Procedures e Funções

PL/SQL

- **P**rocedural **L**anguage / **S**tructured **Q**uery **L**anguage une o estilo modular de linguagens de programação à versatilidade no acesso a banco de dados obtidas via SQL.

- PL/SQL é proprietária da Oracle

- Caso seja necessário migrar para outro SGBD, perde-se todo o trabalho em termos de Stored Procedures, Triggers e Functions

Processamento condicional

If *condição1* then

Comandos executados caso a condição1 seja verdadeira

[Elseif *condição2*

Comandos executados caso a condição2 seja verdadeira]

[Else

Comandos executados caso nenhuma condição seja verdadeira]

End if;

PL: Processamento condicional

Case

When ***condição*** ***(****atributo* *op relacional* *valor****)***

then *valor que o atributo assume se a condição for verdadeira*

When ***condição***

then *valor que o atributo assume se a condição for verdadeira*

Else *valor que o atributo assume se nenhuma condição anterior for verdadeira;*

End;

Processamento Repetitivo

- FOR: repete n vezes com n conhecido

FOR I in 1..max LOOP

comandos que devem ser repetidos

END LOOP;

Obs.: as variáveis que controlam o número de repetições (*I*) não precisam ser declaradas nem incrementadas.

Processamento Repetitivo

- FOR: pode ter contagem regressiva

```
FOR I in REVERSE max..1 LOOP
  comandos que devem ser repetidos
END LOOP;
```

Processamento Repetitivo

- WHILE: efetua a iteração mediante teste.

WHILE condição LOOP

comandos que devem ser repetidos

END LOOP;

Obs: as variáveis que controlam a iteração devem ser declaradas e explicitamente incrementadas.

Processamento Repetitivo

- LOOP: repete infinita vezes até que seja explicitamente forçado o fim do laço.

LOOP

comandos que devem ser repetidos

EXIT WHEN *condição;*

END LOOP;

Obs: as variáveis que controlam a iteração devem ser declaradas e explicitamente incrementadas.

Processamento Repetitivo

- FORALL: implementa a técnica **bulk binds**, que consiste em pré-armazenar um conjunto de comandos DML e envia-los de uma vez ao núcleo SQL.

FORALL j in 1..Max

*comando (insert, update ou delete) a repetir; **

** Admite um único comando por vez!*

Processamento Repetitivo

```
create or replace procedure Alimenta_Historico_Forall
  (ultima_turma in number, ultimo_aluno in number)
is
  type tlista is table of number index by binary_integer;
  lista tlista;
begin
  for j in 1..100 loop
        lista(j) := j;
  end loop;
  delete historico;
  for i in 1..ultima_turma loop
    forall j in 1..ultimo_aluno
      insert into historico (cod_turma, matricula) values (i,lista(j));
  end loop;
  commit;
END;
/
```

Unidades de Programa

- Um bloco possui a seguinte estrutura:

[declare] // *declaração de variáveis, constantes e cursores. Contém inicializações.*

Begin //*comandos SQL e estruturas de programação (if, while, etc)*

[Exception] //*identificação dos erros e emissão de mensagens*

End;

Tipos de Unidades de Programa

Procedure	Pode receber parâmetros de entrada ou de saída. Ativado como se fosse um comando da linguagem.
Function	Pode receber parâmetros apenas de entrada e, necessariamente, retorna um valor em seu nome. A ativação ocorre em expressões.
Package	Reunião física de procedures, functions e cursores.
Trigger	Rotina disparada automaticamente antes ou depois de comandos update, insert ou delete.

Procedure

- Pequenas porções de código que realizam tarefas específicas e ativadas como comandos

- Podem receber parâmetros de:
  - Entrada (In)
  - Saída (Out)
  - Entrada e Saída (InOut)

Procedure

- Sintaxe:

CREATE [OR REPLACE] PROCEDURE *nome_procedure*

([*lista de parâmetros*])

IS

  declarações locais

BEGIN

  comandos

END;

Procedure

```
create or replace procedure AlimentaHistorico
  (ultima_turma in number, ultimo_aluno in number)
is
begin
  delete historico;    /* comentário: elimina registros atuais */
  for i in 1..ultima_turma loop
    for j in 1..ultimo_aluno loop
      insert into historico (cod_turma, matricula) values (i,j);
    end loop;
  end loop;
  commit;
end;
/
```

Execução: **exec** alimentahistorico(10,10);

Verificando a existência: Select object_name **from** user_objects
where object_type='PROCEDURE';

Function

- Podem receber apenas parâmetros de entrada e devolvem um valor em seu nome.
- Sintaxe:

```
CREATE [OR REPLACE] FUNCTION
Nome_função ([lista de parâmetros])
RETURN tipo de retorno
IS
   declarações locais
BEGIN
  comandos
END;
```

Function

```
create or replace function ValorEmDolar
   (reais in number, cotacao in number)
return number
is
begin
   return reais/cotacao;
end;
/
```

Execução:

```
Select nome_curso, preco "Em R$", ValorEmDolar(preco, 2.97) "Em
   US$"
From cursos;
```

Verificando a existência:

```
Select object_name from user_objects
                    where object_type='FUNCTION';
```

Cursor

- Representa uma tabela temporariamente armazenada em memória e criada como resultado dos comandos: *select, insert, update, delete, commit* ou *rollback.*
- Contém os registros afetados pelo comando que provocou sua criação.
- ***Explícitos****: gerados apenas pelo* ***Select,*** *deve ser declarado e manipulado via comandos próprios.*
- **Implícitos**: dispensam qualquer tipo de tratamento.

Atributos de Cursores

Sql%rowcount	Informa quantas linhas foram afetadas pelo comando que gerou o cursor.
Sql%found	Será **true** caso alguma linha tenha sido afetada.
Sql%notfound	Será **false** caso alguma linha tenha sido afetada.
Sql%isopen	Será **true** caso o cursor esteja aberto (cursores explícitos).

Em cursores explícitos a palavra SQL é trocada pelo nome do cursor.

Exemplo de Cursor Implícito

```
create or replace function
 exclui_instrutores_cursor_imp
  return varchar2
  is
  begin
     delete instrutores
        where cod_instrutor not in
        (select distinct cod_instrutor from
 turmas);
     if sql%found then
    return ('Foram eliminados: '||
 to_char(sql%rowcount) || ' instrutores');
     else
    return ('Nenhum instrutor eliminado.');
     end if;
  end;
/
```

Visualizando o resultado...

Set serveroutput on; *//variável de ambiente*.

```
declare saida varchar2(40);
   begin
    saida:=exclui_instrutores_cursor_imp;
    dbms_output.put_line('Saida: '||saida);
   end;
 /
```

Comandos Cursor Explicíto

Open	Cria fisicamente a tabela temporária e posiciona o ponteiro de leitura no primeiro registro.
Fetch	Carrega para variáveis locais o conteúdo da linha indicada pelo ponteiro de leitura.
Close	Fecha o cursor.

Exemplo Cursor Explícito

```
create or replace procedure Classifica_Cursos_Cur_Exp
IS
  cursor ccursos is select nome_curso, preco from cursos;
  v_nome_curso cursos.nome_curso%type;
  v_preco   cursos.preco%type;
  v_classifica     varchar2(10);
BEGIN
  open ccursos;
  fetch ccursos into v_nome_curso, v_preco;
  while ccursos%found loop
     if v_preco < 300 then v_classifica := 'Barato';
      elsif v_preco < 600 then v_classifica := 'Médio';
         else  v_classifica := 'Caro';
     end if;
     dbms_output.put_line ('Curso:  '|| v_nome_curso || ' é ' ||
  v_classifica);
               fetch ccursos into v_nome_curso, v_preco;
  end loop;
  close ccursos;
END;
/
```

Cursor Parametrizado

```
create or replace procedure Class_Cursos_Cur_Exp_Param
(v_valor_minimo number)
IS
  cursor ccursos (v_valor_minimo in number) is
     select nome_curso, preco from cursos
      where preco > v_valor_minimo;
  v_nome_curso cursos.nome_curso%type;
  v_preco cursos.preco%type;
  v_classifica    varchar2(10);
BEGIN
  open ccursos (v_valor_minimo);
  fetch ccursos into v_nome_curso, v_preco;
  while ccursos%found loop
     if v_preco < 300 then v_classifica := 'Barato';
     elsif v_preco < 600 then v_classifica := 'Médio';
     else v_classifica := 'Caro';
     end if;
          fetch ccursos into v_nome_curso, v_preco;
  end loop;
  close ccursos;
END;
/
```