.Ícone

Descrição gerada automaticamente

Faculdade de Informática e Administração Paulista

DATABASE APPLICATION & DATA SCIENCE

1º CheckPoint – 2º Semestre

RM 93613 – Vinicius de oliveira

RM: 94311 - MARCOS BILOBRAM

RM: 96320 - NATHÁLIA MAIA

RM: 94972 - RAFAELA DA SILVA

Sumário

[1 - O código fonte atualizado do projeto (Python) 3](#_Toc145623912)

[2 - Os códigos fontes dos procedimentos criados 10](#_Toc145623913)

[3 - O código fonte da Package e Package Body criadas 13](#_Toc145623914)

[4 - O Código fonte do Trigger criado 16](#_Toc145623916)

[5 - Prints da evidência do funcionamento de cada uma das ações propostas no script do Python 19](#_Toc145623917)

Link do projeto no GitHub: <https://github.com/ViniOlr/CheckPoint1-2Semestre-DBA>

## 1 - O código fonte atualizado do projeto (Python)

# ============================================================================ #

#                                                                              #

# Antes de executar o Script Python, instale as Bibliotecas necessárias        #

# Execute no Terminal:                                                         #

#                                                                              #

# pip install oracledb                                                         #

# pip install pandas                                                           #

#                                                                              #

# ============================================================================ #

#

# Importação dos módulos

#

import os

import platform

import oracledb

import pandas as pd

# Try para tentativa de Conexão com o Banco de Dados Oracle

try:

    # Conectar ao banco de dados

    # ================================================================================================================== #

    # CUIDADO!!! ==>>> COM 3 TENTATIVAS ERRADAS SUA CONTA FICA EM LOCK (Solicitar ao Help Desk via Whats para desbloquear)

    # ================================================================================================================== #

    conn = oracledb.connect(*user*="rm93613", *password*="150503", *dsn*="oracle.fiap.com.br:1521/orcl")

    # Cria as instruções para cada módulo

    inst\_cadastro = conn.cursor()

    inst\_consulta = conn.cursor()

    inst\_alteracao = conn.cursor()

    inst\_exclusao = conn.cursor()

except Exception as e:

    # Informa o erro

    print("Erro: ", e)

    # Flag para não executar a Aplicação

    conexao = False

else:

    # Flag para executar a Aplicação

    conexao = True

margem = ' ' \* 4 # Define uma margem para a exibição da aplicação

# Recupera o Sistema Operacional do Cliente

sist\_oper = platform.system()

# Enquanto o flag conexao estiver apontado com True

while conexao:

    # Limpa a tela via SO - Windows: cls   Mac / Linux: clear

    if sist\_oper == 'Windows':

       os.system('cls')

    else:

        os.system('clear')

    # Apresenta o menu

    print("------- CRUD - PETSHOP -------")

    print("""

    1 - Cadastrar Pet

    2 - Listar Pets por ID

    3 - Listar Todos os Pets

    4 - Alterar Pet

    5 - Excluir Pet

    6 - EXCLUIR TODOS OS PETS

    7 - SAIR

    """)

    # Captura a escolha do usuário

    escolha = input(margem + "Escolha -> ")

    # Verifica se o número digitado é um valor numérico

    if escolha.isdigit():

        escolha = int(escolha)

    else:

        escolha = 7

        print("Digite um número.\nReinicie a Aplicação!")

    # Limpa a tela via SO

    if sist\_oper == 'Windows':

       os.system('cls')

    else:

        os.system('clear')

    # VERIFICA QUAL A ESCOLHA DO USUÁRIO

    # CADASTRAR UM PET

    if escolha == 1:

        print("----- CADASTRAR PET -----\n")

        # Recebe os valores para cadastro

        tipo = input(margem + "Digite o tipo....: ")

        nome = input(margem + "Digite o nome....: ")

        idade = int(input(margem + "Digite a idade...: "))

        # Chama a procedure para inserir um novo Pet - SEM SE PREOCUPAR COM A ORDEM DOS PARÂMETROS

        # inst\_cadastro.callproc('CP1\_CADASTRO\_PET',

        # keywordParameters={"p\_tipoPet": tipo, "p\_nomePet": nome, "p\_nomePet": idade})

        inst\_cadastro.callproc('CP1\_CADASTRO\_PET',

*parameters*=[tipo, nome, idade])

        conn.commit()

        #

        # Para realizar um Roolback, caso necessário:

        #

        #conn.rollback()

        # Caso haja sucesso na gravação

        print("\n##### Dados GRAVADOS #####")

    # LISTAR PETS POR ID

    elif escolha == 2:

        print("----- LISTAR PET POR ID-----\n")

        # Permite o usuário escolher um Pet pelo id

        pet\_id = int(input(margem + "Escolha um Id: "))

        # Chama a Função para Recuperar um Pet (Uma linha somente)

        lista = inst\_consulta.callfunc('CP1\_GET\_PET',

                                str,  ## int para retorno com NUMBER na Função. Em nosso caso retorna uma String (VARCHAR2)

                                [pet\_id])

        print(lista)

        print("\n##### LISTADO! #####")

    # LISTAR TODOS OS PETS

    elif escolha == 3:

        # Lista para a captura de dados do Banco

        lista\_dados = []

        # Monta a instrução SQL de seleção de todos os registros da tabela

        inst\_consulta.execute('SELECT \* FROM petshop')

        # Captura todos os registros da tabela e armazena no objeto data

        data = inst\_consulta.fetchall()

        # Insere os valores da tabela na Lista

        for dt in data:

            lista\_dados.append(dt)

        # ordena a lista

        lista\_dados = sorted(lista\_dados)

        # Gera um DataFrame com os dados da lista utilizando o Pandas

        dados\_df = pd.DataFrame.from\_records(lista\_dados, *columns*=['Id', 'Tipo', 'Nome', 'Idade'], *index*='Id')

        # Verifica se não há registro através do dataframe

        if dados\_df.empty:

            print(f"Não há um Pets cadastrados!")

        else:

            print(dados\_df) # Exibe os dados selecionados da tabela

        print("\n##### LISTADOS! #####")

    # ALTERAR OS DADOS DE UM REGISTRO

    elif escolha == 4:

        # ALTERANDO UM REGISTRO

        print("----- ALTERAR DADOS DO PET -----\n")

        # Lista para a captura de dados da tabela

        lista\_dados = []

        # Permite o usuário escolher um Pet pelo id

        pet\_id = int(input(margem + "Escolha um Id: "))

        # Constroi a instrução de consulta para verificar a existencia ou não do id

        consulta = f""" SELECT \* FROM petshop WHERE id = {pet\_id}"""

        inst\_consulta.execute(consulta)

        data = inst\_consulta.fetchall()

        # Preenche a lista com o registro encontrado (ou não)

        for dt in data:

            lista\_dados.append(dt)

        # analisa se foi encontrado algo

        if len(lista\_dados) == 0: # se não há o id

            print(f"Não há um pet cadastrado com o ID = {pet\_id}")

            input("\nPressione ENTER")

        else:

            # Captura os novos dados

            novo\_tipo = input(margem + "Digite um novo tipo: ")

            novo\_nome = input(margem + "Digite um novo nome: ")

            nova\_idade = input(margem + "Digite uma nova idade: ")

            # Chama a procedure para atualizar um novo Pet - SEM SE PREOCUPAR COM A ORDEM DOS PARÊMETROS

            # inst\_alteracao.callproc('nomeProcedure',

            # keywordParameters={"nomeParam1\_naProc": param1, "nomeParam2\_naProc": param2, "nomeParam3\_naProc": param3, "nomeParam4\_naProc": param4})

            inst\_cadastro.callproc('CP1\_ALTERA\_PET',

*parameters*=[pet\_id, novo\_tipo, novo\_nome, nova\_idade])

            conn.commit()

            if len(lista\_dados) == 0:

                print("\n##### ID NÃO ENCONTRADO! #####")

            else:

                print("\n##### Dados ATUALIZADOS! #####")

    # EXCLUIR UM REGISTRO

    elif escolha == 5:

        print("----- EXCLUIR PET -----\n")

        # Lista para a captura de dados da tabela

        lista\_dados = []

        # Permite o usuário escolher um Pet pelo ID

        pet\_id = input(margem + "Escolha um Id: ")

        if pet\_id.isdigit():

            pet\_id = int(pet\_id)

            consulta = f""" SELECT \* FROM petshop WHERE id = {pet\_id}"""

            inst\_consulta.execute(consulta)

            data = inst\_consulta.fetchall()

            # Insere os valores da tabela na lista

            for dt in data:

                lista\_dados.append(dt)

            # Verifica se o registro está cadastrado

            if len(lista\_dados) == 0:

                print(f"Não há um pet cadastrado com o ID = {pet\_id}")

            else:

                # Chama a procedure para deletar um novo Pet - SEM SE PREOCUPAR COM A ORDEM DOS PARÂMETROS

                inst\_exclusao.callproc('CP1\_EXCLUI\_PET', *parameters*=[pet\_id] )

                conn.commit()

                print("\n##### Pet APAGADO! #####")  # Exibe mensagem caso haja sucesso

        else:

            print("O Id não é numérico!")

    # EXCLUIR TODOS OS REGISTROS

    elif escolha == 6:

        print("\n!!!!! EXCLUI TODOS OS DADOS TABELA !!!!!\n")

        confirma = input(margem + "CONFIRMA A EXCLUSÃO DE TODOS OS PETS? [S]im ou [N]ÃO?")

        if confirma.upper() == "S":

            # Chama a procedure para limpar a tabela

            inst\_exclusao.callproc('CP1\_EXCLUI\_TODOS') ## A Procedure não recebe parâmetros

            conn.commit()

            # Depois de excluir todos os registros ele zera o ID

            data\_reset\_ids = """ ALTER TABLE petshop MODIFY(ID GENERATED AS IDENTITY (START WITH 1)) """

            inst\_exclusao.execute(data\_reset\_ids)

            print("##### Todos os registros foram excluídos! #####")

        else:

            print(margem + "Operação cancelada pelo usuário!")

    # SAI DA APLICAÇÃO

    elif escolha == 7:

        # Modificando o flag da conexão

        conexao = False

    # CASO O NUMERO DIGITADO NÃO SEJA UM DO MENU

    else:

        input(margem + "Digite um número entre 1 e 7.")

    # Pausa o fluxo da aplicação para a leitura das informações

    input(margem + "Pressione ENTER")

else:

    # Fechar a conexão com o Banco - NUNCA ESQUEÇA DE FECHAR SUA CONEXÃO COM O BANCO DE DADOS

    inst\_cadastro.close()

    inst\_consulta.close()

    inst\_alteracao.close()

    inst\_exclusao.close()

    conn.close()

    print("Obrigado por utilizar a nossa aplicação! :)")

    print("⡶⠛⢲⡄⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀")

    print("⣇⠀⠀⣇⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀")

    print("⢹⠀⠀⢻⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⢀⡤⢤⡀")

    print("⠸⡆⠀⠘⡇⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⡼⠀⢸⠇")

    print("⠀⣇⠀⠀⢷⠀⣀⣄⣀⠀⢀⣀⡀⠀⢠⠇⠀⢸⠀")

    print("⠀⢹⠀⢀⣼⣿⡿⣆⠈⣷⠏⠉⠉⣷⣾⠀⠀⡏⠀")

    print("⠀⣸⢷⠟⠛⠋⢰⠏⠀⣿⠀⠀⠀⣿⠀⠀⢠⡇⠀")

    print("⠀⡏⠀⠀⢀⡴⣏⣀⣀⣿⣄⣀⡀⡇⠀⠀⢸⠁⠀")

    print("⠀⢳⠀⠀⠘⣧⣿⡉⠸⣿⡏⢀⣿⠃⠀⠀⢸⠀⠀")

    print("⠀⠈⣧⠀⠀⠘⣟⠛⠚⠃⠙⠚⠛⠀⠀⠀⣾⠀⠀")

    print("⠀⠀⠸⡄⠀⠀⠉⠀⠀⠀⠀⠀⠀⠀⠀⢠⡇⠀⠀")

    print("⠀⠀⠀⢻⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⠀⡼⠀⠀⠀")

    print("⠀⠀⠀⠘⠣⠤⠤⠀⣀⣀⣀⡀⠤⠤⠖⠃⠀⠀⠀")

## 2 - Os códigos fontes dos procedimentos criados

SET SERVEROUTPUT ON

-- Uma Procedure para Cadastrar um Pet

CREATE OR REPLACE PROCEDURE CP1\_CADASTRO\_PET (

    p\_tipoPet IN VARCHAR,

    p\_nomePet IN VARCHAR,

    p\_idade IN NUMBER

) AS

BEGIN

    INSERT INTO PETSHOP (TIPO\_PET, NOME\_PET, IDADE)

    VALUES (p\_tipoPet, p\_nomePet, p\_idade);

END;

-- Uma Função para recuperar dados de um Pet com base no parâmetro ID.

CREATE OR REPLACE FUNCTION CP1\_GET\_PET(id\_pet NUMBER)

RETURN VARCHAR2

IS

    dados\_pet VARCHAR2(4000);

BEGIN

    SELECT

        'ID: ' || ID || CHR(10) ||

        'Nome: ' || NOME\_PET || CHR(10) ||

        'Tipo: ' || TIPO\_PET || CHR(10) ||

        'Idade: ' || IDADE

    INTO dados\_pet

    FROM PETSHOP

    WHERE ID = id\_pet;

    IF dados\_pet IS NOT NULL THEN

        RETURN dados\_pet;

    ELSE

        RETURN 'Pet não encontrado para o ID ' || id\_pet;

    END IF;

EXCEPTION

    WHEN NO\_DATA\_FOUND THEN

        RETURN 'Pet não encontrado para o ID ' || id\_pet;

END;

-- Uma Procedure para alterar dados de um Pet com base no parâmetro ID

CREATE OR REPLACE PROCEDURE CP1\_ALTERA\_PET (

    p\_id IN NUMBER,

    p\_tipoPet IN VARCHAR,

    p\_nomePet IN VARCHAR,

    p\_idade IN NUMBER

)

AS

BEGIN

    UPDATE PETSHOP

    SET TIPO\_PET = p\_tipoPet,

    NOME\_PET = p\_nomePet,

    IDADE = p\_idade

    WHERE ID = p\_id;

END;

-- Uma Procedure para excluir um Pet com base no parâmetro ID

CREATE OR REPLACE PROCEDURE CP1\_EXCLUI\_PET (

    p\_id IN NUMBER

)

AS

BEGIN

    DELETE PETSHOP

    WHERE ID = p\_id;

END;

-- Uma Procedure para excluir todos as linhas da Tabela

CREATE OR REPLACE PROCEDURE CP1\_EXCLUI\_TODOS

AS

BEGIN

    DELETE PETSHOP;

END;

Evidência:

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

## 3 - O código fonte da Package e Package Body criadas

CREATE OR REPLACE PACKAGE RM93613\_PKG\_PETS AS

  PROCEDURE CP1\_CADASTRO\_PET(p\_tipoPet VARCHAR, p\_nomePet VARCHAR, p\_idade NUMBER);

  FUNCTION CP1\_GET\_PET(p\_id NUMBER) RETURN VARCHAR2;

  PROCEDURE CP1\_ALTERA\_PET(p\_id NUMBER, p\_tipoPet VARCHAR, p\_nomePet VARCHAR, p\_idade NUMBER);

  PROCEDURE CP1\_EXCLUI\_PET(p\_id NUMBER);

  PROCEDURE CP1\_EXCLUI\_TODOS;

END RM93613\_PKG\_PETS;

CREATE OR REPLACE PACKAGE BODY RM93613\_PKG\_PETS AS

    -- Uma Procedure para Cadastrar um Pet

    PROCEDURE CP1\_CADASTRO\_PET (

        p\_tipoPet IN VARCHAR,

        p\_nomePet IN VARCHAR,

        p\_idade IN NUMBER

    ) IS

    BEGIN

        INSERT INTO PETSHOP (TIPO\_PET, NOME\_PET, IDADE)

        VALUES (p\_tipoPet, p\_nomePet, p\_idade);

        commit;

    END CP1\_CADASTRO\_PET;

    -- Uma Função para recuperar dados de um Pet com base no parâmetro ID.

    FUNCTION CP1\_GET\_PET(p\_id NUMBER)

    RETURN VARCHAR2

    IS

        dados\_pet VARCHAR2(4000);

    BEGIN

        SELECT

            'ID: ' || ID || CHR(10) ||

            'Nome: ' || NOME\_PET || CHR(10) ||

            'Tipo: ' || TIPO\_PET || CHR(10) ||

            'Idade: ' || IDADE

        INTO dados\_pet

        FROM PETSHOP

        WHERE ID = p\_id;

        IF dados\_pet IS NOT NULL THEN

            RETURN dados\_pet;

        ELSE

            RETURN 'Pet nÃ£o encontrado para o ID ' || p\_id;

        END IF;

    EXCEPTION

        WHEN NO\_DATA\_FOUND THEN

            RETURN 'Pet nÃ£o encontrado para o ID ' || p\_id;

    END CP1\_GET\_PET;

    -- Uma Procedure para alterar dados de um Pet com base no parâmetro ID

    PROCEDURE CP1\_ALTERA\_PET (

        p\_id IN NUMBER,

        p\_tipoPet IN VARCHAR,

        p\_nomePet IN VARCHAR,

        p\_idade IN NUMBER

    )

    IS

    BEGIN

        UPDATE PETSHOP

        SET TIPO\_PET = p\_tipoPet,

        NOME\_PET = p\_nomePet,

        IDADE = p\_idade

        WHERE ID = p\_id;

        commit;

    END CP1\_ALTERA\_PET;

    -- Uma Procedure para excluir um Pet com base no parâmetro ID

    PROCEDURE CP1\_EXCLUI\_PET (

        p\_id IN NUMBER

    )

    IS

    BEGIN

        DELETE PETSHOP

        WHERE ID = p\_id;

        commit;

    END CP1\_EXCLUI\_PET;

    -- Uma Procedure para excluir todos as linhas da Tabela

    PROCEDURE CP1\_EXCLUI\_TODOS

    IS

    BEGIN

        DELETE FROM PETSHOP;

        commit;

    END CP1\_EXCLUI\_TODOS;

END RM93613\_PKG\_PETS;

Evidência:

## Interface gráfica do usuário, Texto, Aplicativo, Email Descrição gerada automaticamente

## 4 - O Código fonte do Trigger criado

CREATE TABLE audit\_petshop (

    id NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,

    usuario VARCHAR2(30),

    data\_registro DATE,

    comando VARCHAR2(6),

    valor\_tipo\_antigo VARCHAR2(30),

    valor\_nome\_antigo VARCHAR2(30),

    valor\_idade\_antigo NUMBER,

    valor\_tipo\_novo VARCHAR2(30),

    valor\_nome\_novo VARCHAR2(30),

    valor\_idade\_novo NUMBER

)

CREATE OR REPLACE TRIGGER RM93613\_trg\_audit\_petshop

AFTER INSERT OR UPDATE OR DELETE ON petshop

FOR EACH ROW

DECLARE

    v\_comando VARCHAR2(6);

    v\_valorTipoAntigo VARCHAR2(30);

    v\_valorNomeAntigo VARCHAR2(30);

    v\_valorIdadeAntigo NUMBER;

    v\_valorTipoNovo VARCHAR2(30);

    v\_valorNomeNovo VARCHAR2(30);

    v\_valorIdadeNovo NUMBER;

BEGIN

    IF INSERTING THEN

        v\_comando := 'INSERT';

        v\_valorTipoAntigo := null;

        v\_valorNomeAntigo := null;

        v\_valorIdadeAntigo := null;

        v\_valorTipoNovo := :new.tipo\_pet;

        v\_valorNomeNovo := :new.nome\_pet;

        v\_valorIdadeNovo := :new.idade;

    ELSIF UPDATING THEN

        v\_comando := 'UPDATE';

        v\_valorTipoAntigo := :old.tipo\_pet;

        v\_valorNomeAntigo := :old.nome\_pet;

        v\_valorIdadeAntigo := :old.idade;

        v\_valorTipoNovo := :new.tipo\_pet;

        v\_valorNomeNovo := :new.nome\_pet;

        v\_valorIdadeNovo := :new.idade;

    ELSIF DELETING THEN

        v\_comando := 'DELETE';

        v\_valorTipoAntigo := :old.tipo\_pet;

        v\_valorNomeAntigo := :old.nome\_pet;

        v\_valorIdadeAntigo := :old.idade;

        v\_valorTipoNovo := null;

        v\_valorNomeNovo := null;

        v\_valorIdadeNovo := null;

    END IF;

    INSERT INTO audit\_petshop(usuario, data\_registro, comando, valor\_tipo\_antigo, valor\_nome\_antigo, valor\_idade\_antigo, valor\_tipo\_novo, valor\_nome\_novo, valor\_idade\_novo)

    VALUES (USER, SYSDATE, v\_comando, v\_valorTipoAntigo, v\_valorNomeAntigo, v\_valorIdadeAntigo, v\_valorTipoNovo, v\_valorNomeNovo, v\_valorIdadeNovo);

EXCEPTION

    WHEN OTHERS THEN

        DBMS\_OUTPUT.PUT\_LINE('Ocorreu um erro: ' || SQLERRM);

END;

Evidência:

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

## 5 - Prints da evidência do funcionamento de cada uma das ações propostas no script do Python

Tabela PETSHOP antes das ações:

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente Tabela AUDIT\_PETSHOP antes das ações:

1. **Cadastrar Pet:**

Texto

Descrição gerada automaticamente

Evidência de funcionamento:

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamenteEvidência do trigger:

1. **Listar Pet por ID:**

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

1. **Listar todos os Pets:**

Tela preta com letras brancas

Descrição gerada automaticamente

1. **Alterar Pet:**

Texto

Descrição gerada automaticamente

Evidência do funcionamento:

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Evidência do trigger:

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

1. **Excluir Pet:**

Texto

Descrição gerada automaticamente

Evidência do funcionamento:

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Interface gráfica do usuário, Aplicativo, Tabela

Descrição gerada automaticamente Evidência do trigger:

1. **Excluir todos os Pets:**

Texto

Descrição gerada automaticamente

Evidência do funcionamento:

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Evidência do trigger:

Interface gráfica do usuário, Aplicativo, Tabela

Descrição gerada automaticamente