



Laboratório de Bases de Dados

Prof. José Fernando Rodrigues Júnior

Aula 7 – PL/SQL – Coleções

Material: Profa. Elaine Parros Machado de Sousa



PL/SQL

- **Coleções**

- **nested table**: equivale a um array, mas

- admite operações em qq posição
- assemelha-se a uma tabela com um único atributo

```
TYPE type_name IS TABLE OF element_type [NOT NULL];
```

- **index-by table**

- associative arrays: similar a *hash structures*

```
TYPE type_name IS TABLE OF element_type [NOT NULL]
```

```
INDEX BY [BINARY_INTEGER | PLS_INTEGER | VARCHAR2(size_limit)];
```

- **varray**

- variable-size array:
 - remoções (delete) apenas na extremidade
 - tamanho pré-definido

```
TYPE type_name IS {VARRAY | VARYING ARRAY} (size_limit)  
OF element_type [NOT NULL];
```



Coleções

x(1)	x(2)		x(4)		x(6)
João	Ana		José		Lia

■ ***Nested Tables***

- semelhantes a *arrays* unidimensionais
 - valores não contíguos em memória
 - não há número máximo de elementos – mas há nro de posições ocupáveis
 - começam densas, mas podem ficar esparsas após remoções
- chaves devem ser seqüenciais
 - não podem ser negativas
 - número de chaves: 2GB
- precisam ser inicializadas
- podem ser armazenadas no banco (Aula 12 – Objeto Relacional)
 - armazenados como tabela com duas colunas: chave, valor

```
TYPE tipo_tabela IS TABLE OF tipo [NOT NULL];
```

[illegible]



Coleções

x(1)	x(2)	x(3)	x(4)		
João	Ana	Lia	José		

■ VARRAY

- semelhante a um *array* (de tamanho variável) em C ou Java
 - valores contíguos em memória
- chaves devem ser seqüenciais
 - não podem ser negativas
 - número máximo de posições definido na declaração
 - tamanho do vetor corresponde ao nro de elementos armazenados
- não podem ser esparsos
 - elementos removidos somente do final do *array*
- precisam ser inicializados
- podem ser armazenados no banco (Aula 12 – Objeto Relacional)

```
TYPE type_name IS {VARRAY | VARYING ARRAY} (size_limit)  
OF element_type [NOT NULL];
```

declare

-- declaração

TYPE t_alunos IS VARRAY(3) OF varchar(30);

-- inicializa array com 2 elementos

v_alunos t_alunos := t_alunos('usp', 'unesp');

begin

v_alunos.extend; -- primeiro aloca espaço, depois usa
v_alunos(3) := 'UFSCar';

-- exceção: excedendo o tamanho máximo

v_alunos.extend;

exception

when SUBSCRIPT_OUTSIDE_LIMIT then dbms_output.put_line
('Violação do tamanho do array');

end;



Coleções

CHAVE	VALOR
-1	João
2	Lisa
1	Ana

■ *Index-by tables*

- semelhantes a tabelas com duas colunas: *chave*, *valor*
 - **chaves podem ser *integer* ou *string***
 - valores não contíguos em memória
 - podem ser esparsas
- chaves não precisam ser seqüenciais
 - *integer* - podem ser negativas
 - número de elementos limitado pelo tipo da chave: `PLS_INTEGER`, `VARCHAR2`
- não precisam ser inicializadas
- não podem ser armazenadas no banco

```
TYPE tipo_tabela IS TABLE OF tipo INDEX  
  BY [BINARY_INTEGER | PLS_INTEGER | VARCHAR2(size_limit)];
```

DECLARE

```
TYPE population_type IS TABLE OF NUMBER INDEX BY VARCHAR2(64);  
country_population population_type;  
howmany NUMBER;  
which VARCHAR2(64);
```

BEGIN

```
country_population('Greenland') := 1;  
country_population('Iceland') := 2;  
howmany := country_population('Greenland');  
country_population('Japao') := 3;  
country_population('Franca') := 4;  
  
country_population('Japao') := 5;  
  
--vale a ordem alfabética  
which := country_population.FIRST;  
dbms_output.put_line(which);  
howmany := country_population(which);  
dbms_output.put_line(howmany);  
  
which := country_population.NEXT(country_population.FIRST);  
dbms_output.put_line(which);  
howmany := country_population(which);  
dbms_output.put_line(howmany);  
  
which := country_population.PRIOR(country_population.LAST);  
dbms_output.put_line(which);  
howmany := country_population(which);  
dbms_output.put_line(howmany);  
  
which := country_population.LAST;  
dbms_output.put_line(which);  
howmany := country_population(which);  
dbms_output.put_line(howmany);
```

END;



PL/SQL

- Registros e coleções
 - manipulação de várias variáveis
- Registro

```
TYPE t_aluno IS RECORD (  
    nome aluno.nome%type,  
    nusp aluno.nusp%type  
);
```

```
v_aluno t_aluno;
```

declare

```
TYPE t_aluno IS RECORD (  
    nome aluno.nome%type,  
    nusp aluno.nusp%type  
);
```

-- declaração

```
TYPE t_tab_alunos IS TABLE OF t_aluno INDEX BY PLS_INTEGER;  
v_alunos t_tab_alunos; -- coleção vazia
```

begin

-- atribuição de valores - semelhante a INSERT se elemento *i* não existe

```
v_alunos(0).nome := 'Aline';
```

```
v_alunos(0).nusp := 444;
```

```
v_alunos(-2).nome := 'Lia';
```

```
v_alunos(-2).nusp := 999;
```

```
dbms_output.put_line ('Acessando aluno (0): ' || v_alunos(0).nome);
```

-- acesso a elemento inexistente - semelhante a SELECT que retorna vazio

```
dbms_output.put_line ('Acessando aluno (2): ' || v_alunos(2).nome);
```

exception

```
    when NO_DATA_FOUND then dbms_output.put_line ('Elemento 2 não  
exite!');
```

end;

INDEX-BY	NESTED	VARRAY
Sem tamanho máximo	Sem tamanho máximo	Com tamanho máximo = LIMIT
Podem ser esparsas - inserções e remoções em qualquer posição - sem ordem	Podem ser esparsas - inserções e remoções em qualquer posição - sem ordem	Densas, com indexação em sequência - não admite remoções
	AMBAS	
Sem inicialização	A estrutura, bem como cada posição, devem ser inicializadas/alocadas antes do uso	
Utilizam atributos de tabela	Utilizam métodos adicionais de coleção	
NÃO podem ser armazenadas no banco de dados	Podem ser armazenadas no banco de dados	
Não podem assumir valores NULL	Podem armazenar valores NULL	
Raise NO_DATA_FOUND para elementos inexistentes	Raise SUBSCRIPT_BEYOND_COUNT para referências a elementos inexistentes	
Apenas dentro do PL/SQL	Podem ser declaradas fora do PL/SQL com CREATE TYPE	

Método	Retorno	Validade
EXISTS (i)	true/false	<ul style="list-style-type: none">• nested• varray• index-by
COUNT	NUMBER	<ul style="list-style-type: none">• nested• varray• index-by
LIMIT	NUMBER	<ul style="list-style-type: none">• nested (sem aplicação – use COUNT)• varray
FIRST/LAST	BINARY_INTEGER	<ul style="list-style-type: none">• nested• varray• index-by
NEXT (i) /PRIOR (i)	BINARY_INTEGER	<ul style="list-style-type: none">• nested• varray• index-by
EXTEND/EXTEND (n)	Void	<ul style="list-style-type: none">• nested• varray (até LIMIT)
TRIM/TRIM (n)	Void	<ul style="list-style-type: none">• nested• varray
DELETE/DELETE (i) /DELETE (i,j)	void	<ul style="list-style-type: none">• nested• index-by

Coleção	Inicialização	Alocação
Nested	<pre>n_my:= typename(); n_my:= typename(e1,e2, ..., en);</pre>	<pre>n_my.extend ➔ +1 com valor NULL n_my.extend(n) ➔ +n com valor NULL</pre>
Varray	<pre>v_my:= typename(); v_my:= typename(e1,e2, ..., en);</pre>	<pre>n_my.extend ➔ +1 com valor NULL n_my.extend(n) ➔ +n com valor NULL Até o valor LIMIT</pre>
Index-by	Não necessária	Não necessária



CURSOR FOR LOOP

■ Antes:

DECLARE

```
CURSOR c_old IS  
    SELECT L1.nome, L1.nrousp  
    FROM    lbd01_vinculo_usp L1;  
v_old c_old%ROWTYPE;
```

BEGIN

```
OPEN c_old;  
LOOP  
    FETCH c_old INTO v_old;  
    EXIT WHEN c_old%NOTFOUND;  
    dbms_output.put_line(v_old.nrousp);  
END LOOP;  
CLOSE c_old;
```

END;



CURSOR FOR LOOP

- **Com CURSOR FOR LOOP:**

DECLARE

```
CURSOR c_new IS  
    SELECT L1.nome, L1.nrousp  
    FROM    lbd01_vinculo_usp L1;
```

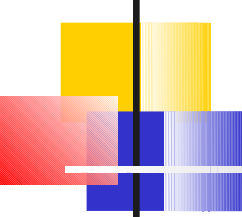
BEGIN

```
FOR v_new IN c_new LOOP
```

```
    dbms_output.put_line(v_new.nrousp);
```

```
END LOOP;
```

```
END;
```



PL/SQL - Coleções

- Manual de consulta:
 - comparação entre tipos de coleções
 - quando usar cada tipo
 - métodos e atributos de coleções
 - tipo de retorno, descrição, comportamento em cada tipo de coleção, ...
 - **exceções** pré-definidas para coleções
 - ...

PL/SQL User's Guide and Reference