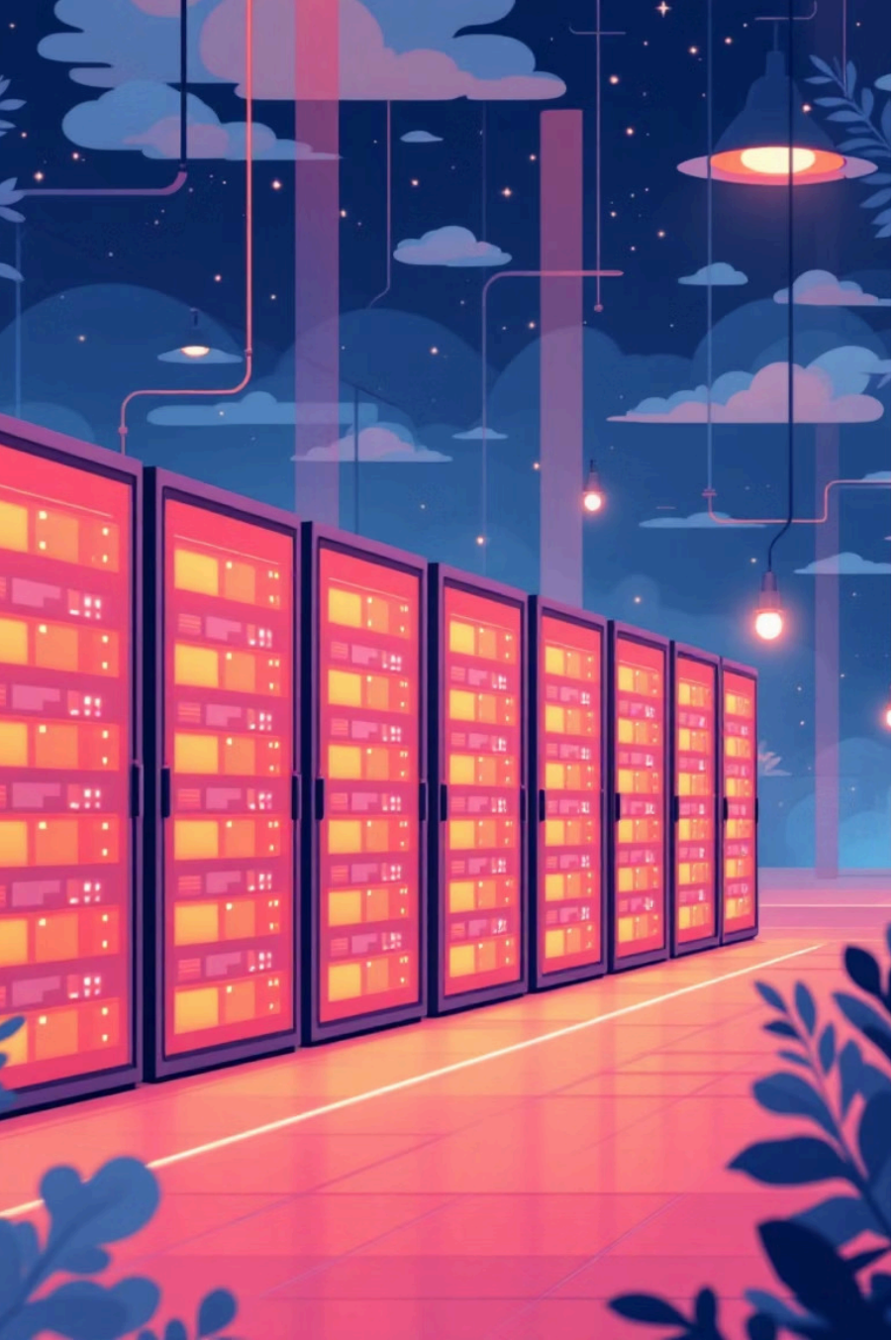

PostgreSQL: Funções vs. INSERT Direto

Descubra por que funções armazenadas podem transformar sua aplicação

Visão Geral do Que Vamos Explorar

Trafego de Rede

Menos idas e voltas entre aplicação e banco de dados

Segurança

Controle granular e validação centralizada de dados

Performance

Execução otimizada e reutilização de plano de execução

Manutenibilidade

Lógica centralizada que facilita updates e debugging

Vantagem #1: Menos Tráfego de Rede

Menos Bytes

Nome da função + parâmetros vs. query completa

Menos Round Trips

Operações complexas em uma única chamada

Mais Velocidade

Redução significativa em conexões lentas

Vantagem #2: Segurança Aumentada

Benefícios de Segurança

- Usuários da aplicação não precisam de privilégios diretos em tabelas
- Validação centralizada de entrada de dados
- Controle de acesso baseado em funções
- Proteção contra SQL injection mais robusta
- Auditoria centralizada de operações

Vantagem #3: Performance Otimizada

Primeira Execução

Plano de execução é analisado e otimizado pelo PostgreSQL

1

2

Cache de Plano

Plano compilado é armazenado para reutilização

3

Execuções Futuras

Uso imediato do plano cacheado = mais rápido

Dica: Funções também permitem lógica complexa sem múltiplas round trips, reduzindo latência geral

Comparativo de Performance

40%

Redução de Latência

Em operações complexas com múltiplas validações

65%

Menos Tráfego

Bytes enviados pela rede comparado ao INSERT direto

3x

Mais Eficiente

Reutilização de plano de execução otimizado

Outras Vantagens Importantes

Consistência de Dados

Regras de negócio aplicadas sempre que a função é chamada, garantindo integridade em todos os pontos de acesso

Manutenção Simplificada

Alterações na lógica de negócio em um único lugar, sem precisar modificar múltiplas aplicações

Versionamento Controlado

Atualizações de schema podem incluir novas versões de funções mantendo compatibilidade

Quando Usar Cada Abordagem

✓ Use Funções Quando:

- Operações complexas com múltiplas etapas
- Validação de dados necessária
- Múltiplas aplicações acessando o banco
- Segurança rigorosa é prioridade
- Performance é crítica

⚠️ INSERT Direto Pode Ser OK:

- Operações simples e rápidas
- Protótipos e testes iniciais
- Scripts de migração pontuais
- Aplicações isoladas com controle total

Exemplo Prático: Implementação

Vamos ver o código SQL para a criação da tabela `produto`, uma inserção direta e a definição completa da função `inserir_produto`.

Tabela e Inserção Direta

```
CREATE TABLE produto(
  codigo integer primary key,
  nome varchar(120) not null,
  preco decimal not null
);

INSERT INTO
produto (codigo, nome, preco)
VALUES (1, 'Produto A', 12.55);

COMMIT;

SELECT * FROM produto;
```

A função implementa validações de negócio e tratamento de erros, retornando códigos numéricos para diferentes cenários.

Exemplo Prático: Implementação

Vamos ver o código SQL para a criação da tabela `produto`, uma inserção direta e a definição completa da função `inserir_produto`.

Definição da Função `inserir_produto`

```
DROP FUNCTION IF EXISTS inserir_produto(integer, varchar, numeric);

CREATE FUNCTION inserir_produto(
    p_codigo integer,
    p_nome varchar,
    p_preco numeric
)
RETURNS integer
LANGUAGE plpgsql
AS $$
DECLARE
    v_count     integer;
    v_sqlstate  text;
BEGIN
    IF p_codigo <= 0 OR p_codigo > 2147483647 THEN
      RETURN 2;
    END IF;
    SELECT COUNT(*) INTO v_count FROM produto WHERE codigo = p_codigo;
    IF v_count > 0 THEN
      RETURN 1;
    END IF;
    IF p_nome IS NULL THEN
      RETURN 3;
    END IF;
    IF length(p_nome) < 3 OR length(p_nome) > 120 THEN
      RETURN 4;
    END IF;
    SELECT COUNT(*) INTO v_count FROM produto WHERE nome = p_nome;
    IF v_count > 0 THEN
      RETURN 5;
    END IF;
    IF p_preco <= 0 OR p_preco >= 10000 THEN
      RETURN 6;
    END IF;
    INSERT INTO produto (codigo, nome, preco) VALUES (p_codigo, p_nome, p_preco);
    RETURN 0;
EXCEPTION
    WHEN OTHERS THEN
        GET STACKED DIAGNOSTICS v_sqlstate = RETURNED_SQLSTATE;
        RETURN v_sqlstate::integer;
END;
$$;
```

A função implementa validações de negócio e tratamento de erros, retornando códigos numéricos para diferentes cenários.

Cenário: Inserir um Produto

INSERT Direto

```
INSERT INTO produtos
(nome, preco)
VALUES ('Caneta Azul', 2.50);
```

Aplicação constrói a query completa e envia para execução

Função Armazenada

```
SELECT inserir_produto(
-1, 'Caneta Azul', 2.50
);
```

Aplicação chama função que encapsula a lógica de inserção

Conclusão: O Melhor dos Dois Mundos

01

Comece com Funções

Para operações principais da aplicação

02

Centralize Validação

Toda lógica de negócio em um lugar

03

Meça Performance

Compare com seu caso de uso real

04

Ajuste Conforme Necessário

Flexibilidade para otimizações específicas

Links úteis: Scripts completos no meu blog | Documentação PostgreSQL