Disciplina de Modelação de Dados em Engenharia

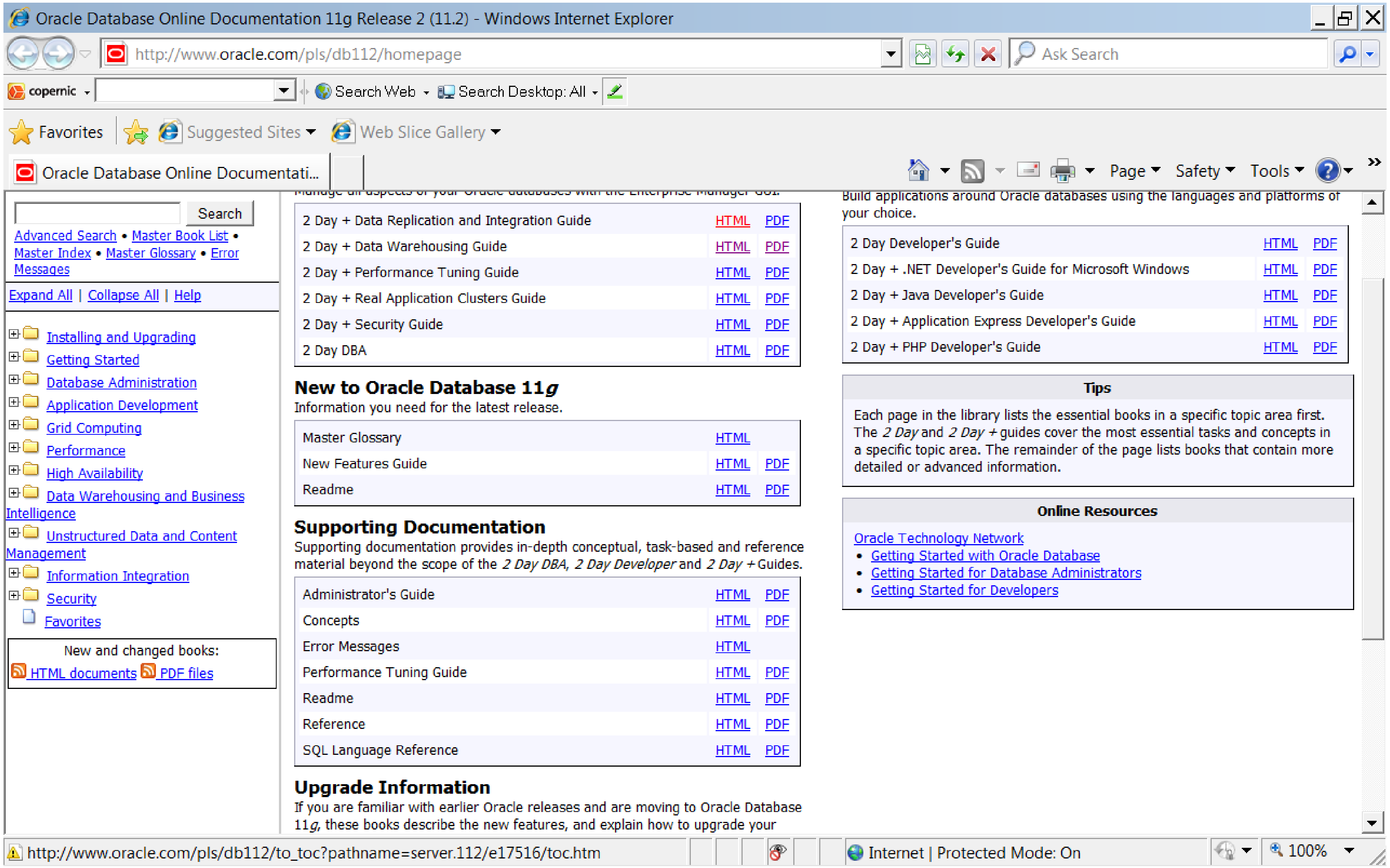
Material de Apoio às Aulas Teóricas (Português/Ingles)

Resp. Disciplina: **Luis Camarinha Matos**
João Rosas
Yves Rybarczyk
Pedro Santana

DEE / FCT / UNL
Secção de Robótica e Manufactura Integrada

Tópicos para hoje

- Tópicos avançados
- Exemplos
  - Projecto electrico
- TPC


Oracle Database Online Documentation 11g Release 2 (11.2) - Windows Internet Explorer
http://www.oracle.com/pls/db112/homepage
Ask Search
copernic
Search Web
Search Desktop: All
Favorites
Suggested Sites
Web Slice Gallery
Oracle Database Online Documentati...
Page
Safety
Tools

Search

Advanced Search • Master Book List • Master Index • Master Glossary • Error Messages

Expand All | Collapse All | Help

- Installing and Upgrading
- Getting Started
- Database Administration
- Application Development
- Grid Computing
- Performance
- High Availability
- Data Warehousing and Business Intelligence
- Unstructured Data and Content Management
- Information Integration
- Security
- Favorites

New and changed books:
HTML documents PDF files

Manage all aspects of your Oracle databases with the Enterprise Manager GUI.

2 Day + Data Replication and Integration Guide	HTML	PDF
2 Day + Data Warehousing Guide	HTML	PDF
2 Day + Performance Tuning Guide	HTML	PDF
2 Day + Real Application Clusters Guide	HTML	PDF
2 Day + Security Guide	HTML	PDF
2 Day DBA	HTML	PDF

New to Oracle Database 11*g*

Information you need for the latest release.

Master Glossary	HTML	
New Features Guide	HTML	PDF
Readme	HTML	PDF

Supporting Documentation

Supporting documentation provides in-depth conceptual, task-based and reference material beyond the scope of the *2 Day DBA*, *2 Day Developer* and *2 Day +* Guides.

Administrator's Guide	HTML	PDF
Concepts	HTML	PDF
Error Messages	HTML	
Performance Tuning Guide	HTML	PDF
Readme	HTML	PDF
Reference	HTML	PDF
SQL Language Reference	HTML	PDF

Upgrade Information

If you are familiar with earlier Oracle releases and are moving to Oracle Database 11*g*, these books describe the new features, and explain how to upgrade your

Build applications around Oracle databases using the languages and platforms of your choice.

2 Day Developer's Guide	HTML	PDF
2 Day + .NET Developer's Guide for Microsoft Windows	HTML	PDF
2 Day + Java Developer's Guide	HTML	PDF
2 Day + Application Express Developer's Guide	HTML	PDF
2 Day + PHP Developer's Guide	HTML	PDF

Tips

Each page in the library lists the essential books in a specific topic area first. The *2 Day* and *2 Day +* guides cover the most essential tasks and concepts in a specific topic area. The remainder of the page lists books that contain more detailed or advanced information.

Online Resources

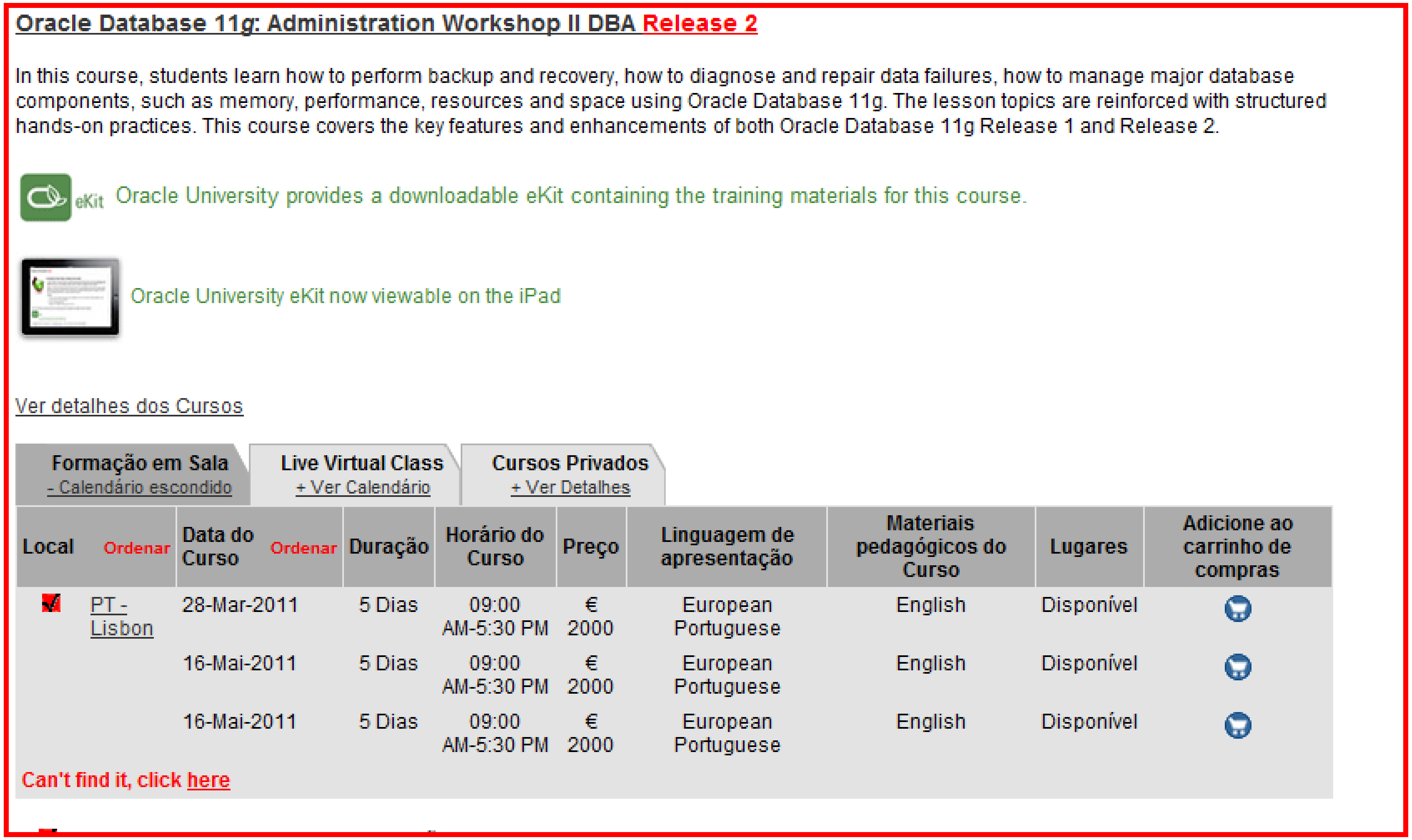
Oracle Technology Network
- Getting Started with Oracle Database
- Getting Started for Database Administrators
- Getting Started for Developers

http://www.oracle.com/pls/db112/to_toc?pathname=server.112/e17516/toc.htm

Internet | Protected Mode: On

100%

Quanto custa?



Oracle Database 11g: Administration Workshop II DBA Release 2

In this course, students learn how to perform backup and recovery, how to diagnose and repair data failures, how to manage major database components, such as memory, performance, resources and space using Oracle Database 11g. The lesson topics are reinforced with structured hands-on practices. This course covers the key features and enhancements of both Oracle Database 11g Release 1 and Release 2.

eKit Oracle University provides a downloadable eKit containing the training materials for this course.

Oracle University eKit now viewable on the iPad

Ver detalhes dos Cursos

Formação em Sala - Calendário escondido	Live Virtual Class + Ver Calendário	Cursos Privados + Ver Detalhes

Local Ordenar	Data do Curso Ordenar	Duração	Horário do Curso	Preço	Linguagem de apresentação	Materiais pedagógicos do Curso	Lugares	Adicione ao carrinho de compras
✓ PT - Lisbon	28-Mar-2011	5 Dias	09:00 AM-5:30 PM	€ 2000	European Portuguese	English	Disponível	🛒
	16-Mai-2011	5 Dias	09:00 AM-5:30 PM	€ 2000	European Portuguese	English	Disponível	🛒
	16-Mai-2011	5 Dias	09:00 AM-5:30 PM	€ 2000	European Portuguese	English	Disponível	🛒

Can't find it, click here

Iterating thorugh a “view”

- create vew employees_View
- Select * from emp, dept,
- Where emp.deptno=dept.deptno;

```
BEGIN
  FOR someone IN (
    SELECT * FROM employees_view
    WHERE empno< 120
    ORDER BY empno
  )
  LOOP
    DBMS_OUTPUT.PUT_LINE('First name = ' || someone.first_name ||
                         ', Last name = ' || someone.last_name);
  END LOOP;
END;
/
```

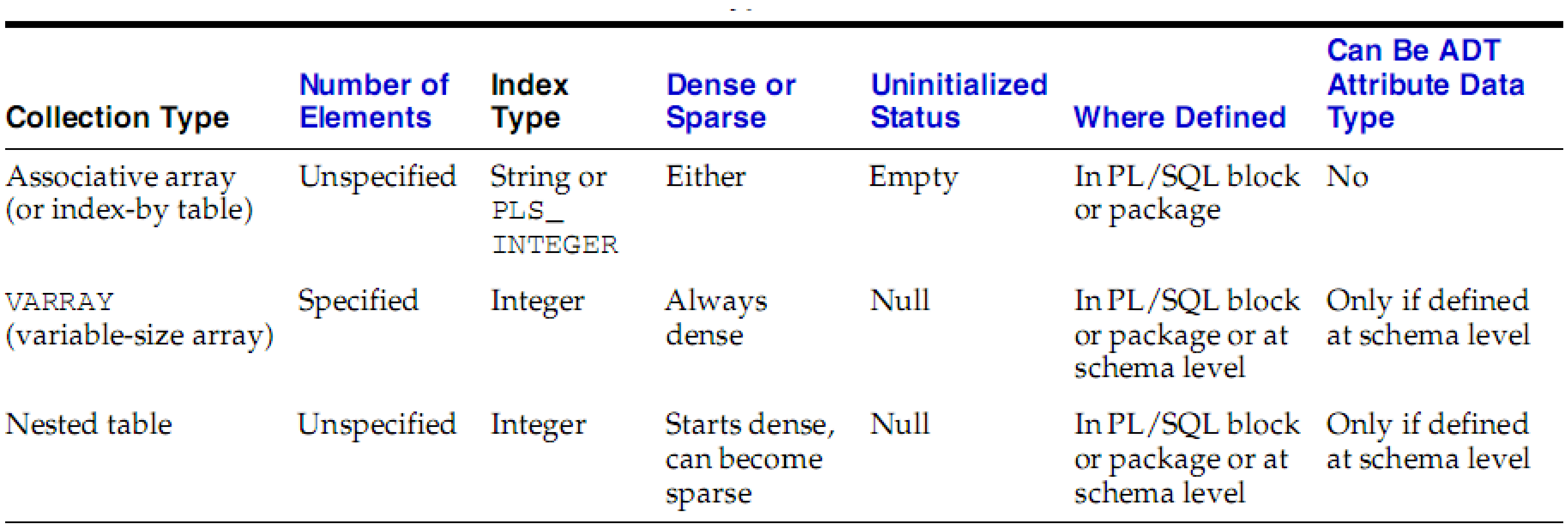
PL/SQL Collections and Records

Collection Types:

- Associative Arrays
- VARRAY (variable-size array)
- Nested Tables

```
• CREATE OR REPLACE PROCEDURE print (n
  INTEGER) IS
• BEGIN
•   IF n IS NOT NULL THEN
•     DBMS_OUTPUT.PUT_LINE(n);
•   ELSE
•     DBMS_OUTPUT.PUT_LINE('NULL');
•   END IF;
• END print;
```

Similaridades/diferenças



Collection Type	Number of Elements	Index Type	Dense or Sparse	Uninitialized Status	Where Defined	Can Be ADT Attribute Data Type
Associative array (or index-by table)	Unspecified	String or PLS_ INTEGER	Either	Empty	In PL/SQL block or package	No
VARRAY (variable-size array)	Specified	Integer	Always dense	Null	In PL/SQL block or package or at schema level	Only if defined at schema level
Nested table	Unspecified	Integer	Starts dense, can become sparse	Null	In PL/SQL block or package or at schema level	Only if defined at schema level

- Number of elements: nº máximo de elementos na colecção.
- Dense: sem gaps (valores “null”) entre elements.
- ADT: abstract data type.
- (ver resto em PL-SQL Language Reference.pdf, pag. 147 [5-3])

“Associative arrays”

- Similar a um hashtable.
- Conjunto de tuplos key-value {(k1,v2),…, (kn,vn)}.
- Um ki funciona como índice único para o valor vi correspondente.
- Ki pode ser to tipo varchar(2) ou PLS_integer.

Exemplo: index como pls_integer

```
create or replace PROCEDURE test_associative_arrays as
  type TAlunos is table of varchar2(30) index by PLS_integer;
  alunos TAlunos;
  idx PLS_integer;
BEGIN
  alunos(1) :=' josé socrates';
  alunos(2) :=' passos coelho';
  alunos(3) :=' jerónimo';
  alunos(4) :=' Compaio';
  alunos(5) :=' yourself...';
  idx := alunos.FIRST;
  while idx is not null LOOP
     dbms_output.put_line('aluno ->' || idx || ' Nome->' || alunos(idx) );
     idx := alunos.NEXT(idx);
  end loop;
END test_associative_arrays;
```

```
anonymous block completed
aluno ->1 Nome-> josé socrates
aluno ->2 Nome-> passos coelho
aluno ->3 Nome-> jerónimo
aluno ->4 Nome-> Compaio
aluno ->5 Nome-> yourself...
```

Exemplo: index como string

```
create or replace
PROCEDURE test_associative_arrays_2 as
  type TNotas is table of numeric(4,2) index by varchar2(20);
  notas TNotas;
  idx varchar2(20);
BEGIN
  notas('josé socrates') :=17.0;
  notas('passos coelho') :=15.0;
  notas('jerónimo') :=14.0;
  notas('Compaio') :=16.0;
  notas('yourself') :=20.0;
  idx := notas.FIRST;
  while idx is not null LOOP
     dbms_output.put_line('aluno ->' || idx || ' NOTA->' || notas(idx) );
     idx := notas.NEXT(idx);
  end loop;
END test_associative_arrays_2;
```

```
nonymous block completed
aluno ->Compaio NOTA->16
aluno ->jerónimo NOTA->14
aluno ->josé socrates NOTA->17
aluno ->passos coelho NOTA->15
aluno ->yourself NOTA->20
```

“associative array” como argumentos de procedimentos/funções

```
declare

 type TNotas is table of numeric(4,2) index by
      varchar2(20);

 as_notas TNotas;

 function calcular_notas return TNotas   is
   notas TNotas;
 begin
   notas('josé socrates') :=17.0;
   notas('passos coelho') :=15.0;
   notas('jerónimo') :=14.0;
   notas('Compaio') :=16.0;
   notas('yourself') :=20.0;
   return notas;
 end calcular_notas;
```

```
 procedure print_notas(nnn in TNotas)
as
   idx varchar2(20);
  begin
   idx := nnn.FIRST;
   while idx is not null LOOP
    dbms_output.put_line('aluno ->' ||
idx || ' NOTA->' || nnn(idx) );
    idx := nnn.NEXT(idx);
   end loop;
  END print_notas;
begin
  as_notas  :=  calcular_notas();
  print_notas(as_notas);
end;
```

usefulness

An **associative array** is appropriate for:

- A relatively small lookup table, which can be constructed in memory each time you invoke the subprogram or initialize the package that declares it.
- Passing collections to and from the database server

(ver PL-SQL Language Reference.pdf, pag. 152)

Varrays (Variable-Size Arrays)

- A varray (variable-size array) is an array whose number of elements can vary from zero (empty) to the declared maximum size.
- Lower bound of index is 1.
- Upper bound is the current number of elements.
- The upper bound changes as you add or delete elements, but it cannot exceed the maximum size.

Example

```
declare
  type TNomes is varray(5) of varchar2(20);
  type TNotas is varray(5) of numeric(4,2);

  nomes TNomes := TNomes('compaio','jeronimo','socrates', 'coelho','yourself');
  notas TNotas := TNotas(15,16,17,14, 20);
begin
    FOR i IN 1..5 LOOP
      DBMS_OUTPUT.PUT_LINE(nomes(i) || ' ->' || notas(i));
    END LOOP;
end;
```

```
anonymous block completed
compaio ->15
jeronimo ->16
socrates ->17
coelho ->14
yourself ->20
```

Appropriate Uses for Varrays

- A varray is appropriate when:
- You know the maximum number of elements.
- You usually access the elements sequentially.

  Because you must store or retrieve all elements at the same time, a varray might be impractical for large numbers of elements.

(ver PL-SQL Language Reference.pdf, pag. 154)

Nested table

- In the database, a nested table is a column type that stores an unspecified number of rows in no particular order.
- When you retrieve a nested table value from the database into a PL/SQL nested table variable, PL/SQL gives the rows consecutive indexes, starting at 1.
- Using these indexes, you can access the individual rows of the nested table variable.
- The syntax is variable_name(index).
- The indexes and row order of a nested table might not remain stable as you store and retrieve the nested table from the database.
- The amount of memory that a nested table variable occupies can increase or decrease dynamically, as you add or delete elements.
- An uninitialized nested table variable is a null collection. You must initialize it, either by making it empty or by assigning a non-NULL value to it.

Example

```
declare
  type TSelectedForRaise is table of varchar2(20);
  seleccionados TSelectedForRaise :=
      TSelectedForRaise('Antonio','Manela','Julio');
begin
  for eee in seleccionados.FIRST .. seleccionados.LAST
  LOOP
     dbms_output.put_line('seleccionado -> '|| seleccionados(eee) );
  end LOOP;
end;
```

```
anonymous block completed
seleccionado -> Antonio
seleccionado -> Manela
seleccionado -> Julio
```

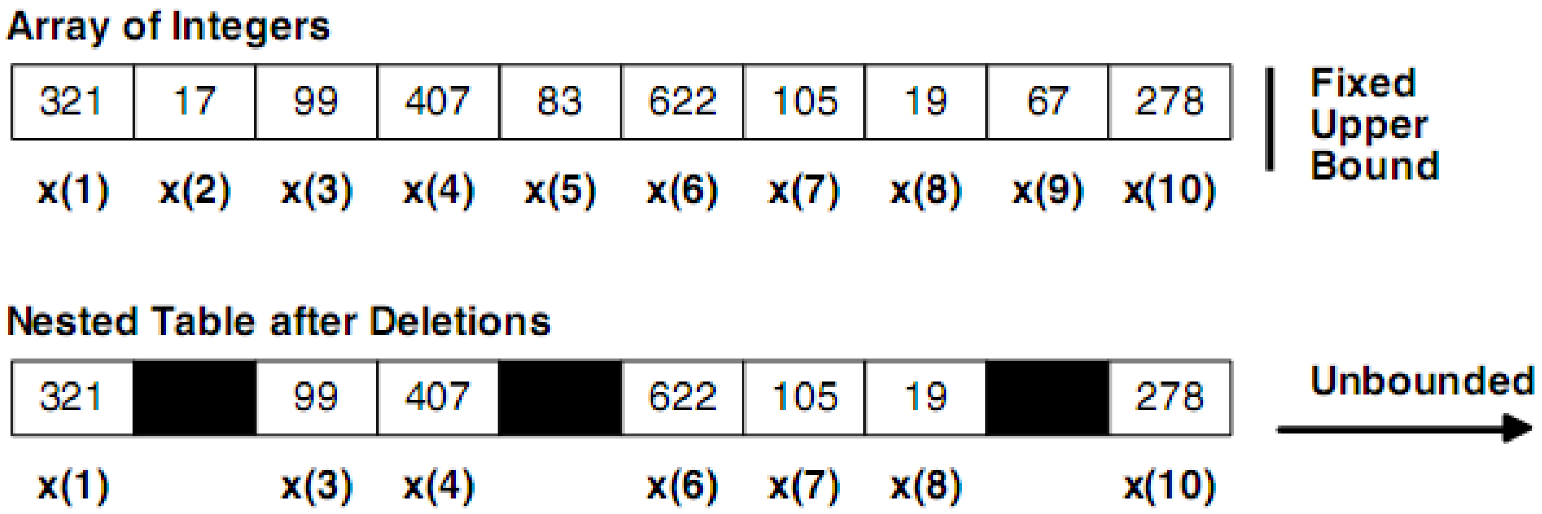
Nested Table of Standalone Stored Type (pag. 156)

```
CREATE OR REPLACE TYPE nt_type IS TABLE OF NUMBER;
CREATE OR REPLACE PROCEDURE print_nt (nt nt_type) IS
 i  NUMBER;
BEGIN
 i := nt.FIRST;
 IF i IS NULL THEN
   DBMS_OUTPUT.PUT_LINE('nt is empty');
 ELSE
   WHILE i IS NOT NULL LOOP
     DBMS_OUTPUT.PUT('nt.(' || i || ') = '); print(nt(i));
     i := nt.NEXT(i);
   END LOOP;
 END IF;
 DBMS_OUTPUT.PUT_LINE('---');
END print_nt;
```

```
DECLARE
   -- nested table variable initialized to empty
  nt nt_type := nt_type();
 BEGIN
  print_nt(nt);
  nt := nt_type(90, 9, 29, 58);
  print_nt(nt);
END;
/
```

```
anonymous block completed
nt is empty
---
nt.(1) = 90
nt.(2) = 9
nt.(3) = 29
nt.(4) = 58
---
```

Array and Nested Table



A nested table is appropriate when:

- The number of elements is not set.
- Index values are not consecutive.
- You must delete or update some elements, but not all elements simultaneously.

Multidimensional Collections (pag. 162)

- Although a collection has only one dimension, you can model a multidimensional collection with a collection whose elements are collections.

- Exemplo (next)

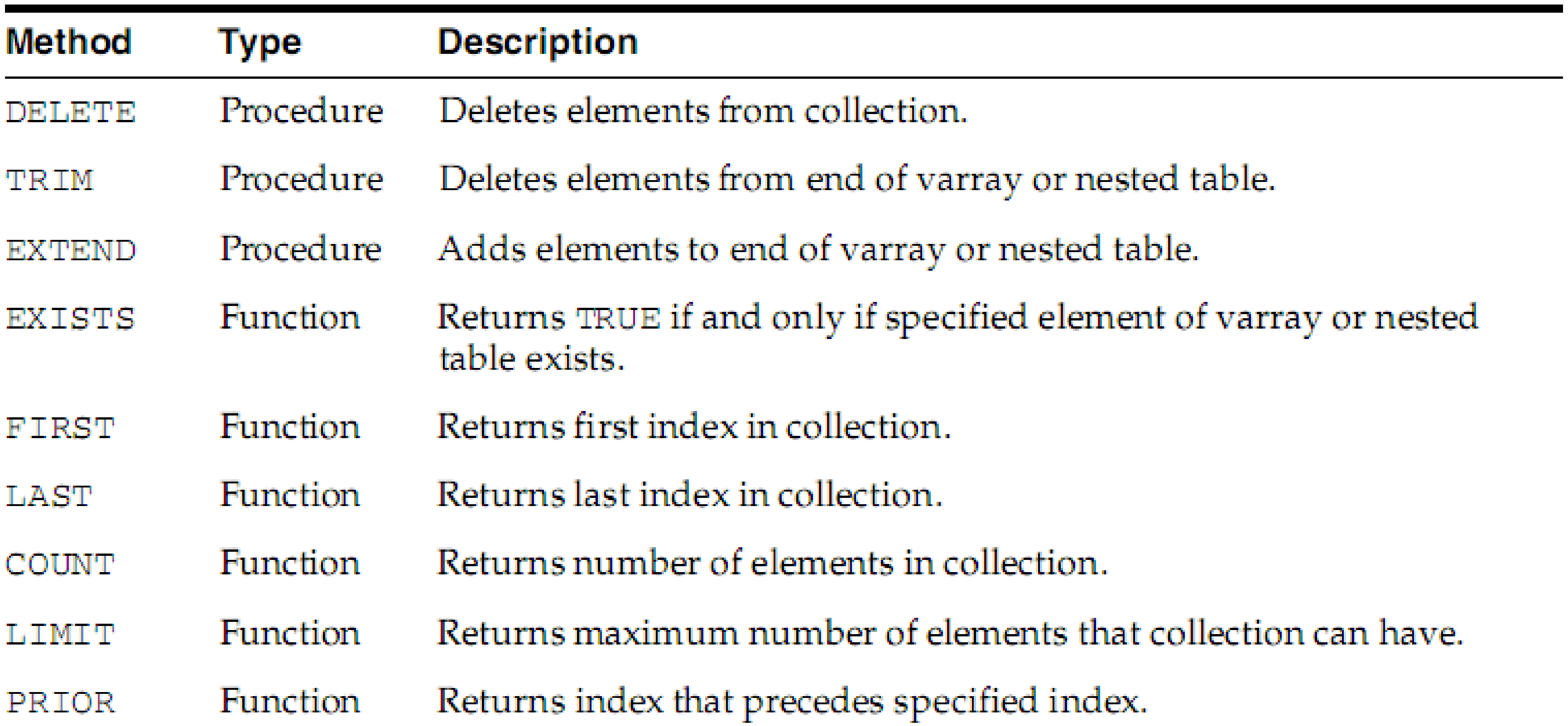
Two-Dimensional Varray (Varray of Varrays)

```
DECLARE
  TYPE t1 IS VARRAY(10) OF INTEGER;  -- varray of integer
  va t1 := t1(2,3,5);
  TYPE nt1 IS VARRAY(10) OF t1;      -- varray of varray of integer
  nva nt1 := nt1(va, t1(55,6,73), t1(2,4), va);
  i INTEGER;
  va1 t1;
BEGIN
  i := nva(2)(3);
  DBMS_OUTPUT.PUT_LINE('i = ' || i);
  nva.EXTEND;
  nva(5) := t1(56, 32);          -- replace inner varray elements
  nva(4) := t1(45,43,67,43345);  -- replace an inner integer element
  nva(4)(4) := 1;                -- replace 43345 with 1
  nva(4).EXTEND;    -- add element to 4th varray element
  nva(4)(5) := 89;  -- store integer 89 there
END;
/
```

```
anonymous block completed
i = 73
```

Experimentar com diferentes indices.

Collection Methods (page. 166)



Method	Type	Description
DELETE	Procedure	Deletes elements from collection.
TRIM	Procedure	Deletes elements from end of varray or nested table.
EXTEND	Procedure	Adds elements to end of varray or nested table.
EXISTS	Function	Returns TRUE if and only if specified element of varray or nested table exists.
FIRST	Function	Returns first index in collection.
LAST	Function	Returns last index in collection.
COUNT	Function	Returns number of elements in collection.
LIMIT	Function	Returns maximum number of elements that collection can have.
PRIOR	Function	Returns index that precedes specified index.

Exemplos (page 168)

```
DECLARE
  TYPE t1 IS VARRAY(10) OF INTEGER;  -- varray of integer
  va t1 := t1(2,3,5);
  TYPE nt1 IS VARRAY(10) OF t1;      -- varray of varray of integer
  nva nt1 := nt1(va, t1(55,6,73), t1(2,4), va);
  i INTEGER;
  va1 t1;
BEGIN
  i := nva(2)(3);
  DBMS_OUTPUT.PUT_LINE('i = ' || i);
  nva.EXTEND;
  nva(5) := t1(56, 32);                  -- replace inner varray elements
  nva(4) := t1(45,43,67,43345);          -- replace an inner integer element
  nva(4)(4) := 1;                        -- replace 43345 with 1
  nva(4).EXTEND;                         -- add element to 4th varray element
  nva(4)(5) := 89;                       -- store integer 89 there
END;
/
```

Record Variables

You can create a record variable in any of these ways:

- Define a RECORD type and then declare a variable of that type.
- Use %ROWTYPE to declare a record variable that represents either a full or partial row of a database table or view.
- Use %TYPE to declare a record variable of the same type as a previously declared record variable.

Record type

```
DECLARE
  TYPE TPoint IS RECORD (id varchar2(20),x NUMBER, y NUMBER,z number);
  p1 TPoint;
  p2 TPoint;
  p3 TPoint;

  function soma_pontos(p1 in TPoint, p2 in TPoint) return TPoint is
     res TPoint;
  begin
    res.x:=p1.x+p2.x; res.y:=p1.y+p2.y; res.z:=p1.z+p2.z;
    res.id:='soma('||p1.id||','||p2.id||')';
    return res;
  end soma_pontos;

BEGIN
  p1.id:='p1';p1.x:=4;   p1.y:=2;   p1.z:=2;
  p2.id:='p2';p2.x:=1;   p2.y:=2;   p2.z:=3;
  p3:=soma_pontos(p1,p2);
  dbms_output.put_line('p3.ID=' || p3.id);
  dbms_output.put_line('p3.x:' || p3.x||'   p3.y:' || p3.y||'   p3.z:'||p3.z);
END;
```

```
anonymous block completed
p3.ID=soma(p1,p2)
p3.x:5   p3.y:4   p3.z:5
```

Rowtype

```
/*Obter um motor da relação de motores, dado o seu
    número como entrada */

Create Procedure get_motor (MotID  IN
    motores.motorID%TYPE,
                     motor_ret    OUT motores%ROWTYPE) IS
BEGIN
   SELECT * INTO motor_ret
   FROM motores
   WHERE MotorID = MotID;
END;
```

Type

```
DECLARE
  TYPE RecordTyp IS RECORD (
    last employees.last_name%TYPE
    id   employees.employee_id%TYPE
  );

  rec1 RecordTyp;

BEGIN
  SELECT last_name, employee_id INTO rec1
  FROM employees
  WHERE job_id = 'AD_PRES';
  DBMS_OUTPUT.PUT_LINE ('Employee #' || rec1.id || ' = ' || rec1.last);
END;
```

Result:

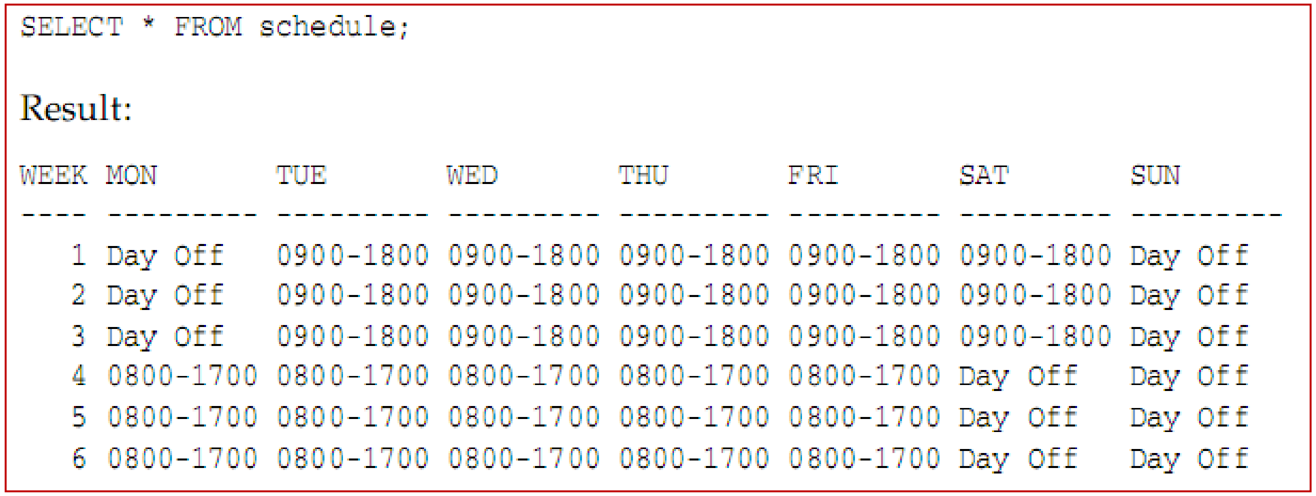
```
Employee #100 = King
```

Updating Rows with Record (p. 197)

```
DECLARE
  default_week  schedule%ROWTYPE;
BEGIN
  default_week.Mon := 'Day Off';
  default_week.Tue := '0900-1800';
  default_week.Wed := '0900-1800';
  default_week.Thu := '0900-1800';
  default_week.Fri := '0900-1800';
  default_week.Sat := '0900-1800';
  default_week.Sun := 'Day Off';

  FOR i IN 1..3 LOOP
    default_week.week    := i;

    UPDATE schedule
    SET ROW = default_week
    WHERE week = i;
  END LOOP;
```



```
SELECT * FROM schedule;
```

Result:

```
WEEK MON       TUE       WED       THU       FRI       SAT       SUN
---- --------- --------- --------- --------- --------- --------- ---------
   1 Day Off   0900-1800 0900-1800 0900-1800 0900-1800 0900-1800 Day Off
   2 Day Off   0900-1800 0900-1800 0900-1800 0900-1800 0900-1800 Day Off
   3 Day Off   0900-1800 0900-1800 0900-1800 0900-1800 0900-1800 Day Off
   4 0800-1700 0800-1700 0800-1700 0800-1700 0800-1700 Day Off   Day Off
   5 0800-1700 0800-1700 0800-1700 0800-1700 0800-1700 Day Off   Day Off
   6 0800-1700 0800-1700 0800-1700 0800-1700 0800-1700 Day Off   Day Off
```

Overriding Default Locking

- By default, Oracle Database locks data structures automatically.
- This enables different applications to write to the same data structures without harming each other's data or coordinating with each other.

LOCK TABLE Statement

FOR UPDATE Cursor in CURRENT OF Clause of UPDATE Statemen

- The SELECT statement with the FOR UPDATE clause (SELECT FOR UPDATE statement) selects the rows of the result set and **locks them**.

```
DECLARE
 my_emp_id NUMBER(6);
 my_job_id VARCHAR2(10);
 my_sal    NUMBER(8,2);
 CURSOR c1 IS
  SELECT employee_id, job_id, salary
  FROM employees FOR UPDATE;
BEGIN
 OPEN c1;
 LOOP
  FETCH c1 INTO my_emp_id, my_job_id, my_sal;
  IF my_job_id = 'SA_REP' THEN
   UPDATE employees
   SET salary = salary * 1.02
   WHERE CURRENT OF c1;
  END IF;
  EXIT WHEN c1%NOTFOUND;
 END LOOP;
END;
```

SELECT FOR UPDATE Statement for Multiple Tables

```
DECLARE
 CURSOR c1 IS
  SELECT last_name, department_name
  FROM employees, departments
  WHERE employees.department_id =
  departments.department_id
  AND job_id = 'SA_MAN'
  FOR UPDATE OF salary;
BEGIN
 NULL;
END;
```

Mecanismo de Excepções

- Sebenta2.pdf (pag. 52)
- Reference guide (page 395)

Outros tópicos a explorar

- Estudar a Álgebra relacional (modelo matemático subjacente ao modelo relacional)
  - Para quem precisar de aprofundar BDs
  - Fazer investigação.
- Aprofundar Entidades e relacionamentos
- Aprofundar a Normalização das BD
- Estudar as arquitecturas
  - Integração de sistemas
  - Suporte a sistemas distribuídos, WEB
  - Balanceamento de carga numa BD de grandes dimensões.

- http://download.oracle.com/docs/cd/E12840_01/wls/docs103/jdbc_admin/oracle_rac.html
- http://www.hardware.com.br/artigos/cluster-carga/

Exemplos

- Fazer uma BD para suportar o projecto eléctrico.