# 객체-관계형 데이터베이스 설계를 위한 UML 확장 -Oracle11g를 중심으로-

## UML Extension for Object-Relational Database Design - Focusing on Oracle11g -

주 경 수* 조 도 형**
Kyung-Soo Joo Do-Hyung Jho

### 요 약

현재의 응용시스템들은 복합관계성으로 관련지어진 복합객체를 갖고 있다는 특징을 갖는다. 이러한 특징은 기존의 관계형 데이터베이스로 표현하기에는 한계가 있어 관계형 데이터베이스는 객체-관계형 데이터베이스로 확장되었다. 이에 따라서 기존의 관계형 데이터베이스 설계 방법론과 같은 객체-관계형 데이터베이스 설계 방법론이 요구된다.

본 논문에서는 스테레오타입과 태그값 그리고 제약조건을 이용한, UML 클래스다이어그램 확장에 기반하여 객체-관계형 데이터베이스를 위한 통합된 설계방법론을 개발한다. 아울러 확장된 UML 클래스다이어그램을 객체-관계형 데이터베이스 스키마로의 변환을 위한 가이드라인을 제시한다. 개발한 설계방법론에서 사용하는 객체-관계형 모델은 SQL:1999를 이용하였으며, 객체-관계형 데이터베이스 구현은 Oracle11g로 하였다. 많은 객체-관계형 DBMS는 존재하지만 객체-관계형 데이터베이스 설계에 있어서 일관된 방법론은 제시되지 않았다. 본 논문에서 제시하는 방법론을 적용함으로써 일관된 방법으로 객체-관계형 데이터베이스 설계를 가능하게 한다. 사례 연구로, 제안한 설계방법론을 컴퓨터교실 예약시스템에 대하여 적용한다.

### ABSTRACT

The current applications can be characterized as consisting of complex objects related by complex relationship. Therefore the relational database has been extended to object-relational database because of the complex objects. Accordingly, we need database design methodology for object-relational database.

In this pager, we develop an integrated design methodology based on a extended UML class diagram for object-relational database. We make the extend UML class diagram by adding new stereotypes for object-relational database. Also we propose a guideline for transforming the extended UML class diagram to object-relational database schema. We use SQL:1999 as a object-relational data model and Oracle11g as a target object-relational database. We can build more easily and efficiently object-relational database for Oracle11g by using our methodology. Finally we applied our methodology on a computer classroom reservation system for evaluation the methodology.

☞ keyword : 객체-관계형(Object-Relational), 데이터베이스(Database), UML, SQL:1999, Oracle 11g

## 1. 서 론

지난 수십 년간 관계형 데이터베이스가 발전을 거듭하고 있지만 현재의 응용시스템이 요구하는 복합 객체의 데이터를 표현하기에는 한계점을 가지고 있다. 하드웨어 발전에 힘입어, 전통적인 정보처리 분야 이외에 CAD/CAM(Computer-Aided Design/Computer-Aided Manufacturing), CASE(Computer-Aided Software Engineering), GIS(Geographic Information System) 등, 더 정교한 시스템이 나타나게 되었다. 이러한 응용시스템들은 복합관계성으로 관련지어진 복합객체로 구성되어 있다는 특징을 갖고 있다. 관계형 모델에서 복합객체는 많은 수의 튜플(tuples)로 분해되어 상당한 수의 조인(join)이

* 종신회원 : 순천향대학교 컴퓨터소프트웨어공학과 교수
gsoojoo@sch.ac.kr
** 준 회 원 : 순천향대학교 컴퓨터소프트웨어공학과 석사과정 jhodohyung@sch.ac.kr
[2011/04/27 투고 - 2011/05/17 심사(2011/08/22 2차) - 2011/09/26 심사완료]

발생하는데, 이는 복합 객체를 검색할 경우 시스템 전체 성능을 크게 떨어뜨리게 한다[1]. 이를 극복하기 위하여 관계형 데이터베이스는 객체-관계형 데이터베이스로 확장되었다[2].

이에 따라 기존의 관계형 데이터베이스 설계 방법론과 같은 객체-관계형 데이터베이스 설계를 위한 방법론이 요구된다. 몇 가지 객체-관계형 데이터베이스 설계 방법론들이 제안되었다. 그러나 이러한 제안된 방법론 중 아무것도 객체-관계형 데이터베이스를 위한 일관된 방안은 제시하지 못하고 있다[3-7].

UML은 통합 모델링 언어로서 데이터 뷰(Data View)를 포함한 시스템 전체 모델링을 가능하게 한다. 그리고 응용시스템에 있어서 필요한 경우에 스테레오타입과 태그값 그리고 제약조건을 통해 확장이 가능하다[8-10].

본 논문에서는 스테레오타입과 태그값 그리고 제약조건 등의 UML 클래스다이어그램 확장에 기반한 객체-관계형 데이터베이스 설계를 위한 방법론을 제시한다. 또한 UML 클래스다이어그램을 객체-관계형 데이터베이스를 위한 개념스키마를 논리스키마로 변환시키기 위한 가이드라인을 제시한다.

객체-관계형 모델로는 다른 객체-관계형 제품에 독립적인 SQL:1999 객체-관계형 모델을 사용하였으며, 구현 제품으로서는 Oracle11g를 사용했다. 아울러 사례 연구로 제안한 방법론을 컴퓨터교실 예약시스템에 대하여 적용한다.

객체-관계형 DBMS(예를 들어, Oracle11g, Oracle9i, Oracle10g, Oracle11g와 MS SQL Server 2000, 2005, 2008과 DB2 7, 8, 9 그리고 Informix 9, 10, 11 등)는 존재하지만 객체-관계형 데이터베이스 설계에 있어서 일관된 방법론은 제시되지 않았다. 본 논문에서 제시하는 방법론을 적용함으로써 UML 클래스다이어그램 확장에 기반한 객체-관계형 데이터베이스 설계와 개념스키마를 논리스키마로의 변환까지의 일관된 방법으로 객체-관계형 데이터베이스 설계를 가능하게 한다.

본 논문의 구성은 다음과 같다. 2절에서 UML 확장 메커니즘(mechanism)을 다루고, 3절에서는 객체-관계형 데이터베이스를 위한 객체-관계형 모델인 SQL:1999와 객체-관계형 DBMS인 Oracle 11g 그리고 객체-관계형 데이터베이스 설계를 위한 UML 확장을 제시한다. 4절에서는 제시한 확장을 사용한 통합된 객체-관계형 데이터베이스 설계방법론 및 예를 보여주고, 마지막으로 5절에서는 결론을 요약한다.

## 2. 관련 연구

UML 확장은 언어를 확장하기 위한 메커니즘이다. 이 메커니즘은 스테레오타입과 태그값 그리고 제약조건을 포함한다[8,11].

1. 스테레오타입 : 스테레오타입은 기존의 파생된 구성요소의 새로운 종류를 생성하는 UML의 어휘 확장과 관련된 개념이다. 스테레오타입은 이중 꺽음표(<<이름>>)로 둘러싸인 이름으로 표현되며 다른 요소의 이름 위에 위치한다. 또한 스테레오타입과 관련된 아이콘 정의가 가능하다.
2. 태그값 : 태그값은 요소의 명세에 새로운 정보를 생성하는 UML의 구성요소로서 속성 확장과 관련된 개념이다. 스테레오타입과 태그값으로 새로운 속성을 추가할 수 있다. 태그값은 중괄호({태그값})로 둘러싸인 문자열로 표현되며 다른 요소의 이름 아래에 위치한다.
3. 제약조건 : 제약은 새로운 규칙을 추가하거나 기존의 수정을 허용하는 UML의 구성요소로서 의미 확장과 관련된 개념이다. 제약은 중괄호({제약조건})로 둘러싸인 문자열로 표현되며 관련되는 요소의 근처에 위치시키거나 종속관계를 이용하여 해당 요소를 연결시킨다.

기존에 확장된 UML 스테레오타입과 복합관계를 지원하기 위한 집합, 복합, 연관 개념의 재정의를 통한 객체 관계형 데이터베이스 연구가 진행되었다[12,13]. 그러나 이는 객체-관계형 데이터베이스 설계와 관련하여 스테레오타입 정의 부분만을

(표 1) 관계형-데이터베이스 설계를 위한 스테레오타입

| DB 요소 | UML 요소 | 스테레오타입 | Icon |
|---|---|---|---|
| Database | Component | <<Database>> |  |
| Schema | Package | <<Schema>> |  |
| Tablespace | Component | <<Tablespace>> | |
| Table | Class | <<Table>> |  |
| View | Class | <<View>> | |
| Index | Class | <<Index>> | |
| Column | Attributes | <<Column>> | |
| Primary Key | Attributes | <<PK>> | |
| Foreign Key | Attributes | <<FK>> | |
| Multivalued Attribute | Attribute | <<AM>> | |
| Calculated Attribute | Attribute | <<AD>> | |
| Composed Attribute | Attribute | <<AC>> | |
| NOT NULL Restriction | Attributes | <<NOT NULL>> | |
| Unique Restriction | Attributes | <<Unique>> | |
| Trigger | Restriction | <<Trigger>> | |
| Restriction | Restriction | <<Check>> | |
| Store Procedure | Class | <<Stored Procedure>> | |

다루고 있으며 개념스키마를 논리스키마로 변환시키는 부분에 있어서는 명확하게 정의하지 못하고 있다.

본 논문에서는 객체-관계형 모델의 표준인 SQL:1999와 객체-관계형 제품인 Oracle 11g의 스테레오타입 정의를 통하여 UML 클래스 다이어그램을 확장한다. 그리고 표준 논리적 설계인 SQL:1999로 변환 후에 제품인 Oracle 11g로의 변환 그리고 개념스키마를 논리스키마로의 변환을 진행함으로써 일관된 방법으로 객체-관계형 데이터베이스 설계를 가능하게 한다.

(표 1)은 기본키(PK)와 외래키(FK) 등을 표현한 스테레오타입이 포함된 관계형 모델이다. 그러나 객체 모델을 위한 컬렉션 타입(collection types)과 참조 타입의 구체적인 스테레오타입은 제공하지 않는다. 다음 절에서 객체-관계형 데이터베이스 설계를 위한 확장을 소개한다.

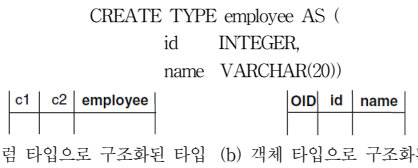
## 3. 객체-관계형 데이터베이스 설계를 위한 확장

객체-관계형 데이터베이스 설계를 위한 UML 확장을 설명하기 전에 객체-관계형 제품에 독립적인 SQL:1999 객체-관계형 모델과 구현 제품인 Oracle11g를 소개한다. SQL:1999 객체-관계형 모델은 관계형 데이터 모델을 확장한다.

### 3.1 객체-관계형 데이터베이스

#### 3.1.1 객체-관계형 모델 SQL:1999

객체-관계형 모델 SQL:1999는 객체-관계형 데이터베이스를 위한 표준으로 객체 모델과 관계형 모델을 통합한다[14,15]. 관계형 모델과 객체-관계형 모델의 주요한 차이점 중 하나는 관계형 모델의 기본 원칙 중 제1정규형(1NF)이 객체-관계형 모델에




(a) 컬럼 타입으로 구조화된 타입 (b) 객체 타입으로 구조화된 타입

(그림 1) 값과 객체 타입으로 사용된 구조화된 타입

서는 제거되었다. 그래서 객체 테이블의 컬럼(column)은 컬렉션 데이터 타입을 포함한다.

객체-관계형 모델 SQL:1999는 응용시스템에 필요한 데이터 타입을 사용자가 정의할 수 있다. 객체-관계형 모델 SQL:1999의 주요한 특징으로 구조화된 타입(structured type)을 제공하고 행동과 캡슐화 그리고 다형성 및 동적 바인딩을 지원한다.

구조화된 타입은 테이블 타입(table type) 또는 컬럼 타입(column type)으로 사용될 수 있으며, 컬럼 정의에서 기본 타입으로 사용된 구조화된 타입은 복합속성 표현을 허용한다. 이 경우, 구조화된 타입은 값 타입(value type)을 대표한다. 구조화된 타입은 객체 타입의 정의에 해당되는 테이블 정의에 사용될 수 있다. 객체-관계형 모델 SQL:1999에서는 이런 종류의 테이블을 타입 테이블(typed tables)이라한다. 객체-관계형 모델 SQL:1999에서 객체는 타입 테이블의 행(row)이다. (그림 1)은 객체-관계형 모델 SQL:1999에서 정의된 구조화된 타입의 예를 보여준다. (a)에서 구조화된 타입은 값 타입으로 사용되고, (b)에서는 객체 타입으로 사용된다[11,13].

OID의 값은 테이블에 새로운 객체가 추가될 때 시스템에 의해 생성되는데 이 컬럼의 타입은 참조 타입(REF)이다. 그래서 각 타입 테이블은 OID 값을 포함하는 컬럼을 가진다. 각 객체 타입을 위한 다른 참조 타입이 있다. 이는 타입 생성자에 해당한다. 참조 타입으로 정의된 속성은 참조된 객체의 OID를 포함한다. 따라서 참조 타입은 외래키 사용 없이 관계를 정의할 수 있다.

객체-관계형 모델 SQL:1999는 추가로 다른 구조화된 행타입을 지원한다. 행타입은 사용자에 의해 정의된 구조화된 타입으로 OID가 아니기 때문에 객체 타입으로 사용될 수 없다. 객체-관계형 모델 SQL:1999는 컬렉션 타입인 배열(ARRAY)을 지원한다.

#### 3.1.2 객체-관계형 DBMS Oralce11g

객체-관계형 DBMS인 Oracle11g는 객체-관계형 모델 SQL:1999와 서로 다른 문법이지만 동일하게 사용자에 의해 구조화된 데이터 타입을 정의할 수 있다[16]. 구조화된 데이터 타입은 객체-관계형 모델 SQL:1999처럼 컬럼 타입 또는 테이블 타입으로 사용된다. 컬럼 타입으로 사용된 구조화된 타입은 값 타입을 대표하며 테이블 타입으로 사용된 구조화된 타입은 객체 타입(타입의 확장인 테이블)을 대표한다. 이 종류의 각 테이블의 각 행은 객체이며 SQL:1999와 같은 방식으로 개별 객체를 식별하기 위한 OID를 허용하는 특수한 참조 타입(REF)의 컬럼을 갖는다. 객체-관계형 DBMS인 Oracle11g는 메소드의 특징 정의를 통해 객체 타입의 행동을 연관시키며 메소드의 몸체는 별도로 정의한다.

객체-관계형 DBMS Oracle11g는 객체-관계형 모델 SQL:1999의 ARRAY와 같은 VARRAYS와 중첩된 테이블(nested table)인 2종류의 컬랙션(collection)을 지원한다. 중첩된 테이블은 다른 테이블에 끼워 넣는 테이블로서 테이블 데이터 타입 정의와 테이블에서 컬럼 타입처럼 사용이 가능하다. 그래서 컬럼은 컬렉션 값과 객체 또는 참조 테이블을 포함한다.

### 3.2 객체-관계형 데이터베이스 설계를 위한 UML 확장

UML을 객체-관계형 데이터베이스 설계를 위한 표준 모델링 언어로 사용하기 원한다면 객체-관계형 스키마를 나타낼 수 있도록 수정되어야 한다. 그래야 배열과 중첩된 테이블 또는 참조 타입을 위해 생성자를 정의할 수 있다.

첫 번째 수정 방안은 UML의 메타모델(meta-model)을 수정하는 것이다. 그러나 메타 모델 수정은 나중에 관리가 불가능하다. 따라서 UML은 관리가 가능한 방법으로 확장되어야 한다. 필요한 부분을 충족시키고 유연성을 제공하기 위해서 2절에서

(표 2) 객체-관계형 데이터베이스 설계를 위한 UML 확장

**설명**
객체-관계형 모델의 각 요소들은 UML로 표현되어야 한다. 스테레오타입 요소를 사용하기 위해 다음과 같은 기준을 사용한다.
- SQL:1999는 구조화된 타입과 스테레오타입 클래스처럼 타입 테이블을 고려한다. 이유는 SQL 스키마에 명백하게 정의되기 때문이다. 타입의 나머지 (예를 들어, 참조와 행 그리고 배열)는 스테레오타입 속성처럼 고려된다.
- Oracle 11g는 객체 타입(object type)과 객체 테이블(object table) 그리고 SQL:1999와 동일하게 중첩된 테이블을 고려한다. 참조 타입은 스테레오타입 속성처럼 고려된다. 이유는 배열처럼 SQL 스키마에 정의할 수 없기 때문이다. 스테레오타입 속성과 스테레오타입 클래스를 정의하도록 한다. 배열 타입이 스키마에서 명백하게 정의될 때 스테레오타입 클래스를 사용한다.

확장 전제조건으로는 정의된 관계형 데이터베이스 모델을 통하여 확장을 고려한다[4,18].

**SQL:1999 스테레오타입**

**구조화된 타입**
메타모델 클래스 : 클래스
설명 : <<udt>>는 새로운 사용자 정의 데이터 타입의 대표한다.
아이콘 : 없음
제약조건 : 값 타입을 정의하는데 사용할 수 있다.
태크값 : 없음

**타입 테이블**
메타모델 클래스 : 클래스
설명 : <<persistent>>로 정의한다. 이것은 구조화된 데이터 타입의 테이블로 정의해야 하는 데이터베이스 스키마의 클래스를 대표한다.
아이콘 : 

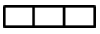
제약조건 : 타입 테이블은 테이블 타입인 구조화된 타입 정의를 의미한다.
태그값 : 없음

**구성**
메타모델 클래스 : 연관
설명 : <<composes>> 연관은 그것을 사용하는 클래스와 사용자가 정의한 데이터 타입 <<udt>>와 조인하는 특별한 관계이다.
이는 단방향 관계이다. 연관의 방향은 사용자가 정의한 타입을 사용하는 클래스의 끝 부분에 화살표로 표시된다.
아이콘 : 없음
제약조건 : 오직 <<udt>>클래스와 <<persistent>>클래스 조인을 위해 사용될 수 있다.
태그값 : 없음

**참조 타입**
메타모델 클래스 : 속성
설명 : <<ref>>는 일부 <<persistent>>클래스에 대한 연결을 대표한다.

아이콘 : 
제약조건 : <<ref>> 속성은 오직 <<persistent>>클래스를 참조할 수 있다.
태그값 : <<persistent>> 클래스는 이것을 참조한다.

**배열**
메타모델 클래스 : 속성
설명 : <<array>>은 인덱스와 제한 컬렉션 타입을 대표한다.

아이콘 : 
제약조건 : <<array>>의 요소는 <<array>>타입을 제외한 모든 데이터 타입일 수 있다.
태그값 : 배열의 기본 유형, 요소의 수

**행 타입**
메타모델 클래스 : 속성
설명 : <<row>>타입은 요소의 속성을 대표하고 각각 다른 데이터 타입일 수 있다.
아이콘 : 
제약조건 : 메소드를 가지지 않는다.
태그값 : 각 요소를 위한 이름과 데이터 타입이다.

**재 정의된 메소드**
메타모델 클래스 : 메소드
설명 : <<redef>>메소드는 자식 클래스에 의해 다시 구현되는 상속된 메소드이다.
아이콘 : 없음
제약조건 : 없음
태그값 : 데이터 타입과 메소드 매개변수의 목록이다. 데이터 타입은 메소드에 의해 반환된다.

(표 2) 객체-관계형 데이터베이스 설계를 위한 UML 확장(계속)

**지연 메소드**
메타모델 클래스 : 메소드
설명 : <<def>> 메소드는 자식 클래스에서 실행을 지연시키는 메소드이다.
아이콘 : 없음
태그값 : 데이터 타입과 메소드 매개변수의 목록이다. 데이터 타입은 메소드에 의해 반환된다.

**Oracle11g 스테레오타입**

**객체 타입**
메타모델 클래스 : 클래스
설명 : <<udt>>는 새로운 사용자 정의 데이터 타입을 허용한다. 이것은 SQL:1999에서 구조화된 타입에 해당된다.
아이콘 : 없음
제약조건 : 값 타입 정의하는데 사용될 수 있다.
태그값 : 없음

**객체 테이블**
메타모델 클래스 : 클래스
설명 : <<persistent>>처럼 정의된다. 객체 타입의 테이블로 정의되어야 하는 데이터베이스 스키마의 클래스를 나타낸다. 이것은 SQL:1999에서 타입 테이블에 해당된다.
아이콘 : 

제약조건 : 타입 테이블은 테이블의 어떤 타입인 구조화된 타입의 정의를 의미한다.
태그값 : 없음

**구성**
메타모델 클래스 : 연관
설명 : <<composes>> 연관을 사용하는 클래스와 사용자가 정의한 데이터 타입 <<udt>>와 조인하는 특별한 관계이다. 이는 단방향 관계이다. 연관의 방향은 사용자가 정의한 타입을 사용하는 클래스의 끝 부분에 화살표로 표시된다.
아이콘 : 없음
제약조건 : 오직 <<udt>>클래스와 <<persistent>>클래스 조인을 위해 사용될 수 있다.
태그값 : 없음

**참조 타입**
메타모델 클래스 : 속성
설명 : <<ref>>는 일부 <<persistent>>클래스에 대한 연결을 대표한다.
아이콘 : 
제약조건 : <<ref>> 속성은 오직 <<persistent>>클래스를 참조할 수 있다.
태그값 : <<persistent>> 클래스가 참조한다.

**배열**
메타모델 클래스 : 속성/클래스
설명 : <<array>>은 인덱스와 제한 컬렉션 타입을 대표한다. 이것은 SQL:1999에서 배열 타입에 해당된다.

아이콘 : 
제약조건 : <<array>>의 요소는 다른 컬렉션 타입을 제외한 모든 데이터 타입일 수 있다.
태그값 : 배열의 기본 유형, 요소의 수

**중첩된 테이블**
메타모델 클래스 : 클래스
설명 : <<nt>>는 비인덱스와 무제한 컬렉션 타입을 나타낸다.
아이콘 : 
제약조건 : <<nt>>의 요소는 다른 컬렉션 타입을 제외한 모든 데이터 타입일 수 있다.
태그값 : 중첩된 테이블의 기본 유형

**규칙**
각 <<udt>>와 <<array>> 또는 <<nt>>클래스는 어느 클래스와 함께 <<composes>>연관에 의해 조인되어야 한다. <<persistent>>클래스에서 <<ref>>속성은 다른 클래스와 연관을 의미한다. 컬렉션 속성을 포함하는 <<persistent>>클래스는 <<persistent>>의 객체와 <<ref>>의 클래스 구성 요소인 두 클래스 사이의 연관을 의미한다. 각 <<persistent>>클래스는 객체-관계형 모델 SQL:1999에서 확장을 포함하는 구조화된 타입에 대응된다. 확장은 타입 테이블로 각 <<persistent>>클래스는 Oracle11g에서 확장을 포함하는 객체 타입에 대응된다. 확장은 객체 타입의 테이블로 객체 타입과 확장은 <<persistent>>클래스처럼 UML 확장을 대표한다.

설명했던 확장 메커니즘을 이용한다. 확장 메커니즘은 스테레오타입과 태그값 그리고 제약조건을 통해 새로운 타입 생성을 가능하게 하는 것이다.

UML 확장에 있어서 스테레오타입과 태그값 그리고 제약조건의 설명 그리고 잘 정의된 집합은 의미적으로 내용이 일치하는지 확인해야 한다[17]. 각각의 스테레오타입을 위해서 스테레오타입이 되는 기본 요소의 의미를 지정해야 한다[9]. 이러한 스테레오타입 요소를 시각적으로 사용하기 원한다면 스테레오타입을 위한 새로운 아이콘을 정의해야 한다[8]. (표 2)는 객체-관계형 데이터베이스와 함께 응용시스템 설계를 가능하게 하는 스테레오타입 집합과 테그값 그리고 제약조건 등 UML 확장에 대한 내용을 다룬다. 확장은 객체-관계형 모델 SQL:1999와 구현 제품은 Oracle 11g이다.

(표 2)의 확장은 객체-관계형 모델 SQL:1999와 구현 제품으로는 Oracle11g를 사용한다. 객체-관계형 모델의 새로운 버전에 따라서 수정되어야 한다. 또한 다른 제품을 사용하기를 원한다면 이 확장을 적용시켜야 한다. 예를 들어 Informix 경우에 집합, 멀티집합 그리고 컬렉션 타입 리스트를 위해 새로운 확장을 정의해야 한다.

그리고 객체-관계형 데이터베이스 확장에 있어서 집합과 복합 그리고 연관에 대한 부분도 고려해야 한다[12,13].

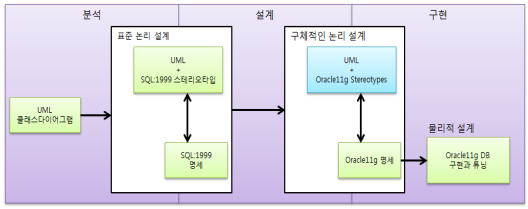
## 4. 객체-관계형 데이터베이스를 위한 설계 방법론

이번 절에서는 앞에서 제안한 UML 확장을 사용한 객체-관계형 데이터베이스 설계방법론과 그 사용 예를 보여준다.

### 4.1 객체-관계형 데이터베이스 설계 방법론

객체-관계형 데이터베이스 설계를 위한 방법론은 (그림 2)와 같다.

방법론은 분석과 설계 그리고 구현의 세 단계로 구성되어 있다. 분석 단계에서는 개념스키마 설계




(그림 2) 객체-관계형 데이터베이스 설계 방법론

를 위해 확장된 UML 클래스다이어그램을 이용한다. E-R 모델과 달리 UML은 다른 시스템 뷰 사이를 쉽게 통합하여 전체 시스템 설계를 할 수 있는 이점을 가지고 있다. 설계 단계는 다시 두 단계로 분할된다. 첫 번째 단계에서는 앞서 도출된 UML 클래스다이어그램을 객체-관계형 표준모델인 SQL:1999로 변환한다. 두 번째 단계에서는 앞 단계에서 SQL:1999로 표현된 객체-관계형 데이터베이스 설계 방안을 특정 제품인 Oracle11g 스키마로 변환한다.

첫 번째 단계에서 SQL:1999는 객체-관계형 데이터베이스 설계의 표준으로서 특정 제품에 의존하지 않고 그래픽 표기법을 통해 논리적 설계를 표현할 수 있다[19]. 그래픽 표기법은 3절에서 정의한 UML 확장을 이용한다. 두 번째 단계에서는 앞 단계에서 SQL:1999로 표현된 논리적 설계가 SQL:1999 스테레오타입으로 변환된다. 마지막으로 구현의 단계는 물리적 설계 작업을 포함한다. 이 단계에서는 앞 단계에서 얻어진 스키마를 개선한다. 특정한 응용시스템에 따라서 반응시간과 저장공간 개선을 위해 수정작업을 해야 한다.

관계형 데이터베이스 방법론에서 개념 스키마를 물리적 스키마로 변환시키는 일부 규칙을 제안하는데, 유사한 방법으로 본 논문에서 제시하는 객체-관계형 데이터베이스 설계 방법론은 특정 제품인 Oracle11g 스키마로 변환시키는 방법을 제시한다. 이 방법은 (표 3)에서 요약한다[20,21].

### 4.2 설계 방법론 적용 예

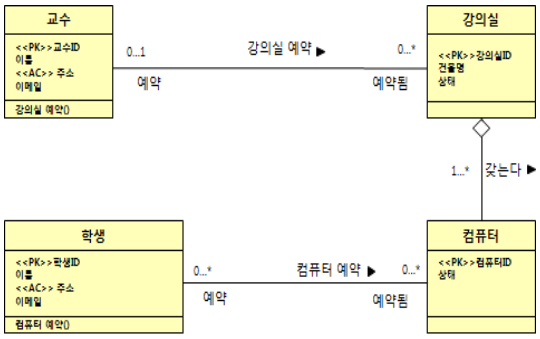
이번 절에서는 객체-관계형 데이터베이스 설계 방법론을 적용한 응용시스템에 대하여 다룬다. 그

(표 3) 객체-관계형 데이터베이스 설계를 위한 UML 확장

| UML | SQL:1999 | Oracle11g |
|---|---|---|
| Class | Structured Type | Object Type |
| Class extension | Typed table | Table of Object Type |
| Attribute | Attribute | Attribute |
| Multivalued | ARRAY | VARRY |
| 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 |
| Many-To-Many | ARRAY/ARRAY | Nested Table/Nested Table |
| Aggregation | ARRAY | Nested Table |
| Generalisation | Types/Typed Tables | Oracle cannot directly represent The generalization concept |

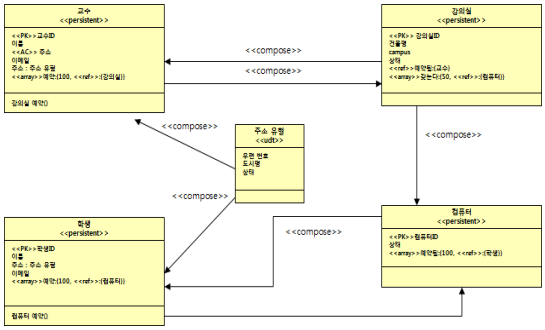
리고 객체-관계형 데이터베이스 설계를 위해 스테레오타입 정의의 예를 보여준다. 응용시스템 개발을 위해 (그림 2)에서 세 단계로 구성된 방법론을 적용하였다.

분석 단계 : (그림 3)은 UML 클래스다이어그램을 이용한 개념적 설계를 보여준다.




(그림 3) UML의 개념 설계

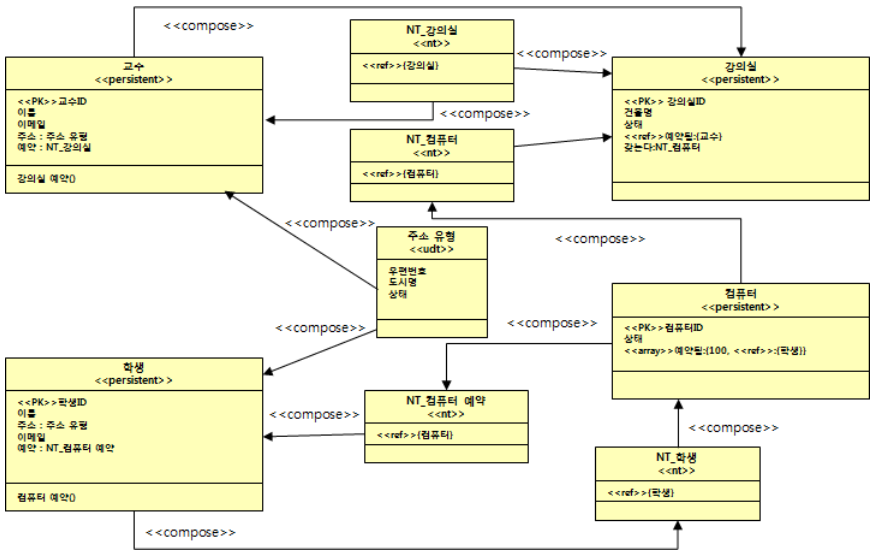
분석-설계 단계 : (그림 4)는 객체-관계형 표준 모델인 SQL:1999를 위한 UML 확장으로 구성된 그래픽 표기법과 표준 논리적 설계를 보여준다.




(그림 4) SQL:1999의 논리적 설계

설계-구현 단계 : (그림 5)는 제안된 표기법과 특정 제품인 Oracle11g의 논리적 설계를 보여준다.

그래픽 표기법은 구현 제품인 Oracle11g 스테레오타입과 객체-관계형 모델 SQL:1999 스테레오타입으로 대체된다. 그리고 논리적 설계를 바탕으로 (그림 6)과 (그림 7)에서처럼 구현 제품인 Oracle11g 스키마로 변환된다.




(그림 5) Oracle11g의 논리적 설계

```
CREATE OR REPLACE TYPE 주소유형 AS OBJECT
( 우편번호 NUMBER
, 도시명 VARCHAR(20)
, 상태 VARCHAR(10))
/
CREATE OR REPLACE TYPE 컴퓨터 AS OBJECT
(컴퓨터ID VARCHAR(5)
, 상태 VARCHAR(2)
, 예약됨 NT_학생)
/
CREATE OR REPLACE TYPE NT_컴퓨터
AS TABLE OF REF 컴퓨터
/
CREATE OR REPLACE TYPE NT_컴퓨터 예약
AS TABLE OF REF 컴퓨터
/
CREATE OR REPLACE TYPE 학생 AS OBJECT
( 학생ID VARCHAR(5)
, 이름 VARCHAR(50)
, 주소 주소 유형
, 이메일 VARCHAR(100)
, 예약 NT_컴퓨터 예약)
/
CREATE OR REPLACE TYPE NT_학생
AS TABLE OF REF 학생
/
```

(그림 6) Oracle11g 코드1

```
CREATE OR REPLACE TYPE 교수 AS OBJECT
( 교수ID VARCHAR(5)
, 이름 VARCHAR(50)
, 이메일 VARCHAR(100)
, 주소 주소 유형
, 예약 NT_강의실)
/
CREATE OR REPLACE TYPE 강의실 AS OBJECT
(강의실ID VARCHAR(5)
, 건물명 VARCHAR(50)
, 상태 VARCHAR(2)
, 예약됨 REF 교수
, has    NT_컴퓨터)
/
CREATE OR REPLACE TYPE NT_강의실
AS TABLE OF Ref 강의실
/
/* 재컴파일 */
CREATE OR REPLACE TYPE 컴퓨터 AS OBJECT
(컴퓨터ID VARCHAR(5)
, 상태 VARCHAR(2)
, 예약됨 NT_학생)
/
CREATE OR REPLACE TYPE 교수 AS OBJECT
( 교수ID VARCHAR(5)
, 이름 VARCHAR(50)
, 이메일 VARCHAR(100)
, 주소 주소 유형
, 예약 NT_강의실)
/
/* 테이블 생성 */
CREATE TABLE T강의실 OF 강의실
(PRIMARY KEY (강의실ID))
NESTED TABLE has STORE AS 컴퓨터 테이블;
CREATE TABLE T컴퓨터 OF 컴퓨터
(PRIMARY KEY (컴퓨터ID))
NESTED TABLE reserved_by STORE AS 학생 테이블;
CREATE TABLE T학생 OF 학생
(PRIMARY KEY (학생ID))
NESTED TABLE reserves STORE AS 컴퓨터 테이블;
CREATE TABLE T교수 OF 교수
(PRIMARY KEY (교수ID))
NESTED TABLE reserves STORE AS 강의실 클래스;
```

(그림 7) Oracle11g 코드2

## 5. 결 론

본 논문에서는 스테레오타입과 태그값 그리고 제약조건을 이용한, UML 클래스다이어그램 확장에 기반하여 객체-관계형 데이터베이스를 위한 설계 방법론을 개발하였다. 아울러 확장된 UML 클래스다이어그램을 객체-관계형 데이터베이스 스키마로의 변환을 위한 가이드라인을 제시하였다. 제안한 방법론은 분석과 설계 그리고 구현의 세 단계로 구성되어 있다. 개념스키마 설계를 위해 UML 클래스 다이어그램을 이용하여 객체-관계형 표준 모델인 SQL:1999와 구현 제품인 Oracle11g를 사용했다. 아울러 사례 연구로 제안한 설계방법론을 컴퓨터 교실 예약시스템에 적용했다.

객체-관계형 DBMS는 존재하지만 객체-관계형 데이터베이스 설계를 위한 일관된 방법론은 제시되지 않았다. 따라서 본 논문에서 제시하는 방법론을 적용함으로써 UML 클래스다이어그램 확장에 기반한 객체-관계형 데이터베이스 설계와 개념스키마를 논리스키마로의 변환까지의 일관된 방법으로 객체-관계형 데이터베이스 설계를 가능하게 되었다.

## 참 고 문 헌


[1] Bertino and Marcos, “Object Oriented Database Systems”, In Advanced Databases: Technology and Design, O. Díaz and M. Piattini (Eds. Artech House), 2000.

[2] Stonebraker and Brown, “Object-Relational DBMSs”, Traking the Next Great Wave, Morgan Kauffman, 1999.

[3] Blaha and Premerlani, “Object-Oriented Modeling and Design for Database Applications”, Prentice Hall, 1998.

[4] C. Kovács and P. Van Bommel, “Conceptual modelling-based design of objectoriented databases”, Information and Software Technology, Vol. 40, No. 1, pp. 1-14, 1998.

[5] Muller, “Database Design for Smarties”, Morgan Kaufmann, 1999.

[6] Silva and Carlson, “MOODD, a method for object-oriented database design”, Data & Knowledge Engineering, Vol. 17, pp.159-181, 1995.

[7] Ullman and Widom, “A First Course in Database Systems”, Prentice-Hall, 1997.

[8] Booch, Rumbaugh and Jacobson, “The Unified Modeling Language User Guide”, Addison Wesley. 1999.

[9] 김영규, 양해술, 최형진, “객체지향 환경에서 소프트웨어 생산성 향상을 위한 프레임워크 모델”, 한국산학기술학회 논문지, 제 9권, 제6호, 2008.

[10] 이현우, 박찬석, 고석하, “객체지향 개발 프로세스에서 비즈니스 프로세스 모델과 소프트웨어 아키텍처의 관계 연구를 위한 접근 방법의 제언”, Entrue Journal of Information Technology, Vol. 8, No. 2, 2009.

[11] E. Marcos, B. Vela and J. M. Cavero, “Extending UML for Object-Relational Database Design”, Kybele Research Group, Rey Juan Carlos University, Madrid. UML 2001, LNCS 2185, pp. 225-239, 2001.

[12] Eric Pardede, J. Wenny Rahayu, David Taniar, “Mapping Methodos and Query for Aggregation and Association in Object-Relational Database using Collection”, Proceedings of the International Conference on Information Technology : Cooding and Computing(ITCC’04), 2004.

[13] 김인철, 김영웅, “확장된 UML 클래스 다이어그램을 이용한 객체 관계형 데이터베이스 설계 기법”, 정보처리학회 추계학술발표회, 제 12권, 제2호, pp.91-94, 2005.

[14] Eisenberg and Melton, “SQL:1999, formerly known as SQL3”, ACM SIGMOD Record, Vol. 28, No. 1, pp. 131-138, 1999.

[15] Mattos, N. M, “SQL:1999, SQL/MM and SQLJ: An Overview of the SQL Standards”, IBM Database Common Technology, 1999.
[16] Oracle Corporation, Oracle11g SQL Reference Release 2 (11.2), In: www.oracle.com. 2011.
[17] J. Conallen, “Building Web Application with UML”, Addison-Wesley, 2000.
[18] Ambler, Persistence Modeling in the UML, In: http://www.sdmagazine.com/articles/1999/0008/0008 q/0008q.htm, 1999.
[19] Atzeni, Ceri, Paraboschi and Torlone, “Database Systems. Concepts, Languages and Architectures”, McGraw-Hill, 1999.
[20] E. Marcos, B. Vela and J. M. Cavero, “A Methodology for Object-Relational Database Design Using UML”, submitted to 12th International Conference and Workshop on Database and Expert Systems and Applications, 2001.
[21] E. Marcos, B. Vela and J. M. Cavero, “Aggregation and Composition in Object-Relational Database Design”, submitted to Fifth East European Conference on Advances in Databases and Information Systems, 2001.

## ◐ 저 자 소 개 ◑

**주 경 수**
1993년 고려대학교 대학원 전산학과 졸업(박사)
1986년~현재 순천향대학교 컴퓨터소프트웨어공학과 교수
관심분야 : Database System, XML, System Integration, Object Oriented System.
E-mail : gsoojoo@sch.ac.kr

**조 도 형**
2010년 순천향대학교 컴퓨터학과 졸업(학사)
2010년~현재 순천향대학교 컴퓨터소프트웨어공학과 석사과정
관심분야 : Database System, Object Oriented System, UML.
E-mail : jhodohyung@sch.ac.kr