PL/SQL – Parte 2

Curso de Introdução a Oracle 11g: SQL/Avançado

Prof.: Marlon Mendes Minussi
marlonminussi@gmail.br

PROGRAMAÇÃO PL/SQL

- **ATRIBUTOS ESPECIAIS**
- As variáveis PL/SQL e constantes possuem atributos, que são propriedades que permitem a referência ao tipo e estrutura do objeto sem necessidade de repetição de sua definição.
- As tabelas e colunas do database possuem atributos similares, que podemos usar para facilitar a manutenção.

PROGRAMAÇÃO PL/SQL

- %TYPE este atributo copia os atributos de uma variável, constante ou coluna do database. É particularmente usado quando declaramos variáveis que pertençam as colunas do database.
- **sintaxe**: <variável>/<constante>/<coluna>%TYPE
- Ex.:

```
DECLARE
  V_COD_PAC   NUMBER(3)
  V_COD_PAC2  V_COD_PAC %TYPE;- variável de tipo idêntico a código
  V_COD_ATENDIMENTO ATENDIMENTO. COD_ATENDIMENTO%TYPE
   /*variável de tipo idêntico à variável da base de dados cod_atendimento*/
...
```

PROGRAMAÇÃO PL/SQ

- %ROWTYPE este atributo gera um tipo de registro que representa uma linha da tabela.
- O registro pode armazenar uma linha de dados selecionados da tabela ou recebidos de um cursor.
- **sintaxe:** <tabela>/<cursor>%ROWTYPE

PROGRAMAÇÃO PL/SQ

- Ex.:

```
DECLARE
    V_ATEND_ROW    ATENDIMENTO%ROWTYPE; - variável do tipo row
    V_COD_PAC      INTEGER;
    CURSOR V_C1 IS                 - cursor com apenas 2 colunas
    SELECT cod_atendimento, dt_atedimento
    FROM    atendimento;                    - colunas da tabela
    V_C1_ROW       V_C1%ROWTYPE;- possui as mesmas colunas de V_C1
BEGIN
 ...
```

Cursor

- Um Cursor representa uma tabela temporária que pode ser processada de maneira seqüencial, ou seja, linha a linha.
- Os cursores são áreas compostas de linhas e colunas armazenadas em memória que servem para armazenar o resultado de uma seleção.
- Com o uso de cursores é possível selecionar um conjunto de linhas e manipular o resultado dessa consulta linha a linha, dentro de um código PL/SQL (Program Language SQL), como stored procedure, function e triggers.

Cursor

- O cursor deve ser declarado na cláusula DECLARE de um código PL/SQL:
- Sintaxe:

```
CURSOR cursor_name [(parameter[,parameter]...)]
    IS select_statement
 [FOR UPDATE OF colunas];
```

- Vamos analisar um exemplo de cursor denominado c_medicos que permite listar o nome de todas médicos cadastrados.

Cursor

OBS:
- **ROWTYPE**, é como um vetor, descreve uma linha, por exemplo, o resultado de uma consulta. para usar em outro select, em uma tabela ou view.
- **FETCH,** retorna as linhas utilizando um cursos previamente criado.
- **%NOTFOUND,** indica se o último FETCH retornou uma row ou não.

- Ex.:

```
DECLARE
        /*DECLARANDO O CURSOR  DE MEDICOS*/
        CURSOR c_medico IS
                        SELECT * FROM medico;
        /*DECLARANDO UMA VARIAVEL QUE SERA O REGISTRO DA TABELA*/
        reg_medico c_medico%ROWTYPE;
BEGIN
        /*ABRE CURSOR*/
        OPEN c_medico;
        LOOP
                /*LÊ UM REGISTRO DO CURSOR*/
                FETCH c_medico INTO reg_medico;
                /*ABANDONA O LOOP CASO SEJA O FINAL DO CURSOR*/
                EXIT WHEN c_medico%NOTFOUND;
                DBMS_OUTPUT.PUT_LINE('Nome do Médico:'|| reg_medico.nome);
                 /*CODIGO QUE IRA MANIPULAR OS DADOS*/
        END LOOP;
        /*FECHA O CURSOR*/
        CLOSE c_medico;
END;
/
```

Cursor

- Vamos analisar o exemplo de um cursor de atualização chamado c_atendimento que permite listar o diagnostico de todos os atendimentos e também atualizar o desconto de todos os atendimentos para 15 reais.

```
DECLARE
  /*DECLARANDO O CURSOR  DE ATENDIMENTO*/
  CURSOR c_atendimento IS
       SELECT * FROM atendimento
       FOR UPDATE OF desconto;
   /*DECLARANDO UMA VARIAVEL QUE SERA O REGISTRO DA TABELA*/
   reg_atendimento c_atendimento%ROWTYPE;
BEGIN
   /*ABRE CURSOR*/
   OPEN c_atendimento;
   LOOP
   /*LÊ UM REGISTRO DO CURSOR*/
      FETCH c_atendimento INTO reg_atendimento;
    /*ABANDONA O LOOP CASO SEJA O FINAL DO CURSOR*/
      EXIT WHEN c_atendimento%NOTFOUND;
      DBMS_OUTPUT.PUT_LINE('Diagnostico do Atendimento: '||
      reg_atendimento.diagnostico||CHR(10)||'Desconto: '|| reg_atendimento
      .desconto);
     /*CODIGO QUE IRA MANIPULAR OS DADOS*/
     UPDATE atendimento
          SET desconto = 15
          WHERE CURRENT OF c_atendimento;
    END LOOP;
    /*FECHA O CURSOR*/
    CLOSE c_atendimento;
END;
/
```

EXERCÍCIOS

1) Criar uma rotina PL/SQL que leia 3 notas e calcule a média aritmética das notas e exiba as seguintes mensagens conforme a situação:
   - Se Média > 5 aluno terá de fazer avaliação substitutiva;
   - Se Média >= 7 aluno aprovado;
   - Se Média <= 4 aluno reprovado.
2) Criar um código PL/SQL que define um cursor chamado lista_paciente, que liste o nome e a cidade de todos os pacientes cadastrados na tabela Paciente do banco de dados Clínica;

EXERCÍCIOS

3) Crie um cursor para a seguinte situação:
   - Suponha a existência de uma tabela (RESULTADO) com o layout (NOME VARCHAR2(30),VALOR NUMBER (10,2).
   - Deseja-se preenchê-la a partir da tabela atendimento e paciente, com o nome paciente(nome) e a expressão trunc(valor-desconto) na coluna valor.
4) Crie um cursor para a seguinte situação:
   - Com base no exercícios anterior faça a inserção na tabela RESULTADO, agrupando por paciente e somando os valores dos atendimentos.

SUBPROGRAMAS

- Subprogramas são blocos PL/SQL com nome, que podem receber parâmetros e ser invocados.
- O PL/SQL possui dois tipos de subprogramas chamados **procedures** e **functions**. Geralmente usa-se uma procedure para executar uma ação e uma função para calcular um valor.
- Da mesma forma que qualquer outro bloco (anônimo) PL/SQL, os subprogramas possuem uma parte declarativa, uma parte executável e uma parte opcional para tratamento de exceção.

Procedure ou Stored Procedure

- Uma procedure corresponde a um bloco PL/SQL nomeado.
- A vantagem sobre um bloco anônimo é que pode ser compilado e armazenado no banco de dados como um objeto do schema.
- As procedures ficam armazenadas no banco de dados e são muito utilizadas, principalmente pela sua rapidez de execução e por diminuírem o tráfego da rede, pois obedecem a um comando do cliente e executam a operação no servidor.

Procedure

- A sintaxe básica de uma procedure é:

```
CREATE [OR REPLACE] PROCEDURE [schema.] nome_da_procedure
[(parâmetro 1 [modo1] tipo 1,
        parâmetro 2 [modo2] tipo 2,
                ...)]
IS|AS
BEGIN
/*Bloco de instruções PL/SQL*/
END;
```

Procedure

- De acordo com a sintaxe tem-se (GOYA,2006b):
  - **REPLACE:** esse comando faz com que, caso a procedure exista, ela seja substituída.
  - **NOME_DA_PROCEDURE:** nome do procedimento.
  - **PARÂMETRO:** indica o nome da variável PL/SQL que é passada na chamada da procedure ou o nome da variável que retorna os valores da procedure ou ambos. O que conterá um parâmetro depende de MODO.
  - **MODO:** indica que o parâmetro é de entrada (IN) que é o default, saída (OUT) ou ambos (IN OUT).
  - **TIPO:** indica o tipo de dado do parâmetro. Pode se qualquer tipo de dado do SQL ou do PL/SQL. Não é possível fazer nenhuma restrição ao tamanho do tipo de dado neste ponto.
  - **IS|AS:** a sintaxe do comando tanto IS como AS. Por convenção usa-se IS na criação de procedures e AS para criar pacotes.
  - **BLOCO PL/SQL:** corresponde ao código que será executado pela procedure. Indica com a cláusula BEGIN e termina com o END ou END nome_da_procedure.

Procedure

- Vejamos um exemplo e procedure para ajudar o entendimento:

```
CREATE OR REPLACE PROCEDURE aumenta_desconto
(p_cod_pac IN atendimento.cod_pac%TYPE)
IS
BEGIN
UPDATE atendimento
SET desconto = desconto+(desconto * 0.15)
WHERE cod_pac = p_cod_pac;
END aumenta_desconto;
/
```

Procedure

- Neste exemplo criamos uma procedure para aumentar o desconto de um determinado paciente em 15%.
- A primeira linha define o nome da procedure, que se chamar AUMENTA_DESCONTO.
- A linha dois define o parâmetro p_cod.pac no modo IN. Ou seja, vai ser um dado informado na chamada da procedure.
- Em seguida determinamos que ele será do mesmo tipo e tamanho que a coluna cod_pac da tabela atendimento. Isso feito pela referência atendimento.cod_pac%TYPE.

Procedure

- Podemos verificar o estado da procedure por uma simples consulta:

```
SELECT object_name, status
FROM user_objects
WHERE object_name LIKE '%AUMENTA%'
```

  - Agora podemos verificar o funcionamento da procedure.

  - Primeiramente devemos selecionar os dados da tabela atendimento para verificar o valor dos seus descontos e em seguida executar o comando:

```
SELECT cod_pac, desconto
FROM atendimento;
```

Procedure

- Em seguida é preciso fazer chamada à stored e passar como parâmetro o código do paciente para efetuar o aumento do desconto em 15%.
- Neste caso pode-se usar o código do paciente 1, e para executar a chamada à stored pode-se executar um dos comandos mostrados a seguir;

```
CALL AUMENTA_DESCONTO(1);
ou
EXECUTE AUMENTA_DESCONTO(1);
```

Procedure

- Caso haja necessidade, é possível obter o código-fonte da procedure pelas views de dicionario de dados ALL_SOURCE, USER_SOURCE e DBA_SOURCE.
- Exemplo:

```
SELECT text
FROM user_source
WHERE name= 'AUMENTA_DESCONTO'
ORDER BY line;
```

Procedure

- Eventualmente pode-se precisar ver todas as procedures e funções do usuário. Neste caso pode-se usar;
- Exemplo:

```
SELECT object_name, object_type
FROM user_objects
WHERE object_type in ('PROCEDURE‘, 'FUNCTION')
ORDER BY object_name;
```

Procedure

- Caso precisemos apenas dos argumentos da procedure, o comando DESCRIBE (ou sua abreviatura DESC) permite identificá-los rapidamente.
- Exemplo:

  describe aumenta_desconto;

  Ou

  desc aumenta_desconto;

Exemplo

```
/* PROCEDURE QUE RETORNA NOME DO MÉDICO E SUA ESPECIALIDADE */
CREATE OR REPLACE PROCEDURE SP_DADOS_MED (v_cod_med IN medico.cod_med%TYPE,
    v_nome OUT medico.nome%TYPE,
    v_esp OUT medico.especialidade%TYPE)
    IS
    BEGIN
       IF (v_cod_med = NULL) OR (v_cod_med <= 0) THEN
           DBMS_OUTPUT.PUT_LINE('CÓDIGO INVÁLIDO!!!');
          ELSE
           BEGIN
               SELECT nome, especialidade
               INTO v_nome, v_esp
               FROM medico
               WHERE cod_med = v_cod_med;
            EXCEPTION
            WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE(‘MEDICO não encontrado!');
            WHEN TOO_MANY_ROWS THEN
            DBMS_OUTPUT.PUT_LINE('Mais de um médico com mesmo código’);
           END;
      END IF;
  END SP_DADOS_MED;
```

Exemplo

```
// --- EXECUTANDO A PROCEDURE ANTERIOR
DECLARE
   v_nome            medico.nome%TYPE;
   v_esp             medico.especialidade%TYPE;
BEGIN
  SP_DADOS_MED(2, v_nome, v_esp);
  DBMS_OUTPUT.PUT_LINE(v_nome);
  DBMS_OUTPUT.PUT_LINE(v_esp);
END;
/
Ou
      DECLARE
          v_nome medico.nome%TYPE;
          v_esp    medico.especialidade%TYPE;
      BEGIN
        SP_DADOS_MED(1, v_nome,v_esp);
      END;
      /
```

OBS: Terá de ser colocadas as linhas abaixo na Procedure

```
DBMS_OUTPUT.PUT_LINE(v_nome);
DBMS_OUTPUT.PUT_LINE(v_esp);
```

Procedure

- Para executar uma procedure Via SQL*Plus:

```
SQL>EXEC NOME_PROCEDURE(argum1,argum2,...,argumN);
```

Ou

```
SQL> Begin
        ...
            NOME_PROCEDURE(argum1,argum2,...,argumN);
        ...
         End;
```

- Exclusão

```
DROP PROCEDURE nome-procedure;
```

Functions

- As funções sempre retornam um valor:
- Sintaxe básica:

```
CREATE [OR REPLACE] FUNCTION nome_da_function
[(parâmetro 1 [modo1] datatype1,
        parâmetro 2 [modo2] datatype2,
            ...)]
IS|AS
BEGIN
/*Bloco de instruções PL/SQL*/
END;
```

Function

- Função para calcular INSS dos médicos.
- A tabela seguinte mostra as faixas de valores para o cálculo do INSS (Instituto Nacional de Seguridade Social).

Salário	Desconto
Até R$840,47	7,65%
De R$840,48 a R$1.050	8,65%
De R$1.050,01 a R$1.400,77	9,00%
De R$ 1.400,78 a R$2.801,56	11,0%
Acima de R$ 2.801,56	O desconto é de R$308,16

Function

- Descreve-se a seguir a função f_calcula_inss que recebe como parâmetro o salário do funcionário e retorna o valor do desconto de INSS.

```
CREATE OR REPLACE FUNCTION f_calcula_inss (p_salario IN NUMBER)
    RETURN NUMBER IS
BEGIN
    If (p_salario <= 840.47) then
      RETURN (p_salario * 0.0765);
    ElsIf (p_salario >= 840.48) and (p_salario <= 1050.00) then
      RETURN (p_salario * 0.0865);
    ElsIf (p_salario >= 1050.01) and (p_salario <= 1400.77) then
      RETURN (p_salario * 0.09);
    ElsIf (p_salario >= 1400.78) and (p_salario <= 2801.56) then
      RETURN (p_salario * 0.11);
    ElsIf (p_salario > 2801.56) then
      RETURN (308.16);
    END IF;
END f_calcula_inss;
/
```

Function

- Para efetuar a chamada à função pode-se usar uma expressão SQL:
- Ex.:

```
SELECT cod_med, nome, salario, f_calcula_inss (salario)
FROM medico;
```

Function

- Ou também pelo código mostrado a seguir:

```
DECLARE
  retorno1 NUMBER;
BEGIN
  RETORNO1 := f_calcula_inss (3000);
  DBMS_OUTPUT.PUT_LINE(retorno1);
END;
```

Function

- Ou também usando SQLPlus:

```
VARIABLE retorno_inss NUMBER

EXECUTE :retorno_inss := f_calcula_inss(1000);

PRINT retorno_inss
```

Exercícios

1) Criar um bloco anônimo não rotulado de saída PL/SQL onde mostre o salário de todos os médicos que acrescido de 10% e o valor do acréscimo.

2) Criar um bloco anônimo não rotulado que altere o salário em 10% para todos os médicos.

3) Criar uma função que calcula a média dos atendimentos de um determinado data:

Exercícios

4) Criar um bloco anônimo não rotulado que conforme o salário dos pacientes exiba as seguintes mensagens:
   - Se salario <= 500 exiba mensagem:= 'Este paciente ganha menor ou igual a salário mínimo.';
   - Se salario > 500 e salario < 1200 'Este paciente ganha acima do mínimo, mas não terá desconto de imposto de renda.';
   - Se salario >= 1200 'Este paciente ganha um salário que terá desconto de imposto de renda.';

   Resolva de duas maneiras:

   a) Utilizando FOR ou WHILE;

   b) Utilizando CURSOR.

Exercícios

5) Criar um Procedimento PL/SQL que leia 3 notas e calcule a média aritmética das notas e exiba as seguintes mensagens conforme a situação:
   - Se Média >=4 aluno terá de fazer avaliação substitutiva;
   - Se Média >=7 aluno aprovado;
   - Se Média <4 aluno reprovado.
6) Crie uma função que calcule a Fatorial de um número.

- Dica:
  - Para que seja calculado o fatorial, deve-se utilizar uma estrutura de repetição (looping);
  - A Fórmula para calcular a fatorial de um número é:
    - Ex: 5! = 1X2X3x4x5 = 120