데이터베이스
강의 노트

제 11 회차
개념적 설계 단계 심화

❖ 학습목표

- 개체의 심화 요소를 나열할 수 있다.
- 관계의 심화 요소를 나열할 수 있다.
- 속성의 심화 요소를 나열할 수 있다.
- 슈퍼-서브 타입의 3가지 정제 방법을 설명할 수 있다.
- 속성을 개체로 전환하는 경우를 설명할 수 있다.

❖ 학습내용

- ER 모델 세분화 방법
- ER 모델 정제 방법

ER 모델 세분화 방법

1. 개체의 심화 요소
2. 관계의 심화 요소
3. 속성의 심화 요소
4. 관계 행렬 활용

1. 개체의 심화 요소

1) 개체의 심화 요소

보다 세분화된 개념적 모델링을 위해서 개체를 다음과 같이 6가지로 세분화

키 개체 (Key Entity)

- 해당 업무에서 원래부터 존재하는 개체
- 다른 개체와의 관계에 의해 생성된 개체가 아닌, 원래 독립적으로 존재하는 개체
- 다른 개체의 부모 개체가 됨

예) 사원, 부서, 고객, 상품 등

메인 개체 (Main Entity)

- **키 개체들 간의 업무적인 관련성 때문에 생성**되는 것
- 해당 업무에서 핵심적으로 관리되는 데이터로부터 추출됨

예) 사고청구, 보험계약, 주문, 출하지시 등

액션 개체 (Action Entity)

- **2개 이상의 부모 개체로부터 생성되고 내용이 자주 바뀌는 개체**
- 초기 분석 단계에서는 잘 드러나지 않지만 모델링을 세분화하는 단계에서 도출됨

예) 청구내역, 계약진행, 주문내역, 출하상태이력 등

약한 개체 (Weak Entity)

- **독립적으로 존재하지 못하고 소유 개체(Owner Entity)가 있어야 존재**
- 고유한 식별자가 없기 때문에 자신의 구분자(부분 키)를 소유 개체의 식별자와 결합해서 복합 식별자를 지정해야 함

예) 부양가족 ← 소유 개체는 사원 개체

1. 개체의 심화 요소

1) 개체의 심화 요소

코드 개체 (Code Entity)

- **정보를 간단히 표현하기 위한 기호를 나타내는 개체**
- 실제 업무에서 사용하는 코드도 있지만 데이터를 효율적으로 표현하기 위해서 생성하는 경우도 있음
  예) 전공 코드 개체 – 1: 컴퓨터, 2: 전자, 3: 전기, 4: 기계, ...
- 너무 많은 코드 개체를 식별하면 개념적 모델링이 너무 복잡해지므로 반드시 개념적 설계 단계에서 식별할 필요는 없음

관계 개체 (Association Entity)

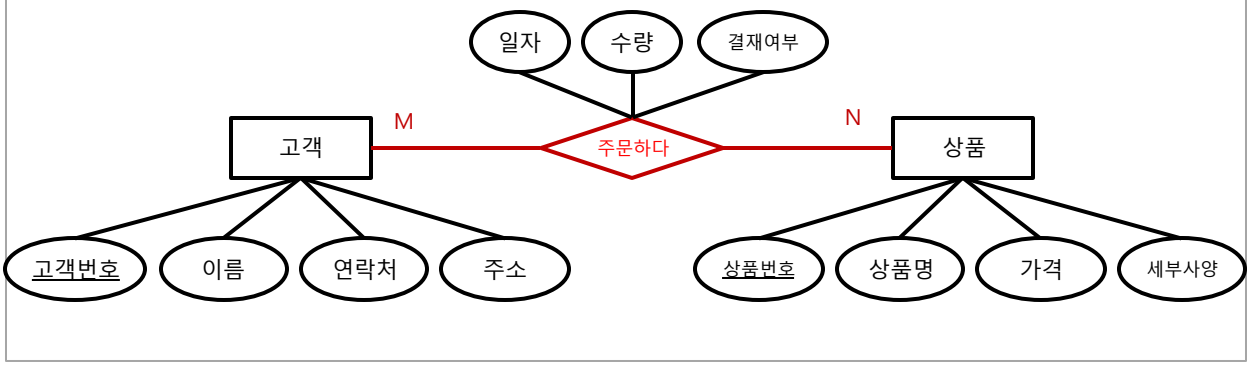
- **두 개체 간의 M : N 관계로 인해 생성되는 개체로, 교차 개체라고도 함**
- 피터 첸 표기법으로 표현할 때 : M : N 관계도 마름모로 표현
- IE 표기법으로 표현할 때 : M : N 관계를 정확히 표현하기 위해서
  관계 개체를 생성해서 표현
- 관계 개체의 식별자는 양쪽 개체의 식별자를 합친 복합 식별자로 지정

- 예) '주문한다' 관계 개체 표현

1. 개체의 심화 요소

1) 개체의 심화 요소

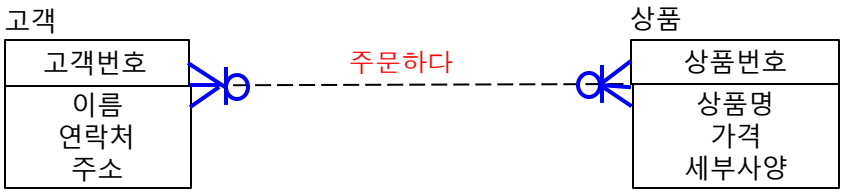
피터 첸 표기법



IE 표기법

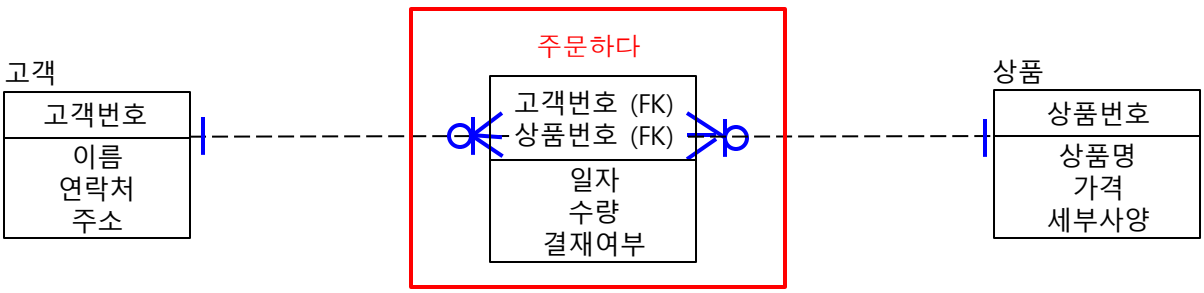
① 기본 표현

- M : N 관계 유형을 까마귀 발로 표현하고, 관계 이름(주문하다)만 표현한다.



② 관계 개체 추가 표현

- 관계 개체(주문하다)를 추가해서 관계의 모든 속성을 표현하고, 기존 기체와는 1 : N 관계로 표현할 수 있다.



2. 관계의 심화 요소

보다 세분화된 개념적 모델링

1) 병렬 관계(Parallel Relationship)

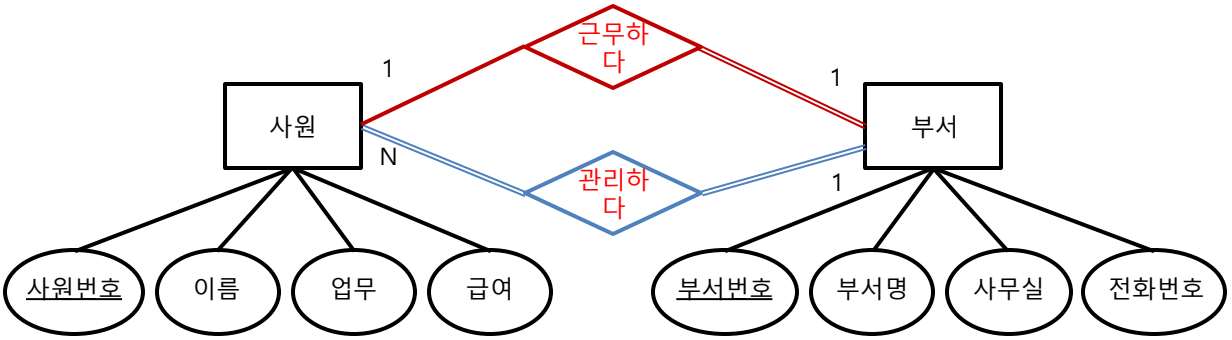
병렬 관계란?

두 개체 사이에 두 개 이상의 관계가 존재하는 것으로, 다중 관계라고도 함

예제 1

사원과 부서 개체 사이의 '근무하다'라는 관계와 '관리하다'라는 2개 관계 존재

- 근무하다 : 모든 사원 개체가 참여하는 **전체 참여**
- 관리하다 : 일부 사원, 즉 부서장에 해당하는 사원만 참여하는 **부분 참여**

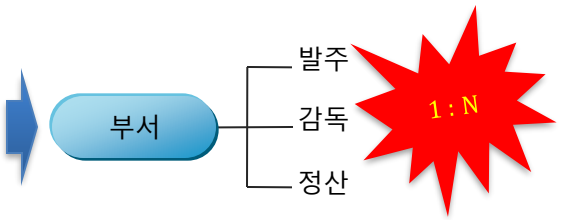


2. 관계의 심화 요소

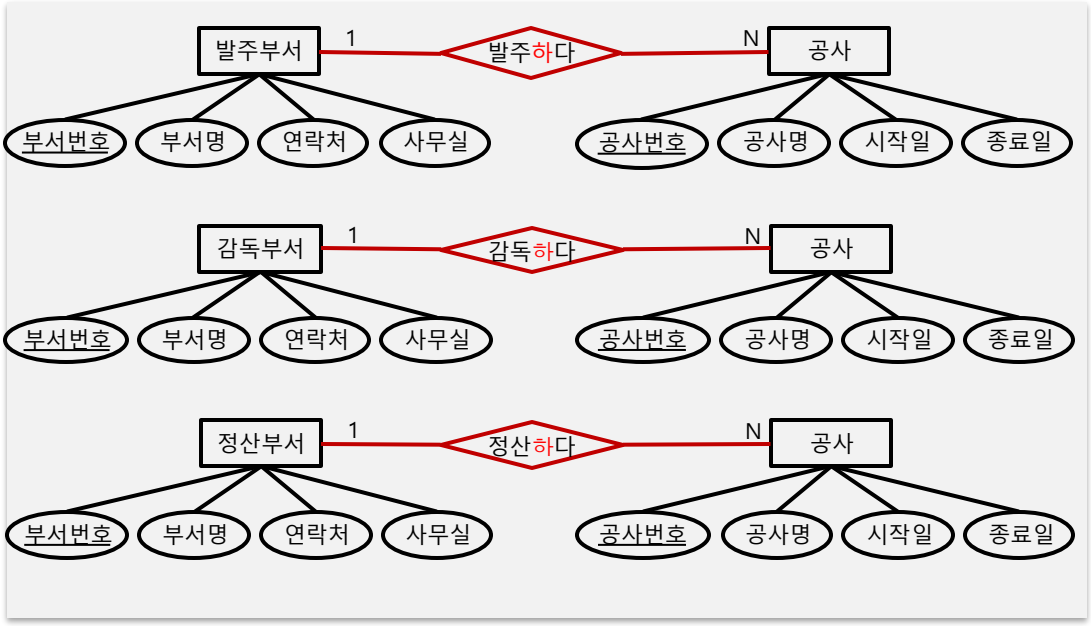
1) 병렬 관계(Parallel Relationship)

예제 2

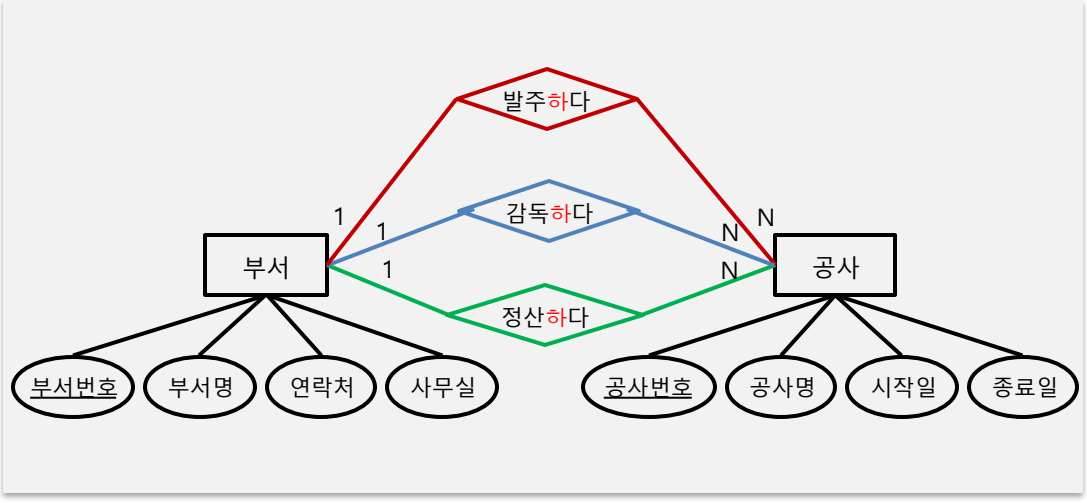
부서 개체와 공사 개체 사이의 관계를 분석



각각 표현하는 경우



병렬 관계로 표현하는 경우



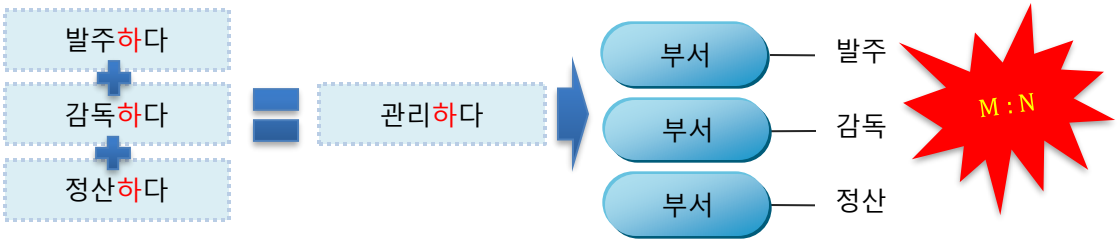
2. 관계의 심화 요소

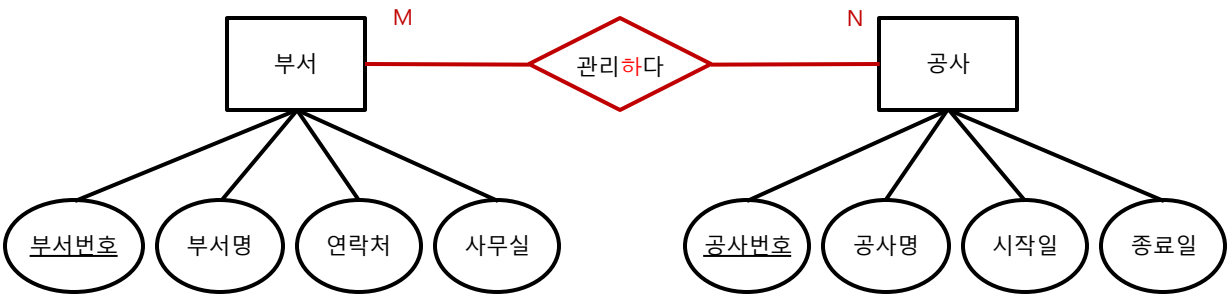
2) 직렬 관계(Serial Relationship)

직렬 관계란?

병렬 관계로 나열한 관계를 하나로 통합해서 M : N 관계로 바꾸어서 표현한 것

예제





2. 관계의 심화 요소

3) 병렬관계와 직렬관계의 특징 비교

병렬 관계의 특징	직렬 관계의 특징
관계를 표현하기 위해 별도의 테이블(릴레이션)이 불필요함	관계를 표현하기 위해 별도의 테이블(릴레이션)이 필요함
하나의 관계가 하나의 속성으로 표현됨	하나의 관계가 하나의 개체로 표현됨
새로운 관계가 추가되거나 변경될 때 유연하게 대처하기 어려움	새로운 관계가 추가되거나 변경될 때 유연하게 대처할 수 있음
관계 내용별로 상세 정보를 관리할 수 없음	관계 내용별로 상세 정보를 관리할 수 있음

4) 순환 관계(Self Relationship)

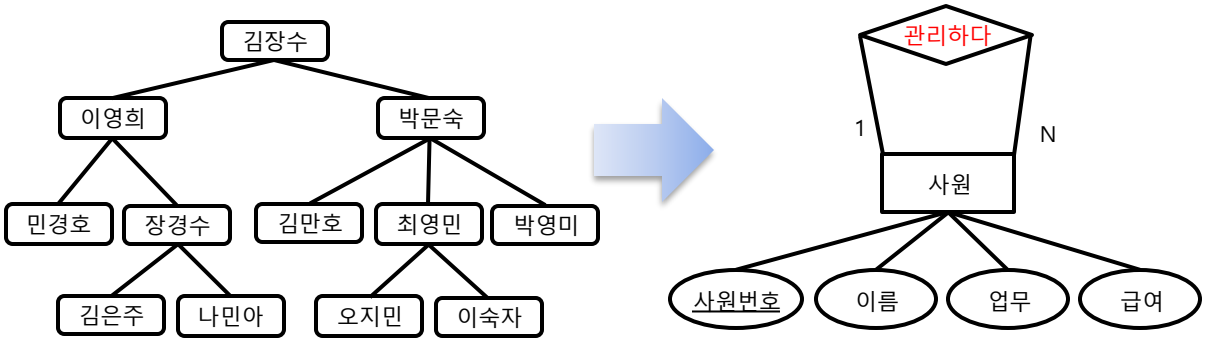
순환 관계란?

하나의 개체가 다른 개체가 아닌 자기 자신과 관계를 맺는 것을 의미

1 : N 순환 관계

기관의 조직도, 관리자 정보 등과 같이 계층 구조를 표현할 때 발생

사원의 관리자 관계 표현



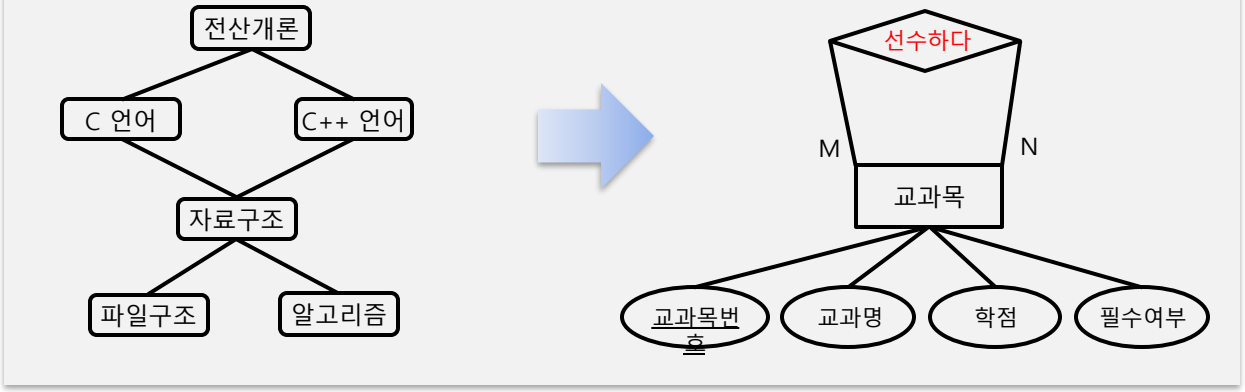
2. 관계의 심화 요소

4) 순환 관계(Self Relationship)

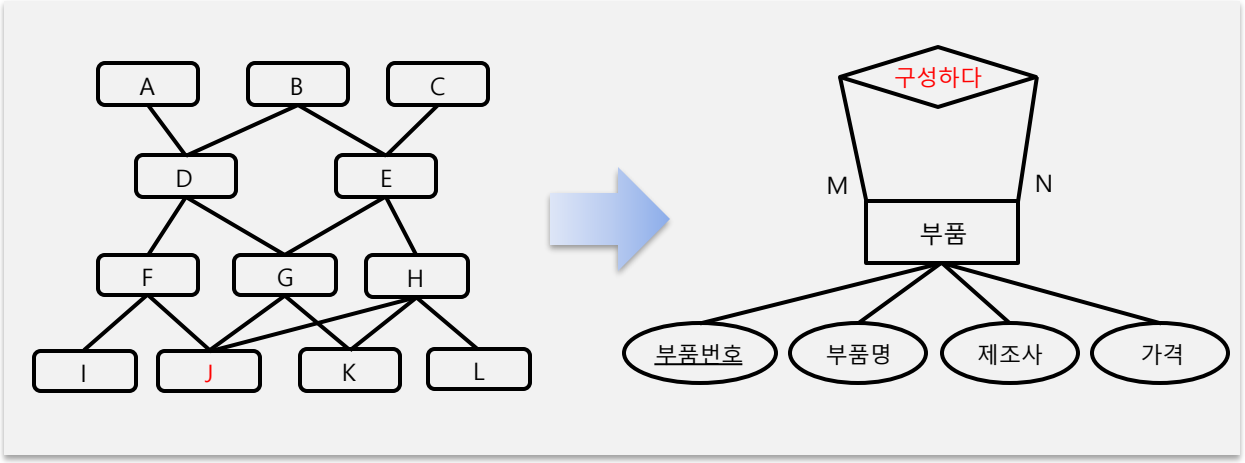
M : N 순환 관계

선수 과목이나, 구성부품 정보 등과 같이 네트워크 구조를 표현할 때 발생

선수 과목 관계 표현



구성 부품 관계 표현



3. 속성의 심화 요소

1) 속성 유형

기초 속성 (Basic Attribute)	• 업무로부터 추출된 일반적인 속성 • 데이터 요구 분석 명세서에 포함되어 있으며, 현업에서 제공해야 속성이 유지될 수 있음 • 예: 상품명, 가격, 주문수량 등
설계 속성 (Designed Attribute)	• 원래 존재하지는 않지만 **필요에 따라 설계자가 추가**한 속성 • 데이터 요구 분석 명세서에 포함되어 있지는 않지만, 설계를 진행하면서 새로 생성됨 • 대부분의 코드 속성이나 일련번호처럼 식별자 역할을 하도록 추가된 속성이 해당함 • 예: 주문번호, 예약번호, 고객번호, 상품코드 등
유도 속성 (Derived Attribute)	• 추출 속성이라고도 칭하며, 기본 속성으로부터 **계산 등의 가공 처리**를 통해서 생성된 속성 • **저장 속성 (Stored Attribute)***의 영향을 받으므로, 저장 속성의 값이 변경되면 함께 변경됨 • 중복의 의미가 있으므로 대개 **개념적 모델링 단계에서 식별하지 않음** (반드시 필요한 경우라면 별도로 정리해서 구현 단계에서 참조함) • 예: 나이(생년월일 속성에서 유도됨), 근무기간(입사일 속성에서 유도됨) 등

4. 관계 행렬 활용

1) 관계 행렬 활용

관계 행렬이란?

개체들 간의 관계를 정의하기 위해서 사용하는 보조 도구

관계 행렬 작성 방법

1. 개체를 가장 상위 행과 가장 좌측 열에 모두 표시함
2. 개체들 간의 관계 유무를 셀(Cell)에 표시함
3. 1차적으로 조금이라도 관련성이 있으면 모두 표시함
4. 식별된 관계의 구체적인 이름을 부여함
5. 상관 관계가 없으면 셀에 '-'을 표시함
6. 1열의 개체가 주어에 해당함
7. 순환 관계는 대각선 셀에 표시함
8. 관계 행렬의 내용을 참조해서 최종 관계를 판단함

관계 행렬 작성 예제

	고객	부품	주문	창고
고객	-	주문하다	-	-
부품	-	구성하다	포함되다	-
주문	-	발주하다	-	-
창고	-	보관하다	-	-

ER 모델 정제 방법

1. 슈퍼-서브 타입 정제
2. 카테고리 정제
3. 속성을 개체로 전환하기
4. 개체·관계·속성 검증

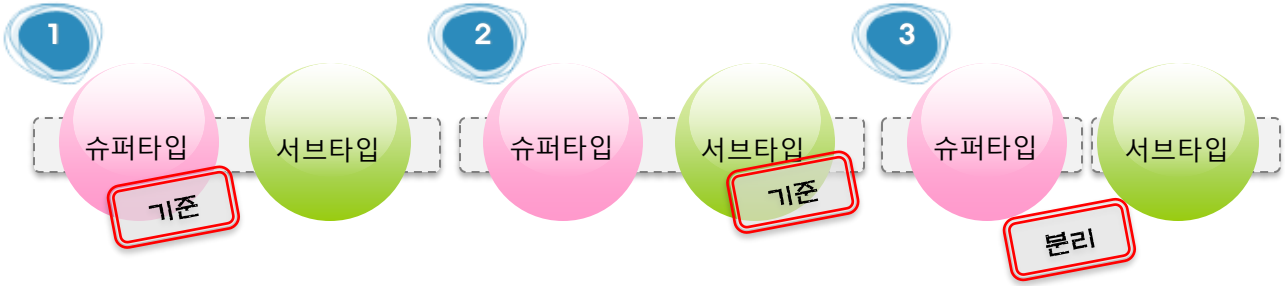
1. 슈퍼-서브 타입 정제

1) 슈퍼-서브 타입 정제 기준 및 방법

슈퍼-서브 타입 정제 기준

1. 서브타입 개체에 독자적으로 고유한 속성이 없거나 독자적인 관계가 없는 경우 속성으로 표현함
2. 서브타입으로 분할했을 때 개념적 모델링이 너무 복잡해지면 분할하지 않음
3. 서브 타입으로 분할된 수가 너무 많은 경우, 각각을 독립된 개체로 분리할 것을 고려함

슈퍼-서브 타입 정제 방법



1. 슈퍼-서브 타입 정제

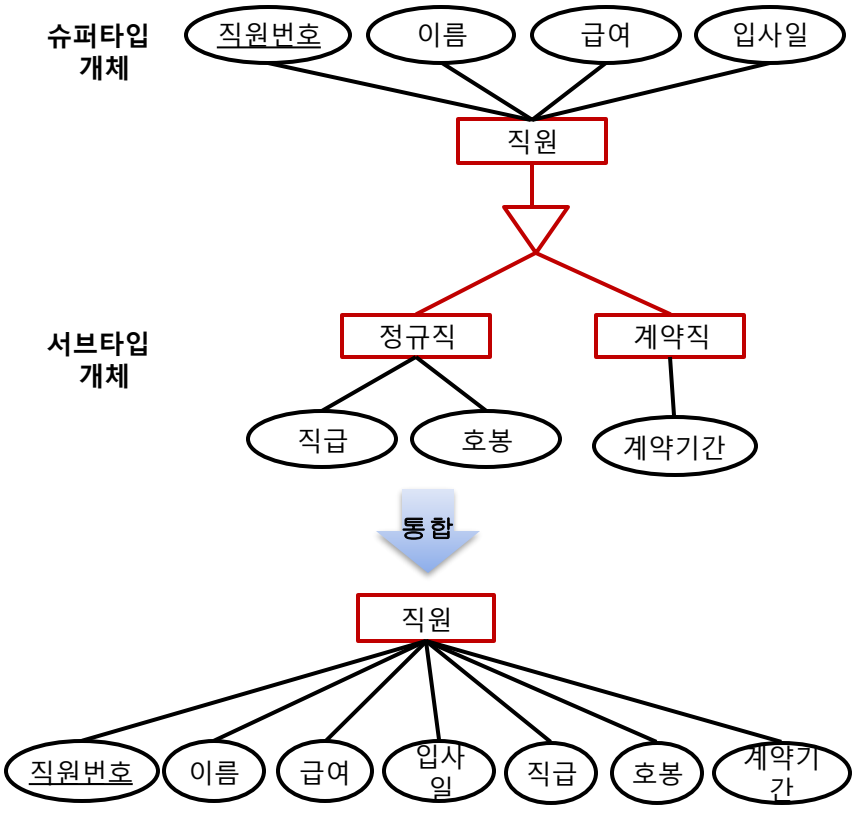
2) 슈퍼타입 개체 기준 통합 방법

서브타입 개체에 있던 속성을 모두 슈퍼타입 개체에 포함 → 서브타입 개체와 연결된 관계를 모두 슈퍼타입 개체와 연결 → 서브타입 개체를 모두 삭제

슈퍼타입 개체를 기준으로 통합하는 경우

1. 응용 프로그램에서 서브타입을 구분해서 처리하지 않는 경우
2. 서브타입 개체에 속하는 데이터를 명확하게 구분하기 어려운 경우
3. 서브타입 개체에 사용되는 속성 수가 매우 적은(1-2개) 경우

예제 정규직과 계약직이라는 2개의 서브타입 개체를 직원이라는 슈퍼타입 개체에 통합하기 (대부분의 응용 프로그램에서 정규직과 계약직을 구분해서 처리하지 않는 경우)



1. 슈퍼-서브 타입 정제

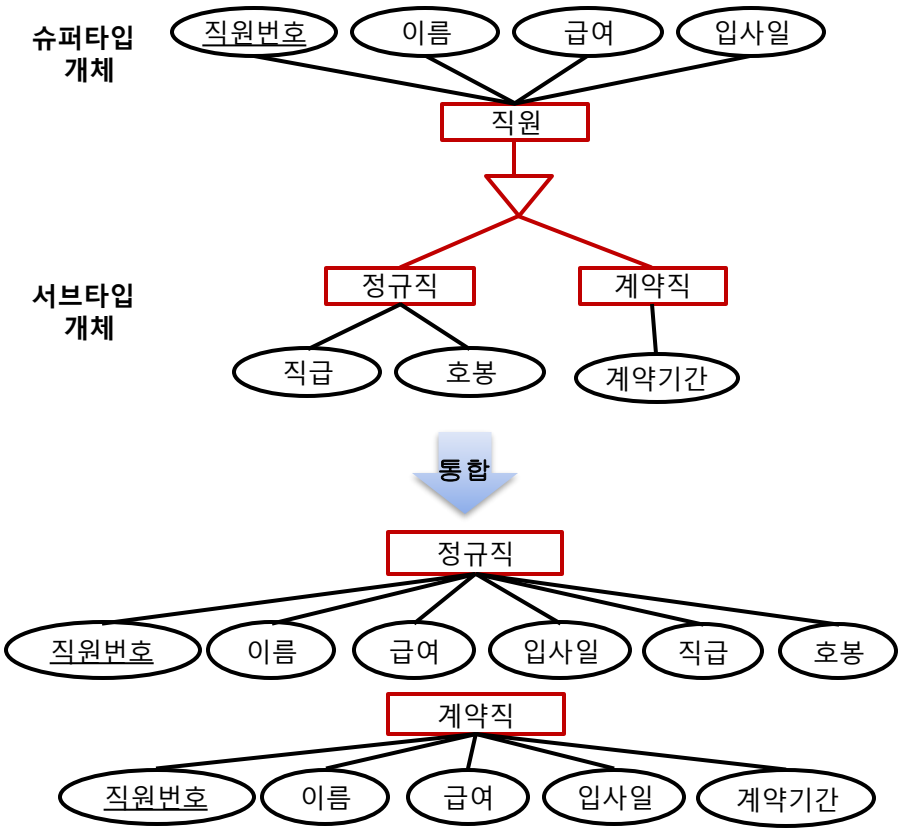
3) 서브타입 개체 기준 통합 방법

슈퍼타입 개체에 있던 모든 속성을 각각의 서브타입 개체로 이동 → 슈퍼타입 개체에 연결된 관계를 각각의 서브타입 개체에 연결 → 슈퍼타입 개체 삭제

서브타입 개체를 기준으로 통합하는 경우

1. 슈퍼타입 개체에 속하는 속성의 수가 많지 않은 경우
2. 슈퍼타입 개체의 관계가 적은 경우
3. 슈퍼타입 개체에 접근하는 응용 프로그램의 수가 적은 경우

예제 직원이라는 슈퍼타입 개체를 정규직과 계약직이라는 2개의 서브타입 개체와 통합하기 (대부분의 응용 프로그램에서 정규직과 계약직을 확실히 구분해서 처리하는 경우)



1. 슈퍼-서브 타입 정제

4) 슈퍼타입과 서브타입 개체를 각각 분리하는 방법

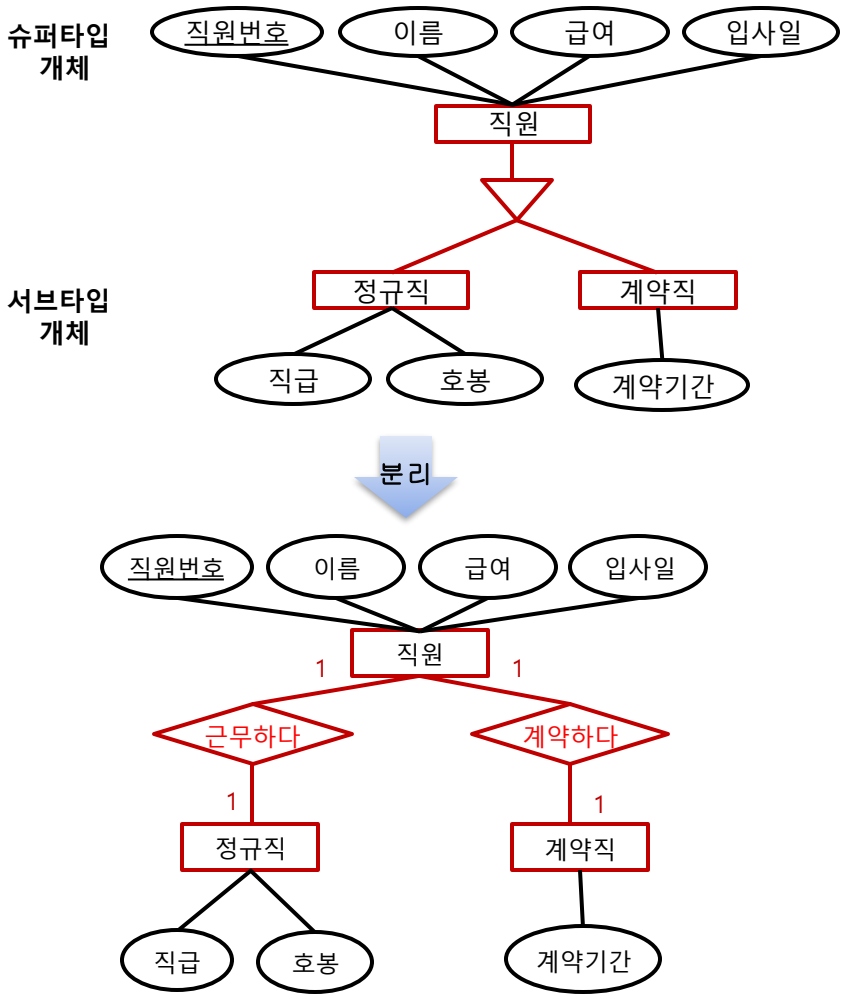
슈퍼타입, 서브타입 개체를 각각 개별 개체로 분리 → 분리된 슈퍼타입 개체와 서브타입 개체를 1 : 1 관계로 연결

슈퍼타입, 서브타입 개체를 각각 분리하는 경우

데이터 모델의 유연성을 보장해야 하는 경우

예제

슈퍼 타입인 직원 개체와 서브 타입인 정규직 및 계약직 개체를 각각 분리시키기
(정규직과 계약직을 확실히 구분해서 처리하는 응용 프로그램과 구분 없이 처리하는 응용 프로그램의 수가 비슷하고, 추후 다양한 응용 프로그램이 추가될 예정인 경우)



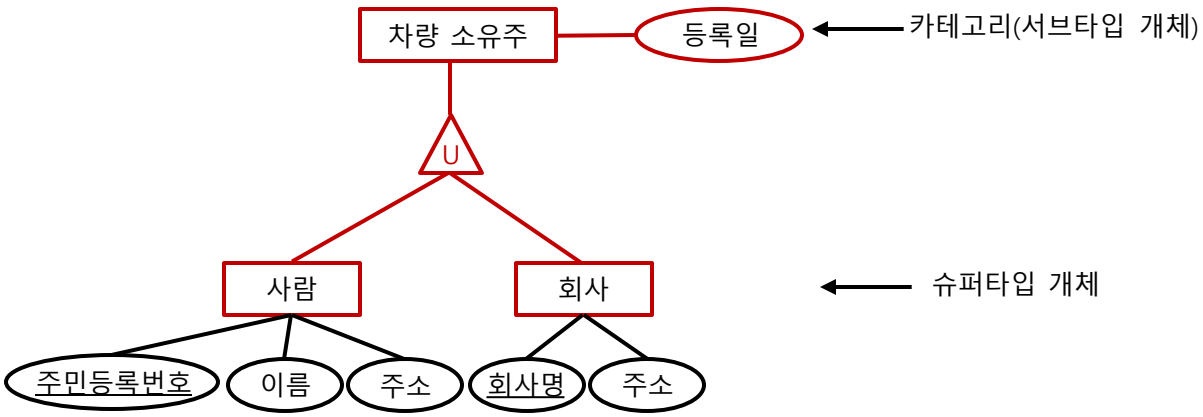
2. 카테고리 정제

1) 카테고리 정제 기준

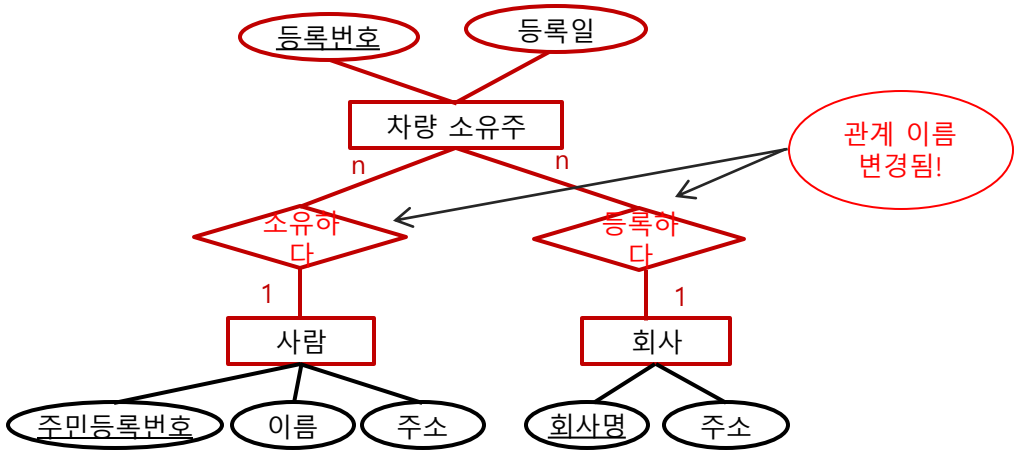
카테고리(서브 타입 개체)를 독립된 개체로 만들고, 기본 키를 포함시킴

카테고리에 속하는 모든 개체 타입(슈퍼 타입 개체)을 카테고리 개체와 각각 1: N 관계로 연결

예제



정제

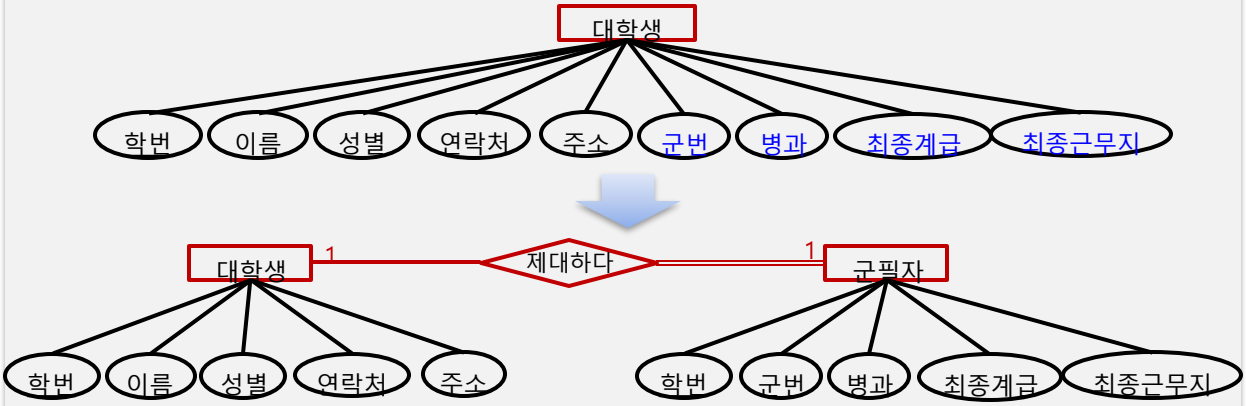


3. 속성을 개체로 전환하는 경우

1) 속성 값이 없는 경우와 여러 부분으로 구성된 경우

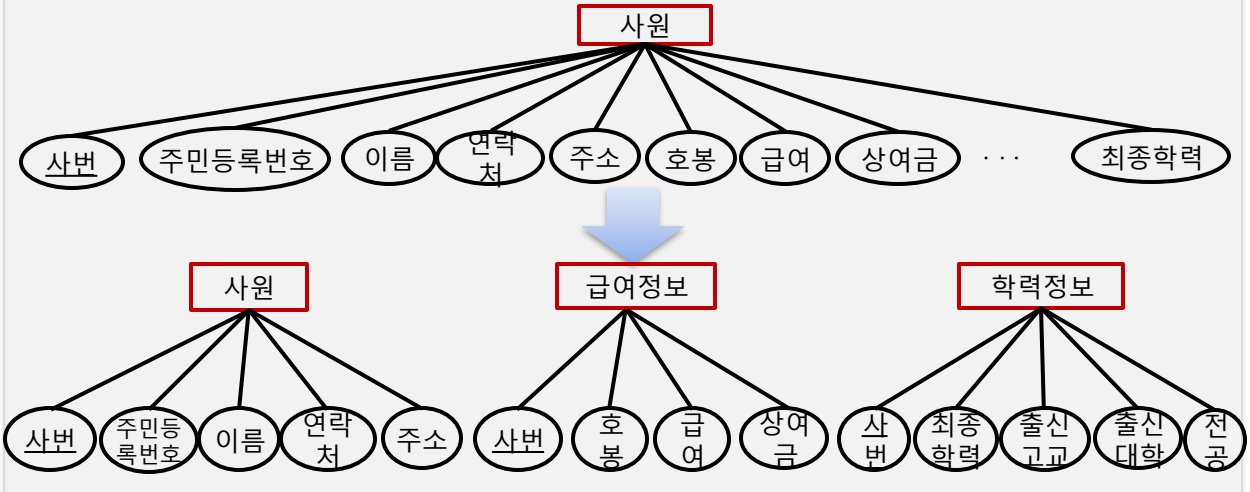
속성 값이 없는 경우

- 많은 개체가 널(NULL) 값을 갖는 속성은 별도의 개체로 분리하고, 원래 개체와 1 : 1 관계로 연결
- 예 : 병력 관련 속성을 포함하는 대학생 개체
  - 군 미필이거나 여학생인 경우, 해당 속성들은 모두 널 값을 갖게 됨
  - 병력 관련 속성들을 모아서 별도의 군필자 개체를 생성하고, 해당 대학생과 1 : 1 관계를 표시함



속성 값이 여러 부분으로 구성된 경우

- 속성의 수가 많으면서 특징을 구분할 수 있는 여러 부분으로 구성된 경우, 특징 별로 별도의 개체로 분리
- 예 : 개인신상정보, 급여정보, 학력정보 등으로 구분되면서 수십 개의 속성을 갖는 사원 개체
  - 개인신상정보, 급여정보, 학력정보 등을 각각 별도의 개체로 분리함



4. 개체·관계·속성 검증

1) 개체·관계·속성 검증 방법

개체 검증 방법

1. 유용한 정보를 제공하는가?
   - 업무에 활용되지 않고 값이 변하지 않는 개체는 삭제함
2. 유일한 식별자(후보 키)를 포함하고 있는가?
   - 식별자가 없으면 데이터의 일관성과 무결성을 보장하지 못하므로, 필요하다면 설계 속성을 추가함
3. 속성의 수가 2개 이상인가?
   - 속성이 하나 밖에 없는 경우, 다른 개체의 속성으로 표현하는 것이 바람직함
   - 식별자가 2개의 속성으로 결합된 경우라도 2개 속성으로 인정함
4. 다른 개체와 관계(Relationship)가 있는가?
   - 일부 코드 개체나 통계용 개체를 제외하면 모두 다른 개체와 관계가 있어야 함
5. 실제 데이터가 2개 이상 존재하는가?
   - 데이터가 하나 밖에 없다면 업무적으로 관리할 필요가 없는 개체일 가능성이 크기 때문에 삭제함

4. 개체·관계·속성 검증

1) 개체·관계·속성 검증 방법

관계 검증 방법

1. 관계가 현재 업무 규칙에 적절한 것인가?
2. 관계명이 두 개체 사이의 업무적 연관성을 표현하는 구체적인 이름인가?
3. 관계의 유형이 적절한가?
4. 관계의 카디널리티가 적절한가?
5. 필수/선택 여부, 즉 전체 참여와 부분 참여가 적절한가?

속성 검증 방법

1. 각기 다른 의미를 갖는 여러 개의 속성을 묶어서 하나의 속성으로 정의한 것은 아닌가?
   - 그렇다면 각각을 별개의 속성으로 구분해야 함
2. 속성이 단 하나의 값만 갖는 것이 아닌가?
   - 모든 개체가 동일한 하나의 값만을 갖는 속성이 있다면 제거해야 함
   - 예) 한국인 개체의 국적 속성
3. 코드 값을 갖는 속성이 현업에서 통상적으로 사용되는 코드 값을 갖는가?
   - 그렇지 않다면 코드 사용을 억제해야 한다.
4. 전산 처리에 필요한 플래그(Flag) 형태의 속성이 아닌가?
   - 플래그 형태의 속성은 제외시켜야 함

쉼터

생각하라, 발견하라, 창조하라!

존재하지 않는 것을 상상할 수 없다면,
새로운 것을 만들 수 없다.

남의 눈을 통해서만 세상을 보면,
다른 사람이 묘사하는 세계에만 머무르게 된다.

- 폴 호건(화가)