확장된 UML 클래스 다이어그램을 이용한 객체 관계형 데이터베이스 설계 기법

김인철*, 김영웅*
*한성대학교 컴퓨터공학과
e-mail:darkich@hansung.ac.kr, yukim@hansung.ac.kr

A Methode for Object-Relational Database Design with Extended UML Class Diagram

In-Chul Kim*, Young-Ung Kim*
*Dept of Computer Engineering, Han-Sung University

요 약

공학적 기반의 응용 프로그램에서는 복합관계(complex relationship) 및 복합객체(complex object)의 개념이 요구되는데, 이러한 개념들은 비즈니스 응용에 적합한 관계형 데이터베이스로 다루기에는 저장과 검색 시 많은 문제점을 야기한다. 이와 같은 문제점을 해결하기 위해서 객체 관계형 데이터베이스 시스템이 출현하게 되었다.

한편, 고전적인 데이터베이스 설계 기법은 개체 관계형 모델(Entity Relationship Model)과 같은 개념적 모델을 사용하며 데이터 중심의 구조적 관점(structural aspect)만을 고려하는 반면, UML(Unified Modeling Language)같은 객체지향형 설계 도구를 사용하여 데이터베이스를 설계할 경우 구조적 관점 및 행위적 관점(behavioral aspect)을 모두 포함한다. UML은 확장 가능한 언어로서, 특정 응용프로그램에 대한 새로운 스테레오타입(stereotype)의 사용이 가능하다. 데이터베이스 설계를 위한 확장된 UML의 스테레오타입이 제안되었지만, 대부분 관계형 데이터베이스에 초점이 맞추어져 있다.

본 논문에서는 객체 관계형 데이터베이스 설계를 위한 확장된 UML 스테레오타입을 기술하며, 복합관계 및 복합객체를 지원하기 위해 Aggregation, Composition, Association의 개념을 재정의한 설계기법을 제안하고, 제안한 설계기법을 지원하는 설계 도구(ORDesigner)의 구현에 대해서 기술한다.

1. 서론

관계형 데이터베이스는 현재 공학적 기반의 응용 프로그램에서 요구되는 모든 데이터들을 지원하기에는 한계점을 갖고 있다. CAD/CAM(Computer-Aided Design/Computer-Aided Manufacturing), CASE(Computer-Aided Software Engineering), GIS(Geographic Information System)등과 같은 응용들의 특징은 복합관계와 복합객체 개념을 내포하고 있으며, 이러한 개념들을 관계형 데이터베이스에 적용시킬 경우 제 1정규형(First Normal Form)을 만족하기 위해 객체들은 다수의 튜플로 분해되므로 테이블이 너무 많이 생성되어 저장뿐만 아니라, 검색의 경우에 많은 조인연산으로 인해 성능이 상당히 저하된다[1].

이와 같은 문제점들을 해결하기 위해 복합객체 타입, 복합관계, 상속 등의 기능을 지원하는 객체 관계형 데이터베이스라는 새로운 데이터베이스가 출현하였는데, 관계형 데이터베이스의 확장으로서 관계형 데이터베이스의 기능을 지원하면서 복합관계 및 복합객체를 지원할 수 있으므로 기존의 응용 프로그램뿐만 아니라 복잡한 응용 프로그램을 지원할 수 있다.

전통적으로 데이터베이스의 논리적 설계 기법은 개체 관계형 다이어그램(ERD : Entity Relationship Diagram)과 같은 그래픽 기술을 이용하여 관계형 스키마를 표현하지만, ERD는 단지 객체 정적인 측면인 구조적 관점만을 고려하므로, 객체의 동적인 측면인 행위적 관점이 포함된 객체 관계형 데이터베이스의 설계에는 부적합하다.

UML(Unified Modelling Language)과 같은 객체지향형 설계 언어(object-oriented design language)는 객체의 구조적 관점과 행위적 관점을 모두 포함하며 각 응용프로그램에 맞는 특별한 스테레오타입(stereotype), 꼬리표 값(tagged value), 제약조건(constraint)을 정의하여 사용할 수 있다. UML의 클래스 다이어그램(class diagram)은 이러한 개념을 지원하므로 객체 관계형 데이터베이스 설계에 적합

하다.

본 논문에서는 확장된 UML의 클래스 다이어그램을 적용한 객체 관계형 데이터베이스 설계방법론을 기술하며, 확장된 UML의 스테레오타입을 기술하고, 기존의 관계를 객체 관계형 데이터베이스에 적합하게 재정의하며, 제안한 설계기법을 적용할 수 있는 설계도구 구현에 대해 기술한다.

본 논문의 구성은 다음과 같다. 제 2장에서는 확장된 UML과 객체 관계형 데이터베이스 관련 연구들을 기술하고, 제 3장에서는 객체 관계형 데이터베이스에 적합하게 재정의한 aggregation, composition, association을 제안하며, 제 4장에서는 제안한 기법을 적용한 설계 도구(ORDesigner)에 대해 기술한다. 마지막으로 제 5장에서는 결론 및 향후 연구방향에 대해 기술한다.

2. 데이터베이스를 위한 UML의 확장

본 장에서는 기존의 관계형 데이터베이스를 위한 UML의 확장과 객체 관계형 데이터베이스를 위한 UML의 확장 및 ORDesigner에 반영된 UML의 확장에 대해서 기술한다.

2.1. 관계형 데이터베이스를 위한 UML의 확장

UML의 확장 방법은 스테레오타입, 꼬리표 값, 제약조건을 포함한다. 스테레오타입(Stereotype)은 모델요소에 새로운 개체를 생성할 수 있는 기능이며, 꼬리표 값(Tagged value)는 모델요소의 기본속성을 확장할 수 있는 기능이다. 제약조건(Constraint)은 모델요소가 반드시 지켜야 할 조건을 명시한다.

<표 1> 관계형 데이터베이스 설계의 스테레오타입

	Database Element	UML Element	Stereotype
Architectural	Database	Component	<<Database>>
	Schema	Package	<<Schema>>
Conceptual	Persistent class	Class	<<Persistent>>
	Multivalued Attribute	Attribute	<<MA>>
	Calculated Attribute	Attribute	<<DA>>
	Composed Attribute	Attribute	<<CA>>
	Identifier	Attribute	<<ID>>
Logical	Table	Class	<<Table>>
	View	Class	<<View>>
	Column	Attributes	<<Column>>
	Primary Key	Attributes	<<PK>>
	Foreign Key	Attributes	<<FK>>
	NOT NULL Constraint	Attributes	<<NOT NULL>>
	Unique Constraint	Attributes	<<Unique>>
	Trigger	Constraint	<<Trigger>>
	CHECK Constraint	Constraint	<<Check>>
	Stored Procedure	Class	<<Stored Procedure>>
Physical	Tablespace	Component	<<Tablespace>>
	Index	Class	<<Index>>

<표1>은 Conallen의 연구[2] 및 Booch의 연구[3]를 바탕으로 정리한 관계형 데이터베이스 설계를 위해 반영되어야 할 스테레오타입의 예를 보여준다.
하지만 collection type, REF type, method 등의 객체나 객체 관계형 모델에서 사용되는 스테레오타입을 제공하지는 않는다.

2.2. 객체 관계형 데이터베이스를 위한 UML의 확장

SQL:1999는 현재 객체 관계형 데이터베이스의 표준으로 복합객체와 복합관계에 사용되는 새로운 구조적 데이터 타입을 정의할 수 있다. <표 2>는 SQL:1999의 value type과 객체타입으로 정의된 구조적 데이터 타입의 예를 보여준다.

<표 2> SQL:1999의 구조적 데이터 타입의 예

```
CREATE TYPE employee AS (
        id      INTEGER,
        name    VARCHAR(20))

CREATE TABLE department (
        c1'     INTEGER,
        c2'     VARCHAR(20),
        director employee);

CREATE TABLE T_employee OF employee;

TABLE department
| c1 | c2 | director |

TABLE T_employee
| OID | id | name |
```

SQL:1999가 표준이긴 하지만, 현재 상용화되고 있는 객체 관계형 데이터베이스들 마다 차이점을 갖고 있다. 대표적인 예로 <표 3>의 Oracle 9i를 살펴보면 SQL:1999와 다른 문법을 사용함을 알 수 있다.

<표 3> Oracle 9i의 구조적 데이터 타입의 예

```
CREATE TYPE employee AS OBJECT(
        id      INTEGER,
        name    VARCHAR(20))

CREATE TABLE department (
        c1'     INTEGER,
        c2'     VARCHAR(20),
        director employee);

CREATE TABLE T_employee OF employee;
```

<표 4>에서 기술한 Marcos의 연구[4]는 객체 관계형 데이터베이스 설계시 반영되어야 할 스테레오타입을 기술한다.

<표 4> 객체 관계형 데이터베이스의 스테레오타입

UML	SQL:1999	Oracle 9i
Class	Structured Type	Object Type
Class Extension	Typed Table	Tables of Object Type
Attribute	Attribute	Attribute
Multivalued	ARRAY	VARRAY / Nested Table
Composed	ROW / Structured Type in column	Object Type in column
Calculated	Trigger / Method	Trigger / Method
Association		
One-To-One	REF / [REF]	REF / [REF]
One-To-Many	[REF]/[ARRAY]	[REF] / [Nested Table/VARRAY]
Many-To-Many	ARRAY / ARRAY	Nested Table / Nested Table VARRAY / VARRAY
Aggregation	ARRAY	Nested Table / VARRAY of References
Compostion	ARRAY	Nested Table / VARRAY of Objects
Generlization	Types / Typed Tables	Tables of Object Type

3. 관계 정의

관계형 데이터베이스의 관계는 복합객체 및 복합관계 개념을 지원하지 않으므로 객체 관계형 데이터베이스 설계에 대한 새로운 정의가 필요하다. 본 장에서는 Eric Pardede의 연구[5]를 바탕으로 객체 관계형 데이터베이스의 Aggregation, Composition, Association의 개념을 기술한다.

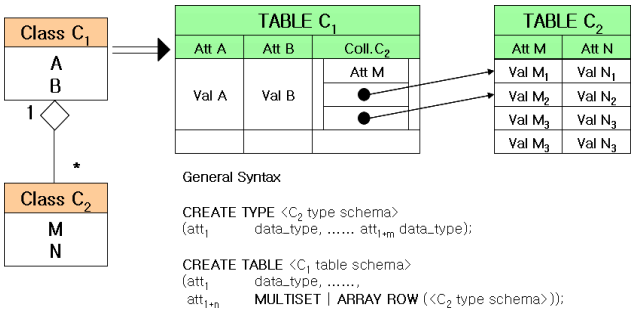
3.1. Aggregation과 Composition

Aggregation과 Composition은 전체-부분(whole-part)의 개념으로 표현하는 클래스들 사이의 연관관계이다. Aggregation이나 Composition인 클래스의 각 객체는 다른 Aggregation이나 Composition의 객체로 구성될 수 있는데, 구성된 클래스를 “전체”라 하며, 구성요소가 된 클래스를 “부분”이라 부른다.

3.1.1. Aggregation

Aggregation은 "부분"클래스가 공유 가능하기 때문에, "전체"에 대해 독립적인 관계를 갖는다. 즉, "전체"의 객체가 삭제되어도 "부분"의 객체는 삭제되지 않는다.

Rule 1: (C1(A, B), C2(M, N))의 두 개의 클래스가 있고, C1은 공유가 가능하며 독립적인 C2로 구성된다면, C2는 C1 테이블의 UDT collection의 속성으로 구현한다. 변환 결과는 테이블 C1(A, B, UDT C2(M, N))이 된다.

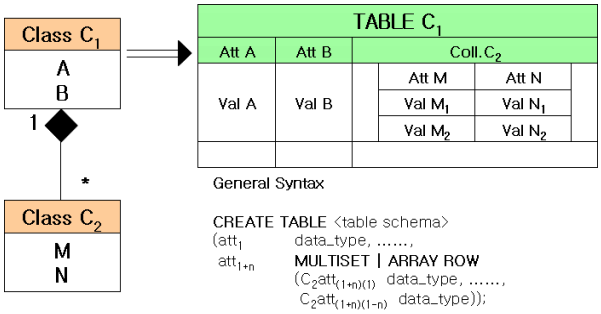


(그림 1) Aggregation Type

3.1.2. Composition

Composition은 Aggregation과는 다르게 "부분" 객체가 "전체" 객체에 의존적이며 공유가 불가능하므로 "부분" 객체를 "전체" 객체의 내부에 정의한다.

Rule 2: (C1(A, B), C2(M, N))의 두 개의 클래스가 있고, C1은 공유가 불가능하며 의존적인 C2로 구성된다면, C2는 C1 테이블의 ROW collection 속성으로 구현한다. 변환 결과는 테이블 C1(A, B, ROW C2(M, N))이 된다.



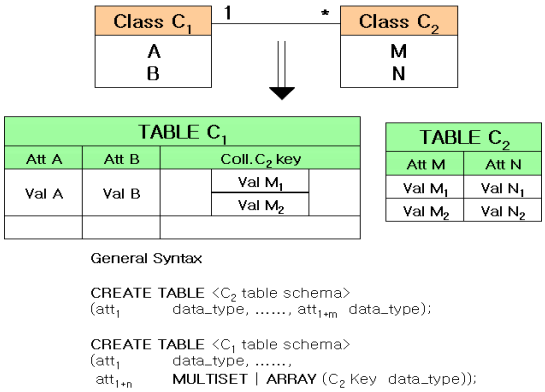
(그림 2) Composition Type

3.2. Association

Association은 1:1, 1:N(일:다), N:N(다:다)의 매핑 대응수(mapping cardinality)에 따라서 차이점을 갖고 있다. 1:1일 경우 관계형 데이터베이스의 경우와 마찬가지로 객체 관계형 데이터베이스도 두 클래스 중 한쪽이 다른 한쪽의 주키를 갖고 있으면 되지만 복합객체를 포함한 클래스들의 관계는 다르게 구현되어야 하므로 N의 매핑 대응수만 고려한다.

3.2.1. 1:N association

Rule 3: (C1(A, B), C2(M, N))의 두 개의 클래스가 있고, C1:C2가 1:N의 association을 갖고 있다면, C1이 C2를 참조할 수 있는 속성을 collection으로 포함한다. 변환 결과는 테이블 C1(A, B, C2_Key)과 C2(M, N)가 된다.

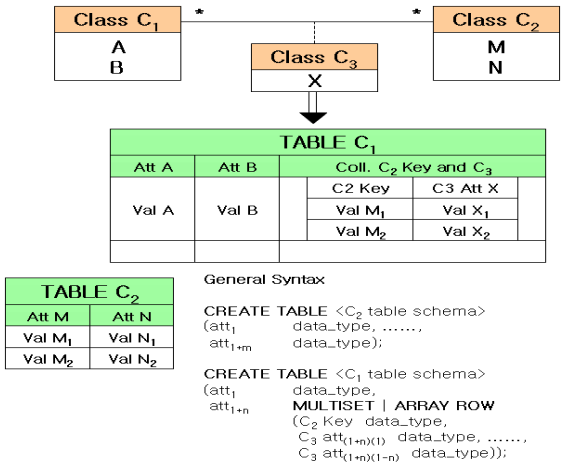


(그림 3) 1:N Association

제시한 방법은 1:N association 관계의 새로운 접근방법으로 collection 의미를 유지하고 있으며, collection과 전통적인 관계형 모델로 수행할 수 없는 순차적 술어 같은 복잡한 질의에 가장 적합하다. 이 방법은 테이블들(multiple-table-query와 "N"쪽이 매우 크지 않은 관계에 대해 자주 사용되는)에 대한 이익을 얻을 수 있다.

3.2.2. N:N association

Rule 4: (C1(A, B), C2(M, N))의 두 개의 클래스가 N:N의 association을 갖고 관계인 C3(X)을 갖고 있다면, C1은 C3과 C2를 참고할 수 있는 속성을 collection으로 포함한다. 변환 결과는 테이블 C1(A, B, ROW(C2 Key, C3(X))과 C2(M, N)가 된다.



(그림 4) N:N Association

1:N association에서 제시한 방법과 마찬가지로 이 방법은 collection과 순차적 술어를 갖는 복잡한 질

의에 적합하다. 만약 참여하는 클래스들이 서로 다른 중요도(importance level)를 갖거나, 관계 속성들의 수가 크지 않을 경우 유용하다.

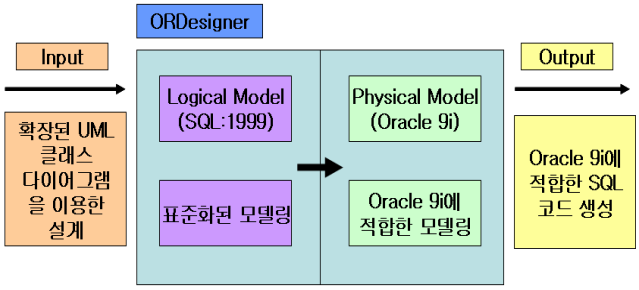
제시한 association 관계는 객체 관계형 데이터베이스 collection에서 무결성 제약 조건이 내포된 SQL을 사용할 수 없다는 단점이 있다. 기존의 관계형 데이터베이스의 association 관계는 외래키 또는 REF를 포함하며, on delete cascade, on update nullify 등과 같은 무결성 제약 조건 정의가 가능하다. 이러한 조건을 collection 속성에 대해서는 적용할 수 없다. 그러나, 제시한 모델에서 무결성 제약 조건을 내포할 수 없다는 의미는 아니다. 만약 테이블이 typed table일 경우 트리거(trigger)나 메소드(method)를 사용하여 무결성 제약 조건을 명시할 수 있다.

4. ORDesigner

ORDesigner는 본 논문에서 제시한 객체 관계형 데이터베이스 설계 기법을 적용한 설계 도구이다.

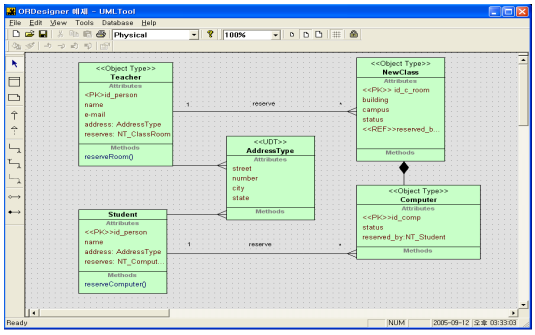
확장된 UML 클래스 다이어그램을 이용하여 데이터의 구조적 관점 및 행위적 관점과 재정의한 aggregation, composition, association을 적용하여 복합객체 및 복합관계를 고려하였다. ORDesigner는 논리적 모델 기반으로 객체 관계형 모델 표준인 SQL:1999를 사용하며, 물리적 모델 기반으로는 Oracle 9i(release version 9.2.0.1.0)를 사용한다. 구현환경은 Windows XP 상태에서 MFC를 이용하여 개발 중이다,

(그림 5)는 ORDesigner의 간략한 구성도이다. ORDesigner는 다루기 쉬운 사용자 인터페이스로 확장된 UML 클래스 다이어그램을 이용하여 설계를 할 수 있다. SQL:1999에 적합한 표준화 모델을 Oracle 9i가 지원하는 데이터 형태에 맞게 변환하여 Oracle 9i에 적합한 SQL 코드를 생성하여 적용한다.



(그림 5) ORDesigner의 간략한 구성도

(그림 6)은 ORDesigner의 프로토타입을 보여준다. 확장된 UML 클래스 다이어그램을 지원하며 논리적 모델링과 물리적 모델링을 구분하여 보여준다. 현재 SQL:1999를 사용한 표준화된 모델링을 Oracle 9i에 적용하여 구현하였다.



(그림 6) ORDesigner 프로토타입

5. 결론 및 향후연구

기존의 관계형 데이터베이스는 복합객체 및 복합관계를 지원함에 있어서 많은 문제점을 내포함으로 객체 관계형 데이터베이스가 출현하게 되었다. 기존의 개체 관계형 모델(ERD)과 같은 관계형 데이터베이스 설계기법은 데이터의 구조적 관점만을 고려하므로 행위적 관점을 포함하는 객체 관계형 데이터베이스에는 적합하지 않다.

본 논문에서는 구조적 관점과 행위적 관점을 모두 포함하는 객체 관계형 데이터베이스 설계를 위한 확장된 UML 스테레오타입을 기술하였으며, 복합객체 및 복합관계를 지원하기 위한 Aggregation, Composition, Association의 개념을 재정의한 설계기법을 기술하였다. 또한 제시한 설계기법을 적용한 설계도구인 ORDesigner의 간략한 소개를 기술하였다.

향후 연구로는 본 논문에서 기술한 설계기법을 지원하는 ORDesigner 개발의 완성과 Oracle 9i 외의 다른 객체 관계형 데이터베이스도 지원하도록 확장되어야 하며, ORDesigner를 이용하여 실질적인 활용 예를 제시하여야 할 것이다.

참고문헌

[1] Bertino and Marcos, "Object Oriented Database Systems.". In *Advanced Databases: Technology and Design*, O. Diaz and M. Piattini (Eds.. Artech Hous,. 2000

[2] Conallen, J. "Building Web Application with UML." Addison-Wesley, 2000

[3] Booch, G., Rumbaugh, J., and Jacobson, I. " The Unified Modeling Language User Guide." Addison Wesley, 1999

[4] Marcos, E. Vela, B. and Cavero, J.M. "Extending UML for Object-Relational Database Design." UML 2001. Springer Verlag. Lectures Notes in Computer Science. LNCS 2185: 225-239, 2001

[5] Eric Pardede, J. Wenny Rahayu. D. Taniar. "Mappting Methods Query for Aggregation and Association in Object-Relational Database using Collection". Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04)