1. JPA 소개

SQL 중심으로 개발하면 문제점
➜ CRUD(데이터 등록, 조회, 수정, 삭제) 무한 반복, 지루한 코드
관계형 DB를 쓰니까 SQL에 대한 의존도 높아짐

객체 ➜ sql 변환 ➜ sql+관계형 DB
sql변환하는게 매퍼 (개발자가 매퍼를 해야함)

상속

상속관계를 디비에 저장하기가 매우 어려움.
보통 이걸 객체 분해 - 슈퍼타입 서브타입으로 저장해서 항상 조인을 하는 식으로..
pk fk 형식으로?
서브타입 테이블을 조회하려면? 슈퍼타입과 조인하고 어쩌구….

=> 객체 세상으로 자바 컬렉션에 저장한다고 생각하면? list.add(subtype)으로 저장하면 끝!
& 조회한다고 생각하면? getter로 조회하면 끝. 부모 타입으로 조회 후 다형성을 활용할 수 있음.

연관관계

객체는 참조를 사용함. getter를 쓰기때문에
테이블은 외래 키를 사용함.
=> 객체를 테이블에 맞추어 모델링한다. 하지만 이것은 객체다운 모델링은 아님! 객체는 연관관계를 참조로 맺기 때문에..
=> 원래는 다른 테이블 객체의 기본키를 외래키로 가져와서 썼지만 그냥 테이블 객체의 참조를 가져옴
=> 자바 컬렉션으로 관리한다면? 그냥 getter로 다른 테이블 자체를 가져올 수 있다.
ex) Member mebmer = list.get(memberId); Team team = member.getTeam();

객체 그래프 탐색

객체는 자유롭게 객체 그래프를 탐색할 수 있어야 한다. 어떻게? member.team.teamname…..

sql은 처음에 작성하는 쿼리에 따라 탐색 범위가 제한된다!
➜ 엔티티 신뢰에 대한 문제. 계층형 아키텍쳐는 다음 계층을 믿고 쓸 수 있어야 하는데 이게 안됨..
=> 진정한 의미의 계층 분할이 어려워짐. 물리적으로는 분리가 되어있는데 논리적으로는 분리가 어렵다.

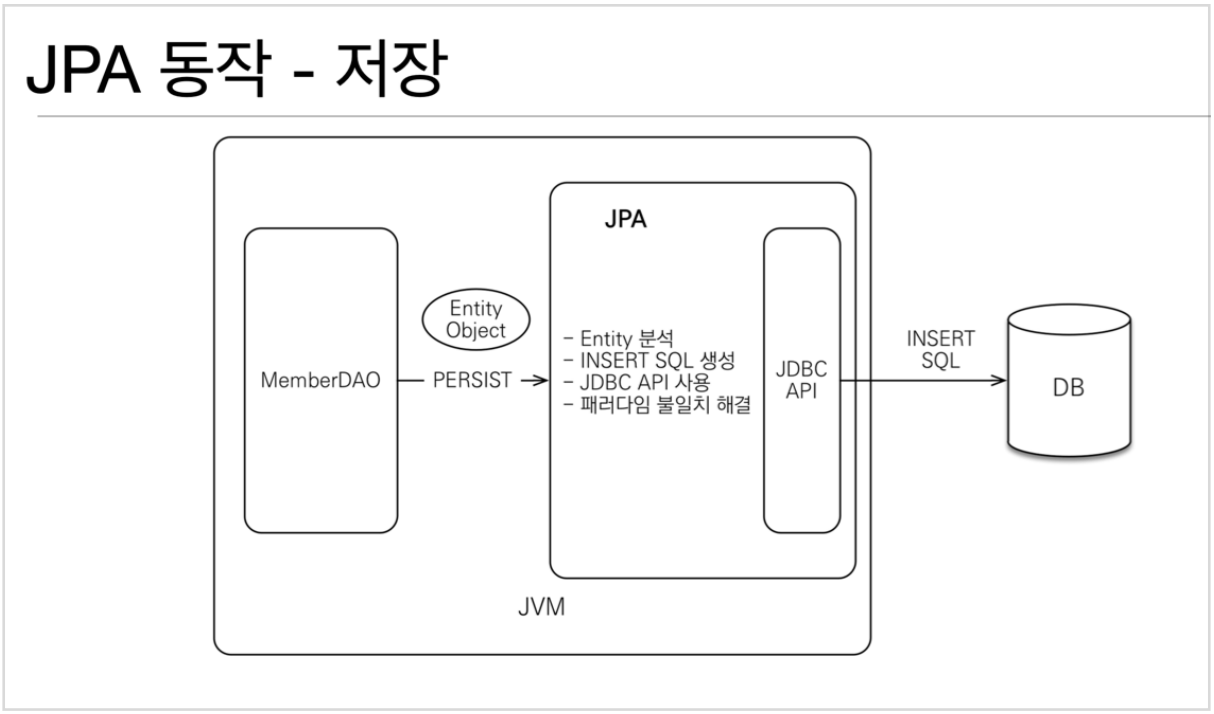
비교하기

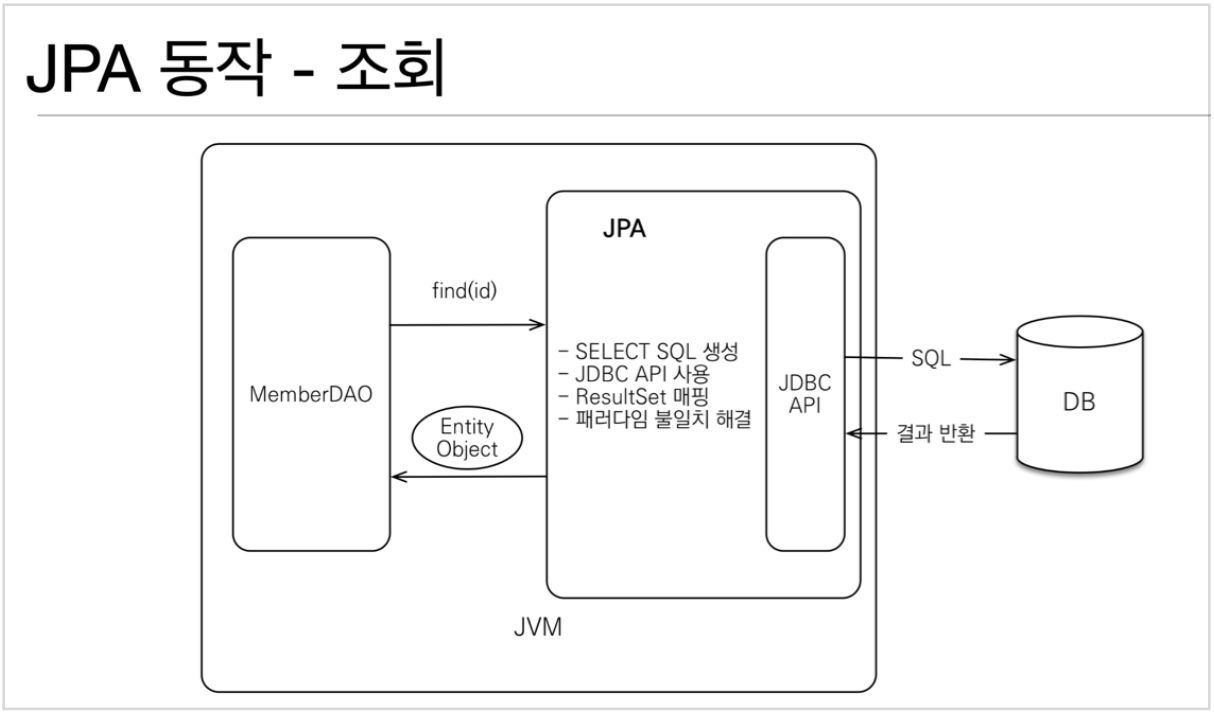
만약에 JDBC API, SQL을 실행해서 member1과 member2 객체를 비교하면? 다른 두 개의 인스턴스가 생성되는 것이기 때문에 다르다고 뜬다.
그러나 JPA의 방식을 써서 자바 컬렉션의 member1과 member2 객체를 getter를 사용하여 아이디로 불러와서 비교한다면 같은 인스턴스를 리턴해오기 때문에 같다고 뜬다.

===> 객체 지향적으로 설계한다고 하는데 객체답게 모델링 할수록 매핑작업만 복잡해진다!!!

jpa : java persistence API

java 진영의 ORM(OBJECT-RELATIONAL-MAPPING)
객체는 객체대로 설계하고 관계형 DB는 관계형 DB대로 설계함.
ORM 프레임워크가 중간에서 매핑해준다!





Hibernate : JPA(인터페이스의 모음임) 표준 명세를 구현한 구현체.

JPA의 상속

저장 - 그냥 persist 쓰면 다 해줌..
조회 - jpa.find쓰면 다 조회됨.. 조인할 필요없음1

연관관계와 객체 그래프 탐색

저장 조회 등 상속과 동일한 방법으로 가능.
=> 엔티티 계층 신뢰 가능. jpa가 관리하는 객체가 Entity. 엔티티가 한 행동 모두가 데이터로 저장되고, 이걸 모두 꺼낼 수 있음. 지연 로딩이라는 기술로 꺼낼수있다

JPA 비교하기

jpa에서 꺼낸 엔티티는 같다!
동일한 트랜잭션에서 조회한 엔티티는 같음을 jpa가 보장함.

JPA 성능 최적화 기능

1차 캐시와 동일성 보장

➜ jpa는 같은 트랜잭션 안에서는 같은 엔티티를 항상 반환한다. 약간의 조회 성능 향상.
중간에 기술이 하나 끼게 되면 두 가지 성능 향상. 모아서 보내는게 가능(버퍼 라이팅), 캐시(이미 조회한건 중간에 저장하고 필요할때꺼내줌)
=> 같은 트랜잭션 안에서 같은 엔티티를 항상 반환하는걸 어떻게 보장하냐?

- 처음 가져오는 엔티티는 sql에서 가져옴 ➜ 그런 다음 cache
- 두 번째 가져오는 엔티티는 첫 번째 가져왔던걸 캐시한 것에서 가져옴. => sql 한 번만 실행. 같은 트랜잭션에서만 가능!!!! 고객 요청이 여러개 오는 상황에선 불가능하다

트랜잭션을 지원하는 쓰기 지연(INSERT)

트랜잭션을 커밋할 때 까지 insert sql을 모음
jdbc batch sql 기능을 사용해서 한 번에 sql 쿼리문 전송함. 커밋 때 한번에 터뜨리기!!!

지연 로딩과 즉시 로딩

예를 들어서 멤버와 팀이 있는데 어떨 때는 멤버를 쓰고 어떨 때는 팀을 쓰는데 멤버를 조회할 때 팀을 항상 사용한다면 멤버를 조회할 때 팀을 조인해서 한번에 갖고오는게 나음.. => 즉시 로딩
근데 만약에 팀을 좀 자주 안쓰고 멤버만 자주 쓴다면 멤버만! ➜ 연관된걸 바로 안땡겨와도 된다.. 팀과 멤버가 연관되어있으나 멤버만 땡겨옴 => 지연 로딩

- 지연 로딩 : 객체가 실제로 사용될 때 로딩
- 즉시 로딩 : join sql로 한번에 연관된 객체까지 미리 조회

#JPA