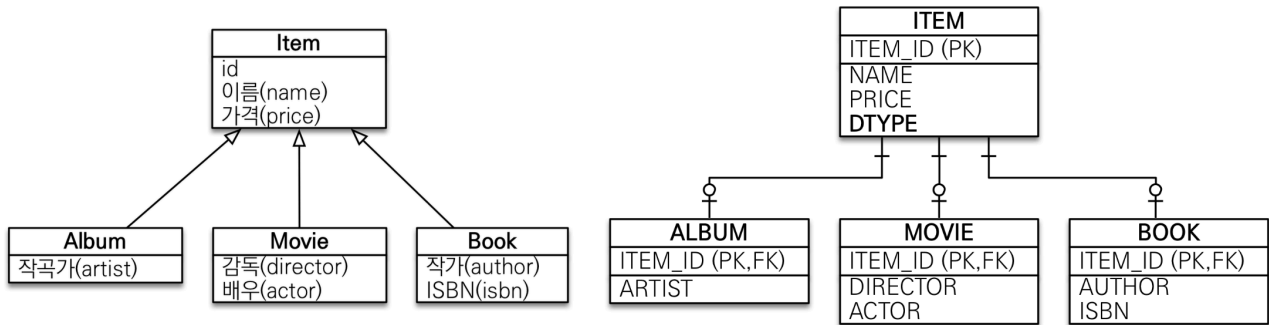


JPA 상속관계 매핑 & @MappedSuperclass

- 상속매핑
 - Mapped Superclass - 매핑 정보 상속
-
- 관계형 데이터베이스는 상속 관계가 없다
 - 슈퍼타입 서브타입 관계라는 모델링 기법이 객체 상속과 유사
 - 상속관계 매핑: 객체의 상속과 구조와 DB의 슈퍼타입 서브타입 관계를 매핑
-
- 각각 테이블로 변환 → 조인 전략
 - 통합 테이블로 변환 → 단일 테이블 전략
 - 서브타입 테이블로 변환 → 구현 클래스 마다 테이블 전략
- @Inheritance(strategy=**InheritanceType.XXX**)
 - **JOINED**
 - **SINGLE_TABLE**
 - **TABLE_PER_CLASS**
 - @DiscriminatorColumn(name="DTYPE") : 컬럼을 타입으로 가는 방법
 - @DiscriminatorValue("XXX") : 타입별 값을 설정 방법
-

조인 전략

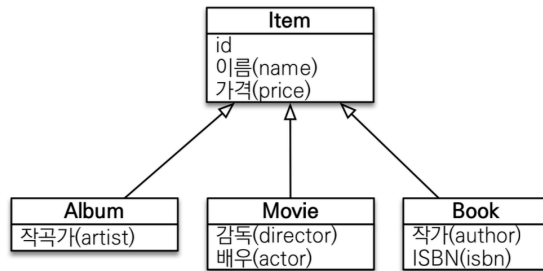
조인 전략



- 장점
 - 데이터 정규화
 - 외래 키 참조 무결성 제약조건 활용 가능
 - 저장 공간 효율화
- 단점
 - 조인 시 조회를 많이 사용하여 성능 저하
 - 조회 쿼리가 복잡함
 - 데이터 저장시 INSERT 2번 호출
- → 생각보다 성능이 아주 많이 저하되진 않는다.

단일 테이블 전략

단일 테이블 전략



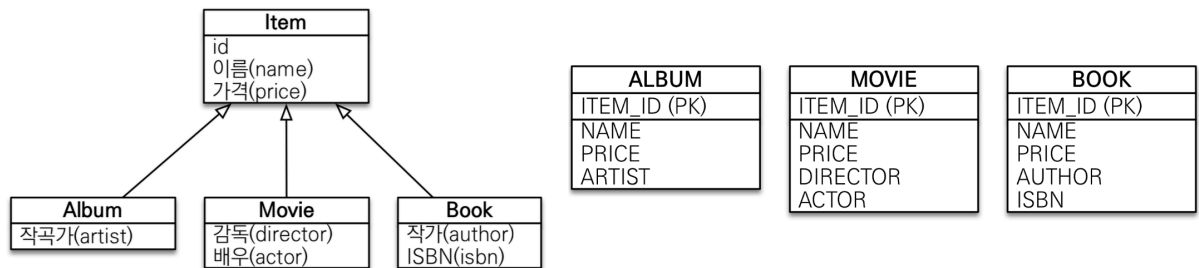
ITEM
ITEM_ID (PK)
NAME
PRICE
ARTIST
DIRECTOR
ACTOR
AUTHOR
ISBN
DTYPE

- 장점
 - 조인이 필요 없으므로 일반적으로 조회 성능이 빠름
 - 조회 쿼리가 단순함
- 단점
 - 자식 엔티티가 매핑한 컬럼은 모두 null 허용
 - 단일 테이블에 모든 것을 저장하므로 테이블이 커질 수 있다.
 - 상황에 따라서 조회 성능이 오히려 느려짐

구현 클래스 각각 구현 전략

- Item에 추상클래스 사용

구현 클래스마다 테이블 전략



- 이 전략은 데이터베이스 설계자와 ORM 전문가 둘 다 추천****X
- 장점
 - 서브 타입을 명확하게 구분해서 처리할 때 효과적
 - not null 제약조건 사용 가능
- 단점
 - 여러 자식 테이블을 함께 조회할 때 성능이 느림(UNION SQL 필요)
 - 자식 테이블을 통합해서 쿼리하기 어려움

단순하면 단일 테이블

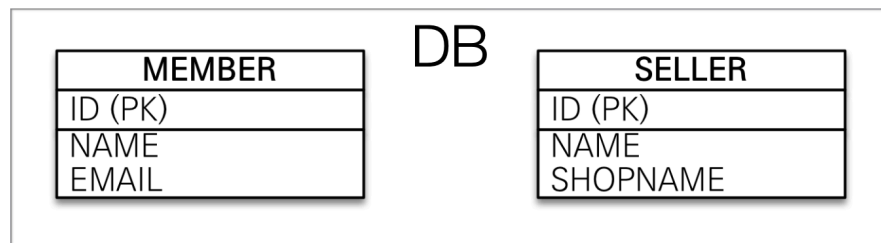
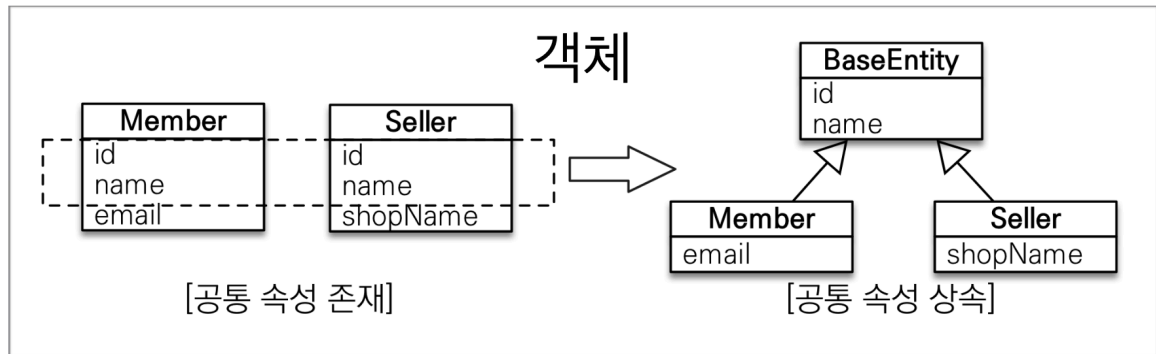
복잡하면 조인 테이블

@MappedSuperclass

- 공통 매핑 정보가 필요할 때 사용

@MappedSuperclass

- 공통 매핑 정보가 필요할 때 사용(id, name)



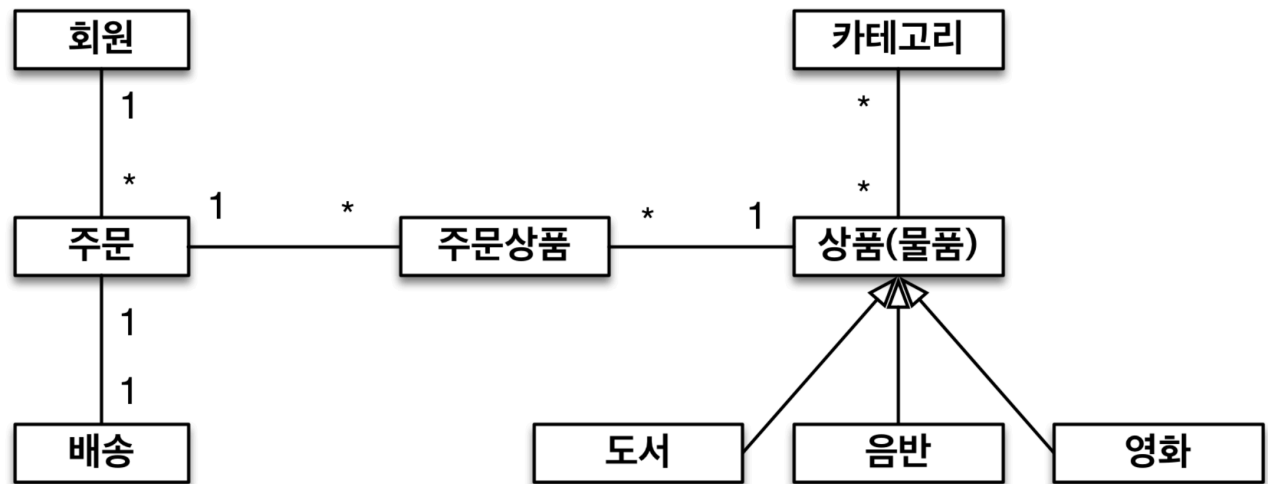
- 상속관계 매핑 아니다
- 엔티티가 아니고 테이블과 매핑되지 않는다.
- 부모 클래스를 상속 받는 자식 클래스에 매핑 정보만 제공
- 조회, 검색 불가(**em.find(BaseEntity)** 불가)
- 직접 생성해서 사용할 일이 없으므로 추상 클래스 권장
 - ex) @MappedSuperclass

```
public abstract class BaseEntity {}
```
 - ex)

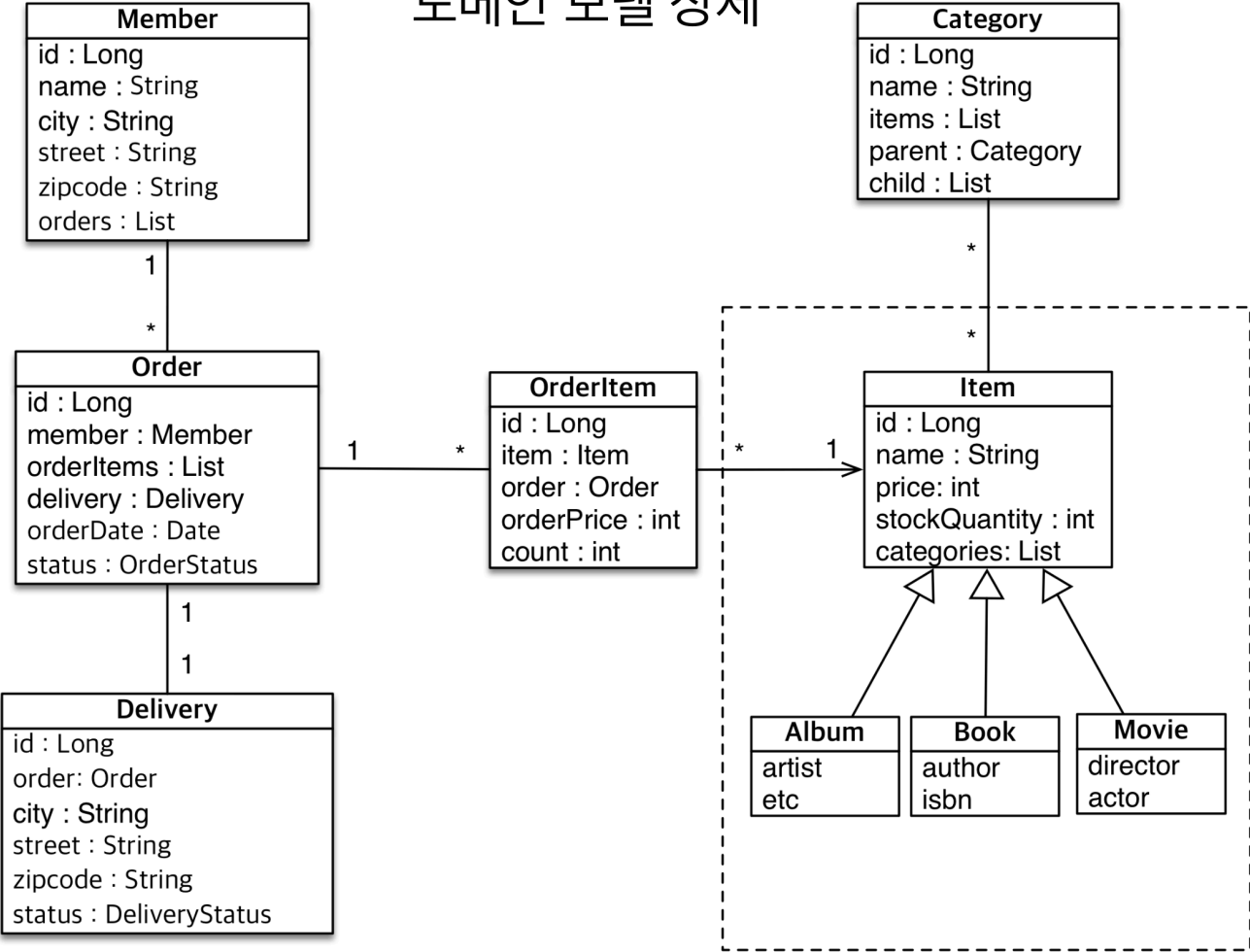
```
@Entity
@MappedSuperclass
public abstract class BaseEntity {}
```
- 테이블과 관계 없고, 단순히 엔티티가 공통으로 사용하는 매핑 정보를 모으는 역할
- 주로 등록일, 수정일, 등록자, 수정자 같은 전체 엔티티에서 공통으로 적용하는 정보를 모을 때 사용
- 참고: @Entity 클래스는 엔티티나 @MappedSuperclass로 지정한 클래스만 상속 가능

실습

도메인 모델



도메인 모델 상세



테이블 설계

