
FIAP

Procedimentos, Funções e Tratamento de erros

A linguagem PL/SQL permite a construção de rotinas robustas e que podem à medida que a lógica se torna complexa, crescer muito. Uma forma de organizar melhor lógicas complexas é dividindo a mesma em partes menores, dessa forma poderíamos desenvolver um procedimento que tenha um bloco que chame um ou mais procedimentos e ou funções que por sua vez estão organizados **DENTRO DO PRÓPRIO PROCEDIMENTO**.

Isso ajuda a organizar a lógica em partes menores e torna o corpo do procedimento mais organizado

O que é uma procedure?

Uma procedure em banco de dados Oracle é um bloco de código PL/SQL que retorna ou valor. Procedures são usados para realizar ações, como inserir dados, atualizar registros ou excluir dados.

Características:

- *Retornam ou não um valor:* Procedures não precisam retornar um valor, mas podem retornar mensagens de erro ou informações de status.
- **Não podem ser usados em expressões SQL:** Procedures não podem ser usados diretamente em consultas SQL(Select).
- **São geralmente usados para tarefas complexas:** Procedures são ideais para tarefas que envolvem várias operações, como validação de dados, cálculos complexos ou manipulação de várias tabelas.

O que é uma procedure?

Vantagens:

- **Reuso de código:** Procedures permitem que você reutilize código em diferentes partes do seu banco de dados.
- **Modularidade:** Procedures ajudam a organizar o código em unidades menores e mais gerenciáveis.
- **Facilidade de leitura:** Procedures tornam o código mais legível e fácil de entender.

O que é uma Função

Uma função no banco de dados Oracle é um bloco de código PL/SQL que retorna um único valor. Funções são usadas para realizar cálculos, manipular strings e executar outras tarefas que podem ser expressas como um único valor.

O que é uma Função

Características:

- Retornam um único valor: O tipo de dado do valor retornado deve ser especificado na definição da função.
- Podem ser usadas em expressões SQL: Funções podem ser usadas em consultas SELECT, INSERT, UPDATE e DELETE.
- São geralmente usadas para tarefas simples: Funções são ideais para cálculos matemáticos, manipulação de strings e outras operações que retornam um único valor.

O que é uma Função

Vantagens:

- Reuso de código: Funções permitem que você reutilize código em diferentes partes do seu banco de dados.
- Modularidade: Funções ajudam a organizar o código em unidades menores e mais gerenciáveis.
- Facilidade de leitura: Funções tornam o código mais legível e fácil de entender.

Procedimentos e Funções

- Através de procedimentos realizamos operações de insert, update, delete e merge nas tabelas em um banco dados.
- Podemos criar n procedimentos e os mesmos podem se chamar uns ao outros conforme a necessidade da aplicação.
- Os procedimentos podem ter ou não parâmetros de entrada e/ou saída
- Procedimentos NÃO PODEM SER EXECUTADOS JUNTO COM COMANDOS DML
- Um procedimento não salva apenas o que ele mesmo realizou, salva tudo aquilo que foi feito inclusive por outros procedimentos executados dentro dele.
- A eliminação de uma tabela afeta os procedimentos ligados a ela diretamente e afeta indiretamente os procedimentos que usam outros procedimentos ligados a tabela eliminada/renomeada.

Quando utilizar uma Procedure ou uma Função

Característica	Procedimentos	Funções
Retorna valor	Não	Sim
Uso	Ações	Cálculos e manipulação de strings
Uso em expressões SQL	Não	Sim
Complexidade	Complexas	Simples

Para criarmos procedimentos devemos respeitar a sintaxe abaixo:

```
CREATE [OR REPLACE] PROCEDURE nome_da_procedure
    (parâmetro1 TIPO, parâmetro2 TIPO, ...)
IS
    -- Declaração de variáveis locais
BEGIN
    -- Corpo da procedure
    -- Instruções SQL e lógica de programação
EXCEPTION
    -- Tratamento de exceções
    WHEN nome_da_exceção1 THEN
        -- Tratamento para a primeira exceção
    WHEN nome_da_exceção2 THEN
        -- Tratamento para a segunda exceção
    ...
END nome_da_procedure;
```

Para criarmos uma função devemos respeitar a sintaxe abaixo:

```
CREATE [OR REPLACE] FUNCTION nome_da_funcao
    (lista_de_parametros)
RETURN tipo_de_retorno
IS
    declarações_locais;
BEGIN
    -- corpo da função
    -- código para processar os parâmetros e calcular o resultado
    RETURN resultado;
EXCEPTION
    WHEN excecao THEN
        -- tratamento de exceção
END nome_da_funcao;
/
```

Precedure X Funções

- Função: Uma função é um bloco de código que, a partir de uma lógica pré-definida, retorna um valor.
- Uma procedure: é uma unidade/módulo de um programa que executa uma tarefa bem específica.

- Funções Obrigatoriamente tem que ter retorno
- Procedures retorno opcional

Prática

```
CREATE OR REPLACE FUNCTION fun_calcula_fgts (
    p_val NUMBER
) RETURN NUMBER IS
BEGIN
    RETURN p_val * 0.08;
END fun_calcula_fgts;
```

```
CREATE OR REPLACE PROCEDURE proc2 IS
    v_valor NUMBER;
BEGIN
    v_valor := fun_calcula_fgts(5000);
    dbms_output.put_line('O VALOR DO FGTS É: ' || to_char(v_valor));
END proc2;
/
```

```
BEGIN
    proc2;
END;

EXEC PROC2();
```

TRATAMENTO DE ERROS E EXECEÇÕES

RAISE_APPLICATION_ERROR :

É usado para exibir mensagens de erro definidas pelo usuário com número de erro cujo intervalo está entre -20000 e -20999. Quando RAISE_APPLICATION_ERROR é executado, ele retorna uma mensagem e um código de erro que se parecem com o erro interno do Oracle.

Tratamento de Erros no Bloco PL/SQL

```
DECLARE
    minhaexe EXCEPTION;
    n NUMBER := 10;
BEGIN
    FOR i IN &i..&n LOOP
        dbms_output.put_line(i * i);
        IF i * 2 = 10 THEN
            RAISE minhaexe;
        END IF;
    END LOOP;
EXCEPTION
    WHEN minhaexe THEN
        raise_application_error(-20015, 'Você caiu na exceção');
END;
```

WHEN OTHERS, SQLERRM e SQLCODE

TIPO	DESCRIÇÃO
INVALID_CURSOR	Referência feita sobre um cursor inexistente (ex: FETCH sobre um cursor já fechado).
INVALID_NUMBER	Operação de atribuição ou conversão de tipos resultou em um número inválido.
NO_DATA_FOUND	Disparada em três cenários: (i) quando um SELECT INTO que não retorna valor é executado; (ii) tentativa de acesso de uma linha não existente em uma coleção PL/SQL; (iii) fim de arquivo alcançado (quando estamos fazendo leitura sequencial com o uso do pacote UTL_FILE).
PROGRAM_ERROR	A PL/SQL encontrou um problema interno. Nesse caso, pode ser preciso chamar o suporte da Oracle.
TOO_MANY_ROWS	Uma operação SELECT INTO retornou mais de uma linha.
VALUE_ERROR	Erro de conversão de tipos ou na atribuição de um valor a uma variável.
ZERO_DIVIDE	Tentativa de divisão por zero

Usando um procedimento para inserir dados

```
CREATE OR REPLACE PROCEDURE inserir_cliente(p_cod_cliente       IN NUMBER,
                                            p_nom_cliente       IN VARCHAR2,
                                            p_des_razao_social  IN VARCHAR2,
                                            p_tip_pessoa        IN CHAR,
                                            p_num_cpf_cnpj      IN NUMBER,
                                            p_dat_cadastro      IN DATE,
                                            p_dat_cancelamento  IN DATE,
                                            p_sta_ativo         IN CHAR) IS
BEGIN
  INSERT INTO cliente
    (COD_CLIENTE,
     NOM_CLIENTE,
     DES_RAZAO_SOCIAL,
     TIP_PESSOA,
     NUM_CPF_CNPJ,
     DAT_CADASTRO,
     DAT_CANCELAMENTO,
     STA_ATIVO)
  VALUES
    (p_cod_cliente,
     p_nom_cliente,
     p_des_razao_social,
     p_tip_pessoa,
     p_num_cpf_cnpj,
     p_dat_cadastro,
     p_dat_cancelamento,
     p_sta_ativo);

  COMMIT; -- Confirma a transação
  DBMS_OUTPUT.PUT_LINE('Cliente inserido com sucesso.');
END inserir_cliente;
```

Parâmetros em Procedimentos

Conforme já falamos anteriormente procedimentos podem ter parâmetros ou não, veremos então os modos como os parâmetros se comportam.

Procedimento de entrada:

```
CREATE OR REPLACE PROCEDURE proc1 (
    p_val IN NUMBER
) IS
BEGIN
    NULL;
END proc1;
/

DECLARE
    v_val NUMBER := 30;
BEGIN
    proc1(20); -- OU PODEMOS CHAMAR USANDO UMA VARIAVEL)
    proc1(v_val);
    DBMS_OUTPUT.PUT_LINE(v_val);
END;
/
```

Exemplo parâmetros de saída:

```
CREATE OR REPLACE PROCEDURE consultar_cliente (
    p_cod_cliente            IN NUMBER,
    p_out_nome               OUT VARCHAR2,
    p_out_razao_social       OUT VARCHAR2,
    p_out_tipo_pessoa        OUT CHAR,
    p_out_cpf_cnpj           OUT NUMBER,
    p_out_data_cadastro      OUT DATE,
    p_out_data_cancelamento  OUT DATE,
    p_out_status             OUT CHAR
) IS
BEGIN
    SELECT
        nom_cliente,
        des_razao_social,
        tip_pessoa,
        num_cpf_cnpj,
        dat_cadastro,
        dat_cancelamento,
        sta_ativo
    INTO
        p_out_nome,
        p_out_razao_social,
        p_out_tipo_pessoa,
        p_out_cpf_cnpj,
        p_out_data_cadastro,
        p_out_data_cancelamento,
        p_out_status
    FROM
        cliente
    WHERE
        cod_cliente = p_cod_cliente;

EXCEPTION
    WHEN no_data_found THEN
        dbms_output.put_line('Nenhum cliente encontrado para o código fornecido.');
    WHEN OTHERS THEN
        dbms_output.put_line('Erro ao consultar o cliente: ' || sqlerrm);
END consultar_cliente;
/
```

Procedimento para realizar Update

```
CREATE OR REPLACE PROCEDURE atualizar_produto(
    p_cod_produto IN NUMBER,
    p_nom_produto IN VARCHAR2,
    p_cod_barra IN VARCHAR2,
    p_sta_ativo IN VARCHAR2,
    p_out_dat_cadastro OUT DATE,
    p_out_dat_cancelamento OUT DATE
)
 IS
BEGIN
    -- Atualizar o produto com os novos valores
    UPDATE produto
    SET NOM_PRODUTO = p_nom_produto,
        COD_BARRA = p_cod_barra,
        STA_ATIVO = p_sta_ativo
    WHERE COD_PRODUTO = p_cod_produto
    RETURNING DAT_CADASTRO, DAT_CANCELAMENTO INTO p_out_dat_cadastro, p_out_dat_cancelamento;

    -- Se o produto foi atualizado com sucesso, COMMIT
    COMMIT;
 EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Nenhum produto encontrado para o código fornecido.');
    WHEN OTHERS THEN
        -- Em caso de erro, ROLLBACK e exibir mensagem de erro
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Erro ao atualizar o produto: ' || SQLERRM);
END atualizar_produto;
 /
```

Testando a procedure de update

```
DECLARE
    v_dat_cadastro DATE;
    v_dat_cancelamento DATE;
BEGIN
    atualizar_produto(
        p_cod_produto => 123,  -- Substitua pelo código do produto desejado
        p_nom_produto => 'Novo Nome',  -- Substitua pelo novo nome do produto
        p_cod_barra => '1234567890',  -- Substitua pelo novo código de barras do produto
        p_sta_ativo => 'S',  -- Substitua pelo novo status do produto
        p_out_dat_cadastro => v_dat_cadastro,
        p_out_dat_cancelamento => v_dat_cancelamento
    );

    DBMS_OUTPUT.PUT_LINE('Produto atualizado com sucesso.');
    DBMS_OUTPUT.PUT_LINE('Data de cadastro atualizada: ' || v_dat_cadastro);
    DBMS_OUTPUT.PUT_LINE('Data de cancelamento atualizada: ' || v_dat_cancelamento);
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Erro ao atualizar o produto: ' || SQLERRM);
END;
/
```

Procedimento para realizar Delete

```
CREATE OR REPLACE PROCEDURE deletar_item_pedido(
    p_cod_pedido IN NUMBER,
    p_cod_item_pedido IN INTEGER,
    p_out_qtd_item_deletado OUT NUMBER,
    p_out_val_unitario_item_deletado OUT NUMBER,
    p_out_val_desconto_item_deletado OUT NUMBER
)
IS
BEGIN
    -- Deleta o item do pedido e retorna os valores deletados
    DELETE FROM item_pedido
    WHERE COD_PEDIDO = p_cod_pedido
    AND COD_ITEM_PEDIDO = p_cod_item_pedido
    RETURNING QTD_ITEM, VAL_UNITARIO_ITEM, VAL_DESCONTO_ITEM
    INTO p_out_qtd_item_deletado, p_out_val_unitario_item_deletado, p_out_val_desconto_item_deletado;

    -- Verifica se algum item foi deletado
    IF SQL%ROWCOUNT = 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Nenhum item encontrado para exclusão.');
    END IF;

    -- Se o item foi deletado com sucesso, COMMIT
    COMMIT;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20002, 'Nenhum item encontrado para exclusão.');
    WHEN OTHERS THEN
        -- Em caso de erro, ROLLBACK e exibir mensagem de erro
        ROLLBACK;
        RAISE_APPLICATION_ERROR(-20003, 'Erro ao deletar o item do pedido: ' || SQLERRM);
END deletar_item_pedido;
/
```

Testando a procedure

```
set SERVEROUTPUT on;

DECLARE
    v_qtd_item_deletado NUMBER;
    v_val_unitario_item_deletado NUMBER;
    v_val_desconto_item_deletado NUMBER;
BEGIN
    deletar_item_pedido(
        p_cod_pedido => 123,  -- Substitua pelo código do pedido desejado
        p_cod_item_pedido => 456,  -- Substitua pelo código do item do pedido desejado
        p_out_qtd_item_deletado => v_qtd_item_deletado,
        p_out_val_unitario_item_deletado => v_val_unitario_item_deletado,
        p_out_val_desconto_item_deletado => v_val_desconto_item_deletado
    );

    DBMS_OUTPUT.PUT_LINE('Item deletado com sucesso.');
    DBMS_OUTPUT.PUT_LINE('Quantidade do item deletado: ' || v_qtd_item_deletado);
    DBMS_OUTPUT.PUT_LINE('Valor unitário do item deletado: ' || v_val_unitario_item_deletado);
    DBMS_OUTPUT.PUT_LINE('Valor de desconto do item deletado: ' || v_val_desconto_item_deletado);
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Erro ao deletar o item do pedido: ' || SQLERRM);
END;
/
```

Procedure com Valor default

```
CREATE OR REPLACE PROCEDURE inserir_usuario(
    p_cod_usuario IN NUMBER,
    p_nom_usuario IN VARCHAR2,
    p_sta_ativo IN CHAR DEFAULT 'S'  -- Valor padrão 'S' para o parâmetro STA_ATIVO
)
IS
BEGIN
    INSERT INTO usuario (COD_USUARIO, NOM_USUARIO, STA_ATIVO)
    VALUES (p_cod_usuario, p_nom_usuario, p_sta_ativo);

    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Usuário inserido com sucesso.');
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Erro ao inserir o usuário: ' || SQLERRM);
END inserir_usuario;
/
```

Invalidando Procedimentos

- Eliminarmos uma tabela base de um procedimento
- Renomearmos uma tabela base de um procedimento
- Eliminarmos colunas de uma tabela base de um procedimento
- Incluirmos uma nova coluna em um tabela base de um procedimento
- Eliminarmos ou renomearmos visões base de um procedimento
- Eliminarmos ou renomearmos funções usados em um procedimento
- Eliminarmos ou renomearmos procedimentos usados em um outro procedimento
- Eliminarmos parâmetros em procedimentos e funções usados em um procedimento
- Adicionarmos um novo parâmetro a um procedimento, desde que não use condição DEFAULT
- Eliminarmos ou renomearmos uma sequência usada em um procedimento
- Eliminarmos um usuário e seus objetos

Exercícios

1) Crie um procedimento chamado prc_insere_produto para todas as colunas da tabela de produtos, valide:

    Se o nome do produto tem mais de 3 caracteres e não contem números (0 a 9)

2) Crie um procedimento chamado prc_insere_cliente para inserir novos clientes, valide:

    Se o nome do cliente tem mais de 3 caracteres e não contem números (0 a 9)

3) Crie uma função chamada FUN_VALIDA_NOME que valide se o nome tem mais do que 3 caracteres e não tenha números.

4) Altere os procedimentos dos exercícios 1 e 2 para chamar a função do exercício 3

5) Altere o procedimento do exercício 1 para que tenha um último parâmetro chamado P_RETORNO do tipo varchar2 que deverá retornar a informação ‘produto cadastrado com sucesso’.

6) crie um bloco anônimo e chame o procedimento do exercício 1.