제 6 장 관계형 데이터베이스 설계

이 장에서는 개념적 설계에서부터 관계형 데이터베이스의 논리적 설계에 이르기까지의 전 과정을 절차적으로 소개한다.
개념적 설계를 위해 2장에서 이미 소개한 개념적 모델인 ER 모델을 이용한다. 우리는 2장에서 다룬 개념적 설계 단계에서 실세계를 대상으로 실세계를 구성하는 구성요소, 구성요소 특성, 구성요소들 사이에 존재하는 관계 등을 표현하는 개념적 모델링에 대하여 이미 다루었다.

개념적 설계를 기반으로 하는 관계형 데이터베이스의 논리적 설계란 개념적 설계로부터 도출한 결과를 가지고 관계형 릴레이션들을 도출해내는 과정이다.

5장의 데이터 이상과 정규화에서 관계형 데이터베이스의 논리적 설계에 관한 정확한 언급은 없었지만 사실 상 정규화 과정은 관계형 데이터베이스의 논리적 설계의 일부분이다.

그러나, 개념적 모델링에 의해 도출된 결과를 이용하여 관계형 데이터베이스의 릴레이션들을 도출하는 방법에 대해서는 아직 소개되지 않았기 때문에 이에 대한 방법이 이 장에서 소개된다.

그러므로 개념적 설계 결과를 토대로 관계형 데이터베이스의 릴레이션들을 도출하는 과정을 제외하고는 개념적 설계 과정과 정규화 과정을 통한 논리적 설계 과정 일부에 대하여 우리는 이미 다룬 것이다.

그럼에도 불구하고 여러분들은 개념적 설계를 어떻게 행하고 그 설계 결과를 어떻게 관계형 데이터베이스의 논리적 설계에 이용하며 도대체 정규화를 논리적 설계 시 어디에 이용하는지에 대해 의아해 할 것이다. 이러한 의구심은 당연한 것이며 이 장의 내용은 이러한 의구심을 해결함과 아울러 개념적 설계 과정과 관계형 데이터베이스의 논리적 설계 과정의 단계적 흐름을 일관성 있게 여러분들에게 제시하는데 그 목적을 두고 있다.

따라서 여러분들은 개념적 설계 과정과 관계형 데이터베이스의 논리적 설계 과정과의 연관성을 주시하면서 이 장의 내용에 집중해 주길 바란다.

6.1 데이터베이스 설계 고려 사항

2 장에서 이미 설명한 바와 같이 데이터베이스 설계는 개념적 설계, 논리적 설계, 그리고 물리적 설계로 분류되며 설계 절차 또한 언급된 순서대로 행한다. 본 강의에서는 관계형 데이터베이스 설계에 초점을 맞추고 있기 때문에 이와 관련된 개념적 설계와 논리적 설계시의 고려사항에 대하여 살펴보고자 한다.

6.1.1 개념적 설계 고려사항

개념적 설계 단계는 실세계로부터 실세계를 구성하는 구성요소, 구성요소 특성, 구성요소들

사이에 존재하는 관계 등을 표현하는 단계이기 때문에 이와 관련하여 고려해야 할 사항들을 살펴보면 다음과 같다.

(1) 데이터베이스화 할 실세계에 대한 정의가 정확히 이루어져야 한다.

요구사항에 따라 데이터베이스화 할 실세계의 영역을 정확히 구분하고 그 실세계가 정확히 무엇을 하는 시스템인지, 또한 어떻게 구성되어 있는지 등에 대해 정확히 파악해야 한다. 이와 같이 데이터베이스화 할 실세계를 명확히 구분하고 그 기능과 역할 또는 추구하는 목표를 정확히 파악할 때 비로소 후속 설계 고려사항인 실 시스템을 구성하는 구성요소, 각 구성요소의 속성, 구성요소들 간의 관계, 실세계 내에 존재하는 제약사항에 대한 식별이 명확하게 이루어질 수 있다. 데이터베이스화 할 실세계에 대한 정의가 불명확해짐에 따라 데이터베이스화 할 대상이 모호해지며 따라서 개념적 설계의 후속 진행 절차를 수행하기가 어렵게 된다.

(2) 실세계를 구성하는 구성요소들은 개체 집합 차원에서 식별되어야 하며 개체 집합 차원에서 식별된 구성요소들은 데이터베이스로의 반영 여부를 판단하여 선택적으로 개념적 설계에 반영하도록 한다.

실세계는 구성요소들로 이루어지며 이 구성요소들 사이의 동적·정적 관계에 의해 그 기능과 역할을 수행한다. 실세계의 각 구성요소는 자체적으로 정보를 포함하고 있을 뿐만 아니라 관계성을 이루는 기본이 된다. 따라서 몇몇 구성요소들이 개념적 설계에 반영되지 않게 되면 이들과 관련된 구성요소들 사이의 관계성 정보도 반영되지 않게 된다. 이러한 정보가 개념적 설계 결과를 바탕으로 이루어지는 논리적 설계에서도 반영될 수 없기 때문에 결국 실세계 정보를 정확히 반영하지 못하는 데이터베이스를 생성하는 결과를 가져온다.
실세계의 구성요소는 개체 차원이 아닌 개체 집합 차원에서 식별되어야 한다. 개체(entity)란 실세계를 구성하면서 독립적으로 구별되는 물리적 또는 논리적 구성요소이며, 개체 집합(entity set)이란 동종의 개체들이 모여서 이루어진 개념적 집단이다. 일반적으로 우리가 명사를 고유명사, 보통명사, 추상명사, 물질명사 등으로 구분할 때 개체는 고유명사를 이용하여 지칭하고 개체 집합은 보통명사를 가지고 그 집합을 지칭한다. 개체와 개체 집합 사이의 차이점을 설명하기 위해 '홍길동'과 '사람' 사이의 차이점을 예로 들어보자. 먼저 '홍길동'이란 고유명사이고 실세계에 존재하는 독립적 실체로서 개체에 해당하며 '사람'이란 보통명사로서 '홍길동'을 포함함과 동시에 '홍길동'과 같은 특성을 갖는 모든 인간들의 집합적 개념으로서 개체 집합에 해당한다. 실세계의 구성요소가 개체 집합 차원에서 식별되어야 한다는 것은 개체 집합을 구성하는 개체들까지 모두 표현해야 하는 것을 의미하는 것은 아니다. 개념적 설계에서 개체 집합은 개체 집합의 명칭을 가지고 표현되며 이를 구성하는 개개의 개체들은 표현하지 않는다. 다만, 개념적 설계는 개체 집합들을 중심으로 이들간의 관계성, 각 개체집합의 속성, 이들 사이에 부여되는 제약조건 등을 표현함으로써 이루어진다.
개체들을 표현하지 않는다고 해서 개념적 설계에서 이들이 전혀 무시되는 것은 아니다. 예를 들어 개체 집합들 사이의 관계성(relationship)과 이 관계성의 대응 카디널리티(cardinality)를 식별하려면 실제적으로 이들 개체 집합에 속할 수 있는 임의의 개체들이 고려하게 된다.
개체 집합 차원에서 구성요소가 식별되었다 해서 이들 모두를 개념적 설계에 반드시 반영할 필요는 없다. 그 이유는 식별된 개체 집합이 데이터베이스에 반영할 정도로 중요하지 않거나

혹은 데이터베이스에 반영하지 않더라도 아무런 문제가 야기되지 않는다면 개념적 설계에서 제외할 수도 있다.
예를 들어 대학 실세계에서 단과대학과 단과대학에 소속된 컴퓨터의 두 개체 집합과 이들 사이에 존재하는 소속 관계성의 표현을 선택함에 있어서 만약 단과대학에 대한 정보는 관심의 대상인 반면 컴퓨터 및 이에 관련된 정보(즉, 컴퓨터 개체 집합과의 관계성을 통하여 얻을 수 있는 정보)는 관심 밖이라면 비록 컴퓨터 개체 집합과 소속 관계성이 각각 대학 실세계를 이루는 구성요소와 관계성이라 하더라도 개념적 설계에서 제외될 수 있다. 그 이유는 데이터베이스 구축 후 검색하지도 않을 불필요한 정보를 데이터베이스화함으로써 저장 공간의 낭비를 굳이 초래할 필요가 없기 때문이다.
개념적 설계 후 후속적인 논리적 설계 및 구현을 통해 데이터베이스가 생성되면 개체 집합에 속한 각 개체는 자신을 나타내는 데이터를 사용자가 삽입함으로써 데이터베이스에 반영된다.

(3) 실세계를 구성하는 개체 집합의 속성은 충분히 식별되어야 하며 식별된 속성들은 데이터베이스로의 반영 여부를 판단하여 선택적으로 그 개체 집합에 할당되도록 한다.

실세계로부터 식별한 각 개체 집합을 대상으로 그 개체 집합이 갖는 속성들을 가능한 한 충분히 구분하고 식별한다. 개체 집합의 속성들은 설계를 통해 데이터베이스를 구축한 후 개체 집합에 삽입되는 개체에 대한 정보를 나타내는데 사용된다. 즉, 개체에 부여되는 속성값들이 바로 그 개체에 대한 정보이다.
식별된 모든 속성이 개념적 설계에 반드시 반영되어야 하는 것은 아니다. 데이터베이스에 반영할 필요나 가치가 없는 속성들은 개념적 설계에 반영될 필요가 없다. 예를 들어 한 회사의 회사원 집단 개체에 대한 속성에서 손 모양에 대한 정보는 일반적으로 매우 불필요한 정보일 수 있다. 이 정보가 반드시 필요하고 이용되는 것이라면 개념적 설계에서 반영되어야 하지만 만약 향후에 이용되지 않는다면 이러한 속성 표현은 개념적 설계에서 제외되어야 한다. 그러므로 필요한 정보를 포함하는 속성들은 반드시 관련 개체 집합에 할당되도록 한다.

(4) 실세계 개체집합들 사이에 존재하는 관계성 집합이 정확히 식별되어 되어야 하며 식별된 관계성 집합들은 데이터베이스로의 반영 여부를 판단하여 선택적으로 개념적 설계에 반영되도록 한다.

실세계 구성요소들 사이의 관계 역시 개체 차원이 아닌 개체집합 사이의 관계성 집합(relationship set)차원에서 식별되어야 한다. 관계성 집합의 식별은 개체 집합들에 속한 임의의 개체들을 간에 존재하는 관계성(relationship)을 중심으로 이루어진다. 관계성 집합을 표현할 때 이에 속한 모든 관계성(relation)을 일일이 표현하는 것은 아니다. 단지 개체 집합들간의 관계성 집합을 식별할 때 이에 속할 수 있는 개체들 사이의 관계성이 고려될 뿐이다.
데이터베이스 구축 후 개체들간의 관계 정보 역시 데이터로 사용자에 의해 삽입된다. 따라서 개념적 설계에 개체 집합 사이의 관계성 집합은 데이터베이스 구축 후 관계성 집합에 관여하는 개체들 사이의 관계성 정보가 데이터베이스에 데이터로 반영되는 점을 주시하자.
관계성 집합이 모두 식별되었다 해서 이들이 모두 개념적 설계에 반영되는 것은 아니다. 개체 집합 및 속성과 마찬가지로 관심의 대상이 아니거나 데이터베이스로 반영할 가치가 없는 관계성 집합은 개념적 설계에서 제외하도록 한다.

(5) 관계성 집합의 속성들이 충분히 식별되어야 하고 식별된 속성들은 데이터베이스로의 반영 여부를 판단하여 선택적으로 그 개체 집합에 할당되도록 한다.

실세계로부터 식별한 각 관계성 집합을 대상으로 그 관계성 집합이 갖는 속성들을 가능한 한 충분히 구분하고 식별한다. 앞에서 설명했지만 속성에 대응하는 속성값은 개체에 대한 정보를 나타낸다. 따라서 개체에 대한 중요한 정보를 나타내는 속성이 개념적 설계에 반영되지 않는다면 데이터베이스 구축 후 활용에서 그 정보는 알 수 없게 된다.
식별된 모든 속성이 개념적 설계에 반드시 반영되어야 하는 것은 아니다. 데이터베이스에 반영할 필요나 가치가 없는 속성들은 개념적 설계에 반영될 필요가 없다.

(6) 실세계에 존재하는 제약 사항들이 충분히 식별되어 설계에 반영되어야 한다.

실세계에는 여러 가지 규칙, 즉, 제약 조건들이 존재한다. 예를 들어 대학교에는 학칙이 존재하고 국가에는 법이 존재한다. 이러한 규칙들은 실세계 개체 집합에 속하는 개체들의 자격 조건을 제한하고 개체들 간의 관계를 제한한다. 이러한 규칙들이 개념적 설계에 반영되어야 한다. ER 모델은 제약 사항들을 표현하는데 한계가 있다. 따라서 ER 모델을 이용하여 개념적 설계를 행할 때 ER 모델로 표현이 불가능한 제약 사항들은 문서화하여 논리적 설계와 구현 단계에서 이 제약 사항들이 반영되도록 해야 한다.

제약 사항들은 3 장에서 소개한 무결성 제약조건 표현 방법(도메인 제약조건, 참조무결성, assertion, trigger)을 통하여 논리적 설계와 데이터베이스 구현에서 반영된다. 특히 도메인 제약조건이나 참조무결성을 제외한 나머지 제약조건들은 트랜잭션(데이터베이스를 접근하는 특별한 성질을 갖는 프로그램)이나 응용 프로그램에 반영된다.

6.1.2 논리적 설계 고려사항

개념적 설계가 실세계를 인간의 개념적 입장에서 표현하는 과정이라면 논리적 설계는 DBMS (데이터베이스 관리 시스템)가 사용하는 데이터 모델 측면에서 표현하는 과정으로 논리적 설계에서 고려해야 할 사항들을 살펴보면 다음과 같다.

(1) 논리적 설계는 개념적 설계 결과를 정확히 반영해야 한다.

일반적으로 논리적 설계는 개념적 설계 결과를 기초로 수행하기 때문에 논리적 설계는 개념적 설계 결과를 정확히 반영해야 한다.
개체 집합, 속성, 관계성 등이 정확히 논리적 데이터베이스 스키마로 반영되어야 한다. 관계형 데이터베이스에서 개체 집합과 관계성 집합 모두는 릴레이션으로 표현된다. 또한 개념적 설계에서 표현된 속성은 릴레이션을 구성하는 속성으로 표현된다.
제약조건은 릴리이션 스키마 선언이나 트랜잭션 설계, 혹은 응용 프로그램 설계에서 반영되도록 해야 한다. 일반적으로 트랜잭션은 릴레이션을 이용하기 때문에 논리적 데이터베이스 설계 이후 세부적으로 설계된다.

(2) 데이터 이상을 최소화하도록 설계되어야 한다.

데이터 이상은 데이터 무결성에 손상을 주기 때문에 최소화 되어야한다. 데이터 이상을 최소화시킴으로써 데이터 중복성의 최소화와 참조 무결성의 극대화를 기할 수 있다. 관계형 데이터베이스에서 데이터 이상을 최소화하기 위해 각 릴레이션 별로 정규화 (normalization)가 이루어져야 한다.

6.2. 관계형 데이터베이스 설계 단계

실세계를 데이터베이스로 반영하기 위한 설계 절차는 개념적 설계, 논리적 설계, 물리적 설계 순으로 이미 2 장에서 설명되었다. 또한 ER 모델을 이용한 개념적 설계 방법에 대해서도 이미 소개되었다.
이 장에서는 ER 모델을 이용한 개념적 설계 절차를 포함하여 도출된 개념적 설계 결과를 관계형 데이터베이스 스키마로 논리적 설계하는 절차와 이 절차를 구성하는 각 단계를 그림 6.1.과 같이 실세계 정의, 개념적 설계, 논리적 설계 부분으로 나누어 각 단계에서 행하는 작업들을 자세히 소개하고자 한다.

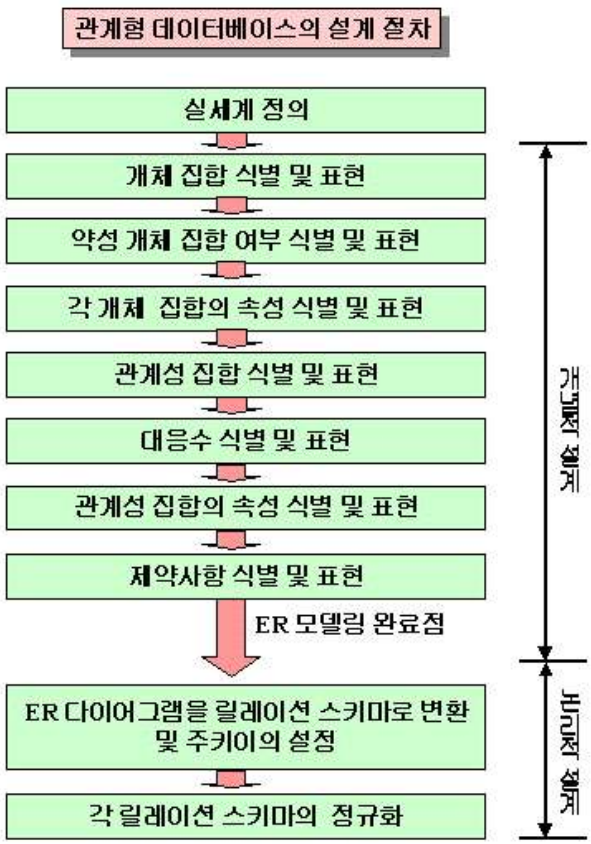

그림 6.1 관계형 데이터베이스의 설계 절차

6.2.1 실세계 정의

실세계 정의는 실세계를 시스템적 측면에서 식별하는 과정으로 실세계를 구성하는 구성요소, 각 구성요소의 특성, 구성요소들간의 관계, 존재하는 제약 조건들을 표현한다.

단계 1. 실세계를 시스템 측면에서 문장으로 자연스럽게 정의한다.

실세계를 문장으로 나타내는 단계이다. 실세계의 구성요소들과 이들간의 관계를 자연스럽게 문장으로 표현한다. 꼭 필요한 구성요소 속성도 문장으로 언급하도록 한다. 또한, 실세계에 존재하는 규칙 혹은 제약조건들도 문장으로 표현하도록 한다. 이 규칙 혹은 제약조건에는 대응 카디널리티(cardinality)에 대한 내용도 포함되어야 한다.
실세계 정의는 개체(entity)나 관계성(relationship) 차원이 아닌 개체 집합(entity set)과 관계성 집합(relationship ser) 차원에서 문장으로 서술한다. 이렇게 개체 집합과 관계성 집합 차원에서 서술하는 이유는 논리적 설계의 릴레이션은 개체 차원에서 이루어지지 않고 개체 집합 차원에서 이루어지기 때문이다. 실세계나 개념적 설계의 개체 집합과 관계성 집합은 논리적 설계에서 릴레이션으로 표현된다.
다음은 실세계 정의 단계에서 문장으로 자연스럽게 표현되어야 할 요소들이다.

- 실세계를 구성하는 구성요소 및 구성요소의 특성
- 구성요소들간의 관계
- 실세계에 존재하는 제약조건

실세계를 문장으로 표현하는 이유는 이 문장을 바탕으로 하면 개념적 설계가 더 수월해지기 때문이다.

6.2.2 ER 모델을 이용한 개념적 설계

ER 모델을 이용한 개념적 설계 부분은 이미 2 장에서 구체적으로 설명되었으므로 이 부분을 다시 참고하길 바란다. 이 장에서는 설명 없이 단계만 제시하기로 한다. ER 모델을 이용한 개념적 설계는 단계 1.의 실세계 정의를 바탕으로 문장 하나 하나를 파악함으로써 행한다.
일반적으로 문장 중에 나타나는 명사들은 대부분 개체 집합을 나타내며 동사를 비롯한 술어들은 관계성 집합을 나타낸다. 명사를 수식하는 형용사 등의 수식어는 일반적으로 속성에 해당한다. 이러한 점을 유념하면서 다음 단계들을 행한다.

단계 2. 실세계를 구성하는 개체 집합을 식별하고 표현한다.

단계 3. 각 개체 집합이 강개체 집합인지 약개체 집합인지를 구분하고 표현한다.

단계 4. 각 개체 집합의 속성을 식별하고 표현한다.

단계 5. 실세계로부터 개체 집합들 사이에 존재하는 관계성 집합을 식별하고 표현한다.

단계 6. 실세계 정의의 제약사항으로부터 관계성 집합에 관련된 개체 집합들 사이의 대응 카디널리티(cardinality)를 식별하고 표현한다.

단계 7. 관계성 집합이 갖는 속성들을 식별하고 표현한다.

단계 8. 실세계 정의로부터 제약 사항을 식별하여 ER 모델로 반영하거나 ER 모델로 반영할 수 없는 제약 사항은 문장으로 정리하여 나열한다.

실세계 정의 문장 중에서 제약사항에 해당하는 부분을 식별해야 한다. 식별된 제약사항들을 모두 ER 모델로 반영할 수 있는 것은 아니다. 단계 6의 대응 카디널리티는 ER 모델로 표현할 수 있지만 개체가 개체 집합에 속할 조건이나 참조 무결성을 유지하기 위한 조건, 속성 자체나 속성들 간에 존재하는 조건 등은 ER 모델로 표현할 수 없으며 이 조건들은 릴레이션 스키마 선언이나 트랜잭션, 혹은 응용 프로그램에 반영되어야 할 부분이다. 이 제약조건들을 반영하기 위해 문장으로 정리하도록 한다.

6.2.3 관계형 데이터베이스의 논리적 설계

논리적 설계는 개념적 설계 결과로서 도출된 ER 다이어그램을 기반으로 행한다. 관계형 데이터베이스의 논리적 설계는 ER 다이어그램을 관계형 데이터베이스의 릴레이션으로 변환하고 변환된 각 릴레이션을 대상으로 정규화를 거침으로써 이루어진다. 이에 대한 자세히 설명하면 다음과 같다

.

단계 9. ER 다이어그램을 릴레이션으로 변환하고 변환된 릴레이션에 대한 주키를 설정한다.

ER 다이어그램을 릴레이션으로 변환하는 방법과 각 릴레이션의 주키 설정은 다음의 규칙들에 따라 행하며 다음 그림 6.2를 중심으로 각 규칙들을 자세히 설명하도록 한다.
그림 6.2에서 실선으로 밑줄 친 속성은 해당하는 강개체 집합의 주키를 나타내고 점선으로 밑줄 친 속성 A21은 해당하는 약개체 집합 E2의 구별자(discriminator)임을 주시하자. 구별자에 대한 설명이 앞 장에서 필요*****

단계 10. ER 다이어그램으로부터 변환된 각 릴레이션에 대하여 정규화를 행한다.

단계 9에서 변환해야 할 구체적 대상과 방법에 대해서는 6,2,3.1절부터 6.2.3.5절까지 상세하게 소개한다. 또한, 단계 10에 대해서는 6.2.3.6절에서 설명한다.

6.2.3.1 강개체 집합의 변환

ER 다이어그램에서 개체 집합이 강개체 집합인지 또는 약개체 집합인지에 따라 개체 집합을

릴레이션으로 변환하는 방법이 다르다. 강개체 집합인 경우 변환하는 규칙은 다음과 같다.

규칙 1. (강재체 집합의 릴레이션과 주키 설정) ER 다이어그램에서 각 강개체 집합은 하나의 독립된 릴레이션으로 대응시키고 ER 다이어그램에서 강객체 집합에 연결된 속성은 대응하는 릴레이션의 속성이 되도록 한다. 릴레이션의 명칭은 객체 집합 명칭으로 동일하게 정한다. 개체 집합의 주키가 바로 이 릴레이션의 주키가 된다.

그림 6.2의 ER 다이어그램에서 강객제 집합은 단일 사각형으로 표시된 E1, E3, E4, E5, E6 이며 규칙 1에 따라 각 객체 집합에 연결된 속성들을 포함시켜 다음과 같이 각각 E1, E3, E4, E5, E6 명칭으로 변환된다. 여기에서 E2 개체 집합은 약개체 집합이므로 제외된다.

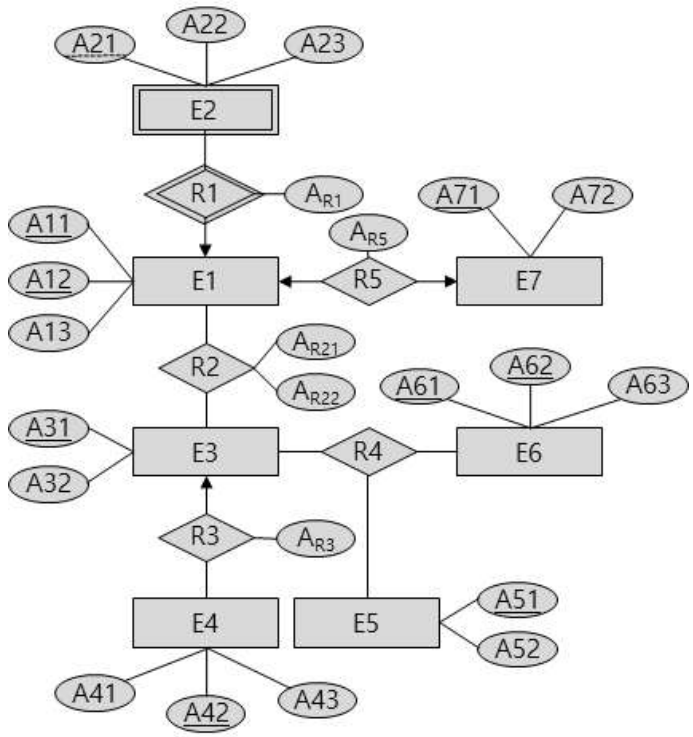

그림 6.2 관계형 데이터베이스 스키마로의 변환을 위한 ER 다이어그램 예

E1(A11, A12, A13)
E3(A31, A32)
E4(A41, A42, A43))
E5(A51, A52)
E6(A61, A62, A63)
E7(A71, A72)

이들 릴레이션의 속성에서 주키에 해당하는 속성은 밑줄로 표시되었다.

6.2.3.2 약개체 집합의 변환

약개체 집합의 개체는 단독으로 존재할 수 없기 때문에 그 개체를 존재하게 하는 강개체가 요구되는데 이를 고려하여 약개체 집합을 릴레이션으로 변환하는 규칙은 다음과 같다.

규칙 2. (약개체 집합의 릴레이션과 주키 설정) ER 다이어그램에서 각 약객체 집합은 그 개체 집합을 결정하는 강개체 집합에 의해 그 존재가 식별되기 때문에 단독으로 릴레이션으로 변환될 수 없다. 따라서 약개체 집합을 릴레이션으로 변환하기 위해서 그 개체 집합을 결정하는 강개체 집합의 주키를 외래키로 하여 약개체 집합의 속성들과 합쳐서 약개체 집합의 명칭으로 릴레이션으로 변환한다. 변환된 릴레이션의 주키는 강개체 집합의 주키와 약개체 집합의 **구별자**를 합쳐서 구성한다. 이 때, 강개체 집합의 주키 속성은 변환되는 릴레이션의 외래키로서 이 강개체 집합을 참조하는 제약 조건을 갖게 된다.

그림 6.2에서 약개체 집합 E2는 규칙 2에 의해 자신의 속성 A21, A22, A23와 강개체 집합인 E1의 주키인 A11과 A12를 포함하여 다음의 릴레이션 *E2(A11, A12, A21, A22, A23)*로 변환된다.
규칙 2에 따라 릴레이션 E1에서 강개체 집합 주키 A11, A12와 E2의 구별자 A21을 합쳐서 {A11, A12, A21}이 주키가 된다. 또한, 속성 A11과 A12는 릴레이션 E1을 참조하는 외래키이다. 변환된 최종 릴레이션은 다음과 같다.

E2(A11, A12, A21, A22, A23)

6.2.3.3 관계성 집합의 변환

ER 다이어그램의 관계성 집합 R에서 이 관계에 참여하는 개체 집합들의 주키를 모두 모았을 때 그 속성들을 $a_1, a_2, \ldots, a_m$이라 하고 R이 원래 갖고 있는 속성을 $b_1, b_2, \ldots, b_n$이라 하자. 이 때 릴레이션 변환은 다음 규칙을 따른다.

규칙 3. (관계성 집합의 릴레이션으로의 일반적인 변환) 관계성 집합 R과 동일한 명칭의 릴레이션을 생성하되 관여하는 모든 개체 집합들의 주키를 모은 $a_1, a_2, \ldots, a_m$ 속성들과 R이 원래 갖고 있는 속성들 $b_1, b_2, \ldots, b_n$이 이 릴레이션의 속성이 된다. 즉, 릴레이션 R의 속성은 다음과 같다.

$\{a_1, a_2, \ldots, a_m\} \cup \{b_1, b_2, \ldots, b_n\}$
여기에서 $a_1, a_2, \ldots, a_m$는 해당 릴레이션을 참조하는 외래키

이 때 관계성 집합에 참여하는 각 개체 집합의 주키는 생성하는 릴레이션의 외래키로서 그 개체 집합을 참조하는 제약 조건을 갖게 된다. 주키 설정은 관계성 집합의 조건에 따라 규칙 3.1부터 규칙 3.5 사이에서 해당하는 경우를 적용하여 선택한다.

규칙 3을 적용하여 변환한 릴레이션의 주키는 관계성 집합에 관여하는 개체 집합의 수와 대응 카디널리티에 따라 달라진다. 즉, 관여하는 개체 집합의 수가 둘인 관계성 집합을 **이진 관계(binary relation)**라 하고 관여하는 개체 집합의 수가 n(≥3)인 관계성 집합을 **n차 관계(n-ary relation)**라 하는데 관계성 집합이 이진 관계인지 아니면 n차 관계인지에 따라 주키 설정 방법이 다르다. 주키 설정 방법은 각 경우에 따라 규칙 3.1부터 규칙 3.5를 따른다.

규칙 3.1. (관계성 집합이 이진 관계이고 대응 카디널리티가 1 : 1인 경우의 주키 설정) 관계성 집합에 관여하는 개체 집합을 A, B라 할 때 규칙 3의 개체 집합들의 주키를 모은 a_1, a_2, ..., a_m 속성들 중에서 A 혹은 B 둘 중의 한 개체 집합의 주키를 규칙 3에서 도출한 릴레이션 R의 주키로 선택한다. A와 B 중에서 주키의 선택은 속성의 수가 동일하다면 어느 것을 선택하든 상관없지만 다르다면 주키의 최소성 원칙에 따라 속성의 수가 더 적은 것으로 한다.

그림 6.2에서 규칙 3.1의 조건에 해당하는 관계성 집합은 R5이다. 관계성 집합 R5를 릴레이션으로 변환하는 것이므로 우선, 규칙 3을 적용하면 관계성 집합 명칭 R5가 릴레이션 명칭이 된다. 그리고 관계성 집합 R5에 관여하는 개체 집합 E1의 주키 속성 A11, A12와 E7의 주키 속성 A71 그리고 R5가 갖고 있는 속성 A_{R5}가 릴레이션 R5를 구성하는 속성이 되어 결과적인 릴레이션 *R5(A11, A12, <u>A71</u>, A_{R5})* 가 도출된다.

R5의 주키는 규칙 3.1에 따라 E1의 주키 {A11, A12}와 E2의 주키 {A71} 중 속성 수가 적은 {A71}을 주키로 선택하면 최종 릴레이션은 다음과 같게 된다.

R5(A11, A12, <u>A71</u>, A_{R5})

또한, {A11, A12}는 개체 집합 E1을 변환한 릴레이션 *E1(<u>A11, A12</u>, A13)*을 참조하는 외래키이며 {A71}은 개체 집합 E2를 변환한 릴레이션 *E7(<u>A71</u>, A72)*을 참조하는 외래키가 된다.

규칙 3.1은 1 : 1 대응 관계를 갖는 관계성 집합을 새롭게 생성한 릴레이션으로 변환하는 방법을 나타낸다. 개체 집합 E1과 E7 사이의 1 : 1 대응 관계성 집합 R5의 예에서 도출되는 릴레이션을 나열하면 다음과 같다.

E1(<u>A11, A12</u>, A13)
E7(<u>A71</u>, A72)
R5(A11, A12, <u>A71</u>, A_{R5})

관계성 집합 R5에서 만일 개체 집합 E1의 모든 개체에 대해 개체 집합 E1의 한 개체가 대응된다면 *R5(A11, A12, <u>A71</u>, A_{R5})*에서 주키 A71의 값들이 *E7(<u>A71</u>, A72)*에서 A71의 값들과 동일하게 되므로 두 릴레이션 *E7(<u>A71</u>, A72)*와 *R5(A11, A12, <u>A71</u>, A_{R5})*를 하나로 합칠 수 있다. 합칠 때 두 릴레이션은 개체 집합을 변환한 릴레이션 명칭으로 하며 이렇게 하면 다음과 같이 하나의 릴레이션이 된다.

E1(A11, A12, A13)
E7(A71, A11, A12, A72, A_{R5}) (릴레이션 E7과 R5를 합친 스키마)
(릴레이션 *R5(A11, A12, A71, A_{R5})*는 삭제)

따라서 관계성 집합 R을 릴레이션으로 변환하는 규칙 3.1의 조건에서 한 개체 집합 E1의 각 개체가 개체 집합 E2의 한 개체와 관계성을 갖는 경우의 릴레이션 결합 규칙은 다음과 같다.

규칙 3.1.1. (규칙 3.1의 조건에서 개체 집합 E1의 각 개체가 반드시 개체 집합 E2의 어느 한 개체와 관계성을 갖는 경우의 릴레이션 결합) 규칙 3.1에서 도출한 릴레이션 R에서 E1의 주키 속성을 뺀 남은 속성들을 E1에 추가하고 릴레이션 R은 삭제한다.

한편, 관계성 집합이 이진 관계이고 대응 카디널리티가 1 : n 혹은 n : 1이면서 강개체 집합 사이의 관계인 경우의 주키 설정 규칙은 다음과 같다.

규칙 3.2. (관계성 집합이 이진 관계이고 대응 카디널리티가 1 : n 혹은 n : 1이면서 강개체 집합 사이의 관계인 경우의 주키 설정) 관계성 집합의 대응 카디널리티에서 n에 해당하는 개체 집합의 주키를 규칙 3에서 도출한 릴레이션 R의 주키가 된다.

그림 6.2에서 관계성 집합 R3는 개체 집합 E3과 E4 사이에서 1 : n 대응 카디널리티를 갖는다. 우선 규칙 3을 적용하면 릴레이션 *R3(A31, A42, A_{R5})* 가 도출된다.
게다가 관계성 집합 R3는 규칙 3.2 조건에 해당하므로 규칙 3.2를 적용하면 주키는 n에 해당하는 개체 집합인 E4의 주키인 A42가 된다. 따라서, 변환된 최종 릴레이션은 다음과 같다.

R3(A31, A42, A_{R5})

규칙 3.2의 조건에 따라 규칙 3과 규칙 3.2를 적용할 경우 중복성이 발생하는 경우가 있다. 그림 6.3의 ER 다이어그램은 약개체 집합에 관여하는 관계성 집합을 나타낸다. 기본적으로 이 종류의 관계성 집합은 강개체 집합과 약개체 집합 사이에서 1 : n 대응 관계를 가지며 어떠한 속성도 갖지 않는 특성이 있다. 또한, 약개체 집합의 각 개체는 강개체 집합의 한 개체와 관계성을 갖는다. 그 이유는 약개체는 강개체 없이는 존재할 수 없기 때문이다. 그림 6.3의 개체 집합과 관계성 집합에 규칙 1, 규칙 2, 규칙 3, 규칙 3.2를 적용하여 릴레이션으로 변환하면 다음과 같다.

E8(A81, A82) (규칙 1을 적용한 릴레이션)
E9(A81, A91, A92, A93) (규칙 2을 적용한 릴레이션)
R4(A81, A91) (규칙 3과 규칙 3.2을 적용한 릴레이션)

이 때, 릴레이션 E9과 R4는 주키로 동일한 {A81, A91}를 갖게 되며 개체 집합 E9의 모든 개체는 독립적으로 존재할 수 없으므로 E9의 각 개체에 대해 하나의 강개체 집합인 E8에 속한 강개체의 주키인 A81의 값이 대응된다. 또한, 릴레이션 R4 역시 두 개체 집합 사이에서 E9의 약개체 주키 A91 값에 대해 하나의 E8의 강개체 주키 A81 값이 대응되는데, 이렇게 되면 R4

의 주키 {A81, A91} 값들이 E9의 주키 {A81, A91} 값들과 동일하게 되어 데이터 중복성(data redundancy)이 발생한다. 따라서, *R4(A81, A91)* 의 정보는 불필요하다. 그러므로 그림 6.3의 ER 다이어그램을 릴레이션으로 변환하는데 중복성을 갖는 *R4(A81, A91)*는 삭제해도 되므로 다음 두 릴레이션으로의 변환이면 충분하다. 만일 관계성 집합에 속성이 있다면 6.2.3.2절의 규칙 2에 의해 약개체 집합을 변환한 릴레이션에 그 속성을 추가한다.

E8(A81, A82)
E9(A81, A91, A92, A93)

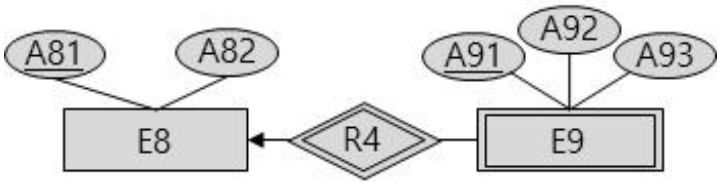

그림 6.3 약개체집합이 관여하고 속성을 갖지 않는 관계성 집합 예

앞의 예를 기반으로 약개체 집합이 관련된 관계성 집합을 릴레이션으로 변환하는 일반적 규칙을 소개하면 다음과 같다.

규칙 3.2.1 (관계성 집합이 약개체 집합 사이에 관여하는 관계인 경우의 릴레이션) 관계성 집합에 규칙 3을 적용하여 도출된 릴레이션을 삭제하고 관계성 집합의 속성들을 6.2.3.2절의 규칙 2에 의해 약개체 집합을 변환한 릴레이션에 추가한다.

그림 6.4는 1 : n 대응 관계에서 n에 해당하는 개체 집합의 각 개체에 대해 1에 해당하는 개체 집합의 한 개체가 대응하는 관계성 집합 예를 나타낸다. 이 그림에서 R4와 E11 사이의 이중선은 R4가 개체 집합 E11의 각 개체에 대해 개체 집합 E10의 한 개체와 관계성을 갖는 관계성 집합임을 나타내는 기호이다.
그림 6.4의 ER 다이어그램은 지금까지의 변환 규칙 조건들 중에서 규칙 1, 규칙 3, 규칙 3.2의 조건에 해당하며 이 규칙들을 적용하여 도출한 릴레이션들은 다음과 같다.

E10(A101, A102) (규칙 1을 적용한 릴레이션)
E11(A111, A112, A113) (규칙 1을 적용한 릴레이션)
R4(A101, A111) (규칙 3, 규칙 3.2를 적용한 릴레이션)

*E11(A111, A112, A113)*과 *R4(A101, A111)*에서 주키는 {A111}으로 동일한 속성임을 주시하자. 게다가 개체 집합 E11의 모든 개체는 E10의 한 개체와 관계성을 갖는다. 따라서, *R4(A101, A111)*에서 주키 A111에서 나타나는 모든 주키 값들이 *E11(A111, A112, A113)*의 A111에서 동일하게 나타난다. 그러므로 두 릴레이션 *E11(A111, A112, A113)*과 *R4(A101, A111)*를 하나의 릴레이션으로 합쳐도 무방하다. 즉, 이 두 릴레이션을 하나로 합쳐서 다음과

같이 n에 해당하는 릴레이션 명칭으로 도출할 수 있다.

E10(A101, A102)
E11(A111, A112, A113, A101)
(E11으로 합쳤으므로 릴레이션 *R4(A101, A111)* 는 삭제함)

원래의 E*11(A111, A112, A113)* 릴레이션에 *R4(A101, A111)*와의 공통 주키인 A111과 더불어 A101 속성을 추가하여 새롭게 *E11(A111, A112, A113, A101)* 스키마로 합치고 주키는 E11의 주키로서 변함없음을 주시하자.

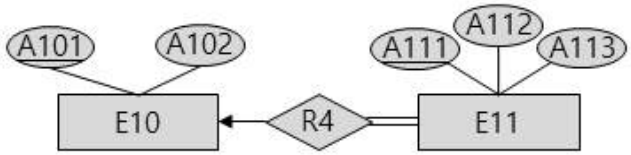

그림 6.4 1 : n 대응 관계에서 n에 해당하는 개체 집합의 각 개체에 대해 1에 해당하는 개체 집합의 한 개체가 대응하는 관계성 집합 예

규칙 3.2.2 (1 : n 대응 관계에서 n에 해당하는 개체 집합의 모든 개체에 대해 1에 해당하는 개체 집합의 한 개체가 대응하는 관계성 집합인 경우의 릴레이션 결합) 규칙 3.2에서 도출한 관계성 집합의 릴레이션에서 주키를 제외한 나머지 속성들을 n에 해당하는 개체 집합의 릴레이션에 추가하고 규칙 3.2에서 도출한 관계성 집합의 릴레이션은 제거한다. 한편, n에 해당하는 개체 집합의 주키가 합쳐진 릴레이션의 주키가 된다.

규칙 3.2.2는 결과적으로는 규칙 3.2.를 적용하는 대신 규칙 1에서 n에 해당하는 개체 집합을 변환한 스키마에 1에 해당하는 개체 집합의 릴레이션의 주키를 추가하고 관계성 집합의 속성들을 추가하여 구성한 릴레이션이 된다. n에 해당하는 개체 집합의 주키가 이 릴레이션의 주키가 된다.

규칙 3.3. (관계성 집합이 이진 관계이고 대응 카디널리티가 n : m인 경우의 주키) 관계성 집합에 관여하는 두 개체 집합의 주키 속성들을 모두 모은 속성들, 즉, $\{a_1, a_2, \ldots, a_m\}$ 이 규칙 3에서 도출한 릴레이션 R의 주키가 된다.

그림 6.2에서 관계성 집합 R2가 규칙 3.3의 조건에 해당한다. 이에 따라 우선 규칙 3를 적용하면 두 개체 집합 E1과 E3의 주키 {A11, A12}와 {A31} 그리고 R2의 속성 A_{R21}, A_{R22}로 구성된 릴레이션 *R2(A11, A12, A31, A_{R21}, A_{R22})*가 도출된다.
여기에서 규칙 3.3을 적용하면 R2의 주키는 개체 집합 E1과 E3의 주키 속성들을 합해서 {*A11, A12, A31*}가 되어 변환된 최종 릴레이션은 다음과 같다.

R2(A11, A12, A31, A_{R21}, A_{R22}*)*

규칙 3.4. (관계성 집합이 n 차 관계이고 대응 카디널리티가 모두 다 대 다인 경우의 주키) 관계성 집합에 관여하는 모든 개체 집합의 주키 속성들을 모두 모은 속성들, 즉, {a_1, a_2, …, a_m}이 규칙 3에서 도출한 릴레이션의 주키가 된다.

그림 6.2에서 관계성 집합 R4는 n 차 관계이고 모두 다 대 다인 대응 카디널리티를 가지므로 규칙 3.4의 조건에 해당한다. 우선, 규칙 3를 적용하면 관계성 집합 R4의 속성은 없으므로 개체 집합 E3, E5, E6 각각의 주키인 {A31}, {A51}, {A61, A62}의 속성들로 구성된 릴레이션 *R4(A31, A51, A61, A62)*가 도출된다.
이 릴레이션에서 주키는 규칙 3.4를 적용하면 개체 집합 E3, E5, E6의 주키 속성들로 이루어진 {A31, A51, A61, A62}이므로 변환된 최종 릴레이션은 다음과 같다.

R4(A31, A51, A61, A62)

규칙 3.5. (관계성 집합이 n차 관계이고 대응 카디널리티에서 1 에 해당하는 개체 집합이 오직 한 개인 경우의 주키) 대응 카디널리티가 1에 해당하지 않는 개체 집합들의 주키 속성들을 모두 모은 속성들이 규칙 3에서 도출한 릴레이션 R의 주키가 된다.

그림 6.5에서 관계성 집합 R4는 규칙 3.5의 조건에 해당한다. 따라서, 우선 규칙 3를 적용하면 R4에 관여하는 개체 집합 E3, E5, E6의 각 주키 {A31}, {A51}, {A61, A62}와 R4의 속성 A_{R4}를 합쳐서 릴레이션 *R4(A31, A51, A61, A62, A_{R4})*가 도출된다.
주키를 설정하기 위해 규칙 3.5를 적용하면 대응 관계에서 1에 해당하는 개체 집합 E5의 주키를 제외한 나머지 개체 집합 E3와 E6의 주키 속성을 합친 {A31, A61, A62}가 주키가 되며 이를 반영하여 변환한 최종 릴레이션은 다음과 같다.

R4(A31, A51, A61, A62, A_{R4})

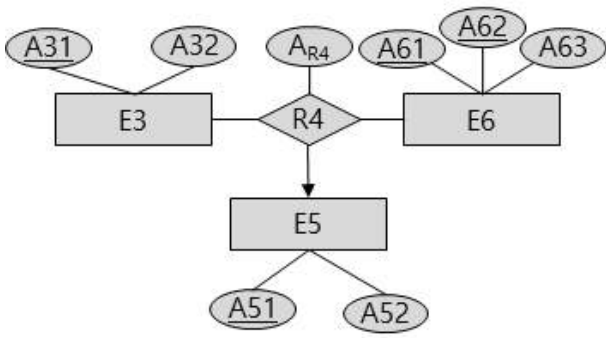

그림 6.5 관계성 집합이 n차 관계이고 대응 카디널리티에서 1 에 해당하는 개체 집합인 예

6.2.3.4 일반화(Generalization) 관계성의 변환

ISA 관계성과 상속성에 대해 2.5.4절에서 이미 소개되었다. 일반화(Generation) 관계성인 ISA 관계성은 상위 개체 집합의 속성을 하위 개체 집합이 상속받는 상속성의 특성을 갖는다. ISA 관계성과 상위 개체 집합 그리고 하위 개체 집합들을 릴레이션으로 변환하는데 상속성이 반영되는데 상위 개체 집합과 하위 개체 집합들 간의 조건에 따라 변환 방법이 달라진다.

설명을 위해 ISA 관계성에서 상위 개체 집합을 A, 하위 개체 집합들을 B_i(i=1, 2, ..., n)라 하자.

만약 상위 개체 집합 A에 속하는 개체가 반드시 오직 하나의 하위 개체 집합 B_i에만 속한다고 하자. 이 때, B_i들의 모든 개체를 합치면 개체 집합 A의 개체들과 같으며 B_i들 사이에 공유되는 개체는 존재하지 않는다. 이 때, 개체 집합 A가 ISA 관계성 이외의 다른 관계성 집합에 참여하지 않는다면 개체 집합 A는 B_i들로 대체될 수 있다. 그러나, 만일 개체 집합 A가 ISA 관계성 이외의 다른 관계성 집합에 참여한다면 개체 집합 A를 B_i들로 대체할 수 없으며 대신 개체 집합 A에 대응하는 릴레이션에 최소의 속성들만 남기고 나머지 개체 집합 A의 속성들은 상속하여 개체 집합 B_i의 릴레이션에 새롭게 추가한다. 이들 각 조건에 대한 릴레이션 변환 규칙을 소개하면 다음과 같다.

규칙 4.1. (개체 집합 A에 속하는 개체가 반드시 오직 하나의 하위 개체 집합 B_i에만 속하고 개체 집합 A가 ISA 관계성 이외의 관계성 집합에 참여하지 않는 경우의 릴레이션과 주키 설정)

(상위 개체 집합의 변환) 상위 개체 집합 A의 릴레이션 변환은 행하지 않는다.

(하위 개체 집합의 변환) 각 하위 개체 집합 B_i는 우선 규칙 1을 적용하여 릴레이션 B_i로 변환하고 변환된 스키마에 상위 개체 집합 A의 모든 속성을 새롭게 추가하여 변경한다.

(주키 설정) 개체 집합 B_i의 주키를 릴레이션 B_i의 주키로 한다. 만약 개체 집합 B_i에 주키가 없다면 상위 개체 집합 A의 주키를 릴레이션 B_i의 주키로 한다.

규칙 4.2. (개체 집합 A에 속하는 개체가 반드시 오직 하나의 하위 개체 집합 B_i에만 속하고 개체 집합 A가 ISA 관계성과 다른 관계성 집합에 참여하는 경우의 릴레이션과 주키 설정)

(상위 개체 집합의 변환) 상위 개체 집합 A의 릴레이션은 개체 집합 A의 주키 속성만으로 구성하도록 한다.

(하위 개체 집합의 변환) 각 하위 개체 집합 B_i는 우선 규칙 1을 적용하여 릴레이션 B_i로 변환하고 상위 개체 집합 A의 모든 속성을 새롭게 추가하여 변경한다.

(주키 설정) 릴레이션 A의 주키는 개체 집합 A의 주키로 한다. 또한, 개체 집합 B_i의 주키를 릴레이션 B_i의 주키로 한다. 만약 개체 집합 B_i에 주키가 없다면 상위 개체 집합 A의 주키를 릴레이션 B_i의 주키로 한다.

2.5.4절의 그림 2.3에서 만약 사람 개체 집합의 각 개체가 학생 개체이거나 아니면 교수 개체이라고 하면 사람 개체 집합은 학생 개체 집합과 교수 개체 집합으로 완전히 분리된다. 이 경우, 그림 2.3에서는 사람 개체 집합이 다른 개체 집합과 ISA 관계성 이외의 관계성 집합에 참여하지 않으므로 규칙 4.1를 적용하여 상위 개체 집합인 사람에 대한 릴레이션으로의 변환은 행해지지 않으며 하위 개체 집합인 학생과 교수에 대해서만 다음과 같이 변환된다.

*학생(학번, 학교, 학년, **이름**, **주민등록번호**, **주소**)*
*교수(교수번호, 학교, 직급, **이름**, **주민등록번호**, **주소**)*

이 두 릴레이션에서 굵은 글꼴로 표시한 이름, 주민등록번호, 주소 속성들은 상위 개체 집합 사람의 속성들로서 상속성에 의해 상속받는 것들을 각각 명시적으로 표시한 것이다.

그러나, 그림 2.3에서 상위 개체 집합 사람이 ISA 관계성 이외에 다른 관계성 집합에 참여한다면 규칙 4.2를 적용한다. 따라서 상위 개체 집합 사람은 주키만으로 구성된 릴레이션으로 변환되며 각 하위 개체 집합에 대한 릴레이션은 앞과 동일하게 사람 개체 집합으로부터 상속받는 속성들과 합쳐서 구성한다. 규칙 4.2를 적용한 결과의 릴레이션들은 다음과 같으며 아래에서 상속받은 속성들은 굵은 글꼴로 표현했다.

사람(주민등록번호)
*학생(학번, 학교, 학년, **이름**, **주민등록번호**, **주소**)*
*교수(교수번호, 학교, 직급, **이름**, **주민등록번호**, **주소**)*

규칙 4.1과 4.2의 조건에 해당하지 않을 경우, 즉, 어떠한 하위 개체 집합 B_i에도 속하지 않는 상위 개체 집합 A의 개체가 존재하거나 혹은 둘 이상의 하위 개체 집합에 속하는 개체 집합 A의 개체가 존재할 경우 다음의 규칙에 따라 릴레이션으로 변환한다.

규칙 4.3. (규칙 1과 규칙 2의 조건에 해당하지 않는 경우의 릴레이션과 주키)

(상위 개체 집합의 변환) 상위 개체 집합 A의 릴레이션 변환은 규칙 1을 적용한다. 즉, 릴레이션을 개체 집합 A와 동일한 명칭으로 하고, 개체 집합 A의 속성들을 릴레이션 A의 속성으로 한다.

(하위 개체 집합의 변환) 하위 개체 집합 B_i(i=1, 2, …, n)의 변환은 우선 규칙 1을 적용한 후 변환된 릴레이션 B_i에 릴레이션 A의 주키를 새로 추가한다.

(주키 설정) 릴레이션 A의 주키는 개체 집합 A의 주키로 한다. 각 릴레이션 B_i의 주키는 개체 집합 B_i의 주키로 한다. 만약 개체 집합 B_i에 주키가 없었다면 새로 추가된 릴레이션 A의 주키가 릴레이션 A의 주키가 된다. 이 때 새로 삽입되는 속성들은 릴레이션 A를 참조하는 외래키가 된다.

2.5.4절의 그림 2.3은 사람, 학생, 교수 개체 집합들 사이의 ISA 관계성을 보여준다. 이 개체 집합들에서 규칙 4.1의 조건과 규칙 4.2의 조건을 모두 만족하지 않는다면 규칙 4.3을 적용하여 다음 릴레이션들을 도출한다.

사람(이름, 주민등록번호, 주소)
*학생(학번, 학교, 학년, **주민등록번호**)*
*교수(교수번호, 학교, 직급, **주민등록번호**)*

그림 2,3의 ISA 관계성의 상위 개체 집합 사람은 규칙 4.3에서 제시했듯이 규칙 1을 적용하여 릴레이션 *사람(이름, 주민등록번호, 주소)* 로 변환된다. 하위 개체 집합 학생과 교수는 우선 규칙 1이

적용되어 각각 *학생(학번, 학교, 학년)* 와 *교수(교수번호, 학교, 직급)* 로 변환되지만 상위 개체 집합의 주키인 {주민등록번호} 속성을 새로 추가하여 각각 *학생(학번, 학교, 학년, **주민등록번호**)* 와 *교수(교수번호, 학교, 직급, **주민등록번호**)*로 최종으로 변환된다. 이 두 릴레이션의 주키는 규칙 4.3에 따라 각각 두 개체 집합 학생과 교수의 주키인 {학번}과 {교수번호}가 된다.

6.2.3.5 종합화(Aggregation) 관계성의 변환

2.5.5절에서 소개했듯이 ER 모델에서는 관계성 집합이 개체 집합 사이에만 존재하기 때문에 관계성 집합과 관계성 집합과의 관계를 표현하기 위해 종합화 관계성을 도입했다. 2.5.5.절 그림 2.5의 수행 관계성 집합과 관리 관계성 집합을 ER 모델에서 직접 링크로 연결할 수 없기 때문에 수행 관계성 집합과 이에 참여하는 개체 집합들을 하나로 묶어 추상화하여 하나의 개체 집합처럼 취급했다. 이렇게 종합화한 개체 집합을 수행 관계성 집합 명칭을 붙여서 수행 종합화 개체 집합이라 부르기로 하자. 그러면 그림 2.5에서 관리 관계성 집합은 수행 종합화 개체 집합과 관리자 개체 집합이 참여하는 이진관계가 된다.
이 때, 관리자 개체 집합은 강개체 집합이므로 규칙 1을 적용하면 다음의 릴레이션이 도출된다.

관리자(ID, 이름, 주소)

그림 2.5에서 관리 관계성 집합은 수행 종합화 개체 집합과 관리자 개체 집합 사이에서 n : m 대응 관계를 가지므로 규칙 3,3에 따라 별도의 관리 릴레이션으로 변환해야 하는데 이를 위해서는 수행 종합화 개체 집합의 주키가 결정되어야 한다. 종합화 개체 집합에서 대표 관계성 집합의 주키를 종합화 관계 개체에서의 가상적인 주키로 삼는다. 따라서, 그림 2.5에서 수행 관계성 집합의 주키를 수행 종합화 개체 집합의 가상적인 주키로 하면 되는데 수행 관계성 집합은 참여 하는 개체 집합들 간에 다 대 다 관계성을 가지므로 규칙 3.4에 따라 {직원:ID, 업무코드, 지점명}이 주키가 된다. 여기에서 직원:ID는 직원 개체 집합의 ID 속성을 나타낸 것으로 관리자 개체 집합의 ID 속성과 구분하기 위해 릴레이션 명칭을 표시한 것이다.
수행 종합화 개체 집합의 주키가 가상적으로 결정되었으므로 이제 규칙 3.3을 적용하여 관리 관계성 집합을 릴레이션으로 변환할 수 있다. 관리 관계성 집합에 참여하는 속성들은 다음과 같다.

{관리자:ID} : 관리자 개체 집합의 주키 속성들
{직원:ID, 업무코드, 지점명} : 수행 종합화 개체 집합의 주키 속성들
{시작날짜, 종료날짜} : 관리자 개체 집합이 가진 속성들

이 속성들을 합쳐서 관리 릴레이션 스카마를 구성하고 주키는 두 개체 집합의 주키 속성들을 모아서 결정하면 다음의 릴레이션이 된다.

관리(관리자:ID, 직원:ID, 업무코드, 지점명, 시작날짜, 종료날짜)

지금까지의 그림 2.5의 예를 기반으로 하여 종합화 관계성과 관련하여 릴레이션으로 변환하는 일반적 규칙을 소개하면 다음과 같다.

규칙 5. (종합화 관계성에서 종합화된 추상 개체 집합 A가 참여하는 관계성 집합 R의 릴레이션)

종합화 개체 집합 A의 주키를 그 개체 집합 내부의 대표 관계성 집합의 주키로 하고 관계성 집합 R의 릴레이션 변환은 6.2.3.3에서 소개한 관계성 집합 변환 규칙 조건에 해당하는 규칙을 따른다.

6.2.3.6 각 릴레이션의 정규화

ER 다이어그램을 가지고 6.2.3.1절부터 6.2.3.5절까지의 규칙을 적용하여 도출된 각 릴레이션들을 대상으로 5장에서 소개한 정규화를 행한다. 각 릴레이션이 무슨 정규형에 있도록 할 것인가를 정하고 각 릴레이션이 어느 정규형에 있는지를 체크하면서 5장에서 소개한 규칙에 따라 무손실 분해를 진행한다. 모든 릴레이션이 목표 정규형을 만족하면 그 릴레이션이 논리적 설계인 관계형 데이터베이스 설계의 최종 결과가 된다.
설계 후속 작업으로 이 최종 릴레이션 각각에 대해 데이터 정의어(DDL)을 이용하여 릴레이션을 정의함으로써 데이터베이스가 형성된다.