

Tratamento de Exceções em Functions e Procedures

Modelagem de Dados

Tratamento de Exceções

- A linguagem PL/pgSQL permite o tratamento de exceções nas suas funções e procedimentos de maneira similar a outras linguagens de programação
- AS exceções são capturadas usando blocos BEGIN...EXCEPTION...WHEN...END
- Dentro do bloco é colocado o código que pode potencialmente gerar uma exceção
- Diferentes blocos de exceção podem ser especificados para lidar com diferentes tipos de exceções, ou seja, podem existir blocos para tratar exceções específicas e ter um bloco para lidar com outras as exceções.

Sintaxe no PL/pgSQL

BEGIN

-- Código que pode gerar uma exceção

EXCEPTION

WHEN exceção THEN

-- Tratamento para a exceção

END;

Exemplo

```
CREATE OR REPLACE PROCEDURE dividir_por_zero(num1 NUMERIC, num2 NUMERIC)
LANGUAGE plpgsql;
DECLARE
    resultado NUMERIC;
BEGIN
    resultado := num1/num2;
EXCEPTION
    WHEN division_by_zero THEN -- Tratamento para exceção de divisão por zero
        resultado = 0;
    WHEN others THEN -- Tratamento para outras exceções não especificadas
        resultado = -1;
END;
$$
```

WHEN others



- Pode ser usado no PostgreSQL com PL/pgSQL para capturar qualquer exceção não tratada especificamente acima
- Funciona como um "catch genérico" dentro de blocos EXCEPTION
- Deve ser a última cláusula dentro do EXCEPTION
- Não pode estar sozinha, pelo menos um WHEN específico deve vir antes

Algumas exceções capturadas

- **foreign_key_violation**: violações de chave estrangeira
- **unique_violation**: violações de restrições de chave única
- **check_violation**: violações de restrições de verificação
- **not_null_violation**: tentativas de inserir valores nulos em colunas que não permitem
- **string_data_right_truncation**: truncamento de dados ao tentar inserir valores de strings longas em colunas com tamanho limitado
- **division_by_zero**: tentativas de divisão por zero
- **invalid_text_representation**: tentativas de converter valores de texto em tipos de dados inválidos
- **integrity_constraint_violation**: violações de integridade de dados

Exemplo de foreign_key_violation

```
CREATE OR REPLACE PROCEDURE inserir_pedido(pedido_id INT, cliente_id INT)
LANGUAGE plpgsql
AS $$
BEGIN
    INSERT INTO pedidos (id, cliente_id) VALUES (pedido_id, cliente_id);
EXCEPTION
    WHEN foreign_key_violation THEN
        RAISE EXCEPTION 'Erro: Cliente ID % não existe na tabela clientes!', cliente_id;
END;
$$;
```

Exemplo de foreign_key_violation

```
CREATE TABLE clientes (  
  id INT PRIMARY KEY,  
  nome TEXT  
);
```

```
CREATE TABLE pedidos (  
  id INT PRIMARY KEY,  
  cliente_id INT REFERENCES clientes(id)  
);
```

```
CALL inserir_pedido(1, 999);
```

- Se 999 não existir na tabela clientes, ocorrerá uma foreign_key_violation:
ERROR: insert or update on table "pedidos" violates foreign key constraint "pedidos_cliente_id_fkey"
DETAIL: Key (cliente_id)=(999) is not present in table "clientes".
- O erro tratado por EXCEPTION mostra a mensagem em RAISE:
Erro: Cliente ID 999 não existe na tabela clientes!

Exemplo de unique_violation

```
CREATE OR REPLACE PROCEDURE inserir_usuario(nome TEXT, email TEXT)
LANGUAGE plpgsql
AS $$
BEGIN
    INSERT INTO usuarios (nome, email) VALUES (nome, email);
EXCEPTION
    WHEN unique_violation THEN
        RAISE EXCEPTION 'Erro: O email % já está cadastrado!', email;
END;
$$;
```

Exemplo de unique_violation

```
CREATE TABLE usuarios (  
    id SERIAL PRIMARY KEY,  
    email TEXT UNIQUE  
);
```

```
CALL inserir_usuario('João', 'joao@email.com');
```

```
CALL inserir_usuario('Carlos', 'joao@email.com');
```

- O segundo INSERT falha porque o e-mail já existe:

ERROR: duplicate key value violates unique constraint "usuarios_email_key"

DETAIL: Key (email)=(teste@email.com) already exists.

- O erro tratado por EXCEPTION mostra a mensagem em RAISE:

Erro: O email joao@email.com já está cadastrado!

Exemplo de check_violation

```
CREATE OR REPLACE PROCEDURE inserir_funcionario(nome TEXT, idade INT)
LANGUAGE plpgsql
AS $$
BEGIN
    INSERT INTO funcionarios (nome, idade) VALUES (nome, idade);
EXCEPTION
    WHEN check_violation THEN
        RAISE EXCEPTION 'Erro: A idade % não atende à restrição!', idade;
END;
$$;
```

Exemplo de check_violation

```
CREATE TABLE funcionarios (  
  id SERIAL PRIMARY KEY,  
  nome VARCHAR(50),  
  idade INT CHECK (idade >= 18)  
);
```

```
CALL inserir_funcionario('Lucas', 15); -- menor que 18
```

- O banco impede a inserção de um funcionário com idade < 18

ERROR: new row for relation "funcionarios" violates check constraint
"funcionarios_idade_check"

DETAIL: Failing row contains (15).

- O erro tratado por EXCEPTION mostra a mensagem em RAISE:

Erro: A idade15 não atende à restrição!

Exemplo de not_null_violation

```
CREATE OR REPLACE PROCEDURE inserir_cliente(nome TEXT)
LANGUAGE plpgsql
AS $$
BEGIN
    INSERT INTO clientes (nome) VALUES (nome);
EXCEPTION
    WHEN not_null_violation THEN
        RAISE EXCEPTION 'Erro: O campo "nome" não pode ser nulo!';
END;
$$;
```

Exemplo de not_null_violation

```
CREATE TABLE clientes (  
  id SERIAL PRIMARY KEY,  
  nome TEXT NOT NULL  
);  
CALL inserir_cliente(NULL); -- nome é obrigatório
```

- O banco exige que a coluna nome tenha um valor válido:
ERROR: null value in column "nome" violates not-null constraint
DETAIL: Failing row contains (NULL).
- O erro tratado por EXCEPTION mostra a mensagem em RAISE:
Erro: O campo "nome" não pode ser nulo!

Exemplo de string_data_right_truncation



```
CREATE OR REPLACE PROCEDURE inserir_artigo(titulo TEXT)
LANGUAGE plpgsql
AS $$
BEGIN
    INSERT INTO artigos (titulo) VALUES (titulo);
EXCEPTION
    WHEN string_data_right_truncation THEN
        RAISE NOTICE 'Título truncado. O título informado é muito longo!';
END;
$$;
```

Exemplo de string_data_right_truncation



```
CREATE TABLE artigos (  
    id SERIAL PRIMARY KEY,  
    titulo VARCHAR(10)  
);
```

```
CALL inserir_artigo('Esse título é muito longo'); -- > 10 caracteres
```

- A string ultrapassa o limite de 10 caracteres:

```
ERROR: value too long for type character varying(10)
```

- O erro tratado por EXCEPTION mostra a mensagem em RAISE:
Título truncado. O título informado é muito longo!

Exemplo de division_by_zero

```
CREATE OR REPLACE PROCEDURE dividir_valores(a INT, b INT)
LANGUAGE plpgsql
AS $$
BEGIN
    resultado := a / b;
    RAISE NOTICE 'Resultado = %', resultado;
EXCEPTION
    WHEN division_by_zero THEN
        RAISE NOTICE 'Divisão por zero! Resultado ajustado para zero';
        resultado = 0;
END;
$$;
```

Exemplo de division_by_zero

CALL dividir_valores(10, 0); -- b = 0 causa erro

- Em matemática, a divisão por zero é indefinida, então o banco gera um erro:
ERROR: division by zero

- O erro tratado por EXCEPTION mostra a mensagem em RAISE:

NOTA: Divisão por zero! Resultado ajustado para zero

Exemplo de invalid_text_representation

```
CREATE OR REPLACE PROCEDURE converter_para_inteiro(texto TEXT)
LANGUAGE plpgsql
AS $$
BEGIN
    numero := texto::INT;
    RAISE NOTICE 'Numero = % ', numero;
EXCEPTION
    WHEN invalid_text_representation THEN
        RAISE NOTICE 'O texto % não pode ser convertido para número.', texto;
        numero := NULL;
END;
$$;
```

Exemplo de invalid_text_representation

CALL converter_para_inteiro('abc'); -- não é número

- O banco não pode converter 'abc' para um número:
ERROR: invalid input syntax for type integer: "abc"
- O erro tratado por EXCEPTION mostra a mensagem em RAISE:
O texto abc não pode ser convertido para número.



integrity_constraint_violation

- Categoria ampla que abrange várias restrições do banco de dados
- Útil porque permite capturar qualquer violação de integridade sem precisar tratar cada erro separadamente
- Restrições capturadas:
 - Violação de Chave Estrangeira (foreign_key_violation)
 - Violação de Restrição CHECK (check_violation)
 - Violação de Chave Única (unique_violation)
 - Violação de NOT NULL (not_null_violation)

Exemplo de integrity_constraint_violation



```
CREATE OR REPLACE PROCEDURE inserir_produto(nome TEXT, preco NUMERIC)
LANGUAGE plpgsql
AS $$
BEGIN
    INSERT INTO produtos (nome, preco) VALUES (nome, preco);
EXCEPTION
    WHEN integrity_constraint_violation THEN
        RAISE EXCEPTION 'Erro: Violação de integridade ao inserir o produto %.', nome;
END;
$$;
```

Exemplo de integrity_constraint_violation



```
CREATE TABLE produtos (  
  id SERIAL PRIMARY KEY,  
  nome TEXT UNIQUE,  
  preco NUMERIC CHECK (preco > 0)  
);  
CALL inserir_produto('Notebook', 3500);  
CALL inserir_produto('Notebook', 4200);  
ERROR: duplicate key value violates unique constraint "produtos_nome_key"  
DETAIL: Key (nome)=(Notebook) already exists.
```

- O erro tratado por EXCEPTION mostra a mensagem em RAISE:
Erro: Violação de integridade ao inserir o produto Notebook.

Exemplo de integrity_constraint_violation

```
CREATE TABLE produtos (  
  id SERIAL PRIMARY KEY,  
  nome TEXT UNIQUE,  
  preco NUMERIC CHECK (preco > 0)  
);
```

```
CALL inserir_produto('Produto Teste', -10); -- preco inválido
```


```
ERROR: new row for relation "produtos" violates check constraint  
"produtos_preco_check"
```

```
DETAIL: Failing row contains (Produto Teste, -10).
```

- O erro tratado por EXCEPTION mostra a mensagem em RAISE:
Erro: Violação de integridade ao inserir o produto Notebook.



Exercícios

- Crie o banco de dados dbBiblioteca
 - Baixe da plataforma Odette o script para criar as tabelas e registros do BD
 - Crie as funções e procedimentos a seguir fazendo as verificações necessárias e retornando os erros encontrados
 - Trate as exceções com mais probabilidade de acontecer
 - Faça os testes para todas as possibilidades
- 



Exercício 1



Crie a função `contar_livros` que retorna a quantidade de livros em estoque no banco de dados.

Exercício 2



Crie a função `listar_autores` que retorna o nome dos autores que tem uma determinada quantidade de livros publicados, recebida como parâmetro. Mostrar uma mensagem caso não tenha autor algum com a quantidade recebida.

Exercício 3



Crie o procedimento `adicionar_novos` que adiciona um novo livro e um novo autor recebidos como parâmetros e faz o vínculo entre eles. Caso o livro ou o autor já exista no banco, NÃO inserir e retornar erro específico. Trate os erros de restrições das tabelas: UNIQUE e NOT NULL.

Exercício 4



Crie a função `verificar_livro_autor` que recebe como parâmetros o nome do livro e o nome do autor e verifica se eles estão no banco. Ela deve retornar uma palavra que indica uma de quatro possibilidades: os dois estão, somente o livro está, somente o autor está ou nenhum dos dois está no banco.

Exercício 5



Crie o procedimento chamado `adicionar_livro_autor` que recebe como parâmetros os dados do livro e do autor. Verifica se já estão no banco (função `verificar_livro_autor`), insere o que for necessário (livro e/ou autor) e faz o vínculo entre livro e autor, caso tenha inserido um deles (use a função `adicionar_novos` para adicionar os dois). Trate os erros de restrições das tabelas: `UNIQUE` e `NOT NULL`.

Exercício 6



Crie o procedimento `adiciona_livro_estoque` que recebe com parâmetros o id do livro e a quantidade de livros que deve ser adicionada ao estoque, faz essa adição e mostra uma mensagem com o novo estoque.

Exercício 7



Crie o procedimento `processa_venda` que recebe como parâmetros as informações da venda e, caso tenha estoque suficiente, registra a venda e atualiza o estoque. Trate os erros de restrições de valores da tabela: quantidade e preço maiores que zero.