를 포함한다.

<공리 7> 관계 R은 C' is-a C이며 클래스 C와 클래스 E를 연결하는 관계라면, 클래스 C'와 클래스 E 또는 E' 연결하는 관계 R이 존재한다(E' is-a E).

(4) 오퍼레이션(operations)

오퍼레이션은 서브클래스에 의해 상속될 수 있다. 다음 두 가지 면을 포함한다.

(a) 속성을 가진 경우 서브클래스는 상속받는 오퍼레이션 외의 오퍼레이션을 가질 수 있다.

(b) 서브클래스의 오퍼레이션은 상위 클래스의 오퍼레이션을 확장할 수 있다. 이것은 다른 함수를 호출하는 함수와 유사하다. 오퍼레이션의 상속은 오퍼레이션 명칭과 코드를 포함한다. 오퍼레이션의 상속은 <공리 8>에 정의된다.

<공리 8> O가 C' is-a C인 클래스 C의 오퍼레이션이면, C'의 오퍼레이션 O'는 다음 세 가지 경우 중 하나이다.

(a) $O' \supseteq O$

(b) $O' \subseteq O$

(c) $O' = O$

4. 예외 처리

<규칙 4.1> 무결성 제약 조건의 완화와 제거

클래스 C가 특정 P:D를 가지고(D는 도메인 범위), $O=V_1$, O_2. $P=V_2$, …, O_n. $P=V_N$과 같은 예외 객체 O_1, O_2,…, O_n 이 있고 V_1, V_2, …, V_n이 D의 범위에 속하지 않는다면, V_1, V_2,…, V_n 값을 가진 새로운 도메인 D'의 특성 P:D'를 만든다.

<규칙 4.2> 기존 클래스의 세분화 변경

클래스 C가 특정 P:D를 가진다면(D는 도메인 범위), O_1. $P=V_1$, O_2. P=V2, …, O_n. $P=V_N$과 같은 예외 객체 O_1, O_2,… , O_n 이 있고 V_1, V_2,…,V_n이 D의 범위에 속하지 않는다면, 특성 P:D를 가진 클래스 C'를 만든다. C'는 클래스 C의 서브클래스이며 특성 P:D'를 가지기 위한 클래스 C를 일반화한 D'를 V_1, V_2, …, V_n를 포함한 도메인 범위를 일반화한다.

<규칙 4.3> 기존 클래스에 새로운 특성 추가

클래스 C가 특정 P:D를 가진다면(D는 도메인 범위), O_1. $P=V_1$, O_2. $P=V_2$, …, O_n. $P=V_N$과 같은 예외 객체 O_1, O_2, …, O_n이 있고 V_1, V_2, …, V_n이 D의 범위에 속하지 않는다면, 특성 P:D를 만든다.

<규칙 4.4> 새로운 서브클래스를 정의함

클래스 C가 특정 P:D를 가진다면 (D는 도메인 범위), O_1. $P=V_1$, O_2. $P=V_2$, …, O_n. $P=V_n$인 클래스 C의 예외 객체 O_1, O_2,…, O_n 이 있고 특성 P가 클래스 C에 존재하지 않는다면, C의 새로운

도메인 D가 V_1, V_2,…, V_n값에 따르는 특성 P:D를 가진 서브 클래스 C'를 만든다.

5. 결론

본 연구에서는 OOER 모델 연구를 바탕으로 여러 속성들의 전환 규칙, 다양한 관계와 복합 객체의 전환 규칙을 추가한 후, 상속에서 일어나는 문제들과 예외 처리 등에 관한 규칙을 정리하였으며, 메소드와 오퍼레이션에 대한 전환 규칙을 정립하여 OOER 모델에서 객체지향 스키마로 전환하는 통합적인 규칙을 만들고자 하였다.

완벽한 개념적 모델링이란 사용자와 설계자 개인의 특성과 시스템 유형에 따라 의견이 다를 수 있다고 본다. 그러나 CIM과 같은 다양한 데이타로 구성되는 이질적 분산 환경에서의 통합 데이타 모델링은 시스템에서 일어나는 데이타의 정확한 표현을 위하여 많은 사항들이 고려되어야 하며, 기존의 데이타 모델보다는 동적 표현이 가능한 객체지향 데이타 모델이 적합하다는 것이 일반적인 주장이다. 그러나 객체지향 데이타 모델은 기존의 데이타 모델과의 유사성으로 기존 데이타 모델을 확장하여 얻을 수 있다. OOER 모델을 객체지향 데이타 모델로 전환하는 규칙을 정립함으로써 이미 생성된 정보를 활용할 수 있으며, 데이타베이스와 지식베이스를 결합하려는 시도에도 활용할 수 있다.

본 연구의 한계점으로는 복잡한 시스템을 설계하는 완벽한 개념적 모델로 제시하기에는 수리적인 베이스가 부족함을 느낀다. 또한 CIM의 엔지니어링 데이터의 상관관계를 표현하는 동적 모델링의 사례 적용이 부족하여 계속적인 연구가 필요하다. 그리고 향후 연구로는 모델 정확성에 영향을 미치는 변수에 대한 체계적인 연구와 지식베이스와 데이타베이스 통합에 대한 연구로 데이타베이스 설계 영역을 확장하는 것이 가능하다. 또한 동적 모델에 시간 특성을 더 강조하여 객체지향적 실시간 시스템 개발 방법을 체계화할 수 있을 것이다.

참고문헌

[1] Cattell, G. G., "Object Data Management: Object-Oriented and Extended Relational Database Systems", Addison-Wesley, 1991.

[2] Chen P.P.S., "The Entity-Relation Model: Toward a Unified View of Data", *ACM Transaction on Database Systems,* Vol.1, No.1, 1976. pp.9-36.

[3] Gorman K, and J. Choobineh, "The Object-Oriented Entity-Relationship Model(OOERM)", *Journal of MIS,* Vol.7, No.3, Winter, 1990-91. pp.41-65.

[4] Hughes J.G., "Object-Oriented Database", Prentice-Hall, 1991.

[5] Hull R., and R. King, "Semantic Database Modeling: Survey, Applications and Research Issues", ACM Computing Surveys, Vol.19, No.3, Sep.1987, pp.201-260.

[6] Kim W. and F.H. Lochovsky, "Object-Oriented Database and Applications", ACM Press, New York, 1989.

[7] Ling T.W., "A Normal Form for Entity-Relationship Diagrams", Proceedings the 4th International Conference on Entity-Relationship Approach, Oct.28-30, 1985-Chicago, Illinois, pp.24-35.

[8] Ling T.W., P.K. Teo and L.L. Yan. "Generating Object-Oriented Views from an ER-Based Conceptual Schema", Proceedings of The Third International Symposium on Databases for Advanced Applications, Taejon, Korea, Appil 6-8, 1993. pp.148-155.

[9] Navathe S.B. and M.K. Pillalamarri, "OOER: Toward Making the E-R Approach Object-Oriented", *Entity-Relationship Approach* 1989, pp.185-205.

[10] Song Il-Yeol and H. M. Godsey, "A Knowledge Based System Converting ER Model into an Object-Oriented Database Schema", Database System for Advanced Application, Korea, 1993, pp.287-294.