Prova II

Disciplina: Banco de Dados Relacional

Período: 4°

Valor da Prova: 08 Pontos

Professor: Diego Ramos Inácio

Curso de Engenharia de Software

Município: Saquarema – RJ

Aluno:_____

Matricula:_____

Definições:

1 – Modelagem + Criação de usuário com permissão apenas de leitura (2 pontos)

Abaixo está a **modelagem já pronta** que o aluno deverá implementar. O aluno **não pode alterar o modelo**, apenas **rodar o SQL** criando as tabelas conforme especificado, além de **criar um usuário somente-leitura**.

Modelo lógico fornecido (fixo para o aluno)

Tabelas:

1. **departamentos**
    - id_departamento (PK)
    - nome
2. **funcionarios**
    - id_funcionario (PK)
    - nome
    - cargo
    - salario
    - id_departamento (FK → departamentos)
3. **projetos**

- id_projeto (PK)
- nome
- descricao
- id_departamento (FK → departamentos)

4. **log_projetos**
   - id_log (PK)
   - id_projeto
   - operacao
   - data_operacao

Tarefa do aluno

1. Criar **todas as tabelas conforme o modelo acima** com PK, FK e NOT NULL onde necessário.
2. Criar um **usuário somente-leitura**, seguindo as regras:
   - Deve ter permissão **apenas de SELECT** em todas as tabelas criadas.
   - Nome do usuário deve seguir o padrão: **usuario_read_nomeDoAluno**
   - A senha pode ser definida pelo aluno.

SQL:

```
-- ============================

-- TABELA DEPARTAMENTOS

-- ============================

CREATE TABLE departamentos (

   id_departamento SERIAL PRIMARY KEY,

   nome VARCHAR(255) NOT NULL

);


-- ============================

-- TABELA FUNCIONARIOS

-- ============================

CREATE TABLE funcionarios (

   id_funcionario SERIAL PRIMARY KEY,

   nome VARCHAR(255) NOT NULL,

   cargo VARCHAR(255) NOT NULL,

   salario NUMERIC(10,2) NOT NULL,
```

```
    id_departamento INT NOT NULL REFERENCES departamentos(id_departamento)
);


-- ============================
-- TABELA PROJETOS
-- ============================
CREATE TABLE projetos (
    id_projeto SERIAL PRIMARY KEY,
    nome VARCHAR(255) NOT NULL,
    descricao TEXT,
    id_departamento INT NOT NULL REFERENCES departamentos(id_departamento)
);


-- ============================
-- TABELA DE LOGS DE PROJETOS
-- ============================
CREATE TABLE log_projetos (
    id_log SERIAL PRIMARY KEY,
    id_projeto INT NOT NULL,
    operacao VARCHAR(50) NOT NULL,
    data_operacao TIMESTAMP NOT NULL
);
```

Questão 2 – Trigger (3 pontos)

A seguir está um **modelo incompleto** de trigger **(NÃO É A TRIGGER FINAL)**. O aluno deve utilizar este modelo como referência, mas **construir a trigger correta** para registrar operações **de INSERT** realizadas na tabela **projetos**, gravando na tabela **log_projetos**.

Modelo entregue (exemplo base)

Atenção: este modelo está **incompleto** propositalmente. O aluno deve montar a versão correta.

```
CREATE OR REPLACE FUNCTION modelo_trigger()
RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO log_projetos(id_projeto, operacao, data_operacao)
```

```
    VALUES (NEW.id_projeto, 'OPERACAO_AQUI', NOW());

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_modelo
AFTER INSERT ON projetos
FOR EACH ROW
EXECUTE FUNCTION modelo_trigger();
```

Tarefa do aluno

Criar **sua própria função e trigger**, corrigindo:

- Nome adequado para a função e trigger
- Registrar a operação como 'INSERT'
- Garantir que os dados inseridos sejam capturados de NEW
- A trigger deve atuar **apenas para INSERT**

Questão 3 – VIEW (1 ponto)

Você deve criar **1 VIEW**, conforme a descrição abaixo, e deve também utilizar o **modelo sugerido**.

A VIEW deve conter:

Nome sugerido: **vw_funcionario_projetos**

Colunas obrigatórias:

- nome_do_funcionario
- cargo
- departamento
- projeto
- descricao_projeto

Modelo de VIEW incompleto (aluno deve ajustar e montar a dele):

```
CREATE VIEW modelo_view AS
SELECT
    f.nome,
    f.cargo,
```

```
    d.nome,
    p.nome
FROM funcionarios f
JOIN departamentos d ON f.id_departamento = d.id_departamento
JOIN projetos p ON p.id_departamento = d.id_departamento;
```

Tarefa do aluno

1. Criar a VIEW com o nome correto.
2. Adaptar o SELECT incluindo **TODAS as colunas exigidas**.
3. Garantir que a VIEW utilize **JOINs adequados**.

Questão 4 – TKINTER (2 pontos)

Abaixo está um **modelo de interface Tkinter incompleto**.
O aluno deve escrever o código completo, adicionando:

- Campos para **usuário**, **senha**, **nome do banco**
- Um campo para digitar um comando SQL
- Um botão "Executar"
- A execução deve mostrar o resultado em um Text ou Label

Modelo inicial (aluno deve completar):

Código:

O aluno deve completar o código, preenchendo as partes faltantes e implementando a lógica SQL.

```
import tkinter as tk
from tkinter import messagebox
import psycopg2

# ----------------------------
# Função principal para executar SQL livre
# ----------------------------
def executar_sql():
    try:
        usuario = entry_user.get()
        senha = entry_pass.get()
```

```
        banco = entry_db.get()
        comando = entry_sql.get("1.0", "end")

        conn = psycopg2.connect(
            dbname=banco,
            user=usuario,
            password=senha,
            host="localhost"
        )
        cur = conn.cursor()
        cur.execute(comando)

        if comando.strip().lower().startswith("select"):
            dados = cur.fetchall()
            texto = "\n".join([str(linha) for linha in dados])
            messagebox.showinfo("Resultado", texto)
        else:
            conn.commit()
            messagebox.showinfo("Sucesso", "Comando executado!")

    except Exception as e:
        messagebox.showerror("Erro", f"Ocorreu um erro: {e}")

# ----------------------------
# CADASTRAR DEPARTAMENTO
# ----------------------------
def cadastrar_departamento():
    try:
        usuario = entry_user.get()
        senha = entry_pass.get()
        banco = entry_db.get()
```

```
        nome_dep = entry_dep_nome.get()

        conn = psycopg2.connect(
            dbname=banco,
            user=usuario,
            password=senha,
            host=host
        )
        cur = conn.cursor()

        cur.execute("""
            INSERT INTO departamentos (nome)
            VALUES (%s)
        """, (nome_dep,))

        conn.commit()
        messagebox.showinfo("Cadastro", "Departamento cadastrado!")

    except Exception as e:
        messagebox.showerror("Erro", f"{e}")

# ----------------------------
# CADASTRAR FUNCIONÁRIO
# ----------------------------
def cadastrar_funcionario():
    try:
        usuario = entry_user.get()
        senha = entry_pass.get()
        banco = entry_db.get()

        nome = entry_fun_nome.get()
        cargo = entry_fun_cargo.get()
```

```
        salario = entry_fun_salario.get()
        id_dep = entry_fun_dep.get()

        conn = psycopg2.connect(
            dbname=banco,
            user=usuario,
            password=senha,
            host=host
        )
        cur = conn.cursor()

        cur.execute("""
            INSERT INTO funcionarios (nome, cargo, salario, id_departamento)
            VALUES (%s, %s, %s, %s)
        """, (nome, cargo, salario, id_dep))

        conn.commit()
        messagebox.showinfo("Cadastro", "Funcionário cadastrado!")

    except Exception as e:
        messagebox.showerror("Erro", f"{e}")

# ----------------------------
# CADASTRAR PROJETO
# ----------------------------
def cadastrar_projeto():
    try:
        usuario = entry_user.get()
        senha = entry_pass.get()
        banco = entry_db.get()

        nome = entry_proj_nome.get()
```

```
        descricao = entry_proj_desc.get()
        id_dep = entry_proj_dep.get()

        conn = psycopg2.connect(
            dbname=banco,
            user=usuario,
            password=senha,
            host=host
        )
        cur = conn.cursor()

        cur.execute("""
            INSERT INTO projetos (nome, descricao, id_departamento)
            VALUES (%s, %s, %s)
        """, (nome, descricao, id_dep))

        conn.commit()
        messagebox.showinfo("Cadastro", "Projeto cadastrado!")

    except Exception as e:
        messagebox.showerror("Erro", f"{e}")

# ----------------------------
# INTERFACE TKINTER
# ----------------------------
janela = tk.Tk()
janela.title("Gerenciador de Banco – PostgreSQL")

# AUTENTICAÇÃO
tk.Label(janela, text="Usuário:").grid(row=0, column=0)
entry_user = tk.Entry(janela)
entry_user.grid(row=0, column=1)
```

```
tk.Label(janela, text="Senha:").grid(row=1, column=0)
entry_pass = tk.Entry(janela, show="*")
entry_pass.grid(row=1, column=1)

tk.Label(janela, text="Banco:").grid(row=2, column=0)
entry_db = tk.Entry(janela)
entry_db.grid(row=2, column=1)

# EXECUTOR SQL
tk.Label(janela, text="SQL Livre:").grid(row=3, column=0)
entry_sql = tk.Text(janela, height=4, width=45)
entry_sql.grid(row=3, column=1)

tk.Button(janela, text="Executar SQL", command=executar_sql).grid(row=4, column=1)

# ------------------------------
# SEÇÃO: CADASTRO DE DEPARTAMENTO
# ------------------------------
tk.Label(janela, text="--- Cadastrar Departamento ---").grid(row=5, column=0, columnspan=2)

tk.Label(janela, text="Nome do Departamento:").grid(row=6, column=0)
entry_dep_nome = tk.Entry(janela)
entry_dep_nome.grid(row=6, column=1)

tk.Button(janela, text="Cadastrar Departamento", command=cadastrar_departamento).grid(row=7,
column=1)

# ------------------------------
# SEÇÃO: CADASTRO DE FUNCIONÁRIOS
# ------------------------------
tk.Label(janela, text="--- Cadastrar Funcionário ---").grid(row=8, column=0, columnspan=2)
```

```
tk.Label(janela, text="Nome:").grid(row=9, column=0)
entry_fun_nome = tk.Entry(janela)
entry_fun_nome.grid(row=9, column=1)

tk.Label(janela, text="Cargo:").grid(row=10, column=0)
entry_fun_cargo = tk.Entry(janela)
entry_fun_cargo.grid(row=10, column=1)

tk.Label(janela, text="Salário:").grid(row=11, column=0)
entry_fun_salario = tk.Entry(janela)
entry_fun_salario.grid(row=11, column=1)

tk.Label(janela, text="ID Departamento:").grid(row=12, column=0)
entry_fun_dep = tk.Entry(janela)
entry_fun_dep.grid(row=12, column=1)

tk.Button(janela, text="Cadastrar Funcionário", command=cadastrar_funcionario).grid(row=13, column=1)

# ------------------------------
# SEÇÃO: CADASTRO DE PROJETOS
# ------------------------------
tk.Label(janela, text="--- Cadastrar Projeto ---").grid(row=14, column=0, columnspan=2)

tk.Label(janela, text="Nome Projeto:").grid(row=15, column=0)
entry_proj_nome = tk.Entry(janela)
entry_proj_nome.grid(row=15, column=1)

tk.Label(janela, text="Descrição:").grid(row=16, column=0)
entry_proj_desc = tk.Entry(janela)
entry_proj_desc.grid(row=16, column=1)
```

```
tk.Label(janela, text="ID Departamento:").grid(row=17, column=0)
entry_proj_dep = tk.Entry(janela)
entry_proj_dep.grid(row=17, column=1)

tk.Button(janela, text="Cadastrar Projeto", command=cadastrar_projeto).grid(row=18, column=1)

janela.mainloop()
```

Tarefa do aluno

Completar a interface, incluindo:

1. Entradas:
    - Em todas as tabelas
2. Ler a tabela indicada