Cursor

Definições

Cursor

- Quando uma query é executada no Oracle, um result set é produzido e salvo em memória. Oracle permite ao programador acessar este result set em memória através de Cursores.

- Muitas vezes, quando uma query retorna mais de uma linha como resultado, seria interessante iterar sobre cada linha e processar os dados de uma maneira diferente para elas. Cursores são úteis neste caso.

Cursor (passos para uso)

- Declarar (Declare)
  - Dá um nome ao cursor e o associa a uma query que retornará múltiplas linhas
- Abrir (Open)
  - Executa a query
- Acessar (Fetch)
- Fechar (Close)
  - Encerra o processamento do cursor

Function

Exemplos (contextualizado ou não) e exercício

Function (sintaxe e utilização)

- Utilizado para modularizar uma consulta, a qual pode ser utilizada em diversos locais, sem necessidade de repetição de código.
- **Sempre possui retorno.**

```
CREATE [OR REPLACE ]FUNCTION <nome>
[(parâmetro [{IN | OUT | IN OUT}] tipo, ....)]
RETURN <tipo-retorno> {IS | AS}
BEGIN <corpo-do-procedimento>
END <nome>;
```

Function (exemplo)

- Criar uma função que, passada um cargo como argumento, retorne a média salarial dos funcionários com aquele cargo.

```
CREATE OR REPLACE FUNCTION
media_por_cargo (crg Funcionario.Cargo%TYPE)
RETURN NUMBER
IS
v_media NUMBER;
BEGIN
   SELECT AVG (salario) INTO v_media FROM
funcionario F WHERE F.Cargo LIKE crg;
   RETURN v_media;
END media_por_cargo;
```

Function (exercício)

- Faça uma função que recebe o nome do esporte e retorna a quantidade de atletas que praticam esse esporte.

Procedure

Exemplos e exercício

Procedure (sintaxe e utilização)

- Utilizado para modularizar uma ação, a qual pode ser reutilizada diversas vezes, sem necessidade de repetição de código.
- **Não possui retorno.** Todas as ações necessárias são realizadas dentro do corpo do procedimento

```
CREATE [OR REPLACE] PROCEDURE <nome>
[(parâmetro [{IN | OUT | IN OUT}] tipo, ....)]
{IS | AS}
<definições de variáveis>
BEGIN <corpo-do-procedimento>
END <nome>;
```

Procedure (exemplo)

- Crie uma Procedure que, dada uma equipe, imprima todos os títulos que ela possui.

```
CREATE OR REPLACE PROCEDURE equipeTitulos (codigoEq
EQUIPE.CODIGOEQUIPE%TYPE) IS
CURSOR cur_titulos IS SELECT CodigoTit FROM
DisputaEquiCamp WHERE CodigoEquipe = codigoEq;
titulo DISPUTAEQUICAMP.CODIGOTIT%TYPE;
BEGIN
OPEN cur_titulos;
LOOP
FETCH cur_titulos INTO titulo;
EXIT WHEN cur_titulos%NOTFOUND;
DBMS_OUTPUT.PUT_LINE('Titulo de código: ' || to_char(titulo));
END LOOP;
CLOSE cur_titulos;
END;
```

Procedure (exercício)

- Criar uma procedure que atualiza o salário de determinado funcionário. Caso o funcionário não exista, emita uma mensagem de alerta;

Trigger

Exercício

Trigger (utilização)

- Utilizado para executar uma ação quando uma outra acontecer, ou seja:
  - Quando uma determinada ação acontece, uma reação é acionada.
- Cuidado com:
  - Tabelas mutantes
  - Especificação das restrições de acontecimento

Trigger (sintaxe)

```
CREATE [ OR REPLACE ] TRIGGER <nome>
[ BEFORE | AFTER | INSTEAD OF ] <evento>
                                ON <tabela>
[REFERENCING NEW AS <novo_nome>
             OLD AS <antigo_nome> ]
[ FOR EACH ROW [ WHEN <condição> ] ]
[ DECLARE PRAGMA
          AUTONOMOUS_TRANSACTION ]
BEGIN <corpo-do-procedimento>
END <nome>;
```

Trigger (observações)

- Uso de BEFORE possibilita o acesso a valores antigos e novos
- Uso de FOR EACH ROW gera várias execuções do gatilho
- A opção FOR EACH ROW determina se o gatilho é do tipo row trigger ou statement trigger
  - Se especificada, o gatilho é executado UMA vez para cada tupla afetada pelo evento
  - Se omitida, o gatilho é executado UMA ÚNICA vez para cada ocorrência de evento

Trigger (observações)

- A opção WHEN:
  - É usada apenas com *row triggers*
  - Consiste em uma expressão booleana – SQL
  - Não pode incluir:
    - Subconsultas
    - Expressões em PL/SQL
    - Funções definidas pelo usuário

Trigger (exercício)

- Criar uma tabela de auditoria para as alterações feitas na tabela funcionário que guarde o usuário, o tipo e a data da modificação feita na tabela bem como os dados modificados do funcionário (código, cargo, salário).

```
CREATE TABLE auditorlog (
   audit_date       DATE,
   audit_user       VARCHAR2(30),
   audit_desc       VARCHAR2(30),
   old_func_id      NUMBER(7),
   old_func_carg    VARCHAR2(40),
   old_func_sal     NUMBER(7,2),
   new_func_id      NUMBER(7),
   new_func_carg VARCHAR2(40),
   new_func_sal     NUMBER(7,2)
);
```

Package

Exemplo e exercício

Package (utilização)

- Utilizado para melhor organização dos elementos do banco de dados.
  - Na definição do pacote só é apresentada a especificação do mesmo
  - A implementação é apresentada à parte através do corpo do pacote

Package (sintaxe)

```
CREATE [OR REPLACE] PACKAGE <nome> {IS|AS}
    <especificação de procedimento> |
    < especificação de procedimento função>|
    <declaração de variável> |
    <definição de tipo> |
    <declaração de exceção> |
    <declaração de cursor>
END <nome>;
```

```
CREATE [OR REPLACE] PACKAGE BODY <nome> {IS|AS}
    <implementações do que foi especificado>
END <nome>;
```

Package (exemplo)

- Crie um Package onde, dada uma equipe sejam listadas as modalidades relacionadas a ela.

```
CREATE OR REPLACE PACKAGE manipula_modalidade IS

        FUNCTION contar_modalidade (codigo_esporte IN
        esporte.codigoesp%TYPE) RETURN BINARY_INTEGER;

        PROCEDURE listar_modalidade (codigo_esporte IN
        esporte.codigoesp%TYPE);

END manipula_modalidade;
/
```

Package (exemplo)

- Continuação

```
CREATE OR REPLACE PACKAGE BODY manipula_modalidade AS

FUNCTION contar_modalidade (codigo_esporte IN esporte.codigoesp%TYPE)
RETURN BINARY_INTEGER IS quantidade BINARY_INTEGER;

        BEGIN
        SELECT COUNT(*) INTO quantidade FROM modalidade m where
m.codigoesp = codigo_esporte;
        RETURN quantidade;
        END contar_modalidade;

PROCEDURE listar_modalidade (codigo_esporte IN esporte.codigoesp%TYPE) IS

        quantidade NUMBER(3);
        CURSOR selecao_modalidades IS (SELECT m.descricao, m.codigoesp
FROM modalidade m where m.codigoesp = codigo_esporte);
        registro selecao_modalidades%rowType;
```

Package (exemplo)

- Continuação

```
        BEGIN
        SELECT contar_modalidade (codigo_esporte) INTO quantidade from
dual;
        OPEN selecao_modalidades;
        WHILE quantidade > 0 LOOP
                FETCH selecao_modalidades INTO registro.descricao,
registro.codigoesp;
                dbms_output.put_line (registro.descricao);
                quantidade := quantidade - 1;
        END LOOP;
        CLOSE selecao_modalidades;
END listar_modalidade;

END manipula_modalidade;
/
```

Package (exercício)

- Crie um Package que insere uma nova equipe e logo em seguida imprime a tabela de equipes atualizada;