**Coleções**

Tipos de dados especiais

**VARARRAY**

Criação de array.

Passo 1: criação do tipo.

Sintaxe: TYPE <nome do tipo> is VARRAY(<tamanho>) of <tipo de dados>;

EX:

type tvalor is varray(20) of number(8,2); - criou-se um array de 20 posições do tipo number(8,2)

type tvalor is varray(10) of varchar2(20); - criou-se um array de 10 posições do tipo varchar2(20)

Em seguida declaramos uma variável do tipo criado.

Ex:

v\_valor tvalor;

E finalmente atribuímos os valores para v\_valor.

Ex:

v\_valor:=tvalor(4,8,7,6,0,2,5,6,8,9);

Exemplo de utilização:

Vamos fazer um programa onde o usuário informará um número x (entre 1 e 10) e o programa retornará o xo. número impar. Exemplo: usuário informa 5, temos que o 5º elemento impar é o 9 (1, 3, 5, 7, 9).

Exemplo:

DECLARE

type tvalor is varray(10) of number(2);

v\_valor tvalor;

BEGIN

v\_valor:=tvalor(1,3,5,7,9,11,13,15,17,19);

DBMS\_OUTPUT.put\_line(v\_valor(&posicao));

END;

Para atribuir valores dinamicamente para um vetor devemos fazer como no exemplo abaixo:

1 – declarar variáveis como já visto.

2 – inicializar variável do tipo varray ( teste:=matriz(); )

3 – Informar número de posições que iremos carregar. Usar extend (teste.extend(3); )

4 – Carregar valores no vetor (

teste(1):=&val1;

teste(2):=&val2;

teste(3):=&val3;

)

Exemplo:

DECLARE

type matriz is varray(20) of number(8,2);

teste matriz;

soma number(8,2);

BEGIN

teste:=matriz();

teste.extend(3);

teste(1):=&val1;

teste(2):=&val2;

teste(3):=&val3;

soma:=teste(1)+teste(2)+teste(3);

dbms\_output.put\_line(soma);

END;

**Tabelas aninhadas (nested tables)**

Suponha que precisemos armazenar os dados do funcionario como também de seus dependentes (CPF) em uma mesma tabela. Como o número de dependentes é variável uma possível solução seria com a utilização de tabelas aninhadas. Neste caso o campo dependente na tabela funcionario será uma tabela aninhada.

Para se criar uma tabela aninhada deve-se:

1 – Criar um tipo de dados do tipo tabela

CREATE OR REPLACE TYPE tabdep AS TABLE OF number(11);

/

2 – Criar tabela funcionario indicando a tabela aninhada

CREATE TABLE funcionario(mat NUMBER(5), nome varchar2(40), dependente tabdep)

NESTED TABLE dependente STORE AS dependente\_tab;

Para incluir dados na tabela aninhada.

INSERT INTO funcionario VALUES (1, 'leonardo', tabdep(11));

INSERT INTO funcionario VALUES (2, 'monica', tabdep(21, 22));

INSERT INTO funcionario VALUES (3, 'tais', tabdep(31, 32, 33));

Consultando os dados incluídos

Select \* from funcionario;

OU ENTÃO

SELECT mat, nome, COLUMN\_VALUE FROM funcionario t1, TABLE(t1.dependente) t2;

**Tabelas Index-by**

Primeiramente deve-se criar um novo tipo de dados.

TYPE <nome do tipo de dados> IS TABLE OF <tipo de dados A> index by <tipo de dados B>;

Onde podemos imaginar que será criada uma tabela com duas colunas, A e B, sendo respectivamente do tipo de dados A e tipo de dados B.

A coluna B (tipo de dados B) será a “chave primária da tabela a ser criada”. Todo acesso a informação acontecerá por esta coluna. O dado acessado será justamente a coluna A (tipo de dados A).

Posteriormente deve-se declarar uma variável sendo como do tipo que foi criado.

Além disso, podemos usar os comandos FIRST (posicionará no primeiro registro), LAST (posicionará no último registro) e NEXT (posicionará no registro seguinte ao informado).

Exemplo:

DECLARE

TYPE tipoPop IS TABLE OF NUMBER INDEX BY VARCHAR2(64);

cidade tipoPop;

qtde NUMBER;

nome VARCHAR2(64);

BEGIN

cidade('Monte Alegre do Sul') := 5000;

cidade('Sao Caetano') := 60000;

cidade('Sao Caetano') := 35000;

cidade('Amparo') := 50000;

cidade('Sao Paulo') := 1000000;

END;

/

Na declaração acima criamos o tipo tipoPop que será uma tabela com duas colunas (VARCHAR2(64) e NUMBER). A coluna que atuará como chave primária será a VARCHAR2(64). Foi criada a variável cidade que é do tipo tipoPop.

Verifique que para utilizarmos a variável cidade informamos dois valores.

Ex: cidade(‘Amparo’):=50000;

O valor que vai logo após o nome da variável (Amparo) é aquele que corresponde a chave primária e o outro campo (50000) é o que corresponde ao outro campo declarado (NUMBER).

No exemplo acima temos a inclusão de 4 valores na variável cidade. Note que Sao Caetano aparece duas vezes, por se tratar de campo único, quando da segunda declaração de São Caetano (informando o valor 35000) ocorre uma ALTERAÇÃO no valor associado a Sao Caetano, ou seja, a população de São Caetano é alterada de 60.000 para 35.000.

Suponha agora que queiramos que o usuario digite uma cidade o nosso programa retorne a população da mesma, desta forma teríamos:

DECLARE

TYPE tipoPop IS TABLE OF NUMBER INDEX BY VARCHAR2(64);

cidade tipoPop;

qtde NUMBER;

nome VARCHAR2(64);

BEGIN

cidade('Monte Alegre do Sul') := 5000;

cidade('Sao Caetano') := 60000;

cidade('Sao Caetano') := 35000;

cidade('Amparo') := 50000;

cidade('Sao Paulo') := 1000000;

dbms\_output.put\_line(cidade('&cidade'));

exception

when no\_data\_found then

dbms\_output.put\_line('Cidade nao cadastrada');

END;

/

Suponha agora que queiramos mostrar todas as cidades cadastradas como a população de todas elas.

DECLARE

TYPE tipoPop IS TABLE OF NUMBER INDEX BY VARCHAR2(64);

cidade tipoPop;

qtde NUMBER;

nome VARCHAR2(64);

BEGIN

cidade('Monte Alegre do Sul') := 5000;

cidade('Sao Caetano') := 60000;

cidade('Sao Caetano') := 35000;

cidade('Amparo') := 50000;

cidade('Sao Paulo') := 1000000;

nome := cidade.FIRST;

WHILE nome IS NOT NULL

LOOP

qtde := cidade(nome);

dbms\_output.put\_line(nome || ': ' || to\_char(qtde));

nome := cidade.NEXT(nome);

end loop;

END;

/

Para o caso acima nos utilizamos de FIRST e NEXT.

Primeiramente posicionamos na primeira cidade (ordenada de modo ascendente).

nome:=cidade.FIRST

enquanto houverem cidades cadastradas (WHILE nome IS NOT NULL), mostra os dados da cidade e posiciona-se na próxima cidade.

**nome**:=cidade.NEXT(**nome**), ou seja, vai posicionar na **cidade** seguinte a **cidade informada**.

1 - Faça um bloco PL/SQL onde informado o número do mês retorne o mês em seu formato literal. Ex: se o usuário digitar 3, retorna Março, se o usuário digitar 5 retorna maio.

DECLARE

type tvalor is varray(12) of varchar2(15);

v\_valor tvalor;

BEGIN

v\_valor:=tvalor('JANEIRO','FEVEREIRO','MARCO','ABRIL','MAIO','JUNHO','JULHO','AGOSTO','SETEMBRO','OUTUBRO','NOVEMBRO','DEZEMBRO');

DBMS\_OUTPUT.put\_line(v\_valor(&posicao));

END;

2- Crie uma tabela onde seja possível armazenar vários emails em tabela aninhada e outra coluna onde seja possível armazenar vários telefones em tabela aninhada.

CREATE OR REPLACE TYPE tabemail AS TABLE OF varchar2(20);

/

CREATE OR REPLACE TYPE tabtel AS TABLE OF number(9);

/

CREATE TABLE usuario(mat NUMBER(5), nome varchar2(10), email tabemail, tel tabtel)

NESTED TABLE email STORE AS tabemail\_tab,

NESTED TABLE tel STORE AS tabtel\_tab;

Gravar dados

INSERT INTO usuario VALUES (1, 'Ricardo', tabemail('ricardo@1'), tabtel(678901234) );

INSERT INTO usuario VALUES (2, 'Maria', tabemail('maria@1','maria@2'), tabtel(23112345));

INSERT INTO usuario VALUES (3, 'Monica', tabemail('monica@1'), tabtel(99790056,981837766));

INSERT INTO usuario VALUES (4, 'Leonardo', tabemail('leonardo@1','leonardo@2','leonardo@3'), tabtel(983506655,23112345));

SELECT mat, nome, t2.COLUMN\_VALUE FROM usuario t1, TABLE(t1.email) t2;

SELECT mat, nome, t2.COLUMN\_VALUE FROM usuario t1, TABLE(t1.tel) t2;

SELECT mat, nome, t2.COLUMN\_VALUE,t3.COLUMN\_VALUE FROM usuario t1, TABLE(t1.email) t2, TABLE(t1.tel) t3;

3 - Crie uma tabela index-by onde ao informar a sigla do estado, retornará o nome do estado. Se eu digitar como sigla do estado o valor XX, listar todos os estados. Trabalhar somente com estados da região Sudeste (SP, MG, RJ e ES).

DECLARE

TYPE tipoEstado IS TABLE OF varchar2(40) INDEX BY varchar2(2);

estado tipoEstado;

nm\_estado varchar2(40);

sigla varchar2(2):= '&sigla\_estado';

v\_sg varchar2(2);

BEGIN

estado('SP'):= 'Sao Paulo';

estado('MG'):= 'Minas Gerais';

estado('RJ'):= 'Rio de Janeiro';

estado('ES'):= 'Espirito Santo';

If sigla = 'XX' THEN

v\_sg := estado.FIRST;

WHILE v\_sg IS NOT NULL

LOOP

nm\_estado := estado(v\_sg);

dbms\_output.put\_line(v\_sg || ' - ' || nm\_estado);

v\_sg := estado.NEXT(v\_sg);

END LOOP;

ELSE

nm\_estado:= estado(sigla);

dbms\_output.put\_line(sigla || ' - ' || nm\_estado);

END IF;

END;