NHN Academy

Module 09
개념적 설계와 ER 모델

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book and NHN Academy was aware of a trademark claim, the designations have been printed in caps or initial caps.

While every precaution has been taken in the preparation of this book, the publisher assumes no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

Module 09: 개념적 설계와 ER 모델

모듈 목표

데이터베이스 설계는 조직에 필요한 데이터를 어떻게 구조화하고 사용할 것인지를 결정하는 관계형 데이터베이스에서 가장 중요한 작업 중의 하나입니다. 데이터베이스의 설계는 데이터의 조직과 조직된 정보의 활용, 데이터의 품질뿐만 아니라 데이터베이스의 성능에도 크게 영향을 미칩니다. 이 모듈에서는 개체-관계 모델(ER Model, Entity-Relationship Model)을 사용하여 데이터베이스를 개념적으로 설명하는 방법에 관해 공부합니다.

이 장을 마치면, 다음과 같은 것들을 할 수 있게 됩니다:

- 관계형 데이터베이스 설계의 주요 프로세스를 설명할 수 있습니다.
- 개체 관계 모델(ER 모델)을 이해하고 주요 설계 방법을 설명할 수 있습니다.
- 개체 관계 모델을 사용하여 관계형 데이터베이스를 설계할 수 있습니다.
- 개념적 데이터베이스 설계 단계에서 고려해야 할 사항들을 이해합니다.

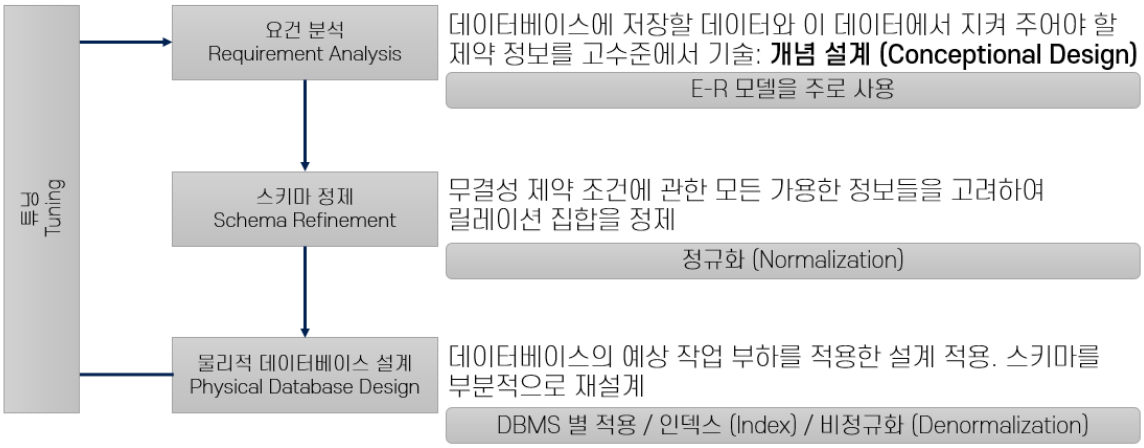
목차

데이터베이스 설계 개요

강나루 건너서 밀밭길을 구름에 달 가듯이 가는 나그네 길은 외줄기 남도 삼백리 술 익는 마을마다 타는 저녁놀 구름에 달 가듯이 가는 나그네.

Table of Contents

데이터베이스 설계



데이터베이스를 설계할 때 가장 먼저 해야 할 일은 그 조직체에서 데이터베이스에 어떠한 정보를 저장해야 하는가와 그 데이터에 어떠한 무결성 제약조건 또는 규칙을 적용해야 하는가를 이해하는 것입니다. 이런 첫 단계를 **요건 분석(Requirement Analysis)**이라고 하는데, 사용자 그룹과의 토론, 현행 운용 환경과 앞으로의 변화 전망에 대한 연구 검토, 이 데이터베이스를 활용하게 될 기존 응용들에 대한 분석등과 같이 비공식적인 과정을 거칩니다. 이 과정에서 모든 정보들을 사용해서 데이터베이스에 저장할 데이터와 이 데이터에서 지켜 주어야 할 제약조건들을 고수준에서 기술하게 되는데, 이 단계를 **개념적 설계(Conceptual Design)**라고 합니다. ER 모델이 데이터를 고도의 추상적인 형태로 기술할 수 있도록 해 주기 때문에 개념적 설계를 할 때에는 ER 모델을 자주 사용합니다. ER 모델을 직접 지원할 수 있는 데이터베이스 시스템은 없기 때문에, 관계형 데이터베이스 관리 시스템을 사용한다면 구축된 ER 모델 명세 내용을 여러 릴레이션의 집단으로 변환해 주어야 합니다.

설계 과정상의 두 번째 단계는 무결성 제약 조건에 관한 모든 가용한 정보들을 고려해서 이 릴레이션 집합을 정제하는 일입니다. 이 단계를 **스키마 정제(Schema Refinement)**라고 합니다. 첫 번째 단계인 개념적 설계 단계는 기본적으로 주관적인 단계임에 비하여 스키마 정제는 어떤 세련되고 강력한 이론을 통해 방향을 잡을 수 있습니다. 이 이론이 릴레이션의 **정규화(Normalization)**로서, 릴레이션들이 바람직한 성질을 갖도록 재조직하는 방식입니다.

데이터베이스 설계의 세 번째 단계는 물리적 데이터베이스 설계(Physical Database Design)입니다. 이 단계에서는 먼저 데이터베이스가 지원해 주어야 할 예상 작업 부하를 고려해 주어야 하고, 그 이우호를 차차 데이터베이스 설계를 정제하여 원하는 성능 기준에 맞도록 해 주어야 합니다. 이 단계를 단순화해서 생각한다면 릴레이션 몇 개에 인덱스를 구축한다거나 앞의 두 설계 단계에서 구축된 데이터베이스 스키마를 부분적으로 재설계하는 등의 작업입니다.

처음부터 순서대로 개념적 설계, 스키마 정제, 물리적 설계를 수행해 나간다 하더라도 결국은 그 후에 이 세 종류의 설계 단계를 반복적으로 교차 수행하는 조정(Tuning, 튜닝) 과정을 거쳐야 합니다.

개체 관계 데이터모델

ER 모델이라고 부르는 개체 관계 데이터모델은 개체(Entity), 애트리뷰트(Attribute), 관계(Relationship)을 사용하여 데이터베이스를 모델링하는 기법의 총칭입니다. 이 단원에서는 개체 관계 모델에 대해 알아봅니다.

Table of Contents

개체 관계 데이터모델

- 데이터를 개체(Entity), 속성(Attribute), 관계(Relationship)로 나타내는 데이터 모델
- 실 세계를 개체(Entity)라 불리는 기본 객체들과 그 객체들 사이의 관계로 인식
- 데이터베이스의 전체 논리적 구조를 나타내는 조직의 스키마(Enterprise Schema)를 명시함으로써 데이터베이스를 쉽게 설계하도록 개발됨
- 실세계의 조직의 의미와 상호작용을 개념적 스키마로 나타내는데 매우 유용함

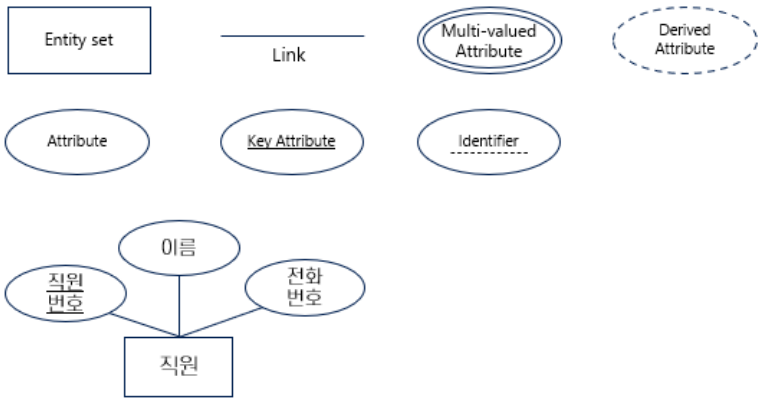
ER 데이터 모델의 기반이 되는 개념은 개체(Entity), 속성(Attribute), 관계(Relationship)입니다. ER 모델에서는 실 세계를 개체(Entity)라고 부르는 기본 객체들과 그 객체들 사이의 관계로 인식합니다. 개체 관계 모델에서는 데이터베이스의 전체적인 논리 구조를 나타내는 조직의 스키마를 명시함으로써 데이터베이스를 쉽게 설계할 수 있도록 합니다.

ER 모델은 실세계의 조직의 의미와 상호 작용을 개념적 스키마로 나타내는데 매우 유용합니다.

개체, 속성, 개체 집합

- 개체 (Entity)
  - 실제 세계에서 다른 객체와 구별되는 유, 무형의 사물
- 속성(Attribute)
  - 한 개체를 기술하기 위한 속성
  - 개체 집합에 속한 모든 개체들은 동일한 속성을 가짐
  - 가능한 값의 집합인 Domain을 지정하며, 개체를 식별하기 위한 Key 지정
  - 단순 속성Simple Attribute과 복합 속성Composite Attribute
  - 단일값 속성Single-value Attribute과 다중값 속성Multi-valued Attribute
  - 유도된 속성Derived Attribute
- 개체 집합(Entity set)
  - 개체들의 집합

Diagram Convention



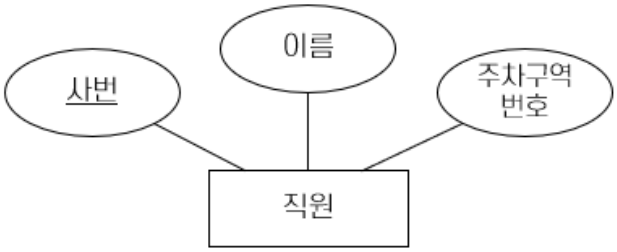
개체(Entity)란 실제 세계에서 서로 구분되는 객체를 말합니다. 회사, 부서, 부서장, 주소등이 실세계의 개체입니다. 비슷한 개체들을 하나로 묶은 것을 개체 집합(Entity Set)이라고 부릅니다. 개체는 실제 세계에서 다른 객체와 구별되는 유, 무형의 사물을 말합니다.

데이터베이스에서는 한 개체를 여러 속성(Attribute)의 집합을 사용해서 기술합니다. 어떤 개체 집합에 속한 모든 개체들은 동일한 속성을 갖습니다. 개체에 대한 정보를 얼마나 자세히 표현하느냐 하는 목표에 따라 속성들을 선정하게 됩니다. 예를 들어 직원이라는 개체 집합에 대해 사번, 이름, 주차 구역 번호 등을 애트리뷰트로 선정할 수 있습니다.

개체 집합에 대해 속성을 선정할 때에는 각 속성들에 대해 가능한 값의 집합인 도메인(Domain)을 지정

해야 합니다. 예를 들어 직원의 이름 속성의 도메인은 4자리의 문자열의 집합으로 지정할 수 있습니다. 또한 객 개체 집합에 대해 키(Key)를 선정합니다. 여기서 키의 개념은 앞에서 알아봤던 키의 개념과 동일합니다. 키는 해당 집합 내에서 한 개체를 유일하게 식별하는 값을 갖는 속성의 최소 집합입니다. 후보키는 하나 이상 있을 수 있는데, 그럴 경우에는 후보 키 중 하나를 기본 키로 지정합니다.

사번, 이름, 주차 구역 번호를 속성으로 갖는 직원 개체 집합을 Chen's notation으로 나타내면 아래와 같이 됩니다. 개체 집합은 사각형으로 표시하고 속성은 타원형으로 표시합니다. 기본 키 속성은 밑줄을 쳐서 표시합니다. 속성에 도메인 정보를 표시할 수 있으나 여기에서는 생략합니다.



관계 모델로 변환

개체집합 하나는 하나의 릴레이션으로 바로 매핑됩니다. 개체집합의 각 속성들은 릴레이션의 필드가 됩니다.

직원 개체가 두 개인 직원 개체집합의 한 인스턴스를 릴레이션 형태로 표현하면 아래와 같이 됩니다.

사번	이름	주차구역번호
1	이순신	47
2	홍길동	16

직원 테이블을 만드는 DDL 문은 아래와 같이 됩니다.

```
CREATE TABLE 직원 (
    사번    int,
    이름    varchar(10)m
    주차구역번호    int,

    CONSTRAINT pk_employee PRIMARY KEY(사번)
)
```

관계, 관계 집합

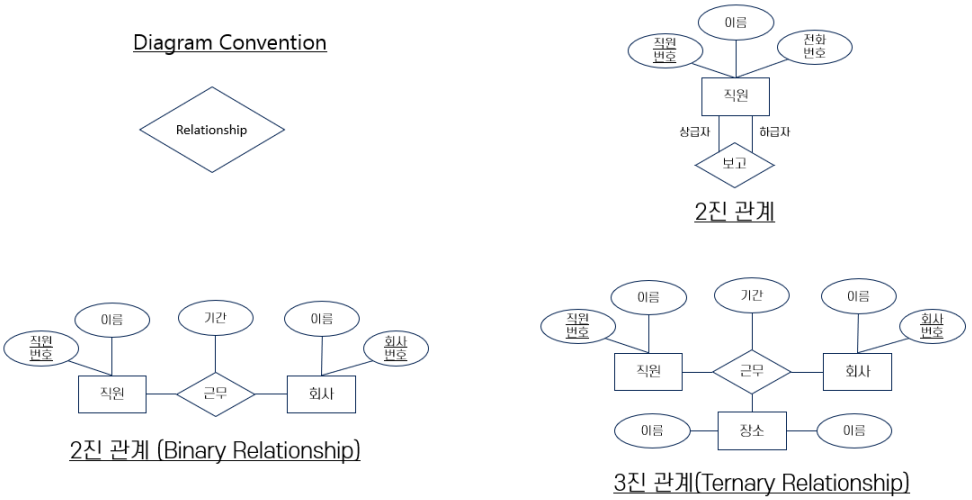
- 관계 (Relationship)
  - 여러 개체들 사이의 연관성
- 관계 집합(Relationship set)
  - 같은 유형의 관계들의 집합. n ≥ 2 개의 개체집합(중복 허용) 사이의 수학적 관계

만약 E_1, E_2… E_n이 개체 집합들일 때 관계집합 R은

$\{(e_1, e_2, \ldots e_n) | e_1 \in E_1 \ldots e_n \in E_n\}$

의 부분집합이다. 여기에서 $(e_1, e_2, \ldots e_n)$은 관계

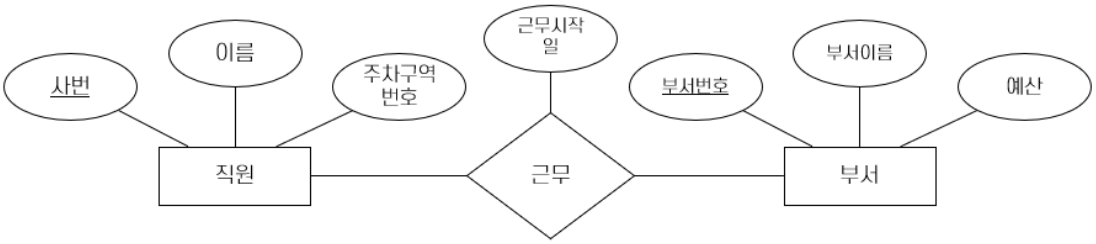
- 개체 집합 사이의 연관을 관계 집합에의 참가(participation)라고 함
- 관계에서 개체가 행하는 기능을 개체의 역할(Role)이라고 함
- 관계는 설명형 속성(descriptive attribute)이라는 속성을 가짐



관계(Relationship)는 둘 이상의 개체간의 관련성을 말합니다. 한 직원이 한 부서에서 근무한다는 개체간의 관계가 있을 수 있습니다. 개체와 마찬가지로 관계들도 비슷한 항목끼리 하나의 집합으로 묶을 수 있는데 이를 관계집합(Relationship Set)이라고 합니다. 관계 집합은 n-투플의 집합으로 생각할 수 있습니다.

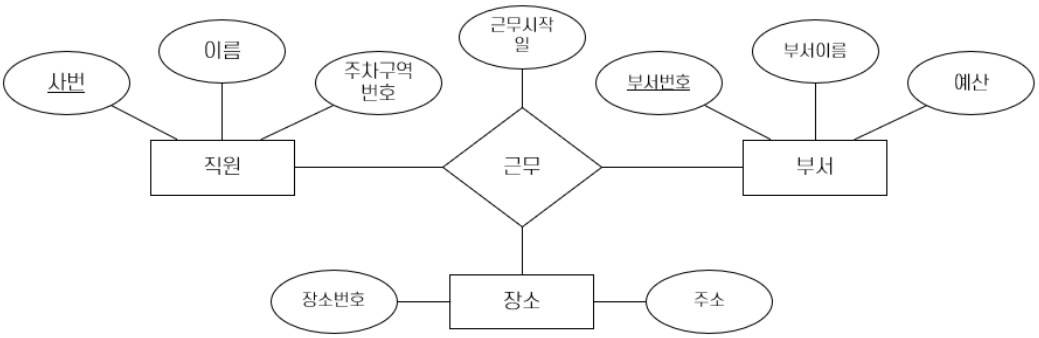
$\{(e_1, e_2, \ldots e_n) | e_1 \in E_1 \ldots e_n \in E_n\}$

개체 집합 E_i의 한 개체를 e_i라고 할 때 n-투플 하나는 e1에서 en까지 n개의 개체가 관련된 관계 하나를 나타냅니다. 아래 다이어그램은 한 명의 직원이 하나의 부서에 근무한다는 관계를 묶은 개체집합 근무를 나타냅니다.

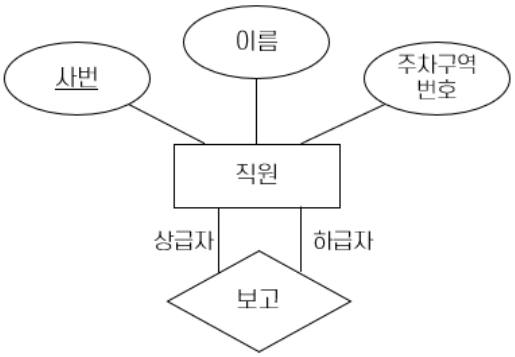


관계 자체에도 내용을 기술하는 **설명형 속성(Descriptive Attribute)**들이 들어갈 수 있습니다. 이 속성들은 관계 자체의 정보를 기록하는 것이며 이에 관련되는 개체 정보를 기록하지는 않습니다. 만약 홍길동 사원이 인사 부서에서 2022년 12월 3일부터 근무하고 있다는 사실을 기록하고자 할 때, 근무 시작일에 대한 정보는 직원 개체 또는 부서 개체 어디에도 기록될 수 없습니다. 이 정보는 직원과 부서 간의 관계에서 비롯되는 것이므로, 근무 개체집합에 추가된 근무 시작일 속성에 표현됩니다.

한 부서가 여러 지역에 지부를 가지고 있는 경우, 각 직원들이 어디에서 근무하고 있는지를 기록하고 싶을 경우, 이 관계는 직원과 부서, 장소와 관계됩니다. 이를 **3진 관계(Ternary Relationship)** 관계라고 하며, ER 다이어그램으로는 아래와 같이 표시할 수 있습니다.



관계집합에 참여하는 개체집합은 동일한 개체집합일 수 있습니다. 한 개체 집합의 두 개체(투플)을 한 관계가 연결시켜 주기도 합니다. 만약 한 직원이 다른 직원에게 보고하는 관계가 있다고 하면, 직원 개체집합과 직원 개체집합 사이에 관계가 필요하게 됩니다. 이 관계는 **역할(Role)**을 표현하는 관계가 되고, 이들은 **역할 지시자(Role Indicator)**를 사용하여 관계에서 상급자와 하급자를 표시하게 됩니다. 한 개체집합이 여러 역할을 수행할 때에는 해당 개체 집합의 속성 이름에 역할 지시자를 붙여서 관계집합 내의 각 속성에 유일한 이름을 주게 됩니다.



관계 모델로 변환

개체집합처럼 관계집합도 관계 모델의 한 릴레이션으로 매핑됩니다. 관계 하나를 표현하려면 이에 참여하는 각 개체를 식별하고 관계의 설명형 속성에 값을 줄 수 있어야 합니다. 이 릴레이션에는 다음과 같은 속성이 존재하게 됩니다.

- 참여하는 각 개체 집합의 기본 키 속성(외래 키 필드의 자격)
- 관계집합 자체의 설명형 속성

설명형 속성 이외의 속성 집합은 이 릴레이션의 수퍼키를 형성합니다. 제약조건이 없다면 이 속성 집합이 후보 키가 됩니다.

근무 개체집합의 인스턴스는 아래와 같이 표현될 수 있습니다. 사번과 부서번호가 근무 관계집합의 기본키가 됩니다.

사번	부서번호	근무시작일
1	12	2022-01-23
2	21	2022-10-22

근무 릴레이션을 생성하는 DDL 구문은 다음과 같습니다.

```
CREATE TABLE 근무 (
    사번 int,
    부서번호 int,
    근무시작일 date,

    CONSTRAINT pk_근무 PRIMARY KEY(사번, 부서번호),
    CONSTRAINT fk_근무_직원 FOREIGN KEY(사번) REFERENCES 직원(사번),
    CONSTRAINT fk_근무_부서 FOREIGN KEY(부서번호) REFERENCS 부서(부서번호)
)
```

역할지시자를 사용하는 보고 릴레이션을 생성하는 DDL 구문은 아래와 같습니다.

```
CREATE TABLE 보고 (
    상급자_사번     int,
    하급자_사번     int,
    CONSTRAINT pk_보고 PRIMARY KEY(상급자_사번, 하급자_사번),
    CONSTRAINT fk_보고_상급자 FOREIGN KEY(상급자_사번) REFERENCES 직원(사번),
    CONSTRAINT fk_보고_하급자 FOREIGN KEY(하급자_사번) REFERENCES 직원(사번)
)
```

ER 모델 기능

강나루 건너서 밀밭길을 구름에 달 가듯이 가는 나그네 길은 외줄기 남도 삼백리 술 익는 마을마다 타는 저녁놀 구름에 달 가듯이 가는 나그네.

Table of Contents

대응 수

- 참여 제약 조건(Participation Constraint)을 이루는 관계 비율
- 대응 수(Mapping Cardinality) 또는 수비율(Cardinality Ratio)
  - One-to-One A의 한 개체는 B의 한 개체와 연관을 가지고 B의 한 개체는 A의 한 개체 연관을 가진다.
  - One-to-Many A의 한 개체는 임의의 수 (0 또는 그 이상)의 B 개체와 연관을 가진다. 그러나 B의 개체는 A의 한 개체만 연관을 가진다.
  - Many-to-One A의 한 개체는 B의 한 개체와 연관을 갖는다. 그러나 B의 개체는 A의 임의의 수 (0또는 그 이상)의 개체와 연관을 갖는다.
  - Many-to-Many A의 한 개체는 임의의 수(0 또는 그 이상)의 B 개체와 연관을 갖고 B의 한 개체도 임의의 수 (0 또는 그 이상)의 A 개체와 연관을 갖는다.

개체-관계 조직 스키마는 어떤 제약조건을 데이터베이스가 어느 정도까지 준수해야 할 지를 정의할 수 있습니다. 여기에서 대응 수(Mapping Cardinality)와 참여 제약 조건(Participation Constraint) 개념이 등장합니다.

대응 수(Mapping Cardinality)

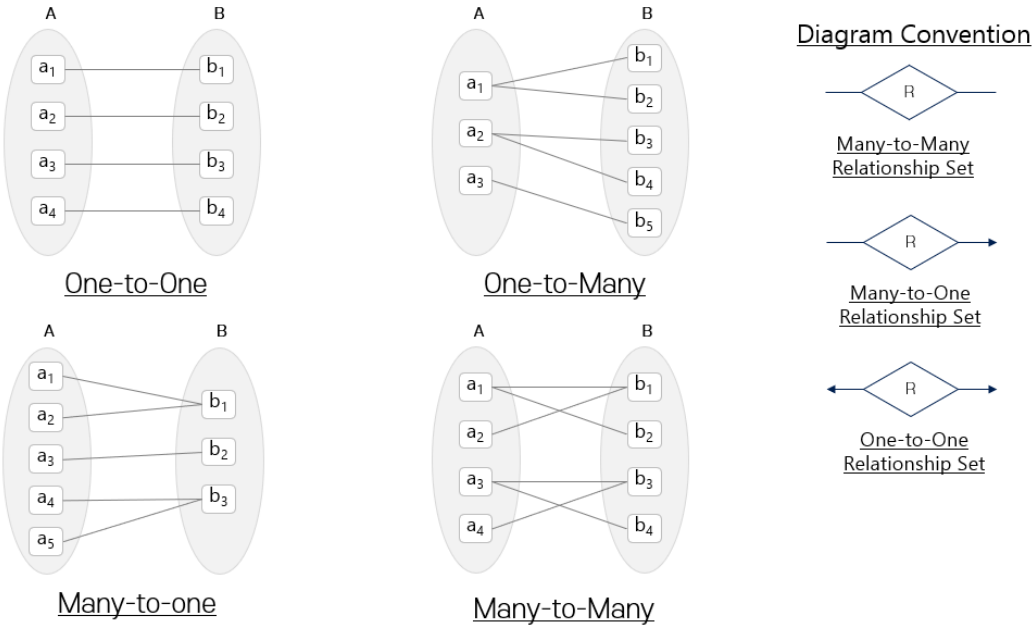
대응 수는 수비율(Cardinality Ratio)라고도 불리며, 관계 집합을 통하여 다른 개체와 관련될 수 있는 개체의 수를 나타냅니다. 대응 수는 개체 집합이 관련된 관계 집합을 기술하는데 사용될 수도 있지만, 이진 관계 집합을 나타내는데 가장 유용합니다.

개체 집합 A와 B사이의 이진 관계 집합 R에서 대응 수는 다음 중 하나여야 합니다.

- 일대일(One to One)_
  A의 한 개체는 B의 한 개체와 연관을 가지고 B의 한 개체는 A의 한 개체와 연관을 가진다.
- 일대다(One to Many)
  A의 한 개체는 임의의 수(0 또는 그 이상)의 B 개체와 연관을 가진다. 그러나 B의 개체는 A의 한 개체와만 연관을 가진다.
- 다대일(Many to One)
  A의 한 개체는 B의 한 개체와 연관을 갖는다. 그러나 B개체는 A의 임의의 수(0 또는 그 이상)의 개체와 연관을 갖는다.
- 다대다(Many to Many)
  A의 한 개체는 임의의 수((0 또는 그 이상)의 B의 개체와 연관을 갖고 B의 한 개체도 임의의 수(0또는 그 이상)의 A의 개체와 연관을 갖는다.

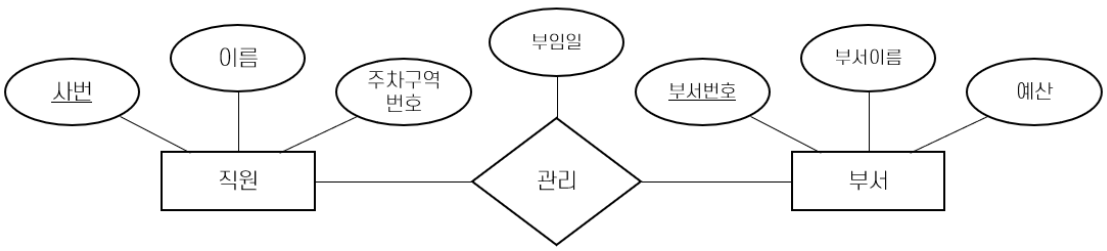
키 제약조건

- 릴레이션에서, 키 제약 조건에 따라 대응수가 정해짐
- 개체집합 E가 관계집합 R에 대해 키 제약조건을 가지고 있다면, E 인스턴스에 속한 객 개체는 R 인스턴스에 속한 관계 중 하나에만 나타남



특정 관계 집합의 적절한 대응 수는 관계집합이 모델링하는 실 세계의 상황을 따라야 합니다.

직원 개체집합과 부서 개체집합 간의 관계집합인 관리 관계집합에서, 각 부서는 한명의 관라자를 가지지만 한 직원은 여러 부서를 관리할 수 있다는 요구사항을 구현할 경우, 각 부서에는 한 명의 부서장이 있다는 것은 키 제약조건(Key Constraint)이 됩니다. 이는 관리 관계집합 인스턴스에서 각 부서 인스턴스의 개체는 하나의 관리 관계에 나타날 뿐이라는 것을 의미합니다.

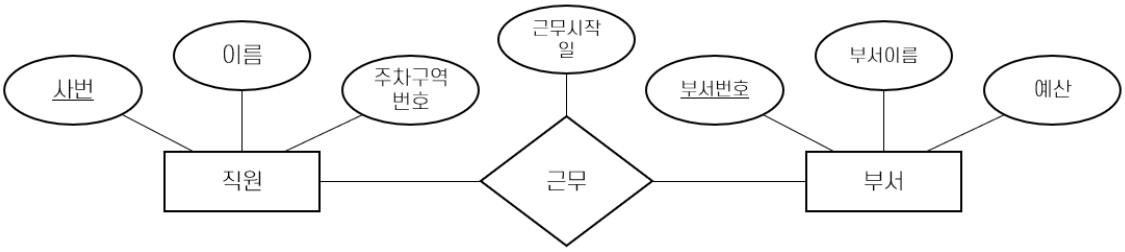


관리 정보에 대한 릴레이션은 근무 정보에 대한 릴레이션과 관이 사번, 부서번호, 부임일 세 개의 속성을 가집니다. 하지만 각 부서에는 부서장이 한 명밖에 존재하지 못하기 때문에 사번이 같은 투플은 여러 개 있을 수 있어도 부서번호가 같은 투플은 두개 이상 존재할 수 없습니다. 관리 관계집합은 부서번호로 관

리 릴레이션에 대한 키가 될 수 있습니다.

이런 관계집합을 일대다(One-to-Many)라고 하는데, 한 직원이 부서장으로서 여러 부서와 관련이 있을 수 있지만 각 부서는 한 명의 직원만 부서장으로 관련될 수 있다는 것을 나타냅니다.

이와는 반대로 근무 관계 집합은 한 명의 직원이 여러 부서에 근무할 수 있고, 또 한 부서에는 여러 명의 직원이 근무할 수 있으므로 다대다(Many-to-Many) 관계가 됩니다.



관계 모델로 변환

어떤 관계집합에 n개의 개체 집합이 참여하고 그 중 m개는 ER 다이어그램 상에서 화살표로 연결되어 있다면, 이들 m개의 개체집합에 대한 키들 중 어떤 것도 이 관계집합을 변환해서 만든 릴레이션의 키가 될 수 있습니다. 따라서 m개의 후보키가 존재하게 되는데 이들 중에서 하나를 기본 키로 지정해야 합니다.

관리 릴레이션은 아래와 같은 DDL 문으로 정의할 수 있습니다.

```
CREATE TABLE 관리 (
    사번     int,
    부서번호 int,
    부임일   date,

    CONSTRAINT pk_관리 PRIMARY KEY(부서번호),
    CONSTRAINT fk_관리_직원 FOREIGN KEY(사번) REFERENCES 직원(사번),
    CONSTRAINT fk_관리_부서 FOREIGN KEY(부서번호) REFERENCES 부서(부서번호)
)
```

다른 방법으로는, 관계집합에 대한 릴레이션을 별도로 만들지 않고 그 정보들을 키 제약조건으로 참여하고 있는 개체집합에 할당하는 방법이 있습니다. 한 부서에는 최대 한 명의 부서장만 있으므로 부서장에 해당하는 직원 투플의 키 필드와 부임일 속성을 부서 레코드에 추가하는 방법입니다.

```
CREATE TABLE 부서 (
    부서번호 int,
    부서이름 nvarchar(10),
    예산     decimal,
    부서장   int,
    부임일   date
```

```
    CONSTRAINT pk_부서 PRIMARY KEY(부서번호),
    CONSTRAINT fk_부서_부서장 FOREIGN KEY(부서장) REFERENCES 직원(사번)
)
```

이 방식을 사용하면 별도로 관리 릴레이션을 만들 필요도 없고 어떤 부서의 부사장을 알기위해 조인할 필요도 없습니다. 단좀은 부서장이 없는 부서가 있을 경우 공간이 낭비될 수 있으며, 함수 종속 관계로 인한 삽입, 삭제, 갱신 이상의 발생할 수 있습니다.

이 개념을 세 개 이상의 개체집합이 참여하는 관계집합으로 확장 적용할 수 있습니다. 일반적으로 말해서, 어떤 관계집합에 n개의 개체집합이 참여하고 그 중 m개는 ER 다이어그램 상에서 화살표로 연결된다면 이 m개의 개체집합을 표현한 어떠한 릴레이션에도 이 관계의 정보를 나타낼 수 있습니다.

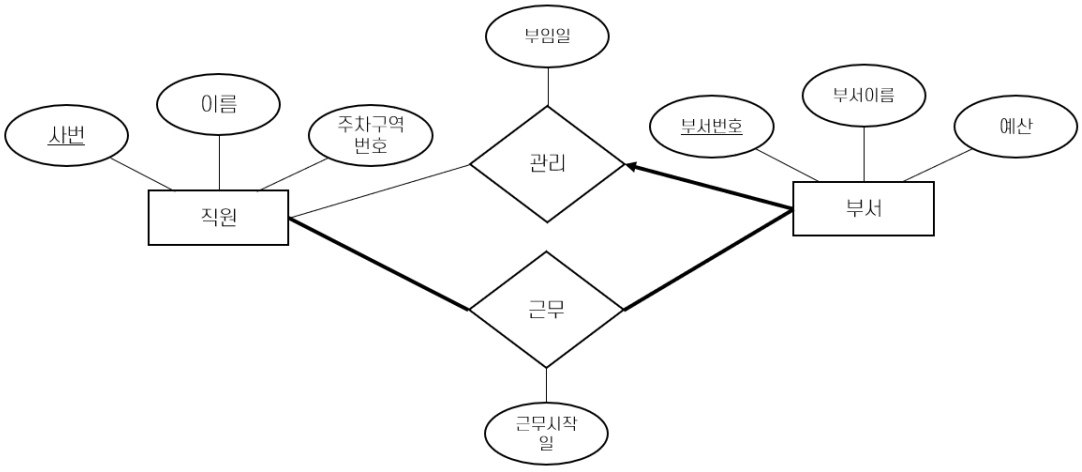
참여 제약 조건

- 한 개체집합 E의 관계집합 R로의 참가는 모든 E의 개체가 적어도 R 내부의 한 관계에 참가한다면 이는 전체적 참가이다.
- 만약 개제 집합 E의 일부 개체들만 R의 관계에 참여한다면 개체 집합 E의 관계 R로의 참가는 부분적 참가이다.

관리 관계에 대한 키 제약조건은 한 부서에 최대 한 명의 부서장이 있을 수 있다는 사실을 말해줍니다. 그렇다면 모든 부서에 부서장이 존재하는가 하는 질문이 있을 수 있습니다. 모든 부서에 부서장이 필요하다고 가정해 봅시다. 이런 요건이 참여 제약 조건(Participation Constraint)의 한 예가 됩니다.

관리 관계집합에서 부서 개체집합의 참여도를 전체적(Total)이라고 합니다. 관리 릴레이션의 입장에서 보면 이 제약조건은 부서 릴레이션에 나타나는 모든 부서번호 값이 관리 릴레이션에 적어도 한 투플 이상 나타내야 하고 사번 필드 값은 널이어서는 안된다는 뜻입니다. 한 부서에는 한 명의 부서장만이 존재해야 한다는 키 제약조건과 같이 생각하면, 각 부서는 부서장이 있어야 하며 반드시 한 명이어야 한다는 것입니다.

전체적이 아닌 참여도를 부분적(Partial)이라고 합니다. 모든 직원이 부서를 관리하지는 않으므로 관리 관계집합에 대한 직원 개체집합의 참여도는 부분적이 됩니다.



근무 관계집합은 모든 직원들은 적어도 한 부서에서 근무하고 또 각 부서에는 적어도 한 명의 직원이 근무한다고 짐작할 수 있습니다. 이 말은 직원 개체집합과 부서 개체집합이 모두 근무 관계집합에 전체적으로 참여하고 있음을 뜻합니다.

관계 모델로 변환

모든 부서에는 부사장이 한 명씩 있어야 한다는 제약조건을 DDL 문으로 표시하면 아래와 같이 됩니다.

```
CREATE TABLE 부서 (
```

```
    부서번호        int,
    부서이름        varchar(10),
    예산            decimal,
    부서장사번      int,
    근무시작일      date,

    CONSTRAINT pk_부서 PRIMARY KEY(부서번호),
    CONSTRAINT fk_부서_부서장 FOREIGN KEY(사번) REFERENCES 직원(사번) ON DELETE NO ACTION
)
```

부서장 사번에는 널 값이 들어갈 수 없으므로 부서의 각 투플은 부서장에 해당하는 직원의 한 투플을 가리키게 되며, 따라서 참여 제약조건이 만족됩니다. 참조 키의 집행이 ON DELETE NO ACTION이므로 부서장사번 레코드가 가리키는 직원 투플은 삭제할 수 없습니다. 직원 레코드를 삭제하려면 이를 가리키는 부서 레코드가 다른 직원을 가리키도록 바꾸어 놓아야 합니다.

만약 아래와 같은 방법으로 ER 모델을 관계 모델로 변환한다면, 모든 부서에 부서장이 한 명씩 반드시 존재해야 한다는 제약조건을 만족하지 못합니다.

```
CREATE TABLE 관리 (
    사번    int,
    부서번호 int,
    부임일   date,

    CONSTRAINT pk_관리 PRIMARY KEY(부서번호),
    CONSTRAINT fk_관리_직원 FOREIGN KEY(사번) REFERENCES 직원(사번),
    CONSTRAINT fk_관리_부서 FOREIGN KEY(부서번호) REFERENCES 부서(부서번호)
)
```

관리 관계집합과 같은 일대다 관계에 있어서, 특히 해당 개체 집합이 키 제약조건과 전체 참여 제약조건을 동시에 가지고 있을 경우 첫 번째 방식이 선호됩니다.

테이블 제약 조건이나 SQL-92 표준으로는 표현할 수 있는 참여 제약조건들이 있습니다. SQL 언어의 모든 표현력을 동연한다면 테이블 제약조건들을 명세할 수 있으나, 이들을 검사하고 집행하는 데는 비용이 많이 소요될 수 있습니다.

만약 근무 릴레이션에 대한 참여 제약조건은 일반적인 제약조건 명세법을 이용하지 않는다면 집행할 수 없습니다. 근무 릴레이션을 보면, 사번 필드와 부서번호 필드가 있는데 이들은 각기 직원 릴레이션과 부서 릴레이션을 참조하는 외래 키 들입니다. 근무에 대해 부서가 전체적으로 참여하도록 하려면 부서 릴레이션의 모든 부서번호 값들이 근무 릴레이션에서도 어느 한 투플에 나타나도록 해 주어야 합니다. 이를 위해 부서 릴레이션의 부서번호를 근무 릴레이션을 참조하는 외래 키로 선언할 수 있겠지만, 부서번호를 근무 릴레이션에서 후보키가 될 수 없으므로 외래 키 제약조건을 만들 수 없습니다.

SQL-92 표준으로 근무에 대한 부서의 전체 참여를 보장하려면 CHECK를 사용해야 할 수도 있습니다. 부서 릴레이션에 나오는 모든 부서번호 값들이 근무 릴레이션의 어느 한 투플에는 나와야 하고, 뿐만 아니

라 이에 해당하는 근무 투플에서 이 관계에 참여하는 다른 개체집합을 참조하는 외래 키 필드에 널 값이 들어가서는 안되도록 보장해 주어야 합니다.

키 제약조건과 외래 키 제약조건으로 참여 제약조건을 표한할 수 있는 특수한 경우는 또 한가지가 있는데, 참여하는 모든 개체집합에 해하여 키 제약조건과 전체 참여 제약조건이 붙은 관계집합입니다. 이 경우 가장 좋은 변환 방식은 모든 개체들과 이 관계를 하나의 테이블로 매핑하는 것입니다.

약 개체 집합

- 키가 존재하지 않는 개체 집합
  - 자신의 일부 속성과 다른 개체의 Primary Key를 조합하여야 유일하게 식별 됨
  - 다른 개체를 식별 소유자(Identity Owner)라고 함
- 다음 조건들을 만족할 때 성립
  - 식별 소유자와 약 개제 집합 사이에는 One-to-Many 관계 집합이 성립
  - 약 개체 집합은 식별 관계 집합에 전체적으로 참여하여야 함
  - 소유자 개체에 대해 약 개제 하나를 유일하게 식별해 주는 속성 집합을 약 개제 집합에 대한 구별자(discriminator) 또는 부분 키(Partial Key)라고 함
  -

개체 집합에 키가 항상 존재해야 하는 것은 아닙니다. 예를 들어 직원들이 피 부양자를 위해 보험증권을 구매하는 경우를 생각해봅시다. 데이터베이스는 보험증권에 관한 정보를 기록합니다. 이런 경우 각 보험증권이 어떤 피부양자를 대상으로 하는지에 대한 정보가 중요합니다. 직원이 회사를 사직할 경우 사직한 직원의 보험증권은 시한이 종료되고 그와 관련된 모든 증권 정보와 피 부양자 정보를 데이터베이스에서 삭제해야 하는 경우를 생각해봅시다.

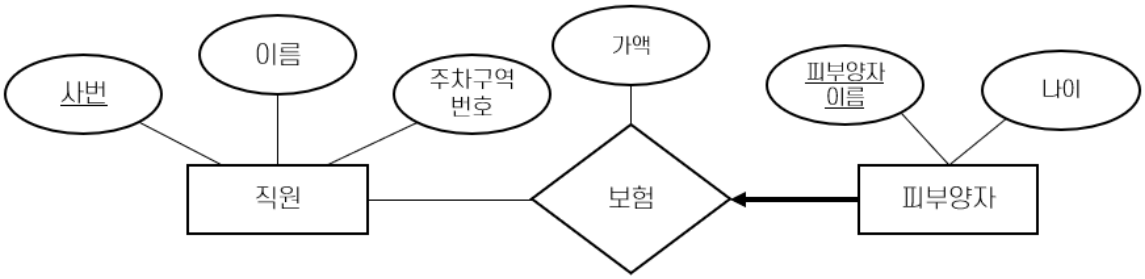
이런 상황에서는 피부양자를 이름만으로 식별하는 것이 일반적입니다. 한 직원의 피 부양자는 이름이 모두 다른 경우가 일반적이기 때문입니다. 이런 경우 피부양자 개체 집합의 속성은 피부양자 이름과 나이 정도가 될 수 있습니다. 피부양자이름 속성은 피 부양자를 유일하게 식별하지 않습니다. 직원에 대한 키는 사번입니다. 회사에는 이순신이라는 이름을 가진 직원이 두 명 이상 있을 수 있고, 이들은 이면이라는 이름을 가진 자녀를 한 명씩 둘 수 있습니다.

이때의 피부양자는 **약 개체 집합(Weakly Entity Set)**입니다. 약 개체는 자기 자신의 일부 속성과 다른 개체의 기본 키를 조합하여야만 유일하게 식별되는데, 이때 이 다른 개체를 **식별 소유자(Identifying Owner)**라고 합니다.

약 개체 집합은 다음과 같은 요건들이 만족 되어야 합니다.

- 소유자 개체집합과 약 개체 집합 사이에는 일대다 관계 집합이 있어야 합니다. 하나의 소유자 개체는 여러 약 개체와 연관되지만, 각각의 약 개체는 하나의 소유자만 갖습니다. 이러한 관계 집합을 약 개체 집합에 대한 식별 관계집합(Identifying Relationship Set)이라고 합니다.
- 약 개체 집합은 식별 관계 집합에 전체적으로 참여해야 합니다.

어떤 피부양자 개체를 유일하게 식별하려면 소유자 직원 개체의 키와 해당 피부양자 개체의 피부양자 이름을 조합해보아야 합니다. 주어진 소유자 개체에 대하여 약 개체 하나를 유일하게 식별해주는 집합을 그 약 개체 집합에 대한 부분 키(Partial Key)라고 합니다. 여기에서 피부양자이름은 피부양자에 대한 부분 키입니다.



피부양자 개제집합이 약 개체 집합이며, 보험 관계집합이 식별 관계집합입니다. 피부양자이름이 피부양자 약 개체 집합의 부분 키가 됩니다.

관계 모델로 변환

약 개체 집합은 언제나 일대다 이진 관계에 참여하며, 키 제약조건과 전체 참여 제약조건을 수반합니다. 약 개체 집합에서는 소유가 개체가 삭제되면 이와 관련된 모든 약 개체들도 삭제됩니다.

아래와 같은 보험 릴레이션으로 정의할 수 있습니다.

```
CREATE TABLE 보험 (
    피부양자이름     varchar(20),
    나이             int,
    가액             decimal,
    사번             int     NOT NULL,

    CONSTRAINT pk_보험 PRIMARY KEY(사번, 피부양자이름),
    CONSTRAINT fk_보험_직원 FOREIGN KEY(사번) REFERENCES 직원(사번) ON DELETE CASCADE
)
```

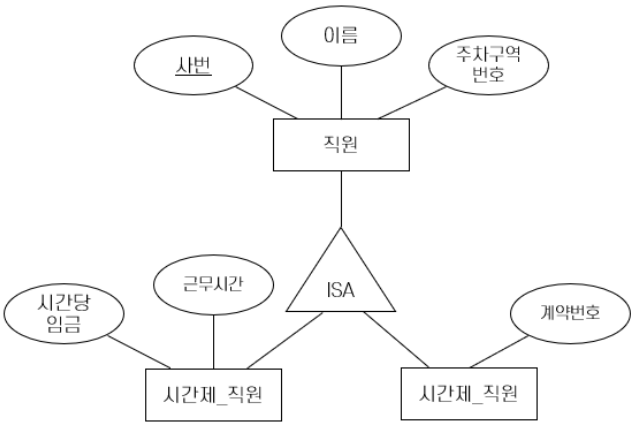
피 부양자는 약 개체이므로 기본 키는 <피부양자이름, 사번>이 됩니다. NOT NULL 제약조건에 의하여 모든 피부양자들은 하나의 직원 개체와 연관되게 되어 피부양자에 대한 전체 참여 제약조건을 만족하게 됩니다. CASCADE 조건을 붙여 직원 레코드가 삭제되면 피부양자 레코드 역시 삭제됩니다.

전문화와 일반화

- 하나의 개체 집합은 집합내의 다른 개체들과 구분되는 개체들의 하위집합을 가질 수 있음
  - 개체 집합 내의 어떤 부분집합은 개체 집합내의 모든 개체들과 공유되지 않는 속성들을 가질 수 있음
  - 개체 집합에 속한 세부 개체들을 세부 부류(subclass)로 분류
- 포함 제약 조건(Inclusion Constraint)
  - 중첩 제약 조건(Overlap Constraints)
    두 서브 클래스에 같은 개체가 포함될 수 있는가를 결정
  - 포괄 제약 조건(Covering Constraints)
    서브 클래스의 모든 개체를 모으면 수퍼 클래스의 모든 개체가 되어야 하는가를 결정

한 개체집합에 속한 개체들을 세부 부류(서브 클래스 - Subclass)로 분류하는 것이 자연스러울 때가 있습니다. 임금 계산 방식에 따라 시간제_직원 개체집합 또는 계약제_직원 개체집합을 생각할 수 있습니다. 이때 시간제_직원에 대해서는 근무시간과 시간당 임금 속성을, 계약제_직원에 대해서는 계약번호 속성을 할당할 수 있습니다.

이 두 집합에 속한 개체들은 모두 일종의 직원 개체이고, 따라서 직원 개체 집합에서 정한 사번, 이름 등의 모든 속성을 가져야 합니다. 이에 따라 시간제_직원 개체에 다한 속성들은 직원에 대한 속성에 시간제_직원의 속성을 더한 것이 되는데, 이때 시간제_직원 개체집합은 직원 개체집합의 속성들을 상속(Inheritance) 받는다고 합니다. 이때 Java등의 프로그래밍 언어들의 클래스 계층구조와는 다르게 포함 제약조건(Inclusion Constraints)이 따르게 됩니다. 즉, 모든 직원 개체들에 대한 질의는 모든 시간제_직원 개체와 계약제_직원 개체들에 대한 질의도 됩니다.



직원 개체집합을 다른 기준에 따라 나눌 수도 있습니다. 예를 들어 직원 중 일부를 임원으로 나눌 수 있는데, 직원의 자식 노드로 또 다른 ISA 노드를 달고 그 임원을 자식으로 설정하면 됩니다. 이런 방법으로 다중 ISA 계층 구조를 만들 수 있습니다.

클래스 계층을 보는 관점은 다음과 같이 두 가지가 있습니다.

- 직원이 여러 클래스로 전문화(Specialization)된 것. 전문화란 어떤 개체집합(슈퍼클래스, Superclass)중에서 다른 것 들로부터 구별되는 어떤 특성을 공유하는 일부 집합을 식별해 내는 과정입니다. 일반적으로 슈퍼클래스가 먼저 정해지고 나서 서브클래스들이 다음으로 정해지며 서브클래스 고유의 속성과 관계집합들이 추가됩니다.
- 시간제_직원과 계약제_직원이 모여서 직원으로 일반화(Generalization)된 것. 여러 개체 집합에서 공통적인 특성들을 추출해서 그러한 공통 특성을 가지는 개체들을 모든 큰 개체집합을 만들어 내는 것입니다. 일반적으로 서브클래스들이 먼저 정해지고 나서 슈퍼클래스가 정해지며 그 슈퍼클래스에 대한 관계 집합이 추가됩니다.

중첩 제약조건(Overlap Constraints)와 포괄 제약조건(Covering Constraints)

ISA 계층에 명세해 줄 수 있는 제약조건으로는 중첩 제약조건과 포괄 제약조건이 있습니다. 중첩 제약조건은 두 서브 클래스에 같은 개체가 포함될 수 있는지를 결정합니다. 홍길동 직원이 계약제 직원이면서 시간제 직원일 수 있는지에 대한 제약조건을 말합니다.

포괄 제약조건은 서브클래스의 모든 개체를 모으면 슈퍼클래스의 모든 개체가 되는지를 결정합니다. 모든 직원 개체들은 그 서브클래스들 중 하나에 속해야 하는지에 대한 제약 조건입니다.

서브클래스를 정하는 이유

전문화나 일반화를 통해서 서브클래스들을 정하는 기본적인 이유는 다음과 같습니다.

1. 서브클래스의 개체들에 대해서만 어떤 설명용 속성들을 추가해야 할 필요가 있을 때
2. 어떤 관계에 참여하는 개체들만 분류해야 할 경우가 있을 때

관계 모델로 변환

ISA 계층을 처리하는 방법에는 두 가지가 있습니다.

1. 개체집합인 직원, 시간제_직원, 계약제_직원을 각기 다른 릴레이션으로 매핑하는 방법. 시간제_직원 개체 집합을 릴레이션으로 변환할 경우, 릴레이션에는 시간당임금과 근무시간 속성이 지정됩니다. 그리고 슈퍼클래스의 키 속성인 사번 역시 지정되는데, 이 속성은 슈퍼클래스의 기본 키일 뿐만 아니라 슈퍼클래스인 직원 릴레이션을 참조하는 외래 키로 지정됩니다. 직원 투플이 삭제되면 시간제_직원 투플 역시 삭제되어야 합니다.

```
CREATE TABLE 시간제_직원 (
        사번            int,
        시간당임금      decimal,
        근무시간        int,

        CONSTRAINT pk_시간제직원 PRIMARY KEY(사번),
        CONSTRAINT fk_시간제직원 FOREIGN KEY(사번) REFERENCES 직원(사번)
)
```

2. 시간제_직원과 계약제_직원에 해당하는 두 개의 릴레이션만 만드는 방법. 시간제_직원 릴레이션에는 시간제 직원에 필요한 모든 속성(직원 개체에 필요한 속성까지)dl 모두 지정됩니다.

첫 번째 방식이 일반적으로 사용되는 방법으로서, 어느 경우에나 적용 가능합니다. 서브클래스에 한정된 속성들을 따지지 않고 모든 직원들을 검사하는 질의는 직원 릴레이션으로 수행하며, 시간제 직원과 관련된 질의는 직원과 조인해서 수행합니다.

만일 직원중에 시간제 직원도 아니고 계약제 직원도 아닌 직원이 있다면 두 번째 방식은 사용할 수 없습니다. 또 한 직원이 계약제 직원이면서 시간제 직원인 경우가 있다면 중복 저장되는 속성이 생길 수 있습니다.

개념적 설계 고려사항

데이터베이스 설계의 첫 단계는 모델링할 조직체와 저장할 정보를 이해하는 것입니다. 설계자가 데이터와 그 사용형태에 대해 숙지한 후 ER 모델등을 사용하여 개념적 설계를 수행하게 됩니다. 이 단원에서는 개념 설계 단계에서 ER 다이어그램을 구성할 때 염두에 두어야 할 사항들을 학습합니다.

Table of Contents

ER 모델을 이용한 개념적 설계

- ER 다이어그램 개발 과정은 선택의 연속
- ER 다이어그램은 데이터를 최대한 비슷하게 설명할 뿐, 모든 의미를 다 표현할 수 없음
- ER 모델링은 스키마 설계에 대한 완벽한 방법이 아님

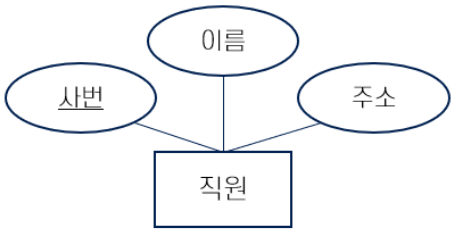
데이터베이스 설계의 제일 처음 단계는 모델링할 조직체와 저장할 정보를 이해하는 것입니다. 일단 설계자가 데이터와 그 사용 형태에 대해서 숙지하고 나면 ER 모델링등의 설계 방법을 이용하여 개념적 설계를 수행하게 됩니다. ER 다이어그램을 구성할 때 고려해야할 몇 가지 사항들은 아래와 같습니다.

1. ER 다이어그램 개발 과정은 선택의 연속이다.
2. ER 다이어그램은 데이터를 최대한 비슷하게 설명할 뿐이며 요구되는 모든 의미를 다 표현할 수는 없다.
3. ER 모델링은 데이터 설계에 대한 완벽한 방식이 아니며, ER 다이어그램을 변환하여 얻은 릴레이션은 스키마 정제가 필요하다.

개체 or 속성

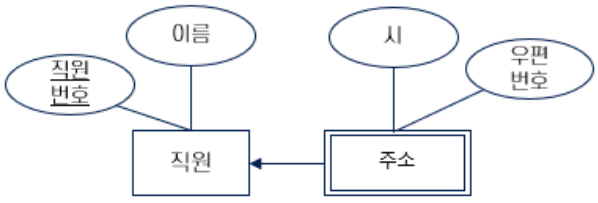
- 속성으로 모델링 하는 경우
  - 한 개체가 속성을 단일 값으로 가지는 경우
  - 요구 사항에서, 수직, 수평적으로 분리될 수 없는 값인 경우
- 개체로 모델링하는 경우
  - 한 개체가 여러 개의 값을 가지는 경우
  - ER 다이어그램에서 주소의 구조를 표현해야 하는 경우

어떤 개체집합이 가지는 것들을 식별해 내는데 있어서, 이를 속성(Attribute)로 표현하는 것이 좋은지 아니면 관계집합으로 표현하는 것이 좋은지 불분명할 때가 있습니다. 예를 들어 직원 개체 집합에 주소 정보를 추가한다고 할 때, 한 가지 안은 주소라는 속성을 이용하는 것입니다. 직원 당 하나의 주소만 기록하면 된다고 하면 적절한 선택이며, 주소를 문자열로 생각할 수 있습니다.

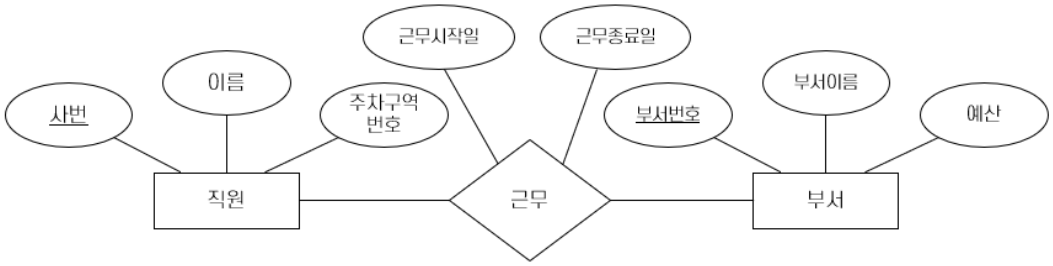


다른 방안은 주소라는 개체집합을 하나 만들고 직원과 그 직원의 주소간에 관계를 표현하는 것입니다. 이는 아래와 같은 경우에 해당합니다.

- 한 직원이 여러 주소를 가지고 있는 경우
- ER 다이어그램에서 주소의 구조를 표현해야 하는 경우, 주소의 내용을 시/도, 구/군, 우편번호 등으로 구분해야 하는 경우. 이런 경우라면 "분당구 판교동에사는 직원을 구하라"와 같은 질의를 작성할 수 있습니다.



속성이 아닌 개체로 모델링 해야 하는 다른 경우로, 아래와 같은 ER 다이어그램이 있습니다.

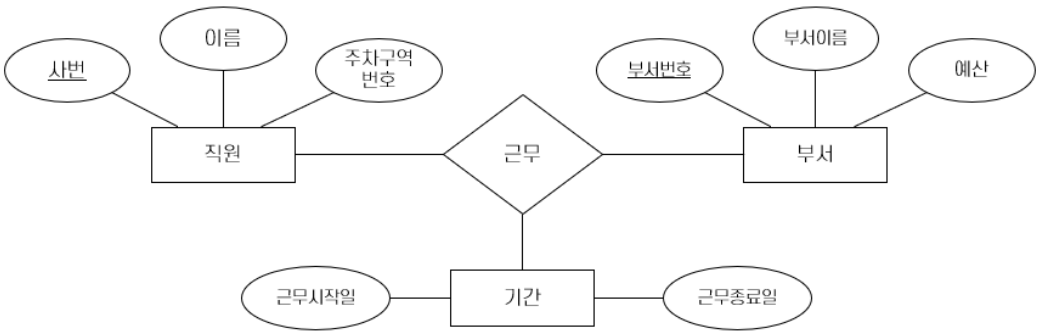


이 모델링은 한 직원이 부서에 근무한 이력을 저장할 수 있도록 합니다. 이 모델링에서 근무 관계집합의 기본 키는 직원 개체집합의 사번과 부서 개체집합의 부서번호 속성이 모여서 만든 수퍼키 <사번, 부서번호>가 됩니다. 이 모델링에서 한 직원이 여러 부서에 근무한 이력이 있는 경우를 생각해봅시다.

사번이 1인 직원이 다른 기간에 같은 부서에서 근무한 이력이 있다면, 근무 릴레이션의 인스턴스는 아래와 같이 될 수 있습니다.

사번	부서번호	근무시작일	근무종료일
1	1	2018-01-23	2019-10-11
1	2	2019-10-12	2020-11-23
1	1	2020-11-23	2021-09-10

이 인스턴스에서 사번과 부서번호는 중복된 값을 가질 수 있게 되어, 릴레이션의 적법한 인스턴스가 아니게 됩니다. 이 문제는 근무 관계에 설명형 속성을 여러 개 두고 싶다는 것에서 발생합니다. 이 문제는 아래와 같이 근무시작일과 근무종료일을 속성으로 가지는 새로운 개체집합을 도입하면 해결됩니다.

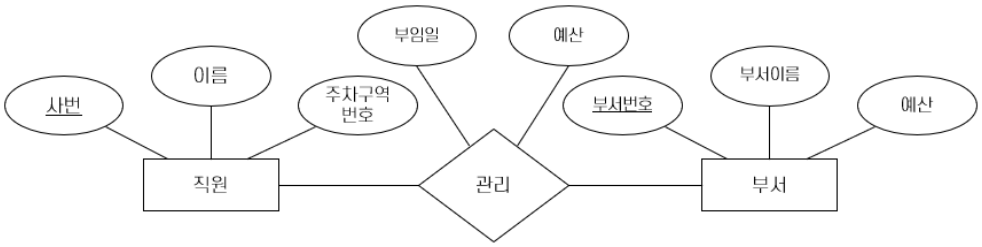


ER 모델의 어떤 버전에서는 속성의 값이 집합인 것을 허용합니다. 이런 기능을 이용할 수 있다면 기간 정보를 개체집합으로 독립시키지 않고 근무 관계의 한 속성으로 만들 수 있습니다. 그렇지만 이런 집합 값을 가지는 속성을 관계 모델로 표현할 때는 관계 모델에서는 집합 값을 갖는 속성을 지원하지 않기 때문에 기간 정보를 개체집합으로 취급한 것과 같은 릴레이션 스키마를 얻게 됩니다.

개체 or 관계

- 개체로 모델링하는 경우
  - 관계가 관계로 생성되는 한 개체에 국한되는 속성을 가짐
  - A 개체집합의 개체가 정확히 B 개체집합의 한 개체에만 해당
- 관계로 모델링 하는 경우
  - 관계집합에서 중복이 발생하는 경우
  - 두 개체 사이에 일어나는 동작을 기술하는 경우

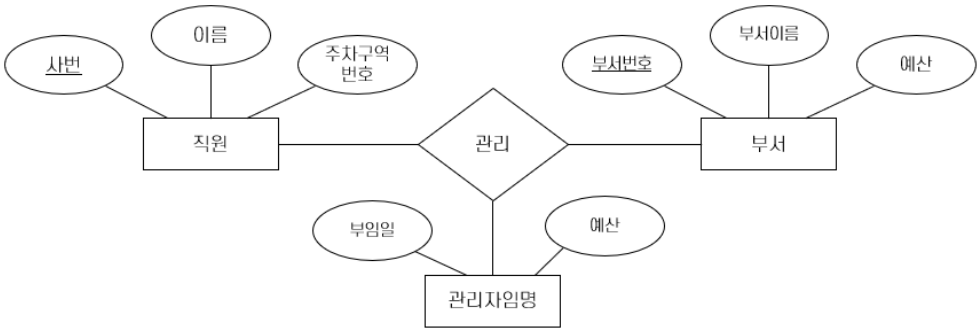
아래와 같은 관리 관계집합을 생각해봅시다. 각 부서의 부서장은 예산을 가지고 있습니다.



한 부서에는 한 명의 관리자가 있지만, 한 관리자는 여러 부서를 관리할 수 있는 경우, 관리 관계집합에 부임일자와 예산을 저장합니다. 요구사항이 여기까지 라면 설계는 자연스럽습니다.

예산액에 부서별로 할당되는 것이 아니라, 관리자에게 할당되는 경우라고 생각하면, 관리 관계집합에는 예산이 같은 값으로 저장됩니다. 이런 중복성은 일반적으로 매우 중요하며, 중복된 데이터는 여러 문제를 일으킬 수 있습니다.

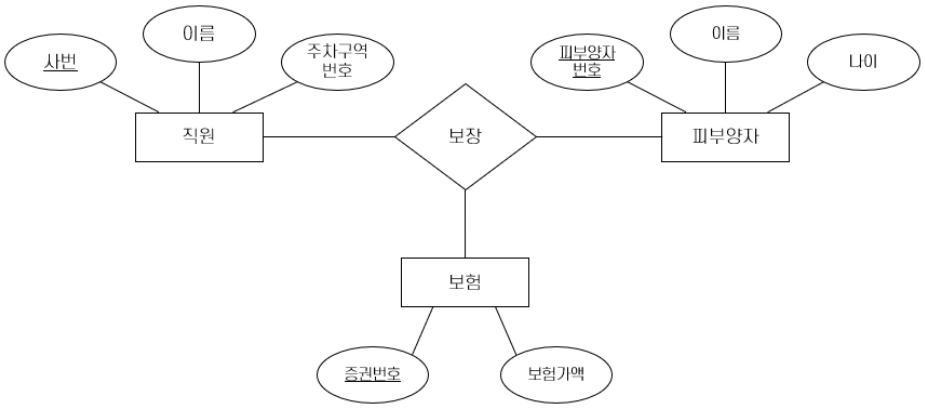
이런 경우, 직원을 부서 그룹의 관리자로 입명하고 그에 대해 예산을 책정하여 해결할 수 있습니다. 임명 내용을 관리자임명이라는 개체 집합으로 모델링하고 관리자, 부서를 3진관계로 연관시킵니다. 임명에 대한 세부 내용(부임일, 예산 등)은 임명에 속한 각 부서에 대해 반복되지 않습니다.



이진 관계 or 삼진 관계

- 이진 관계
  - 약 개체가 포함되는 경우
  - 관계의 변화가 개체에 영향을 미치는 경우
- 삼진 관계
  - 관계의 변화가 개체에 영향을 미치는 경우
  - 식별 관계의 체인이 존재하지 않는 경우

아래 ER 다이어그램은 한 직원이 보험에 가입 할 수 있고 각 보험에는 여러 직원이 가입할 수 있으며 피부양자 한 명은 여러 보험에 들어있을 수 있는 상황을 모델링 한 것입니다.

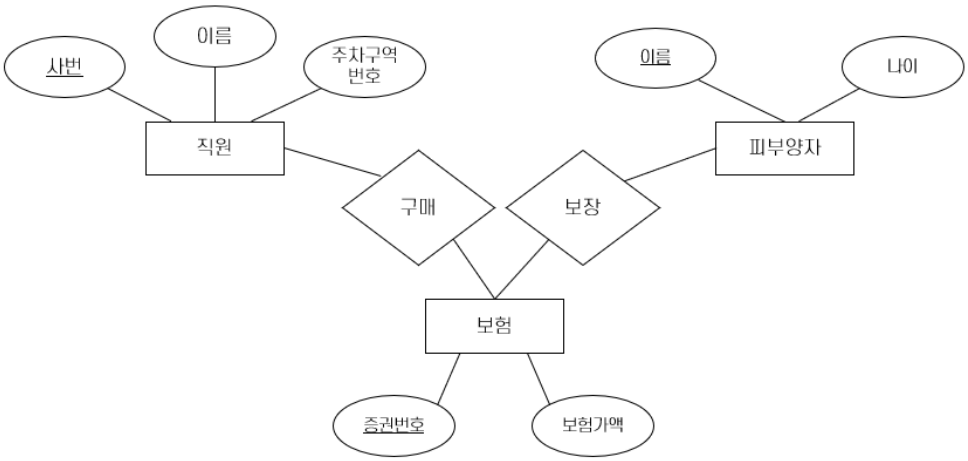


다음과 같은 요구사항이 추가된다고 가정해 봅시다.

- 보험 하나를 둘 이상의 직원이 함께 소유할 수 없다.
- 각 보험증권은 어떤 직원인가가 반드시 소유해야 한다.
- 피부양자는 약 개체 집합이며, 각 피부양자 개체는 그에 대한 피부양자 이름과 그 피부양자를 보장하는 보험증권 개체의 증권번호를 조합해야 유일하게 식별할 수 있다.

첫 번째 조건으로 인해 보장 관계에서 보험에 대해 키 제약조건을 설정해야 할 것 같으나, 그렇게 되면 한 증권이 한 명의 피부양자에게만 보장되게 됩니다. 두 번째 조건은 보험 개체집합에 대해 전체 참여 조건을 설정하게 됩니다. 세 번째 조건은 이진 관계를 만들어야 하는 조건입니다.

세 번째 조건을 무시한다 해도 가장 좋은 모델링 방식은 아래와 같이 두 개의 이진 관계를 사용하는 것입니다.



ER 다이어그램을 관계 모델로 변환하면 아래와 같이 됩니다.

```
CREATE TABLE 보험 (
      증권번호          int,
      보험가액          decimal,
      사번              int     NOT NULL,.

      CONSTRAINT pk_보험 PRIMARY KEY(증권번호),
      CONSTRAINT fk_보험_직원 FOREIGN KEY(사번) REFERENCEWS 직원(사번) ON DELETE CASCADE
)

CREATE TABLE 피부양자 (
      이름              varchar(10),
      나이              int,
      증권번호          int,

      CONSTRAINT pk_피부양자 PRIMARY KEY(이름, 증권번호),
      CONSTRAINT pk_피부양자_보험 FOREIGN KEY(증권번호) REFERENCES 보험(증권번호)
      ON DELETE CASCADE
)
```

한 직원을 삭제하면 그 직원이 소유한 모든 보험들이 따라서 삭제되며, 그 보험의 보장을 받는 모든 피부양자들도 삭제됩니다. 또한 피부양자 테이블에서 증권번호는 기본 키의 일부이므로 암시적으로 NOT NULL 제약조건이 지정되어 각 피부양자들은 자신을 보장하는 증권이 하나씩 있게 됩니다.

검토

- 데이터베이스 설계 개요
- 개체 관계 데이터 모델
- ER 모델 기능
- 개념적 설계 고려사항

Quiz

1. 다음 용어들을 간단히 설명하세요.
   - 속성(Attribute)
   - 도메인(Domain)
   - 개체(Entity)
   - 관계(Relationship)
   - 개체집합(Entity Set)
   - 관계집합(Relationship Set)
   - 일대다 관계(One-to-Many Relationship)
   - 다대다 관계(Many-to-Many Relationship)
   - 참여 제약조건(Participation Constraint)
   - 중첩 제약조건(Overlap Constraint)
   - 포괄 제약조건(Covering Constraint)
   - 약 개체 집합(Weakly Entity Set)
   - 역할 지시자(Role Indicator)
2. 어떤 대학 데이터베이스에 교수(ProfessorID로 식별됨)와 과목(LectureID로 식별됨)에 대한 정보가 저장되어 있습니다. 교수들은 과목을 강의하는 관계에 있습니다. 강의 관계 집합이 다음 각 상황에 있다고 할 때, 각 상황에 맞는 ER 다이어그램을 표시하세요.
   A. 교수는 같은 과목을 여러 학기에 강의할 수 있는데 이 개설학기들을 기록해 두어야 한다.
   B. 교수가 같은 과목을 여러 학기에 강의할 수 있는데 가장 최근의 개설 학기만 기록하면 된다.
   C. 모든 교수들은 무슨 과목인가는 반드시 강의하여야 한다.
   D. 각 교수들은 한 과목을 강의한다.
   E. 각 교수들은 정확히 한 과목을 강의하며 각 과목을 강의하는 교수가 반드시 있어야 한다.
   F. 어떤 과목을 교수진이 연합해서 강의할 수 있으나 이 교수진에 속한 어느 한 교수도 이 과목을 가르칠 수 없다. 필요하다면 개체 집합이나 관계집합을 더 도입해도 무방하다.

3. 다음과 같은 요구사항이 있는 대학 데이터베이스의 ER 다이어그램을 작성하세요.

   A. 교수는 ProfessorID, Name, Age, Grade, ResearchField를 가진다.

   B. 과제는 SubjectID, SupportOrg, StartDate, EndDate, Budget을 가진다.

   C. 대학원생은 StudentID, Name, Age, Degree를 가진다.

   D. 각 과제는 한 교수에 의해서 관리된다.

   E. 각 과제는 한 사람 이상의 교수에 의해 수행된다.

   F. 한 교수가 여러 과제를 관리할 수도 있고 수행할 수도 있다.

   G. 한 과제는 한 명 이상의 대학원생에 의해 수행된다.

   H. 대학원생이 한 과제를 수행할 때는 어떤 교수가 이 작업을 감독하여야 한다. 한 대학원생이 여러 프로젝트를 수행할 수 있는데 각 수행과제마다 감독자가 있어야 하며 감독자는 서로 달라도 무방하다.

   I. 학과는 DepartmentID, Name, Office를 가진다.

   J. 각 학과마다 그 학과를 운영하는 교수가 한 명 있다.

   K. 한 교수가 여러 학과에서 근무할 수 있는데, 이때 학과별로 참여 백분율이 기록된다.

   L. 대학원생은 전공 학과가 하나씩 있다.

   M. 대학원생은 조언자 역할을 하는 선배가 있다.

4. 파인만 레코드사는 데이터베이스를 구축하고자 한다. 아래 요구사항에 맞는 데이터베이스 개념 스키마를 설계하고 이에 따라 ER 다이어그램을 작성하세요. 그리고, 설계에 따라 데이터베이스를 작성하는 SQL DDL 구문을 작성하세요.

   A. 파인만 레코드사에서 녹음을 한 음악가들은 MusicianID, Name Address, TelephoneNo를 가진다. 이 중 같은 주소를 사용하는 음악가들도 있는데, 한 주소에는 한 전화번호만 있다.

   B. 파인만 레코드사에서 녹음에 사용되는 악기는 종류(기타, 신디사이저, 드럼 등)와 음조(C, B-Flat 등)을 가진다.

   C. 녹음된 앨범은 Title, RegisteredDate, ISBN을 가진다.

   D. 모든 노래는 제목과 작가를 가진다.

   E. 한 음악가가 여러 악기를 연주할 수 있으며, 한 악기를 여러 음악가가 연주할 수 있다.

   F. 한 앨범에는 여러 곡이 들어있지만, 노래 한 곡은 단 하나의 앨범에만 수록 될 수 있다.

   G. 여러 음악가가 협동하여 한 곡을 만들 수 있고, 한 음악가는 여러 곡을 만들 수 있다.

   H. 각 앨범에는 프로듀서로 한 음악가가 참여한다. 한 음악가는 여러 곡의 프로듀서가 될 수 있다.