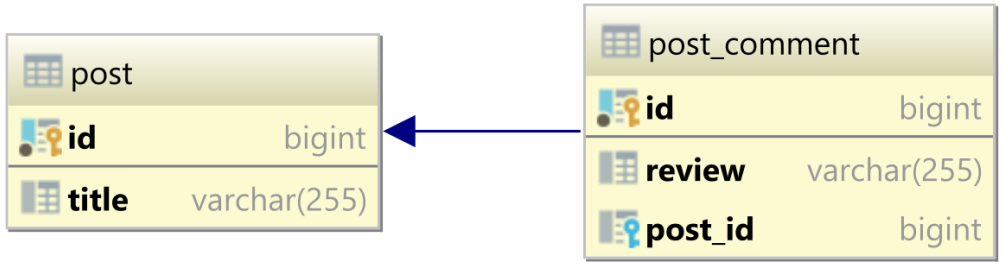
04 다대일 연관관계

연관관계

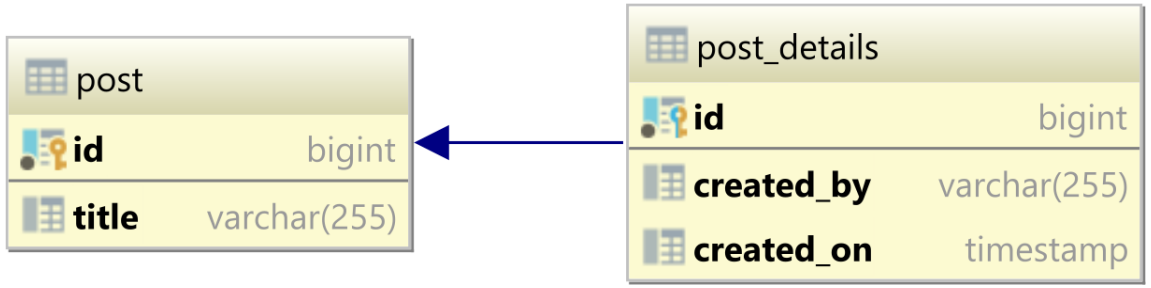
- **연관관계의 종류**
  - One-To-One(일대일)
  - One-To-Many(일대다)
  - Many-To-One(다대일)
  - Many-To-Many(다대다)
- **연관관계 주의 사항**
  - 테이블은 외래키 하나로 테이블 간 연관관계를 설정
  - 객체는 참조를 통해 연관관계를 설정
  - 객체의 연관관계는 방향성을 가짐

연관관계

- One-To-Many(Many-To-One)



- One-To-One



- Many-To-Many

다대일

- **다대일 양방향 연관관계**

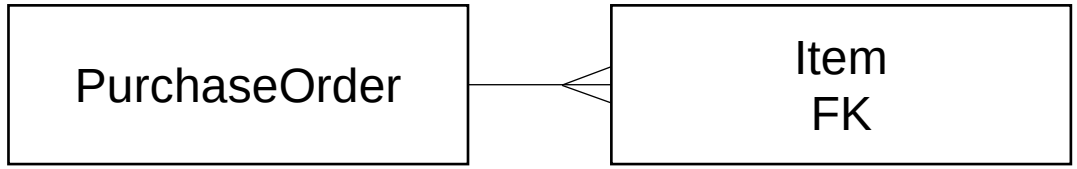
```java
@Entity
public class Item {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "ITEM_ID")
    private Long id;
    private String name;
    private int price;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name="PURCHASE_ORDER_ID")
    private PurchaseOrder order;
```

```java
@Entity
public class PurchaseOrder {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "PURCHASE_ORDER_ID")
    private Long id;

    private String userName;

    @OneToMany(mappedBy = "order")
    private List<Item> itmes = new ArrayList<Item>();
```



janssen.com/best-practices-many-one-one-many-associations-mappings/

다대일

- **어노테이션 설명**
  - @ManyToOne
    - 나의 엔티티를 기준으로 반대편 엔티티와의 연관성
    - 현재 Item은 "다"쪽이므로 다대일 연관관계를 위해 사용
  - @OneToMany
    - 나의 엔티티를 기준으로 반대편 엔티티와의 연관성
    - 현재 PurchaseOrder는 "일"쪽이므로 일대다 연관관계를 위해 사용
  - mappedBy 속성
    - 양방향 연관관계를 설정하는 경우, 연관관계의 ownership을 결정해야 함
    - ownership이란 외래키를 관리하는 주체가 누가 될 것이지를 정하는 것
    - RDB에서 외래키는 "다"쪽에 있으므로 ownership은 Item이 가짐
    - 따라서 "일" 쪽인 PurchaseOrder에서 연관관계의 매핑이 ~~에 의해 관리된다는 의미의 mappedBy 속성을 지정하고 구체적으로 "다"쪽의 order라는 필드에 매핑되었음을 명시

[출처] https://thorben-janssen.com/best-practices-many-one-one-many-associations-mappings/

다대일

- **다대일 양방향 연관관계**

```
Item item1 = new Item();
item1.setName("치킨");
Item item2 = new Item();
item2.setName("치즈볼");

PurchaseOrder order = new PurchaseOrder();
order.setUsername("kim");
order.getItmes().add(item1);
order.getItmes().add(item2);

//item1.setOrder(order);
//item2.setOrder(order);

em.persist(order);
em.persist(item1);
em.persist(item2);
```

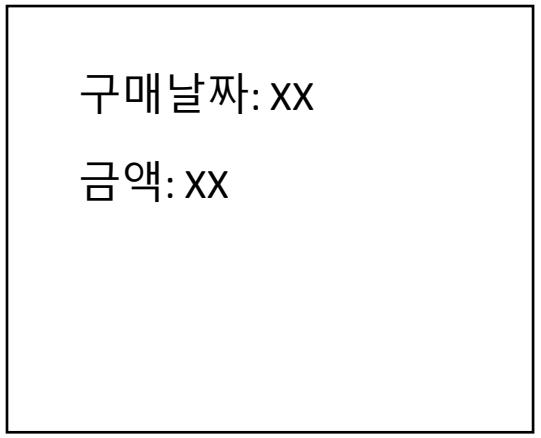
```
em.persist(item1);
em.persist(item2);
em.persist(order);
```

persist 순서를 변경하여 테스트

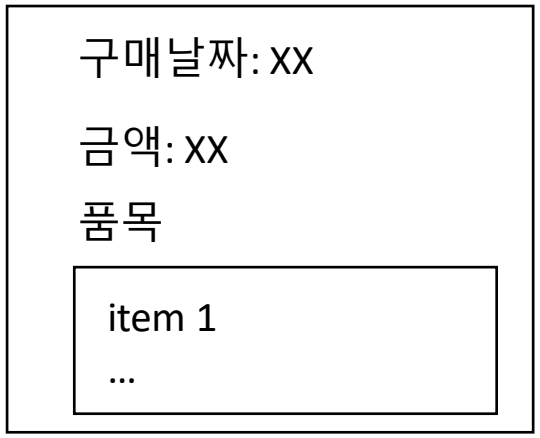
[출처] https://thorben-janssen.com/best-practices-many-one-one-many-associations-mappings/

다대일

- **어노테이션 설명**
  - @JoinColumn
    - 컬럼은 컬럼인데 FK로 사용되는 컬럼임을 명시
  - FetchType: 특정 엔티티를 조회할 때 연관된 엔티티도 함께 조회할지를 결정
    - FetchType.LAZY: 연관된 엔티티는 가져오지 않음
    - FetchType.EAGER: 내가 명시하지 않아도 연관된 엔티티도 함께 조회



구매내역



구매내역

[출처] https://thorben-janssen.com/best-practices-many-one-one-many-associations-mappings/

Best Practice

- **Don’t use unidirectional one-to-many associations**
  - 일대다 단방향으로만 연관관계를 매핑하면 안됨
  - 외래키가 없는 "일" 쪽에서 외래키가 있는 "다" 쪽을 관리하는 구조
  - Hibernate는 예상치 못한 테이블(association table)을 생성하며 더 많은 SQL이 발생함

```
insert
into
    PurchaseOrder_Item
    (PurchaseOrder_id, items_id)
values
    (?, ?)
```

[출처] https://thorben-janssen.com/best-practices-many-one-one-many-associations-mappings/

Best Practice

- **Don’t use unidirectional one-to-many associations**
  - 외래키 컬럼에 *@JoinColumn* 을 명시

```
..
..
update
        Item
    set
        fk_order=?
    where
        id=?
```

  - 그러나 SQL UPDATE문이 추가적으로 실행되는 문제가 있음

[출처] https://thorben-janssen.com/best-practices-many-one-one-many-associations-mappings/

Best Practice

- **Avoid the mapping of huge to-many associations**
  - OneToMany, ManyToMany에서 "다"쪽이 너무 많은 연관관계의 경우 연관관계 매핑을 지양
  - 예) 사용자 로그
  - "다"쪽을 전부 메모리에 올리는 것은 위험하며 시간도 많이 걸림
  - 이러한 경우, 다대일 단방향 연관관계를 맺음
  - 연관 엔티티를 읽을 경우 JPQL로(직접 연관 엔티티를 찾는 SQL을 실행) 끊어 읽는 것이 (pagination) 좋음
    - divide and conquer

[출처] https://thorben-janssen.com/best-practices-many-one-one-many-associations-mappings/

Best Practice

- **Define FetchType.LAZY for @ManyToOne association**
  - JPA는 to-one 관계에서 FetchType의 기본값을 EAGER로 설정
  - 하나의 엔티티를 조회할 때 연관된 엔티티들이 함께 조회되는 것은 큰 문제가 아니지만 여러 엔티티를 한 번에 조회할 때 모든 연관된 엔티티를 조회하는 것은 다른 문제
    - 사용자 A를 조회하면서 A의 구매내역을 함께 조회하는 것과 1000명의 사용자를 조회할 때 모든 사용자의 구매 내역을 조회하는 것은 다른 문제
  - **EAGER로 할 경우, JPQL을 사용할 때 N+1문제가 발생**

[출처] https://thorben-janssen.com/best-practices-many-one-one-many-associations-mappings/

Best Practice

- **N+1문제**
  - 쿼리 1개를 날리면서 추가적으로 N개의 쿼리가 나감
  - 엔티티의 상태를 데이터베이스에 적용하는 시점은 트랜잭션이 종료되는 시점
  - 그러나 JPQL은 실행 즉시 쿼리가 실행
  - DB 관점에서는 실행된 쿼리가 JPA를 통해 전달되었는지, 엔티티가 서로 어떤 연관관계와 fetch 타입을 가졌는지 알지 못함(관심도 없음)
  - 일단 쿼리로 특정 엔티티만 조회했는데 JPA가 결과를 받아서 해석하니 " 어 이거 EAGER로 설정된 엔티티가 있네?"라고 해석하여 뒤늦게 추가 쿼리가 나감(+N)
  - JPQL에서는 패치조인으로 해결할 수 있음

[출처] https://thorben-janssen.com/best-practices-many-one-one-many-associations-mappings/

Best Practice

- **Implement helper methods to update bi-directional associations**
  - 양방향 연관관계 설정은 객체 탐색에서 편리한 기능이지만 주의해야 함
  - 한 쪽의 연관관계 상태가 변경되면 다른 쪽도 이를 반영해야 함

```
@Entity
public class PurchaseOrder {

    ...

    public void addItem(Item item) {
        this.items.add(item);
        item.setOrder(this);
    }
}
```

```
Item item3 = new Item();
item3.setName("Third Item");
order.addItem(item3);
em.persist(item3);
```

[출처] https://thorben-janssen.com/best-practices-many-one-one-many-associations-mappings/

Best Practice

- **정리**
  - 일대다 단방향 연관관계를 사용하지 마라
  - to-many 연관관계에서 "다" 쪽이 너무 많으면 연관관계 매핑을 지양해라
  - 양방향 연관관계에서는 helper methods를 사용하라
  - EAGER보다는 LAZY로 연관관계를 설정하라
  - CascadeType.Remove 사용을 심사숙고하라
  - (논리적으로)부모자식 관계에 있는 모델은 orphanRemoval을 사용해라
    - 부모는 일, 자식은 다: 예) 게시판과 게시글

[출처] https://thorben-janssen.com/best-practices-many-one-one-many-associations-mappings/