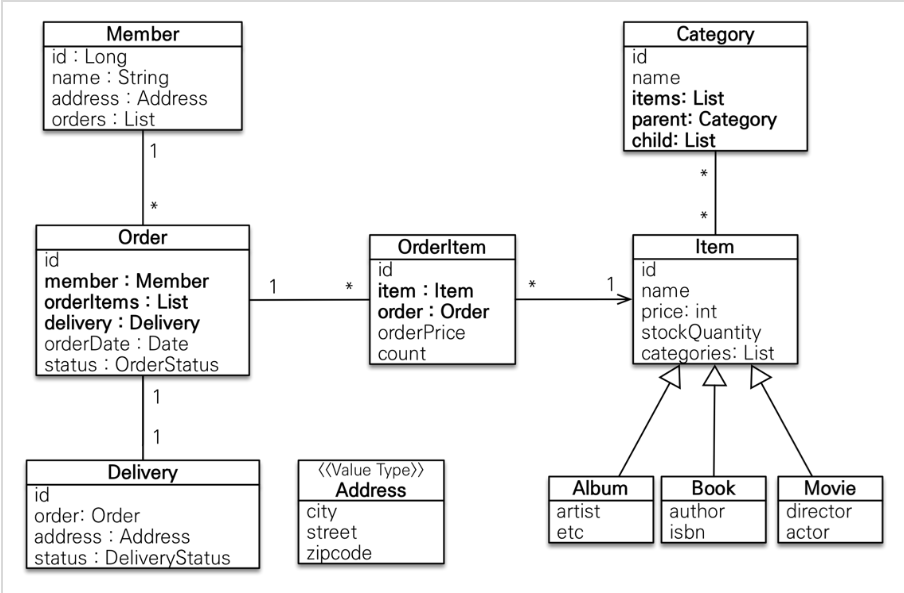
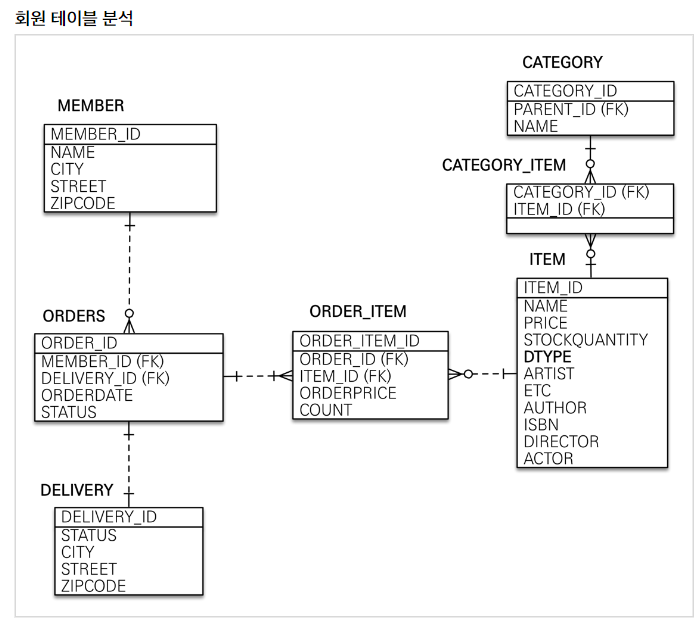


회원, 주문, 상품의 관계: 회원은 여러 상품을 주문할수있다. 그리고 한 번 주문할 때 여러 상품을 선택할 수 있으므로 주문과 상품은 다대다관계다. 하지만 이런 다대다관계는 거의 사용하지 않는다. 따라서 그림처럼 주문상품이라는 엔티티를추가해서 다대다관계를 일대다, 다대일관계로 풀어냈다.

<회원 엔티티 분석>





참고: **실제코드에서는 DB에소문자 + \_(언더스코어) 스타일을 사용**->데이터베이스테이블명, 컬럼명에 대한 관례는 회사마다 다르다. 보통은 대문자 + \_(언더스코어)나소문자+ \_(언더스코어) 방식 중에 하나를 지정해서 일관성있게 사용한다. 강의에서 설명할 때는 객체와 차이를 나타내기 위해 데이터베이스테이블, 컬럼명은 대문자를 사용했지만, 실제 코드에서는 소문자 + \_(언더스코어) 스타일을 사용하겠다.

**엔티티설계 시 주의점**

\* **엔티티에는 가급적 Setter를 사용하지 말자** : Setter가 모두 열려있다. 변경포인트가 너무 많아서, 유지보수가 어렵다. 나중에 리펙토링으로 Setter 제거

\* **모든 연관관계는 지연로딩으로 설정!** 즉시 로딩(EAGER)은 예측이 어렵고, 어떤 SQL이 실행될 지 추적하기 어렵다. 특히 JPQL을 실행할 때 N+1 문제가 자주 발생한다. 실무에서 모든 연관관계는 지연로딩(LAZY)으로 설정해야 한다. 연관된 엔티티를 함께 DB에서 조회해야 하면, fetch join 또는엔티티 그래프 기능을 사용한다. @XToOne(OneToOne, ManyToOne) 관계는 기본이 즉시 로딩이므로 직접 지연로딩으로 설정해야한다.

\* **컬렉션은 필드에서 초기화**하자. 컬렉션은 필드에서 바로 초기화하는 것이 안전하다.

왜냐하면...

null문제에서 안전하다. 하이버네이트는 엔티티를 영속화할 때, 컬랙션을 감싸서 하이버네이트가 제공하는 내장컬렉션으로 변경한다. 만약 getOrders()처럼 임의의 메서드에서 컬력션을 잘못 생성하면 하이버네이트 내부 메커니즘에 문제가 발생할 수 있다. 따라서 필드 레벨에서 생성하는 것이 가장 안전하고, 코드도 간결하다.

**테이블, 컬럼명 생성전략**

: 스프링부트에서 하이버네이트 기본 매핑 전략을 변경해서 실제 테이블필드명은 다름

[https://docs.spring.io/spring-boot/docs/2.1.3.RELEASE/reference/htmlsingle/#howto-configure-hibernate-naming-strategyhttp://docs.jboss.org/hibernate/orm/5.4/userguide/html\_single/Hibernate\_User\_Guide.html#naming](https://docs.spring.io/spring-boot/docs/2.1.3.RELEASE/reference/htmlsingle/#naming)

**하이버네이트기존구현: 엔티티의 필드명을 그대로 테이블의 컬럼명으로 사용(SpringPhysicalNamingStrategy)**