

Agenda:

- Correção dos exercícios
De procedimentos e funções.
- Cursores

Introdução

- Cursores são estruturas de PL/SQL que permitem que um comando SELECT possa retornar múltiplas linhas e as mesmas possam ser tratadas uma a uma pelo bloco PL/SQL.
- São estruturas de programação as quais declaramos na seção declare ou utilizamos em conjunto com um comando FOR na seção BEGIN.
- Possui uma estrutura no SGBD que garante uma melhor performance quando utilizado ao invés de comando SELECT sem o uso de cursores.
- Todos os comandos SQL possuem o chamado cursor implícito, pois são capazes de processar mais de uma linha por vez (insert, update, delete e merge)
- Um bloco PL/SQL ou procedimento, função , pacote ou gatilho pode conter um ou mais cursores
- Podemos encadear cursores de forma que o resultado de um pode servir para alimentar o outro
- Cursores podem receber parâmetros, assim como procedimentos e funções
- Possuem grande performance e são de uso recomendado pela Oracle

- Cursores podem ser reutilizados, se 2 ou mais sessões utilizam o mesmo cursor , este ficará residente em memória (no servidor de banco de dados) evitando que o comando seja analisado a cada execução, fazendo com que a execução fique mais rápida.
- Cursores podem ser usados sem nenhuma restrição no PL/SQL mesmo que sejam para retornar apenas um registro.
- Conforme as aplicações alocam cursores no banco de dados, o DBA deve configurar o mesmo para acomodar a quantidade de cursores utilizadas pelas aplicações.
- A área para alocação de cursores no banco de dados chama-se CURSOR AREA.

Tipos de Cursores

Os tipos de Cursores são:

Explícitos

Implícitos

Quanto ao comportamento podem ser:

Estáticos

Dinâmicos

Opcionalmente podem ser:

FOR UPDATE

Cursors Implícitos

Cursors implícitos são aqueles que não são declarados de forma explícita na seção DECLARE.

Comandos DML (Insert, Update , Delete, Merge e Select INTO) possuem cursores implícitos associados aos mesmos.

Executados em área comum do banco de dados chamada área de trabalho privada (Private Working Area).

Exemplo:

```
Declare v_nome empregados.nome%type;
Begin
    Select nome into v_nome from empregados where código =1;
    Insert into historico values (v_nome, sysdate);
End;
```

Cursorres Explícitos

- Cursorres explícitos são aqueles declarados na seção declare e depois utilizados na seção BEGIN
- Cursorres explícitos tem área exclusiva de execução dentro do banco de dados.
- Contem apenas comandos SELECT.
- Podem ter qualquer condição existente para comandos SELECT (Join, Group by, order by, subquery, etc..)
- Podem ter ou não cláusula where
- Podem ter condições (where) vinculados a variáveis e ou literais


```
CREATE TABLE HISTORICO (COD_PRODUTO NUMBER,  
                           NOME_PRODUTO  
                           VARCHAR(50) ,  
                           DATA_MOVIMENTACAO  
                           DATE) ;
```

Exemplo:

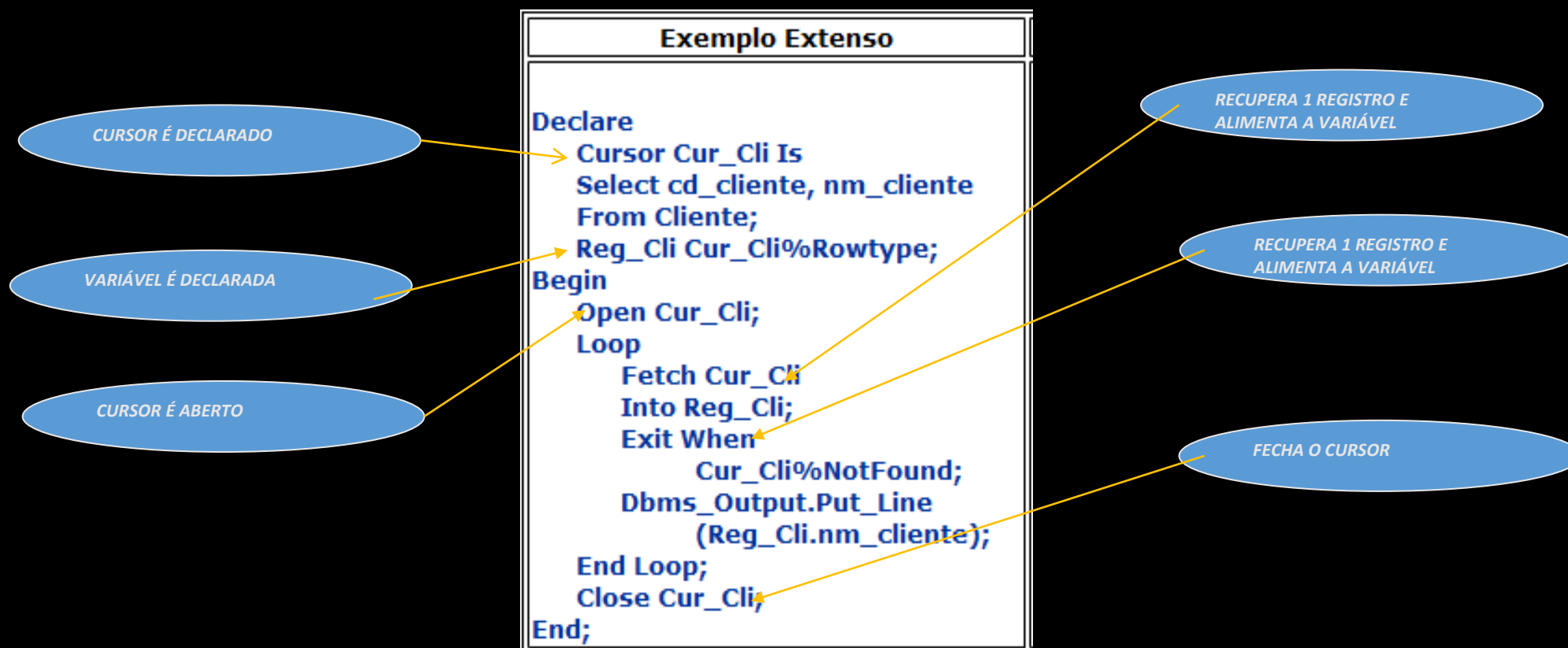
```

Declare
  v_código PRODUTO.COD_PRODUTO%type := 12349;
  cursor cur_emp is
    select NOM_PRODUTO from PRODUTO where
COD_PRODUTO = v_código;
Begin
  for x in cur_emp loop
    Insert into HISTORICO values (v_código,
x.NOM_PRODUTO, sysdate);
    COMMIT;
  End loop;
End;
```

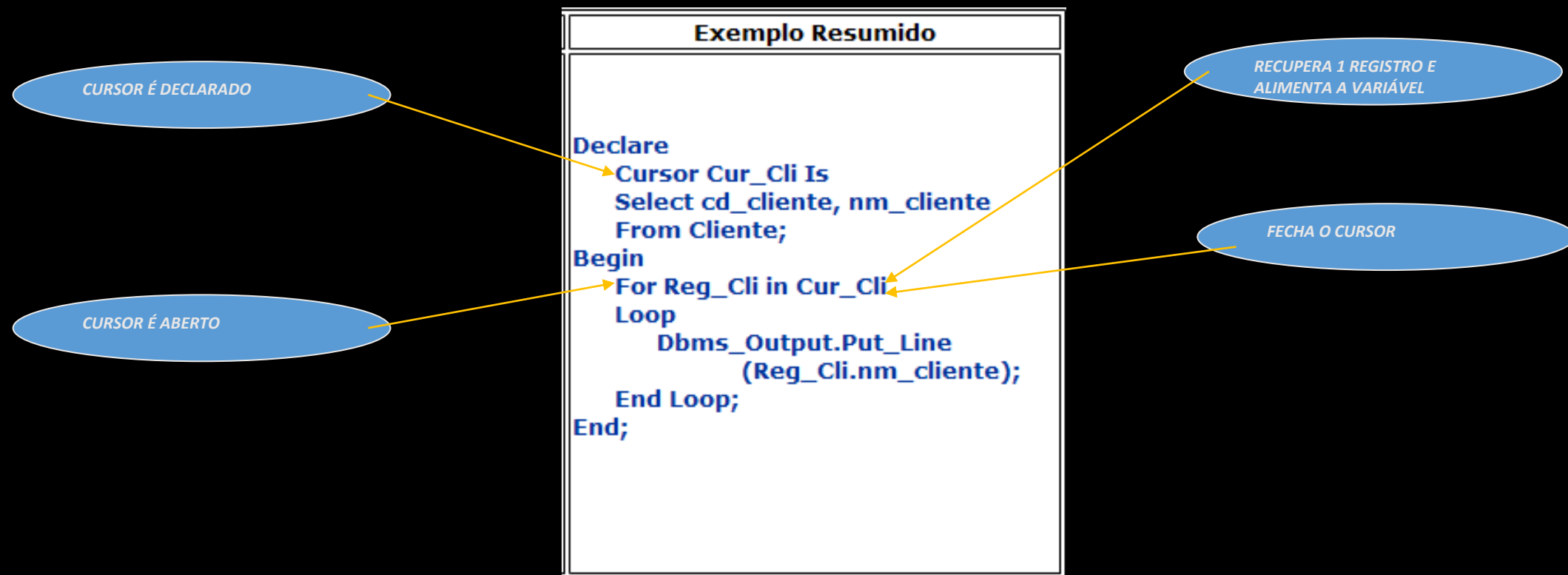
- Cursores explícitos podem ser manipulados de forma declarativa ou de forma resumida. No exemplo anterior vimos a forma resumida.
- A seguir veremos um exemplo que demonstra a forma declarativa (completa) e a resumida

Exemplo Extenso	Exemplo Resumido
<pre> Declare Cursor Cur_Cli Is Select cd_cliente, nm_cliente From Cliente; Reg_Cli Cur_Cli%Rowtype; Begin Open Cur_Cli; Loop Fetch Cur_Cli Into Reg_Cli; Exit When Cur_Cli%NotFound; Dbms_Output.Put_Line (Reg_Cli.nm_cliente); End Loop; Close Cur_Cli; End;</pre>	<pre> Declare Cursor Cur_Cli Is Select cd_cliente, nm_cliente From Cliente; Begin For Reg_Cli in Cur_Cli Loop Dbms_Output.Put_Line (Reg_Cli.nm_cliente); End Loop; End;</pre>

- Um cursor usado no formato declarativo precisa ser declarado na seção DECLARE e depois Aberto (OPEN) depois feita a leitura de cada registro (FETCH) e ao final deve ser encerrado (CLOSE). Além disso é necessário nesse formato a declaração de 1 ou mais variáveis para receber o resultado da execução do mesmo



Um cursor usado no formato resumido precisa ser declarado na seção DECLARE e depois manipulado através de comando FOR, nesse caso o comando FOR (ABRE, EXECUTA E FECHA o cursor). A variável associada ao comando for transforma-se numa variável do tipo %ROWTYPE, não precisando declarar variável para receber o conteúdo de cada registro. Não é possível usar outra estrutura de laço que não o comando FOR. Não precisamos determinar o fim porque o comando FOR entende que o ultimo registro é o fim.



Navegando pelo Cursor

- A navegação em um cursor é sempre para frente , ou seja a cada comando fetch ou a cada nova iteração do comando FOR o próximo registro será posicionado e carregado.
- Em PL/SQL não existe maneira de voltar para os registros anteriores.
- Podemos fechar um cursor e abri-lo novamente durante a execução, o ponteiro do cursor voltará o 1º. Registro da consulta.
- Caso usemos o comando FOR para manipular cursor (Forma resumida) não podemos usar open fetch e close (Forma declarativa ou completa).
- As variáveis de cursor só fazem sentido no formato declarativo ou completo.

Variáveis de Cursores Explícitos

Ao usarmos a forma declarativa dos cursores , temos de controlar o fluxo de execução e para isso contamos com as variáveis de cursores, que são alimentadas automaticamente a medida que os manipulamos, vejamos a seguir quais são:

- ✓ **Nomedocursor%rowcount** : devolve o numero da linha processada até o momento (formato completo ou resumido)
- ✓ **NomedoCursor%isopen** : retorna true ou false para determinar se um cursor está aberto ou não (formato completo)
- ✓ **NomedoCursor%found** : retorna true ou false para determinar se um registro foi encontrado ou não (formato completo)
- ✓ **NomedoCursor%notfound** : retorna true ou false para determinar se um registro NÃO foi encontrado (formato completo)

Cursors com parâmetros

Cursors com parâmetros podem ser usados para aumentar o dinamismo do funcionamento do mesmo, recuperando de forma dinâmica apenas os registros que satisfaçam as condições passadas através dos parâmetros.

A forma de declaração dos parâmetros de cursor são bastante semelhantes a declaração de parâmetros em funções e em procedimentos.

Os parâmetros são apenas de entrada, logo não precisamos determinar o tipo do parametro (IN/ OUT ou IN OUT).

Sintaxe:

```
Cursor nome_do_cursor (param1 tipo[, param2 tipo,... ParamN tipo]) is
Select .... From .... Where col = param1...;
```

Exemplo:

```
Cursor c_emp (p_código empregados.código%type) is
select * from empregados where código = p_código;
```


Cláusula WHERE CURRENT OF

Cursor que forem criados para atualizar registros (consulta, manipula e grava o mesmo registro) podem usar a cláusula WHERE CURRENT OF, isso agiliza o processo de localização pelo comando update. Caso não seja utilizado precisamos recuperar no select as colunas da chave primária para usá-las na cláusula where do comando Update ou Delete.

Sintaxe:

Declare

```
cursor nomedocursor is
select ... From nomedatabela FOR UPDATE OF nome_coluna_tabela;
```

Begin

```
for x in nomedocursor loop
    update nomedatabela set ...
    where CURRENT OF nomedocursor;
    commit;
end loop;
```

End;

Cláusula FOR UPDATE

A cláusula FOR UPDATE pode ser usada para indicar que a cada linha recuperada os registros sofrerão um bloqueio, ou seja, a consulta também realiza bloqueio neste caso. Isso garante que uma vez a linha recuperada esta não será modificada até que receba um comando commit ou rollback. Pode ser usada tanto em operações envolvendo a cláusula WHERE CURRENT OF quanto usando ROWID (que veremos no slide seguinte)

Sintaxe:

Declare

```
cursor nomedocursor is
select ... From nomedatabela FOR UPDATE OF nome_coluna_tabela;
```

Begin

```
for x in nomedocursor loop
    update nomedatabela set ...
    where CURRENT OF nomedocursor;
    commit;
end loop;
```

End;

Cláusula NOWAIT

A cláusula NOWAIT poderá ser usada se utilizarmos a cláusula FOR UPDATE. Isso indica que em caso o registro a ser recuperado esteja bloqueado o PL/SQL não ficará aguardando a liberação do bloqueio. Impedindo o congelamento da execução do código.

Sintaxe:

Declare

```
cursor nomedocursor is  
select ... From nomedatabela FOR UPDATE OF nome_coluna_tabela NOWAIT;
```

Begin

```
for x in nomedocursor loop  
    update nomedatabela set ...  
    where CURRENT OF nomedocursor;  
    commit;  
end loop;
```

End;

USADA ROWID

Podemos usar cursores trazendo o Rowid no select do cursor e para atualizar podemos usar o rowid recuperado no cursor. Isso é particularmente útil quando se pretende atualizar os registros recuperados pelo cursor. Tem efeito semelhante a clausula WHERE CURRENT OF.

Sintaxe:

Declare

cursor nomedocursor is

select ..., rowid From nomedataabela FOR UPDATE OF nome_coluna_tabela;

Begin

for x in nomedocursor loop

update nomedataabela set ...

where rowid = x.rowid;

commit;

end loop;

End;

Cursors Encadeados

Cursors podem ser intercalados ou encadeados, isso implica que um mesmo bloco possa ter mais de um cursor declarado e estes podem ser acionados individualmente ou encadeados entre si . Podemos encadear ou intercalar quantos cursors quisermos.

Sintaxe:

Declare

```
cursor nomedocursor1 is
select ... From nometabela1 ;
cursor nomedocursor2 (p_param tipo) is
select ... From nometabela2;
```

Begin

```
for x in nomedocursor1 loop
    for y in nomedocursor2(x.coluna) loop
        null;
    end loop;
end loop;
```

End;

Exercícios

1) Fazer um bloco anônimo com cursor que realize uma consulta na tabela de clientes e retorne o código e o nome do cliente, use dbms_output para mostrar as informações como o exemplo abaixo:

Cliente: 1 Nome: Jose da Silva

Cliente: 2 Nome: Maria da Silva

Resolução:

```
declare
  cursor c_consulta_cliente is
    select cod_cliente, nom_cliente from cliente;
begin
  for x in c_consulta_cliente loop
    dbms_output.put_line('Cliente: ' || x.cod_cliente || 'Nome: ' ||
                          x.nom_cliente);
  end loop;
end;
```

- 2) Faça um procedimento chamado PRC_VALIDA_TOTAL_PEDIDO que receba como parametro o código do pedido e que utilize dois cursores, um para localizar o pedido e outro para acessar os itens deste pedido , fazendo a soma dos itens e ao final verificar se a soma dos itens (quantidade * preço unitário) – desconto é igual ao total do pedido. Caso os valores coincidam retorne pelo parametro p_retorno a mensagem ‘pedido ok’ , caso contrario retorne ‘total dos itens não coincide com valor total do pedido’
- 3) Faça um procedimento chamado PRC_DELETE_PEDIDO que receba como parametro o numero do pedido e que antes de excluir o pedido execute um cursor na tabela de itens de pedido e faça o delete de cada um deles usando a técnica de ROWID.