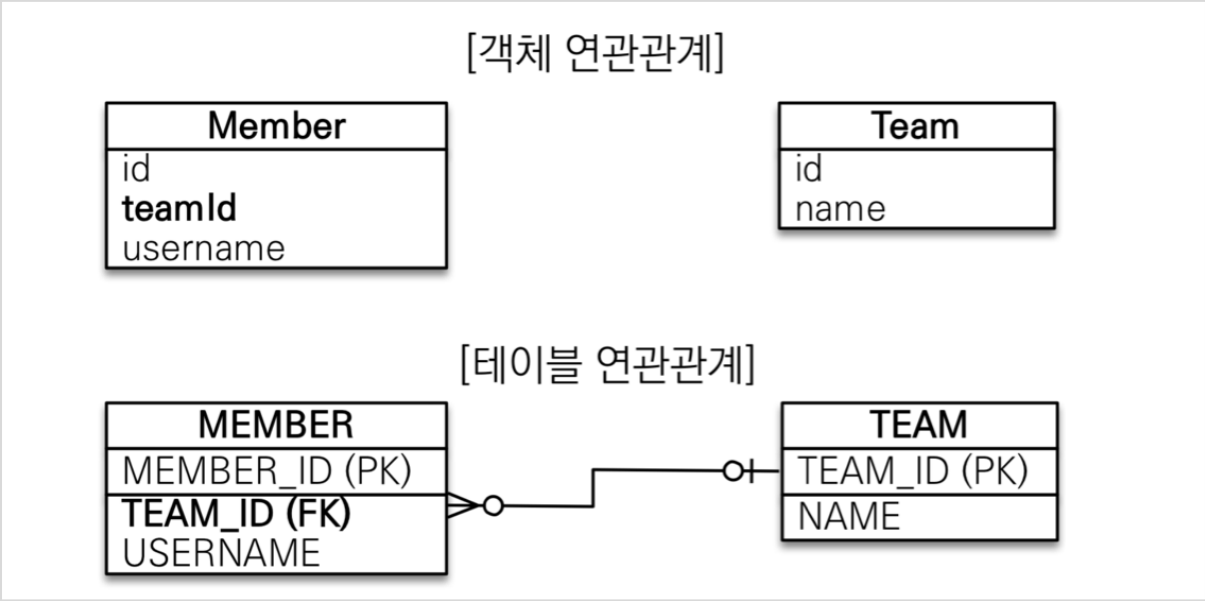
6-0. 연관관계 매핑 기초

연관관계를 맺어서 어떻게 설계하나? = 좀 더 객체지향답게 설계하자!

객체와 테이블 연관관계의 차이를 이해

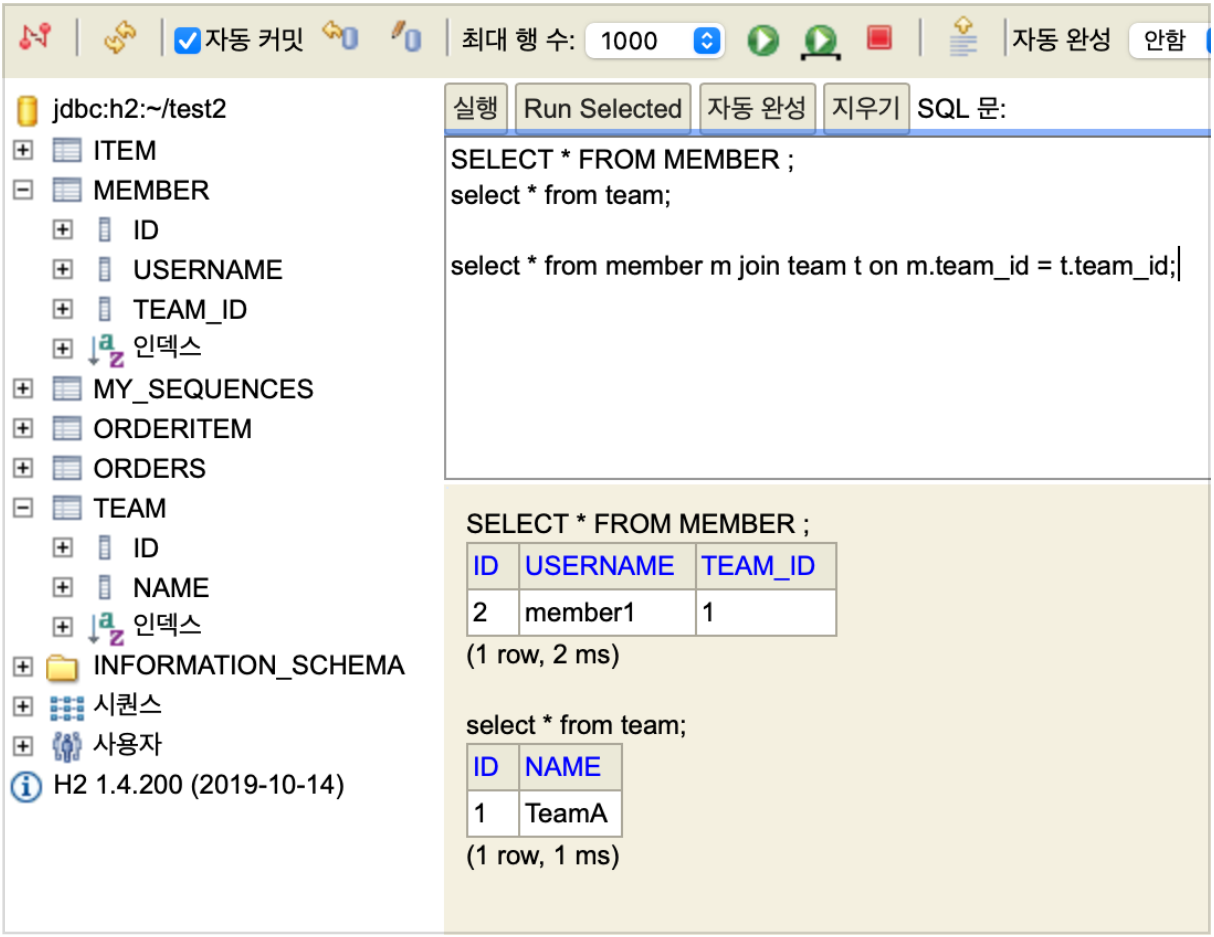
객체 : 레퍼런스 - 테이블 : 외래키 매핑
객체는 참조를 이용해 쭉쭉 따라갈 수 있는데(order.getMember().getName())
테이블에서는 연관관계를 외래키를 이용.

연관관계가 필요한 이유



외래키가 멤버에 있음. 왜나면 팀과 멤버의 관계가 일대다 이기 때문.
➜ 연관관계가 없는 객체로 짠다면? [여기로!] (연관관계 없음)
➜ 연관관계가 있는 객체로 짠다면? [여기로!] (연관관계 있음)

연관관계 없음



조인문처럼 조인해야 갖고올 수 있음. 외래키 사용!

MEMBER의 ID가 2인 이유 : @GeneratedValue에서 Sequence를 끌어왔기 때문에

```
package hellojpa;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;

public class JpaMain {
  public static void main(String[] args) {
    EntityManagerFactory emf = Persistence.createEntityManagerFactory("hello");

    EntityManager em = emf.createEntityManager();

    EntityTransaction tx = em.getTransaction();
    tx.begin();
    try {
```

```
        Team team = new Team();
        team.setName("TeamA");
        em.persist(team); // 영속상태가 되면 무조건 pk값이 세팅되고 영속상태가 됨.

        Member member = new Member();
        member.setName("member1");

        member.setTeamId(team.getId());
        em.persist(member);

        Member findMember = em.find(Member.class, member.getId());

        Long findTeamId = findMember.getTeamId();
        Team findTeam = em.find(Team.class, findTeamId);

        tx.commit();
    } catch (Exception e) {
        tx.rollback();
    } finally {
        em.close();
    }

    emf.close();

  }

}
```

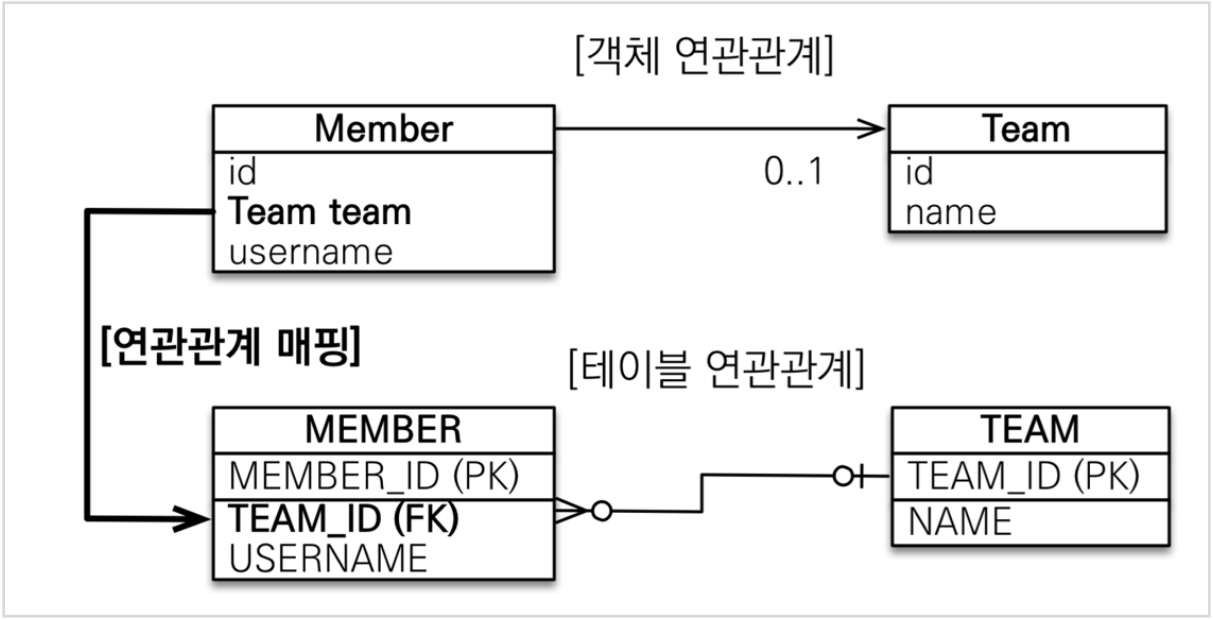
➜ db를 통해서 계속 물어봐야함!!! 객체지향적이지 않다...

➜ 객체를 테이블에 맞춰서 데이터 중심으로 모델링하면 협력관계 절대 못만듦

➜ 테이블은 외래키로 조인 / 객체는 참조를 사용!

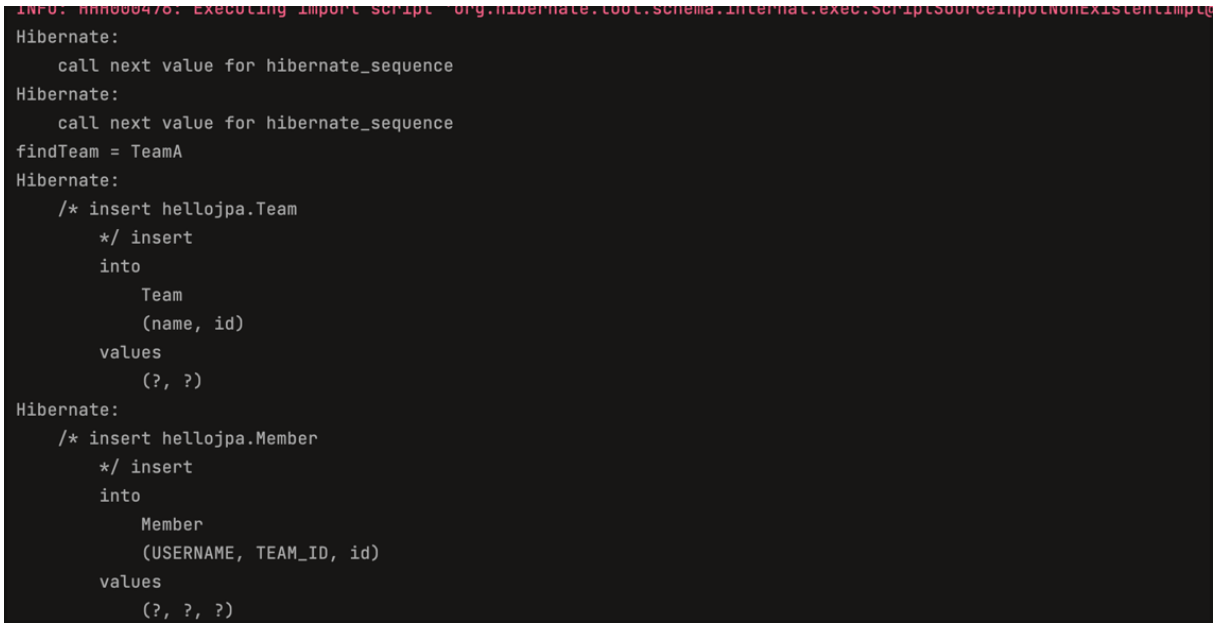
연관관계 있음

- 단방향 연관관계




```java
package hellojpa;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;

public class JpaMain {
  public static void main(String[] args) {
    EntityManagerFactory emf = Persistence.createEntityManagerFactory("hello");

    EntityManager em = emf.createEntityManager();

    EntityTransaction tx = em.getTransaction();
    tx.begin();
    try {

      Team team = new Team();
      team.setName("TeamA");
      em.persist(team); // 영속상태가 되면 무조건 pk값이 세팅되고 영속상태가 됨.

      Member member = new Member();
      member.setName("member1");
      member.setTeam(team);
      em.persist(member);
```

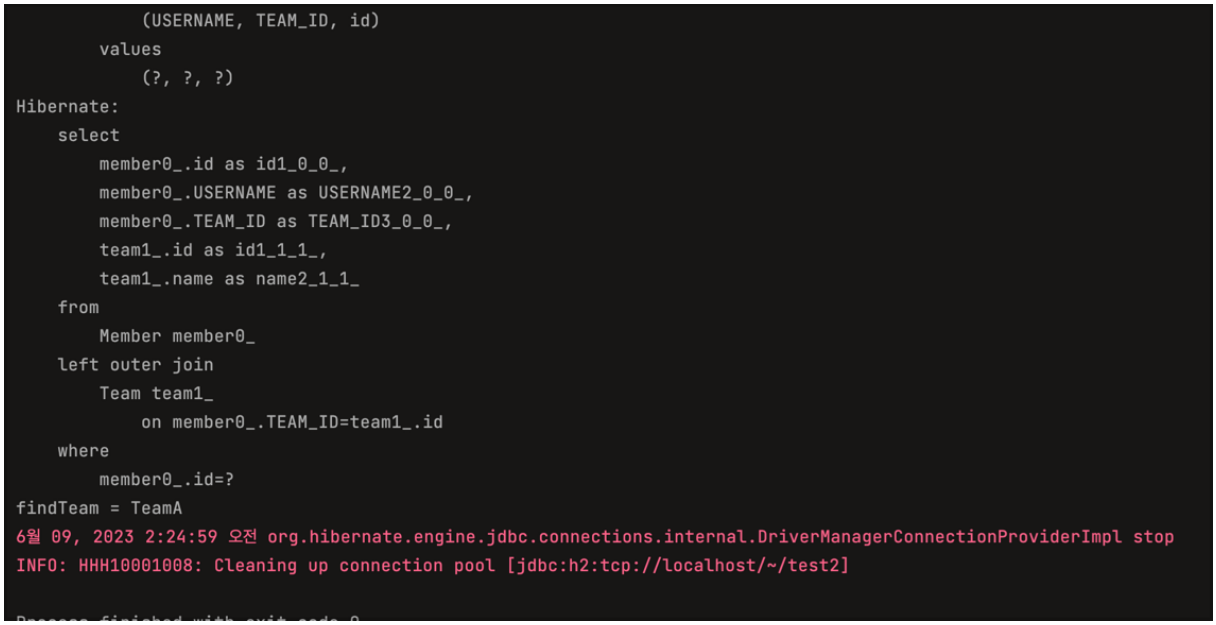
```
        Member findMember = em.find(Member.class, member.getId());

        Team findTeam = findMember.getTeam();
        System.out.println("findTeam = " + findTeam.getName());

        tx.commit();
    } catch (Exception e) {
        tx.rollback();
    } finally {
        em.close();
    }

    emf.close();

  }

}
```



`em.find`시에 쿼리가 안나가는 이유는 persist로 이미 영속화된 객체들에 대해서는 1차캐시에서 값을 찾을

수 있기 때문이다.

만약 db에서 찾는 걸 보고싶다면 `em.find` 이후에 `em.flush(); em.clear();` 를 입력해보자.



```
            (USERNAME, TEAM_ID, id)
        values
            (?, ?, ?)
Hibernate:
    select
        member0_.id as id1_0_0_,
        member0_.USERNAME as USERNAME2_0_0_,
        member0_.TEAM_ID as TEAM_ID3_0_0_,
        team1_.id as id1_1_1_,
        team1_.name as name2_1_1_
    from
        Member member0_
    left outer join
        Team team1_
            on member0_.TEAM_ID=team1_.id
    where
        member0_.id=?
findTeam = TeamA
6월 09, 2023 2:24:59 오전 org.hibernate.engine.jdbc.connections.internal.DriverManagerConnectionProviderImpl stop
INFO: HHH10001008: Cleaning up connection pool [jdbc:h2:tcp://localhost/~/test2]
```

찾은 멤버의 팀을 바꿔주고싶다면, 100L ID를 가지는 팀이 db에 있다고 가정하고 아래 코드를 입력해보자.

```
Team newTeam = em.find(Team.class, 100L);
findMember.setTeam(newTeam); // 멤버에게 새로운 팀 세팅해주기
```

이러면 찾은 멤버의 팀을 새로운 팀으로 바꾸어줄 수 있다. update 쿼리가 자동으로 발생함.

#JPA