(스터디) 10장 객체지향 쿼리 언어

10.1 객체지향 쿼리 소개 p.346

10.1.1 JPQL p.348

- 테이블이 아닌 엔티티 객체를 대상으로 검색하는 객체지향 쿼리다.
- SQL 을 추상화해서 특정 데이터베이스 SQL 에 의존하지 않는다.
- JPQL은 결국 SQL 로 변환된다.

10.2.1 기본 문법과 쿼리 API p.355

```
SELECT m FROM Member AS m where m.username = 'Hello'
```

- 대소문자를 구분한다(JPQL 키워드 `SELECT`, `where` 는 대소문자 구분 x).
- `Membe` 는 클래스 명이 아니라 엔티티 명이다.
- 별칭 필수(as 생략 가능)

- TypeQuery : 반환할 타입을 명확하게 지정할 수 있을 때 사용
- Query : 반환 타입이 명확하지 않을 때 사용

10.2.3 프로젝션 p.360

- 프로젝션 : SELECT 절에 조회할 대상을 지정하는 것
- 대상
    - 엔티티 프로젝션

        - SELECT **m** FROM Member m
        - SELECT **m.team** FROM Member m
        - 이렇게 조회한 엔티티는 영속성 컨텍스트에서 관리된다.
    - 임비디드 타입 프로젝션
        - SELECT **m.address** FROM Member m
        - 임베디드 엔티티 타입이 아닌 값 타입이다.
        - 따라서 이렇게 직접 조회한 임베디드 타입은 영속성 컨텍스트에서 관리되지 않는다.
    - 스칼라 타입 프로젝션
        - 스칼라 타입 : 숫자, 문자 등 기본 데이터 타입
        - SELECT **m.username, m.age** FROM Member m

- 여러 값 조회
    - Query 타입으로 조회
    - Object[] 타입으로 조회
    - new 명령어로 조회
        - 단순 값을 DTO로 바로 조회
        - SELECT **new** jpabook.jpql.UserDTO(m.username, m.age) FROM Member m
        - 패키지명을포함한전체클래스명입력
        - 순서와 타입이 일치하는 생성자 필요

10.2.4 페이징 API p.364

10.2.6 JPQL 조인 p.369

10.2.7 페치 조인 p.373

- JPQL 에서 성능 최적화를 위해 제공하는 기능
- 연관된 엔티티나 컬렉션을 한 번에 같이 조회하는 기능
- 별칭을 사용할 수 없다.

엔티티 페치 조인

- 회원을 조회하면서 연관된 팀도 함께 조회

```
select m from Member m join fetch m.team
```

```
SELECT M.*, T.* FROM MEMBER M
INNER JOIN TEAM T ON M.TEAM_ID=T.ID
```

- 회원(m)과 팀(m.team)을 함께 조회한다.

컬렉션 페치 조인

페치 조인과 DISTINCT

페치 조인과 일반 조인 차이

페치 조인의 특징과 한계

- 글로벌 로딩 전략은 될 수 있으면 지연 로딩을 사용하고 최적화가 필요하면 페치 조인을 적용하는 것이 효과적이다.
- 페치 조인을 사용하면 연관된 엔티티를 쿼리 시점에 조회하므로 지연 로딩이 발생하지 않는다.
    - 따라서 준영속 상태에서도 객체 그래프를 탐색할 수 있다.
- 페치 조인 대상에는 별칭을 줄 수 없다.
- 둘 이상의 컬렉션을 페치할 수 없다.
    - 카테시안 곱이 만들어진다.
- 컬렉션을 페치 조인하면 페이징 API 를 사용할 수 없다.

- 컬렉션(일대다)이 아닌 단일 값 연관필드(일대일, 다대일)들은 페치 조인을 사용해도 페이징 API 를 사용할 수 있다.

10.2.8 경로 표현식 p.382

- **경로표현식** : 점을 찍어 객체 그래프를 탐색하는 것
- **상태필드** : 단순히 값을 저장하기 위한 필드
- **연관필드** : 연관 관계를 위한 필드
  - 단일 값 연관 필드(@ManyToOne, @OneToOne)
  - 컬렉션 값 연관 필드(@OneToMany, @ManyToMany)
- 명시적 조인 : join 키워드 직접 사용
- 묵시적 조인 : 경로 표현식에 의해 묵시적으로 SQL 조인 발생(내부 조인만 가능)

10.2.11 다형성 쿼리 p.396

- TYPE
- TREAT

10.2.12 사용자 정의 함수 호출 p.398

10.2.14 엔티티 직접 사용 p.400 83

- 기본 키 값
  - JPQL에서 엔티티를 직접 사용하면 SQL에서 해당 엔티티의 기본 키 값을 사용한다.
- 외래 키 값

10.2.15 Named 쿼리 : 정적 쿼리 p.402

- 동적 쿼리
- 정적 쿼리

10.1.2 Criteria 쿼리 p.349, p.406

- JPQL 대신 직접 SQL 을 사용할 수 있다.
- 문자가 아닌 프로그래밍 코드로 JPQL 을 작성할 수 있다. `query.select(m).where()`

  ⇒ 컴파일 시점에 오류를 발견할 수 있다.
- IDE 를 사용하면 코드 자동완성을 지원한다.
- 동적 쿼리를 작성하기 편하다.
- 단점 : 사용하기 불편하고 한눈에 들어오지 않는다.

10.1.3 QueryDSL p.351, p.429

- 문자가 아닌 자바 코드로 JPQL 을 작성할 수 있다.
- 코드 기반이면서 단순하고 사용하기 쉽다.
- JPQL 빌더 역할
- 컴파일 시점에 문법 오류를 찾을 수 있다.

10.1.4 네이티브 SQL p.352, p.443

- JPQL 대신 직접 SQL 을 사용할 수 있다.
- JPQL 로 해결할 수 없는 특정 데이터베이스에 의존적인 기능
  - 오라클 CONNTECT BY, 특정 DB 만 사용하는 SQL 힌트

10.1.5 JDBC 직접 사용, MyBatis 같은 SQL 매퍼 프레임워크 사용 p.353

- JPA 르르 사용하면서 JDBC 커넥션을 직접 사용하거나 스프링 JdbcTemplate, 마이바티스 등을 함께 사용 가능
- 단 영속성 컨텍스트를 적절한 시점에 강제로 플러시해야한다.

10.6 객체지향 쿼리 심화 p.456

10.6.1 벌크 연산 p.457

어노테이션 프로세서

- 어노테이션 프로세서 기능을 사용하면 어노테이션을 분석해서 클래스를 생성할 수 있다.
- JPA 는 이 기능을 사용해서 Member 엔티티 클래스로부터 Member_ 라는 Criteria 전용 클래스를 생성하는데 이것을 **메타 모델**이라 한다. 350p