CHAPTER

7

데이터 모델링 및 분석

개요 및 목적

이번 장에서는 가장 보편적인 데이터 모델링 도구인 개체 관계성도의 활용 방법에 대하여 학습한다. 개체 관계성도는 시스템에 의하여 파악되고 저장되어야 하는 데이터를 기록하는 수단으로 그 데이터가 어떤 것인지 어떻게 사용되는 지와 무관하게 즉, 특정 입력이나 출력이나 처리과정과 독립적으로 다룬다. 여러분은 또한 정규화라고 하는 데이터 분석 기법에 대해서도 학습하게 되는데 이것은 '좋은' 데이터 모델을 작성하도록 해주는 기법이다. 여러분은 시스템 분석 도구 및 기법으로서 데이터 모델링과 데이터 분석을 배우게 되며 다음과 같은 능력을 가지게 된다.

- 시스템 모델링을 정의하고, 논리적 및 물리적 시스템 모델을 구분할 수 있다.
- 데이터 모델링을 정의하고 그것의 이득을 설명할 수 있다.
- 데이터 모델의 기본 개념과 구성을 인식하고 이해할 수 있다.
- 개체 관계성 데이터 모델을 읽고 해석할 수 있다.
- 프로젝트 진행 중 데이터 모델이 언제 작성되고 모델들은 어디에 저장되는지 설명할 수 있다.
- 개체와 관계를 발견할 수 있다.
- 개체 관계성 상황도(entity relationship context diagram)를 작성할 수 있다.
- 개체에 대한 키(key)를 발견하거나 생성하며, 키 기반 다이어그램을 작성할 수 있다.
- 모든 속성이 표현된 개체 관계성도를 작성하고, 모든 데이터 구조와 속성을 저장소(repository)나 사전(encyclopedia)에 기술할 수 있다.
- 데이터베이스를 안정적이지 못하고 유연하지 못하며, 규모 변경을 어렵게 만들 수 있는 불순물들을 논리적 데이터 모델에서 제거하기 위하여 정규화를 실시할 수 있다.
- 데이터 요구사항을 비즈니스 운영 위치에 대응시키는 데 유용한 도구를 설명할 수 있다.

개요

사운드스테이지 회원서비스 시스템 프로젝트가 요구사항 분석에서 논리적 설계단계로 이동함에 따라, 그들의 방법론에 따른 첫 번째 과업이 새로운 시스템에 대한 데이터 요구사항 분석이다. 밥 마르티네즈는 대학 시절 존경하던 교수님께서 항상 "'올바른' 데이터를 얻어낸다면, 시스템은 현재의 모든 요구사항을 훌륭하게 지원할 것이고 심지어는 사용자가 아직까지 인지하고 있지 못하는 요구사항도 지원할 수 있을 것이다. 만약 데이터를 잘못 얻게 된다면, 오늘뿐 아니라 내일 그리고 영원히 요구사항을 충족시키지 못해서 목에 가시와 같은 고통을 겪게 될 것이다"라고 말씀하시던 것을 기억해 냈다.

밥은 대학 시절 데이터베이스 수업을 즐겁게 수강하였고 항상 좋은 성적을 얻었다. 물론 회원서비스 시스템은 그가 학교에서 해보았던 어떤 데이터 프로젝트보다도 크고 상세한 것이다. 다행스럽게도 그는 초기 작업을 위하여 시스템의 과거 버전의 데이터베이스를 갖고 있으며, 과거 시스템의 양식과 보고서도 있고, 사용자 인터뷰를 통한 의견 그리고 요구사항 분석 단계에서 생성된 유스케이스도 가지고 있다. 샌드라는 밥에게 전원이 논리적 데이터 모델 개발로 뛰어들 때 책임을 맡아주길 바랐고 그는 그녀에게 깊은 인상을 주기로 결심했다.

데이터 모델링이란 무엇인가?

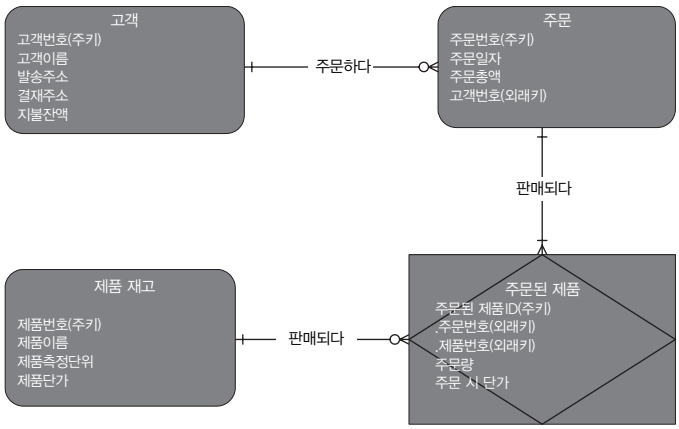
시스템 모델은 시스템 개발에서 중요한 역할을 한다. 이번 장에서는 데이터베이스를 위한 비즈니스 요구사항 정의 기법으로서 데이터 모델링을 소개하고자 한다. 데이터 모델이 결국은 데이터베이스로 구현되기 때문에, 데이터 모델링은 종종 **데이터베이스 모델링**(data modeling)이라고 한다.

데이터 모델링(data modeling)
시스템 데이터를 문서화하고 조직화하는 기법으로 종종 데이터베이스 모델링이라고 불린다.

[그림 7-1]은 개체관계성도(entity relationship diagram) 혹은 ERD라고 불리는 데이터 모델의 단순한 예이다. 이 다이어그램은 다음과 같은 비즈니스 상황을 담고 있다.

- 우리는 고객, 주문, 그리고 제품 재고에 관한 데이터를 저장할 필요가 있다.
- 고객 번호 값은 한 분의 고객을 유일하게 구별할 수 있다. 주문 번호 값은 하나의 주문을 유일하게 판별할 수 있다. 제품 번호 값은 하나의 제품 재고를 유일하게 구분할 수 있다.
- 고객에 대하여 고객 이름, 발송 주소, 결재 주소, 그리고 지불 잔액의 정보가 필요하다. 주문에 대해서는 주문 일자, 주문 총 비용에 대한 정보가 필요하다. 제품 재고를 위해서 제품 이름, 제품 측정 단위, 그리고 제품 단가에 대한 정보가 필요하다.
- 고객은 현재 혹은 최근에 주문을 안 했을 수도 있고, 한 건 혹은 여러 건의 주문을 했을 수도 있다.
- 하나의 주문은 한 명의 고객에 의해서만 내려질 수 있고, 고객 번호(주문에 저장된 값)가 그 고객을 구별해 준다.
- 하나의 주문은 한 개 혹은 여러 개의 제품 재고를 포함할 수 있다. 그러므로 하나의 주문은 반드시 최소 한 개의 주문된 제품을 포함하고 있어야 한다.

그림 7-1
개체 관계성도 샘플



- 제품 재고는 주문된 제품으로 하나도 안 팔릴 수도 있고 한 개 혹은 여러 개가 팔릴 수도 있다.
- 하나의 주문된 제품은 하나의 주문에 포함된 하나의 제품 재고를 구분할 수 있다. 주문 번호(주문된 제품의)는 주문을 판별하고, 제품 번호(주문된 제품의)는 제품 재고를 판별한다. 두 가지를 합하여 유일하게 주문된 제품을 구별할 수 있다.
- 주문된 제품을 위해서 주문량과 주문 시 단가를 알아야 한다.

이 장을 학습한 이후에는 여러분들은 데이터 모델을 해독할 수 있고 작성할 수 있을 것이다.

데이터 모델링을 위한 시스템 개념

개체관계성도(ERD, Entity Relationship Diagram)
데이터를 개체와 데이터에 의하여 묘사되는 관계로 표현하기 위하여 여러 기호를 사용하는 데이터 모델

데이터 모델링을 위한 표현법에는 여러 가지가 있다. 실제 모델은 개체관계성도(ERD, Entity Relationship Diagram)라고 불리는데, 그 이유는 데이터를 개체와 데이터에 의하여 묘사되는 관계로 표현하기 때문이다. ERD를 위한 여러 가지 표기법이 있다. 대부분은 창안자들(예를 들면, Chen, Martin, Bachman, Merise 등)에 의하여 개발되거나 공표된 표준(예를 들면 IDEF1X)에 의하여 개발되었다. 이러한 데이터 모델링 '언어'들은 일반적으로 동일한 기본 개념과 구성물들을 지원한다. 우리는 Martin(정보 공학)의 표기법을 채용하기로 하는데, 이것이 폭 넓게 사용되고 CASE 도구의 지원도 많기 때문이다.

이제 모든 데이터 모델에 내재된 기본 개념을 살펴보기로 하자.

개체

모든 시스템은 데이터를 포함하고 있다.— 보통 아주 많은 데이터가 있다. 데이터는 '실체'를 묘사하고 있다. 학교 시스템을 생각해보자. 학교 시스템은 학생, 선생님, 과목, 교실 등의 실체를 묘사하는 데이터를 포함하고 있다. 이러한 실체에 대하여 그 실체의 예를 묘사하는 데이터를 상상해 보는 것은 어렵지 않다. 예를 들어 어떤 학생을 묘사하는 데이터는 이름, 주소, 전화번호, 생년월일, 성별, 인종, 전공, 평점 등의 데이터를 포함할 것이다.

비슷한 실체의 예들을 모두 추상적으로 표현할 수 있는 개념이 필요하다. 이러한 개념을

개체(entity)라고 한다. 개체는 데이터를 저장해야 하는 비즈니스적 필요가 있는 어떤 것이다. 시스템 모델링에서 각각의 추상적인 개념에 모양을 할당하는 것이 유용하다는 것을 알게 된다. 이 책에서는 개체를 표현하기 위하여 네 모서리가 둥근 사각형을 사용하고자 한다. 이 모양은 이름을 가진 개체의 모든 예들을 표현한다. 예를 들면, 학생 개체는 시스템 내의 모든 학생들을 나타낸다. 개체는 특정 개체들의 클래스를 식별하며 다른 개체들과 구별된다.
개체들의 범주는 다음과 같다.

개체(entiry)
수집하고 데이터로 저장할 필요가 있는 사람, 장소, 객체, 이벤트, 혹은 개념 등의 클래스

개체

- 사람 : 대리인, 계약자, 고객, 부서, 사업 부문, 근로자, 강사, 학생, 공급자. 사람 개체 클래스는 개인, 집단, 혹은 조직을 표현할 수 있음에 주의하기 바란다.
- 장소 : 판매 지역, 건물, 방, 지사 사무실, 캠퍼스
- 객체 : 책, 기계, 부분품, 제품, 원자재, 소프트웨어 라이선스, 소프트웨어 패키지, 도구, 차량 모델, 차량. 객체 개체는 실제 객체(특정 소프트웨어 라이선스와 같이) 혹은 객체 유형에 대한 사양(서로 다른 소프트웨어 패키지 사양과 같은)을 표현할 수 있음에 주의하기 바란다.
- 이벤트 : 응모, 수상, 취소, 등급, 비행편, 청구서, 주문, 등록, 갱신, 요청, 예약, 판매, 여행
- 개념 : 계정, 시간 구획, 채권, 강좌, 펀드, 자격, 주식,

개체와 그 개체의 인스턴스(instance)를 구별하는 것이 중요하다. **개체 인스턴스**(entity instance)는 개체의 하나의 존재를 의미한다. 예를 들면, 학생 개체는 Mary, Joe, Mark, Susan, Cheryl 등의 여러 인스턴스를 갖고 있다. 데이터 모델링에서 개별 학생들이 데이터의 비슷한 부분으로 묘사되는 것으로 인식하기 때문에 개별 학생들을 중요하게 취급하지는 않는다.

개체 인스턴스(entity instance)
개체의 하나의 존재

속성

개체가 데이터로 저장하고자 하는 어떤 것이라면, 주어진 개체의 개별 예에 대하여 저장하고자 하는 데이터의 어느 특정 부분을 식별할 필요가 있다. 데이터의 이러한 부분을 **속성**(attribute)이라고 한다. 이 절의 앞부분에서 언급한 바와 같이, 학생 개체의 각 예들은 이름, 주소, 전화번호, 생년월일, 성별, 인종, 전공, 평점, 기타 등의 속성에 의하여 표현될 수 있다.

개체의 모양 내부에 속성의 이름을 기록하는 방식으로 개체가 속성을 포함하도록 개체의 그래픽 추상화를 확장할 수 있다.

일부 속성은 **복합 속성**(compound attribute)이라고 불리는 상위 속성으로 논리적인 집단화가 가능하다. 예를 들면, 학생의 이름은 실제적으로는 성과 이름이 결합된 복합 속성이다. 옆에서 복합 속성을 표현하는 방법 한 가지를 제시하였다. 복합 속성에 속하는 기본 속성들은 속성 앞에 마침표를 찍어서 표현했음에 주의하길 바란다.

속성(attribute)
개체의 표현적 성질 혹은 특성. 동의어로 엘리먼트(elemnt), 특성(property), 필드(field) 등이 있다

복합 속성(compound attribute)
다른 속성들로 구성된 속성, 다른 데이터 모델링 언어에서 사용하는 동의어는 많이 있다. - 결합 속성(concatenated attribute), 합성 속성(composite attribute), 데이터 구조체(data structure)

■ **도메인** 시스템을 분석할 때, 속성이 가질 수 있는 값으로써 정당하거나 비즈니스적으로 타당한 값들을 정의할 필요가 있다. 속성 값들은 데이터 타입, 도메인, 디폴트 값의 세 가지 특성치로 정의된다.

도메인(domain)
속성이 어떤 값을 정당하게 받아들 수 있는 가를 정의하는 특질

속성의 **데이터 타입**(data type)은 어떤 유형의 데이터가 저장될 수 있는가를 정의한다. 시

속성과 복합속성

데이터 타입(data type)
속성에 어떤 데이터 유형이 저장될 수 있는가를 지정하는 속성의특질

스템분석과 요구사항 분석의 목적에 비추어, 비즈니스 속성에 대한 논리적(비 기술적인) 데이터 타입을 선언해 두는 것이 유용하다. 혼란을 줄이기 위하여 우리는 [표 7-1]과 같은 논리적 데이터 타입을 사용하고자 한다.

속성의 데이터 타입은 자신의 도메인을 포함한다. 속성의 도메인은 속성이 어떤 값을 정당하게 받아들 수 있는 가를 정의한다. [표 7-2]는 각 데이터 타입에 대해서 논리적 도메인이 어떻게 표현될 수 있는가를 예시하고 있다.

표 7-1 대표적인 속성의 논리적 데이터 타입

논리적 데이터 타입	비즈니스상의 논리적 의미
수(number)	수, 정수 혹은 실수
문자(text)	숫자를 포함할 수 있는 문자열. 숫자가 문자 속성 내에 포함된다면, 다른 숫자와 산술계산이나 비교 연산을 수행할 수 없다.
메모(memo)	문자와 동일하지만 크기가 결정되지 않았음. 어떤 비즈니스 시스템은 주어진 데이터베이스 레코드에 충분히 긴 메모를 저장하는 능력을 요구할 수 있다.
날짜(date)	다양한 유형의 날짜
시간(time)	다양한 유형의 시간
예/아니오(yes/no)	두 값 중의 하나를 가질 것이라고 예상되는 속성
값 집합(value set)	값들의 유한 집합. 대부분의 경우, 코드 체계가 설정되어 있다(예를 들어, FR = 1학년, SO = 2학년, JR = 3학년, SR = 4학년)
영상(image)	그림이나 영상

표 7-2 대표적인 논리적 데이터 타입에 대한 논리적 도메인

데이터 타입	도메인	예
수(number)	정수는 범위를 지정한다. {최솟값-최댓값} 실수는 범위와 정밀도를 정의한다. {최솟값. 정밀도-최댓값-정밀도}	 {10 - 99} {1.000 - 799.999}
문자(text)	TEXT(속성의 최대 크기) 실제 값은 통상 무한대이지만, 사용자들이 대략의 제한을 설정한다.	TEXT(30)
메모(memo)	크기나 내용물에 대한 논리적 제한이 없다. 날짜(date)MMDDYYYY 양식에 대한 여러 변형. 2000년 문제 해결을 위해 YY형태는 피한다. 양식 표현을 위한 문자는 저장되지 않으므로 하이픈이나 사선은 포함하지 않는다.	MMDDYYYY MMYYYY YYYY
시간(time)	오전/오후 표기 HHMMT 24시간 표기 HHMM	HHMMT HHMM
예/아니오(yes/no)	{YES, NO}	{YES, NO}, {ON, OFF}
값 집합(value set)	{value #1, value #2, … , value #n} 혹은 {코드와 의미를 정리한 표}	{FRESHMAN, SOPHOMORE, JUNIOR, SENIOR} {FR = FRESHMAN, SO = SOPHOMORE, JR = JUNIOR, SR = SENIOR}
영상(image)	없음. 그러나 영상에 대한 알려진 특성이 제공되면 설계자에게 유용함.	

표 7-3 속성들의 허용 가능한 디폴트 값

디폴트 값	해석	예
도메인 내의 적합한 값	사용자가 특정한 값을 지정하지 않는다면, 속성에 대하여 이 값을 사용한다	0 1.00 FR
NONE or NULL	사용자가 특정한 값을 지정하지 않는다면, 속성 값을 빈칸으로 남겨둔다	NONE NULL
REQUIRED or NOT NULL	속성 값으로 도메인 내의 적합한 값을 입력하는 것이 요구된다.(이것이 도메인 내에서 디폴트 값으로 사용될 만큼 공통적인 값이 없고 어떤 값인가는 반드시 입력되어야 하는 경우이다)	REQUIRED NOT NULL

마지막으로, 모든 속성들은 논리적인 **디폴트 값**(default value)을 가져야 한다. 디폴트 값은 사용자가 특정한 값을 입력하지 않을 경우에 그 속성에 기본으로 저장되는 값을 말한다. [표 7-3]은 속성들에 대한 가능한 디폴트 값을 보여주고 있다. 공값 허용 안 됨(not null)은 속성의 개별 인스턴스들이 반드시 값을 가져야 함을 의미하고 공값 허용(null)은 속성의 어떤 예들은 선택적이어서 값을 갖지 않을 수 도 있음을 나타낸다.

디폴트 값(default value)
사용자에 의하여 특정 값이 지정되지 않는 경우에 저장 되는 값

■ **식별** 하나의 개체는 여러 개 아마도 수천 혹은 수백만 개의 인스턴스를 가질 수 있다 . 한 개 혹은 여러 개의 속성 값에 기반하여 각각의 인스턴스를 유일하게 식별해야 할 필요가 있다. 따라서 모든 개체는 키(key)를 가져야 한다. 예를 들면, 학생 개체의 인스턴스들은 학번이라는 속성에 의하여 유일하게 식별될 수 있는데 동일한 학번을 갖는 학생은 존재하지 않기 때문이다.

종종 하나 이상의 속성이 있어야만 개체 예들을 유일하게 식별할 수 있는 경우도 있다. 여러 개의 속성으로 구성된 **키**(key)를 **복합키**(concatenated key)라고 한다.

키(key)
각 개체 예들에 대하여 유일한 값을 부여할 수 있는 하나의 속성 혹은 속성의 집단. 식별자(identifier)라고도 한다.

예를 들어, 비디오 가게의 개별 DVD 개체 인스턴스들은 타이틀 번호와 복사본 번호의 복합에 의하여 유일하게 식별될 수 있을 것이다. 가게에는 동일한 타이틀에 대하여 여러 개의 복사본을 가지고 있을 수 있기 때문에 타이틀 번호 만으로는 부족하다. 또한 모든 타이틀은 복사본 번호를 갖고 있기 때문에 복사 번호만으로도 부족하다. 특정 테이프를 식별하기 위해서는 두 가지 데이터 모두가 필요하다(예를 들어, 스타워즈 : 시스의 복수, 복사본#7). 이 교재에서는 개별 속성 뿐 아니라 그룹에도 이름을 부여할 것이다. 예를 들면, DVD에 대한 복합 키는 다음과 같이 저장될 것이다.

복합키(concatenated key)
개체의 하나의 예를 유일하게 식별할 수 있는 속성 집단. 동의어로 합성키(composite key, compound key)가 있다.

DVD ID
.TITLE NUMBER
.COPY NUMBER

많은 경우에 개체는 하나 이상의 키를 갖고 있을 수 있다. 예를 들면, 직원 개체는 주민 등록번호 혹은 회사에서 부여한 직원번호, 혹은 이메일 주소 등을 이용하여 유일하게 식별될 수 있을 것이다. 이러한 속성들을 **후보키**(candidate key)라고 한다. 후보키는 개체 인스턴스들에 대하여 '**주키**(primary key)가 될 수 있는 후보' 라는 의미이다. 이것은 종종 후보 식별자

후보키(candidate key)
개체의 주키로 사용될 수 있는 여러 키들 중의 하나. 또한 후보 식별자(candidate identifier)라고도 한다.

주키(primary key)
단일 개체 인스턴스를 유일하게 식별하도록 가장 보편적으로 사용되는 후보키

대체키(alternate key)
주키로 선정되지 않은 후보키. 동의어로는 보조키(secondary key)가 있다

부분집합화 기준 (subsetting criteria)
그 속성에 속한 유한한 값들이 모든 개체 인스턴스들을 유용한 부분 집합으로 분리할 수 있는 속성. 입력 전환(inversion entry)라고도 불림

(candidate identifier)라고도 한다(후보키는 단일 속성이거나 복합키일 수 있다). 주키는 단일 개체 인스턴스를 유일하게 식별하도록 가장 보편적으로 사용되는 후보키이다. 주키의 디폴트 값은 항상 공값 허용 않음(NOT NULL)인데 그 이유는 만약 주키가 값을 갖고 있지 않으면, 개체의 개별 인스턴스를 유일하게 식별하는 본래의 목적을 수행할 수 없기 때문이다. 주키로 선정되지 않은 후보키들은 **대체키**(alternate key)라고 한다. 일반적인 동의어로는 보조키(secondary key)가 있다. 옆에 주키와 대체키에 대한 표현 방법을 제시하였다. 모든 후보키는 주키이거나 대체키여야 한다. 그러므로 후보키에 대한 별도의 표현은 고려하지 않는다. 주키의 일부가 아닌 모든 속성들은 비주키 속성(nonkey attribute)이라고 한다.

때때로 개별 인스턴스와 반대의 개념으로 개체 인스턴스들의 부분 집합을 고려해야 할 필요가 있다. 예를 들면, 남학생과 여학생을 간단하게 구별할 수 있는 방법이 필요할 때가 있을 수 있다. **부분집합화 기준**(subsetting criteria)은 그 속성에 속한 유한한 값들이 모든 개체 인스턴스들을 유용한 부분 집합으로 분리할 수 있는 속성(혹은 복합 속성)이다. 이것은 종종 입력 전환(inversion entry)이라고 한다. 학생 개체에서, 성별이라는 속성은 학생 개체의 인스턴스를 두 개의 부분 집합인 남학생과 여학생으로 구분한다. 보통 부분 집합화 기준은 속성이 유한한(제한된 이라는 의미임) 개수의 정당한 값들을 가지고 있을 때 유용하다. 예를 들면, 평점은 그 값이 0.00에서부터 4.00까지 999개의 가능한 값들이 있기 때문에 좋은 부분 집합화 기준이라고 할 수 없다. 옆을 보면 부분 집합화 기준에 대한 표현을 알 수 있다.

키와 부분집합화 기준

관계

개념적으로 개체와 속성들은 분리되어 존재하지 않는다. 그것들이 표현하고 있는 것은 비즈니스 목표를 지원하기 위한 상호관계 혹은 다른 것에 대한 영향 등이다. 그러므로 **관계**(relationship)라는 개념을 도입하고자 한다. 관계란 하나 혹은 그 이상의 개체 간에 존재하는 자연적인 비즈니스 연관을 말한다. 관계는 개체간의 연결 혹은 개체 간에 존재하는 단순한 논리적 관련성 등을 나타낸다. 예를 들어 학생과 교과과정이라는 개체를 생각해보자. 학생과 과목을 연결하는 다음과 같은 비즈니스 연관을 만들어 볼 수 있다.

관계(relationship)
하나 혹은 그 이상의 개체간의 자연적인 비즈니스 연관

- 한 학생이 하나 혹은 다수의 과목에 수강 신청한다.
- 한 과목에는 0 명이거나, 한 명이거나, 여러 명의 학생이 수업 받고 있다.

밑줄 그은 동사 부분은 두 개체 간에 존재하는 비즈니스 관계를 정의하고 있다.

학생과 과목 간의 연관을 그림으로 표현하면 다음 [그림 7-2]와 같다. 연결된 선은 관계를 표현한다. 모든 관계들은 양방향의 의미를 내포하고 있다. 즉, 양쪽 방향 모두로 해석될 수 있다. 데이터 모델링 방법에 따라 관계에 이름을 부여하는 방식이 서로 다르다. 어떤 경우에는 양쪽의 동사 부분을 모두 포함하고 다른 경우에는 한쪽 동사부분만을 포함한다.

■ **사상수** [그림 7-2]는 또한 각 관계의 복잡도나 정도를 보여주고 있다. 예를 들어서 [그림 7-2]를 읽는 법을 안다면, 다음의 질문에 답을 할 수 있을 것이다.

- 과목의 각 인스턴스에 대응하는 학생의 인스턴스가 존재하여야 하는가? 아니오!

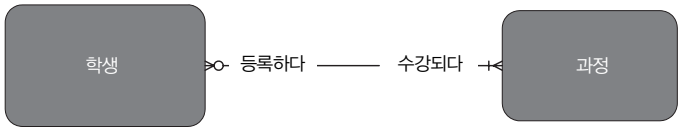


그림 7-2

관계(다대다)

- 학생의 각 인스턴스에 대응하는 과목의 인스턴스가 존재하여야 하는가? 예!
- 학생의 각 인스턴스에 대응하는 과목의 인스턴스는 몇 개인가? 다수!
- 과목의 각 인스턴스에 대응하는 학생의 인스턴스는 몇 개인가? 다수!

이러한 개념을 **사상수**(cardinality)라고 한다. 사상수는 다른 개체의 하나의 인스턴스와 관련을 맺을 수 있는 어떤 개체 인스턴스들의 최소 개수와 최대 개수를 말한다. 모든 관계들은 양방향이므로, 사상수는 모든 관계에 대하여 양방향으로 정의되어야 한다. 사상수에 대한 일반적인 그래픽 표현 방법은 [그림 7-3]에 제시되어 있다. 사상수 심벌에 대한 예는 [그림 7-2]에서 살펴볼 수 있다.

사상수(cardinality)
다른 개체의 하나의 예와 관련을 맺을 수 있는 어떤 개체 예들의 최소 개수와 최대 개수

개념적으로 사상수는 [그림 7-2]에 표현된 데이터 개체에 대한 다음의 다음 사항을 알 수 있게 해준다.

- 데이터베이스에 학생 한 명을 추가할 때, 학생은 반드시 최소한 한 과정과는 연결(연관)을 가져야 한다. 비즈니스적으로는 '학생들은 전공을 선택하지 않고는 입학할 수 없다' 는 의미를 갖고 있다(대부분의 학교는 과정의 인스턴스로 '미정' 을 포함하고 있다).
- 한 학생은 한 과정 이상에서 공부할 수 있고 학생이 선택한 모든 과정의 데이터를 저장할

사상수의 해석	최소 개체 예	최대 개체 예	그래픽 표현
정확히 하나 (일대일)	1	1	– or –
없거나 하나	0	1	
하나 혹은 다수	1	many (>1)	
없거나 하나거나 다수	0	many (>1)	
하나 이상	>1	>1	

그림 7-3

사상수 표현 방법

그림 7-4

재귀 관계

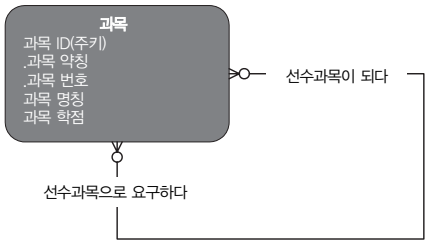
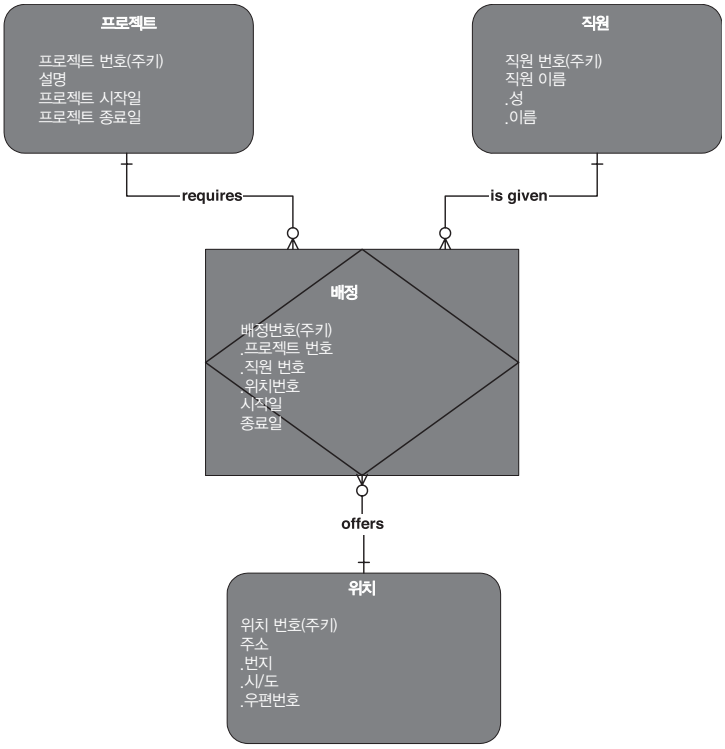


그림 7-5

3원 관계



수 있어야 한다.

- 과정에 학생들을 연결(연관)할 수 있기 이전에 과정을 삽입해야 한다. 이것은 과정은 학생을 한 명도 갖지 않을 수 있기 때문이다(즉, 그 과정에 등록한 학생이 없을 수 있다).
- 과정이 데이터베이스에 삽입된 이후에 많은 학생들이 그 과정과 연결(연관)될 수 있다.

■ **차수** 데이터 관계의 복잡도를 측정하는 또다른 기준은 **차수**(degree)이다. 관계의 차수는 그 관계에 참여하는 개체의 개수를 말한다. 우리가 살펴본 모든 관계들은 2원 관계이다(차수 = 2). 즉, 두 개의 서로 다른 개체가 관계에 참여하고 있다.

차수(degree)
그 관계에 참여하는 개체의 개수를 말한다.

관계는 동일한 개체 안에 있는 서로 다른 개체 인스턴스들 간에도 존재할 수 있는데(차수 = 1) 이것을 **재귀 관계**(recursive relationship)라고 한다. 예를 들어, 학교에서 어떤 과목은 다른 과목의 선수 과목으로 지정될 수 있으며 비슷하게, 한 과목은 여러 개의 다른 과목을 선수 과목을 지정할 수 있다. [그림 7-4]는 이러한 일대일 재귀관계의 예를 보여주고 있다.

재귀 관계(recursive relationship)
동일한 개체 안에 있는 서로 다른 개체 인스턴스들 간에 존재하는 관계

관계는 또한 두 개 이상의 서로 다른 개체 간에 존재할 수 있다. 이것을 다원 관계(N-ary

relationship)라고 한다. 3원 관계의 예를 [그림 7-5]에서 살펴볼 수 있으며 여기에는 **연관 개체**(associative entity)라고 불리는 새로운 개체 형태가 제시되어 있다. 연관 개체는 하나 이상의 다른 개체(부모라고 한다)들로부터 주키를 상속받은 개체를 말한다. 복합키의 각 부분들은 연결된 개별 개체의 예를 정확히 한 개 지적하고 있다.

연관 개체(associative entity)
하나 이상의 다른 개체들로부터 주키를 상속받은 개체

[그림 7-5]에서 연관 개체인 배정은 직원, 위치, 그리고 프로젝트를 연결시킨다. 배정의 각 인스턴스들에서 키는 어떤 직원 번호와 위치 번호 그리고 프로젝트 번호가 결합되어 배정을 이루었는가를 보여준다.

또한 [그림 7-5]에 나타난 바와 같이, 연관 개체는 자신의 비 주키 속성에 의해서 설명될 수 있다. 주키에 덧붙여서, 배정은 시작일과 종료일이라는 속성에 의하여 설명된다. 이 속성들은 직원, 위치 혹은 프로젝트를 개별적으로 설명하는 것이 아니라 이들 개체 인스턴스들 간의 관계에 대한 예를 설명하고 있다.

■ **외래키** 관계란 한 개체의 인스턴스가 다른 개체의 인스턴스와 관계되었다는 의미를 내포하고 있다. 어떤 개체에 대하여 그러한 인스턴스들을 식별할 수 있어야 한다. 연관된 특정 개체 인스턴스들을 식별할 수 있는 능력은 외래키를 설정하는 것을 포함하고 있다. **외래키**(foreign key)는 한 개체의 주키로서 관계의 인스턴스를 식별하기 위하여 다른 개체에(복사되어) 존재하는 것이다. 외래키(항상 자식 개체에 존재한다)는 항상 부모 개체에 있는 주키와 대응관계를 유지한다. [그림 7-6(a)]에서 간단한 데이터 모델에서 외래키의 개념을 예를 들어 보이고 있다. 학과의 최대 사상수가 '하나' 인 반면, 과정의 최대 사상수는 '다수' 인 점을 유

외래키(foreign key)
관계의 인스턴스를 식별하기 위하여 다른 개체 안에서 사용되는 한 개체의 주키

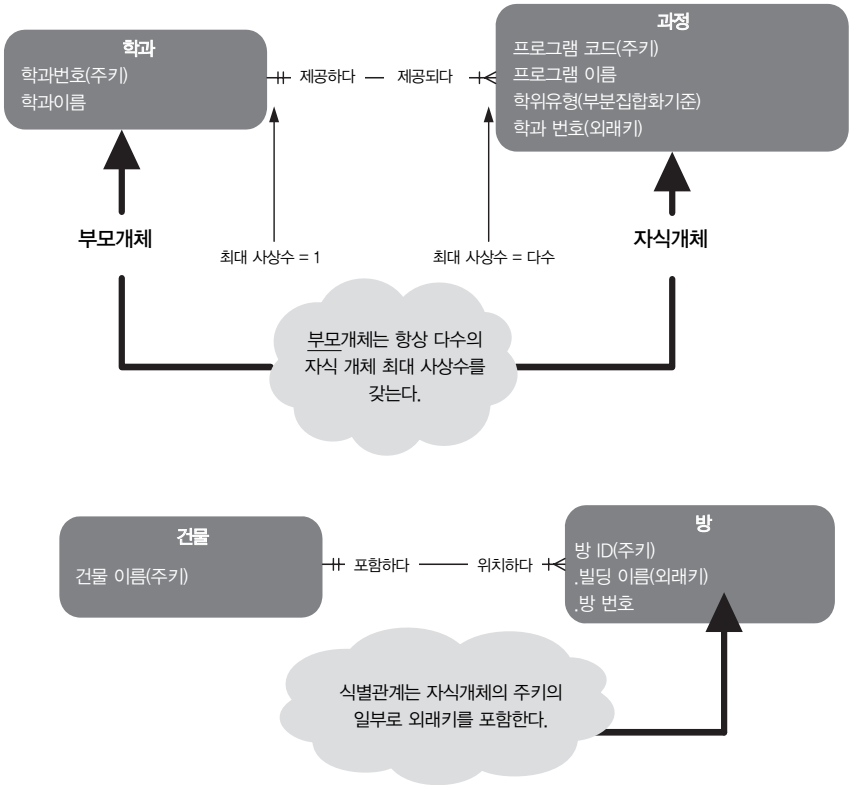


그림 7-6
외래키

부모 개체(parent entity)
자식이라고 불리는 다른 개체에게 하나 이상의 속성을 제공한 데이터 개체. 일대다의 관계에서 부모개체는 일측에 있는 것이다.

자식 개체(child entity)
부모라고 불리는 다른 개체로부터 하나 이상의 속성을 가져온 데이터 개체. 일대다의 관계에서 자식 개체는 다측에 있는 것이다.

비식별 관계(Nonidentifying relationships)
참여한 각각의 개체들이 자신의 독립적인 주키를 갖고 있는 관계

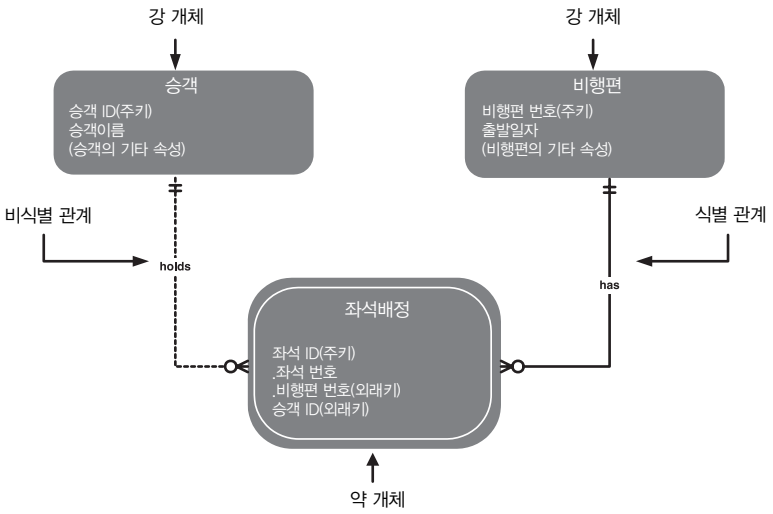
식별 관계(identifying relationship)
부모 개체의 주키가 또한 자식 개체의 주키의 일부로 사용되는 관계

의해서 살펴보기 바란다. 이 경우에 학과는 **부모 개체**(parent entity)라고 불리고 과정은 **자식 개체**(child entity)라고 한다. 부모 개체의 주키가 항상 자식 개체에게 외부키로 제공된다. 그러므로 과정 인스턴스는 현재 그 과정을 제공하고 있는 학과 인스턴스를 가리키는 학과번호의 값을 갖고 있다(외래키는 결코 자식 개체에서 부모개체로 제공될 수 없다).

예제에서, 과정과 학과 사이의 관계는 비식별 관계로 볼 수 있다. **비식별 관계**(Nonidentifying relationships)란 참여한 각각의 개체들이 자신의 독립적인 주키를 가지고 있는 관계를 말한다. 다른 말로, 주키 속성이 공유되지 않는 것을 의미한다. 과정이나 학과 개체는 또한 강한 개체(strong entity) 혹은 독립 개체(independent entity)라고 불리는데 그 이유는 두 개체 모두 식별을 위해서 다른 개체에게 의존하고 있지 않기 때문이다. 그러나 종종 외래키가 자식 개체에서 주키의 일부분으로 참여하는 경우가 있다. 예를 들어, [그림 7-6(b)]에서 부모 개체인 건물은 자신의 키를 방 개체에게 제공하고 있다. 그러므로 건물 이름은 방과 건물을 연결하는 외래키로 사용되고 있으며, 방이라는 개체의 인스턴스들을 유일하게 식별하는 방 ID를 구성하는 요소로 사용되고 있다. 이러한 상황에서 부모 개체와 자식 개체간의 관계는 **식별 관계**(identifying relationship)라고 한다. 식별 관계는 부모 개체의 주키가 자식 개체의 주키의 일부로 사용되는 관계를 말한다. 식별 관계에서 자식 개체는 약한 개체(weak entity)라고 불리는데 그 이유는 자식 개체의 식별이 부모 개체의 존재 여부에 의존하기 때문이다.

가장 대표적인 CASE 도구들과 데이터 모델링 방법들은 식별 관계와 비식별 관계 그리고 강한 개체와 약한 개체를 구별하여 위하여 서로 다른 표현법을 사용하고 있다. [그림 7-7]에서 승객과 좌석 배정간의 비식별 관계를 표현하기 위하여 점선을 사용하였다. 좌석 배정의 주키의 일부가 부모 개체인 비행편으로부터 가져온 비행편 번호라는 외래키이므로 이 관계는 식별 관계이며 직선으로 표현되었다. 마지막으로 좌석 배정은 자신의 주키를 완성하기 위하여 비행편의 주키를 받았으므로 약한 개체가 된다. 약한 개체의 표현은 모서리가 둥근 사각형 내에 모서리가 둥근 사각형을 겹쳐서 사용한다.

그림 7-7
약한 개체와 비식별관계의 표기법



비고 : 식별 관계와 비식별 관계 그리고 강한 개체와 약한 개체의 개념을 강화하고, 가장 대표적인 데이터 모델링 방법과 가장 널리 사용되는 CASE도구와의 일관성을 위해서, 추후에 이 책에서 제시되는 데이터 모델링 예제에서 앞의 표현법을 사용한다.

부모 개체와 자식 개체를 구별하지 못한다면 어떤 일이 일어날까? 예를 들어, [그림 7-8(a)]에서 과정은 한 명도 없거나, 한 명이거나, 여러 명의 학생을 수용할 수 있음을 알 수 있다. 동시에 학생은 하나 혹은 그 이상의 과정에 등록할 수 있음도 알 수 있다. 양쪽의 최대 사상수가 '다수' 가 된다. 그렇다면, 어느 쪽이 부모 개체이고 어느 쪽이 자식 개체일까? 대답할 수 없을 것이다. 이러한 것을 **불특정 관계**(nonspecific relationship)라고 한다. 불특정 관계 혹은 다대다 관계(nonspecific or many-to-many relationship)는 한 개체의 다수 개체 인스턴스 중의 하나가 다른 개체의 다수의 개체 인스턴스와 연관되는 것을 말한다. 이러한 관계는 초기 데이터 모델에만 적합한 것이고 가능한 빨리 해소되어야 한다.

불특정 관계(nonspecific relationship)
한 개체의 다수 개체 예 중의 하나가 다른 개체의 다수의 개체 예와 연관되는 관계. 다대다 관계라고 한다.

많은 불특정 관계들은 한 쌍의 일대다 관계로 해소될 수 있다. [그림 7-8(b)]에서 각 개체는 부모 개체가 되었다. 새로운 연관 개체가 각 부모 개체의 자식 개체로 도입되었다. [그림 7-8(b)]에서 전공의 각 개체 인스턴스는 하나의 과정에 등록한 한 명의 학생을 표현하고 있다. 만약 한 학생이 두 가지 전공을 선택하면, 그 학생은 전공 개체에서 두 개의 개체 인스턴스를 갖게 될 것이다.

[그림 7-8(b)]를 주의 깊게 살펴보자. 연관 개체에 있어서, 자식 개체로부터 부모 개체로의 사상수는 항상 일대일이다. 이것은 전공의 하나의 개체 인스턴스는 반드시 한 명의 학생과 연관되어야 하고 또한 한 과정과 연관되어야 하기 때문에 타당한 것이다. 부모 개체로부터 자식 개체로의 사상수는 비즈니스 규칙에 따라 다르다. 앞의 예제에서, 학생은 반드시 하나 이상의 전공을 선택하여야 한다. 반대로, 과정은 등록하는 학생이 없거나, 한 명이거나, 여러 명일 수 있다(아마도 새로운 과정이거나 과정 등록이 승인된 학생이 없을 수도 있다). 연관 개체는 또한 자신을 설명하기 위한 비주키 속성도 갖고 있다(등록일자나 현재의 학위과정

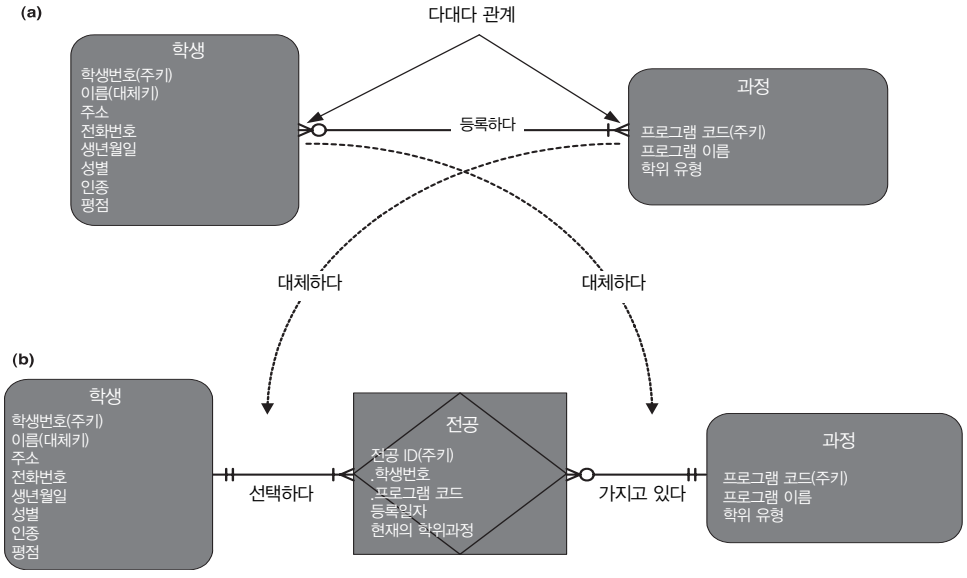


그림 7-8
연관 개체를 이용한 불특정 관계의 해소

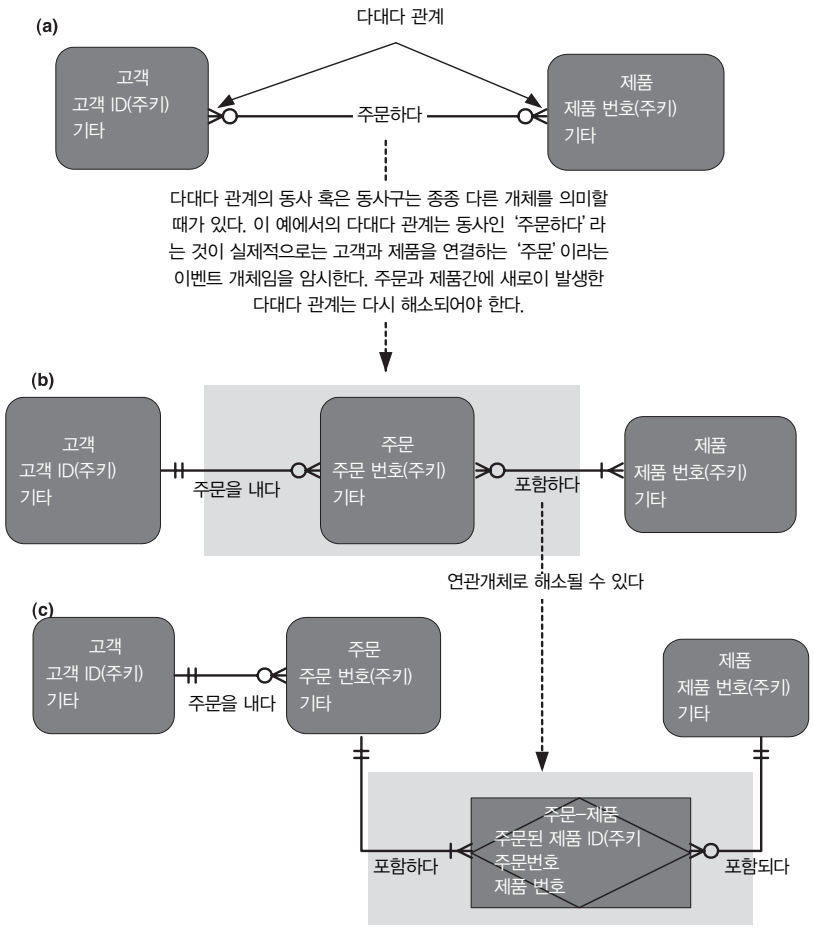
등). 마지막으로, 연관 개체는 부모의 주키를 상속 받는다. 그러므로 모든 연관 개체는 약한 개체이다.

모든 불특정 관계들이 위와 같은 방법에 의하여 자동으로 해소되는 것은 아니다. 우연히 분석가가 다른 개체의 존재를 파악하지 못하여 불특정 관계가 발생하기도 한다. 예를 들면, [그림 7-9(a)]의 고객과 제품 간의 관계를 살펴보자. 고객과 제품 간의 '주문하다' 라는 관계는 어떤 고객이 데이터를 저장하고자 한다는 이벤트를 암시하고 있다. 그 이벤트는 [그림 7-9(b)]에서 '주문' 이라 불리는 이벤트 개체를 나타내고 있다. 사실, 고객과 제품은 [그림 7-9(a)]에서와 같은 자연적이고 직접적인 관계를 갖고 있지 않다. 그 보다는 '주문' 을 통하여 간접적으로 관련되어 있다. 그러므로 다대다 관계는 고객, 주문, 제품 간의 분리된 관계로 대체되었다. 대체된 관계 중에서 주문과 제품 간의 관계가 다대다임에 주의하기 바란다. 그 관계는 [그림 7-8(c)]에서와 같이 연관 개체와 두 개의 일대다 관계로 대체함으로써 해소되어야 한다.

마지막으로, 어떤 불특정 관계는 분리된 관계를 도입함으로써 해소될 수 있다. [그림 7-10(a)] 에서와 같이 이체와 은행 계좌간의 다대다 관계를 살펴보자. 이체라는 트랜잭션은 다수의 은행 계좌를 포함하고, 은행 계좌는 다수의 이체 트랜잭션에 포함될 수 있지만, 데이터 모델링 표현 방법이 우리를 잘못 이끌 수 있기 때문에 주의해야 한다. 기술적으로 하나의 이

그림 7-9

기본적인 비즈니스 개체를 인식함으로써 불특정 관계를 해소



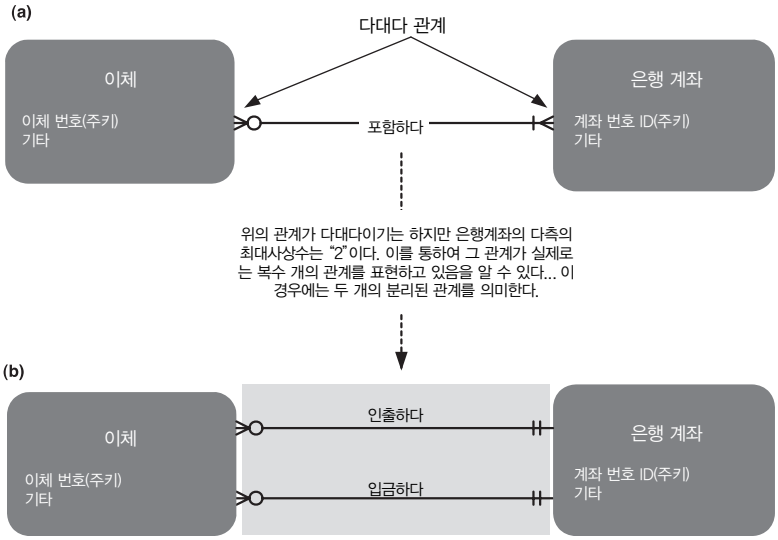


그림 7-10
분리된 관계의 인식에 의한 불특정관계의 해소

체 트랜잭션은 두 개의 은행 계좌를 포함한다. 다대다 관계의 최대 발생 횟수를 알 수 있다면, 그것은 원래의 관계가 약하거나 너무 일반적임을 나타낸다. [그림 7-10(b)]에서와 같이 '포함하다' 라는 관계가 이체와 은행 계좌간의 실제적인 비즈니스 관계를 표현하는 두 개의 분리된 일대다 관계로 대체되었음을 확인할 수 있다.

■ **일반화** 많은 사람들이 객체지향 기법과 일반화의 개념을 연계시키고 있다. 사실, 그 개념들은 수년간 데이터 모델링을 하는 사람들에 의하여 적용되어 왔다. 일반화는 개체들 간의 공통점을 발견하고 탐색해 가는 접근법을 말한다. **일반화**(generalization)는 개체의 여러 유형에 공통으로 존재하는 속성들을 그룹화하여 자체적인 개체로 만드는 기법이다. 예를 들어 전형적인 학교를 생각해보자. 학교에는 학생이 등록해 있고, 직원을 고용하고 있다(대학에서는 어떤 사람은 양쪽 모두에 속할 수 있다). 양쪽의 개체에 공통으로 존재하는 속성이 있는데 예를 들면, 이름, 성별, 인종, 결혼 여부, 그리고 주민등록번호에 기반한 키도 있을 것이다. 이러한 공통 속성들을 사람이라는 상위타입 개체로 결합시킬 수 있을 것이다. **상위타입**(supertype) 개체는 그것에 속한 개체 인스턴스의 속성들이 하나 이상의 **하위타입**(subtype) 개체들에 공통적인 것들을 저장하고 있는 개체이다.

일반화(generalization)
개체의 여러 유형에 공통으로 존재하는 속성들을 그룹화하여 자체적인 개체로 만드는 개념

상위타입(supertype)
그것에 속한 개체 인스턴스의 속성들이 하나 이상의 하위타입(subtype) 개체들에 공통적인 것들을 저장하고 있는 개체이다.

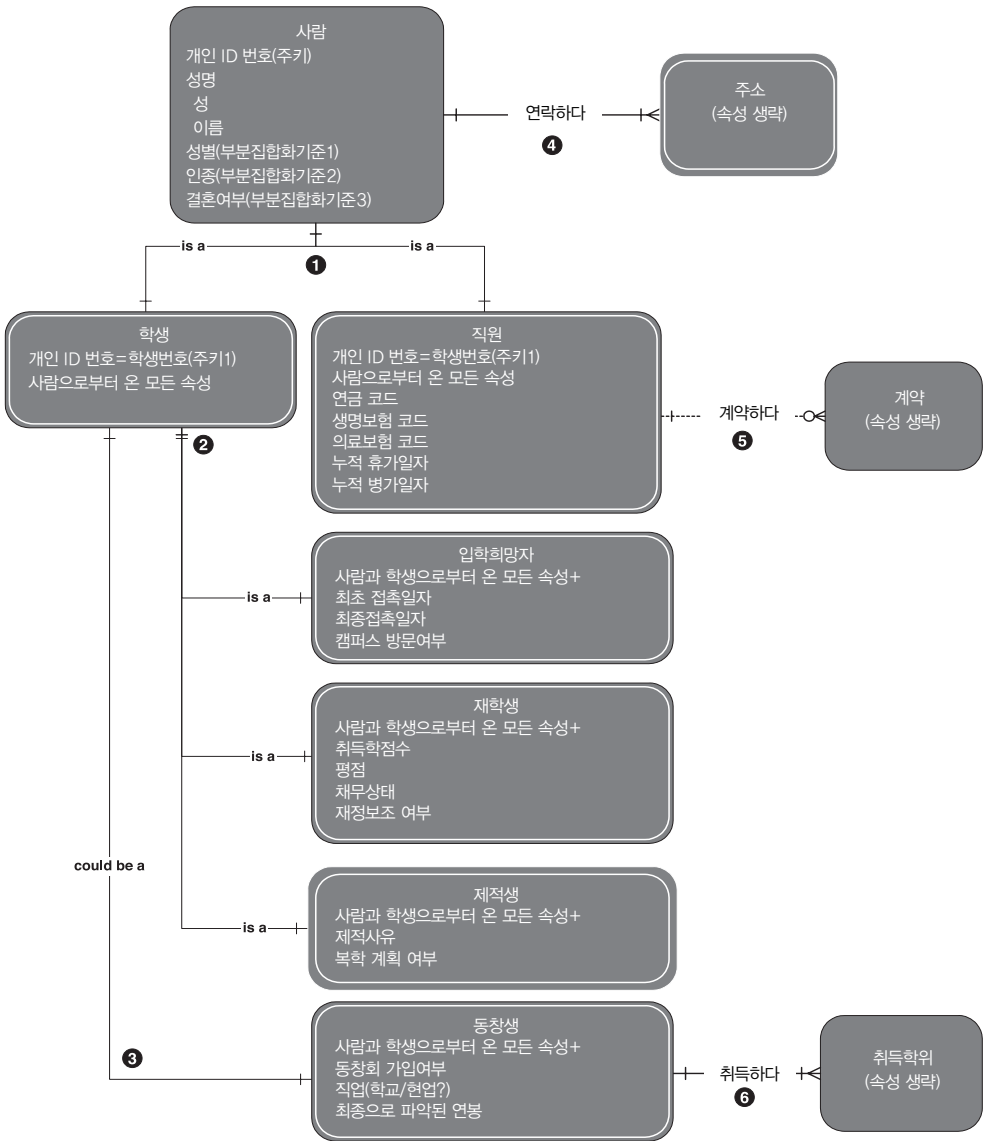
하위타입(subtype)
자신의 개체 인스턴스가 그 개체의 상위타입으로부터 공통 속성을 상속받는 개체

상위타입 개체는 하위타입 개체와 하나 이상의 일대일 관계를 가지게 될 것이다. 이러한 관계들은 'is a' 관계라고 한다(혹은 'was a' , 또는 'could be a'). 그 이유는 상위타입의 각 개체 인스턴스는 또한 하나 혹은 그 이상의 하위타입의 개체 인스턴스이기 때문이다. 하위타입 개체는 상위타입 개체의 공통 속성을 상속받고, 하위타입 개체에 유일한 다른 속성들을 더하여 갖게 된다. 우리의 예에서 "사람은 직원이거나 학생이거나 양쪽 모두이다." [그림 7-11]의 상단부에는 이러한 일반화를 ❶ 계층 구조로 표현하였다. 하위타입인 학생과 직원은 사람으로부터 속성을 상속받고, 또한 자신들의 속성을 더하였다.

의미를 좀 더 확장해 보면, 한 개체는 상위타입과 하위타입 모두 될 수 있다. 다시 [그림 7-11]로 돌아가서, 학생(사람의 하위타입이다) 개체가 자신의 하위타입을 가지고 있음을 볼 수 있다. 예제에서 학생은 ❷ 입학 희망자, 혹은 재학생, 또는 제적생(어떤 이유로 졸업이전에

그림 7-11

일반화 계층



학교를 떠난 학생)이고 또한 ❸ 학생은 동창생일 수 있다. 이러한 추가적인 하위타입 들은 사람의 속성 뿐 아니라 학생이 가진 모든 속성을 상속받는다. 마지막으로 모든 하위타입들은 약한 개체들이다.

일반화의 개념은 상속을 통하여 공통 속성의 공유가 가능하게 하여 속성의 개수를 줄일 수 있도록 해준다. 하위타입은 속성 뿐 아니라, 데이터 타입, 도메인, 그리고 이러한 속성들의 디폴트 값까지 상속받는다. 이것은 수많은 서로 다른 개체들에 적용되는 속성들을 다룰 때 일관성을 높일 수 있도록 해준다(예를 들면, 날자, 이름, 주소, 통화 등).

속성의 상속에 덧붙여서, 하위타입은 또한 다른 개체에 대한 관계도 상속받는다. 예를 들면, 모든 직원과 학생은 ❹ 사람과 주소간의 관계를 상속받는다. 그러나 직원만이 ❺ 계약과의 관계를 상속 받는다. 그리고 동창생만이 ❻ 수여 학위와 관계를 가질 수 있다.

논리적 데이터 모델링 프로세스

여러분은 데이터 모델의 기본 개념에 대하여 학습하였고 이제 데이터 모델링 프로세스에 대하여 살펴보겠습니다. 이것은 언제 할까? 얼마나 많은 데이터 모델들이 작성될까? 이러한 프로세스를 지원하는 기술들은 어떤 것이 있을까?

데이터 모델링은 여러 유형의 프로젝트에서 수행되며, 또한 프로젝트의 여러 단계에서 수행된다. 데이터 모델은 점진적인 것으로 비즈니스나 애플리케이션을 위한 '최종' 데이터 모델이라는 것은 존재하지 않는다. 대신에 데이터 모델은 변화하는 비즈니스에 반응하는 살아있는 문서로 이해되어야 한다. 데이터 모델을 저장소(repository)에 저장되는 것이 이상적이며, 이를 통해서 검색되고, 확장되고, 그리고 시간의 흐름에 따라 편집될 수 있다. 이제 시스템 계획 및 분석 과정에서 데이터 모델링이 어떤 역할을 하는지 살펴보도록 하자.

전략적 데이터 모델링

많은 조직들이 전략적 정보시스템 계획에 따라 애플리케이션 개발 프로젝트를 선정하고 있다. 전략 계획은 별도의 프로젝트이다. 이 프로젝트는 전반적인 비전과 정보시스템의 아키텍처를 정의하는 정보시스템 전략 계획을 생성한다. 거의 대부분의 경우에 이 아키텍처는 **전사적 데이터 모델**(enterprise data model)을 포함한다. 정보 공학이 이러한 접근법을 포함하고 있는 방법론이다.

전사적 데이터 모델은 전형적으로 가장 기본적인 개체들만으로 구성된다. 이 개체들은 전형적으로 정의되지만(사전에 정의된다) 키나 속성 등으로 설명되지는 않는다. 전사적 데이터 모델은 관계를 포함할 수도 있고 포함하지 않을 수도 있다(계획 방법론 표준과 최고 경영층이 바라는 상세화 정도에 따라 다르다). 만약 관계가 포함된다면, 그것들 중 상당수는 불특정 관계일 것이다.

전사적 데이터 모델은 통상 기업 저장소(corporate repository)에 저장된다. 애플리케이션 개발 프로젝트가 시작되었을 때, 전사적 데이터 모델의 부분 집합이(다른 모델과 마찬가지로) 기업 저장소에서 추출되어 프로젝트 저장소로 옮겨진다. 프로젝트 팀이 시스템 분석과 설계를 완료하였을 때, 확장되고 정제된 데이터 모델이 다시 기업 저장소로 저장된다.

시스템 분석단계에서의 데이터 모델링

이번 장에서, 우리는 시스템 분석의 일부분으로 논리적 데이터 모델링에 초점을 맞출 것이다. 단일 정보시스템을 위한 데이터 모델을 **애플리케이션 데이터 모델**(application data model)이라고 한다.

애플리케이션 데이터 모델 (application data model)
단일 정보시스템을 위한 완전한 데이터 모델

데이터 모델이 시스템 분석의 범위 정의 단계 동안에 구축되는 경우는 거의 없다. 범위 정의 단계의 짧은 수행시간으로 말미암아 그 작업이 비현실적이 된다. 만약 전사적 데이터 모델이 존재한다면, 프로젝트에 적용 가능한 모델의 부분 집합이 검색되고 상황을 설정하기 위한 요구사항 분석단계의 일부분으로 검토될 수 있다. 다른 방법으로, 프로젝트 팀이 개체의 간단한 리스트를 작성할 수 있을 것이다. 이것은 프로젝트 팀원들이 시스템이 수집하고 저장

해야 하는 데이터를 생각해보는 방식으로 수행될 수 있다.

문제분석단계에서 모델은 개체와 관계만을 포함하고 속성은 포함하지 않는데 이것을 **상황 데이터 모델**(context data model)이라고 한다. 이것의 목적은 범위에 대한 이해를 깊게 하는 것이지 개체나 비즈니스 규칙 등에 대하여 자세히 살피는 것은 아니다. 많은 관계들이 불특정 관계이다.

요구사항 분석은 다음과 같은 단계에서 개발되는 논리적 데이터 모델을 생성한다.

상황 데이터 모델(context data model)
개체와 관계는 포함하고 있으나 속성은 포함하고 있지 않은 데이터 모델

키 기반 데이터 모델(key based data model)
불특정 관계를 연관 개체로 해소하여 정확한 사상수를 가진 개체와 관계를 포함하며, 주키와 대체키를 가지고 있는 데이터모델

모든 속성이 표현된 데이터 모델(fully attributed data model)
모든 개체, 속성, 관계, 부분 집합화 기준, 그리고 정확한 사상수를 포함하고 있는 데이터 모델

1. 프로젝트 범위를 설정하기 위하여 상황 데이터 모델을 작성하는 것으로 시작한다. 만약 문제분석 과정에서 이미 상황 데이터 모델이 작성되었다면, 그 모델은 새로운 요구사항과 프로젝트 범위를 반영하기 위하여 갱신될 것이다.
2. 다음으로 **키 기반 데이터 모델**(key based data model)이 작성된다. 이 모델은 불특정 관계를 제거하고, 연관 개체를 추가하며, 주키와 대체키를 포함한다. 키 기반 데이터 모델은 정확한 사상수와 일반화 계층 구조를 포함한다.
3. 다음으로 **모든 속성이 표현된 데이터 모델**(fully attributed data model)이 구축된다. 이 모델은 남아 있는 모든 묘사적 속성과 부분 집합화 기준을 포함한다. 각 속성의 데이터 타입, 도메인 그리고 디폴트 값 등은 저장소에 정의된다(종종 완전히 묘사된 데이터 모델 - fully described data model - 이라고도 한다).
4. 완성된 데이터 모델이 정규화(normalization)라고 불리는 과정을 통하여 적용성과 유연

표 7-4 데이터 모델링을 위한 JPR과 인터뷰 질문들

목적	후보 질문들
시스템 개체 발견	비즈니스의 대상이 무엇인가? 다른 말로, 어떤 유형의 사람, 조직, 조직 단위, 장소, 사물, 자재, 혹은 이벤트 등이 데이터가 포착되거나 유지되어야 하는 시스템에 사용되거나 교류하는가? 각 대상에 대하여 얼마나 많은 경우가 존재하는가?
개체의 키 발견	동일한 대상의 다른 경우들로부터 구별해주는 유일한 특성은 무엇인가? 향후에 이러한 식별 체계를 변경할 계획이 있는가?
부분 집합화 기준의 발견	대상의 모든 경우들을 유용한 부분 집합으로 분할할 수 있는 특성이 있는가? 경우들을 그룹으로 묶는 적절한 방법이 없는 대상들의 부분 집합이 있는가?
속성과 도메인의 발견	각 대상을 설명하는 특성들이 무엇인가? 이러한 특성들에 대하여, (1) 어떤 유형의 데이터가 저장되는가? (2) 데이터의 적정한 값을 정의하는 책임은 누가 지고 있는가? (3) 그 데이터에 대한 적정한 데이터는 무엇인가? (4) 값이 요구되는가? (5) 달리 지정하지 않으면 저장되어야 하는 디폴트 값이 있는가?
보안과 통제요구의 발견	누가 그 데이터를 사용하거나 보는 것에 대한 제약이 있는가? 데이터를 생성하는 것은 누구에게 허용되어 있는가? 데이터의 갱신은 누구에게 허용되어 있는가? 누구에게 데이터 삭제가 허용되어 있는가?
데이터 타이밍 요구사항 발견	데이터가 얼마나 자주 변경되는가? 데이터가 어느 정도의 시간 동안 비즈니스에 가치가 있는가? 데이터를 얼마 동안 보관하여야 하는가? 데이터의 시간적 변화나 경향이 필요한가? 만약 특성이 변한다면, 직전 값을 알아야 하는가?
일반화 계층의 발견	각 대상의 모든 경우들이 동일한가? 다르게 묘사되거나 처리되는 각 대상의 특별한 유형이 있는가? 어떤 데이터가 공유를 위하여 결합될 수 있는가?
관계와 차수의 발견	대상간의 연관을 나타내는 어떤 이벤트가 있는가? 동일하거나 다른 유형의 다수의 서로 다른 대상에 대한 데이터를 처리하거나 변경하는데 어떤 비즈니스 활동이나 트랜잭션이 포함되는가?
사상수의 발견	각 비즈니스 활동이나 이벤트가 동일한 방식으로 처리되거나 특별한 상황이 있는가? 한 이벤트가 어떤 연관된 대상하고만 발생하거나, 혹은 모든 대상이 포함되어야 하는가?

성을 평가 받게 된다. 최종적으로 분석된 모델은 정규화된 데이터 모델(normalized data model)이라고 한다.

이러한 데이터 요구사항 모델은 시스템 분석가, 사용자, 그리고 관리자들을 포함하는 협동이 필요하다. 데이터 관리자(data administrator)는 종종 모든 데이터 모델을 위한 표준을 설정하고 데이터 모델들을 승인한다.

마지막으로, 데이터 모델은 사용자 집단에 의하여 제공되는 적절한 사실과 정보가 없이는 작성될 수 없다. 이러한 사실들은 현존하는 양식과 파일에 대한 샘플링이나, 비슷한 시스템의 조사, 혹은 사용자와 관리자에 대한 조사, 그리고 사용자와 관리자에 대한 인터뷰와 같은 다양한 기법으로 수집할 수 있다. 사실과 정보를 수집하고 동시에 데이터 모델을 작성하고 검증하는 가장 빠른 방법은 합동 요구사항 계획(JRP, Joint Requirement Planning)이다. JPR은 사실 수집을 위하여 주의 깊게 준비된 그룹 회의를 하고, 모델을 구축하며, 그 모델을 검증하는데 통상 하루 내지 이틀 정도의 시간을 사용한다. 사실발견과 정보 수집 기법들은 5장에서 자세히 다루었다. [표 7-4]에는 데이터 모델링에 관련된 사실발견과 정보 수집에 유용하게 사용될 수 있는 질문들을 요약해 두었다.

시스템 설계를 위한 대비

시스템 설계 과정 중에, 논리적 데이터 모델은 선정된 데이터베이스 관리 시스템을 위한 물리적 데이터 모델(데이터베이스 스키마라고 한다)로 변환된다. 이 모델은 데이터베이스 기술이 가진 기술적 능력과 제약을 반영하고 있을 뿐 아니라 데이터베이스 관리자에 의하여 제시된 성능 조정 요구사항 또한 반영하고 있다. 데이터베이스에 대한 더 이상의 논의는 13장까지 미뤄두고자 한다.

데이터 모델링을 위한 자동화 도구들

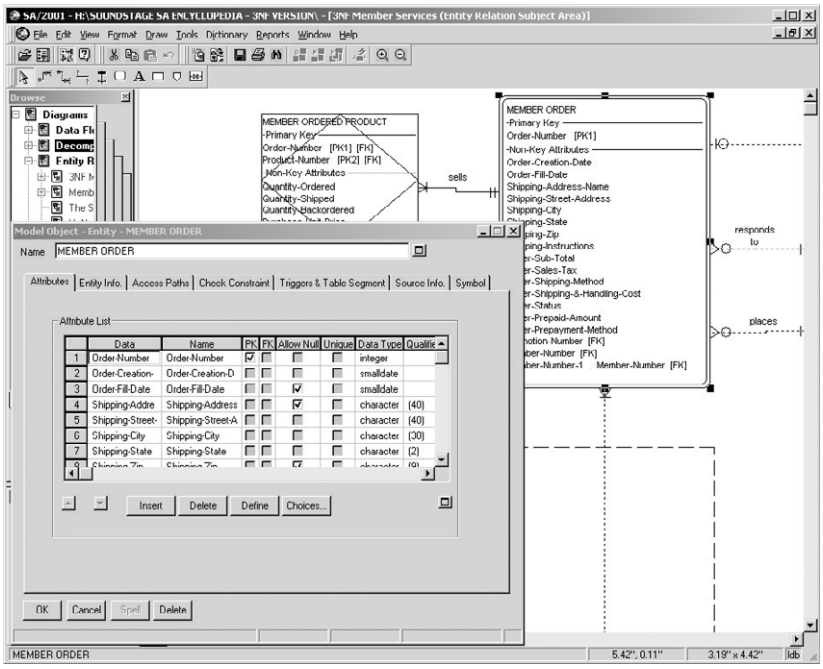
데이터 모델들은 저장소에 저장된다. 데이터 모델은 **메타데이터**(metadata)이다. 그것은 비즈니스 데이터에 대한 데이터라는 것이다. 2장에서 소개한 컴퓨터 지원 시스템 공학(computer-aided systems engineering, CASE) 기술이 데이터 모델과 그것의 상세한 설명을 저장하는 저장소를 제공한다. 대부분의 CASE 제품들은 컴퓨터 지원 데이터 모델링과 데이터베이스 설계를 지원한다. 어떤 CASE 제품은(Logic Works의 ERwin 등) 데이터 모델링과 데이터베이스 설계만을 지원한다. CASE는 이러한 모델들을 작성하고 유지하거나 상세화 작업과 같은 노역을 제거해 준다.

메타데이터(metadata)
데이터의 데이터

CASE 제품을 이용하여 종이, 연필, 지우개, 그리고 템플릿 등을 사용하지 않고 전문적이고 읽기 좋은 데이터 모델을 쉽게 작성할 수 있다. 모델들은 최종사용자에 의하여 제시되는 변경이나 수정사항을 반영하여 손쉽게 변경될 수 있으므로 다시 시작할 필요가 없다. 또한 대부분의 CASE 제품들은 뛰어난 분석 도구들을 제공하여 작성된 모델의 기계적 오류, 완전성 그리고 일관성 등을 확인할 수 있다. 어떤 CASE 제품은 데이터 모델의 일관성, 완전성, 유연성 등을 분석하는 것을 돕기도 한다. 시간과 품질에 있어서의 잠재적인 절감은 상당하다.
앞에서 언급한 바와 같이, 어떤 CASE 도구는 기존의 파일과 데이터베이스 구조를 데이터 모

그림 7-12

CASE 도구인 System Architect의 화면 캡처



델로 변환하는 역공학(reverse engineering)을 지원한다. 결과 데이터 모델은 새로운 파일이나 데이터베이스로 개편될 수 있는 '물리적' 데이터 모델을 나타내거나, 동등한 '논리적' 데이터 모델로 번역될 수 있을 것이다. 그러면 논리적 데이터 모델은 편집되고 개편된 물리적 데이터 모델로 순 공학(forward engineering)되고 파일과 데이터베이스로 구현된다.

CASE 도구들도 제약점을 가지고 있다. 모든 데이터 모델링 표현법을 지원하는 CASE 도구는 없다. 그리고 서로 다른 CASE 도구들은 동일한 데이터 모델링 방법에 대하여 약간씩 다른 표현법을 사용한다. 그러므로 CASE 제품이 자신의 방법론에 있는 데이터 모델링 심벌이나 접근법을 사용하도록 강제할 수 있다.

다음 부분에 나오는 사운드스테이지 데이터 모델은 Popkin Systems and Software의 CASE 도구인 System Architect 2001로 작성되었다. 사례연구를 위해서 출력물을 제공하는데 색상은 사용하지 않았고 흥미 있는 특정 아이템에 집중할 수 있도록 불릿 등만을 수정하였다. 사운드스테이지 데이터 모델의 모든 개체, 속성 그리고 관계들은 자동적으로 System Architect의 프로젝트 저장소에 자동으로 목록화되어 저장된다. [그림 7-12]는 데이터 모델링을 하기 위하여 사용된 System Architect의 화면을 표시하고 있다.

데이터 모델을 작성하는 방법

여러분은 데이터 모델을 읽고 해석할 수 있을 만큼 충분히 알고 있다. 그러나 시스템 분석가나 잘 아는 최종사용자로서, 여러분은 데이터 모델을 작성하는 방법을 알아야 한다. 데이터 모델을 작성하는 방법을 배우기 위하여 사운드스테이지의 회원서비스 프로젝트를 사용할 것이다.

비고 : 이 예제는 메모로부터 데이터 모델을 작성하는 것을 가르친다. 현실에서는 항상 기존의 데이터 모델을 찾아야 한다. 만약 그런 모델이 있다면, 데이터 관리 그룹은 보통 그것들을 유지하고 있다. 다른 방법으로, 기존의 데이터베이스로부터 역공학을 통하여 데이터 모델을 얻을 수 있다.

개체 발견

데이터 모델링의 첫 번째 작업은 상대적으로 쉽다. 데이터에 의하여 설명되거나 설명되어질 시스템 내의 기본적인 개체를 발견할 필요가 있다. 최종사용자가 저장하고자 하는 데이터에 대해서만 생각하는 것으로 사고를 제약할 필요는 없다. 개체를 발견하는데 사용될 수 있는 몇 가지 기법들이 다음과 같다.

- 인터뷰 동안 혹은 시스템 소유자 및 사용자와 JRP 세션을 갖는 동안에 그들의 토의에 등장하는 주요 단어에 집중한다. 예를 들면, 사운드스테이지의 비즈니스 환경과 활동 등에 대하여 개별적인 논의를 진행하는 인터뷰 동안에, 사용자가 "우리는 모든 회원과 그들이 체결한 계약을 추적해야 한다"라고 말하였다. 이 대화에서 주요 단어는 회원과 계약임을 알 수 있고 이 둘은 모두 개체가 된다.
- 인터뷰 혹은 JRP 동안에, 시스템 소유자와 사용자에게 그들이 수집하고, 저장하고 정보로 생성하고자 원하는 것들을 가려달라고 특별히 요청한다. 그러한 것들은 종종 데이터 모델에서 표현되어야 하는 개체를 나타낸다.
- 개체를 식별하는 또 다른 방법은 기존의 양식, 파일, 그리고 보고서를 분석하는 것이다. 어떤 양식들은 이벤트 개체를 식별하게 해준다. 예를 들면, 주문서, 청구서, 지불 명세서, 예치 명세서 등을 생각할 수 있다. 전산화된 등록 시스템의 컴퓨터 파일, 데이터베이스, 혹은 출력 등을 분석함으로써 이러한 개체들을 발견할 수도 있다.
- 요구사항 분석단계에 유스케이스의 설명부분이 작성되었다면, 그것은 데이터 속성과 개체의 원천이 될 수 있다. 유스케이스의 설명부분에서 명사 형태를 찾아낸다. 모든 명사 형태는 데이터 속성이나 개체의 후보가 된다. 9장에서 수행하는 방법을 설명하는데, 객체지향 접근법에 따라 객체와 그들의 속성에 대한 목록을 작성한다. 데이터 개체와 속성을 발견하기 위하여 이와 매우 유사한 접근법을 사용할 수 있다.
- 기술이 또한 개체를 식별하는데 도움을 줄 수 있다. 일부 CASE 도구들은 기존의 파일이나 데이터베이스를 물리적 데이터 모델로 역공학을 이용하여 변환할 수 있다. 분석가는 변환된 모델에 대하여 물리적 이름, 코드 코멘트 등을 그들의 논리적이고 비즈니스적으로 친근한 형태로 대체함으로써 정제해야 한다.

개체가 발견되면, 그들에게 단순하고, 의미 있으면서 비즈니스적인 명칭을 부여한다. 개체는 사람, 이벤트, 장소, 객체, 혹은 데이터를 저장하고 싶은 사물 등을 설명하는 명사로 이름을 작성해야 한다. 약어 혹은 첫 글자만으로 축약한 형태 등은 사용하지 않는 것이 좋다. 명칭은 단수로 표현하고 이를 통하여 실제 개체의 인스턴스와 개체의 논리적 개념을 구분하도록 한다. 명칭은 개체를 좀 더 잘 설명하기 위하여 적절한 형용사나 구절을 포함할 수 있다. 예를

표 7-5 사운드스테이지 프로젝트를 위한 기본 개체들

개체 이름	비즈니스 정의
계약	회원이 특정 시간 내에 특정 수량의 제품을 구매하기로 동의한 계약. 그 계약이 완료된 후에 회원은 무료 제품이나 제품 할인 혜택을 받을 수 있는 보너스 점수를 부여 받게 된다.
회원	하나 혹은 그 이상의 클럽에 속한 활동 회원. 비고: 대상 시스템의 목적은 활동하지 않는 회원을 삭제하는 대신에 재가입시키는 것이다.
회원 주문	월간 판촉의 일부로 회원을 위하여 생성된 주문 혹은 회원에 의하여 주도된 주문. 비고: 현재 시스템은 판촉으로부터 생성된 주문만 지원한다. 그러나 제안된 시스템에서는 부가된 옵션으로 고객이 주도하는 주문이 높은 우선순위가 주어지도록 하였다.
트랜잭션	회원서비스 시스템이 반응해야만 하는 비즈니스 이벤트
제품	회원에게 판매하거나 판촉을 위하여 보관하고 있는 제품. 비고: 시스템 개선 목표는 (1) 창고를 위하여 개발되고 있는 새로운 바코드 시스템과의 호환성을 유지하고 (2) 급격히 변호하고 있는 제품 혼합에 적응하는 능력을 향상 시키는 것을 포함하고 있다.
판촉	회원에게 특별 제품 제공 기회가 주어지는 월별 혹은 분기별 이벤트

들어, 외부에서 들어온 고객 주문은 내부에서 생성되는 재고 주문과 구분되어야 한다.

각 개체에 대하여, 비즈니스 용어로 정의한다. 개체를 기술적인 용어로 정의하지 말아야 하며 또한, '~에 대한 데이터' 등의 정의도 피하여야 한다. 정의의 초안을 만들기 위해서 사전을 이용하고 그것을 비즈니스에 가까운 의미가 되도록 다듬는다. 여러분이 작성한 개체의 이름과 정의는 여러분과 추후에 분석가와 사용자 모두가 사용하게 될 초기 비즈니스 용어 사전이 될 것이다.

사운드스테이지의 관리자와 사용자들이 초기에 식별해 놓은 개체들이 [표 7-5]에 제시되어 있다. 정의들이 시스템의 용어를 설정하는데 어떻게 기여하는가를 살펴보기 바란다.

상황 데이터 모델

데이터 모델링의 다음 과업은 상황 데이터 모델을 작성하는 것이다. 상황 데이터 모델은 앞서 발견된 기본적인 비즈니스 개체와 그들의 자연적인 관계들을 포함하여야 한다.
관계들은 동사구의 형태로 이름 붙여져야 하는데, 개체의 이름과 결합되었을 때 단순한 비즈니스 문장이나 주장이 되도록 만든다. System Architect와 같은 일부 CASE 도구들은 양방향으로 관계에 이름을 붙이도록 한다. 그렇지 않다면, 관계의 이름은 항상 부모에서 자식으로 붙인다.

이 작업을 완성한 것을 [그림 7-13]에서 볼 수 있다. 이 그림은 System Architect에서 작성된 데이터 모델이다. 속성의 대응 작업을 시작하면, 새로운 개체와 관계들이 들어날 수 있다. 아래에 있는 번호들은 [그림 7-13]에 표시된 것과 동일한 것이며 ERD는 다음과 같은 의미를 갖는다.

❶ 하나의 계약은 한 명 혹은 그 이상의 회원과 체결된다. 관계의 이름은 한 방향(부모로부터 자식으로)으로 주어지지만 다른 방향은 유추할 수 있다. 예를 들면, 회원은 오직 하나의 계약을 체결한다는 의미를 갖고 있다.

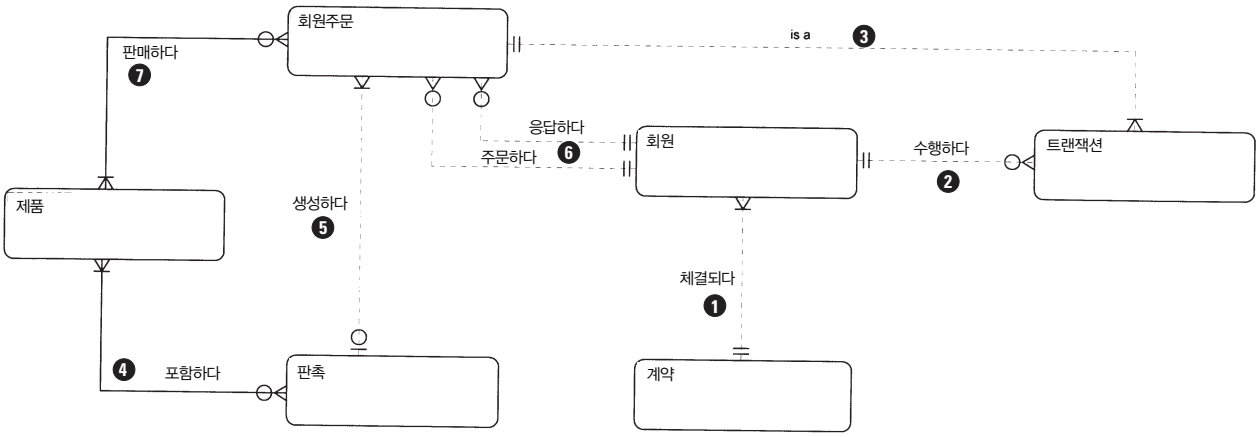


그림 7-13

사운드스테이지의 상황 데이터 모델

❷ 회원은 하나도 없거나, 하나 혹은 그 이상의 트랜잭션을 수행한다. 주어진 트랜잭션은 오직 한 명의 회원에 의하여 수행된다는 의미를 내포하고 있다

❸ 회원 주문은 트랜잭션이다. 사실, 하나의 회원 주문은 다수의 트랜잭션에 대응될 수 있다(예를 들면, 새로운 회원 주문, 취소된 회원 주문, 변경된 회원 주문 등). 그러나 주어진 트랜잭션은 회원 주문을 표현할 수도 표현하지 못할 수도 있다.

❹ 판촉은 하나 혹은 그 이상의 제품을 포함한다. 제품은 없거나, 하나이거나, 혹은 다수의 판촉에 포함된다. 예를 들면, 컨트리/웨스턴 스타일의 음악과 light-rock에 모두 속하는 CD는 두 부분의 판촉에 모두 포함될 수 있다. 판촉에 비하여 제품 수가 훨씬 많으므로 대부분의 제품은 판촉에 포함되지 않는다.

❺ 판촉은 많은 회원 주문을 생성한다. 회원 주문은 없거나 하나의 판촉에 의하여 생성된다. 왜 없을까? 새로운 시스템에서, 회원은 자신의 주문을 낼 수 있을 것이다.

❻ 만약 분리된 관계가 서로 다른 비즈니스 이벤트나 연관을 나타낸다면, 동일한 두 개체간에 하나 이상의 관계가 존재할 수 있다. 그러므로 회원은 없거나, 하나이거나, 다수의 회원 주문에 응답한다. 이 관계는 판촉에 의하여 생성된 주문을 지원한다. 회원은 없거나, 하나이거나 혹은 다수의 회원 주문을 낼 수 있다. 이 관계는 회원이 주도하는 주문을 지원한다. 두 경우에, 회원 주문은 정확히 한 명의 회원에 의하여 작성된다(응답된다).

이 두 관계에는 필요 없지만, 일부 CASE 도구는(System Architect 포함) 불리언(Boolean) 관계(AND, OR 등)를 표현하는 심벌을 제공하고 있다. 그러므로 두 개의 관계에 대하여, 관계의 예들이 상호 배타적(= OR)이거나 상호 부수적인(= AND) 것을 나타낼 때 불리언 심벌이 사용될 수 있다.

❼ 회원 주문은 하나 혹은 그 이상의 제품을 판매한다. 제품은 없거나, 하나 혹은 다수의 회원 주문에 판매된다는 의미를 내포하고 있다. 이것은 불특정 관계로 나중에 해소되어야 하는 것을 기억하기 바란다.

여러분의 앞의 내용을 주의 깊게 읽었다면, 사운드스테이지 시스템에 대하여 상당히 많이 알게 되었을 것이다. 데이터 모델은 시스템 프로젝트에서 비즈니스 상황을 설명하는 대중적인 도구로 점점 더 많이 사용되고 있다.

키 기반 데이터 모델

다음 단계의 작업은 각 개체의 키를 찾아내는 것이다. 다음은 키를 찾기 위한 가이드라인이다.[1)]

1. 키의 값은 각 개체 인스턴스의 전 생애 동안 변해서는 안 된다.
   예를 들면, 이름은 좋지 않은 키로 볼 수 있는데 그 이유는 결혼이나 이혼 등으로 성(last name)이 바뀔 수 있기 때문이다(역자 주 : 우리나라와는 다른 서구의 예임).
2. 키의 값은 공값(null)일 수 없다.
3. 키의 값이 유효한지 확인 할 수 있는 통제가 설치되어야 한다. 이것은 정확한 도메인 정의와 데이터베이스 관리 시스템의 도메인 확인 기능을 활용하여 달성될 수 있다.

지능적인 키(intelligent key)
키의 구조가 개체 인스턴스에 대한 데이터를 갖고 있는 비즈니스 코드

4. 일부 전문가(Bruce)는 지능적인 키를 피할 것을 권한다. **지능적인 키**(intelligent key)는 키의 구조가 개체 인스턴스에 대한 데이터(예를 들면, 분류, 크기 혹은 다른 특성 등)를 갖고 있는 비즈니스 코드이다. 코드는 비즈니스 시스템 내의 무엇인가를 나타내는 숫자나 문자의 그룹이다. 일부 전문가는 그러한 특성들은 변화될 수 있기 때문에 위의 규칙 1에 위배될 것이라고 말한다.

   우리는 여기에 동의하지 않는다. 비즈니스 코드는 컴퓨터의 도움 없이 사람에 의하여 신속하게 처리될 수 있기 때문에 조직에게 가치를 줄 수 있다.

   a. 여러 유형의 코드가 있는데 그것들은 서로 결합되어 개체 인스턴스의 식별을 위한 효과적인 수단을 제공한다.
      (1) 연속 코드(serial code)는 개체 인스턴스에게 연속적으로 생성된 번호를 부여한다. 많은 데이터베이스 관리 시스템들은 비즈니스 요구사항에 따라 연속 코드를 생성하고 통제할 수 있다.
      (2) 블록 코드(block code)는 블록 숫자들이 어떤 비즈니스 의미를 갖는 그룹으로 분할되는 것을 제외하고는 연속 코드와 유사하다. 예를 들면, 위성 방송 사업자는 100-199번을 유료 채널로, 200-299번을 케이블 채널로, 300-399번을 스포츠 채널로, 400-499번을 성인 프로그램 채널로, 500-599번을 음악 채널로, 600-699번을 게임 채널로, 700-799번을 인터넷 채널로, 800-899번을 프리미엄 케이블 채널로 그리고 900-999번을 프리미엄 영화와 이벤트 채널로 배정할 수 있다.
      (3) 알파벳 코드(alphabetic code)는 문자(숫자도 가능함)의 유한한 조합을 이용하여 개체 인스턴스를 표현한다. 예를 들면, 각 상태는 유일한 두 개의 알파벳 문자 코드

1) Adapted from Thomas A. Bruce, Designing Quality Databases with IDEF1X Information Models. Copyright © 1992 by Thomas A. Bruce. Reprinted by permission of Dorset House Publishing, 353 W. 12th St., New York, NY 10014 (212-620-4053/1-800-DH-BOOKS/ www.dorsethouse.com). All rights reserved.

를 갖는다. 알파벳 코드는 대부분 개체의 인스턴스들을 식별하기 위하여 보통 연속 코드나 블록 코드와 결합되어 사용된다.

(4) 유효 위치 코드(significant position code)에서는 각 자리 혹은 자리의 그룹이 개체 인스턴스의 측정 가능하거나 식별 가능한 특성을 설명한다. 유효 자리 코드(significant digit code)가 재고품들을 관리하는데 자주 사용된다. 타이어나 전구에 있는 코드가 유효 위치 코드의 예이다. 그것을 통하여 타이어의 크기나 소비 전력 등의 특성을 알 수 있다.

(5) 계층형 코드(hierarchical code)는 개체 인스턴스들에 대한 하향식 해석을 제공한다. 모든 품목 코드들은 그룹, 하위 그룹 같은 식으로 분할된다. 예를 들면, 직원에 대한 다음과 같은 코드를 고려해 볼 수 있다.

- 첫째 자리는 분류를 나타낸다(사무직, 교수 등).
- 둘째와 셋째 자리는 분류 내에서의 직급을 나타낸다.
- 넷째와 다섯째 자리는 입사 년도를 나타낸다.

b. 다음과 같은 가이드라인이 비즈니스 코드 체계를 만들 때 사용할 수 있다

(1) 코드는 성장에 적응할 수 있도록 확장 가능하여야 한다.

(2) 전체 코드는 각 개체 인스턴스에게 유일한 값을 제공해야 한다.

(3) 코드는 모든 특성을 표현할 수 있을 만큼 충분히 커야 하지만 컴퓨터의 도움 없이 사람이 해독할 수 있을 만큼 충분히 작아야 한다.

(4) 코드는 편리해야 한다. 새로운 개체 인스턴스는 쉽게 만들 수 있어야 한다.

5. 독립적인 개체들의 커다란 복합키를 대체할 수 있는 대용 키 개발을 고려한다. 이 방법은 연관 개체에는 실제적이지 못하다. 그 이유는 연관 개체의 복합키의 각 부분은 부모 개체의 주키와 정확하게 대응되어야 하는 외래키이기 때문이다.

[그림 7-14]는 사운드스테이지 프로젝트의 키 기반 데이터 모델을 나타내고 있다. 각 개체에 대하여 주키가 지정되어 있음을 살펴보기 바란다.

❶ 많은 개체들은 단순하고 속성 한 개로 된 주키를 갖고 있다.

❷ 회원 주문과 제품 간의 불특정 관계를 회원 주문 제품이라는 연관개체를 도입하여 해소하였다. 각 연관 개체의 인스턴스는 하나의 회원 주문에 있는 하나의 제품을 나타낸다. 부모 개체들의 주키는 결합되어 연관 개체의 복합키를 구성한다. System Architect에서는 주문 번호 뒤에 'PK1' 을 써 넣어서 복합 주키의 '첫 부분' 이라는 것을 표시하고 제품 번호 뒤에 'PK2' 를 써 넣어서 복합키의 '두 번째 부분' 임을 나타내고 있다. 또한 복합키의 각 속성들은 외래키이고 정확한 부모 개체 인스턴스를 가리키고 있다.

마찬가지로, 제품과 판촉간의 불특정 관계도 타이틀 판촉이라는 연관 개체를 도입하여 해소 하였고 이것은 부모 개체들로부터 키를 상속 받았다.

이 모델을 개발할 때, 몇 가지 사항을 조심해야 한다. 만약 개체에 대한 키를 정의할 수 없다면, 그 개체는 실제로는 존재하지 않는 것일 수 있다. 그러므로 키를 부여하는 것은 모든 속성이 표현된 데이터 모델을 개발하기 이전에 훌륭한 품질 확인 기능을 수행한다. 또한 두 개

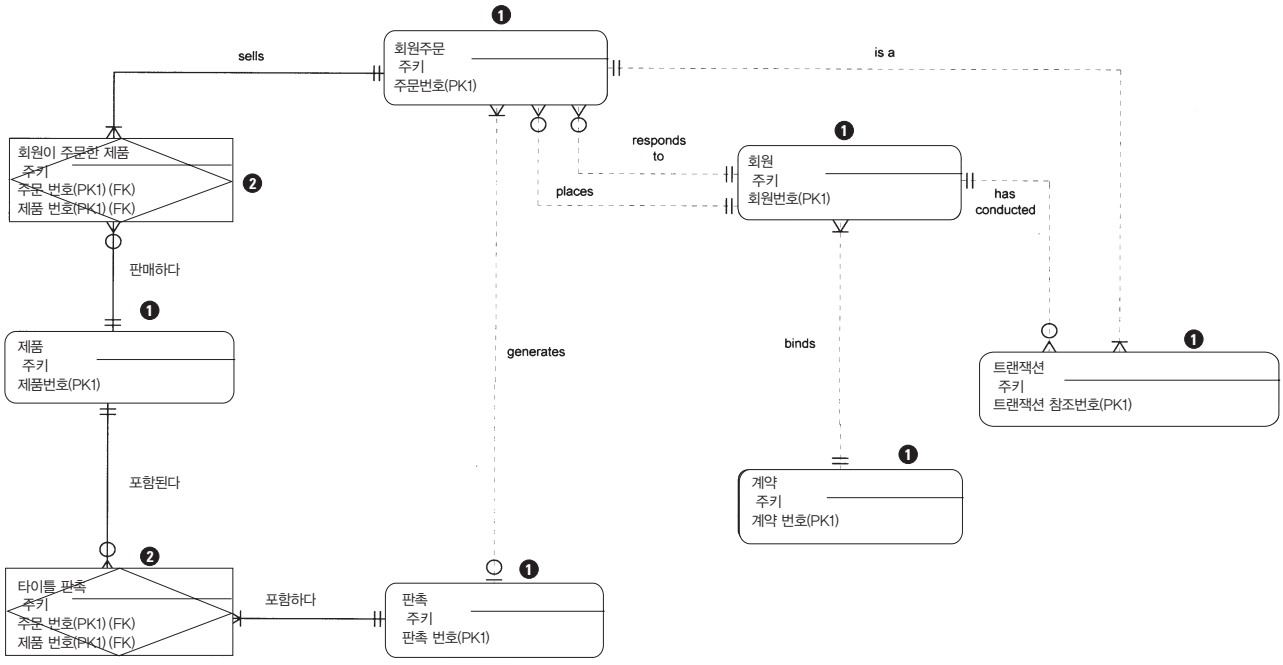


그림 7-14

사운드스테이지의 키 기반 데이터 모델

혹은 그 이상의 개체가 동일한 키를 갖게 된다면, 그들은 동일한 개체일 가능성이 있다.

일반화 계층 구조

이제, 비즈니스 도메인 내의 일반화 계층 구조를 파악하는 것이 유용할 것이다. 이번 장의 앞부분에서 사운드스테이지 프로젝트에 대한 최소한 하나의 상위타입/하위타입 구조를 파악했었고 이를 바탕으로 일반화 계층 구조를 발견하였다. 그래서 키 기반 데이터 모델이 [그림 7-15]와 같이 갱신되었다. 계층 구조 때문에 모델의 배치가 약간 달라졌지만, 앞에서 정의된 관계와 키들은 그대로 유지되었다. 다음 부분에 대하여 주의해서 살펴보기 바란다.

❶ CASE 도구가 일반화 계층 구조 주위에 자동으로 점선 박스를 그렸다.
❷ 하위타입이 상위타입의 키들을 상속받았다.
❸ 판촉(promotion)과 제품(product)의 연결을 끊었는데 그 이유는 판촉이 하위타입인 타이틀과 다시 연결되었기 때문이다. 이것은 잡화(merchandise)는 판촉에 포함되지 않고 단지 타이틀만이 포함된다는 비즈니스 규칙을 정확하게 연관시킨 것이다.

모든 속성이 표현된 데이터 모델

남은 데이터 속성들은 찾아내는 것이 당연한 것으로 보이지만, 분석가는 데이터 모델링에서 자주 부딪치는 문제들에 익숙하지 않다. 이 작업을 완수하기 위해서, 시스템의 데이터 속성에 대하여 깊은 이해를 가져야 한다. 이러한 사실들은 하향식(top-down) 접근법(브레인스

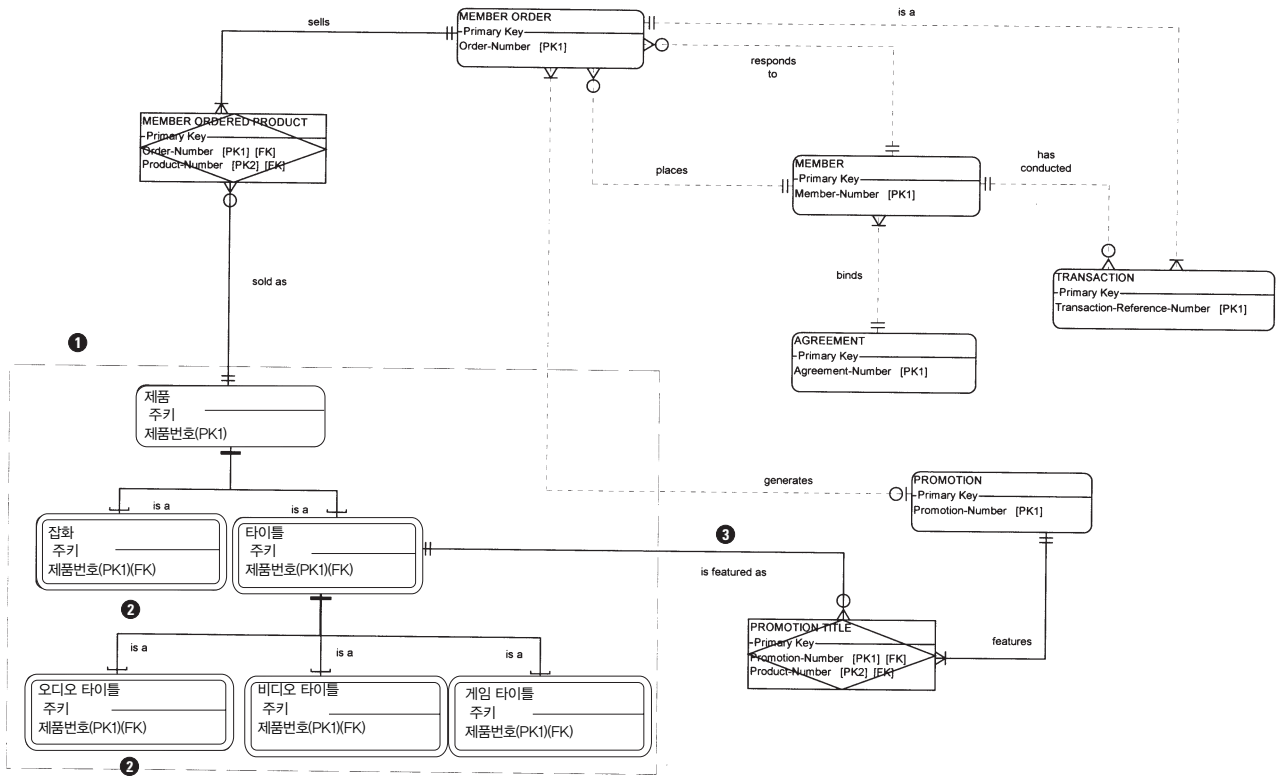


그림 7-15

일반화 계층 구조를 포함한 사운드스테이지의 키 기반 데이터 모델

토밍 같은) 이나 상향식(bottom-up) 접근법(양식이나 파일의 샘플링 같은)을 이용하여 발견할 수 있다. 만약 전사적 데이터 모델이 존재한다면, 일부(아마도 많은) 속성은 이미 파악되어 저장소에 저장되어 있을 것이다.

속성을 파악하는 데 다음과 같은 가이드라인을 참조할 수 있다.

- 많은 조직들은 명칭 부여 표준과 승인된 약어를 가지고 있다. 데이터 관리자가 보통 그러한 표준을 관리한다.
- 속성의 명칭을 주의 깊게 선택한다. 많은 속성들이 이름, 주소, 일자 등과 같은 공통의 기본 명칭을 공유한다. 속성이 상위타입으로 일반화될 수 있지 않은 한, 유일한 이름이 되도록 다음과 같은 변형을 주는 것이 최선의 방법이다.

고객 이름	고객 주소	주문 일자
공급자 이름	공급자 주소	청구 일자
직원 이름	직원 주소	비행 일자

또한, 프로젝트는 다른 프로젝트나 과거 혹은 미래와 단절되어 홀로 존재하지 않는다는 사실을 기억하라. 명칭은 프로젝트들 간에도 구분되어야 한다.

어떤 조직은 이러한 공통의 기본 속성에 대한 재사용 가능한 글로벌 템플릿을 유지하고 있다. 이것은 모든 애플리케이션들 간에 데이터 타입, 도메인 그리고 디폴트 값의 일관성을 높여 준다.

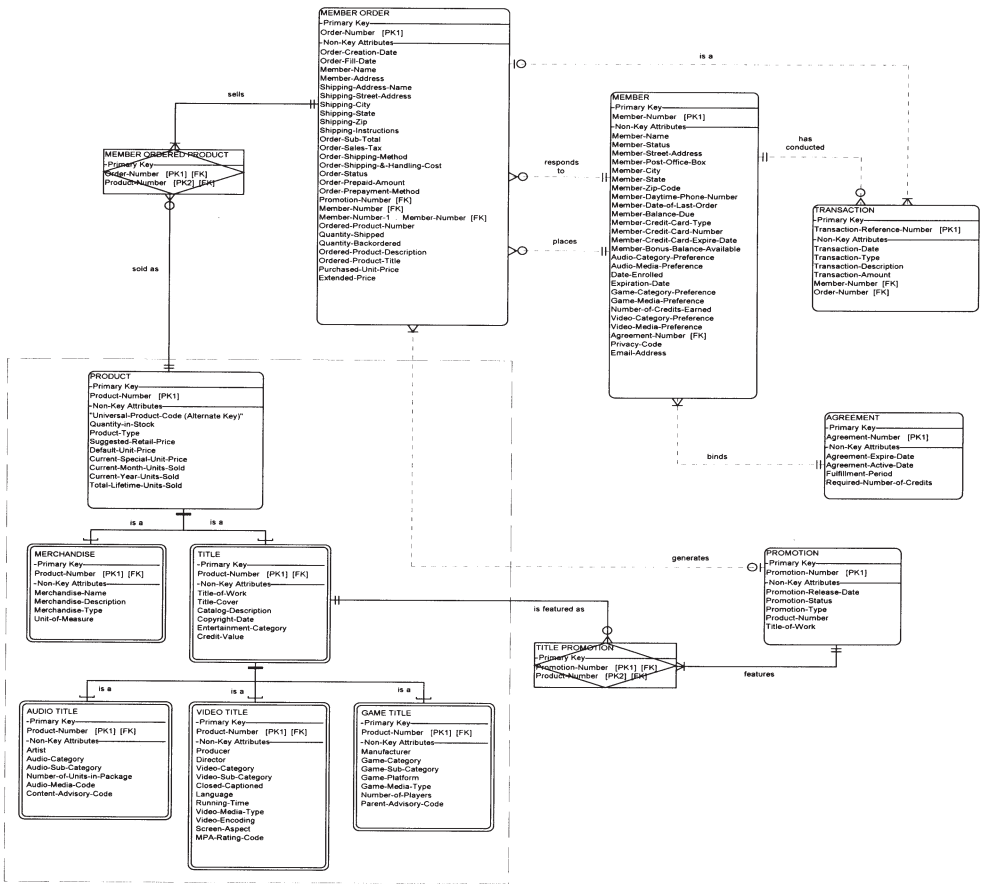
- 기존의 양식과 보고서에 있는 물리적 속성의 이름은 공간 절약을 위하여 자주 축약된다. 논리적 속성의 이름은 좀 더 명확해야 한다. 예를 들어서 주문서 속성인 COD를 논리적 속성으로 동일하게 변환하면 AMOUNT TO COLLECT ON DELIVERY가 되고, QTY는 QUANTITY ORDERED로 변환된다.
- 많은 속성들은 '예/아니오' 값만을 갖는다. 이러한 속성의 이름을 의문형으로 짓도록 하라. 예를 들면, '학위 후보인가?' 라는 속성 이름은 그 값이 예와 아니오 임을 드러낸다.

각 속성은 반드시 하나의 개체에만 대응되어야 한다. 만약 어떤 한 속성이 진짜로 다른 개체들을 설명하고 있다면, 그것은 아마도 몇 개의 서로 다른 속성들일 것이므로 각각에게 유일한 명칭을 부여하라.

- 외래키는 비중복 규칙의 예외이다. 그것은 관련된 개체의 연관된 개체 인스턴스를 식별하게 해준다.
- 속성의 도메인은 논리에 기반할 수가 없다. 예를 들면, 사운드스테이지 사례에서 미디어(media)의 값은 제품의 유형에 따라 달라진다. 만약 제품 유형이 비디오라면, 미디어는 VHS 테이프, 8미리 테이프, 레이저 디스크, 혹은 DVD가 될 것이다. 만약 제품 유형이 오

그림 7-16

사운드스테이지의 모든 속성이 표현된 데이터 모델



디오라면, 미디어는 카세트테이프, CD 혹은 MD가 될 것이다. 가장 좋은 해결책은 오디오 미디어와 비디오 미디어와 같이 각 도메인에 서로 분리된 속성을 배정하는 것이다.

[그림 7-16]은 사운드스테이지 시스템 프로젝트의 정의 단계에서 개체들에게 데이터 속성들을 대응시킨 것을 나타내고 있다. 모든 속성이 표현된 모델이 미래의 데이터베이스에서 포착되고 저장되어야 하는 모든 속성을 나타내고 있지만, 그러한 속성에 대한 설명은 불완전하다. 도메인이 요구된다. 대부분의 CASE 도구들은 저장소에 모든 속성들의 데이터 타입, 도메인 디폴트 값을 설명할 수 있는 다양한 기능을 제공하고 있다. 그리고 나중에 참고하기 위하여 각 속성들은 정의되어야 한다.

데이터 모델의 분석

데이터 모델이 데이터베이스 요구사항을 효과적으로 나타내고 있지만, 그것이 좋은(good) 데이터베이스 설계를 충분히 표현하고 있지는 않다. 그것은 아마도 유연성과 확장성을 축소시키거나, 불필요한 중복을 만드는 구조적 특성을 가지고 있을지도 모른다. 그러므로 데이터베이스 설계와 구현을 위하여 모든 속성이 표현된 데이터 모델을 준비해야 한다.

여기에서는 데이터 모델의 품질 특성에 대하여 살펴보고자 한다. 이것은 이상적인 데이터베이스 구조를 개발할 수 있도록 해줄 것이다. 또한 데이터 모델의 품질을 분석하는 프로세스와 데이터베이스 설계 전에 필요한 수정작업을 하는 것을 알아볼 것이다.

어떤 것이 좋은 데이터 모델인가?

무엇이 좋은 데이터 모델을 만드는가? 다음과 같은 기준을 생각해보자.

- 좋은 데이터 모델은 단순하다. 일반적으로, 어느 주어진 개체를 묘사하는 데이터 속성은 오로지 그 개체만을 설명해야 한다. 예를 들어 다음과 같은 개체의 정의를 살펴보자.

  과정 등록 = 과정 등록 번호(주키) +
  과정 등록 일자 +
  학생 ID 번호(외래키) +
  학생 이름 +
  학생 전공 +
  하나 혹은 그 이상의 과정 번호

  학생 이름과 학생 전공이 진짜로 과정 등록의 개체 인스턴스를 설명하고 있을까? 혹은 그것들이 학생 같은 다른 개체를 설명하는 것은 아닐까? 같은 의문이 학생 ID 번호에도 적용될 수 있겠지만 좀 더 살펴보면 그 속성이 대응하는 학생 개체 인스턴스를 지적하는데 필요함을 알 수 있다. 단순성의 또 다른 측면으로 다음 내용을 살펴보자. 개체 인스턴스의 각 속성은 오직 하나의 값만을 가질 수 있다. 앞의 예를 다시 살펴보면 과정 번호는 한 학생이 선택한 하나의 과정 등록 내에 여러 개의 값이 있을 수 있음을 알 수 있다.

- 좋은 데이터 모델을 필수적으로 비중복적이어야 한다. 이것의 의미는 외래키를 제외한 다른 속성들은 최대한 하나의 속성만을 설명해야 한다는 것이다. 앞의 예에서, 학생 이름과 학생 전공이 학생 개체를 설명하고 있다는 것을 쉽게 알 수 있다. 앞의 설명에 따르면 논리적 선택은 학생 개체가 될 것이다. 데이터 모델에는 약간의 중복이 존재할 수 있다. 예를 들면, 동일한 속성이 다른 이름(이음동의어)으로 한 곳 이상에 저장될 수 있다.
- 좋은 데이터 모델은 유연하고 미래의 요구에 적응할 수 있어야 한다. 이 기준이 없다면, 현재의 비즈니스 요구사항만을 충족시키는 데이터베이스를 설계하려는 경향이 생길 것이다. 그러므로 새로운 요구사항이 알려지면, 개발된 데이터베이스를 사용하는 많은 혹은 모든 프로그램을 재작성하지 않고는 데이터베이스를 변경할 수 없을지 모른다. 많은 프로젝트들이 애플리케이션 지향적인 현실을 바꿀 수는 없지만, 데이터 모델을 가능하면 애플리케이션에 독립적으로 작성하여 현재 프로그램에 영향을 주지 않고 데이터베이스 구조를 확장하거나 변경하게 할 수 있도록 작성해야 한다.

그렇다면 이와 같은 목표를 어떻게 달성할 수 있을까? 어떻게 예상하지 못하는 미래의 요구사항에 적응할 수 있도록 데이터베이스를 설계할 수 있을까? 해답은 데이터 분석에 있다.

데이터 분석

데이터 분석(data analysis)
데이터베이스로 구현하기 위하여 데이터 모델을 개선하는 데 사용되는 기법

정규화(normalization)
데이터를 비중복적이고 안정적이며 유연하고 적응력 있는 개체가 되도록 조직화해서 묶어주는 데이터 분석 기법

1차 정규형(1NF, First Normal Form)
개체의 한 인스턴스에서 한 개 이상의 값을 가진 속성이 없는 개체

2차 정규형(2NF, Second Normal Form)
비주키 속성들이 주키 속성의 전부에 종속된 개체

3차 정규형(3NF, Third Normal Form)
비주키 속성들이 다른 비주키 속성에 종속되지 않는 개체

데이터베이스 설계를 위한 준비에서 데이터 모델을 개선시키는 기법을 **데이터 분석**(data analysis)이라고 한다. 데이터 분석은 단순하고 비중복적이며 유연하고 적응력 있는 데이터베이스 구현을 위한 데이터 모델을 준비하는 프로세스이다. 이러한 기법을 **정규화**(normalization)라고 한다. 정규화는 데이터 속성을 조직화해서 그것들이 비중복적이고 안정적이며 유연하고 적응력 있는 개체가 되도록 묶어주는 데이터 분석 기법이다. 정규화는 3단계 기법으로서 데이터 모델을 1차 정규형, 2차 정규형, 3차 정규형의 형태로 만든다.[2] 이제 이 3가지 형태에 대한 기초 지식을 알아보자.

- 개체의 한 예에 한 개 이상의 값을 가진 속성이 없는 경우 그 개체는 **1차 정규형**(1NF, First Normal Form)이라고 한다. 여러 값을 가질 수 있는 속성은 실제로는 별도의 개체, 혹은 개체와 관계를 설명하는 것이다.
- **2차 정규형**(2NF, Second Normal Form)은 1차 정규형을 만족하고 모든 비주키 속성의 값들이 주키의 일부분이 아니라 주키의 전체에 종속되는 경우이다. 주키의 일부분에만 종속되는 비주키 속성은 그 부분키가 완전키가 될 수 있는 개체로 이동되어야 한다. 이것은 모델에 새로운 개체와 관계를 만들게 할 것 이다.
- **3차 정규형**(3NF, Third Normal Form)은 이미 2차 정규형을 만족하고 비주키 속성의 값들이 다른 비주키 속성에 종속되지 않는 경우이다. 다른 비주키 속성에 종속되는 비주키 속성은 옮겨지거나 삭제되어야 한다. 또다시 새로운 개체와 관계가 데이터 모델에 추가될 수 있다.

2) 데이터베이스 전문가들이 추가적인 정규형을 규명하였다. 3차 정규형은 대부분의 데이터 이상을 제거한다. 추가적인 정규형의 논의는 데이터베이스 교재와 과목에서 다루도록 한다.

정규화의 예

정규화를 하는 수많은 방법이 있다. 여기에서는 이론적이지 않고 수학적이지 않은 방법을 소개하기로 했다. 관계 대수와 같은 이론과 자세한 의미 등은 데이터베이스 과목과 그 과목의 교재를 참고하기 바란다.

단계들을 보이기 위하여 사운드스테이지 사례를 이용하고자 한다. 앞에서([그림 7-16]) 작성된 모든 속성이 표현된 데이터 모델에서 시작하도록 하자. 그것은 정규화된 데이터 모델일까? 아니다. 그 모델에 있는 문제를 찾아보고 단계를 밟아 정규화시켜보도록 하자.

■ **1차 정규형** 데이터 분석의 첫 단계는 각 개체를 1NF로 만드는 것이다. [그림 7-16]에서 어떤 개체가 1NF이 아닐까?

여러분은 회원 주문과 판촉 두 가지를 발견할 수 있을 것이다. 각각은 반복 그룹(repeating group)을 가지고 있다. 그것은 개체의 한 예에 여러 개의 값을 가질 수 있는 속성 그룹을 말한다(괄호로 표시되어 있다). 이 속성들은 그룹으로서 여러 번 반복된다. 예를 들어 회원 주문을 살펴보자. 회원 주문 하나는 여러 개의 제품을 포함할 수 있다. 그러므로 주문된 제품 번호(ordered product number), 주문된 제품 설명(ordered product description), 주문된 제품 타이틀(ordered product title), 주문량(quantity ordered), 발송량(quantity shipped), 이월 주문량(quantity backordered), 구입 단가(purchased unit price), 판매가(extended price) 등의 속성은 회원 주문 의 각 개체 예 안에서 반복된다.

비슷하게, 판촉에서도 하나 이상의 제품 타이틀(product title), 제품 번호(product number), 작업 타이틀(title of work)의 속성들이 반복된다. 이러한 모델에서의 이상을 어떻게 수정할 수 있을까?

[그림 7-17]과 [그림 7-18]은 이 두 개의 개체를 1NF으로 변환하는 것을 보여주고 있다. 원래 개체는 해당 페이지의 왼편에 표시되어 있고 1NF 개체는 오른편에 표시되어있다. 그림을 보면 정규화가 어떻게 데이터 모델과 속성의 배정을 변화시켰는지 알 수 있다. 편의를 위하여 영향을 받은 속성들은 굵은 글씨체의 대문자로 표시하였다.

[그림 7-17]에서 먼저 회원 주문 개체의 하나의 인스턴스에 대하여 하나 이상의 값을 가질 수 있는 속성들을 제거하였다. 이를 통하여 회원 주문은 1NF가 되었다. 그러나 제거된 속성들을 어떻게 하여야 할까? 그것들이 비즈니스 요구사항의 일부분이므로 모델에서 완전히 제거할 수는 없다. 그러므로 회원이 주문한 제품이라는 새로운 개체로 속성 그룹 전체를 옮긴다. 이 속성들의 각 인스턴스들은 단일 회원 주문에 속한 하나의 제품을 설명하고 있다. 그러므로 다섯 개의 제품을 포함한 주문이 있다면, 다섯 개의 회원이 주문한 제품의 개체 인스턴스가 있을 것이다. 각 개체는 각 속성에 대하여 오직 한 개의 값만을 갖는다. 그러므로 새로운 개체 또한 1차 정규형이다.

1NF의 또 다른 예로 판촉 개체에 대한 것이 [그림 7-18]에 주어져 있다. 앞에서와 같이, 반복 속성들을 타이틀 판촉이라는 다른 개체로 옮겼다.

다른 모든 개체들은 반복 그룹을 포함하고 있지 않기 때문에 이미 1NF이다.

■ **2차 정규형** 데이터 분석의 다음 단계는 개체들을 2NF로 만드는 것이다. 모든 개체들을

그림 7-17

1차 정규형(1NF)

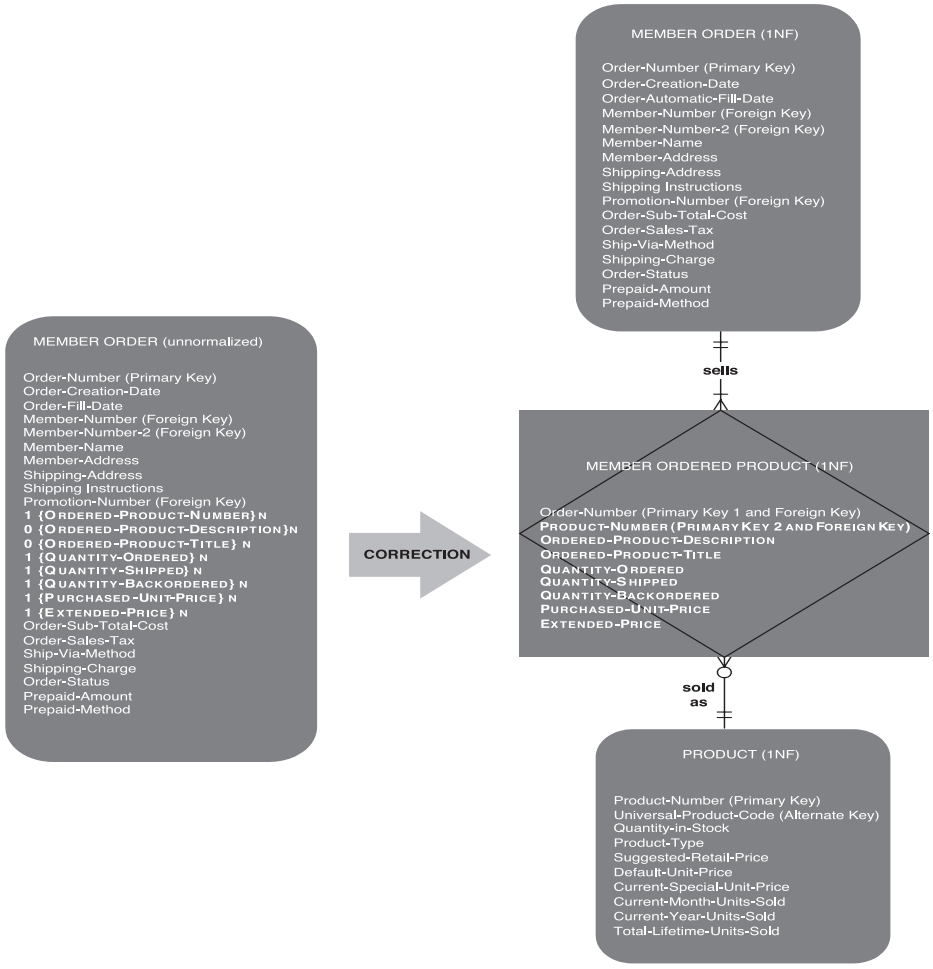
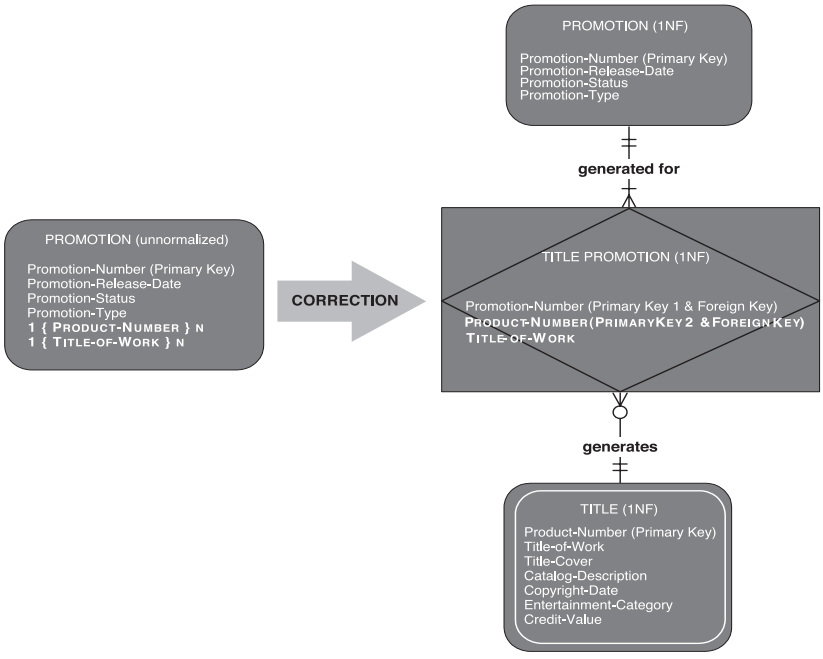


그림 7-18

1차 정규형(1NF)



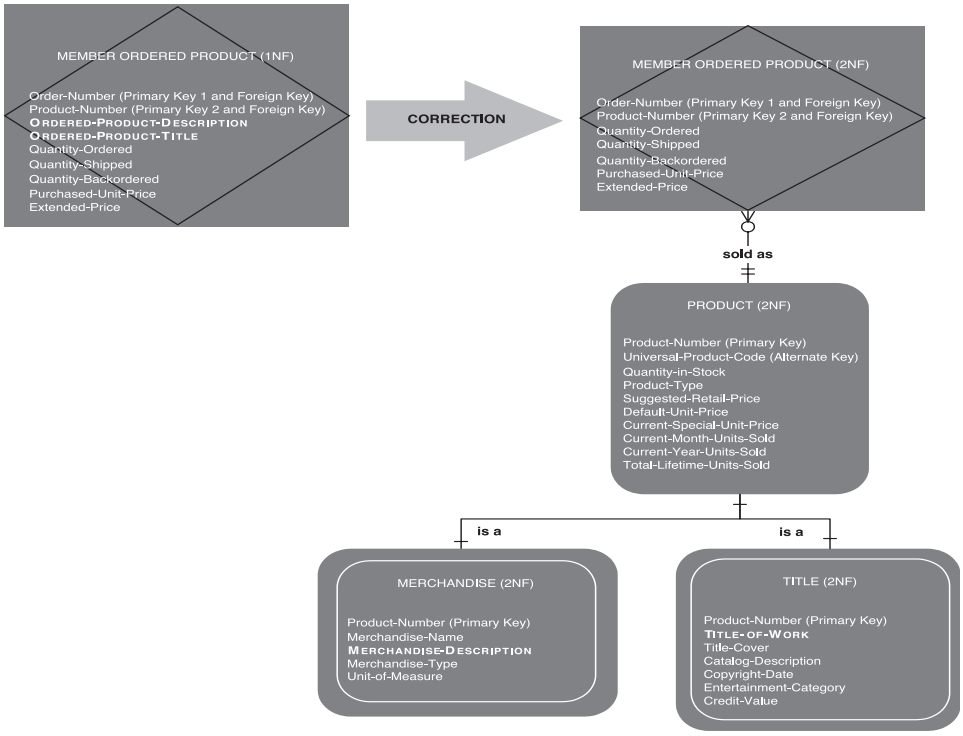


그림 7-19

2차 정규형(2NF)

우선 1NF가 되도록 해야 하는 것을 기억해라. 그리고 2NF는 그 값이 복합키 전체에 종속되지 않고 주키의 일부분에만 종속되는 속성을 찾는 다는 것도 기억해라. 따라서 단일 속성의 주키를 갖고 있는 개체는 이미 2NF이다. 그러므로 제품(하위타입 포함), 회원 주문, 회원, 판촉, 계약, 그리고 트랜잭션은 고려하지 않아도 된다. 그러므로 복합키를 가지고 있는 회원이 주문한 제품과 타이틀 판촉만을 확인하면 된다.

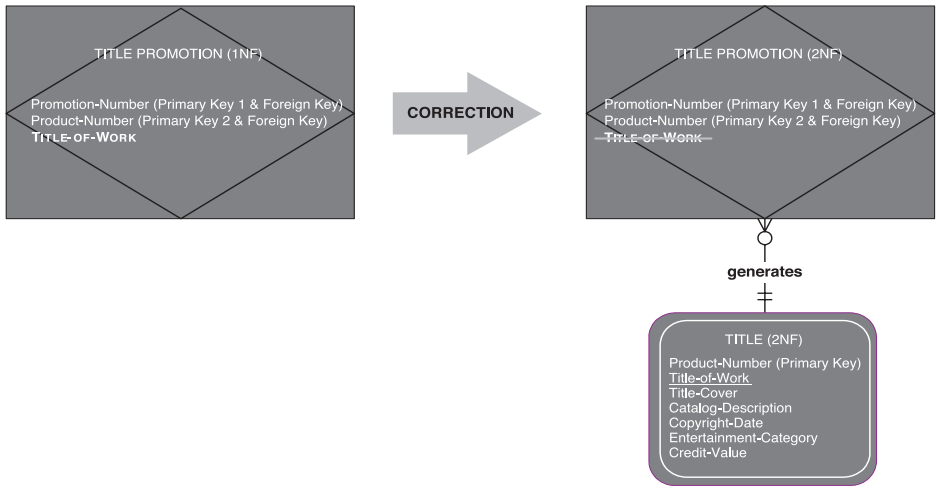
우선 회원이 주문한 제품 개체 먼저 확인하도록 한다. 대부분의 속성은 주키 전체에 종속되어 있다. 예를 들어, 주문량은 주문 번호와 제품 번호를 알지 못하면 어떤 의미도 갖지 못한다. 왜냐하면 주문은 주문에 포함된 제품만큼 여러 개의 주문량을 가질 수 있기 때문에 주문 번호 자체만으로는 부적합하다는 것을 알 수 있다. 마찬가지로, 동일한 제품이 여러 주문에 포함될 수 있기 때문에 제품 번호만으로도 부적합하다. 그러므로 주문량은 키의 양쪽 부분을 모두 요구하고 주키 전체에 종속되어 있다. 발송량, 이월 주문량, 구매단가, 판매가 등도 마찬가지이다.

그러나 주문된 제품 설명과 주문된 제품 타이틀은 어떨까? 각각의 값을 알기 위하여 주문 번호가 필요할까? 아니다. 대신에 이 속성들의 값은 제품 번호의 값에만 종속되어 있다. 그러므로 이 속성들은 주키 전체에 종속되어 있지 않다. 이러한 부분 종속성(partial dependency) 이상을 수정하여야 한다. 이런 형태의 정규화 이상을 어떻게 처리할까?

이 문제를 해결하기 위하여 [그림 7-19]를 보면, 비주키 속성인 주문된 제품 설명과 주문된 제품 타이틀을 제품 번호만을 주키로 갖고 있는 개체로 옮긴다. 필요하다면, 이 개체를 만들어야 하지만 그 키를 가지고 있는 제품 개체가 이미 존재한다. 그러나 제품이 상위타입이기 때문에 조심하여야 한다. 하위타입을 살펴보고, 속성들이 다른 이름이기는 하지만 잡화

그림 7-20

2차 정규형(2NF)



(merchandise)와 타이틀 개체 내에 존재하고 있음을 발견하게 된다. 그러므로 회원이 주문한 제품 개체에서 그 속성들을 옮길 필요 없이 중복된 속성으로 삭제하면 된다.

다음으로 타이틀 판촉 개체를 살펴보자. 복합키는 판촉 번호와 제품 번호의 조합이다. 작업 타이틀은 복합키의 제품 번호 부분에 종속되어 있다. 그러므로 작업 타이틀은 타이틀 판촉에서 제거된다([그림 7-20] 참조). 작업 타이틀은 타이틀 개체 내에 이미 존재하고 그 주키로 제품 번호를 갖고 있다.

■ **3차 정규형** 개체들을 3NF로 만들어서 좀 더 단순화 시킬 수 있다. 3NF 분석을 하기 이전에 개체들은 2NF를 만족하고 있어야 한다. 3차 정규형 분석은 파생 데이터(derived data)와 이행적 종속(transitive dependencies)의 두 가지 문제를 찾게 된다. 두 가지 경우 모두, 기본적인 오류는 비주키 속성이 다른 비주키 속성에 종속된다는 것이다.

파생 속성(derived attributes)
다른 속성으로부터 계산을 통해서 또는 다른 속성의 값으로부터 유도될 수 있는 속성

3NF 분석의 첫 번째 유형은 쉬운데 **파생 속성**(derived attributes)들에 대한 각 개체들을 조사하는 것이다. 파생 속성은 그 값이 다른 속성으로부터 계산을 통하여 산출되거나, 다른 속성 값에 어떤 논리를 적용하여 유도해낼 수 있는 속성을 말한다. 이 점에 비추어 보면 파생 속성을 저장하는 것은 적절치 못하다고 생각할 수 있다. 첫째, 이것은 디스크 저장 공간을 낭비한다. 둘째, 단순해야 하는 갱신작업을 복잡하게 만든다. 왜냐하면 기본 속성 값이 바뀔 때마다 계산을 다시 해서 그 결과 값을 저장해야 하기 때문이다.

예를 들면, [그림 7-21]의 회원이 주문한 제품 개체를 살펴보면, 판매가(extended price)는 주문량(quantity ordered)와 구매 단가(purchased unit price)를 곱하여 구해진다. 그러므로 판매가(비주키)는 비주키 속성인 주문량과 구매단가에 비하여 주키에 덜 종속적이다. 그래서 판매가를 삭제함으로써 개체를 수정하였다.

간단해 보이지만 항상 그런 것은 아니다. 이 규칙을 얼마나 따를 것인가에는 이견이 좀 있다. 일부 전문가들은 이 규칙이 단일 개체 내에서만 적용되어야 한다고 주장한다. 그래서 이 전문가들은 만약 유도를 위해서 요구되는 속성들이 서로 다른 개체에 속해 있다면 파생 속성을 삭제하지 않는다. 우리는 다수의 개체를 포함하고 있는 파생 속성은 한 개체에 속한 속성의 변경에 의하여 촉발되는 데이터 불일치와 다른 개체에 있는 파생 속성의 순차적 갱신을

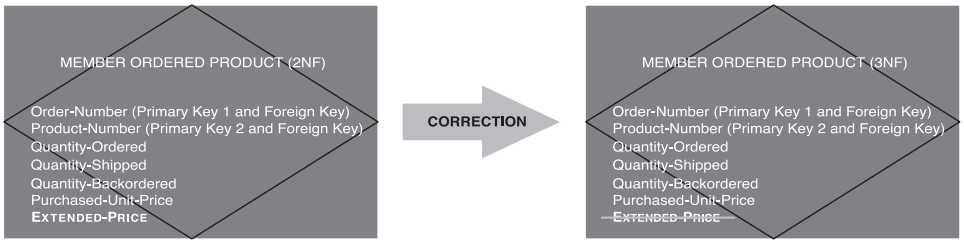


그림 7-21

3차 정규형(3NF)

잊게 되는 등의 위험이 더 커진다는데 동의한다.

3NF 분석의 또 다른 형태는 **이행적 종속**(transitive dependency)이다. 이행적 종속은 비주키 속성이 다른 비주키 속성에 종속되는 경우에 발생한다(유도의 경우를 제외하고. 이 오류는 보통 아직 발견되지 않은 개체가 문제가 된 개체 내에 잠재되어 있다는 것을 나타낸다. 만약 수정되지 않는다면, 만약 새로운 요구사항이 발견하지 못한 개체를 분리된 데이터베이스 테이블로 구현하길 요구하는 경우에 이러한 상태는 유연성과 적응성의 문제를 야기할 수 있다.

이행적 종속(transitive dependency)
비주키 속성의 값이 유도에 의한 경우를 제외하고 다른 비주키 속성의 값에 종속되는 경우

이행적 종속 분석은 복합키를 갖고 있지 않은 개체들에 대해서만 수행된다. 우리의 예에서, 제품, 회원 주문, 판촉, 계약, 회원, 그리고 트랜잭션이 대상이다. 제품 개체에 대해서, 모든 비주키 속성은 오직 주키 속성에만 종속되어 있다. 비슷한 분석에 따라 판촉, 계약, 그리고 트랜잭션이 3NF임을 알 수 있다.

그러나 [그림 7-22]의 회원 주문 개체에서 회원 이름과 회원주소 속성을 살펴보자. 이 속성

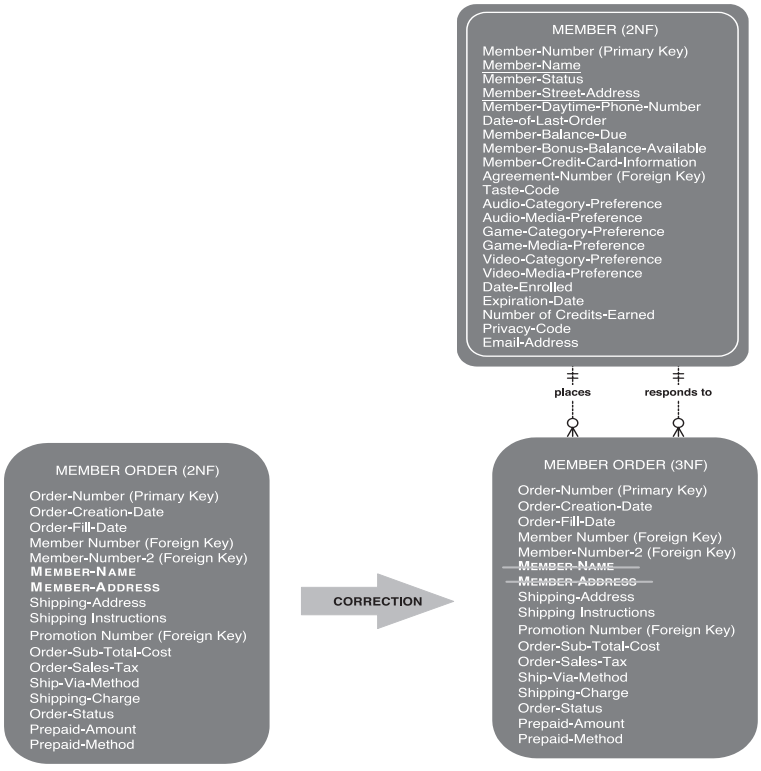
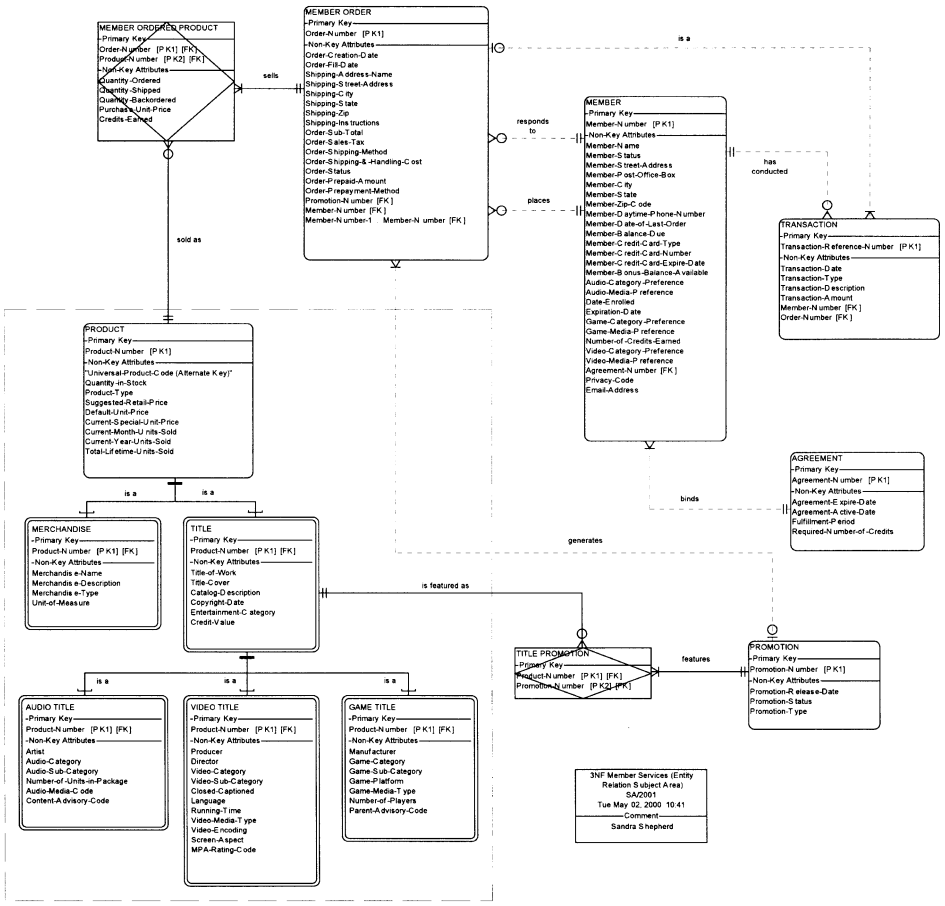


그림 7-22

3차 정규형(3NF)

그림 7-23

사운드스테이지의 3차 정규형 논리적 데이터 모델



들이 주키인 주문 번호에 종속되어 있을까? 아니다. 주키인 주문 번호는 회원 이름과 회원 주소를 결정할 방법이 없다. 대신 회원 이름과 회원 주소의 값은 개체 내에 있는 다른 비주키인 회원 번호에 종속되어 있다.

이 문제를 어떻게 수정할까? 회원 이름과 회원주소는 회원 주문 개체에서 주키가 회원번호만으로 된 개체로 이동되어야 한다. 필요하다면 새로운 개체가 생성되어야 하는데 우리의 예에서는 요구되는 주키를 가진 회원 개체가 이미 존재하고 있다. 실제적으로는 회원 개체에 이미 문제가 된 속성이 있으므로 옮길 필요도 없다. 그러나 회원 주소가 회원 주소-거리명과 동일한 의미임을 알아야 한다. 회원에서 회원 주소-거리명을 사용하기로 결정했다.

3NF을 넘는 정규형이 몇 가지 더 있다. 추가적인 정규형은 데이터 모델을 좀 더 단순하고 중복이 더 적고, 좀 더 유연하게 만들어 준다. 그러나 시스템 분석가들(그리고 대부분의 데이터베이스 전문가들)은 3차 정규형 이상의 데이터 모델을 채택하는 경우가 드물다. 그러므로 정규화에 대한 추가적인 논의는 데이터베이스 교재를 참고하기 바란다.

초반에 여러분이 데이터 모델을 정규화하는 과정은 느리고 지루하다. 그러나 시간을 투자하고 경험이 쌓이면서 이 작업은 점점 빨라지고 일상적인 것이 될 것이다. 노련한 모델 작성자들은 속성을 파악하는 과정에 정규화를 실시함으로써 시간과 노력을 많이 줄이고 있다(그들은 모든 속성이 표현된 데이터 모델을 개발하는 동안에 정규화를 할 수 있다). 1차, 2차, 3

차 정규형을 잘 요약한 다음 문장(출처는 알려지지 않았음)을 기억해 두면 도움이 될 것이다.

> 모든 비주키 속성이 주키에 종속되고, 주키의 모든 부분에 종속되고, 주키에만 종속되면 그 개체는 3차 정규형이다.

■ **검사에 의한 단순화** 정규화는 상당히 기계적인 과정이다. 그러나 이것은 원본 데이터 모델(정규화 이전)의 명칭 일관성에 따라 달라진다. 공통 애플리케이션에 여러 명의 분석자가 작업을 하는 경우, 정규화에 의하여 관리되지 않는 문제들이 발생하는 것은 일반적이다. 이러한 문제들은 미묘한 데이터 중복을 확인하는 것과 같은 노력을 통해서 3NF인 데이터 개체를 더 단순화시키는 과정과 같은, 검사에 의한 단순화를 통하여 가장 잘 해결될 것이다.
최종적으로 정규화된 데이터 모델이 [그림 7-23]에 제시되어 있다.

■ **정규화를 위한 CASE의 지원** 많은 CASE 도구들은 정규화 개념을 지원한다고 말한다. 그것들은 데이터 모델을 읽어서 가능한 정규화 문제들을 분리하려고 한다. 자세히 살펴보면, 많은 CASE 도구들은 단지 1차 정규화 작업만을 수행할 수 있다. 그것들은 다대다 관계를 살펴보고 그 관계들을 연관 개체로 변환한다. 혹은 단일 개체 예에 대해서 여러 개의 값을 가졌다고 특별히 설명된 속성을 찾는다.

CASE 도구가 2차, 3차 정규형 오류를 발견하는 것은 상당히 어렵다. 그것은 CASE 도구가 부분 혹은 이행적 종속성을 인식하는 지능이 있어야 함을 의미한다. 그러한 종속성은 시스템 분석가나 데이터베이스 전문가들에 의한 특별한 조사에 의해서만 발견된다.

데이터 요구사항의 위치별 매핑

논리적 데이터 모델이 어떤 데이터가 새로운 시스템에 저장되어야 하는 가를 표현하는 데는 효과적인 반면에, 위치에 기반한 비즈니스 운영 요구사항은 전달하지 못한다. 어떤 데이터와 접근 권한이 어느 위치에서 요구되는지 파악할 필요가 있다. 특히, 다음과 같은 질문들을 생각해야 한다.

- 각 위치에서 객체와 속성의 어느 부분 집합이 작업을 수행해야 하는가?
- 어느 정도의 접근 수준이 요구되는가?
- 그 위치가 개체 인스턴스를 생성할 수 있는가?
- 그 위치가 개체 인스턴스를 읽을 수 있는가?
- 그 위치가 개체 인스턴스를 삭제할 수 있는가?
- 그 위치가 기존의 개체 인스턴스를 갱신할 수 있는가?

시스템 분석가들은 이러한 논리적 요구사항들은 데이터-위치-CRUD 행렬(data-to-location CRUD matrix)로 정의하는 것이 유용하다는 것을 발견하였다. 데이터-위치-CRUD 행렬은 행은 개체(그리고 가능한 속성)를 나타내며, 열은 위치를 나타낸다. 그리고 셀(행과 열의 교차점)은 접근의 수준을 나타내는데 그 의미는 다음과 같다. C = create(생성), R = read(검

그림 7-24

데이터-위치-CRUD 행렬

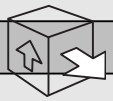
Entity . Attribute / Location	Customers	Kansas City	. Marketing	. Advertising	. Warehouse	. Sales	. A/R	Boston	. Sales	. Warehouse	San Francisco	. Sales	San Diego	. Warehouse
Customer	**INDV**					**ALL**	**ALL**		**SS**	**SS**		**SS**		**SS**
.Customer Number	R				R	CRUD	R		CRUD	R		CRUD		R
.Customer Name	RU				R	CRUD	R		CRUD	R		CRUD		R
.Customer Address	RU				R	CRUD	R		CRUD	R		CRUD		R
.Customer Credit Rating	X					R	RU		R			R		
.Customer Balance Due	R					R	RU		R			R		
Order	**INDV**		**ALL**		**SS**	**ALL**			**SS**	**SS**		**SS**		**SS**
.Order Number	SRD		R	CRUD	R	CRUD	R		CRUD	R		CRUD		R
.Order Date	SRD		R	CRUD	R	CRUD	R		CRUD	R		CRUD		R
.Order Amount	SRD		R	CRUD		CRUD	R		CRUD	R		CRUD		R
Ordered Product	**INDV**		**ALL**		**SS**	**ALL**			**SS**	**SS**		**SS**		**SS**
.Quantity Ordered	SUD		R	CRUD	R	CRUD	R		CRUD			CRUD		
.Ordered Item Unit Price	SUD		R	CRUD		CRUD	R		CRUD			CRUD		
Product	**ALL**		**ALL**	**ALL**	**ALL**	**ALL**			**ALL**	**ALL**		**ALL**		**ALL**
.Product Number	R		CRUD	R	R	R			R	R		R		R
.Product Name	R		CRUD	R	R	R			R	R		R		R
.Product Description	R		CRUD	RU	R	R			R	R		R		R
.Product Unit of Measure	R		CRUD	R	R	R			R	R		R		R
.Product Current Unit Price	R		CRUD	R		R			R	R		R		R
.Product Quantity on Hand	X				RU	R			R	RU		R		RU

INDV = individual **ALL** = ALL **SS** = subset X = no access

S = submit C = create R = read U = update D = delete

색), U = update(갱신), D = delete 혹은 deactivate(삭제). [그림 7-24]는 전형적인 데이터-위치 CRUD 행렬을 나타낸다. 속성의 포함 여부에 대한 결정은 그들이 접근할 수 있는 속성에 제약을 가할 필요가 있는가의 여부에 달려있다. [그림 7-24]는 또한 위치가 개체 인스턴스들의 부분 집합(SS로 표현되어 있음)에만 접근하면 된다는 것을 표현할 수 있는 능력이 있음을 보여주고 있다. 예를 들면, 각 판매 사무소는 그 지역에 속한 고객에 대한 접근만이 필요할 것이다.

일부 방법론과 CASE 도구에서 각 위치에 대한 데이터 모델의 뷰(view)를 정의할 수 있다. 뷰는 단일 위치에서 접근 가능한 개체와 속성만을 포함하고 있다. 만약 뷰가 정의 되었다면, 그것들은 마스터 데이터 모델과 동기화되어 유지되어야 한다(대부분의 CASE 도구들이 이것을 자동으로 수행한다).



요약 *Chapter Review*

1. 데이터 모델링은 시스템의 데이터를 조직화 하고 문서화하는 기법이다. 데이터 모델링은 데이터베이스 모델링이라고도 불리는데, 그 이유는 데이터 모델이 보통 데이터베이스로 구현되기 때문이다.
2. 데이터 모델링에는 여러 표현 방법이 있다. 실제 모델이 데이터를 개체와 그 데이터에 의하여 설명되는 관계로 작성되기 때문에 개체 관계성도(ERD)라고 한다.
3. 개체는 비즈니스에서 데이터를 저장할 필요가 있는 무엇이다. 개체의 클래스는 사람, 장소, 대상, 이벤트, 그리고 개념 등을 포함한다.
4. 개체 인스턴스는 개체 클래스에서 하나의 구성요소이다.
5. 주어진 개체의 각 개체 인스턴스에 대해서 저장하고자 하는 데이터 조각을 속성이라고 한다. 속성은 묘사적인 성질이거나 개체의 특징이다. 일부 속성은 복합속성(compound

attribute)라고 불리는 상위속성으로 논리적 그룹을 형성할 수 있다.

6. 시스템을 분석할 때, 속성에 대해서 적절하거나, 비즈니스 의미에 적합한 속성 값들을 정의하여야 한다. 각 속성의 값들은 데이터 유형, 도메인, 그리고 디폴트 값의 세 가지 성질로 정의된다.
   a. 데이터 유형은 속성에 데이터의 어떤 부류가 저장될 수 있는 가를 정의한다.
   b. 속성의 도메인은 속성의 값들로 어떤 것들이 적합한 가를 정의한다.
   c. 속성의 디폴트 값은 사용자에 의하여 지정되지 않았을 경우에 저장되는 값이다.

7. 모든 개체는 식별자(identifier)나 키를 가져야 한다. 키는 각 개체 인스턴스에 대하여 유일한 값을 갖는 단일 속성이거나, 속성 그룹이다.
   a. 개체의 인스턴스를 유일하게 식별할 수 있는 속성 그룹은 복합 키(concatenated key)라고 한다.
   b. 후보키(candidate key)는 개체의 인스턴스들에 대하여 '주된 식별자가 될 수 있는 후보' 이다.
   c. 주키(primary key)는 하나의 개체 인스턴스를 식별하는데 가장 공통적으로 사용되는 후보키이다.
   d. 주키로 선정되지 않은 후보키는 대체키(alternate key)라고 한다.
   e. 종종 단일 개체 인스턴스 대신에 개체 인스턴스들의 부분 집합을 식별하는 것이 필요할 때가 있다. 부분 집합화 기준(subsetting criteria)은 모든 개체 인스턴스들을 유용한 부분 집합으로 나눌 수 있는 유한한 값을 가진 속성(혹은 복합 속성)이다.

8. 관계(relationship)는 하나 혹은 그 이상의 개체 사이에 존재하는 자연적인 비즈니스 연관이다. 관계는 개체를 연결하는 어떤 이벤트를 표현하거나 개체 사이에 존재하는 논리적 관련성을 나타낸다. 모든 관계들은 양 방향으로 해석될 수 있는 의미의 양방향성을 갖고 있다.

9. 사상수(cardinality)는 연관된 개체의 단일 인스턴스에 대하여 대응하는 다른 개체 인스턴스의 최대, 최소 대응 개수를 정의한다. 모든 관계가 양방향적이므로 사상수는 모든 관계에 대해서 양방향으로 정의되어야 한다.

10. 관계의 차수(degree)는 관계에 참여한 개체의 수를 의미한다. 모든 관계가 이원(binary) 관계인 것은 아니다. 어떤 관계는 동일한 개체 내의 서로 다른 개체 인스턴스 사이에 관계가 존재하는 재귀 관계(recursive relation)일 수 있다. 3원(3-ary 혹은 ternary)관계와 같이 관계는 또한 두 개 이상의 서로 다른 개체 간에 존재할 수 있다.

11. 연관개체(associative entity)는 하나 이상의 다른 개체(부모)로부터 주키를 상속받은 개체를 말한다. 복합키의 각 부분은 각각 연결된 개체의 정확히 한 개체 인스턴스를 가리키고 있다.

12. 외래키(foreign key)는 관계의 인스턴스를 식별하기 위하여 다른 개체에 제공되는(혹은 복사되는) 한 개체의 주키를 말한다. 외래키(항상 자식 개체에 존재한다)는 언제나 부모 개체의 주키와 대응된다.

13. 비식별 관계(nonidentifying relationship)는 참여하는 각 개체가 자신의 독립적인 주키를 가지고 있고, 어떤 주키도 공유되지 않는 관계이다. 불특정 관계의 개체는 강(strong) 혹은 독립(independent) 개체라고 한다. 그 이유는 어떤 개체도 다른 개체에게 식별을 위해서 의존하지 않기 때문이다. 식별 관계(identifying relationship)는 부모 개체가 자신의 주키를 자식 개체의 주키의 일부로 제공하는 것이다. 식별 관계의 자식 개체는 약한(weak) 개체라고 불리는데 식별이 부모 개체의 존재 여부에 의존하기 때문이다.

14. 불특정 관계(nonspecific relationship) 또는 다대다 관계(many-to-many relationship)는 한 개체의 여러 개체 인스턴스가 다른 개체의 여러 개체 인스턴스와 연관된 것을 말한다. 이러한 관계는 초기 데이터 모델에만 적합하며 가능한 빨리 해소되어야 한다.

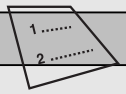
15. 일반화(generalization)는 개체간의 공통점을 발견하고 개발하고자 하는 접근법이다. 이것은 속성을 그룹화하여 상위타입(supertype)과 하위타입(subtype)을 구성하는 방법이다.

16. 논리적 데이터 모델은 다음과 같은 단계를 거쳐서 개발된다.
   a. 개체들이 발견되고 정의된다.
   b. 상황 데이터 모델(context data model)이 작성된다. 상황 데이터 모델은 시스템 소유자와 사용자에 의하여 식별된 비즈니스 개체와 관계만을 포함한다.
   c. 키 기반 데이터 모델(key-based data model)이 작성된다. 키 기반 데이터 모델은 불특정관계를 제거하고 연관개체를 추가한다. 모델의 모든 개체들에게 키가 설정된다.
   d. 모든 속성이 표현된 모델(fully attributed model)이 작성된다. 이 모델은 시스템에 저장되어야 하는 모든 속성을 보여준다.

e. 완전히 설명된 모델(fully described model)이 작성된다. 각 속성이 사전(dictionary)에 정의되고 도메인과 보안과 같은 특성의 용어로 설명된다.

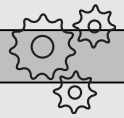
f. 완성된 데이터 모델(completed data model)이 정규화(normalization)라고 불리는 과정을 통하여 적응력과 유연성을 향상시키기 위해서 분석된다. 최종적으로 분석된 모델을 3차 정규형(third normal form) 데이터 모델이라고 한다.

17. 논리적 데이터 모델은 위치에 기반한 비즈니스 운영의 데이터 요구사항을 나타내지는 않는다. 시스템 분석가는 이러한 요구사항을 데이터-위치-CRUD 행렬(data-location-CRUD matrix)의 형태로 정의하는 것이 유용하다는 것을 발견하였다.



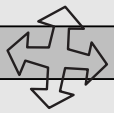
복습문제 *Review Questions*

1. 논리적 모델과 물리적 모델의 차이점은 무엇인가?
2. 시스템의 구현 독립적인 모델을 개발하는 것이 중요한 이유는 무엇인가?
3. 시스템의 구현 종속적인 모델을 개발하는 것이 필요한 이유는 무엇인가?
4. 개체가 무엇인가? 개체 인스턴스가 무엇인가?
5. 관계는 개체간의 자연스러운 비즈니스 연관이다. 학생과 선생님간의 관계는 무엇이 있는가? 그것은 학생이 몇 과목을 수강할 수 있는가 혹은 선생님은 몇 과목을 가르치실 수 있는가와 같은 것에 관련되어 있는가?
6. 사상수는 무엇인가? 예를 드시오.



연습문제 *Problems and Exercises*

1. 학생의 성(last name)을 위한 데이터 속성에 적합한 도메인은 무엇인가?
2. 학생의 성(last name)의 디폴트 값으로 무엇을 사용하겠는가?
3. 성별(gender)의 디폴트 값으로 무엇을 사용하겠는가?
4. 학생 ID, 이름, 전화 번호, 전공 등의 속성을 갖고 있는 학생 테이블이 있다. 3차 정규형으로 만드시오.
5. 영화를 설명하기 위해서 테이블 내에 어떤 속성을 포함시키겠는가?
6. 다대다 관계(불특정 관계라고 불림)는 연관 개체를 갖는 한 쌍의 일대다의 관계로 변환될 수 있다. 이렇게 할 수 없는 경우는 언제인가?
7. 다대다 관계의 예를 하나 드시오. 개체나 연관 개체를 이용하여 해소하시오. 어떤 것을 사용하였는가? 이유는?
8. 처음 세 가지 정규형을 각각 설명하시오. 각각에 대하여 예를 하나씩 제시하라.
9. 한 고객이 신발 가게에 가서 여러 켤레의 신발을 구입하였다. 이 관계를 그림으로 표현하라.
10. 3원 관계, 식별 관계, 비식별 관계에 대해서 각각의 예를 들어라.
11. 표면적으로 데이터 모델링은 창의성을 많이 요구하는 것 같지 않아 보인다. 왜 이것이 틀렸는가?
12. 잘 설계된 데이터베이스가 비즈니스에 전략적 이점을 줄 수 있는가? 어떻게?



프로젝트 및 조사 *Project and Research*

1. 학교 도서관에 가서 대출 데스크의 사서에게 그들이 보유하고 있는 정보를 출력해 달라고 부탁하시오. 어떤 종류의 데이터가 저장되어 있는가? 여러분을 놀라게 하는 것이나 도서를 확인하는데 관련 없는 것들이 있는가? 그렇다면, 왜 그 정보가 수집되었는지 물어보자.
2. 앞의 문제에서 여러분이 발견한 정보로부터 학생 개체에 대한 속성들의 목록을 작성하시오. 그것을 3차 정규형으로 만들자.
3. 식료품점에 가서 상품을 구매해 보시오. 좋은 정보시스템은 이러한 트랜잭션에서 어떤 유형의 데이터를 유지 관리해야 할까? 좋은 정보시스템은 비즈니스를 위해서 무엇을 할까?
4. 위의 문제(식료품점)에 대한 여러분의 해답을 최소한 한 명 이상의 다른 학생의 결과와 비교해보자. 여러분의 결과와 어떻게 다른가?
5. 식료품점에서 사용되는 데이터베이스와 관련된 법적, 사생활 보호 이슈는 어떤 것이 있는가? http://www.findlaw.com에서 주제와 관련된 판결 사례를 찾아보시오. 수업시간에 여러분의 연구 결과에 대하여 짧게(5 페이지 정도) 발표해보자.
6. 데이터베이스와 데이터베이스 내에 저장된 정보가 어떻게 비즈니스에서 전략적 이점을 갖고 활용될 수 있는가? 소그룹으로 나눠서, 원하는 비즈니스를 선택한다. 선택한 비즈니스에 대하여 기존의 문제를 해결할 수 있거나 비즈니스에서 존재하는 기회를 개발할 수 있는 데이터베이스를 생성할 수 있도록 브레인스토밍을 해보시오. 창의적이어야 함을 잊지 말자.

미니케이스 *Minicases*

1. Wow Munchies라는 가상의 온라인 식료품점을 생각해보자. 이 회사는 전국적인 프랜차이즈이며 마케팅, 회계, 선적, 그리고 고객 서비스 부서 등을 완전히 갖추고 있다. CIO는 인터넷 상점에 사용되는 데이터베이스를 갱신해서 서로 다른 부서들을 위한 정확한 트랜잭션과 쇼핑 정보를 수집할 수 있게 만들려고 결정하였다. 이러한 데이터베이스를 지원하는데 사용될 소프트웨어나 하드웨어는 아직 결정되지 않았다.

Step 1. 새로운 데이터베이스에서 어떤 데이터가 중요하게 수집되고 유지 관리되어야 하는가를 어떻게 결정할 것인가? 상세히 결정하라.

Step 2. 조사서와 설문서 등을 만들어서 영향을 받는 부서의 적합한 사람들로부터 의견을 수집한다. 각 부서가 사용하는 양식들을 검토한다.

Step 3. 어떤 응답을 얻었는가? 온라인 식료품점에 가서 경쟁회사들이 수집하는 데이터를 살펴보라. 여러분이 수집한 응답 중 어떤 것들은 부서 담당자와 추가적인 만남이 필요한가? 그렇다면, 조사서와 설문서를 수정하고 다시 인터뷰하라.

Step 4. 위의 질문에서 개요를 파악한 방법을 사용하고, 여러분의 데이터에 포함하여야 할 개체와 속성을 확정한다.

Step 5. 개체간의 관계와 사상수를 결정한다. 어떤 유형의 데이터 모델을 사용하는가? 구현에 특화된 모델 혹은 구현에 무관한 모델? 이유는?

Step 6. 데이터 모델을 수정해서 다대다 관계가 없고 모델이 3차 정규형으로 정규화되도록 한다.

Step 7. 모든 설문서, 조사서, 양식, 그리고 응답들을 최종 데이터 모델과 함께 여러분의 교수님께 제출한다. 개체와 속성 그리고 관계를 도출한 방법에 대한 간략한 설명과 여러분의 데이터 모델에 있는 가정과 제약점 등도 포함하라.

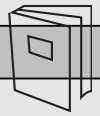
2. 차량 대여점을 분석해 보고 차량 대여를 위한 데이터베이스의 데이터 모델을 작성하시오. 차량 대여에 의하여 영향을 받는 부서는 무엇인가? 공개적으로 사용가능한 양식들을 검토하고, 설문서를 만들어서 데이터베이스가 무엇을 포함해야 하는가를 결정하는데 도움을 줄 수 있는 사람들을 인터뷰하라. 여러분의 데이터 모델과 관련된 모든 문서들을 여러분의 교수님께 제출하라.
3. 마피아를 생각해보자. 이 조직범죄단체는 체포를 피하고 그들의 비즈니스를 영위하기 위하여 데이터베이스를 유지하

고 있다고 가정하자. 그들은 어떤 정보를 유지해야 할까? 그 이유는?

4. 범죄와의 전쟁에서 사용되는 데이터베이스와 관련된 법적, 윤리적, 그리고 사생활 보호와 관련된 이슈는 무엇이 있는가? 여러분의 수업에서 연구 결과를 짧게(10페이지 이내) 발표하자.

팀 및 개인 연습과제 *Team and Individual Exercises*

1. 지리적 그리고 문화적으로 이질적인 팀에서는 프로젝트 관리가 어렵다 각 팀원들에게 서로 다른 시간대와 다른 언어를 사용하는 나라를 배정한다(설령 그 나라의 언어가 그 팀원의 모국어가 아니라 해도). 예를 들면, 미국, 인도, 이스라엘, 파키스탄, 이란, 프랑스, 페루, 일본 등이다.
2. 개인 : 우리가 지닌 창의력의 한계는 우리 자신과 우리의 경험이라는 말이 있다. 그 말은 우리를 만드는 것도 우리의 한계를 만드는 것도 우리 자신이라는 말이다. 창의적이라는 것은 어떤 의미일까? 어떻게 하면 창의적이 되고 자유롭게 생각할 수 있을까?
3. 개인/팀 : 공통적인 수명 프로세스를 재설계해보시오. 만약 '하늘이 제약요소' 라면 여러분은 어떻게 이 프로세스를 바꾸겠는가? 예전 프로세스의 각 단계를 확인하고 그 후에 새로운 프로세스에서의 각 단계를 설명해보자(예를 들면, 화장을 한다. 집안 청소를 한다. 식료품점에 간다. 수업에 간다). 토론을 위해서 여러분의 노트를 가져오자.



읽어볼 만한 추천자료 *Suggested Readings*

- Bruce, Thomas A. *Designing Quality Databases with IDEF1X Information Models*. New York: Dorset House Publishing, 1992. 우리의 데이터베이스 분석 및 설계 과목의 교재로 이 책을 사용하고 있다. IDEF1X는 데이터 모델링에 있어서 풍부하고 표준화된 표현방법을 제공하고 있다(Bruce는 정보 모델링이라고 한다). 그래픽 언어는 달라 보이지만 우리 교재에 사용된 것과 동일한 시스템 개념을 나타내고 있다. 그 언어는 최소한 두 개의 CASE도구에서 지원하고 있다. Logic Works의 ERwin과 Popkin의 System Architect. 책에는 두 개의 사례 연구가 실려 있다.
- Hay, David C. *Data Model Patterns: Conventions of Thought*. New York: Dorset House Publishing, 1996. 아주 훌륭한 책이다. 이 책은 대부분의 비즈니스 데이터 모델은 어떤 기본적이고 반복되는 패턴에서 추출된다는 가정에서 출발한다. CASE 공급자들이 이러한 패턴을 CASE 도구 내에 재사용 가능한 패턴으로 포함시키면 어떨까?
- Martin, James, and Clive Finkelstein. *Information Engineering*, 3 vols. New York: Savant Institute, 1981. 정보 공학은 정형화되고, 데이터베이스와 4세대 언어 중심의 방법론이다. 정보 공학의 그래픽 데이터 모델링 언어는 우리의 것들과 유사하다. 데이터 모델링은 볼륨 I과 II에 실려 있다.
- Reingruber, Michael, and William Gregory. *The Data Modeling Handbook*. New York: John Wiley & Sons, 1994. 데이터 모델링에 대한 아주 훌륭한 책이며, 데이터 모델의 품질과 정밀성에 대해서 아주 실제적인 도움을 줄 수 있다.
- Schlaer, Sally, and Stephen J. Mellor. *Object-Oriented Systems Analysis: Modeling the World in Data*. Englewood Cliffs, NJ: Yourdon Press, 1988. 제목은 잊어라. '객체지향' 에 대해 1988년에 생각하는 것과 오늘날은 다르다. 하지만, 이 책은 데이터 모델링 주제에 대해서 가장 쉽게 읽을 수 있는 책 중의 하나다.
- Teorey, Toby J. *Database Modeling & Design: The Fundamental Principles*, 2nd ed. San Francisco: Morgan Kaufman Publishers, 1994. 이 책은 목록에 있는 다른 책에 비하여 좀 더 개념적이다. 하지만 데이터 모델링 실무에 유용한 통찰력을 제공한다.