

JPA P2

Nguyễn Huy Cường - nh.cuong@hutech.edu.vn

06/2024

Nội dung

1. Không sử dụng ràng buộc (khóa ngoại)
2. Sử dụng ràng buộc

Demo

- ✓ Thêm sản phẩm: Name, Image, Price, Category được lấy từ bảng danh mục category
- ✓ Danh sách sản phẩm: Lấy tất cả sản phẩm + tên danh mục
- ✓ Edit: Thay đổi sản phẩm, có thể thêm chi tiết sản phẩm (mô tả, nhà Sx, năm Sx)

Home Products



Name
Lenovo thinkpad new

Image: thinkpad_x1_..._gen_11_1.png

Price:
12345


Category:
Laptop

Home Products

#	Name	Image	Price	Category Name	
1	Máy tính Laptop 2022		27000	Laptop	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
2	Lenovo thinkpad new		22000	Laptop	<input type="button" value="Edit"/> <input type="button" value="Delete"/>

Home Products

Name
Lenovo thinkpad new

Image:


Change Image: Không có tệp nào được chọn

Price:
22000

Description
sản phẩm tốt
sản phẩm tốt
sản phẩm tốt

Manufacturer
Lenovo

Manufacture Year
2023

1. Không sử dụng mối quan hệ

- Khi không sử dụng ràng buộc (khóa ngoại)
- ❑ Ưu điểm: Hiệu suất cao hơn, linh hoạt hơn như: dễ dàng thay đổi cấu trúc dữ liệu
- ❑ Nhược điểm: Khó phát hiện lỗi, tính toàn vẹn dữ liệu, phát triển & bảo trì tốn thời gian hơn..

- Với yêu cầu bài toán Demo => Xây dựng CSDL

product (id, name, image, price, category_id)

category (id, name)

product_detail (id, description, manufacturer, manufacture_year)

product	
id 🔗	integer
category_id	integer NN
image 📄	varchar(200)
name	varchar(255) NN
price	bigint NN

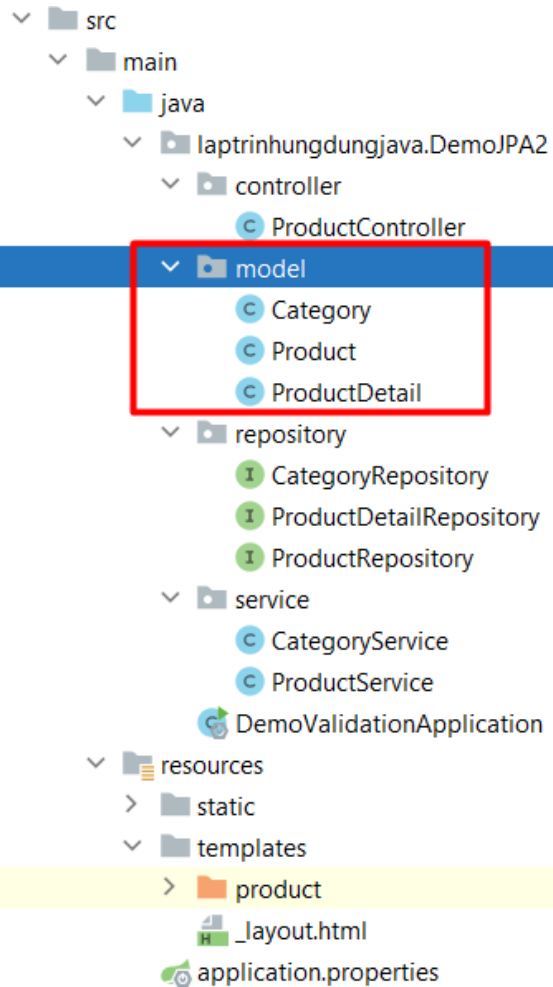
product_detail	
id 🔗	integer
description 📄	varchar(255)
manufacture_year	integer NN
manufacturer 📄	varchar(255)

category	
id 🔗	integer
name 📄	varchar(255)

Thực hiện Demo

- **Entity:** Tạo các thực thể JPA: **Product**, **ProductDetail** và **Category**
- **Repository:** Tạo **ProductRepository**, **ProductDetailRepository**, **CategoryRepository** để cung cấp các phương thức và công cụ để làm việc với các thực thể JPA.
- **Service:** Tạo các lớp dịch vụ trung gian giúp mã nguồn dễ bảo trì, kiểm thử và mở rộng: **ProductService**, **CategoryService**, **ProductDetailsService**
- **Controller:** Xử lý các yêu cầu HTTP từ người dùng và phản hồi: **ProductController**
- **Thymeleaf**

Entity



```
@Getter @Setter @NoArgsConstructor @AllArgsConstructor
@Entity
public class Product {
    no usages
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    no usages
    @NotBlank(message = "Tên sản phẩm không được để trống")
    private String name;
    no usages
    @Length(min= 0,max = 200, message = "Tên hình ảnh không quá 200 kí tự")
    private String image;
    no usages
    @NotNull(message = "Giá sản phẩm không được để trống")
    @Min(value = 1, message = "Giá sản phẩm không được nhỏ hơn 1")
    @Max(value = 9999999, message = "Giá sản phẩm không được lớn hơn 9999999")
    private long price;
    no usages
    private int categoryId;
    no usages
    @Transient
    private ProductDetail productDetail;
    no usages
    @Transient
    private Category category;
}
```

```
8 usages
@Getter @Setter @NoArgsConstructor @AllArgsConstructor
@Entity
public class Category {
    no usages
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    no usages
    private String name;
}
```

```
13 usages
@Getter @Setter @NoArgsConstructor @AllArgsConstructor
@Entity
public class ProductDetail {
    no usages
    @Id
    private int id;
    no usages
    private String description;
    no usages
    private int manufactureYear;
    no usages
    private String manufacturer;
    //etc..
}
```

Repository

laptoptrinhungdungjava.DemoJPA2

controller

ProductController

model

repository

CategoryRepository

ProductDetailRepository

ProductRepository

service

CategoryService

ProductService

DemoValidationApplication

resources

static

templates

product

_layout.html

application.properties

@Repository

```
public interface CategoryRepository extends JpaRepository<Category, Integer> {
}
```

@Repository

```
public interface ProductDetailRepository extends JpaRepository<ProductDetail, Integer> {
}
```

@Repository

```
public interface ProductRepository extends JpaRepository<Product, Integer> {
}
```

- Các phương thức sẵn có: *save, delete, saveAndFlush, findById, findAll...*

// Thêm đối tượng mới

Product **save**(Product product);

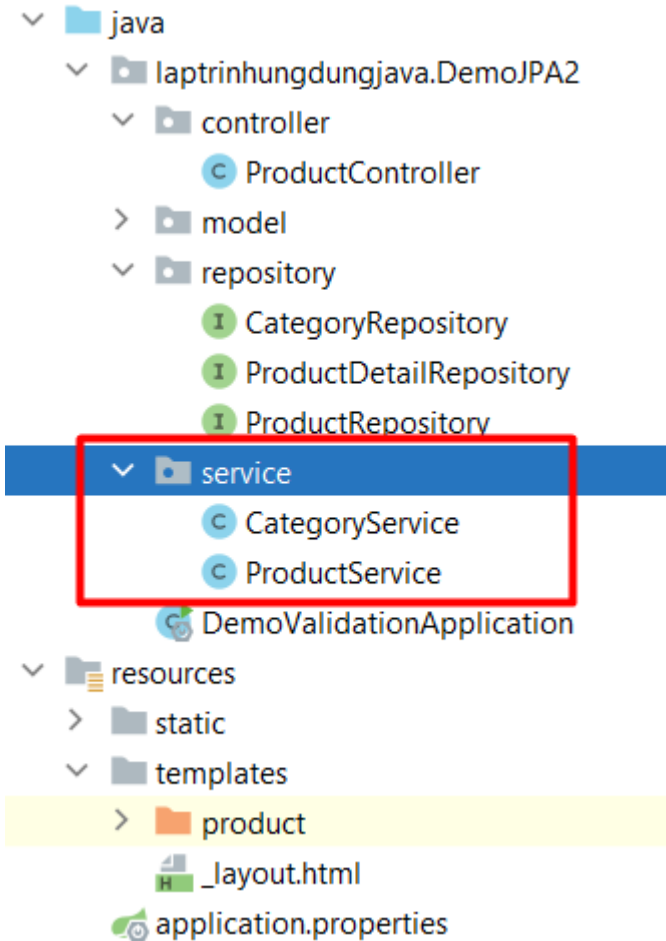
// Xóa đối tượng

void **delete**(Product product);

// Sửa đối tượng

Product **saveAndFlush**(Product product);

Service



```
@Service
public class CategoryService {
    1 usage
    @Autowired
    private CategoryRepository categoryRepository;
    3 usages
    public List<Category> getAll() {
        return categoryRepository.findAll();
    }
}
```

```
public class ProductService {
    4 usages
    @Autowired
    private ProductRepository productRepository;
    5 usages
    @Autowired
    private ProductDetailRepository productDetailRepository;
    1 usage
    @Autowired
    private CategoryRepository categoryRepository;
    1 usage
    public List<Product> getAll() {
        List<Product> listProduct = productRepository.findAll();
        for (Product p : listProduct) {
            ProductDetail d = productDetailRepository.findById(p.getId()).orElse( other: null);
            if(d!= null)
                p.setProductDetail(d);

            Category c = categoryRepository.findById(p.getCategoryId()).orElse( other: null);
            if(c!= null)
                p.setCategory(c);
        }
        return listProduct;
    }
}
```



```

public void add(Product newProduct) {
    Category c = newProduct.getCategory();
    if(c!= null)
        newProduct.setCategoryId(c.getId());
    productRepository.save(newProduct);
}

public void updateImage(Product newProduct, MultipartFile imageProduct)
{
}

public void update(Product editProduct)
{
    Product find = get(editProduct.getId());
    if(find!= null) {
        find.setPrice(editProduct.getPrice());
        find.setName(editProduct.getName());
        if(editProduct.getImage()!= null)
            find.setImage(editProduct.getImage());
        find= productRepository.saveAndFlush(find);
        ProductDetail d = productDetailRepository.findById(find.getId()).orElse(null);
        if(d == null) {
            ProductDetail item = editProduct.getProductDetail();
            item.setId(find.getId());
            productDetailRepository.save(item);
        }
        else
        {
            ProductDetail item = editProduct.getProductDetail();
            d.setDescription(item.getDescription());
            d.setManufacturer(item.getManufacturer());
            d.setManufactureYear(item.getManufactureYear());
            productDetailRepository.saveAndFlush(d);
        }
    }
}
}

```

Controller

✓ Lấy danh sách sản phẩm, thêm, sửa


```
@Controller
@RequestMapping("/products")
public class ProductController {
    @Autowired
    private ProductService productService;
    @Autowired
    private CategoryService categoryService;

    @GetMapping()
    public String Index(Model model)
    {
        model.addAttribute("listproduct", productService.getAll());
        return "product/products";
    }
    @GetMapping("/create")
    public String Create(Model model) {
        model.addAttribute("product", new Product());
        model.addAttribute("categories", categoryService.getAll());
        return "product/create";
    }
    @PostMapping("/create")
    public String Create(@Valid Product newProduct,
        BindingResult result,
        @RequestParam MultipartFile imageProduct,
        Model model) {
        if (result.hasErrors()) {
            model.addAttribute("product", newProduct);
            model.addAttribute("categories", categoryService.getAll());
            return "product/create";
        }
        productService.updateImage(newProduct, imageProduct);
        productService.add(newProduct);
        return "redirect:/products";
    }
}
```

```
@GetMapping("/edit/{id}")
public String Edit(@PathVariable int id, Model model) {
    Product find = productService.get(id);
    if(find == null)
        throw new IllegalStateException("Product not found with ID: " + id); //error page
    model.addAttribute("product", find);
    model.addAttribute("categories", categoryService.getAll());
    return "product/edit";
}
@PostMapping("/edit")
public String Edit(@Valid Product editProduct,
    BindingResult result,
    @RequestParam MultipartFile imageProduct,
    Model model) {
    if (result.hasErrors()) {
        model.addAttribute("product", editProduct);
        model.addAttribute("categories", categoryService.getAll());
        return "product/edit";
    }
    productService.updateImage(editProduct, imageProduct);
    productService.update(editProduct);
    return "redirect:/products"; // Redirect to the products page after successful update
}
```

Thymeleaf

✓ products

Image	Price	Category Name
	27000	Laptop

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout" layout:decorate="_layout" xmlns:custom="http://www.w3.org/1999/xhtml">
<head>
  <meta charset="UTF-8">
  <title>List products</title>
</head>
<body>
<div layout:fragment="content" class="container body-content">
  <a th:href="@{products/create}" class="btn btn-primary">Create New Product</a>
  <table class="table table-striped">
    <thead>
      <tr>
        <th scope="col">#</th>
        <th scope="col">Name</th>
        <th scope="col">Image</th>
        <th scope="col">Price</th>
        <th scope="col">Category Name</th>
        <th scope="col"></th>
      </tr>
    </thead>
    <tbody>
      <tr th:each="product : ${listproduct}" >
        <th scope="row" th:text="${product.id}"></th>
        <td th:text="${product.name}"></td>
        <td>
          
        </td>
        <td th:text="${product.price}"></td>
        <td th:text="${product.category?.name ?: ''}"></td>
        <td>
          <a th:href="@{/products/edit/{id}(id=${product.id})}" custom:linkMethod="post" class="btn btn-secondary">Edit</a>
          <a th:href="@{/products/delete/{id}(id=${product.id})}" custom:linkMethod="post" class="btn btn-danger">Delete</a>
        </td>
      </tr>
    </tbody>
  </table>
</div>
</body>
</html>
```

Thymeleaf

✓ create

Products

Name

Product Name

Image: Không có tệp nào được chọn

Price:

0

Category:

Laptop

Laptop

Điện thoại

Điện lạnh


```
<!DOCTYPE html>
<html lang="en"
  xmlns:th="http://www.thymeleaf.org"
  xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
  layout:decorate=" _layout">
<head>
  <meta charset="UTF-8">
  <title>Create product</title>
</head>
<body>
<div layout:fragment="content" class="container body-content">
  <form th:action="@{/products/create}" th:object="${product}" method="post" class="form" enctype="multipart/form-data">
    <div class="form-group">
      <label for="name">Name</label>
      <input class="form-control" type="text" th:field="*{name}" id="name" placeholder="Product Name">
      <div class="alert alert-warning" th:if="${#fields.hasErrors('name')}" th:errors="*{name}"></div>
    </div>
    <div class="form-group">
      <label for="image">Image:</label>
      <input class="form-control-file" type="file" id="image" name="imageProduct" accept="image/png,image/jpeg">
      <div class="alert alert-warning" th:if="${#fields.hasErrors('image')}" th:errors="*{image}"></div>
    </div>
    <div class="form-group">
      <label for="price">Price:</label>
      <input class="form-control" type="number" th:field="*{price}" id="price" placeholder="price">
      <div class="alert alert-warning" th:if="${#fields.hasErrors('price')}" th:errors="*{price}"></div>
    </div>
    <div class="form-group">
      <label for="category" class="form-label">Category:</label>
      <select th:field="*{category}" class="form-control" id="category">
        <option th:each="category : ${categories}"
          th:value="${category.id}" th:text="${category.name}"></option>
      </select>
      <div class="alert alert-warning" th:if="${#fields.hasErrors('category')}" th:errors="*{category}"></div>
    </div>
    <input type="submit" class="btn btn-success" value="Add Product">
  </form>
</div>
</body>
</html>
```

Thymeleaf

✓ edit

Products

Name
22

Image:


Change Image: Không có tệp nào được chọn

Price:
22

Category:

Description
Description
ProductDetail

Manufacturer

Manufacture Year

```
<div class="form-group">
  <label for="category" class="form-label">Category:</label>
  <select th:field="**{category}" class="form-control" id="category">
    <option th:each="category : ${categories}"
      th:value="${category.id}"
      th:text="${category.name}"
      th:selected="${category.id == product.category.id}"></option>
  </select>
  <div class="alert alert-warning" th:if="${#fields.hasErrors('category')}" th:errors="**{category}"></div>
</div>

<!-- Product Detail Fields -->
<div class="form-group">
  <label for="description">Description</label>
  <textarea class="form-control" id="description" name="description" rows="4" placeholder="Description"
    th:field="**{productDetail.description}"
    th:value="${product.productDetail != null ? product.productDetail.description : 'Enter description here'}">
  </textarea>
  <div class="alert alert-warning" th:if="${#fields.hasErrors('productDetail.description')}" th:errors="**{productDetail.description}"></div>
</div>
<div class="form-group">
  <label for="manufacturer">Manufacturer</label>
  <input class="form-control" type="text" th:field="**{productDetail.manufacturer}"
    th:value="${product.productDetail != null ? product.productDetail.manufacturer : 'Enter manufacturer here'}"
    id="manufacturer" placeholder="Manufacturer">
  <div class="alert alert-warning" th:if="${#fields.hasErrors('productDetail.manufacturer')}"
th:errors="**{productDetail.manufacturer}"></div>
</div>
<div class="form-group">
  <label for="manufactureYear">Manufacture Year</label>
  <input class="form-control" type="number" th:field="**{productDetail.manufactureYear}"
    th:value="${product.productDetail != null ? product.productDetail.manufactureYear : '2023'}"
    id="manufactureYear" placeholder="Manufacture Year">
  <div class="alert alert-warning" th:if="${#fields.hasErrors('productDetail.manufactureYear')}"
th:errors="**{productDetail.manufactureYear}"></div>
</div>
<input type="submit" class="btn btn-success" value="Save Product">
</form>
```

2. Sử dụng ràng buộc (khóa ngoại)

- Các loại quan hệ:

- ❑ 1 - 1

- ❑ 1 – N

- ❑ N – N

- Các loại ánh xạ:

- ❑ **Một chiều** (Unidirectional)

- ❑ **Hai chiều** (bidirectional)

Thể hiện ràng buộc trong JPA

- Sử dụng các Annotation: **@OneToOne** **@OneToMany** **@ManyToOne** **@ManyToMany**

❑ **FetchType**: định nghĩa phương thức lấy các đối tượng liên quan.

Mặc định: EAGER (@ManyToOne, @OneToOne) : Lấy tất cả đối tượng liên quan

LAZY (@ManyToMany, @OneToMany): Không lấy các đối tượng liên quan

❑ **Cascading**: Mỗi quan hệ thực thể thường phụ thuộc vào sự tồn tại của một thực thể khác

CascadeType.ALL: Tương ứng với tất cả các loại cascade

CascadeType.PERSIST: Nếu đối tượng cha được thêm mới, các thực thể phụ thuộc cũng dc tạo mới

CascadeType.MERGE: tương tự cập nhật

CascadeType.REMOVE: tương tự xóa

- **@JoinColumn** vs **MappedBy**

❑ **@JoinColumn**: dùng để chỉ định cột khóa ngoại và ghép một liên kết thực thể.

❑ Thuộc tính **mappedBy** được dùng để định nghĩa bên tham chiếu.

Mối quan hệ giữa các entity

Quan hệ 1-1: Liên kết 1 chiều

● Thực hiện ở product: @OneToOne

@OneToOne(cascade = CascadeType.ALL, fetch = FetchType.LAZY)

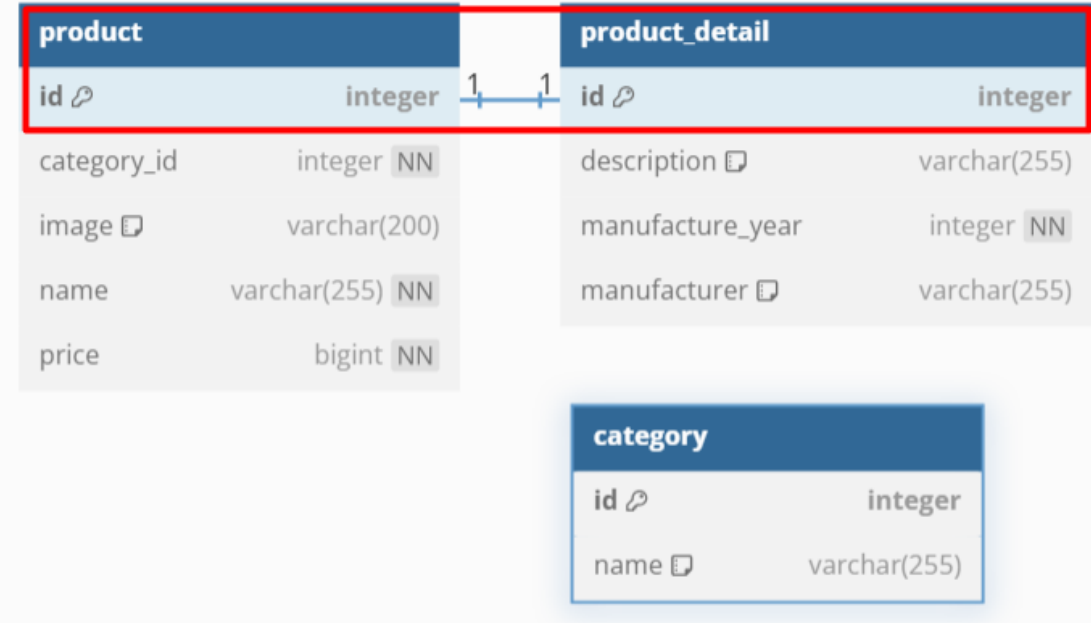
@JoinColumn(name = "detail_id", referencedColumnName = "id")

private ProductDetail productDetail;

```
@Entity
public class Product {
    no usages
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    no usages
    @NotBlank(message = "Tên sản phẩm không được để trống")
    private String name;
    no usages
    @Length(min= 0,max = 200, message = "Tên hình ảnh không quá 200 kí tự")
    private String image;
    no usages
    @NotNull(message = "Giá sản phẩm không được để trống")
    @Min(value = 1, message = "Giá sản phẩm không được nhỏ hơn 1")
    @Max(value = 9999999, message = "Giá sản phẩm không được lớn hơn 9999999")
    private long price;
    no usages
    private int categoryId;

    no usages
    @OneToOne(cascade = CascadeType.ALL, fetch = FetchType.LAZY)
    @JoinColumn(name = "detail_id", referencedColumnName = "id" )
    private ProductDetail productDetail;

    no usages
    @Transient
    private Category category;
}
```



@Getter @Setter @NoArgsConstructor @AllArgsConstructor

@Entity

```
public class ProductDetail {
    no usages
    @Id
    private int id;
    no usages
    private String description;
    no usages
    private int manufactureYear;
    no usages
    private String manufacturer;
    //etc..
}
```




● Productdetail được load tự động khi get, getAll, update.

```
public void update(Product editProduct)
{
    Product find = get(editProduct.getId());
    if(find != null) {
        find.setPrice(editProduct.getPrice());
        find.setName(editProduct.getName());
        if(editProduct.getImage() != null)
            find.setImage(editProduct.getImage());
        productRepository.saveAndFlush(find);
        ProductDetail productDetail =
            editProduct.getProductDetail();
        if(productDetail != null)
        {
            productDetail.setId(find.getId());
            productDetailRepository.save(productDetail);
        }
    }
}
```

```
public List<Product> getAll() {
    List<Product> listProduct = productRepository.findAll();
    for (Product p : listProduct) {
        ProductDetail d = productDetailRepository.findById(p.getId()).orElse( other: null);
        if(d != null)
            p.setProductDetail(d);

        Category c = categoryRepository.findById(p.getCategoryId()).orElse( other: null);
        if(c != null)
            p.setCategory(c);
    }
    return listProduct;
}

blic Product get(int id) {
    Product p = productRepository.findById(id).orElse( other: null);
    if(p != null)
    {
        ProductDetail d = productDetailRepository.findById(p.getId()).orElse( other: null);
        if(d == null)
            d = new ProductDetail();
        p.setProductDetail(d);
        Category c = categoryRepository.findById(p.getCategoryId()).orElse( other: null);
        p.setCategory(c);
    }
}
```

Không cần tìm Productdetail vì đã được load cùng product. Xóa đoạn code tương ứng

Mối quan hệ giữa các entity

Quan hệ 1-1: Liên kết 2 chiều

- Thực hiện ở product: @OneToOne

@OneToOne(cascade = CascadeType.ALL, fetch = FetchType.LAZY))

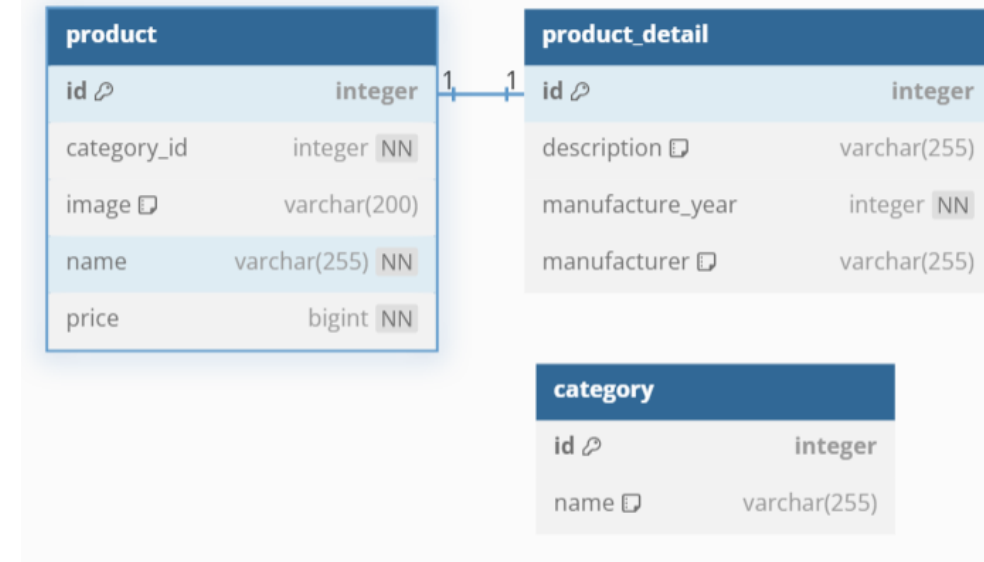
@JoinColumn(name = "id", referencedColumnName = "id")

ProductDetail productDetail;

- Thực hiện ở product_detail:

@OneToOne(mappedBy = "productDetail", fetch = FetchType.LAZY)

private Product product;



Mối quan hệ giữa các entity

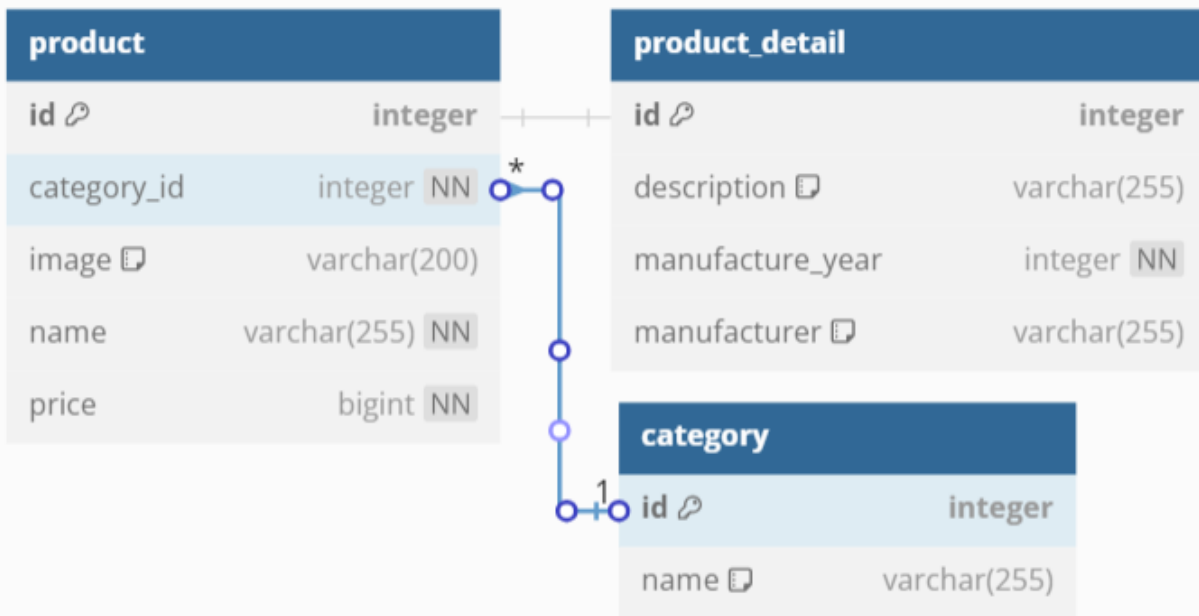
Quan hệ 1-N: Liên kết 1 chiều

● Thực hiện ở Product: @ManyToOne

@ManyToOne(fetch = FetchType.LAZY)

@JoinColumn(name="category_id")

private Category category;



```
@Getter @Setter @NoArgsConstructor @AllArgsConstructor
@Entity
public class Product {
    no usages
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    no usages
    @NotBlank(message = "Tên sản phẩm không được để trống")
    private String name;
    no usages
    @Length(min= 0,max = 200, message = "Tên hình ảnh không quá 200 kí tự")
    private String image;
    no usages
    @NotNull(message = "Giá sản phẩm không được để trống")
    @Min(value = 1, message = "Giá sản phẩm không được nhỏ hơn 1")
    @Max(value = 9999999, message = "Giá sản phẩm không được lớn hơn 9999999")
    private long price;
    no usages
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name="category_id")
    private Category category;
    no usages
    @OneToOne(cascade = CascadeType.ALL, fetch = FetchType.LAZY)
    @JoinColumn(name = "id", referencedColumnName = "id" )
    private ProductDetail productDetail;
}
```



- Thực hiện ở Product: @ManyToOne
- Thay đổi ProductService: tự động lấy Thông tin category

```
@Service
public class ProductService {
    4 usages
    @Autowired
    private ProductRepository productRepository;
    1 usage
    @Autowired
    private ProductDetailRepository productDetailRepository;
    no usages
    @Autowired
    private CategoryRepository categoryRepository;
    1 usage

    public List<Product> getAll() {
        List<Product> listProduct = productRepository.findAll();
        return listProduct;
    }

    public Product get(int id) {
        Product p = productRepository.findById(id).orElse( other: null);
        return p;
    }

    public void add(Product newProduct) {
        Category c = newProduct.getCategory();
        productRepository.save(newProduct);
    }

    1 usage
    public void update(Product editProduct)
    {
        Product find = get(editProduct.getId());
        if(find!= null) {
            find.setPrice(editProduct.getPrice());
            find.setName(editProduct.getName());
            if(editProduct.getImage()!= null)
                find.setImage(editProduct.getImage());
            productRepository.saveAndFlush(find);
            ProductDetail productDetail = editProduct.getProductDetail();
            if(productDetail!= null)
            {
                productDetail.setId(find.getId());
                productDetailRepository.save(productDetail);
            }
        }
    }
}
```

Mối quan hệ giữa các entity

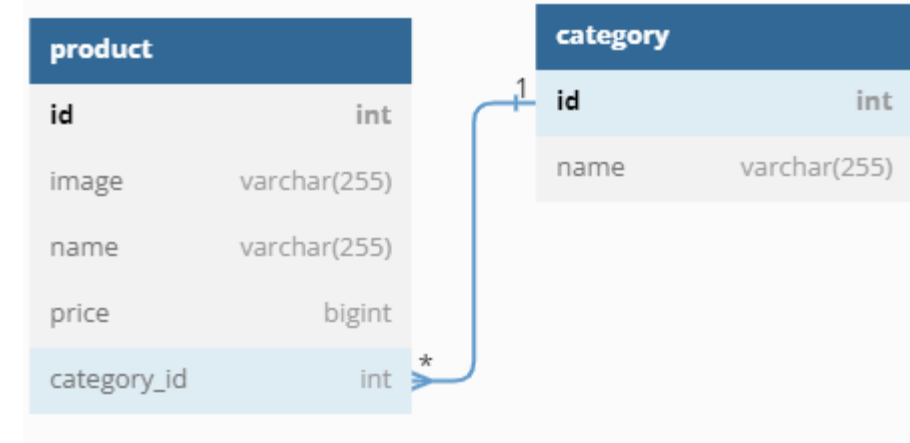
Quan hệ 1-N: Liên kết 2 chiều

- Thực hiện ở Product: @ManyToOne

```
@ManyToOne(fetch = FetchType.LAZY)
@JoinColumn(name="category_id")
private Category category;
```

- Thực hiện ở Category: @OneToMany

```
@OneToMany(mappedBy = "category" , fetch = FetchType.LAZY)
private List<Product> listproducts;
```



Mối quan hệ giữa các entity

Quan hệ N-N

- Có nhiều cách để thể hiện mối quan hệ N-N

TH1: Sử dụng 1 khóa chính:

C1: Ánh xạ 1 chiều @ManyToOne

C2: Ánh xạ 2 chiều: @ManyToOne & @OneToMany

TH2: Sử dụng composite key

C1: Sử dụng @ManyToMany

C2: Sử dụng @ManyToMany (không tạo entity cho UserRole)

User (userid, username, password)

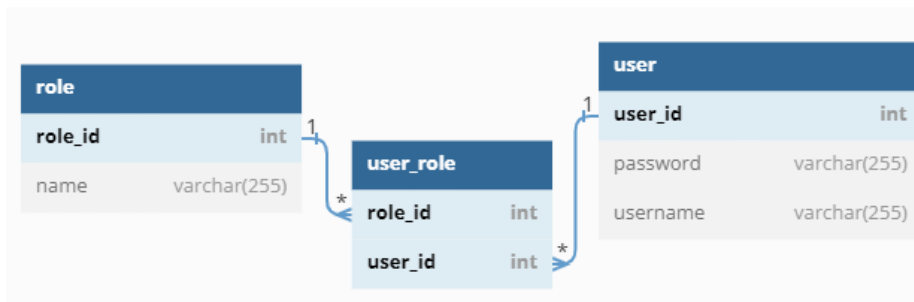
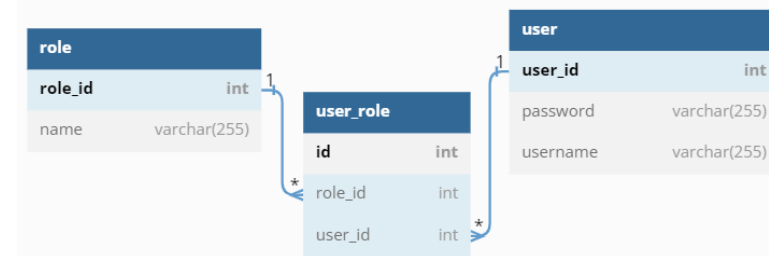
Roles (roleid, name)

TH1: Sử dụng 1 khóa chính

UserRole (id, user_id, user_role)

TH2: Sử dụng composite key

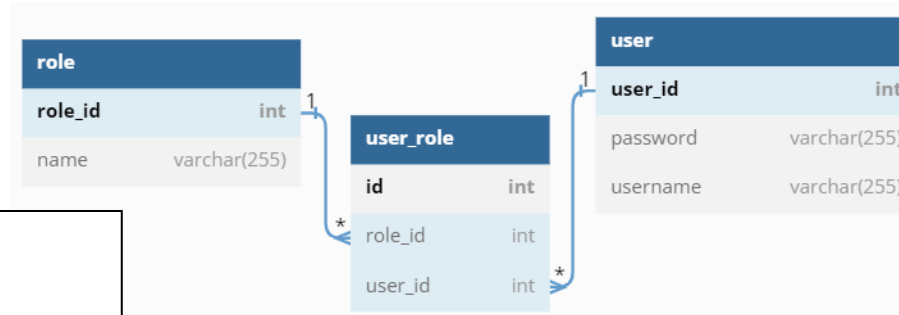
UserRole (user_id, user_role)



Mối quan hệ giữa các entity

Quan hệ N-N: TH1 – single key & ánh xạ 1 chiều

● Sử dụng @ManyToOne ở UserRole



```

@Data
@Entity
@Table(name = "user_role")
public class UserRole {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    @ManyToOne(fetch = FetchType.LAZY, optional = false)
    @JoinColumn(name = "user_id")
    private User user;

    @ManyToOne(fetch = FetchType.LAZY, optional = false)
    @JoinColumn(name = "role_id")
    private Role role;
}
  
```

```

@Data
@Entity
public class User {
    @Id
    @GeneratedValue(strategy =
    GenerationType.IDENTITY)
    private Integer user_id;
    @Column
    private String username;
    @Column
    private String password;
}
  
```

```

@Data
@Entity
public class Role {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer role_id;
    @Column
    private String name;
}
  
```

Mối quan hệ giữa các entity

Quan hệ N-N: TH1 – single key & ánh xạ 2 chiều

- Sử dụng @ManyToOne ở **UserRole**
- Sử dụng thêm @OneToMany ở **Role** và **User**

```
@Data
@Entity
@Table(name = "user_role")
public class UserRole {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    @ManyToOne(fetch = FetchType.LAZY, optional = false)
    @JoinColumn(name = "user_id")
    private User user;

    @ManyToOne(fetch = FetchType.LAZY, optional = false)
    @JoinColumn(name = "role_id")
    private Role role;
}
```

```
@Data
@Entity
public class Role {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer role_id;

    @Column
    private String name;

    @OneToMany(mappedBy = "role", cascade = CascadeType.ALL)
    private Set<UserRole> roles = new HashSet<>();
}
```

```
@Data
@Entity
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer user_id;
    @Column
    private String username;
    @Column
    private String password;

    @OneToMany(mappedBy = "user", cascade = CascadeType.ALL)
    private Set<UserRole> roles = new HashSet<>();
}
```


Mối quan hệ giữa các entity

Quan hệ N-N: TH2 – composite key

- Tạo composite key bằng @Embeddable và triển khai **serializable**

@Embeddable

```
public class UserRoleId implements Serializable {
```

```
    @Column(name = "user_id")
```

```
    private Integer user;
```

```
    @Column(name = "role_id")
```

```
    private Integer role;
```

```
}
```

Mối quan hệ giữa các entity

Quan hệ N-N: TH2 – composite key & ánh xạ 1 chiều

- Sử dụng **@ManyToOne** ánh xạ 1 chiều

@Data

@Entity

@Table(name = "user_role")

```
public class UserRole {
```

```
    @EmbeddedId
```

```
    private UserRoleId id;
```

```
    @ManyToOne(fetch = FetchType.LAZY, optional = false)
```

```
    @MapsId("user_id")
```

```
    @JoinColumn(name = "user_id")
```

```
    private User user;
```

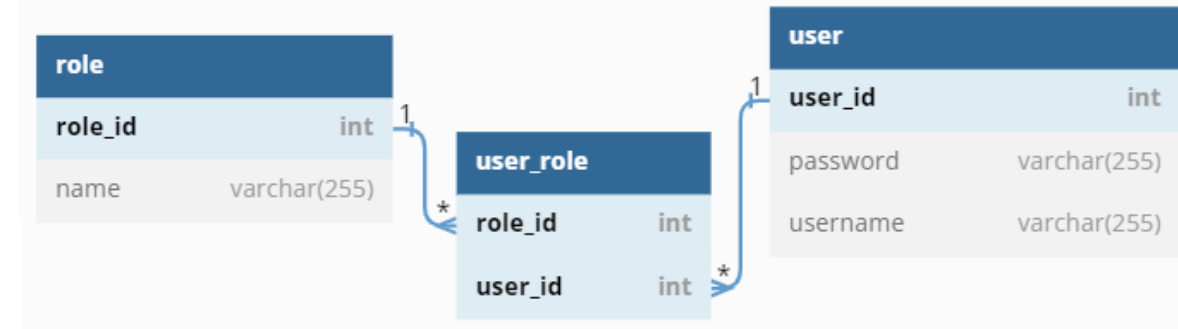
```
    @ManyToOne(fetch = FetchType.LAZY, optional = false)
```

```
    @MapsId("role_id")
```

```
    @JoinColumn(name = "role_id")
```

```
    private Role role;
```

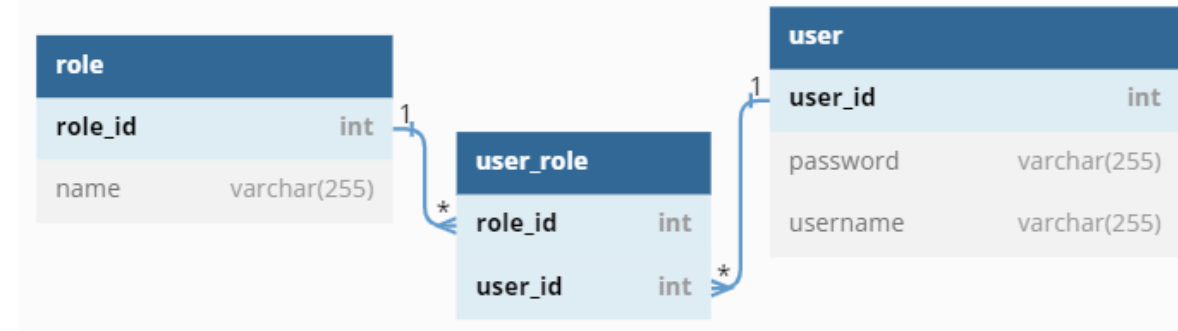
```
}
```



Mối quan hệ giữa các entity

Quan hệ N-N: TH2 – composite key & ánh xạ 2 chiều

- Sử dụng @ManyToOne ở user_role
- Sử dụng @OneToMany



```

@Data
@Entity
public class Role {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer role_id;
    @Column
    private String name;
    @OneToMany(mappedBy = "role", cascade = CascadeType.ALL)
    private List<UserRole> roles;
}
  
```

```

@Data
@Entity
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer user_id;
    @Column
    private String username;
    @Column
    private String password;

    @OneToMany(mappedBy = "user", cascade = CascadeType.ALL)
    private Set<UserRole> roles;
}
  
```

Mối quan hệ giữa các entity

Quan hệ N-N: TH2– composite key & không tạo entity cho UserRole

- Sử dụng **@ManyToMany**
- Không tạo entity cho **UserRole**

```
@Data
@Entity
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer user_id;
    @Column
    private String username;
    @Column
    private String password;
```

```
    @ManyToMany(cascade = CascadeType.ALL)
    @JoinTable(name = "UserRole",
        joinColumns = @JoinColumn(name = "user_id", referencedColumnName = "user_id"),
        inverseJoinColumns = @JoinColumn(name = "role_id", referencedColumnName = "role_id"))
    private Set<Role> roles = new HashSet<>();
}
```

```
@Data
@Entity
public class Role {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer role_id;
    @Column
    private String name;
    @ManyToMany(mappedBy = "roles")
    private Set<User> user = new HashSet<>();
}
```



Thymeleaf + Validation + JPA + MySQL

ASM6: BigSchool - Quản lý khóa học

Viết chương trình quản lý (xem, thêm , xóa, sửa) khóa học

Tạo cơ sở dữ liệu gồm 2 bảng: **Course** và **Category** có mối quan hệ n - 1

- Điền sẵn các category.name như: business, marketing, accountant ...

Basic Options Indexes (2) Foreign keys (1) Check constraints (0) Partitions CREATE code ALTER code

Name: course

Comment:

Columns: + Add - Remove ▲ Up ▼ Down

#	Name	Datatype	Length/Set	Unsigned	Allow N...	Zerofill	Default	Comment	Collation
1	id	INT	10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	AUTO_INCREME...		
2	lecture_name	VARCHAR	200	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No default		utf8mb4_0900_ai_ci
3	place	VARCHAR	255	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No default		utf8mb4_0900_ai_ci
4	start_date	DATETIME	6	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	No default		
5	category_id	INT	10	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL		

Basic Options Indexes (1) Foreign keys (0) Check constraints (0) Partitions CREATE code ALTER code

Name: category

Comment:



Columns: + Add - Remove ▲ Up ▼ Down

#	Name	Datatype	Length/Set	Unsigned	Allow N...	Zerofill	Default	Comment
1	id	INT	10	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	AUTO_INCREME...	
2	name	VARCHAR	255	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL	

ASM6: BigSchool – Danh sách khóa học

Yêu cầu 1: Lấy ds các khóa học (/courses)

- Lấy ds các khóa học từ course với categoryName từ bảng **category**

  localhost:8080/courses

[Home](#) [Add Course](#)

#	Lecture Name	Place	Category Name	Start Date - Time	
1	NGUYEN HUY CUONG	BINH DUONG	Marketing	20/11/2024 08:30	<input type="button" value="Edit"/> <input type="button" value="Delete"/>
2	Nguyen Dinh Anh	Dong Nai	Business	24/12/2024 12:30	<input type="button" value="Edit"/> <input type="button" value="Delete"/>

ASM6: BigSchool – Thêm khóa học mới

Yêu cầu 2: Add Course (Thêm khóa học)

- Category Name được lấy từ bảng category
- Kiểm tra các trường bắt buộc phải nhập

❑ **Start Date** Nhập theo định dạng

dd/MM/yyyy HH:mm

Ví dụ: 10/03/2025 07:30

- **Add Course:**
 - + Lưu thông tin nhập liệu vào CSDL
 - + Redirect về trang home

Home **Add Course** Search Search

Thông tin khóa học

Name
Lecture Name
Tên giảng viên không được để trống

Place:
Place
Nơi học không để được để trống

Start Date:
dd/MM/yyyy HH:mm
Ngày bắt đầu không để trống

Category:
Business
Marketing
Accountant

© Nguyễn Huy Cường - Lập trình ứng dụng với Java

ASM6: BigSchool – Edit Khóa học

Yêu cầu 3: Edit Course (Thay đổi khóa học)

- Khi Edit (Từ Home) điền lại thông tin theo mã khóa
- Kiểm tra các trường bắt buộc phải nhập

❑ **Start Date** Nhập theo định dạng

dd/MM/yyyy HH:mm

Ví dụ: 10/03/2025 07:30

- **Save Course:**

- + Thay đổi thông tin nhập liệu vào CSDL
- + Redirect về trang home

localhost:8080/courses/edit/1

Home Add Course Search Search

Thông tin khóa học

Name

NGUYEN HUY CUONG

Place:

BINH DUONG

Start Date:

20/11/2024 08:30

Category:

Marketing

Save Course

ASM6: BigSchool – Delete Khóa học

Yêu cầu 4: Delete Course (Xóa khóa học)

- Hiển thị confirm message
- OK
 - + Xóa thông tin khóa học
 - + Redirect về trang Home

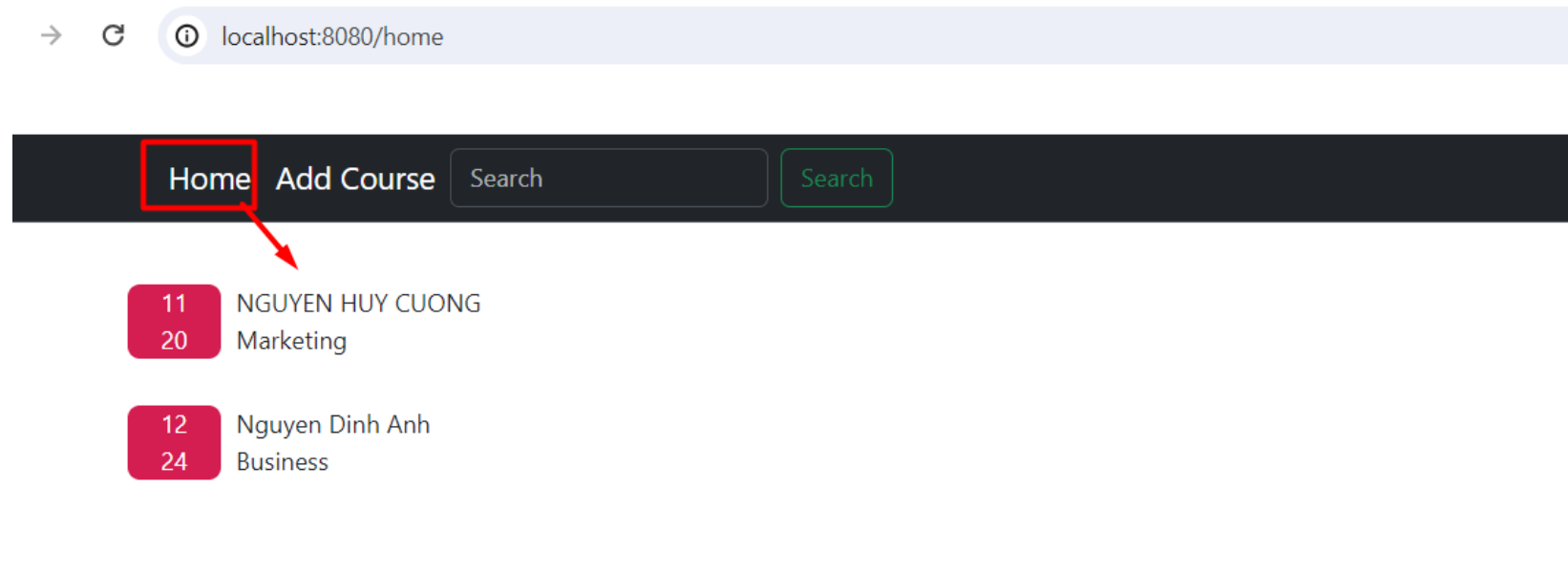
The screenshot shows a web browser at `localhost:8080/courses`. The application has a dark header with 'Home', 'Add Course', and a search bar. A confirmation dialog is open, asking 'Are you sure to delete?' with 'OK' and 'Huỷ' (Cancel) buttons. Below the dialog is a table of courses. The 'Delete' button for the second course is highlighted with a red box, and a red arrow points from the 'OK' button in the dialog to it.

#	Lecture Name	Place	Category Name	Start Date Time	Edit	Delete
1	NGUYEN HUY CUONG	BINH DUONG	Marketing	20/11/2024 08:30	Edit	Delete
2	Nguyen Dinh Anh	Dong Nai	Business	24/12/2024 12:30	Edit	Delete

ASM6: BigSchool – Khóa học sắp tới

Yêu cầu 5: Home – Khóa học sắp tới

- Trang Home: chỉ lấy các khóa học sắp tới (là khóa học có **thời gian bắt đầu > thời gian hiện tại** trong cơ sở dữ liệu)
- Home có giao diện: Chứa thông tin Tháng & ngày học (sắp tới), LectureName và CategoryName

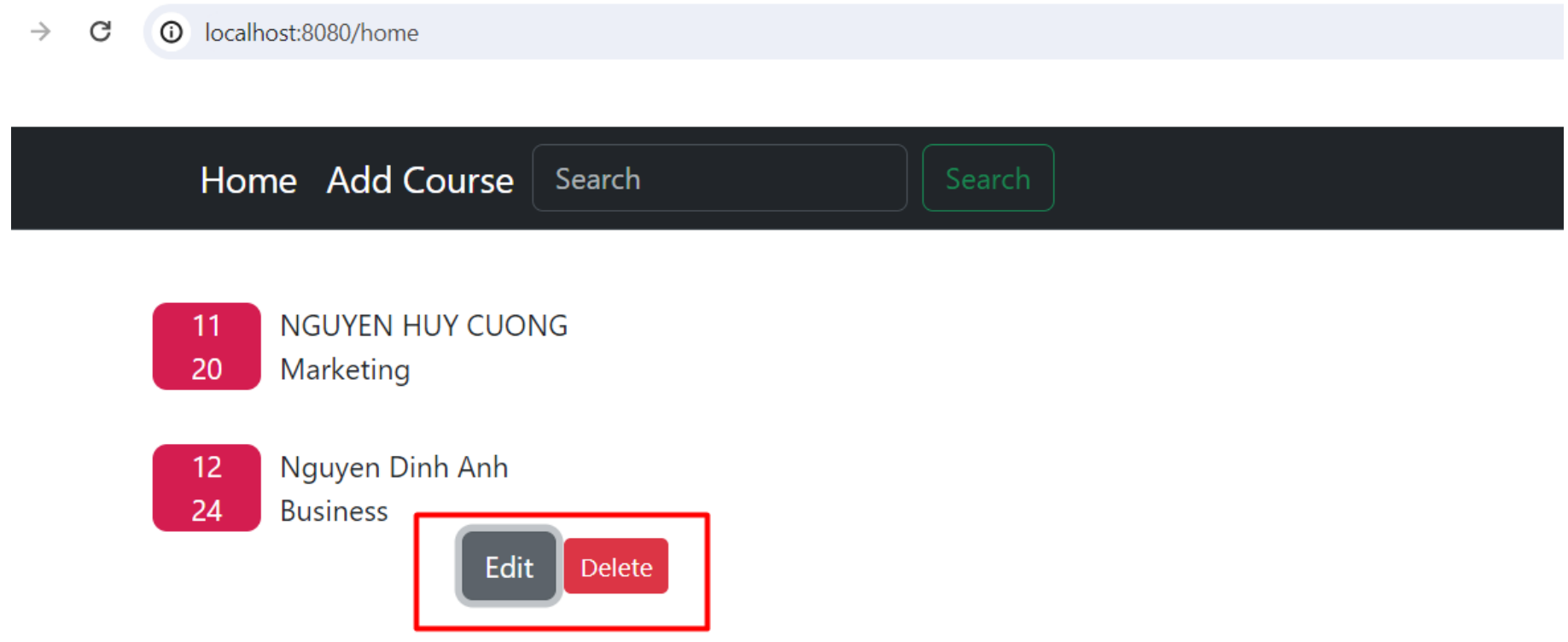


ASM6: BigSchool – Edit, Delete Khóa học

Yêu cầu 6: Home – Hiển thị Edit, Delete khi hover chuột vào các item

Edit: như ở y/c 3

Delete: như y/c 4



ASM6: BigSchool – Tìm kiếm

Yêu cầu 7: Search – Tìm kiếm các khóa học có tên giảng viên chứa kí tự tìm kiếm (không phân biệt hoa thường)

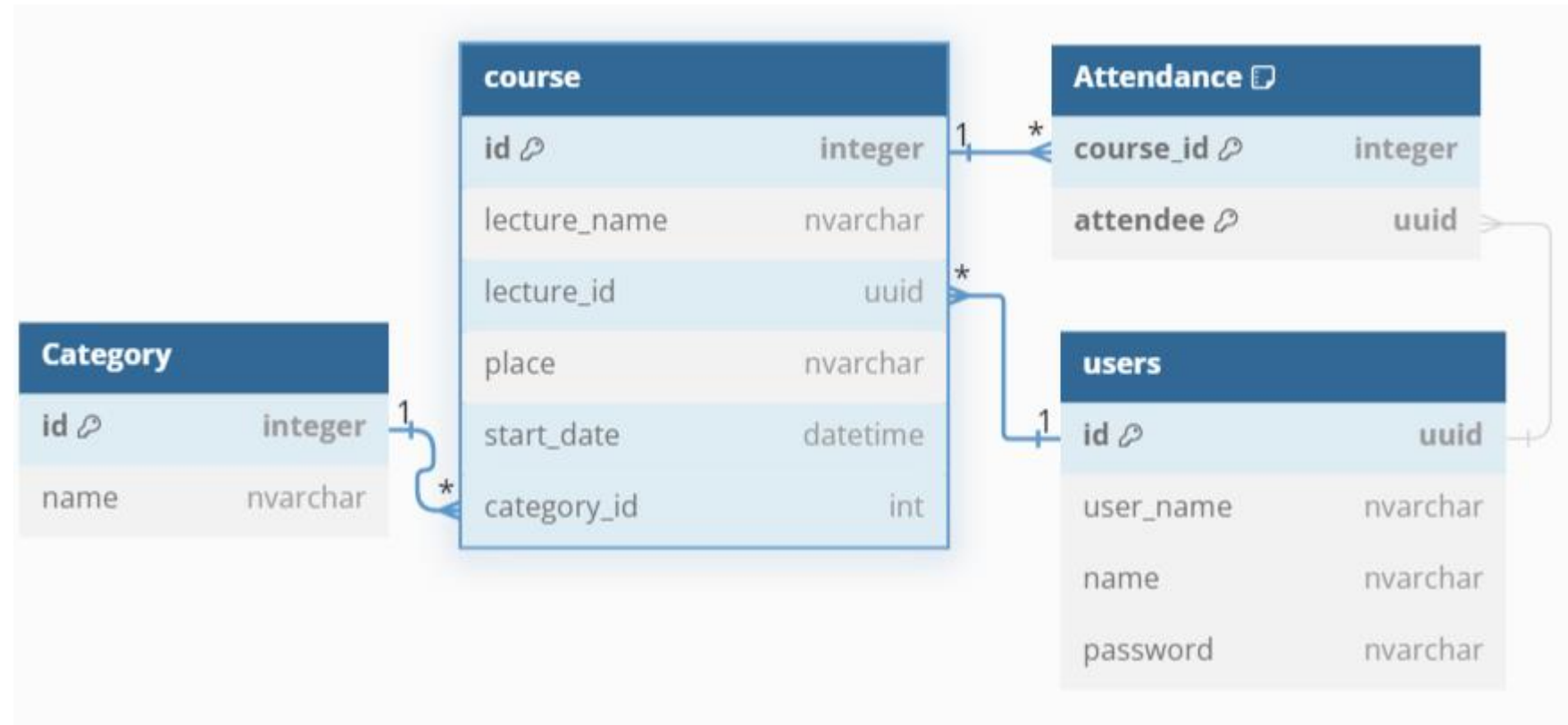
The screenshot shows a dark-themed navigation bar with 'Home' and 'Add Course' links. To the right is a search input field containing the letter 'c', which is highlighted with a red rectangle. A green 'Search' button is to the right of the input field. A red arrow points from the 'Search' button to a search result below. The result consists of a pink rounded square containing the numbers '11' and '20' stacked vertically, followed by the text 'NGUYEN HUY CUONG' and 'Marketing' on two lines.

Search Input	Search Button	Search Result
c	Search	11 20 NGUYEN HUY CUONG Marketing

ASM6: BigSchool – tạo users và attendance

Yêu cầu 8: Bổ sung bảng **users**, **attendance** – Thông tin users và tham dự)

- Thực hiện các entity cho users và attendance tương ứng (chú ý: attendance chứa composite key là **course_id** và **attendee**)



ASM6: BigSchool – Đăng ký tài khoản

Yêu cầu 9: Đăng ký tài khoản, lưu thông tin đăng ký cho user

- Hiển thị thông tin đăng ký
- Validate các dữ liệu
- Khi register -> lưu

Thông tin vào bảng user

- Khi thành công -> trở về trang addcourse

Home Add Course Search Search Register

Đăng ký tài khoản

Username
cuong719

Email
nh.cuong@hutech.edu.vn

Password
.....

Name
NGUYEN HUY CUONG

Register

Database filter Table filter

HuyCuong Database: bigschool2 Table: users Data Query

bigschool2.users: 2 rows total (exact)

#	id	email	name	user_name
1	z]0'□àC6£q□□†Y3^	nd.anh@gmail.com	NGUYEN DINH ANH	nguyendinhanh
2	Đ□Q'H□O□□¶ä6¼%ook	nh.cuong@hutech.edu.vn	NGUYEN HUY CUONG	cuong719

Database filter Table filter

HuyCuong Database: bigschool2 Table: users Data Query

bigschool2.users: 2 rows total (exact)

#	id	email	name	user_name
1	z]0'□àC6£q□□†Y3^	nd.anh@gmail.com	NGUYEN DINH ANH	nguyendinhanh
2	Đ□Q'H□O□□¶ä6¼%ook	nh.cuong@hutech.edu.vn	NGUYEN HUY CUONG	cuong719

Yêu cầu 10: Lấy thông tin giảng viên khi đăng kí từ Name của user.Name

Thông tin khóa học

© Nguyễn Huy Cường - Lập trình ứng dụng với Java

The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'Database filter' pane shows the 'bigschool2' database selected. The 'Table filter' pane shows the 'users' table selected. The main pane displays the 'bigschool2.users' table with 2 rows total. The columns are 'id', 'email', 'name', and 'user_name'. The first row is for 'nguyendinhnh' and the second row is for 'cuong719'.

#	id	email	name	user_name
1	z]0'□ac6εq□□+Y3^	nd.anh@gmail.com	NGUYEN DINH ANH	nguyendinhnh
2	Đ□Q'H□O□¶¶đ0%4%ok	nh.cuong@hutech.edu.vn	NGUYEN HUY CUONG	cuong719

Q&A

Thank you!