PL-pgSQL - Módulo I – parte 2

- **PL-pgSQL – Procedural Language Functions**

  - **Mais flexível que a linguagem SQL permite a utilização de variáveis e estruturas de controle.**

PL-pgSQL – Procedural Language Functions

- **Sintaxe**

```
CREATE [or REPLACE] FUNCTION nome_funcao(par_1 tipo,
    par_2 tipo ... par_n)
RETURNS [SETOF] [VOID || DATATYPE || ESCALAR] AS  $$
DECLARE
BEGIN
RETURN;
END;
$$ LANGUAGE 'PLpgSQL';
```

PL-pgSQL – Procedural Language Functions

Executando uma função

```
SELECT nome_funcao(parametros);
```

ou

```
SELECT * FROM nome_Funcao(parâmetros);
```

Excluindo uma função

```
DROP FUNCTION nome_funcao(parâmetros);
```

PL-pgSQL – Procedural Language Functions

Exemplo de Função

```
CREATE or REPLACE FUNCTION fn_exemplo_a() RETURNS integer AS $$
<<bloco_externo>> -- bloco nomeado
DECLARE
_qtd integer := 30;
BEGIN
RAISE NOTICE 'Quantidade aqui vale: %', _qtd; -- 30
_qtd := 50;
--***************
-- subbloco
--***************
DECLARE
_qtd integer := 80;
BEGIN
RAISE NOTICE 'Quantidade (interno) aqui vale: %', _qtd; -- 80
RAISE NOTICE 'Quantidade (externo) aqui vale: %', bloco_externo._qtd; -- 50
END;
RAISE NOTICE 'Quantidade (externo) aqui vale: %', _qtd; -- 50
RETURN _qtd;
END;
$$ LANGUAGE plpgsql;
```

PL-pgSQL – Procedural Language Functions

Para executar a Função:

```
select fn_exemplo_a();
```

PL-pgSQL – Procedural Language Functions

- **Declaração de variáveis**

Abaixo alguns exemplos para declaração de variáveis em PostGreSQL

...

```
DECLARE
a integer;
b numeric(5);
c varchar(50);
linha tabela%ROWTYPE; -- declara variável que terá a mesma estrutura de uma linha inteira da tabela
```

PL-pgSQL – Procedural Language Functions

campo tabela.coluna%TYPE; -- declara variável de mesmo tipo da coluna da tabela

rec RECORD; -- declara variável de tipo registro, muito semelhante a variável linha, porém sem tipos definidos

d integer NOT NULL DEFAULT:=80; -- variável inteira que não pode ser nula, recebe por padrão o valor 80

e CONSTANT integer := 10; -- constante inteira inicializada com valor 10

- ...

PL-pgSQL – Procedural Language Functions

Passagem de parâmetros

Como qualquer outra linguagem de programação podemos passar parâmetros para um procedimento.

Exemplo:

```
Create or Replace function fn_exemplo_parametros(in_a int, in_b int) returns decimal(8,2)
AS $$
DECLARE
_result decimal(8,2);
BEGIN
_result := in_a / in_b;
return _result;
END;
$$ language 'PlpgSQL‘;
```

PL-pgSQL – Procedural Language Functions

Para executar com os parâmetros

select fn_exemplo_parametros(100,3)

Por quê nunca há número com casas decimais?

PL-pgSQL – Procedural Language Functions

```
create or replace function fn_exemploTipos(p_cod_cli integer) returns varchar(50) as
$$
declare
_cliente cliente%rowtype; -- registro do tipo cliente (possui todos as colunas da tabela cliente e irá armazenar
      uma linha
_cod_cli cliente.cod_cli%type;
_msg varchar(80);
begin
select * into _cliente from cliente where cod_cli = p_cod_cli;
raise notice '%', 'Atribui um valor a variável do tipo type';
_cod_cli := 'c1';
if _cod_cli = _cliente.cod_cli then
raise notice '%', 'Se cliente for igual a c3 listar o nome do cliente c1';
_msg := 'Cliente já existe : ';
select * into _cliente from cliente where cod_cli = _cod_cli;
end if;
return _msg || _cliente.nome;
end;
$$ language 'plpgsql';
```

PL-pgSQL – Procedural Language Functions

Para executar o exemplo

select * from fn_exemploTipos('c1');

Qual(ais) o(s) problema(s) com a função acima?

PL-pgSQL – Procedural Language Functions

Exercício 01:

- **Conforme as tabelas abaixo escreva um procedimento que atenda ao solicitado:**
  - **O procedimento deve receber dois valores por parâmetro CNH do motorista e velocidadeApurada do veículo por ele conduzido;**
  - **O procedimento deve retornar um texto com a seguinte mensagem:**
    - 'O motorista [nome] soma [X] pontos em em multas ‘;

PL-pgSQL – Procedural Language Functions

- **O cálculo da pontuação do motorista é efetuado da seguinte forma: -**
  - **Se a velocidade estiver entre 80.01 e 110 então o motorista deve ser multado em 120,00 e receber 20 pontos**
  - **Se a velocidade estiver entre 110.01 e 140 então o motorista deve ser multado em 350 e receber 40 pontos**
  - **Se a velocidade estiver eacima de 140 então o motorista deve ser multado em 680 e receber 60 pontos**
- **O sistema deve considerar somente 90% da velocidade apurada para o cálculo da multa.**

PL-pgSQL – Procedural Language Functions

- **Após o cálculo o sistema deve incluir a multa na tabela ex_multa (se o contribuinte foi multado)**
- **Então retornar o total acumulado de multas para o motorista.**

PL-pgSQL – Procedural Language Functions

```
create table ex_motorista
(cnh char(5) primary key,
nome varchar(20) not null,
totalMultas decimal(9,2) );
```

PL-pgSQL – Procedural Language Functions

```
create table ex_multa
(id serial primary key,
cnh char(5) references ex_motorista(cnh) not null,
velocidadeApurada decimal(5,2) not null,
velocidadeCalculada decimal(5,2),
pontos integer not null,
valor decimal(9,2) not null);
```

- insert into ex_motorista values ('123AB', 'Carlo');

PL-pgSQL – Procedural Language Functions

Exercício 02:

- **Escreva um outro procedimento que atualize o campo totalMultas da tabela ex_motorista a partir dos totais apurados para cada motorista autuado na tabela ex_multa.**
- **OBS1: motorista sem multa deverão possuir valor 0.00 no campo total multa;**
- **OBS2:cuidado para não duplicar valores na coluna totalMultas para os casos em que a rotina for disparada mais de uma vez.**