# JPA 를 잘 쓰고 있는가?

Java Persistence API

JDBC

MyBatis

JPA

Java Persistence API

HIBERNATE

# ORM

Object Realation Mapping

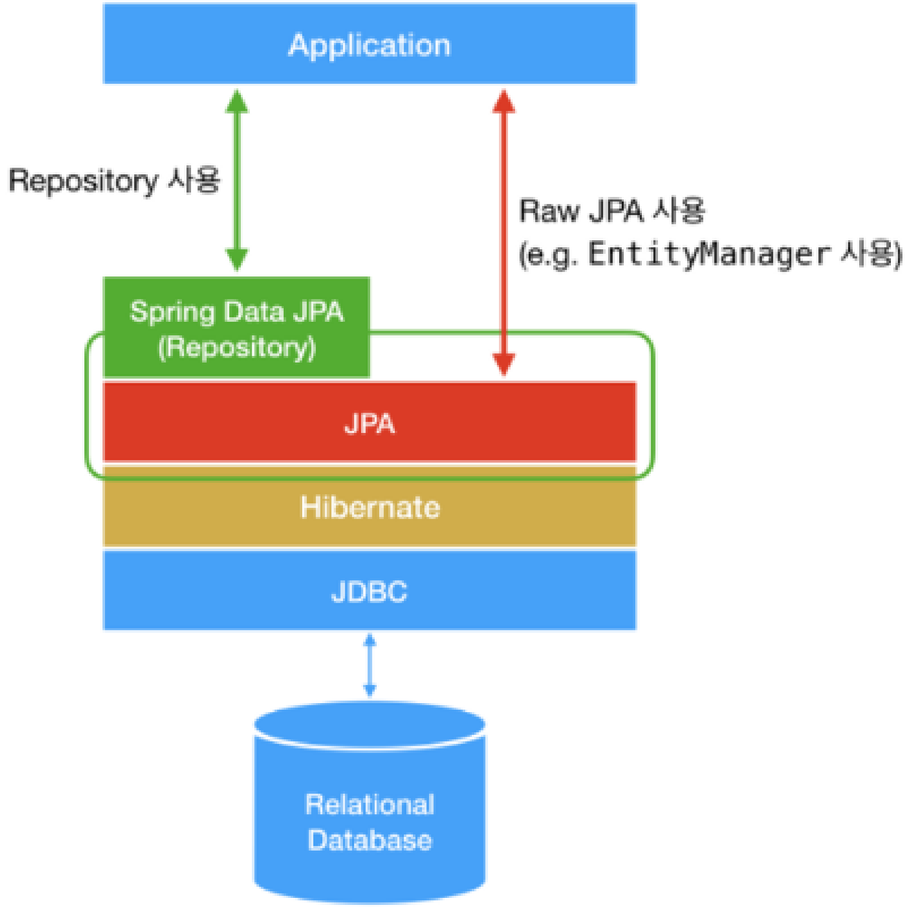
# JPA

Java Persistence API

HIBERNATE

# JPA

Java Persistence API



JPA
Java Persistence API

# N+1

- 데이터의 개수 N개 만큼 + 1 하여 쿼리문이 실행되는 현상

# JPA
Java Persistence API

구미_1반_권민우_D108의 어시스턴트 GPT-4

N+1 문제는 주로 데이터베이스를 사용하는 애플리케이션에서 발생하는 성능 이슈입니다. 이러한 문제는 ORM(Object-Relational Mapping)을 통해 데이터베이스와 상호작용하는 경우 특히 빈번하게 발생합니다.

N+1 문제의 이름은 이 문제가 발생하는 방식을 설명합니다. 예를 들어, 한 번의 쿼리로 N개의 결과를 가져온 후, 각 결과에 대해 추가 쿼리를 실행하는 경우입니다. 따라서 총 쿼리 수는 N (각 결과에 대한 쿼리) + 1 (초기 결과 쿼리)이 되어 'N+1'이라는 이름이 붙었습니다.

이 문제는 주로 'lazy loading' 전략 때문에 발생합니다. 이 전략은 필요한 데이터가 실제로 필요해질 때까지 로드를 연기하는 방식입니다. 이는 메모리 사용량을 줄이는 데는 효과적이지만, N+1 문제를 일으키는 주요 원인 중 하나입니다.

이 문제를 해결하는 한 가지 방법은 'eager loading' 전략을 사용하는 것입니다. 즉, 처음 쿼리를 실행할 때 연관된 데이터도 함께 로드하는 것입니다. 이렇게 하면 쿼리 수를 크게 줄일 수 있지만, 필요하지 않은 데이터까지 로드하게 되므로 메모리 사용량이 증가할 수 있습니다.

따라서 애플리케이션의 요구사항과 성능 목표에 따라 적절한 전략을 선택해야 합니다.

### FetchType.EAGER (즉시로딩)

- findAll() 사용

```
findAll()==============================
Hibernate: select team0_.id as id1_1_, team0_.name as name2_1_ from team team0_
Hibernate: select members0_.team_id as team_id1_2_0_, members0_.members_id as members_2_2_
Hibernate: select members0_.team_id as team_id1_2_0_, members0_.members_id as members_2_2_
Hibernate: select members0_.team_id as team_id1_2_0_, members0_.members_id as members_2_2_
Hibernate: select members0_.team_id as team_id1_2_0_, members0_.members_id as members_2_2_
Hibernate: select members0_.team_id as team_id1_2_0_, members0_.members_id as members_2_2_
After findAll()========================
findAll()==============================
```

즉시로딩 (EAGAR)

- 데이터를 조회하는 시점에서 관계데이터를 모두 가져온다.

---

### FetchType.LAZY (지연로딩)

- findAll() 사용

```
findAll()==============================
Hibernate: select team0_.id as id1_1_, team0_.name as name2_1_ from team team0_
findAll()==============================
```

- findAll() 이후 member 데이터 사용

```
findAll()==============================
Hibernate: select team0_.id as id1_1_, team0_.name as name2_1_ from team team0_
After findAll()========================
Hibernate: select members0_.team_id as team_id1_2_0_, members0_.members_id as members_2_2_
Hibernate: select members0_.team_id as team_id1_2_0_, members0_.members_id as members_2_2_
Hibernate: select members0_.team_id as team_id1_2_0_, members0_.members_id as members_2_2_
Hibernate: select members0_.team_id as team_id1_2_0_, members0_.members_id as members_2_2_
Hibernate: select members0_.team_id as team_id1_2_0_, members0_.members_id as members_2_2_
findAll()==============================
```

지연로딩 (Lazy)

- 데이터를 사용하는 시점에서 관계데이터를 가져온다.

JPA
Java Persistence API

Fetch Join

Batch Size

JPA
Java Persistence API

잘 쓰기 위해서는 알아야할게 너무 많다...

@Embeddable

casecade

QueryDSL