데이터들이 프로그램이 종료되어도 사라지지 않고 어떤 곳에 저장되는 개념을 **영속성(Persistence)** 이라고 한다.

자바에서는 데이터의 영속성을 위한 JDBC를 지원해주는데, 이는 매핑 작업을 개발자가 일일히 수행해야 하는 번거로움이 있다.

**SQL Mapper**와 **ORM**은 개발자가 직접 JDBC Programming을 하지 않도록 기능을 제공해주는 **Persistence Framework** 종류이다.

기존 JDBC 만의 사용으로 쿼리문을 만들어 요청하는 과정은 쿼리문이 조금만 길어져도 관리가 힘들거 번거롭다. 따라서 **JPA**와 **MyBatis** 라이브러리를 사용하여 문제를 해결할 수 있다.

**> SQL Mapper**

* **Object와 SQL의 필드를 매핑하여 데이터를 객체화 하는 기술**
  + 객체와 테이블 간의 관계를 매핑하는 것이 아님
  + SQL문을 직접 작성하고 쿼리 수행 결과를 어떠한 객체에 매핑할지 바인딩 하는 방법
  + DBMS에 종속적인 문제
  + EX) JdbcTemplate, MyBatis

**> MyBatis(SQL Mapper)**

* **MyBatis : Object Mapping 기술**
  + 자바에서 SQL Mapper를 지원해주는 프레임워크
  + SQL문을 이용해서 RDB에 접근, 데이터를 객체화 시켜줌
  + SQL을 직접 작성하여 쿼리 수행 결과를 객체와 매핑
  + 쿼리문을 xml로 분리 가능
  + 복잡한 쿼리문 작성 가능
  + 데이터 캐싱 기능으로 성능 향상
  + but 객체와 쿼리문 모두 관리해야함, CRUD 메소드를 직접 다 구현해야함.

**> ORM**

* **Object와 DB테이블을 매핑하여 데이터를 객체화하는 기술**
  + 개발자가 반복적인 SQL을 직접 작성하지 않음
  + DBMS에 종속적이지 않음
  + 복잡한 쿼리의 경우 JPQL을 사용하거나 SQL Mapper을 혼용하여 사용 가능

**> JPA(ORM)**

* **JPA : ORM(Object Relational Mapping) 기술**
  + 자바 ORM의 기술 표준
  + 대표적인 오픈소스로 Hibernate
  + CRUD 메소드 기본 제공
  + 쿼리를 만들지 않아도 됨
  + 1차 캐싱, 쓰기지연, 변경감지, 지연로딩 제공
  + MyBatis는 쿼리가 수정되어 데이터 정보가 바뀌면 그에 사용 되고 있던 DTO와 함께 수정해주어야 하는 반면에, JPA 는 **객체만 바꾸면 된다**.
  + 즉, 객체 중심으로 개발 가능
  + but 복잡한 쿼리는 해결이 어려움

## > ORM 이란?

**객체(Object)와 DB의 테이블을 Mapping 시켜 RDB 테이블을 객체지향적으로 사용하게 해주는 기술**

객체와 RDB를 별개로 설계하고 중간에서 ORM이 매핑해주는 역할을 한다. 즉, ORM은 SQL문이 아닌 **RDB의 데이터 그 자체와 매핑**하기 때문에 SQL을 직접 작성하지 않는다.

여기서 RDB 테이블이란, 관계형 데이터베이스를 말한다. [참고](https://noahlogs.tistory.com/37)

RDB 테이블은 객체지향적 특성(상속, 다형성, 레퍼런스) 등이 없어서 Java와 같은 객체 지향적 언어로의 접근이 쉽지 않다.

이럴때 **ORM**을 사용하면 보다 객체지향적으로 RDB를 사용할 수 있다!

Java에서 사용하는 대표적인 ORM으로는 JPA와 그의 구현체 Hibernate가 있다. JPA가 등장하기 전에는 위에 설명했듯이 MyBatis라는 Object Mapping 기술을 이용하였는데, MyBatis는 Java 클래스 코드와 직접 작성한 SQL 코드를 매핑 시켜주어야 했다.

하지만! **JPA와 같은 ORM기술**은 객체가 DB에 연결되기 때문에 SQL을 직접 작성하지 않는다.

따라서 JPA라는 ORM 기술에 의해 DB에서 조회한 데이터들이 객체로 연결되어 있고, **객체의 값을 수정하는 것은 DB의 값을 수정하는 것**이라고 할 수 있다.