Curso: Ciência da Computação
Disciplina: Banco de Dados

Prof. Sergio Murilo Stempliuc (smstempliuc@gmail.com)

Aula prática de Banco de Dados 2
PL/SQL - Unidade 2

1. COLEÇÕES – TABELAS INDEX BY

Utilizado para manipular várias variáveis de uma vez como uma unidade, de modo semelhante à arrays em C/C++ e Java.
Para declarar uma tabela index-by, deve-se primeiramente definir o tipo de tabela dentro de um bloco PL/SQL e então declarar uma variável desse tipo (semelhante a declaração de um registro).

Os elementos são compostos por um *índice* e por um *valor*. Os índices, apresentados entre parênteses, não estão necessariamente em uma ordem específica. Pelo fato de não serem armazenados contiguamente na memória como um array em Java ou C/C++, os elementos podem ser inseridos sob chaves arbitrárias.

```
DECLARE

   TYPE Tipo_Numeros IS TABLE OF temp_table.num_col%TYPE
   INDEX BY BINARY_INTEGER;

   Numeros Tipo_Numeros;

BEGIN

   Numeros(0) := 0;
   Numeros(1) := -10;
   Numeros(-1) := 5;

END;
```

2. VINCULAÇÃO EM VOLUME

2.1. FORALL

O comando FORALL realiza um loop de modo semelhante ao comando *FOR*, porém ao invés de realizar inúmeras trocas de contexto entre a PL/SQL e a SQL, apenas uma troca é necessária.
FORALL pode ser utilizado com INSERT, UPDATE e DELETE.

```
DECLARE

   TYPE Tipo_Numeros IS TABLE OF temp_table.num_col%TYPE
   INDEX BY BINARY_INTEGER;

   Numeros Tipo_Numeros;

BEGIN

      FOR Contador IN 1..50 LOOP
         Numeros(Contador) := Contador;
      END LOOP;


      FORALL Contador IN 1..50
         INSERT INTO temp_table(num_col)
         VALUES (Numeros(Contador));

END;
```

2.2. BULK COLLECT (VINCULAÇÃO EM VOLUME)

A cláusula BULK COLLECT é utilizada junto com o comando SELECT para que sejam evitadas várias trocas de contexto entre a PL/SQL e a SQL, fazendo com que somente uma troca seja necessária.

Exemplo 1:

```
DECLARE

TYPE Tipo_Numeros IS TABLE OF temp_table.num_col%TYPE
INDEX BY BINARY_INTEGER;

Numeros Tipo_Numeros;

BEGIN

      SELECT num_col
      BULK COLLECT
      INTO Numeros
      FROM temp_table;

  DBMS_OUTPUT.PUT_LINE(Numeros.COUNT || ' linhas recuperadas.');

      FOR Contador IN 1..Numeros.COUNT LOOP
         DBMS_OUTPUT.PUT_LINE(Numeros(Contador));
      END LOOP;
END;
```

Observe que pode-se também utilizar este comando com cursores através de FETCH BULK COLLECT.

Exemplo 2:

```
DECLARE

TYPE Tipo_Numeros IS TABLE OF temp_table.num_col%TYPE
INDEX BY BINARY_INTEGER;

CURSOR Cursor_Numeros IS
   SELECT num_col
   FROM temp_table;

Numeros Tipo_Numeros;

BEGIN
     OPEN Cursor_Numeros;
     FETCH Cursor_Numeros BULK COLLECT INTO Numeros;

  DBMS_OUTPUT.PUT_LINE(Numeros.COUNT || ' linhas recuperadas.');

     FOR Contador IN 1..Numeros.COUNT LOOP
        DBMS_OUTPUT.PUT_LINE(Numeros(Contador));
     END LOOP;

     CLOSE Cursor_Numeros;
END;
```

3. RETURNING

Com esta função é possível recuperar dados sobre linhas modificadas por uma instrução de DML, uma vez que a instrução tenha sido emitida. Por exemplo, o número inserido na coluna *num_col* e a string inserida na coluna *char_col* da tabela *temp_table* após a última inserção.

```
DECLARE

   Numero temp_table.num_col%TYPE;
   String temp_table.char_col%TYPE;

BEGIN

   INSERT INTO temp_table(num_col, char_col)
   VALUES(1000, 'teste');

   INSERT INTO temp_table(num_col, char_col)
   VALUES(2000, 'teste2')
   RETURNING num_col, char_col INTO Numero, String;

   --retorna a mensagem Numero: 2000   String: teste2
   DBMS_OUTPUT.PUT_LINE('Numero: ' || Numero || '   String: ' ||
                        String);
END;
```

4. SEQUÊNCIAS

Uma *sequência* é um objeto Oracle utilizado para gerar números únicos. Sobre uma sequência podemos utilizar CURRVAL, para recuperar o valor atual da seqüência, e NEXTVAL para incrementar a seqüência e retornar o novo valor. Tanto CURRVAL como NEXTVAL retornam um valor do tipo NUMBER. CURRVAL pode ser acessado apenas em uma dada sessão depois que pelo menos um NEXTVAL foi emitido.
Os valores de seqüência podem ser utilizados na lista de SELECT de uma consulta, na cláusula VALUES de uma instrução INSERT e na cláusula SET de uma instrução UPDATE. Entretanto, elas não podem ser utlizadas na cláusula WHERE.

```
CREATE SEQUENCE temp_table_sequence
  START WITH 100
  INCREMENT BY 1;
DECLARE

UltimoValor NUMBER;

BEGIN

INSERT INTO temp_table (num_col)
VALUES (temp_table_sequence.NEXTVAL);

SELECT temp_table_sequence.CURRVAL
INTO UltimoValor
FROM dual;

DBMS_OUTPUT.PUT_LINE(UltimoValor);
END;
```

Para excluir uma sequência: DROP SEQUENCE *nome_da_seqüência*.

5. ROWNUM

ROWNUM retorna o número atual de uma linha em um consulta. Ele é útil para limitar o número total de linhas e é principalmente utilizado na cláusula WHERE de consulta e na cláusula SET de instruções UPDATE. ROWNUM retorna um valor NUMBER, sendo a primeira linha ROWNUM 1, a segunda ROWNUM 2 e assim por diante.

Por exemplo, sabendo-se que a tabela *temp_table* possui 50 tuplas, sendo *num_col* numerado de 1 à 50, a seguinte consulta retorna as três primeiras linhas:

```
SELECT num_col
FROM temp_table
WHERE ROWNUM < 4;
```

Retorna: **NUM_COL**
1
2
3

6. EXERCÍCIOS

Crie blocos PL/SQL para:

1. Inserir novos hangares no banco de dados de forma que não seja necessário incluir explicitamente o seu código.
   Após cada inserção na tabela *hangar*, apresentar o código atribuído ao novo hangar inserido.

2. Retornar o código atribuído ao novo hangar inserido utilizando outra solução.

3. Saber se ROWNUM funciona de forma correta com a vinculação em volume.

4. A solução para consulta 19 da lista de exercícios de consultas avançadas em SQL possui pelo menos duas respostas possíveis, como pode-se conferir no gabarito.
   Essas soluções, apesar de compactas, podem ser difíceis de compreender.

   Outras soluções que utilizem os recursos da PL/SQL podem ser mais extensas, mas utilizam conceitos já estabelecidos das linguagens de 3ª geração e podem ser compostas por consultas mais básicas em SQL. Dessa forma, podem dividir melhor o problema em pequenas partes e mudar a perspectiva sobre sua resolução.

   Implemente então um programa em PL/SQL que resolva o mesmo problema e compare essa nova solução com as duas anteriores.