Procedural Language/Structured Query Language (PL/SQL)

Các tính năng chính của PL/SQL

- Khối lệnh PL/SQL
- PL/SQL Input và Output
- Biến và hằng số trong PL/SQL
- Cấu trúc điều khiển trong PL/SQL
- Quản lý lỗi trong PL/SQL
- **Trừu tượng dữ liệu PL/SQL (data abstraction)**
- Chương trình con PL/SQL (Subprogram)
- PL/SQL Packages

Trừu tượng dữ liệu PL/SQL

- Con trỏ (Cursor)
- Record
- Thuộc tính %TYPE
- Thuôc tính %ROWTYPE
- Tập hợp (Collection)

Con trỏ (cursor)

- Cursor là một con trỏ tới một vùng nhớ SQL lưu trữ thông tin về việc xử lý một câu lệnh SELECT hoặc DML.
- PL/SQL sử dụng con trỏ tường minh (explicit cursor) và con trỏ tiềm ẩn (implicit cursor).
- Implicit Cursor
  - PL/SQL tạo một implicit cursor mỗi khi chạy một câu lệnh SELECT hoặc DML.
- Explicit Cursor
  - Ta phải khai báo và định nghĩa một explicit cursor, đặt tên và gắn nó với một câu query.

Record

- Record là một nhóm các thành phần dữ liệu, mỗi thành phần này có tên và kiểu dữ liệu riêng của nó.

```
TYPE RecordTyp IS RECORD (
                    field1 NUMBER,
                    field2 VARCHAR2(32) DEFAULT 'something');
```

Thuộc tính %TYPE

- %TYPE cung cấp kiểu dữ liệu của một biến hoặc một column.
  - v_lname employees.last_name%TYPE;
  - v1 v_lname %TYPE

Thuộc tính %ROWTYPE

- Thuộc tính %ROWTYPE cung cấp một kiểu dữ liệu record biểu diễn một row trong table
  - dept_rec departments%ROWTYPE; -- declare record variable
- Ta sử dụng dấu '.' để tham chiếu đến các thành phần trong record này:
  - v_deptid := dept_rec.department_id;

Định nghĩa và khai báo Record

- Định nghĩa record:

```
TYPE type_name IS RECORD (field_declaration[,field_declaration]...);
```

- Ví dụ:

```
DECLARE
        TYPE TimeRec IS RECORD ( seconds SMALLINT,
                                 minutes SMALLINT,
                                 hours SMALLINT);
BEGIN
        ……
END;
```

Định nghĩa và khai báo Record

- Khai báo record

```
DECLARE
        TYPE StockItem IS RECORD ( item_no INTEGER(3),
                                   description VARCHAR2(50),
                                   quantity INTEGER,
                                   price NUMBER(7,2));
        item_info    StockItem; -- declare record
BEGIN
        ...
END;
```

Record (Ví dụ)

```
DECLARE
    TYPE RecordTyp IS RECORD (
                    field1 NUMBER,
                    field2 VARCHAR2(32) DEFAULT 'something');
    rec1  RecordTyp;
BEGIN
    rec1.field1 := 100;

    DBMS_OUTPUT.PUT_LINE
            ('Field1 = ' || TO_CHAR(rec1.field1)  || ', field2 = ' || rec1.field2);
END;
```

Record (ví dụ)

```
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 employee_id = 10;
    DBMS_OUTPUT.PUT_LINE ('Employee #' || rec1.id || ' = ' || rec1.last);
END;
```

Tập hợp (collection)

- Tập hợp cho phép khai báo kiểu dữ liệu cấp cao như là: mảng (array), tập hợp (set) và bảng băm (hash table) như trong các ngôn ngữ khác.
- Trong PL/SQL, mảng được biết đến như là varray (variable-size arrays), tập hợp (set) là nested table, và bảng băm là associative array.

Tập hợp

Collection Type	Number of Elements	Subscript Type	Dense or Sparse	Where Created	Can Be Object Type Attribute
Associative array (or index-by table)	Unbounded	String or integer	Either	Only in PL/SQL block	No
Nested table	Unbounded	Integer	Starts dense, can become sparse	Either in PL/SQL block or at schema level	Yes
Variable-size array (varray)	Bounded	Integer	Always dense	Either in PL/SQL block or at schema level	Yes

Tập hợp

- **Associative Arrays**
  - associative array là một tập hợp các cặp giá trị key-value.
  - Key là giá trị duy nhất, và được dùng để định vị value tương ứng. Key có thể là integer hoặc string.
  - Cú pháp:

```
TYPE type_name IS TABLE OF element_type [NOT NULL]
                    INDEX BY [PLS_INTEGER | string type];
```

  - Ví dụ:

```
TYPE my_array_t IS TABLE OF VARCHAR2(100) INDEX BY PLS_INTEGER;

TYPE country_tab IS TABLE OF VARCHAR2(50) INDEX BY VARCHAR2(5);
```

Associative Array

```
SET SERVEROUTPUT ON
DECLARE
        TYPE country_tab   IS TABLE OF VARCHAR2(50)
                                INDEX BY VARCHAR2(2);
        t_country       country_tab;
BEGIN
    -- Populate lookup
    t_country('UK') := 'United Kingdom';
    t_country('US') := 'United States of America';
    t_country('FR') := 'France';
    t_country('DE') := 'Germany';
    -- Find country name for ISO code "DE"
    DBMS_OUTPUT.PUT_LINE('ISO code "DE" = ' || t_country('DE'));
END;
```

Associative Array (Ví dụ)

```
DECLARE
    TYPE population IS TABLE OF NUMBER INDEX BY VARCHAR2(64);
    city_population     population; -- Associative array variable
BEGIN
    -- Add new elements to associative array:
    city_population('Smallville') := 2000;
    city_population('Midland') := 750000;
    city_population('Megalopolis') := 1000000;

    -- Change value associated with key 'Smallville':
    city_population('Smallville') := 2001;
    -- Print associative array: city_population('Smallville')
    DBMS_Output.PUT_LINE
        ('Population of  Smallville  is ' || city_population('Smallville'));
END;
```

Kết quả: Population of Smallville is 2001

Tập hợp

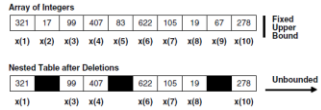
- **Nested Table**
  - Chứa được một số lượng tùy ý các phần tử.
  - Sử dụng số thứ tự để đánh chỉ số và bắt đầu là 1.
  - Có thể định nghĩa như là một kiểu dữ liệu.
  - Cho phép nested table được lưu trữ trong table và thao tác thông qua SQL.

Tập hợp

- **Nested Table:**
  - Cú pháp:

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

  - Sự khác biệt giữa array và nested table:



Nested Table (Ví dụ)

```
DECLARE
    TYPE Roster IS TABLE OF VARCHAR2(15);
    names   Roster;
BEGIN
    names   :=  Roster('D Caruso', 'J Hamil', 'D Piro', 'R Singh');
    DBMS_OUTPUT.PUT_LINE (names(3));
END;
```

Hoặc

```
DECLARE
    TYPE Roster IS TABLE OF VARCHAR2(15);
    names   Roster := Roster('D Caruso', 'J Hamil', 'D Piro', 'R Singh');
BEGIN
    DBMS_OUTPUT.PUT_LINE (names(3));
END;
```

Kết quả: D Piro

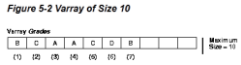
Tập hợp

- Variable-Size Array (Varray)
  - Có kích thước được định sẵn lúc khai báo.
  - Chỉ số (index) trong varray có cận dưới là 1 và cận trên có thể mở rộng.
  - Cũng giống như nested table, varray có thể được định nghĩa như là một kiểu dữ liệu trong SQL và varray có thể lưu trữ trong table nhưng nó ít linh động hơn so với nested table.

Variable-Size Array (Varray)

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

Figure 5-2 Varray of Size 10



Variable-Size Array (Varray)

```
DECLARE
    TYPE varray_type IS VARRAY(5) OF INTEGER;
    v2      varray_type;
BEGIN
        v2 := varray_type(1, 2, 3, 4, 5); -- Up to 5 integers
END;
```

```
DECLARE
        TYPE Calendar IS VARRAY(366) OF DATE;
```

Tập hợp (summary)

Associative Array	type t is table of **something** index by pls_integer;
Nested Table	type t is table of **something**;
VARRAY	type t is varray(123) of **something**;

SQL vs PL/SQL collection types: summary

Scope	What that means	Collection Types
PL/SQL	Declared only in PL/SQL code - no "CREATE OR REPLACE TYPE". SQL doesn't know anything about them. No initialization or extending required - just assign values to any arbitrary element, doesn't even have to be consecutive. Can't treat as a table in queries, e.g. you cannot SELECT * FROM TABLE(myarray)	Associative Array
SQL and PL/SQL	Declared either in PL/SQL code or with "CREATE OR REPLACE TYPE". Must be initialized before use, e.g. myarray mytype := mytype(); Have constructors - you can assign values using mytype('x','y','z'); Must be extended as required, e.g. myarray.EXTEND; to add each array element. Can treat as a table in queries e.g. SELECT * FROM TABLE(myarray) (if created in SQL with CREATE TYPE).	Nested Table VARRAY

Sử dụng các method trong tập hợp

EXISTS
COUNT
LIMIT
FIRST và LAST
PRIOR và NEXT
EXTEND
TRIM
DELETE

Kiểm tra xem phần tử có tồn tại không (EXISTS Method)

- EXISTS(n) trả về TRUE nếu phần tử thứ n trong tập hợp tồn tại. Ngược lại, EXISTS(n) trả về FALSE.

```
IF courses.EXISTS(i) THEN
    courses(i) := new_course;
END IF;
```

Tìm phần tử đầu và cuối của tập hợp (FIRST và LAST Method)

- FIRST và LAST trả về chỉ số (index) đầu và cuối trong một tập hợp.
- Nếu tập hợp đang rỗng (empty), FIRST và LAST sẽ trả về NULL.
- Chỉ số thường là số nguyên, nhưng cũng có thể là chuỗi đối với associative array.

FIRST và LAST Method (ví dụ)

```
set serveroutput on;
DECLARE
   TYPE phone_no_tab IS VARRAY(6) OF VARCHAR2(20);
   phone_nos phone_no_tab ;
BEGIN
   phone_nos := phone_no_tab
                ('0117 942 2508', '03432 345 123', '0111 321 9000');
   FOR i IN phone_nos.FIRST.. phone_nos.LAST
   LOOP
       dbms_output.put_line('phone thu ' || i || ' la: ' || phone_nos(i));
   END LOOP;
END;
```

Đếm số phần tử trong tập hợp (COUNT Method)

- COUNT trả về số phần tử hiện tại của tập hợp.
- Đối với varray, COUNT luôn bằng LAST.
- Đối với nested table, COUNT thường bằng LAST. Nhưng, nếu ta xóa một vài phần tử ở giữa một nested table thì COUNT trở nên nhỏ hơn LAST.
- Khi đếm các phần tử thì COUNT luôn bỏ qua những phần tử đã bị xóa.

COUNT Method (Ví dụ)

```
SET SERVEROUTPUT ON
DECLARE
        TYPE country_tab IS TABLE OF VARCHAR2(50)
                                        INDEX BY VARCHAR2(5);
        t_country       country_tab;
BEGIN
    -- Populate lookup
    t_country('UK') := 'United Kingdom';
    t_country('US') := 'United States of America';
    t_country('FR') := 'France';
    t_country('DE') := 'Germany';

    DBMS_OUTPUT.PUT_LINE
            ('the number of  elements in t_country is:  ' || t_country.COUNT);

END;
```

Kiểm tra kích thước lớn nhất của tập hợp (LIMIT Method)

- Đối với nested table và associative array thì LIMIT luôn trả về NULL (vì 2 loại này không có kích thước lớn nhất).
- Đối với varray, LIMIT trả về số phần tử tối đa mà một varray có thể chứa.

FIRST và LAST Method (ví dụ)

```
DECLARE
    -- Associative array indexed by string:
    TYPE population IS TABLE OF NUMBER INDEX BY VARCHAR2(64);
    city_population      population; -- Associative array variable
    i VARCHAR2(64);
BEGIN
    city_population('Smallville') := 2000;
    city_population('Midland') := 750000;
    city_population('Megalopolis') := 1000000;
    FOR i IN city_population.FIRST.. city_population.LAST  --
    LOOP
        dbms_output.put_line('population of ' || i || ' is: ' || city_population(i));
    END LOOP;
END;
```

Sai vì i tự động ép sang kiểu integer gây ra lỗi

Vòng lặp thông qua các phần tử tập hợp (PRIOR and NEXT Methods)

- PRIOR(n) trả về **chỉ số** của phần tử phía trước n (n là chỉ số).
- NEXT(n) trả về **chỉ số** của phần tử phía sau n.
- Nếu n không có phần tử phía trước thì PRIOR(n) trả về NULL,
- Tương tự, nếu NEXT(n) không có phần tử phía sau thì trả về NULL.
- Khi duyệt các phần tử thì PRIOR và NEXT sẽ bỏ qua các phần tử đã bị xóa.

Vòng lặp thông qua các phần tử tập hợp (PRIOR and NEXT Methods)

```
DECLARE
    -- Associative array indexed by string:
    TYPE population IS TABLE OF NUMBER INDEX BY VARCHAR2(64);
    city_population      population; -- Associative array variable
    city    VARCHAR2(64);
BEGIN
    city_population('Smallville') := 2000;
    city_population('Midland') := NULL;
    city_population('Megalopolis') := 1000000;

    city := city_population.FIRST;
    WHILE city  IS NOT NULL
    LOOP
        DBMS_OUTPUT.PUT_LINE('population of ' || city || '  is: ' || city_population(city));
        city := city_population.NEXT(city);
    END LOOP;
END;
```

Vòng lặp thông qua các phần tử tập hợp (PRIOR and NEXT Methods)

```
DECLARE
    TYPE NumList IS TABLE OF NUMBER;
    n NumList := NumList(1,3,5,7);
    counter   INTEGER;
BEGIN
    counter := n.FIRST;
    WHILE counter IS NOT NULL
    LOOP
        DBMS_OUTPUT.PUT_LINE('Counting up: Element #' || counter || ' = ' ||n(counter));
        counter := n.NEXT(counter);
    END LOOP;
END;
```

Kết quả
Counting up: Element #1 = 1
Counting up: Element #2 = 3
Counting up: Element #3 = 5
Counting up: Element #4 = 7

Tăng kích thước của tập hợp (EXTEND Method)

- Để tăng kích thước của một nested table hoặc varray ta sử dụng EXTEND. Không sử dụng EXTEND với associative array.

Method này có 3 hình thức:
•EXTEND thêm một phần tử null vào tập hợp.
•EXTEND(n) thêm n phần tử null vào tập hợp.
•EXTEND(n,i) thêm n bản copy của phần tử thứ i vào tập hợp.

Ví dụ: Giả sử phần tử ở vị trí 1 có giá trị là 'a', câu lệnh sau thêm 5 phần tử có giá trị là 'a' vào courses:
courses.EXTEND(5,1);

EXTEND Method (ví dụ)

```
set serveroutput on;
DECLARE
        TYPE  phone_no_tab   IS VARRAY(6) OF VARCHAR2(20);
        phone_nos            phone_no_tab ;
BEGIN
        phone_nos := phone_no_tab();
        phone_nos.EXTEND(2);
        phone_nos(1) := '0117 942 2508';
        phone_nos.EXTEND(2,1); -- copy element of phone(1) to 3,4

        FOR i IN phone_nos.FIRST.. phone_nos.LAST    LOOP
                dbms_output.put_line('phone thu ' || i || ' la: ' || phone_nos(i));
        END LOOP;
END;
```

Giảm kích thước một tập hợp (TRIM Method)

- Phương thức này có 2 hình thức:
  - TRIM loại bỏ một phần tử cuối của tập hợp.
  - TRIM(n) loại bỏ n phần tử cuối của tập hợp.
- Không được dùng trim với **associative array**.
- Ví dụ, câu lệnh sau loại bỏ 3 phần tử cuối của courses:
  - courses.TRIM(3);

Xóa phần tử khỏi tập hợp (DELETE Method)

- Phương thức này có nhiều hình thức:
  - DELETE loại bỏ tất cả các phần tử trong tập hợp
  - DELETE (n) loại bỏ phần tử thứ n của tập hợp.
  - DELETE (m, n) loại bỏ tất cả các phần tử có chỉ số dao động từ m...n.
- Trong varray, các phần tử được bố trí dày đặc do đó không thể xóa từng phần tử riêng lẻ.

DELETE Method (Ví dụ)

```
set serveroutput on;
DECLARE
    TYPE NumList IS TABLE OF NUMBER;
    n NumList := NumList(2,4,6,8);
BEGIN
    DBMS_OUTPUT.PUT_LINE ('There are ' || n.COUNT || ' elements in N.');
    n.DELETE(2);
    DBMS_OUTPUT.PUT_LINE ('Now there are ' || n.COUNT || ' elements in N.');
    FOR i IN n.FIRST .. n.LAST LOOP
        IF(n.EXISTS(i)) THEN
            DBMS_OUTPUT.PUT_LINE ('Number  ' || i || ' is: ' || n(i));
        END IF;
    END LOOP;

END;
```

Kết quả

```
There are 4 elements in N.
Now there are 3 elements in N.
Number  1 is: 2
Number  3 is: 6
Number  4 is: 8
```

DELETE Method (Ví dụ)

```
set serveroutput on;
DECLARE
    TYPE NumList IS TABLE OF NUMBER;
    n NumList := NumList(2,4,6,8);
    -- Collection starts with 4 elements.
BEGIN
    DBMS_OUTPUT.PUT_LINE ('There are ' || n.COUNT || ' elements in N.');
    n.EXTEND(3); -- Add 3 new elements at the end.
    DBMS_OUTPUT.PUT_LINE ('Now there are ' || n.COUNT || ' elements in N.');
    n.DELETE(2);
    DBMS_OUTPUT.PUT_LINE ('Now there are ' || n.COUNT || ' elements in N.');
    FOR i IN n.FIRST .. n.LAST LOOP
        IF(n.EXISTS(i)) THEN
            DBMS_OUTPUT.PUT_LINE ('Number ' || i || ' is: ' || n(i));
        END IF;
    END LOOP;

END;
```

DELETE Method (Ví dụ)

```
DECLARE
    TYPE population IS TABLE OF NUMBER INDEX BY VARCHAR2(64);
    city_population     population; -- Associative array variable
     city    VARCHAR2(64);
BEGIN
    city_population('Smallville') := 2000;
    city_population('Midland') := 750000;
    city_population('Megalopolis') := 1000000;
    city_population.DELETE ('Midland');

    city := city_population.FIRST;
    WHILE city  IS NOT NULL
    LOOP
        DBMS_OUTPUT.PUT_LINE('population of ' || city || '  is: ' || city_population(city));
        city := city_population.NEXT(city);
     END LOOP;
END;
```

Kết quả:

```
population of Megalopolis  is: 1000000
population of Smallville  is: 2000
```

Sử dụng tập hợp với record

- Ví dụ: Định nghĩa và khai báo một tập hợp dùng để lưu trữ thông tin nhân viên gồm: manv, hoten, ngayvl.

```
DECLARE
    TYPE RecordTyp IS RECORD (
                        v_manv   nhanvien.manv%TYPE,
                        v_hoten   nhanvien.hoten%TYPE,
                        v_ngayvl  nhanvien.ngayvl%TYPE);

    TYPE  t_nhanvien IS TABLE OF  RecordTyp ;
    ds_nhanvien       t_nhanvien;

BEGIN
    NULL;
END;
```

Sử dụng tập hợp với record

Ví dụ: Lưu trữ nhân viên có id là 2 vào tập hợp này (ở slide trước).

```
DECLARE
    TYPE RecordTyp IS RECORD (
                        v_manv   nhanvien.manv%TYPE,
                        v_hoten   nhanvien.hoten%TYPE,
                        v_ngayvl  nhanvien.ngayvl%TYPE);
    TYPE  t_nhanvien IS TABLE OF  RecordTyp ;

    ds_nhanvien     t_nhanvien  := t_nhanvien();
    v_nv             RecordTyp ;

BEGIN
    SELECT manv, hoten, ngayvl  INTO v_nv  FROM nhanvien WHERE manv=2;
    ds_nhanvien.EXTEND(1);
    ds_nhanvien(1)  := v_nv;
END;
```

Sử dụng tập hợp với record

- Ví dụ: Lưu trữ nhân viên có id là 2 và 3 vào tập hợp.

```
DECLARE
    TYPE RecordTyp IS RECORD (
                        v_manv   nhanvien.manv%TYPE,
                        v_hoten   nhanvien.hoten%TYPE,
                        v_ngayvl  nhanvien.ngayvl%TYPE);
    TYPE  t_nhanvien IS TABLE OF  RecordTyp ;
    ds_nhanvien     t_nhanvien  := t_nhanvien();
    v_nv             RecordTyp ;

BEGIN
    SELECT manv, hoten, ngayvl  INTO v_nv  FROM nhanvien WHERE manv=2;
    ds_nhanvien.EXTEND(1);
    ds_nhanvien(1)  := v_nv;

    SELECT manv, hoten, ngayvl  INTO v_nv  FROM nhanvien WHERE manv=3;
    ds_nhanvien.EXTEND(1);
    ds_nhanvien(2)  := v_nv;
END;
```

Sử dụng tập hợp với record

```
DECLARE
    TYPE RecordTyp IS RECORD (
                        v_manv   nhanvien.manv%TYPE,
                        v_hoten   nhanvien.hoten%TYPE,
                        v_ngayvl  nhanvien.ngayvl%TYPE);
    TYPE  t_nhanvien IS TABLE OF  RecordTyp ;
    ds_nhanvien     t_nhanvien  := t_nhanvien();
    v_nv  RecordTyp ;

BEGIN
    SELECT manv, hoten, ngayvl  INTO v_nv  FROM nhanvien WHERE manv=2;
    ds_nhanvien.EXTEND(1);
    ds_nhanvien(1)  := v_nv;

    SELECT manv, hoten, ngayvl  INTO v_nv  FROM nhanvien WHERE manv=3;
    ds_nhanvien.EXTEND(1);
    ds_nhanvien(2)  := v_nv;

    FOR i IN ds_nhanvien.FIRST.. ds_nhanvien.LAST  LOOP
        IF(ds_nhanvien.EXISTS(i)) THEN
            DBMS_OUTPUT.PUT_LINE (' Nhan vien  ma so: ' ||
                                ds_nhanvien (i).v_manv || ' la: ' ||  ds_nhanvien (i).v_hoten);
        END IF;
    END LOOP;
END;
```

Sử dụng tập hợp với record (cách khác)

```
DECLARE
    TYPE RecordTyp IS RECORD (
                        v_manv   nhanvien.manv%TYPE,
                        v_hoten   nhanvien.hoten%TYPE,
                        v_ngayvl  nhanvien.ngayvl%TYPE);
    TYPE  t_nhanvien IS TABLE OF  RecordTyp ;
    ds_nhanvien    t_nhanvien;

BEGIN
    SELECT manv, hoten, ngayvl  BULK  COLLECT  INTO  ds_nhanvien
    FROM nhanvien WHERE manv=2 OR manv = 3;
    IF(ds_nhanvien.COUNT >0) THEN
        FOR i IN 1 .. ds_nhanvien.COUNT LOOP
            IF(ds_nhanvien.EXISTS(i)) THEN
                DBMS_OUTPUT.PUT_LINE (' Nhan vien  ma so: ' ||
                                ds_nhanvien (i).v_manv || ' la: ' ||  ds_nhanvien (i).v_hoten);
            END IF;
        END LOOP;
    END IF;
END;
```

Sử dụng tập hợp với record

```
DECLARE
    TYPE   EmployeeSet      IS TABLE OF employees%ROWTYPE;
    underpaid        EmployeeSet; -- Holds set of rows from EMPLOYEES table.
BEGIN
    -- With one query, bring all relevant data into collection of records.
    SELECT * BULK COLLECT INTO underpaid FROM employees
    WHERE salary < 5000 ORDER BY salary DESC;

    DBMS_OUTPUT.PUT_LINE (underpaid.COUNT || ' people make less than 5000.');

    FOR i IN 1.. underpaid.COUNT LOOP
        DBMS_OUTPUT.PUT_LINE (underpaid(i).last_name || ' makes ' ||  underpaid(i).salary);
    END LOOP;
END;
```

48

Sử dụng Bulk Binds
For all

```
DECLARE
  TYPE Numlist IS VARRAY (100) OF NUMBER;
  Id NUMLIST := NUMLIST(7902, 7698, 7839);
BEGIN

-- Efficient method, using a bulk bind
FORALL i IN Id.FIRST..Id.LAST   -- bulk-bind the VARRAY
    UPDATE Emp_tab SET Sal = 1.1 * Sal
    WHERE Mgr = Id(i);

-- Slower method, running the UPDATE statements within a regular loop
FOR i IN Id.FIRST..Id.LAST LOOP
    UPDATE Emp_tab SET Sal = 1.1 * Sal
    WHERE Mgr = Id(i);
END LOOP;
END;
```