PL-pgSQL - Módulo III

- **PlPgSql – Estruturas de Controle:**
- **Com os comandos de LOOP, EXIT, CONTINUE, WHILE e FOR , podemos repetir uma série de comandos na função.**

PL-pgSQL – Estrutura de Comandos

- **Exemplo do comando IF**

```
IF number = 0 THEN
   result := 'zero';
ELSIF number > 0 THEN
   result := 'positive';
ELSIF number < 0 THEN
   result := 'negative';
ELSE
   -- hmm, the only other possibility is that number is null
   result := 'NULL';
END IF;
```

- **Comando LOOP: permite um LOOP incondicional de maneira repetitiva para um bloco de comandos, até que seja interrompido pelo comando EXIT ou RETURN**

```
[ <<label>> ]
 LOOP
        statements
END LOOP [ label ];
```

PL-pgSQL – Estrutura de Comandos

- **Comando EXIT: Comando de desvio incondicional. Quando pertencente a um bloco de comandos (begin) passa fluxo para o próximo comando após o final do bloco (end).**

```
EXIT [ label ] [ WHEN expression ];
```

- **Exemplo do comando EXIT**

```
BEGIN
   IF stocks > 100000 THEN
      EXIT;  -- causa saída do bloco  BEGIN
   END IF;
END;
```

PL-pgSQL – Estrutura de Comandos

- **Comando CONTINUE: Comando de desvio incondicional. Permite uma nova iteração a partir de sua execução**

```
CONTINUE [ label ] [ WHEN expression ];
```

- **Exemplos do comando LOOP**

```
LOOP
    ……
    IF count > 0 THEN
       EXIT;
   END IF;
END LOOP;

LOOP
   ……
   EXIT WHEN count > 0;
END LOOP;

LOOP
     ……
     EXIT WHEN count > 100;
   CONTINUE WHEN count < 50;
END LOOP;
```

PL-pgSQL – Estrutura de Comandos

- **Comando WHILE: Comando iterativo condicional. Testa uma expressão a cada loop e somente executa caso true.**

```
[ <<label>> ]
WHILE expression LOOP
   statements
END LOOP [ label ];
```

Exemplos do comando WHILE

```
WHILE amount_owed > 0 AND gift_certificate_balance > 0 LOOP
   -- some computations here
END LOOP;

WHILE NOT boolean_expression LOOP
   -- some computations here
END LOOP;
```

PL-pgSQL – Estrutura de Comandos

- **Comando FOR: Comando iterativo limitado de maneira condicional.**

```
[ <<label>> ]
FOR name IN [ REVERSE ] expression .. expression [ BY expression ] LOOP
    statements
END LOOP [ label ];
```

Exemplos do comando FOR

```
FOR i IN 1..10 LOOP
    -- some computations here
    RAISE NOTICE 'i is %', i;
END LOOP;

FOR i IN REVERSE 10..1 LOOP
    -- some computations here
END LOOP;

FOR i IN REVERSE 10..1 BY 2 LOOP
    -- some computations here
    RAISE NOTICE 'i is %', i;
END LOOP;
```

PL-pgSQL – Estrutura de Comandos

- **Comando FOR-SQL: Comando iterativo limitado de maneira condicional, usando um comando de SQL-consulta.**

```
[ <<label>> ]
FOR target IN query LOOP
   statements
END LOOP [ label ];
```

Exemplos do comando FOR-SQL

```
CREATE or REPLACE FUNCTION listaClientes() RETURNS integer AS $$
DECLARE
   _clientes RECORD;
BEGIN
   raise notice 'Listando Clientes...';

   FOR _clientes IN SELECT * FROM cliente ORDER BY nome LOOP

      raise notice 'Cliente: % ', _clientes.nome;
   END LOOP;

   raise notice 'Lista Concluída.';
   RETURN 1;
END;
$$ LANGUAGE plpgsql;
```

PL-pgSQL – Chamada de Funções

- **Chamando uma função a partir de outra: Podemos facilmente "quebrar" nossos procedimentos em procedimentos menores, seja para melhorar a estrutura (manutenção) ou compreensão do mesmo.**

Exemplo: PROCEDIMENTO I

```
CREATE OR REPLACE FUNCTION FN_OBTER_TOTAL_MULTA(pCNH CHAR(5)) RETURNS DECIMAL(5,2) AS
$$
DECLARE
	_TOTAL DECIMAL(5,2) :=0;
BEGIN

SELECT COALESCE(SUM(VALOR)) AS VALOR
INTO _TOTAL FROM EX_MULTA WHERE CNH = pCNH;

RETURN _TOTAL;

END;
$$ LANGUAGE PLPGSQL
```

PL-pgSQL – Chamada de Funções

Exemplo: PROCEDIMENTO II

```
-- PROCEDIMENTO Ii
CREATE OR REPLACE FUNCTION FN_ATUALIZAR_TOTAL_MULTA(pCNH CHAR(5)) RETURNS VOID AS
$$
BEGIN

IF pCNH IS NULL THEN
        RAISE EXCEPTION 'CNH NÃO PODE SER NULO';
END IF;

IF (SELECT FN_OBTER_TOTAL_MULTA(pCNH)) > 0 THEN
        UPDATE EX_MOTORISTA
        SET TOTALMULTAS = (SELECT FN_OBTER_TOTAL_MULTA(pCNH))
        WHERE CNH = pCNH;
END IF;

RETURN;
END;
$$
LANGUAGE PLPGSQL;

SELECT FN_ATUALIZAR_TOTAL_MULTA('123AB');
```

PL-pgSQL - Chamada de Funções

Exercício: Use o modelo ex_motorista e ex_multa

- **Monte um procedimento o qual possa gerenciar as mensagens de erro fm_msg() o qual receba um tipo de erro e gere uma respectiva mensagem;**
- **Crie um procedimento fm_lista() o qual liste os motoristas (nome) e seu total de multas, caso o parametro sejam TODOS ou apenas de um CNH;**
- **Monte um procedimento fm_totalMultas() o qual retorne o total de multas de cada motorista. Use na função fm_lista()**
- **OBS: use o procedimento fm_msg() para que gere mensagens do tipo:**
  - **Motorista não cadastrado;**
  - **Motorista sem multas, etc**