



**FPT POLYTECHNIC**



Conceive Design Implement Operate

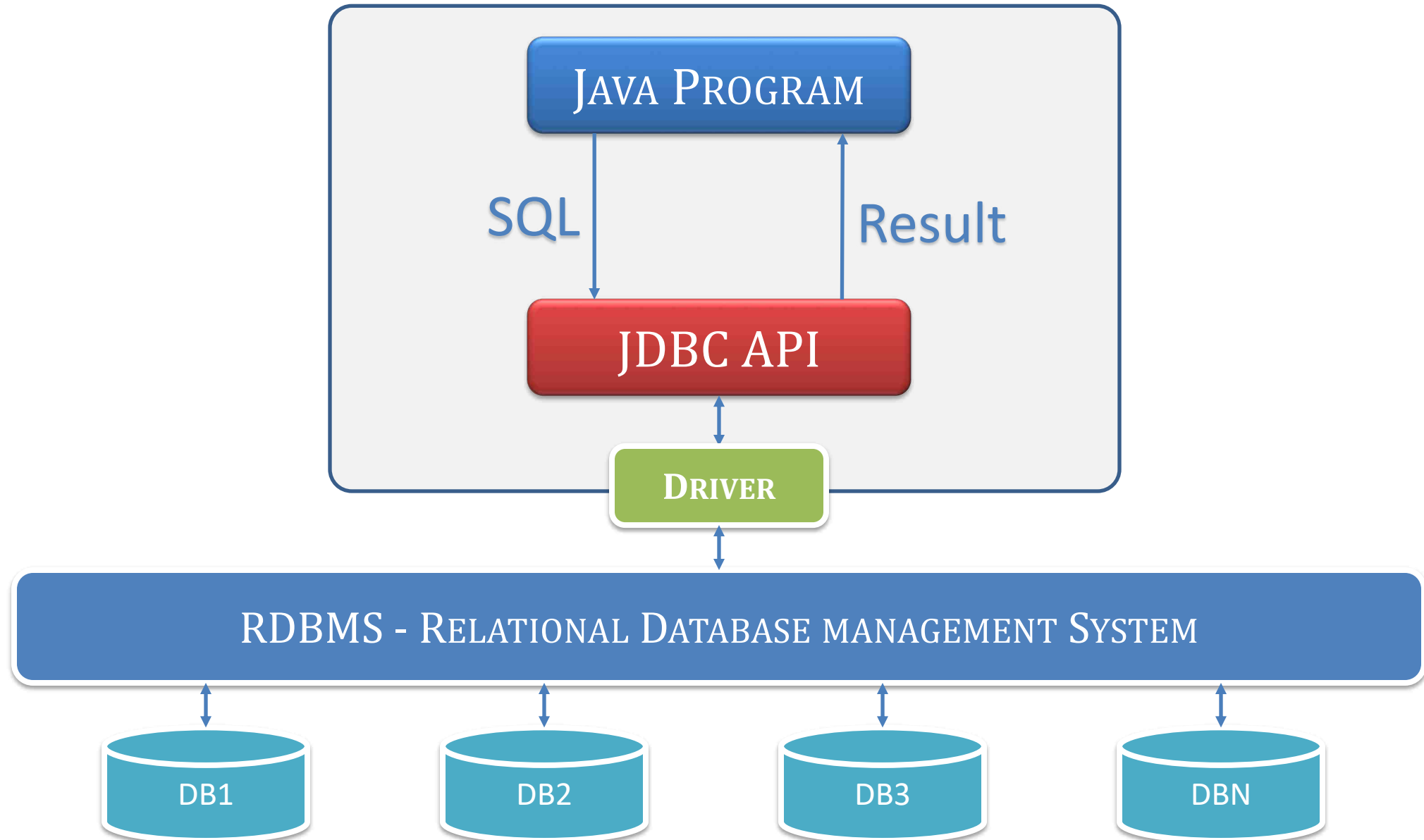
**LẬP TRÌNH JPA**

**LẬP TRÌNH JAVA #4 (P1.1)**

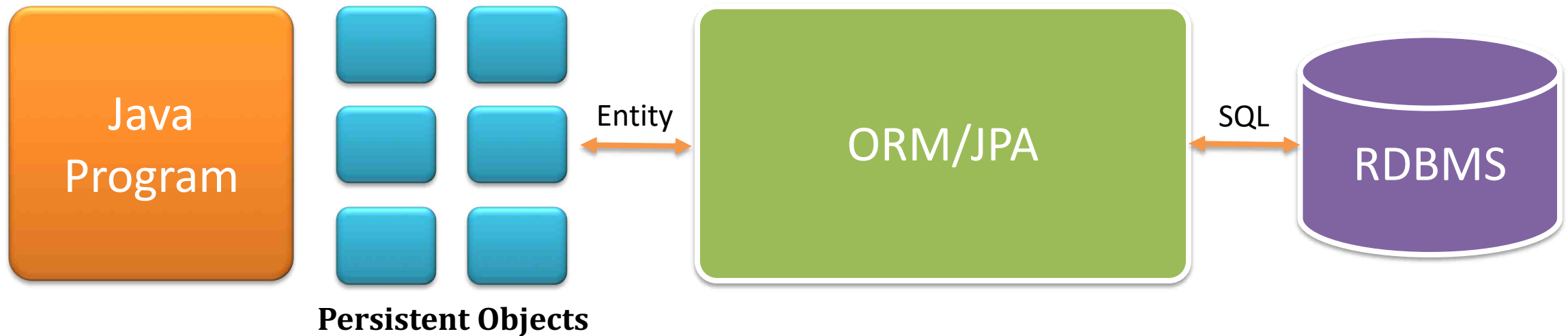
[www.poly.edu.vn](http://www.poly.edu.vn)

- ☐ Hạn chế của lập trình JDBC
- ☐ Giới thiệu JPA
- ☐ Cấu hình thư viện cần thiết
- ☐ Lập trình JPA truy vấn và thao tác dữ liệu cơ bản



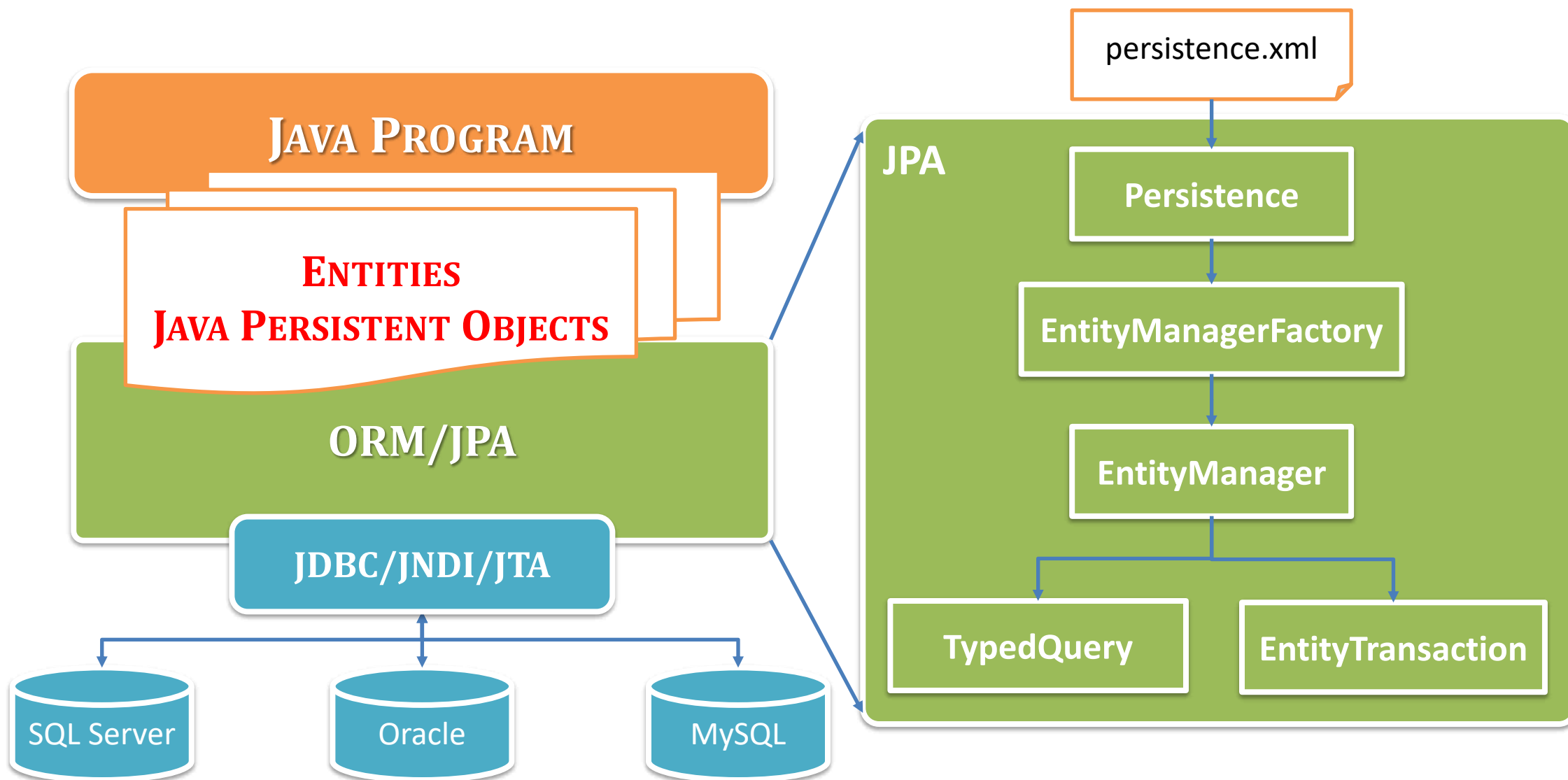


- ❑ Viết nhiều câu lệnh SQL => tốn nhiều thời gian, dễ vấp sai sót.
- ❑ Viết quá nhiều mã java cho việc truy vấn và thao tác dữ liệu (vì phải chuyển đổi từ đối tượng thành SQL và ngược lại từ ResultSet thành Collection)
- ❑ Khá khó khăn trong việc điều khiển Transaction
- ❑ Nâng cấp khó khăn vì SQL phụ thuộc vào hệ quản trị CSDL, nếu thay đổi hệ quản trị CSDL thì phải viết lại mã
- ❑ Không được hỗ trợ bởi các dịch vụ nền (Connection Pool, Instance Pool, Transaction...)



- ❑ **Java Program** chỉ làm việc với **Persistent Objects** (đối tượng lưu trữ lâu dài)
- ❑ *Code Java không phụ thuộc vào SQL của hệ quản trị CSDL từ đó dễ dàng nâng cấp thay đổi hệ quản trị CSDL mà không ảnh hưởng đến Java Program trong lúc đang vận hành.*

- ❑ **ORM** (Object Relational Mapping) ánh xạ dữ liệu của Persistent Objects với các record của các table trong CSDL quan hệ thông qua các lớp thực thể (Entity Class).
- ❑ **Entity Class** (lớp thực thể) là các lớp mô tả cấu trúc dữ liệu của các table bằng XML hoặc annotation giúp ORM chuyển đổi dữ liệu giữa Persistent Objects và các Records một cách chính xác.
- ❑ **JPA** (Java Persistence API) là thư viện cung cấp các interface giúp lập trình truy vấn và thao tác dữ liệu



- ❑ JPA Provider: JPA chỉ là bản đặc tả (interface), các lớp thực thi theo JPA cần phải nạp vào thông qua thư viện phụ thuộc. Có khá nhiều nhà cung cấp thư viện này, Hibernate là một trong số đó.
- ❑ JDBC Driver: Mỗi hệ quản trị CSDL cung cấp mỗi JDBC khác nhau, SQL Server Driver dùng để lập trình với SQL Server

```
<!--Hibernate/JPA-->
```

```
<dependency>
```

```
    <groupId>org.hibernate</groupId>
```

```
    <artifactId>hibernate-core</artifactId>
```

```
    <version>6.4.2.Final</version>
```

```
</dependency>
```

```
<!--SQL Server Driver-->
```

```
<dependency>
```

```
    <groupId>com.microsoft.sqlserver</groupId>
```

```
    <artifactId>mssql-jdbc</artifactId>
```

```
    <version>12.8.0.jre8</version>
```

```
</dependency>
```



- ❑ Tạo CSDL PolyOE và tạo bảng Users có cấu trúc như sau

```
USE master
```

```
GO
```

```
CREATE DATABASE PolyOE
```

```
GO
```

```
USE PolyOE
```

```
GO
```



```
CREATE TABLE Users(
```

```
    Id NVARCHAR(20) NOT NULL,
```

```
    Password NVARCHAR(50) NOT NULL,
```

```
    Fullname NVARCHAR(50) NOT NULL,
```

```
    Email NVARCHAR(50) NOT NULL,
```

```
    Admin BIT NOT NULL,
```

```
    PRIMARY KEY(Id)
```

```
)
```

## ❑ src/META-INF/persistence.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
  <persistence-unit name="PolyOE">
    <properties>
      <property name="javax.persistence.jdbc.driver" value="com.microsoft.sqlserver.jdbc.SQLServerDriver" />
      <property name="javax.persistence.jdbc.url" value="jdbc:sqlserver://localhost;database=DB" />
      <property name="javax.persistence.jdbc.user" value="user" />
      <property name="javax.persistence.jdbc.password" value="password" />
      <property name="hibernate.show_sql" value="true" />
      <property name="hibernate.format_sql" value="true" />
    </properties>
  </persistence-unit>
</persistence>
```



# DEMOSTATION

---

pom.xml và persistence.xml

# XÂY DỰNG LỚP THỰC THỂ (ENTITY CLASS)

- Entity Class là lớp mô tả cấu trúc dữ liệu của một bảng tương ứng trong CSDL.
- Mỗi đối tượng được đồng bộ tương ứng với 1 bảng ghi

THANHVINHCENTER.PolyOE - dbo.Users X			
	Column Name	Data Type	Allow Nulls
▶	Id	nvarchar(20)	<input type="checkbox"/>
	Password	nvarchar(50)	<input type="checkbox"/>
	Fullname	nvarchar(50)	<input type="checkbox"/>
	Email	nvarchar(50)	<input type="checkbox"/>
	Admin	bit	<input type="checkbox"/>

@Entity  
@Table(name="Users")  
User

@Id  
@Column(name="Id")  
id: String

@Column(name="Password")  
password: String

@Column(name="Fullname")  
fullname: String

@Column(name="Email")  
email: String

@Column(name="Admin")  
admin: Boolean

```
@Entity
@Table(name = "Users")
public class User {
    @Id
    @Column(name = "id")
    String id;
    @Column(name = "password")
    String password;
    @Column(name = "fullname")
    String fullname;
    @Column(name = "email")
    String email;
    @Column(name = "admin")
    Boolean admin = false;

    getters/setters
}
```

❑ Sử dụng các annotation để ánh xạ với bảng Users

- ❖ **@Entity**: khai báo lớp thực thể
- ❖ **@Table**: ánh xạ với bảng
- ❖ **@Column**: ánh xạ với cột
- ❖ **@Id**: ràng buộc với cột khóa chính

❑ Chú ý:

- ❖ Lớp User phải là lớp Java Bean
  - Lớp phải khai báo public
  - Có constructor mặc định không tham số
  - Khai báo getters và setters
- ❖ Annotation phải đặt ngay trên các class, field được ánh xạ



// 1. Nạp persistence.xml và tạo EntityManagerFactory

```
EntityManagerFactory factory = Persistence.createEntityManagerFactory("PolyOE");
```

// 2. Tạo EntityManager chuẩn bị lập trình CSDL

```
EntityManager em = factory.createEntityManager();
```

<persistence-unit name="PolyOE">

// 3.1. Lập trình thao tác dữ liệu

```
em.getTransaction().begin();
```

```
try {
```

```
    em.persist(entity); // em.merge(entity); | em.remove(entity);
```

```
    em.getTransaction().commit();
```

```
} catch (Exception e) {
```

```
    em.getTransaction().rollback();
```

```
}
```

// 3.2 Lập trình truy vấn một thực thể theo khóa chính

```
User user = em.find(User.class, "NghiemN");
```

// 4. Đóng EntityManager và kết thúc lập trình CSDL

```
em.close();
```

Lập trình thao tác dữ liệu cần được điều khiển **Transaction**.

**Transaction** = "**None or All**" (được ăn cả, ngã về không)

```
EntityManagerFactory factory = Persistence.createEntityManagerFactory("PolyOE");  
EntityManager em = factory.createEntityManager();
```

*// Câu lệnh JPQL truy vấn thực thể*

```
String jpql = "SELECT o FROM User o";
```

*// Tạo Query/TypedQuery<T> để truy vấn*

```
TypedQuery<User> query = em.createQuery(jpql, User.class);
```

*// Truy vấn danh sách thực thể*

```
List<User> list = query.getResultList();
```

*// Truy vấn một thực thể (nếu JPQL đảm bảo chỉ có 1 thực thể)*

```
User entity = query.getSingleResult();
```

```
em.close();
```

**JPQL** là câu lệnh truy vấn đối tượng (các thành phần bên trong là Entity Class và Property chứ không phải là Table và Column)

**TypedQuery<T>**

- **getResultList(): List<T>**
- **getSingleResult(): T**

*Hãy căn cứ mệnh đề SELECT để xác định T*



# DEMOSTATION

---

Lập trình JPA truy vấn và thao tác Users



- ✓ Hạn chế của lập trình JDBC
- ✓ Giới thiệu JPA
- ✓ Cấu hình thư viện cần thiết
- ✓ Lập trình JPA truy vấn và thao tác dữ liệu cơ bản





**Cảm ơn**