Universidade Tecnológica Federal do Paraná

UTFPR - Campus Cornélio Procópio

AS33C
Banco de Dados 2

Professor: Eduardo Cotrin Teixeira

cotrin@utfpr.edu.br

UTFPR-CP / 2022

SQL – Stored Procedures e Functions

- Cada SGBD oferece um modo do usuário manter junto ao esquema do BD **funções** ou **procedimentos** que podem ser usados nas consultas ou em outros comandos SQL.

- Essas funções e procedimentos são escritos em linguagens simples, mas que nos permitem executar dentro do BD operações que não podem ser expressas em SQL.

SQL – Stored Procedures e Functions

- **Estrutura geral (SQL):**

CREATE PROCEDURE <nome> (<parâmetros>)
    <declarações locais>
    <corpo do procedimento> ;

CREATE FUNCTION <nome>(<parâmetros>) **RETURNS** <tipo>
    <declarações locais>
    <corpo da função> ;

SQL – Stored Procedures e Functions

- O PostgreSQL implementa *procedures* (procedimentos) somente a partir da versão 11, dando ênfase às *functions* (funções).
- Procedimento são executados por meio do comando *CALL*: **`CALL Proc1(parametro1);`**
- Funções são executadas por meio de comandos SQL:
  - Na cláusula SELECT: **`SELECT Func1(parametro1);`**
  - Na cláusula FROM: **`SELECT * FROM Func1();`**
  - Na cláusula WHERE:

```
SELECT * FROM tab WHERE Func1(par1)=42;
```

SQL – Stored Procedures e Functions

- Parâmetros de um procedimento:
  - Aparecem na forma **`<modo nome tipo>`**
  - Modos:
    - IN: define um parâmetro de entrada
    - OUT: define um parâmetro de saída *(PostGreSQL não implementa OUT)*
    - INOUT: define um parâmetro de entrada e saída

SQL – Stored Procedures e Functions

- **Exemplo**- Procedimento para aumentar o preço de uma peça:

```
CREATE PROCEDURE AumentaPreco (IN NroPeca CHAR(5),INOUT
Aumento FLOAT) AS $$
BEGIN
    UPDATE Peca
    SET PePreco = PePreco*Aumento
    WHERE Peca.PeNro= NroPeca;
   SELECT PePreco FROM Peca WHERE Peca.PeNro = NroPeca
          INTO Aumento;
END;$$LANGUAGE PLPGSQL;
```

-- Chamando o procedimento:

```
    CALL AumentaPreco('PE4',1.1)
```

SQL – Stored Procedures e Functions

- Parâmetros de uma função:
  - Aparecem na forma **<nome tipo>**
  - Um parâmetro de uma função é sempre do tipo IN
  - A única forma de obter informações de uma função é por meio de seu valor de retorno

SQL – Stored Procedures e Functions

- Há 4 tipos de funções no PostGreSQL:
  - **Funções escritas em SQL**
  - **Funções em linguagens procedurais** (**PL/pgSQL**, PL/php, etc.)
  - Funções internas (count(), avg(), max(), etc.)
  - Funções na linguagem C

SQL – Stored Procedures e Functions

- **Sintaxe geral:**

```
CREATE OR REPLACE FUNCTION <nome> (parâmetros)
RETURNS <tipo> AS
$$
DECLARE
    -- variáveis
BEGIN
    -- código
END;
$$ LANGUAGE linguagem;
```

SQL – Stored Procedures e Functions

- **Funções escritas em SQL puro:**
  - Total de peças fornecida a um projeto:

```
CREATE FUNCTION TotalPecas(NroProj CHAR(5))
RETURNS BIGINT AS $$
        SELECT Sum(Quant)
        FROM Fornece_para
        WHERE PNro = NroProj;
$$ LANGUAGE SQL;
```

Uso:

```
SELECT TotalPecas('P4');
```

TotalPecas
7

```
SELECT PNome,TotalPecas(PNro) FROM Projeto;
```

PNome	TotalPecas
Sea	7
Alfa	1
Detroit	1
Pegasus	1
Paraíso	5

SQL – Stored Procedures e Functions

- **Funções escritas em <u>SQL puro</u>:**
  - Compara total de peças de dois projetos:

```
CREATE FUNCTION AmaiorqB(ProjA CHAR(5),ProjB CHAR(5))
RETURNS BOOLEAN AS $$
      SELECT TotalPecas(ProjA) > TotalPecas(ProjB);
$$ LANGUAGE SQL;
```

Uso:

```
Select AmaiorqB('P4','P5'); --> True
```

SQL – Stored Procedures e Functions

- **Funções escritas em PL/pgSQL (exemplo):**
  - Custo mensal de um projeto:

```
CREATE OR REPLACE FUNCTION CustoMensal(NroProj CHAR(5))
RETURNS FLOAT AS $$
DECLARE
      Custo FLOAT;
      Duracao INTEGER;
      Mensal FLOAT;
BEGIN
      Duracao := (SELECT PDuracao FROM Projeto WHERE PNro=NroProj);
      Custo := (SELECT PCusto FROM Projeto WHERE PNro=NroProj);
      Mensal := Custo / Duracao;
      RETURN (Mensal);
END;
$$ LANGUAGE PLPGSQL; --> Uso da linguagem procedural
```

Obs.: Quando o resultado do SELECT é um conjunto vazio, o valor retornado é NULL.

- Uso:

```
SELECT CustoMensal('P3');
```

SQL – Stored Procedures e Functions

- **Mostre o nome e o custo mensal de cada projeto.**

```
SELECT PNome, CustoMensal(Pnro)
FROM Projeto;
```

PNome	CustoMensal
Sea	8600,00
Alfa	12333,33
Detroit	13350,00
Pegasus	7066,66
Paraíso	17000,00

SQL – Stored Procedures e Functions

- **Funções escritas em PL/pgSQL (exemplo):**
  - Nome de um projeto obtido pelo número:

```
CREATE OR REPLACE FUNCTION NomeProj(NroProj CHAR(5))
RETURNS TEXT AS $$
BEGIN
      RETURN (SELECT PNome FROM Projeto WHERE PNro=NroProj);
END;
$$ LANGUAGE PLPGSQL;
```

Para usar o Select direto no RETURN o valor retornado pelo Select deve ser compatível com o tipo de retorno da função.

- Uso:

```
SELECT NomeProj('P3'); --> Alfa
SELECT NomeProj('P13'); --> NULL
```

SQL – Stored Procedures e Functions

- **Mostrar o nome de cada projeto e a quantidade de peças fornecidas a ele.**

```
Exemplo de uso da função: SELECT NomeProj('P3');-->Alfa
```

SEM FUNÇÃO:

```
SELECT PNome,SUM(Quant)
FROM Fornece_para JOIN Projeto USING (PNro)
GROUP BY PNome;
```

PNome	SUM
Pegasus	1
Detroit	1
Sea	7
Paraíso	5
Alfa	1

COM FUNÇÃO:

```
SELECT NomeProj(PNro),SUM(Quant)
FROM Fornece_para
GROUP BY PNro;
```

SQL – Stored Procedures e Functions

- **Função que mostra o total de peças fornecidas pelo nome da peça.**

```
CREATE OR REPLACE FUNCTION TotalFornec(NomePeca CHAR(30))
RETURNS INT AS $$
BEGIN
      RETURN (SELECT SUM(Quant)
              FROM Fornece_para JOIN Peca USING (PeNro)
              WHERE PeNome = NomePeca);
END;
$$ LANGUAGE PLPGSQL;

Select TotalFornec('Volante');
```

SQL – Stored Procedures e Functions

- **Relatório com nome da peça e total de peças fornecidas.**

```
Exemplo de uso da função: Select TotalFornec('Volante');
```

SEM FUNÇÃO:

```
SELECT PeNome, SUM(Quant)
FROM Peca JOIN Fornece_para USING (PeNro)
GROUP BY PeNome;
```

COM FUNÇÃO:

```
SELECT PeNome, TotalFornec(PeNome)
FROM Peca;
```

PeNome	SUM
Lanterna	2
Cinto	5
Volante	2
Limpador	5
Painel	1