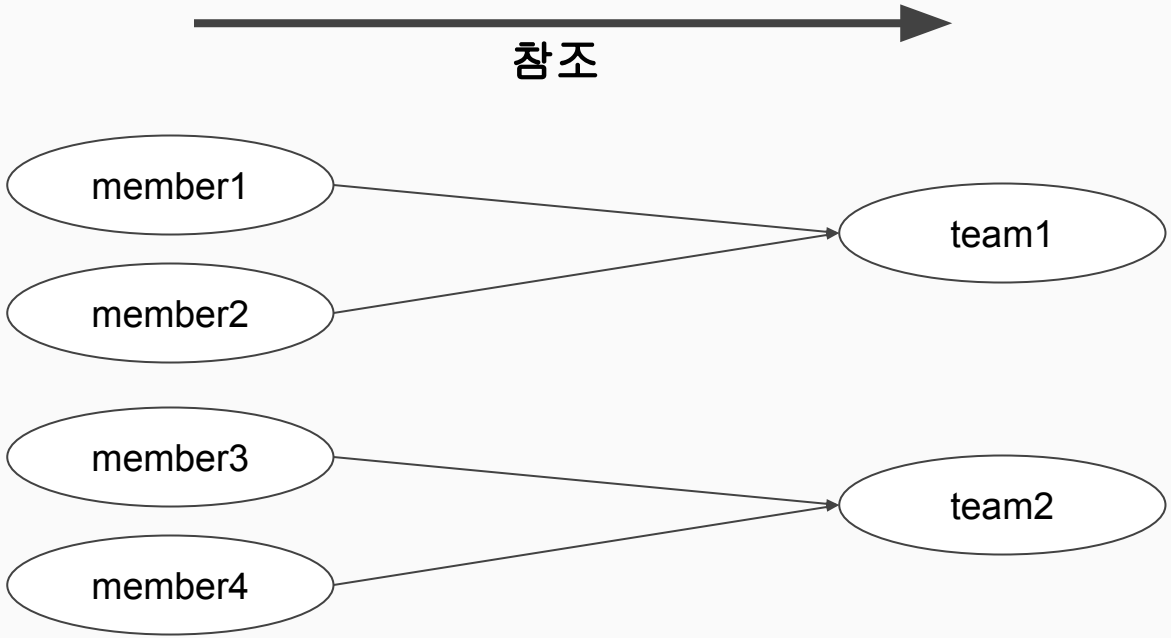
Chap05. 연관관계 매핑 기초

-꼬막조림

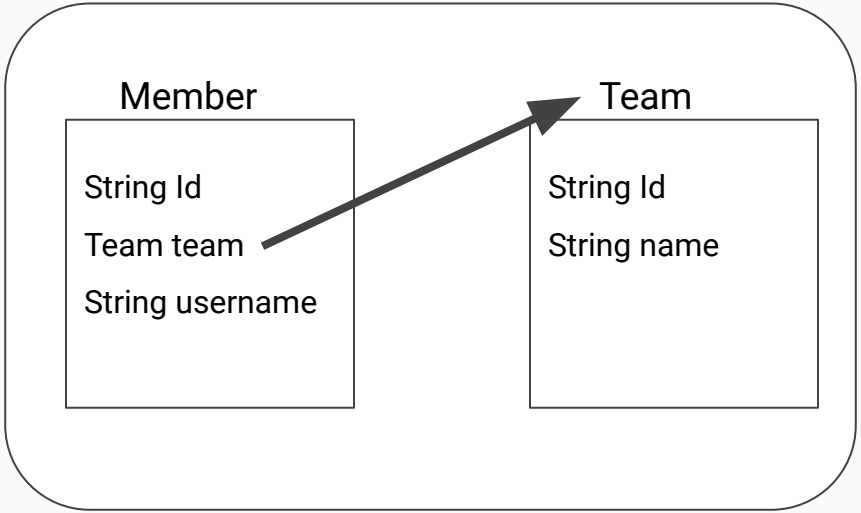
1. 단방향 연관관계

1. 단방향 연관관계

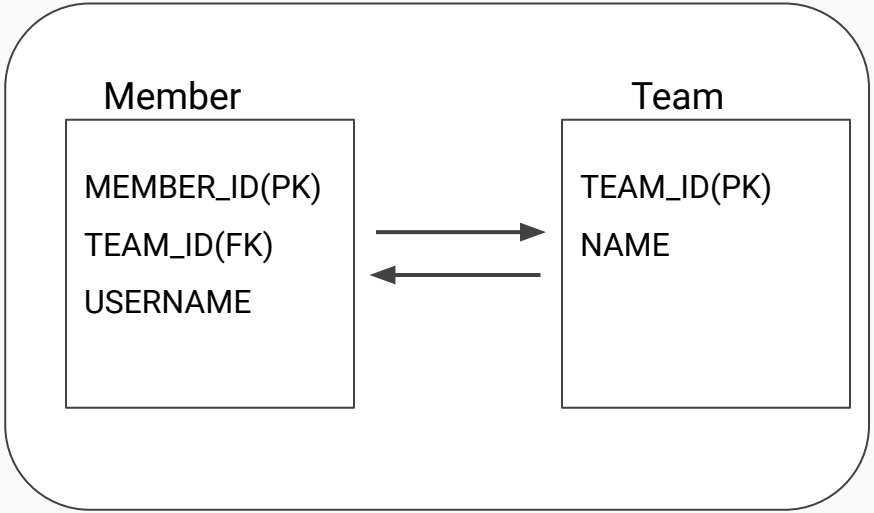


1. 단방향 연관관계

Object



Relational



1. 단방향 연관관계

MEMBER

```
@Id
@Column(name = “MEMBER_ID”)
private String id;

private String username;

@ManyToOne
@JoinColumn(name=”TEAM_ID”)
private Team team;
```

TEAM

```
@Id
@Column(name = “TEAM_ID”)
private String id;

private String name;
```

1. 단방향 연관관계

저장

```
Team team1 = new Team("team1", "팀1");
em.persist(team1);

Member member1 = new Member("member1", "회원1");
member1.setTeam(team1);
em.persist(member1);
```

조회

```
Team team = member1.getTeam();
```

수정

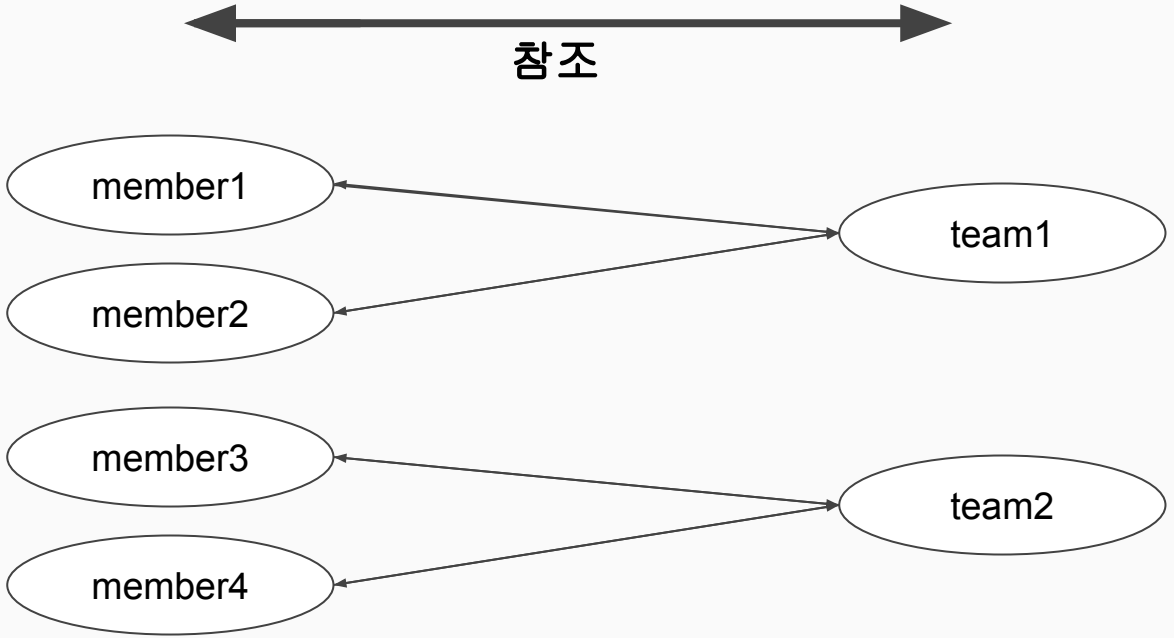
```
Team team2 = new Team("team2", "팀2");
em.persist(team2);

member1.setTeam(team2);
```

제거

```
member1.setTeam(null);
```

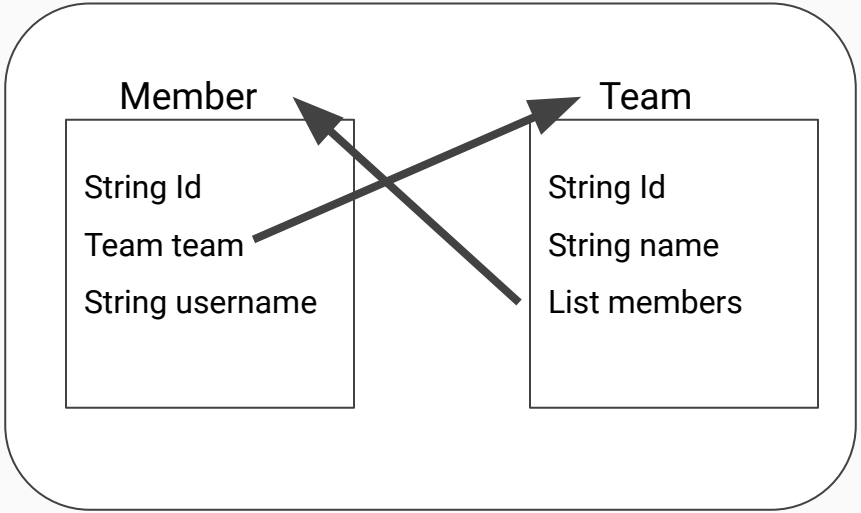
2. 양방향 연관관계

2. 양방향 연관관계

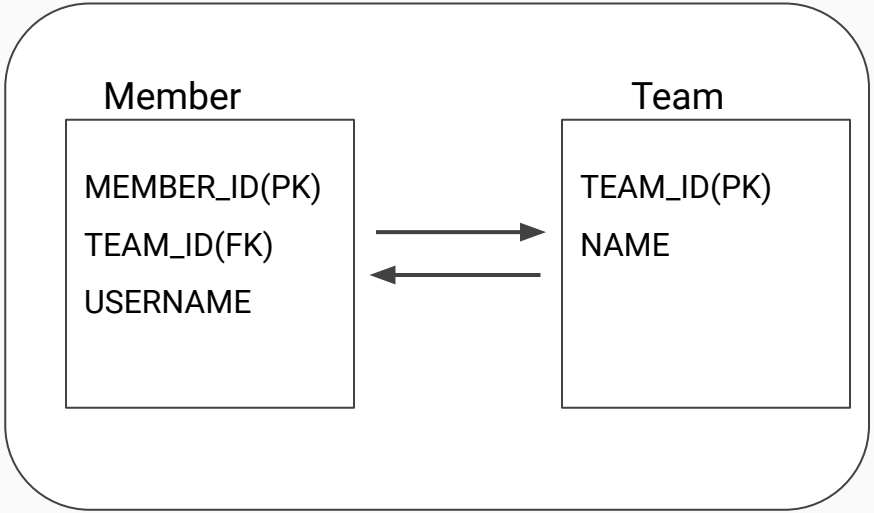


2. 양방향 연관관계

Object



Relational



2. 양방향 연관관계

MEMBER

```
@Id
@Column(name = “MEMBER_ID”)
private String id;

private String username;

@ManyToOne
@JoinColumn(name=”TEAM_ID”)
private Team team;
```

TEAM

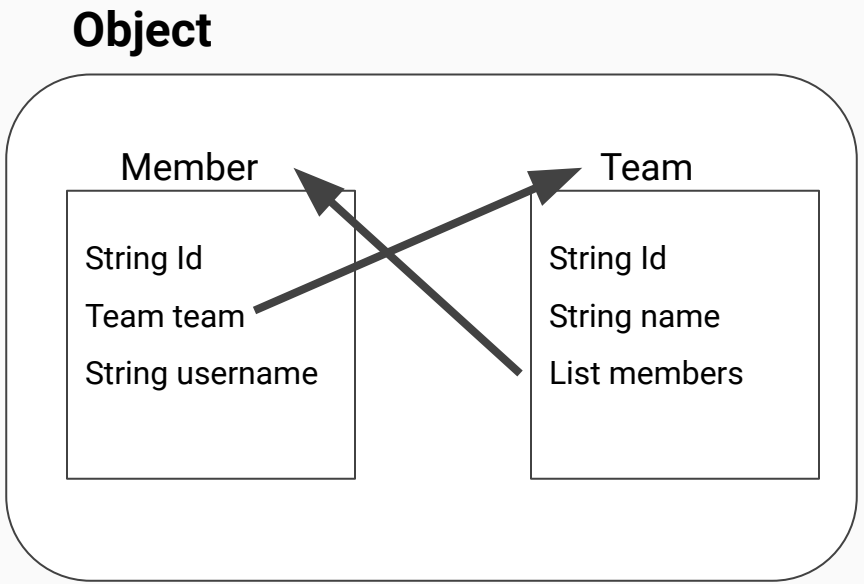
```
@Id
@Column(name = “TEAM_ID”)
private String id;

private String name;

@OneToMany(mappedBy = “team”)
private List<Member> members =
        new ArrayList<Member>();
```

2. 양방향 연관관계

연관관계의 주인

엔티티를 양방향 연관관계로 설정하면 객체의 참조는 둘인데 외래 키는 하나다. 따라서 둘 사이에 차이가 발생한다.

JPA는 두 객체 연관관계중 하나를 정해서 테이블의 외래키를 관리해야 하는데 이것을 연관관계의 주인이라한다.



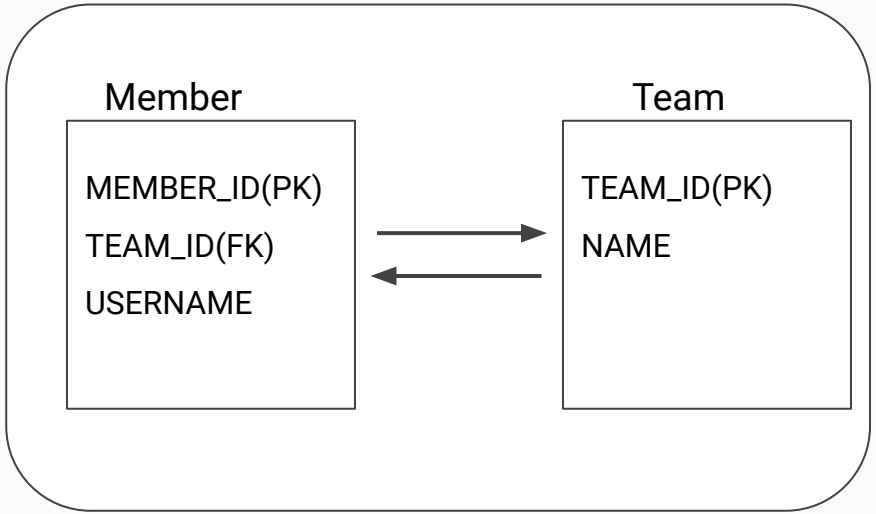
2. 양방향 연관관계

연관관계의 주인

연관관계의 주인만이 데이터베이스 연관관계와 매핑되고 외래 키를 관리(등록, 수정, 삭제)할 수 있다. 반면에 주인이 아닌 쪽은 읽기만 할 수 있다.

연관관계의 주인은 외래 키가 있는 곳으로 정한다.

Relational



2. 양방향 연관관계

조회

```
List<Member> members = team1.getMembers();
```

저장

```
Team team1 = new Team("team1", "팀1");
em.persist(team1);

Member member1 = new Member("member1", "회원1");
member1.setTeam(team1);
team1.getMembers().add(member1);
em.persist(member1);
```

JPA만 고려했을때 주인관계가 아닌 team 쪽에 members정보를 설정해줄 필요가 없음.
하지만 JPA를 쓰지 않는 순수 객체 상태에서 문제 발생야기.

2. 양방향 연관관계

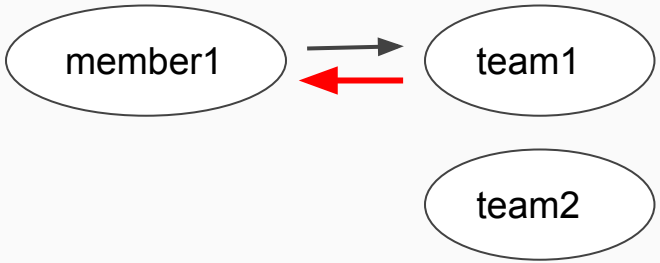
리팩토링 및 버그수정

비즈니스 로직

```
member.setTeam(team);
team.getMembers().add(member);
```

Member 클래스

```
public void setTeam(Team team) {
        If (this.team != null) {
                this.team.getMembers().remove(this);
        }
        This.team = team;
        team.getMembers().add(this);
}
```



감사합니다.