STORED FUNCTION

Definição - PL/pgSQL

PL – Procedural Language (Linguagem Procedural)

Para o desenvolvimento de sub-rotinas em PL/pgSQL é necessário o domínio e conhecimento da estrutura da linguagem de script usada pelo PostgreSQL.

Conceitos Importantes

Rotina: é um conjunto de instruções relativas a um cálculo ou função em um programa. Os termos rotina, sub-rotina e função são equivalentes.

Sub-rotina: é a parte de uma rotina que realiza uma tarefa especifica em um programa de computador.

Função: (reflete um stored function) é similar à descrição de sub-rotina com a característica de sempre retornar um valor de resposta e de ser executada manualmente.

Gatilho: é similar à descrição de sub-rotina com a característica de ser executado automaticamente quando uma condição preestabelecida é satisfeita. É utilizado para associar de forma automatica uma função armazenada a um evento de banco de dados.

Estrutura da Linguagem PL/pgSQL

CREATE [OR REPLACE] FUNCTION nome ([tipo]) **RETURNS** retor **AS** definição

DECLARE
    <variável> <tipo>;

BEGIN
    instruções;
    RETURN <variável>;
END;

definição **LANGUAGE** nome_ling;

Estrutura da Linguagem PL/pgSQL

nome: é o nome da função;

tipo: é a definição do tipo de dados dos parâmetros da função, caso eles existam;

retorno: é a definição do tipo de dado a ser retornado pela função;

definição: é uma String que contém a identificação da definição de uma função (Delimitador de fragmento de função);

nome_ling: é a definição do nome de uma linguagem procedural;

Os tipos de dados de uma variável em PL/pgSQL são: BIGINT, BIGSERIAL, CHAR, DATE, DECIMAL, DOUBLE, INTEGER. MONEY, NUMERIC, SERIAL, SMALLINT, TIME e VARCHAR.

Estrutura da Linguagem PL/pgSQL

IF <(condição)> **THEN**
    Instruções a serem executadas
[**ELSEIF** <(condição)> **THEN**
    Instruções a serem executadas]
[**ELSE**
    Instruções a serem executadas]
END IF;

Estrutura da Linguagem PL/pgSQL

LOOP
  instruções
END LOOP;

WHILE (condição) **LOOP**
  instruções
END LOOP;

FOR variável **IN** faixa **LOOP**
  instruções
END LOOP;

FUNÇÕES DEFINIDAS/STORED FUNCTIONS

```
CREATE FUNCTION sf_saudacao1() RETURNS
VARCHAR AS $$
DECLARE
    mensagem VARCHAR := 'Alo Mundo . . .';
BEGIN
   RETURN mensagem;
END;
$$ LANGUAGE plpgsql;

SELECT sf_saudacao1();
```

FUNÇÕES DEFINIDAS/STORED FUNCTIONS

```
CREATE FUNCTION sf_saudacao2(nome
VARCHAR) RETURNS VARCHAR AS $$
DECLARE
    mensagem VARCHAR := 'Ola ';
 BEGIN
   RETURN mensagem || '  ' || nome;
END;
$$ LANGUAGE plpgsql;

SELECT sf_saudacao2('Delta');
```

FUNÇÕES DEFINIDAS/STORED FUNCTIONS

```
CREATE FUNCTION sf_saudacao3(VARCHAR) RETURNS
VARCHAR AS $$
DECLARE
    mensagem VARCHAR := 'Ola ';
 BEGIN
   RETURN mensagem || '  ' || $1;
END;
$$ LANGUAGE plpgsql;


SELECT sf_saudacao3('Delta');
```

FUNÇÕES DEFINIDAS/STORED FUNCTIONS

```
CREATE FUNCTION sf_soma(A INTEGER, B INTEGER)
RETURNS INTEGER AS $$
DECLARE
    S INTEGER;
BEGIN
   S := A + B;
   RETURN S;
END;
$$ LANGUAGE plpgsql;


SELECT sf_soma(1,3);
```

FUNÇÕES DEFINIDAS/STORED FUNCTIONS

```
CREATE FUNCTION sf_loop() RETURNS INTEGER AS $$
DECLARE
    cont INTEGER;
BEGIN
   cont := 1;
   LOOP
       IF (cont = 30) THEN
           EXIT;
       END IF;
       cont := cont + 1;
   END LOOP;
   RETURN cont;
END;
$$ LANGUAGE plpgsql;

SELECT sf_loop();
```

FUNÇÕES DEFINIDAS/STORED FUNCTIONS

```
CREATE FUNCTION sf_while() RETURNS INTEGER AS $$
DECLARE
    cont INTEGER;
BEGIN
   cont := 1;
   WHILE (cont < 30) LOOP
       cont := cont + 1;
   END LOOP;
   RETURN cont;
END;
$$ LANGUAGE plpgsql;

SELECT sf_while();
```

FUNÇÕES DEFINIDAS/STORED FUNCTIONS

```
CREATE FUNCTION sf_for() RETURNS INTEGER AS $$
DECLARE
    valor INTEGER;
BEGIN
   valor := 0;
   FOR cont IN 1..30 LOOP
       valor := valor + 1;
   END LOOP;
   RETURN valor;
END;
$$ LANGUAGE plpgsql;

SELECT sf_for();
```

FUNÇÕES DEFINIDAS/STORED FUNCTIONS

```
CREATE FUNCTION sf_nome_mes() RETURNS VARCHAR(9) AS
$$
    DECLARE valor INTEGER;
                  nome VARCHAR(9);
 BEGIN
   valor := EXTRACT(MONTH FROM CURRENT_DATE);
   IF (valor = 1) THEN
       nome := 'Janeiro';
   ELSEIF (valor = 2) THEN
       nome := 'Fevereiro';
   ELSE
       nome := 'Mar ... Dez';
   END IF;
   RETURN nome;
END;
$$ LANGUAGE plpgsql;
```

FUNÇÕES DEFINIDAS/STORED FUNCTIONS

Observa agora, que é possível usar stored functions para acessar tabelas no Banco de Dados. Automatizando assim, funções comuns a algumas tabelas do Banco de Dados.

FUNÇÕES DEFINIDAS/STORED FUNCTIONS

```
CREATE FUNCTION sf_demite(codFuncionario
INTEGER) RETURNS VARCHAR AS $$
BEGIN
   UPDATE funcionario SET tipo_enquadramento = 'D'
WHERE id_funcionario = codFuncionario;
   RETURN 'Funcionário Demitido Com Sucesso !!!!';
END;
$$ LANGUAGE plpgsql;



SELECT sf_demite(2);
```

FUNÇÕES DEFINIDAS/STORED FUNCTIONS

```
CREATE OR REPLACE FUNCTION
sf_calcula_salario(codFuncionario INTEGER, valSalario
FLOAT) RETURNS VOID AS $$

DECLARE

   valContrib FLOAT := 300;

BEGIN

   UPDATE funcionario SET salario = (valSalario +
valContrib) WHERE id_funcionario = codFuncionario;

END;

$$ LANGUAGE plpgsql;

SELECT sf_calcula_salario(1,2000);
```

EXERCÍCIOS

1. Crie a Tabela

```
CREATE TABLE aluno
(
  id serial NOT NULL,
  nome character varying(60),
  nota_n1 numeric(5,2),
  nota_n2 numeric(5,2),
  nota_media_final numeric(5,2),
  CONSTRAINT "PK_aluno" PRIMARY KEY (id)
)
```

EXERCÍCIOS

2. Insira alguns dados na tabela “aluno”, através da execução dos comandos listados a seguir:

```
INSERT INTO aluno (nome) VALUES ('Renato Soares');
INSERT INTO aluno (nome) VALUES ('Aline Fernandes');
INSERT INTO aluno (nome) VALUES ('Roberta Silva');
INSERT INTO aluno (nome) VALUES ('Luana do Carmo');
INSERT INTO aluno (nome) VALUES ('Renato Fernandes');
```

EXERCÍCIOS

3. Crie uma STORED FUNCTION que atualize os campos:

- Nota N1: 8
- Nota N2: 10
- Nota Média Final: “Calculado Pela Função” (N1 * 0,40 + N2 * 0,60)
- **Vale lembra que esta Função, terá como parâmetros: idAluno (2), n1 (8), n2 (9) e não retornará nenhum valor.**

EXERCÍCIOS

4. Crie uma STORED FUNCTION que atualize os campos:

- Nota N1: 8
- Nota N2: 10
- Nota Média Final: “Calculado Pela Função” (N1 * 0,40 + N2 * 0,60)
- Retorne se o aluno **passou sem N3** ou **irá fazer N3 e qual será a nota mínima que o mesmo terá que tirar**.
- **Vale lembra que esta Função, terá como parâmetros: idAluno (2), n1 (8), n2 (9) e não retornará VARCHAR.**