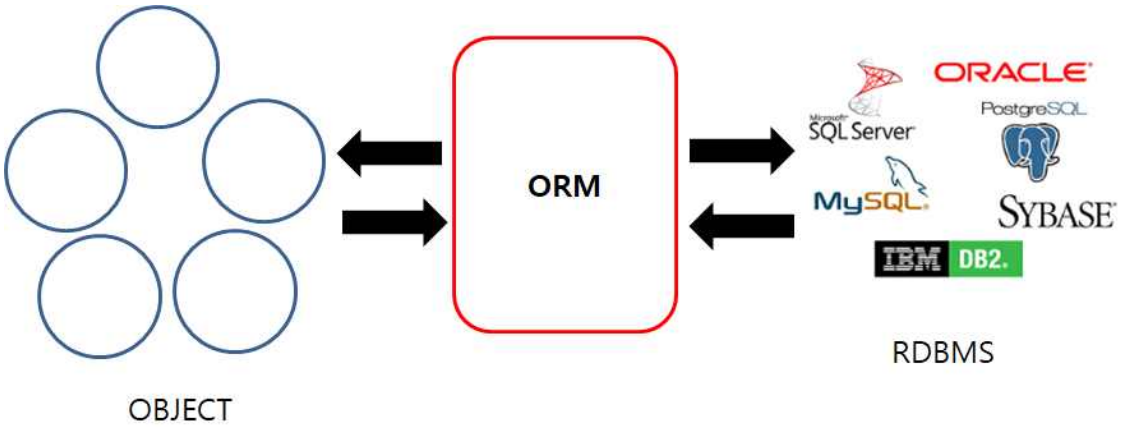
JPA(Java Persistence API) 관련 개념 정리

1. ORM (Object-Relational Mapping)



1) ORM

- ORM(Object-Relational Mapping)은 객체 지향 프로그래밍 언어에서 사용되는 객체와 관계형 데이터베이스의 테이블 간의 데이터를 자동으로 변환하는 기술이다. ORM을 사용하면 개발자는 데이터베이스와 상호작용할 때 SQL 쿼리를 직접 작성할 필요 없이, 객체 지향적인 방법으로 데이터베이스를 다룰 수 있다.

2) ORM의 기본 원리

I) 객체-관계 매핑

- ORM은 프로그램의 객체와 데이터베이스 테이블 간의 매핑을 제공한다. 즉, 객체의 구조가 데이터베이스의 테이블 구조와 매핑되어, 프로그래머가 객체 지향 방식으로 데이터베이스 작업을 할 수 있게 돕는다.

II) 데이터베이스 추상화

- ORM을 사용하면 SQL 쿼리를 직접 작성하는 대신, 고수준의 객체 지향 언어로 데이터베이스 작업을 수행할 수 있다. 이는 데이터베이스와의 상호 작용을 추상화하여 개발자가 복잡한 SQL 쿼리보다는 객체와 그들의 관계에 집중할 수 있게 한다.

3) ORM의 주요 기능

I) CRUD 작업의 간소화
- Create, Read, Update, Delete와 같은 기본적인 데이터베이스 작업을 객체의 메소드 호출로 수행할 수 있다.

II) 객체 관계 관리
- 다대일, 일대다, 다대다 등과 같은 복잡한 객체 관계를 관리하고 표현하는 데 도움을 준다.

III) 데이터 타입 변환
- 프로그래밍 언어의 데이터 타입과 데이터베이스 데이터 타입 간의 변환을 자동으로 처리한다.

4) ORM의 장점

I) 생산성 향상
- SQL 쿼리 작성에 드는 시간과 노력을 줄여준다.

II) 유지보수 용이성
- 데이터베이스 구조 변경 시, ORM 모델을 수정함으로써 쉽게 대응할 수 있다.

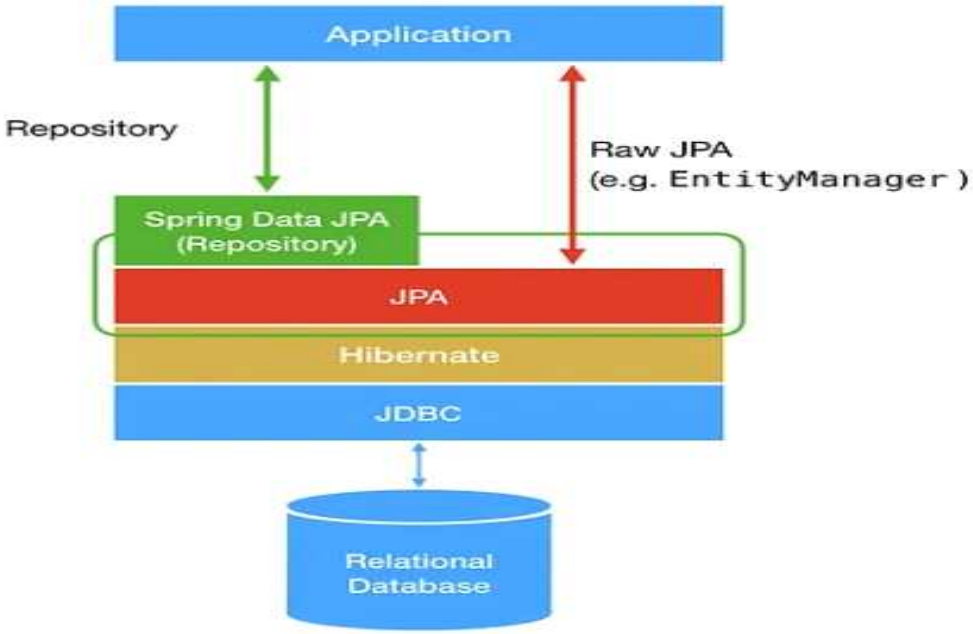
III) 플랫폼 독립성
- 특정 데이터베이스 시스템에 종속되지 않고, 다양한 데이터베이스 시스템과 호환된다.

5) ORM의 단점

I) 성능 저하
- 복잡한 쿼리의 경우, 직접 작성한 SQL에 비해 성능이 떨어질 수 있다.

II) 학습 곡선
- ORM 자체의 복잡성과 세부 사항을 이해하는 데 시간이 필요하다.

2. JPA (Java Persistence API)



이미지 출처
https://medium.com/@dafikabukcu/what-is-hibernate-and-jpa-ef77ba1dac15

1) JPA

- JPA는 Java 어플리케이션에서 관계형 데이터베이스를 관리하기 위한 표준 ORM(Object-Relational Mapping) 프레임워크이다. JPA는 자바 객체와 데이터베이스 테이블 간의 매핑을 쉽게 해주며, 데이터베이스 작업을 간결하고 효율적으로 만들어 준다.

2) JPA 핵심 기능

I) 객체-관계 매핑 (ORM)
- 자바 객체를 데이터베이스 테이블에 매핑하여, 객체 지향적인 방식으로 데이터베이스를 다룰 수 있게 해준다.

II) 쿼리 언어 (JPQL)
- SQL과 유사하지만 엔티티 객체를 대상으로 하는 쿼리 언어를 제공한다.

III) 자동화된 데이터베이스 작업
- 객체의 생성, 조회, 수정, 삭제와 같은 기본적인 데이터베이스 작업을 자동화한다.

3) JPA 사용 장점

I) 생산성 향상
- 반복적인 CRUD 작업을 간단하게 처리할 수 있어 개발 시간을 단축시킨다.

II) 데이터베이스 독립성
- 다양한 데이터베이스 제품에 대해 일관된 접근 방식을 제공한다.

III) 유지보수 용이
- 객체 중심의 개발로 코드가 간결해지고, 유지보수가 용이해진다.

3. Hibernate

- Hibernate는 Java 어플리케이션에서 관계형 데이터베이스를 사용하기 위한 ORM(Object-Relational Mapping) 프레임워크이다. Hibernate는 데이터베이스와의 상호작용을 쉽고 효율적으로 만들어 주는 도구로, 객체 지향 프로그래밍과 관계형 데이터베이스의 간격을 줄여준다.

1) Hibernate의 핵심 기능

I) 객체-관계 매핑 (ORM)
- 자바 객체를 데이터베이스 테이블에 매핑한다. 이를 통해 객체 지향적인 방식으로 데이터베이스를 다룰 수 있다.

II) 데이터베이스 독립성
- 다양한 데이터베이스 시스템에 대해 동일한 API를 제공한다.

III) 편리한 쿼리 작성
- HQL(Hibernate Query Language)을 사용하여 데이터베이스 작업을 수행할 수 있다.

IV) 자동화된 트랜잭션 관리

- 세션(Session)과 트랜잭션(Transaction) 관리를 통해 데이터 일관성을 유지한다.

2) Hibernate 사용의 장점

I) 개발 효율성

- 반복적인 CRUD 작업을 간단하게 처리할 수 있어 개발 시간이 단축된다.

II) 향상된 유지보수

- 객체 중심 개발로 코드가 간결해지고, 유지보수가 용이해진다.

III) 쿼리 최적화

- 캐시 메커니즘, 지연 로딩(Lazy Loading) 등 다양한 최적화 기법을 제공한다.

3) ORM , JPA , Hibernate의 구분

ORM(Object Relation Mapping)

- 객체 지향 프로그래밍 언어를 사용하여 관계형 데이터베이스를 관리하는 방법론이다. 이는 데이터베이스의 테이블을 자바 객체로 표현하며, 이 객체를 통해 데이터베이스와 상호작용한다.

JPA (Java Persistence API)

- JPA는 자바 어플리케이션에서 ORM을 구현하기 위한 표준 명세이다. 이는 데이터베이스 작업을 수행하는 데 필요한 API와 규칙들을 제공한다.

Hibernate

- Hibernate는 JPA 명세를 구현한 구체적인 프레임워크이다. 이는 JPA가 정의한 인터페이스를 실제로 구현하며, 개발자가 데이터베이스 작업을 손쉽게 할 수 있도록 도와준다.