Curso: Sistemas de Informação	
Disciplina: Mapemento Objeto Relacional	**Ciclo: 8**
Professor: Luiz Gustavo Dias	**Tipo:** Material Complementar
Objetivo: Consulta, Delete e Update	

Pré-requisito: necessário ter uma aplicação Java implementada com o uso de JPA com Maven e Hibernate e objetos persistidos em banco de dados.

REALIZANDO CONSULTAS SIMPLES

A realização de consultas simples em um projeto Java com Maven e Hibernate, que faça uso do EntityManager, se dá através do método 'find'.

Esse método deve ser executado a partir de um objeto EntityManager, assim como utilizamos o método 'persist' em uma etapa anterior.

Na chamada do método é necessário passar como parâmetro dois componentes:

a) Tabela.class: corresponde à tabela a ser consultada;
b) Variável que se deseja localizar: corresponde ao valor atribuído à pk da tupla.

Exemplo:

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

em.find(Produto.class, 1);

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

Para que o resultado da busca seja impresso você pode passar o método find como parâmetro do sysout. Dessa forma, o registro da tupla será impresso integralmente:

```
//imprime o protudo integralmente
System.out.println(em.find(Produto.class, 1));
```

Para que o resultado da consulta esteja disponível em tempo de execução do software é necessário instanciar um objeto correspondente ao tipo de registro consultado e utiliza-lo para armazenar o resultado:

```
//armazena toda a tupla em uma variável do tipo produto
Produto x = em.find(Produto.class, 1);
```

Após armazenado, pode-se acessar atributos específicos para realizar outros procedimentos, como a impressão na tela apenas do nome do produto (considerando o exemplo deste conteúdo):

```
//imprime um atributo específico
System.out.println("Nome: "+ x.getNome());
```

Utilizando o Scanner

Em um contexto mais amplo, é possível utilizar a classe Scanner para que seja promovida interação com o usuário no momento da consulta de um objeto.

```
EntityManagerFactory emf = Persistence.createEntityManagerFactory("exemplo-jpa");
EntityManager em = emf.createEntityManager();

Scanner scanner = new Scanner(System.in);

System.out.println("Informe o id do produto: ");
Integer id = scanner.nextInt();

em.getTransaction().begin();

//imprime o protudo integralmente
System.out.println(em.find(Produto.class, id));

//armazena toda a tupla em uma variável do tipo produto
Produto x = em.find(Produto.class, id);

//imprime um atributo específico
System.out.println("Nome: "+ x.getNome());

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

REMOVENDO REGISTROS

A remoção de registros ocorre através do método ‘remove’ do EntityManager. Sua aplicação é similar ao método ‘persist’, devendo ser passado um objeto existente para que seja removido da tabela.

Consideremos o exemplo anterior, no qual armazenamos uma tupla consultada na variável x do tipo Produto.

Suponhamos que desejamos excluir este registro:

```
//remover um registro (descomentar)
em.remove(x);

em.getTransaction().commit();

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

Oberve que, além de utilizar o ‘remove’ é necessário utilizar o ‘getTransaction().commit()’ para proceder com a remoção do registro, pois estamos realizando uma modificação no banco de dados.

Por que não foi preciso usar o ‘getTransaction().commit()’ ao proceder com o find?

Uma simples consulta ao banco não promove modificações, portanto, não há alteração a ser confirmada.

ATUALIZANDO REGISTROS

A atualização de registros no banco ocorre através do método ‘merge’ do EntityManager. O trecho abaixo representa o processo conforme o exemplo de aula:

```
Scanner scanner = new Scanner(System.in);
System.out.println("Informe o id do produto: ");
Integer id = scanner.nextInt();

Produto produto = em.find(Produto.class, id);

if (produto != null) {


    System.out.print("Novo nome: ");
    scanner.nextLine();
    String novoNome = scanner.nextLine();

    System.out.print("Nova categoria: ");
    String novaCategoriaStr = scanner.nextLine();

    try {
        int novaCategoria = Integer.parseInt(novaCategoriaStr);
        produto.setNome(novoNome);
        produto.setCategoria(novaCategoria);

        em.merge(produto);

        em.getTransaction().commit();
        System.out.println("Produto atualizado com sucesso.");
    } catch (NumberFormatException e) {
        System.out.println("A entrada para a categoria não é um número válido.");
    }
} else {
    System.out.println("Produto não encontrado.");
}

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

Nele, estou utilizando o Scanner para promover a interação com o usuário.

O exemplo promove a alteração de ‘nome’ e ‘categoria do produto, mas poderia ser aplicado para apenas um dos atributos.

Fiz um tratamento condicional para que só proceda com as instruções de atualização caso seja informado um produto real pelo usuário.

Sendo real, o usuário promoverá o novo nome e a nova categoria e, em seguida, o sistema entratrá em um tratamento de exceção do tipo try catch para proceder com a atualização.

Por estar utilizando um atributo do tipo Integer é necessário a conversão para o tipo int para que seja atualizado no banco de dados – isso é feito através do método parseInt.

Os métodos ‘set’ fazem passam os novos valores gravados para os atributos e o ‘merge’ atualiza o banco de dados.