1 키 제약조건을 반영하는 사상

01 키 제약조건을 반영하는 사상

1 엔티티타입(Entity Type)과 속성(Attribute)의 사상

엔티티타입의 사상

- 바로 테이블(Relation)로 사상함

속성의 사상

- 복합(Composite) 속성 : 단순 속성만 포함함
- 다치(Multi-valued) 속성 : 새로운 테이블을 만듦

01 키 제약조건을 반영하는 사상

1 엔티티타입(Entity Type)과 속성(Attribute)의 사상

- 예) 복합 속성의 사상
  - 사원(사번, 사원명(성, 이름)) →
    사원(사번, 성, 이름)

- 예) 다치 속성의 사상
  - 고객(주민#, 이름, {연락처}) →
    고객(주민#, 이름)
    고객_연락처(주민#, 연락처) FK={주민#}

01 키 제약조건을 반영하는 사상

2 관계타입(Relation Type)의 사상

🔎 고려 사항

1) Relationship의 제약 조건을 준수할 것
   - Degree(Arity)에 따라 unary/binary/ternary
   - Cardinality에 따라 1:n, n:m, 1:1
   - Participation 제약에 따라 total / partial
2) 키 무결성(key integrity)을 준수할 것
3) 널 값(null value)의 발생을 가능한 피할 것
4) 조인(join)연산의 횟수를 고려할 것
5) 검색 및 저장 성능을 고려할 것

01 키 제약조건을 반영하는 사상

2 관계타입(Relation Type)의 사상

사상 방법

- 내장 방법
  : 한 테이블의 주키를 다른 테이블의
  외래키로 내장시킴

- 관계 테이블 방법
  : 새로운 관계 테이블을 만들고 관계에 참여하는
  각 테이블들에서 주키를 빌려와 이를 외래키로 함

01 키 제약조건을 반영하는 사상

2 관계타입(Relation Type)의 사상

- 테이블이 갯수가 많아질수록 조인 연산이 비용이 증가하게 되고 응답시간이 늦어지게 되므로 되도록이면 테이블을 적게 만들려고 해야 함

- 따라서 내장 방법을 먼저 고려하고, 안 되면 차선으로 관계 테이블 방법을 선택해야 함

01 키 제약조건을 반영하는 사상

3 1:N 사상에 있어서의 주키

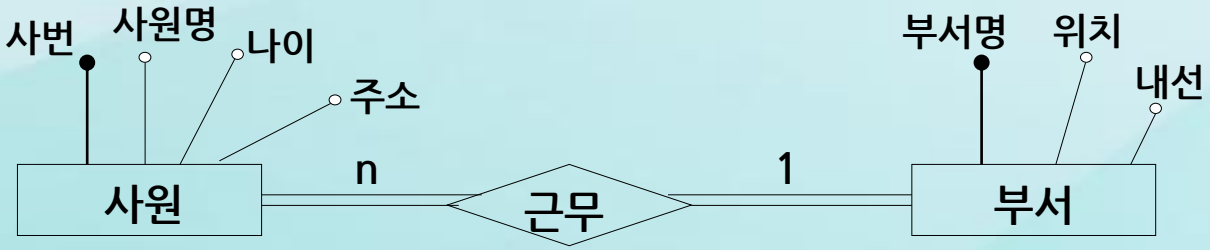
- 1:N 사상에 있어, 왜 1-측의 주키를 n-측으로 빌려 오는지, n-측의 주키를 1-측으로 빌려오면 안 되는지?
- 다치 속성(Multi-valued Attribute)은 허용하지 않음
- 키 제약조건(Key Constraint)을 위반하기 때문
- 1-측의 키를 n-측에서 받기 때문에 부모-자식 관계 사상이라고도 함

01 키 제약조건을 반영하는 사상

3 1:N 사상에 있어서의 주키

예제)
왜 부서 테이블에서 사번을 외래키로 사용하지 못할까?

- 다치 속성을 허용 안함
- 키 제약조건 위배



사원(<u>사번</u>, 사원명, 나이, 주소, 부서명) PK={사번}, FK={부서명}
부서(<u>부서명</u>, 위치, 내선) PK={부서명}

01 키 제약조건을 반영하는 사상

4 N:M 사상에 있어서의 주키

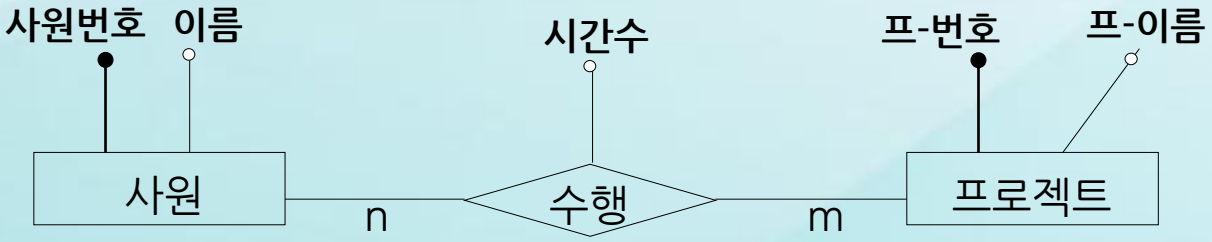
- N:M 사상에 있어서 왜 새로운 테이블을 생성할 수 밖에 없는지?
- 테이블의 개수를 되도록 적게 만드는 설계가 좋은 설계
- 따라서 내장 방식을 먼저 고려하지만 이 방식으로는 다:다 의 의미를 반영할 수 없기 때문에 차선으로 테이블을 새로 생성하는 방식을 선택
- 또한, 다:다의 의미를 표현하기 위해서는 관계에 참여하는 양쪽 테이블의 주키들을 빌려와서 조합하여 새로운 테이블의 주키(복합키 형태의)로 만들 수 밖에 없음

01 키 제약조건을 반영하는 사상

4 N:M 사상에 있어서의 주키

예제) 수행 관계를 사상하기 위해 사원이나
프로젝트 테이블에 내장방식으로 사상할 수 없는가?

- N:M의 의미를 반영하기 위해서는 새로운 테이블을 생성 할 수 밖에 없음



사원(<u>사원번호</u>, 이름) PK={사원번호}
프로젝트(<u>프-번호</u>, 프-이름) PK={프-번호}
수행(<u>사원번호, 프-번호</u>, 시간수) PK={사원번호, 프-번호}, FK={사원번호}, {프-번호}

01 키 제약조건을 반영하는 사상

5 식별 관계와 비 식별 관계

식별 관계(Identifying Relationship)는
존재 의존성을 표시

- 다른 테이블의 주키를 외래키로 빌려오고,
  그것을 주키의 일부가 되도록 함
- 주키 쪽 값이 제거되면 참조 무결성에 의해 같이
  제거되어야 함, 키 값을 null 로 설정하거나
  디폴트 값으로 설정하는 것은 허용되지 않음
- N:M 사상이 이에 해당됨

01 키 제약조건을 반영하는 사상

5 식별 관계와 비 식별 관계

- 비 식별 관계(Non-identifying Relationship)는 존재 독립성을 표시
  - 다른 테이블의 주키를 외래키로 빌려오지만, 그것이 주키의 일부가 되지는 않음
  - 주키 쪽 값이 제거되면, 같이 제거되는 것이 일반적이지만 참조 무결성을 위반하지 않는 범위에서 null 로 설정하거나 default 값으로 설정될 수도 있음
  - 1:N, 1:1 사상이 이에 해당됨

01 키 제약조건을 반영하는 사상

6 인공 키/대행 키의 사용

- N:M 사상은 주키가 복합키 형태로 주로 구성됨
- 복합키 형태는 인덱스를 구성하거나
  테이블을 관리할 때 참조 제약을 강제하기 때문에
  운영 시에 불편함이 존재
- 이런 불편함 때문에 임의의 순열 번호를 부여하여
  주키로 삼는 경우가 있음, 이런 키들을 인공키 또는
  대행키라고 함

01 키 제약조건을 반영하는 사상

6 인공 키/대행 키의 사용

🔎 N:M 사상에서 인공키를 사용하게 되면 존재의존성을 상실할 수 있음, 이 때는 UNIQUE, NOT NULL 설정을 해서 존재 의존성을 보존하는 것이 좋음

01 키 제약조건을 반영하는 사상

6 인공 키/대행 키의 사용

예)

수행(<u>사원번호, 프-번호</u>, 시간수) PK={사원번호, 프-번호}, FK={사원번호}, {프-번호}
→ 인공키/대행키 사용
수행(<u>수행번호</u>, 사원번호, 프-번호, 시간수) PK={수행번호}, FK={사원번호}, {프-번호}
이때 {사원번호, 프-번호} 에 대해 UNIQUE, NOT NULL 설정을 해 주는 것이 좋음

2 외래키 제약조건

02 외래키 제약조건

1 사상에 있어서의 외래키

- 외래 키는 테이블 간 참조의 연결고리 역할을 수행
- 외래 키 무결성 제약조건은 항상 만족해야 함
- CASCADE, SET DEFAULT, SET NULL
  옵션 사항은 주의 깊게 설정
- 외래키가 주키의 일부인 경우
  SET NULL 옵션을 설정 할 수 없음

02 외래키 제약조건

1 사상에 있어서의 외래키

- FK={a1, a2} 와 FK={a1}, {a2} 의 차이점
  - 다른 테이블의 주키가 복합키 형태 {a1, a2}일 때 이를 빌려와서 외래키로 만들 때는 외래키의 형태가 {a1, a2} 이 됨, 1:N 사상에서 1-측 테이블의 주키가 복합키 형태를 가질 때 외래키가 이런 형식으로 표시됨
  - 관계에 참여하는 테이블이 다수이고 각 테이블로부터 외래키를 빌려올 때는 각각 참조하는 테이블이 다르므로 {a1}, {a2} 형태로 표시됨, 보통 N:M 사상에서 이런 형식으로 많이 표시됨

02 외래키 제약조건

2 1:1 사상에 있어서의 외래키

- 관계에 참여하는 두 개 엔티티의 참여 제약에 따라 3가지 접근법이 존재함, 이러한 접근법의 주 목적은 관계의 의미를 보존하면서 null 이 되도록 적게 나오게 하는 것임

- 외래키 값이 null 이라는 것은 '해당 투플은 다른 테이블의 투플과 관계가 없다' 의 의미

02 외래키 제약조건

2 1:1 사상에 있어서의 외래키

1) 외래키 접근법

- 둘 중 하나만 전체 참여일 때
  전체 참여로 참여하는 테이블에 외래키를 사상
  - 전체참여의 의미를 반영하기 때문에
    외래키 값이 null 이 되는 경우가
    발생하지 않음

02 외래키 제약조건

2 1:1 사상에 있어서의 외래키

2) 통합 접근법

- 두 개 엔티티가 모두 전체 참여일 때
  하나의 테이블로 합침, 어떤 테이블로 합칠지는
  설계자의 선택 사항
  - 외래키가 존재하지 않으므로
    외래키 값이 null 이 되는 경우도 발생하지 않음

02 외래키 제약조건

2 1:1 사상에 있어서의 외래키

3) 상호참조 관계(Cross-reference relation) 접근법

- 엔티티가 모두 부분 참여일 때 새로운 상호참조 테이블을 생성한 후, 각 테이블의 주키를 외래키로 빌려오고, 빌려온 외래키들을 합쳐서 주키로 만듦
  - 엔티티가 모두 부분 참여이긴 하지만 상호참조 테이블에서는 실제 관계가 있는 경우에만 값을 갖기 때문에 외래키 값이 null 이 되는 경우가 발생하지 않음

02 외래키 제약조건

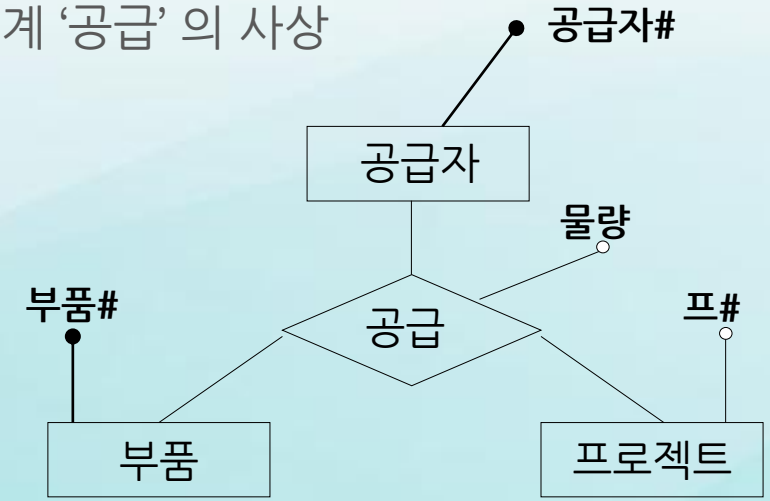
3 n-ary 관계타입의 사상

- n진 관계는 별도의 테이블(보통 관계 테이블이라고 함)을 생성하고
- 관계에 참여하는 n개 테이블의 주키를 빌려와 외래키로 구성함, 이때 각 외래키가 해당 주키를 참조하도록 함
- n개의 외래키가 관계 테이블의 주키(복합키 형태)가 됨
- 관계 테이블의 속성은 이러한 외래키들과 관계 속성(관계에 속하는 속성)으로 구성됨

02 외래키 제약조건

3 n-ary 관계타입의 사상

예) 3진 관계 '공급' 의 사상



공급(<u>공급자#, 부품#, 프로젝트#</u>, 물량)
PK={공급자#, 부품#, 프로젝트#},
FK={공급자#}, {부품#}, {프로젝트#}