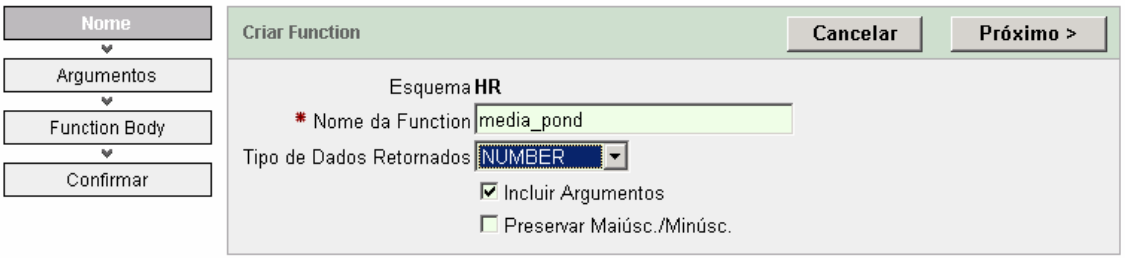
Tutorial PL/SQL

O PL/SQL é uma linguagem procedural que roda diretamente no núcleo do SGBD Oracle. O objetivo deste tutorial é mostrar a criação de funções e procedimentos em PL/SQL, interagindo com comandos SQL padrões.

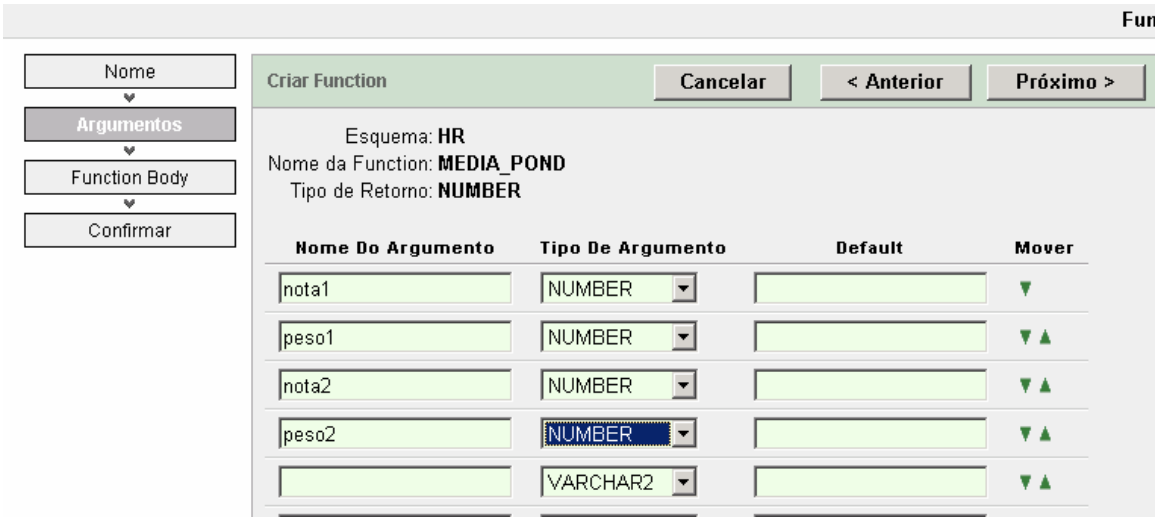
Logado no OracleXE vamos acessar a opção:

Vamos construir uma função que calcula a média ponderada. Esta função terá como parâmetros quatro valores numéricos e irá retornar o calculado. Acompanhe a criação desta função:

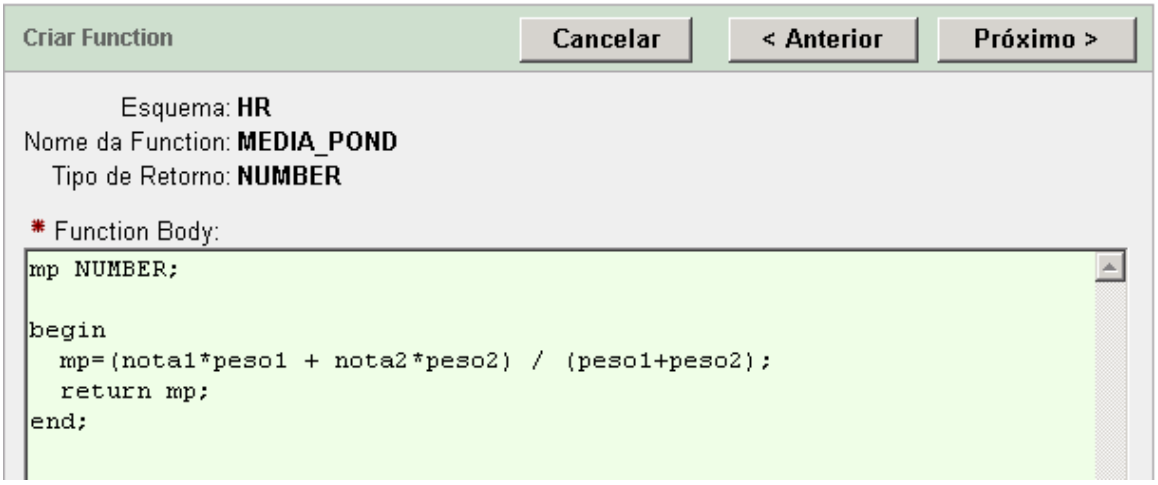
Passo-1: Nome da função e tipo de retorno:



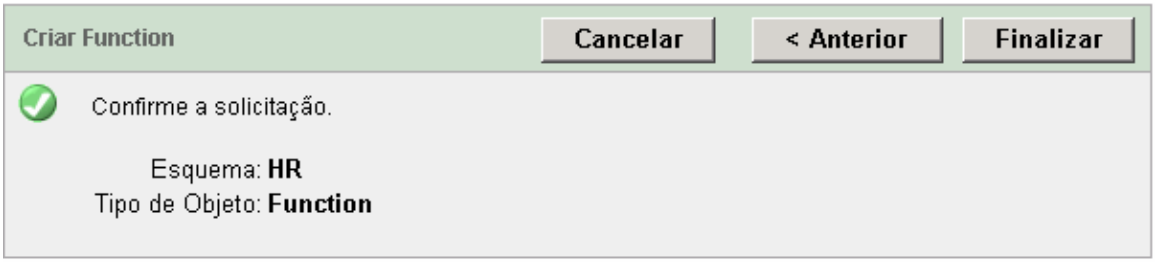
Passo-2: Definição dos parâmetros de entrada da função:



Passo-3: Codificação do corpo da função (lógica de programação):



Passo-4: Vistoria do código gerado e finalização do processo:



SQL

```
create or replace function "MEDIA_POND"
(nota1 in NUMBER,
peso1 in NUMBER,
nota2 in NUMBER,
peso2 in NUMBER)
return NUMBER
is
mp NUMBER;

begin
  mp=(nota1*peso1 + nota2*peso2) / (peso1+peso2);
  return mp;
end;
/
```

Passo-5: O código final deverá ser:

```
1  create or replace function "MEDIA_POND"
2  (nota1 in NUMBER,
3  peso1 in NUMBER,
4  nota2 in NUMBER,
5  peso2 in NUMBER)
6  return NUMBER
7  is
8  mp NUMBER;
9
10 begin
11   mp:=(nota1*peso1 + nota2*peso2) / (peso1+peso2);
12   return mp;
13 end;
```

Vamos analisar o código fonte:

CREATE OR REPLACE FUNCTION “MEDIA_POND”

Estamos criando ou substituindo uma função cujo nome é MEDIA_POND.

(nota1 in NUMBER, peso1 in NUMBER, ...)

É a declaração de parâmetros de entrada, no momento do uso da função deverão ser fornecidos quatro números.

return NUMBER

É o valor de retorno da função, ao final do seu processamento ela deve retornar a quem chamou um número.

is

Indica que a escrita da função irá começar.

mp NUMBER;

Estamos declarando uma variável de escopo local do tipo número para ser usada durante o processo de calculo.

mp:=(nota1*peso1 + nota2*peso2) / (peso1 + peso2);

Armazena temporariamente na variável numérica “mp” o valor calculado.

return mp;

Retorna para quem chamou o valor da média ponderada.

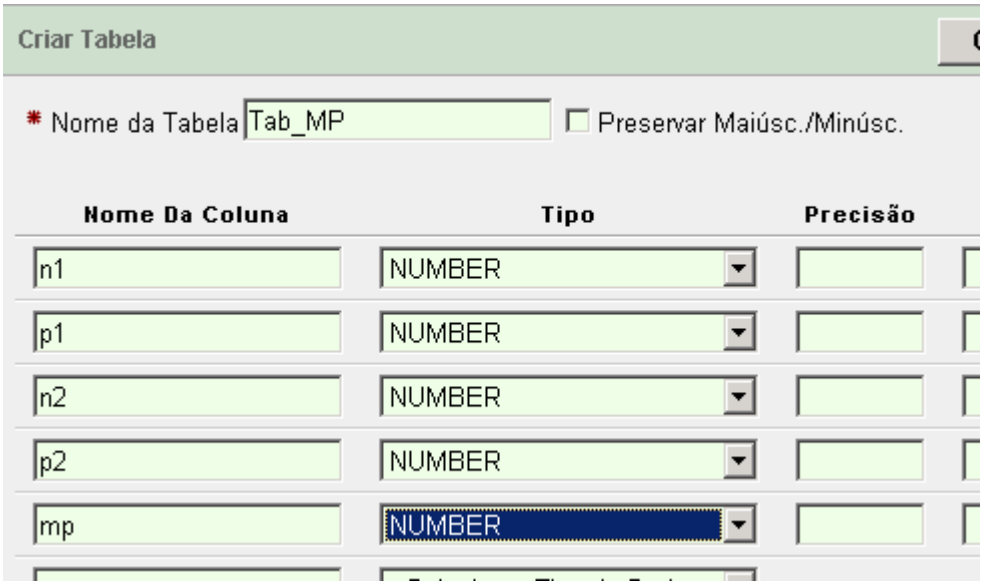
Podemos fazer ainda algumas colocações:

- O PL/SQL não é case-sensitive.
- Os tipos de dados e comandos SQL são compartilhados com o PL/SQL
- Os blocos ficam armazenados dentro do banco de dados.

Vamos agora testar a função:

Passo-1: Criar uma tabela para os testes:

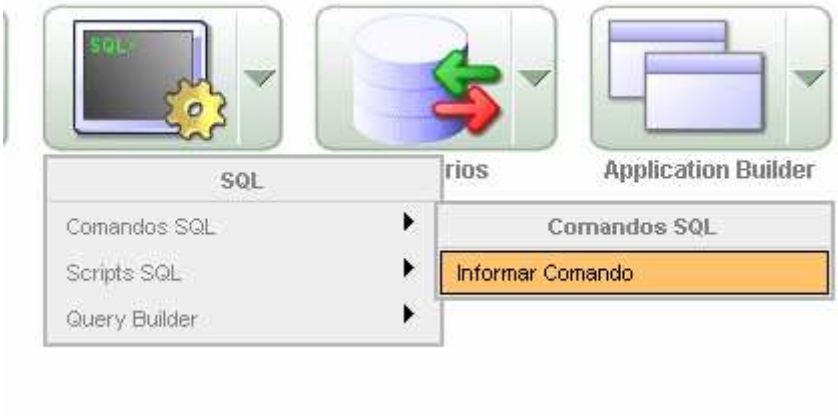
Passo-2: Definir os atributos da tabela (Tab_MP):



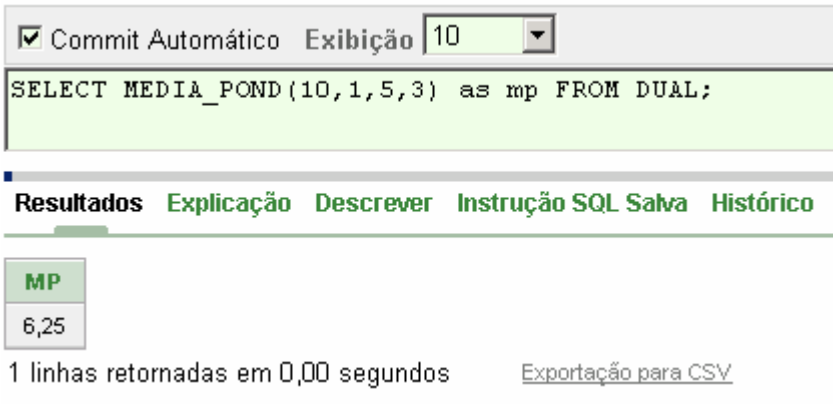
- Pode confirmar os outros passos sem alterações até a criação da tabela (Tab_MP).

Passo-3: Testar no SQL a função criada:

a) Chamar o prompt de comando SQL:



b) Testar com literais:



c) Inserir dados na tabela de testes:

INSERT INTO Tab_MP (n1, p1, n2, p2) VALUES (10, 1, 5, 2);

INSERT INTO Tab_MP (n1, p1, n2, p2) VALUES (6, 1, 8, 2);

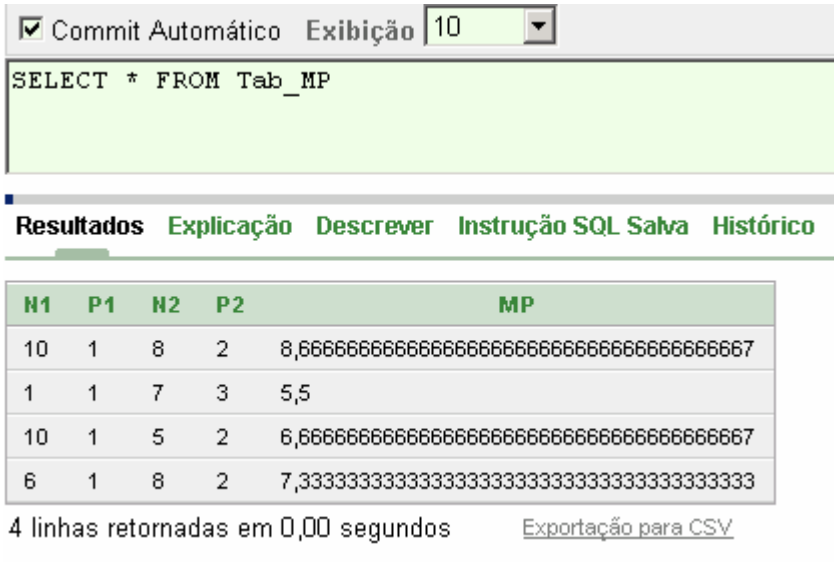
INSERT INTO Tab_MP (n1, p1, n2, p2) VALUES (10, 1, 8, 2);

INSERT INTO Tab_MP (n1, p1, n2, p2) VALUES (1, 1, 7, 3);

d) Preencher os dados do atributo mp com o calculo através da função MEDIA_PON:

UPDATE Tab_MP SET mp=MEDIA_POND(n1, p1, n2, p2);

e) Os dados agora já estão calculados e armazenados na tabela:



N1	P1	N2	P2	MP
10	1	8	2	8,6666666666666666666666666666666666666667
1	1	7	3	5,5
10	1	5	2	6,6666666666666666666666666666666666666667
6	1	8	2	7,3333333333333333333333333333333333333333

4 linhas retornadas em 0,00 segundos Exportação para CSV

Exercício Proposto:

1) Crie uma função que dado o número de minutos retorne o número de horas correspondente.

2) Crie uma função que dado um salário retorne o número de salários mínimos que a pessoa ganha.

Resposta do Exercício Proposto:

1)

```
create or replace function "MINPARAHORAS"
(minutos in NUMBER)
return NUMBER
is
begin
  return minutos/60;
end;
```

2)

```
create or replace function "SALMIN"
(salario in NUMBER)
return NUMBER
is
num_sal_min NUMBER;

BEGIN
  num_sal_min:=salario/380;
  RETURN num_sal_min;
END;
```

Controle de fluxo no PL/SQL:

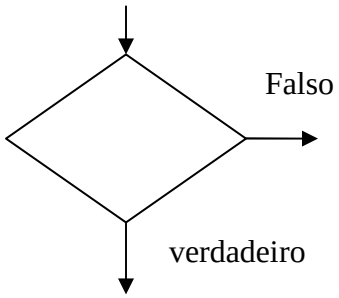
Temos no PL/SQL três formas de fluxo:

A seqüencial ou natural que executa os comandos de cima para baixo e da esquerda para direita, sendo o ";" o caractere que determina o fim de um comando.

A outra forma é de decisão onde escolhemos fazer ou não determinado passo ou executar um passo se verdadeiro ou outro passo se falso.

E por fim temos a repetição que permite voltar no fluxo de execução fazendo que determinados passos sejam repetidos conforme uma condição de saída.

Decisão:




```
IF <CONDIÇÃO> THEN
    <BLOCO-VERDADEIRO>
ELSE
     <BLOCO-FALSO>
END IF;
```

A seguir temos algumas funções criadas com o uso da condição:

Dado um número a função retorna em varchar2 se o número é PAR ou ÍMPAR:

```
1  create or replace function "TESTE"
2  (num in NUMBER)
3  return VARCHAR2
4  is
5  begin
6    if num mod 2=0 then
7       return 'PAR';
8    else
9       return 'ÍMPAR';
10   end if;
11 end;
```

Dada uma nota a função retorna a menção em VARCHAR2:

```
CREATE OR REPLACE FUNCTION MENCAO(nota NUMBER)
  RETURN VARCHAR2 is
m VARCHAR2(2);
begin
  if nota<0 OR nota>10 then
     m:='DC';
  else
     if nota=0 then
        m:='SR';
     else
        if nota<2 then
           m:='II';
        else
           if nota<5 then
              m:='MI';
           else
              if nota<7 then
                 m:='MM';
              else
                 if nota<9 then
                    m:='MS';
                 else
                    m:='SS';
                 end if;
              end if;
           end if;
        end if;
     end if;
  end if;

  return m;
end;
```

Exercício Proposto:

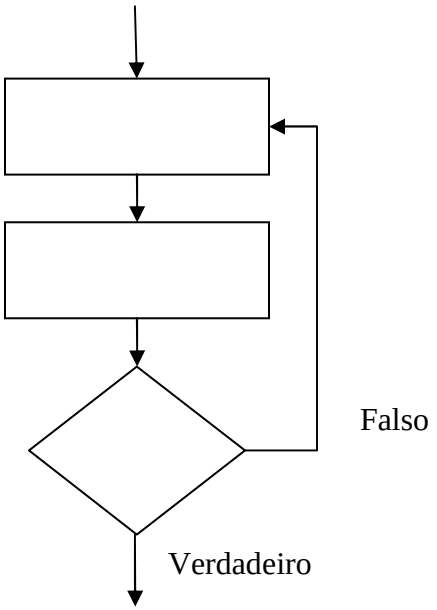
Dado um salário X calcule o salário líquido, sendo que os descontos de IRPF ocorrem conforme a tabela abaixo:

Base de cálculo mensal em R$	Alíquota %
Até 1.313,69	-
De 1.313,70 até 2.625,12	15,0
Acima de 2.625,12	27,5

```
CREATE OR REPLACE FUNCTION IRPF(salario NUMBER)
  RETURN NUMBER is
imp_a_pagar NUMBER;
parcela     NUMBER;
begin
  if salario<=1313.69 then
     imp_a_pagar:=0;
  end if;
  if salario>1313.69 AND salario<=2625.12 then
     parcela:=salario-1313.69;
     imp_a_pagar:=parcela*0.15;
  end if;
  if salario>2625.12 then
     parcela:=salario-2625.12;
     imp_a_pagar:=parcela*0.275 + ((2625.12-1313.69)*0.15);
  end if;

  return imp_a_pagar;
end;
```

Repetição:




```
LOOP
    [BLOCO-REPETIÇÃO]
    EXIT WHEN <CONDIÇÃO>
    [BLOCO-REPETIÇÃO]
END LOOP;
```

Desejamos implementar uma função que mostre contagens progressivas e regressivas:

```
1  create or replace FUNCTION Contar(ni NUMBER, nf NUMBER)
2    RETURN VARCHAR2 is
3  aux VARCHAR2(4000);
4  begin
5    aux:='Contando: ';
6    if ni>nf then
7       loop
8         aux:=aux || to_char(ni);
9         exit when nf>=ni;
10        aux:=aux || ', ';
11        ni:=ni-1;
12      end loop;
13   else
14      loop
15        aux:=aux || to_char(ni);
16        exit when nf<=ni;
17        aux:=aux || ', ';
18        ni:=ni+1;
19      end loop;
20   end if;
21   return aux;
22 end;
```

Desejamos criar uma função que recebe um número e informa se o número é ou primo:

```
create or replace FUNCTION Primo(num NUMBER)
  RETURN VARCHAR2 is
i   NUMBER;
begin
   i:=2;
  loop
    if num mod i=0 then
       return 'Não é PRIMO';
    end if;
    exit when i>(num/2);
    i:=i+1;
  end loop;
  return 'É PRIMO';
end;
```

Exercício Proposto

Dado um número X representando a quantidade de números primos que se deseja retornar. Mostre a lista com os X primeiros primos a partir de 2:

```
create or replace FUNCTION ListaPrimos(num NUMBER)
  RETURN VARCHAR2 is
qtd NUMBER;
i   NUMBER;
ret VARCHAR2(4000);
begin
  i:=2;
  qtd:=0;
  ret:=to_char(num) || ' número primos: ';
  loop
     if primo(i)='É PRIMO' then
        ret:= ret || to_char(i) || '  ';
        qtd:=qtd+1;
     end if;
     exit when qtd>=num;
     i:=i+1;
  end loop;
  return ret;
end;
```

Cursores

Cursor no PL/SQL é uma forma de tratar um resultado de consulta de forma personalizada. Podemos trabalhar registro a registro aplicando uma lógica própria.

Sintaxe Geral:

```
DECLARE
    CURSOR <NOME-CURSOR> IS    <CONSULTA>;

    <VARIAVEL-REGISTRO> <NOME-CURSOR>%ROWTYPE;

  BEGIN

      OPEN <NOME-CURSOR>;

      LOOP

         FETCH <NOME-CURSOR> INTO <VARIAVEL-REGISTRO>;

         EXIT WHEN <NOME-CURSOR>%NOTFOUND;

         /* LÓGICA PERSONALIZADA

            – INSERIR, ALTERAR EM OUTRA TABELA;

            - FAZER ALGUM CALCULO, EXPRESSÕES;

         */

      END LOOP;
      CLOSE <NOME-CURSOR>;
END;
```

Dada a seguinte tabela com os seguintes registros:

```
CREATE TABLE Fun(
  idFun   INTEGER,
  nomeFun VARCHAR2(100),
  salario DECIMAL(8,2),
  CONSTRAINT pk_fun PRIMARY KEY (idFun)
);


INSERT INTO Fun VALUES (1, 'Maria', 2000.50);

INSERT INTO Fun VALUES (2, 'Carolina', 3000.25);

INSERT INTO Fun VALUES (3, 'Sergio', 2500.00);
```

Desejamos ler registro a registro a tabela FUN e mostrar os funcionários e seus salários, informando ainda se o salário do mesmo é acima ou abaixo da média:

```
create or replace PROCEDURE LISTAFUN is
  CURSOR c_fun is SELECT * FROM Fun;
  r_fun  c_fun%ROWTYPE;
  media  NUMBER;
BEGIN
  SELECT AVG(salario) INTO media FROM Fun;
  DBMS_OUTPUT.PUT_LINE('Média=' || to_char(media));
  DBMS_OUTPUT.PUT_LINE('Lista de Funcionários');
  OPEN c_fun;
  LOOP
    FETCH c_fun INTO r_fun;
    EXIT WHEN c_fun%NOTFOUND;
    if r_fun.salario>=media THEN
      DBMS_OUTPUT.PUT_LINE(r_fun.nomeFun || ' - ' || to_char(r_fun.salario) || ' - Salário maior ou
igual a média');
    ELSE
      DBMS_OUTPUT.PUT_LINE(r_fun.nomeFun || ' - ' || to_char(r_fun.salario) || ' - Salário menor
que a média');
    END IF;
  END LOOP;
  CLOSE c_fun;
END LISTAFUN;
```

Para rodar este procedimento:

```
begin
  listafun;
end;
```