Aluno: 330230 – Danimar Henrique Varisa

SQL / CRUD com JDBC e DAO

1. Utilize o MySQL Workbench e crie o banco de dados chamado unoesc\_trabalho\_CRUD. Selecione o banco com o comando USE. Cole abaixo o comando utilizado.

CREATE DATABASE unoesc\_trabalho\_CRUD;

USE unoesc\_trabalho\_CRUD;

1. Use o comando CREATE TABLE para criar a tabela chamada pessoa com a estrutura a seguir. Cole abaixo o comando utilizado.

|  |  |
| --- | --- |
| Nome | Tipo |
| id\_pessoa | INT AUTO\_INCREMENT PRIMARY KEY |
| nome\_pessoa | VARCHAR(50) NOT NULL |
| data\_nascimento | DATE NOT NULL |
| salario | DECIMAL(12, 2) NOT NULL CHECK(salario >= 0) |

CREATE TABLE pessoa (

    id\_pessoa INT(11) AUTO\_INCREMENT PRIMARY KEY,

    nome\_pessoa VARCHAR(50) NOT NULL,

    data\_nascimento DATE NOT NULL,

    salario DECIMAL(12, 2) NOT NULL *CHECK*(salario >= 0)

);

1. Utilize o comando INSERT para inserir registros nesta tabela da seguinte forma, colando um exemplo de cada comando abaixo.
   1. Insira alguns registros usando a forma do comando INSERT que passa todos os valores mas sem especificar os campos. Para o campo id\_pessoa passe o valor nulo de forma que o próprio MySQL gere o código.

INSERT INTO pessoa VALUES (null, "Danimar Henrique Varisa", '1990-05-02', 8900.00);

* 1. Insira alguns registros usando a forma do comando INSERT que especifica uma lista de campos. Use a seguinte lista de campos (nome\_pessoa, data\_nascimento, salario).

INSERT INTO pessoa (nome\_pessoa, data\_nascimento, salario) VALUES ("Jaqueline Candiago de Oliveira", '1990-07-16', 6850.00);

INSERT INTO pessoa (nome\_pessoa, data\_nascimento, salario) VALUES ("Daniel Varisa", '1967-09-09', 11000.00);

INSERT INTO pessoa (nome\_pessoa, data\_nascimento, salario) VALUES ("Diego de Melo Varisa", '2010-10-10', 900.00);

INSERT INTO pessoa (nome\_pessoa, data\_nascimento, salario) VALUES ("Duany Henrique Varisa", '1993-10-20',1500.00);

1. Execute o comando SELECT para listar os dados da tabela em ordem alfabética crescente por nome, depois em ordem de data de nascimento e depois por salário. Cole abaixo os comandos.

SELECT \* FROM pessoa ORDER BY nome\_pessoa ASC;

SELECT \* FROM pessoa ORDER BY data\_nascimento ASC;

SELECT \* FROM pessoa ORDER BY salario ASC;

1. Modifique um registro utilizando o comando UPDATE. NÃO ESQUEÇA de utilizar a cláusula WHERE para indicar qual registro será modificado. Cole abaixo o comando.

UPDATE pessoa SET salario = 1800.00 WHERE id\_pessoa = 4;

1. Com o comando DELETE remova o último registro inserido. NÃO ESQUEÇA de utilizar a cláusula WHERE para indicar qual registro será modificado. Cole abaixo o comando.

DELETE FROM pessoa WHERE id\_pessoa = 5;

1. Faça uma consulta que mostre os nomes com respectivas datas de nascimento. Use a função date\_format() com a máscara '%d de %M de %Y' para formatar a saída. Dê o apelido de ‘data de nascimento’ ao campo. Modifique a ‘localização’ do sistema para português do Brasil com o comando SET lc\_time\_names='pt\_BR' para a consulta mostrar as datas em português. Cole abaixo os comandos.

SET lc\_time\_names='pt\_BR';

SELECT nome\_pessoa, date\_format(data\_nascimento, '%d de %M de %Y') AS 'Data de nascimento' FROM pessoa;

1. Utilize a função count() para retornar o número de registros que existem na tabela. Cole abaixo o comando.

SELECT COUNT(\*) AS 'Qtde pessoas' FROM pessoa;

1. Faça uma consulta que retorne o salário mais alto e mais baixo. Cole abaixo o comando.

SELECT MAX(salario), MAX(salario) FROM pessoa;

1. Execute uma consulta que retorne a média e o somatório dos salários. Cole abaixo o comando.

SELECT AVG(salario), SUM(salario) FROM pessoa;

1. Siga a apresentação ‘aula 38c - CRUD com JDBC e DAO (26.08.2022)’ e refaça o CRUD (DAO – *interface* e implementação, entidade de domínio e programa principal) mas dessa vez utilizando a tabela pessoa   
   criada no início desta lista de exercícios.
2. Desafio: Faça a versão interativa do CRUD.
   1. O menu deve seguir o modelo abaixo, adaptando-o para o cadastro de pessoas:

Texto

Descrição gerada automaticamente

* 1. Estrutura do programa (classe Principal):

Atributos como DAO da entidade pessoa, objetos SimpleDateFormat e NumberFormat.

**static** **private** Scanner *sc* = **new** Scanner(System.***in***);

Construtor:

* Inicializar DAO.
* Inicializar objetos SimpleDateFormat e NumberFormat.

Método void adicionar():

* Data deve ser definida automaticamente pelo sistema e não pelo usuário.

java.sql.Date dataCadastro = Date.*valueOf*(LocalDate.*now*());

* Solicitar confirmação do usuário e em caso afirmativo instanciar um objeto Pessoa e usar o objeto DAO para adicioná-la no banco de dados.

Método void alterar():

* Executar o método solicitarPessoa(). Se o resultado foi nulo sair do método, caso contrário entrar na tela de alteração, conforme modelo abaixo.

Texto

Descrição gerada automaticamente

* O usuário deve poder escolher qual campo modificar. Use um switch-case para testar a opção e solicitar o campo correto.
* No caso da data de cadastro o código abaixo pode ser utilizado para converter uma *string* em um tipo java.sql.Date e gerar uma exceção se for entrada uma data em formato inválido.

**try** {

**long** dataHora = fd.parse(dataCadastro).getTime();

prod.setDataCadastro(**new** java.sql.Date(dataHora));

} **catch** (ParseException e) {

System.***out***.println("Formato inválido! Formato é dd/mm/aaaa");

}

* Ao final das edições solicitar confirmação do usuário e em caso afirmativo usar o objeto DAO para alterar o registro no banco de dados.

Método void excluir():

* Executar o método solicitarPessoa(). Se o resultado foi nulo sair do método.
* Caso o registro tenha sido encontrado, mostrá-lo na tela utilizando para isso o método mostrarPessoa().
* Solicitar confirmação ao usuário para excluir o registro. Em caso afirmativo usar o objeto DAO para excluir o registro no banco de dados.

Método void consultar():

* Executar o método solicitarPessoa(). Se o resultado foi nulo sair do método.
* Caso o registro tenha sido encontrado, mostrá-lo na tela utilizando para isso o método mostrarPessoa().

Método listar():

* Crie uma lista que receba os dados do método listarTodos() da DAO.
* Execute um laço de repetição (**for** (Pessoa p: lista)) e dentro do laço chame o método mostrarProduto() para cada objeto p.

Método Pessoa solicitarPessoa():

* Solicita ao usuário o código da pessoa conforme modelo abaixo.



* Executa o método buscarPorId() da classe DAO para retornar uma pessoa ou nulo.

Método mostrarPessoa(Pessoa p):

* Mostra os dados da pessoa, passada no parâmetro p, semelhante ao modelo abaixo.

Uma imagem contendo Tabela

Descrição gerada automaticamente

Método main():

* Montar menu, programa só deve encerrar ao escolher a opção 6.
* Tratar a exceção de *runtime*.

1. Desafio: Faça agora a versão com *interface* gráfica utilizando Swing.