Sistema de Seguros - Stored Procedures e Triggers

Este projeto implementa um sistema de seguros utilizando PostgreSQL e inclui stored procedures e triggers para garantir a integridade dos dados e realizar ações específicas em determinados eventos.

Stored Procedures

InserirClienteComVerificacao

Esta stored procedure é responsável por inserir um novo cliente na tabela `Cliente` com uma verificação prévia para evitar duplicatas. Ela recebe os parâmetros necessários para criar um novo cliente e verifica se o CPF já existe na tabela. Caso exista, um erro é lançado. Caso contrário, o novo cliente é inserido na tabela.

```
CREATE OR REPLACE PROCEDURE InserirClienteComVerificacao(
    p_nome varchar,
    p_telefone int,
    p_data_nascimento date,
    p_CPF bigint,
    p_tipo_pessoa varchar,
    p_data_registro date,
    p_observacao varchar,
    p_email varchar,
    p_documento varchar,
    p_id_endereco int
)
AS $$
DECLARE
    cliente_existente BOOLEAN;
BEGIN
    -- Verificar se o cliente já existe
    SELECT EXISTS (SELECT 1 FROM Cliente WHERE CPF = p_CPF) INTO cliente_existente;

    IF cliente_existente THEN
        RAISE EXCEPTION 'Cliente com CPF % já existe.', p_CPF;
    ELSE
        -- Inserir o novo cliente
        INSERT INTO Cliente (nome, telefone, data_nascimento, CPF, tipo_pessoa,
data_registro, observacao, email, documento, id_endereco)
        VALUES (p_nome, p_telefone, p_data_nascimento, p_CPF, p_tipo_pessoa,
p_data_registro, p_observacao, p_email, p_documento, p_id_endereco);
    END IF;
END;
$$ LANGUAGE plpgsql;
```

AtribuirSeguroAoVeiculo

Esta stored procedure é responsável por atribuir um seguro a um veículo existente na tabela `Veiculo`. Ela recebe os IDs do veículo e do seguro e verifica se ambos existem nas respectivas tabelas. Caso um dos registros não seja encontrado, um erro é lançado. Caso contrário, o seguro do veículo é atualizado com o ID do seguro fornecido.

```
CREATE OR REPLACE PROCEDURE AtribuirSeguroAoVeiculo(
    p_id_veiculo int,
    p_id_seguro int
)
AS $$
BEGIN
    -- Verificar se o veículo e o seguro existem
    IF NOT EXISTS (SELECT 1 FROM Veiculo WHERE id_veiculo = p_id_veiculo) THEN
        RAISE EXCEPTION 'Veículo com ID % não existe.', p_id_veiculo;
    END IF;

    IF NOT EXISTS (SELECT 1 FROM Seguro WHERE id_seguro = p_id_seguro) THEN
        RAISE EXCEPTION 'Seguro com ID % não existe.', p_id_seguro;
    END IF;

    -- Atualizar o seguro do veículo
    UPDATE Veiculo SET fk_Seguro_id_seguro = p_id_seguro WHERE id_veiculo =
p_id_veiculo;
END;
$$ LANGUAGE plpgsql;
```

TRIGGERS

trg_atualizar_data_registro_cliente

Esta trigger é acionada antes da inserção de um novo registro na tabela `Cliente`. Ela chama a função `AtualizarDataRegistroCliente`, que atualiza a coluna `data_registro` para a data atual.

```
CREATE OR REPLACE FUNCTION AtualizarDataRegistroCliente()
RETURNS TRIGGER AS $$
BEGIN
    NEW.data_registro := current_date; -- Atualiza a data de registro para a data
atual
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_atualizar_data_registro_cliente
BEFORE INSERT ON Cliente
FOR EACH ROW
EXECUTE FUNCTION AtualizarDataRegistroCliente();
```

trg_registrar_exclusao_veiculo

Esta trigger é acionada após a exclusão de um registro na tabela `Veiculo`. Ela chama a função `RegistrarExclusaoVeiculo`, que realiza uma ação desejada ao excluir um veículo. No exemplo fornecido, a ação é inserir uma linha na tabela `LogExclusaoVeiculo` para registrar o ID do veículo excluído e a data e hora da exclusão.

```
CREATE OR REPLACE FUNCTION RegistrarExclusaoVeiculo()
RETURNS TRIGGER AS $$
BEGIN
    -- Realizar ação desejada ao excluir um veículo
    -- Por exemplo, você pode inserir uma linha em uma tabela de log
    INSERT INTO LogExclusaoVeiculo (id_veiculo, data_exclusao) VALUES
(OLD.id_veiculo, current_timestamp);

    RETURN OLD;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_registrar_exclusao_veiculo
AFTER DELETE ON Veiculo
FOR EACH ROW
EXECUTE FUNCTION RegistrarExclusaoVeiculo();
```

trg_validar_idade_minima_cliente

Esta trigger é acionada antes da inserção de um novo registro na tabela `Cliente`. Ela chama a função `s`, que valida se o cliente tem pelo menos 18 anos de idade.

```
CREATE OR REPLACE FUNCTION ValidarIdadeMinimaCliente()
RETURNS TRIGGER AS $$
DECLARE
    idade_cliente integer;
BEGIN
    -- Calcula a idade do cliente
    idade_cliente := date_part('year', current_date) - date_part('year',
NEW.data_nascimento);
    IF idade_cliente < 18 THEN
        RAISE EXCEPTION 'O cliente deve ter no mínimo 18 anos.';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_validar_idade_minima_cliente
BEFORE INSERT ON Cliente
FOR EACH ROW
EXECUTE FUNCTION ValidarIdadeMinimaCliente();
```

Observações

Certifique-se de criar as tabelas e as demais estruturas necessárias, como a tabela `LogExclusaoVeiculo`, antes de utilizar as stored procedures e triggers fornecidas.