5. 실전 예제 1 - 요구사항 분석과 기본 매핑

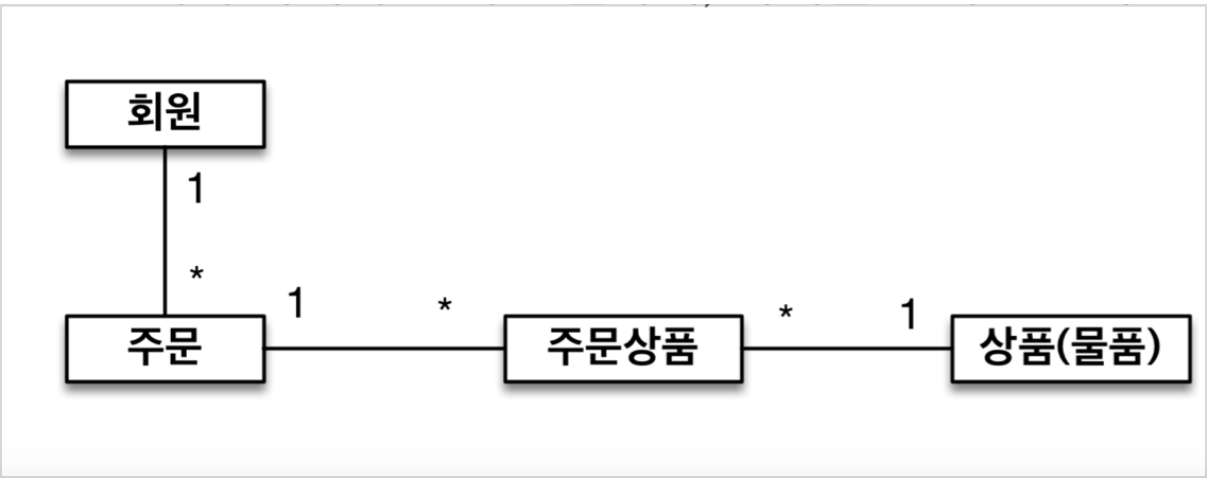
요구사항 분석

- 회원은 상품을 주문할 수 있다.
- 주문 시 여러 종류의 상품 선택 가능.

기능 목록

- 회원기능
- 회원등록
- 회원조회 * 상품기능
- 상품등록 * 상품수정
- 상품조회
- 주문기능
- 상품주문
- 주문내역조회
- 주문취소

도메인 모델 분석



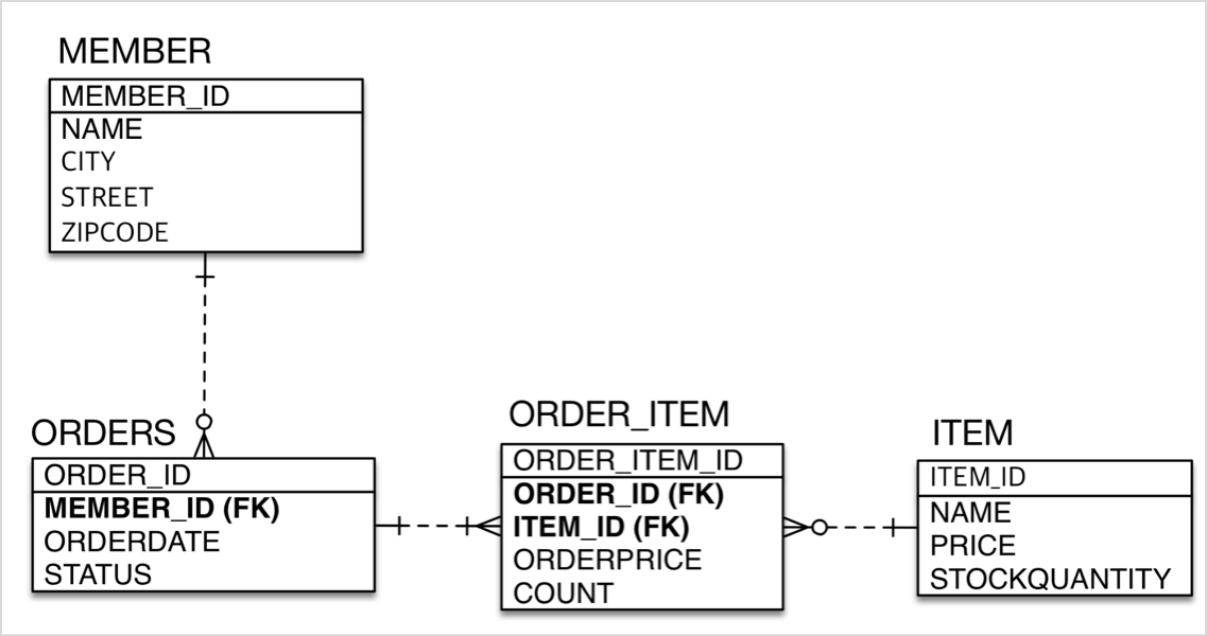
➜ 회원은 주문을 여러번 할 수 있다. 그러나 주문은 한 회원에게 종속된다

➜ 상품은 여러 번 주문 될 수 있다. 그러나 한 주문에서만 가능하다.

➜ 어떤 상품은 여러 번 주문되어 주문 상품이 될 수 있다. 그러나 주문 상품은 상품에 종속된다.

➜ 한 주문에서 여러가지 상품을 주문할 수 있기 때문에 다대다를 일대다 다대일로 풀어낸 것.

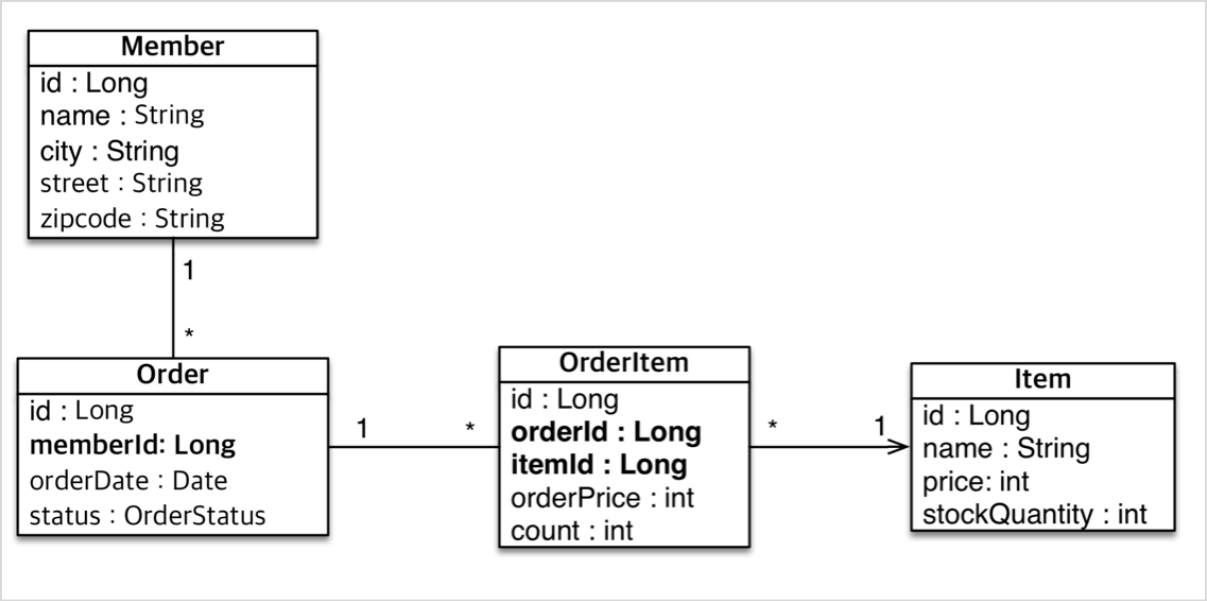
테이블 설계



➜ 맨 위에 따로 떼어져있는게 pk.

➜ ORDER_ITEM 잘 보자!

엔티티 설계와 매핑



➜ 위 엔티티 설계를 매핑해보자.

➜ 먼저 도메인 패키지를 만들고 거기에 위 속성들을 모두 넣어준다.

➜ SpringBoot에 들어가면 관례를 overriding 가능. springboot에서는 카멜케이스를 언더스코어로 바꿔줄수있다

➜ @Column(name = “”)과 같이 적어주는 이유

➜ 멤버변수명과 테이블 설계에서의 컬럼명이 다르면 적어줌

➜ @Table(name = “ORDERS”) : 굳이 Order의 테이블 명을 적어준 이유

➜ order가 예약어로 잡힌 db도 있기 때문

➜ Order의 memberId가 좀 이상하다.

➜ 이걸 그대로 쓴다고 가정하자.

➜ 만약 주문한 사람의 id로 주문 내역을 찾는다면 일단 주문 내역의 id로 주문을 찾고, 그 다음 또 order.getMemberId()를 통해 주문한 사람의 id를 찾아야 한다.

➜ 그런데 이것은 객체지향적이지 않다. 따라서 Order 안에도 Member를 집어넣어주는 것이 맞다고 생각이 든다. 그렇다면 getter로 멤버를 불러온 후 멤버의 id를 찾을 수 있을 것.

➜ 위 항목과 같은 설계가 데이터 중심의 설계이다. 이것의 문제점은 강의록에..

#JPA