

Oracle PL/SQL

(Ch 10.5)

Database Management
COP4540, SCS, FIU

A Introduction to PL/SQL

- PL/SQL is Oracle's procedural language extension to standard SQL.
- PL/SQL provides condition handling, iteration, and nested processing in a block-structured language.
- Users can use PL/SQL to codify their business rules through the creation of procedures and packages, to trigger database events, or to add programming logic to the execution of SQL commands.

Database Management
COP4540, SCS, FIU

The Structures of a PL/SQL Program

- **Declarations**
 - Defines and initializes the variables and cursors used in the block.
- **Executable Commands**
 - Uses flow control commands (such as if commands and loops) to execute the commands and assign values to declared variables.
- **Exception Handling**
 - Provides customized handling of error conditions.

Database Management
COP4540, SCS, FIU

A typical PL/SQL Block

```
declare  
  <declarations section>  
begin  
  <executable commands>  
exception  
  <exception handling>  
end;
```

Database Management
COP4540, SCS, FIU

Declarations Section

- The declarations section starts with the **declare** keyword, followed by a list of variable and cursor definitions.
- Users can define variable to have constant values.
- Variables can inherit datatypes from existing columns and query results.

Database Management
COP4540, SCS, FIU

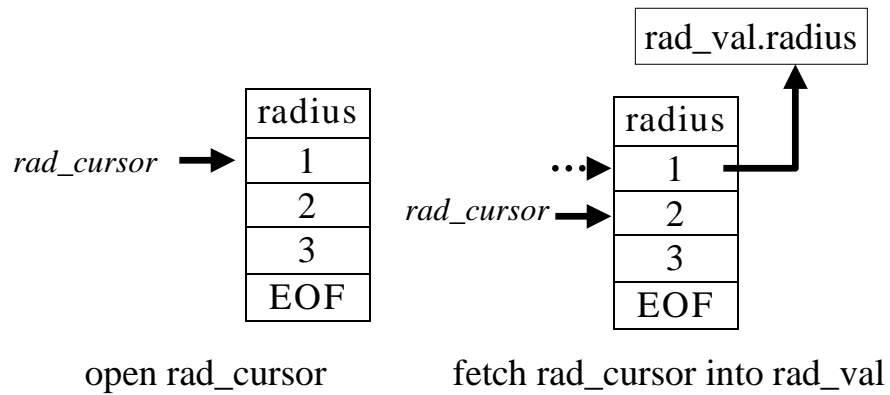
Example

```
declare
  pi      constant NUMBER(9,7) := 3.1415926;
  radius  INTEGER(5);
  area    NUMBER(14,2);
begin
  radius := 3;
  area := pi * power(radius, 2);
  insert into AREAS values (radius, area);
end;
```

Database Management
COP4540, SCS, FIU

About Cursor

```
cursor rad_cursor is  
select * from RADIUS_VALUES;  
rad_val rad_cursor%ROWTYPE
```



Database Management
COP4540, SCS, FIU

Example

```
declare  
  pi      constant Number(9,7) := 3.1415926;  
  area    NUMBER(14,2);  
  cursor  rad_cursor is  
    select * from RADIUS_VALUES;  
  rad_val rad_cursor%ROWTYPE  
begin  
  open rad_cursor;  
  fetch rad_cursor into rad_val;  
  area := pi * power(rad_val.radius, 2);  
  insert into AREAS values (rad_val.radius, area);  
  close rad_cursor;  
end;
```

Database Management
COP4540, SCS, FIU

Executable Commands Section

- The executable commands section starts with the keyword **begin**.
- In the executable commands section, users manipulate the variables and cursors declared in the declarations section.

Database Management
COP4540, SCS, FIU

Conditional Logic

- Within PL/SQL, users can use **if**, **else**, and **elsif** commands to control the flow of commands within the executable commands section.

```
If <some condition>  
    then <some command>  
elsif <some condition>  
    then <some command>  
else <some command>  
endif
```

Database Management
COP4540, SCS, FIU

Example

```
declare
  pi          constant Number(9,7) := 3.1415926;
  area       NUMBER(14,2);
  cursor      rad_cursor is
    select * from RADISU_VALUS;
  rad_val     rad_cursor%ROWTYPE
begin
  open rad_cursor;
  fetch rad_cursor into rad_val;
  area := pi * power(rad_val.radius, 2);
  if area > 30
    then
      insert into AREAS values (rad_val.radius, area);
    endif;
  close rad_cursor;
end;
```

Database Management
COP4540, SCS, FIU

Loops

- Simple loops.
 - A loop that keeps repeating until an *exit* or *exit when* statement is reached within loop
- FOR loops.
 - A loop that repeats a specified number of times.
- WHILE loops.
 - A loop that repeats until a condition is met.

Database Management
COP4540, SCS, FIU

Example

```
declare
  pi      constant NUMBER(9,7) := 3.1415926;
  radius  INTEGER(5);
  area    NUMBER(14,2);
begin
  radius := 3;
  loop
    area := pi * power(radius, 2);
    insert into AREAS values (radius, area);
    radius := radius + 1;
    exit when area > 100;
  end loop;
end;
```

Database Management
COP4540, SCS, FIU

Example

```
declare
  pi      constant NUMBER(9,7) := 3.1415926;
  radius  INTEGER(5);
  area    NUMBER(14,2);
begin
  for radius in 1 .. 7 loop
    area := pi * power(radius, 2);
    insert into AREAS values (radius, area);
  end loop;
end;
```

Database Management
COP4540, SCS, FIU

Example

```
declare
  pi      constant NUMBER(9,7) := 3.1415926;
  radius  INTEGER(5);
  area    NUMBER(14,2);
begin
  radius := 1;
  while radius <= 7
    loop
      area := pi * power(radius, 2);
      insert into AREAS values (radius, area);
      radius = radius + 1;
    end loop;
end;
```

Database Management
COP4540, SCS, FIU

Using Loops with Cursor

- Cursors have four attributes that can be used in the program.
 - FOUND
 - A record can be fetched from the cursor.
 - NOTFOUND
 - No more records can be fetched from the cursor.
 - ISOPEN
 - The cursor has been opened.
 - ROWCOUNT
 - The number of rows fetched from the cursor so far.

Database Management
COP4540, SCS, FIU

Example

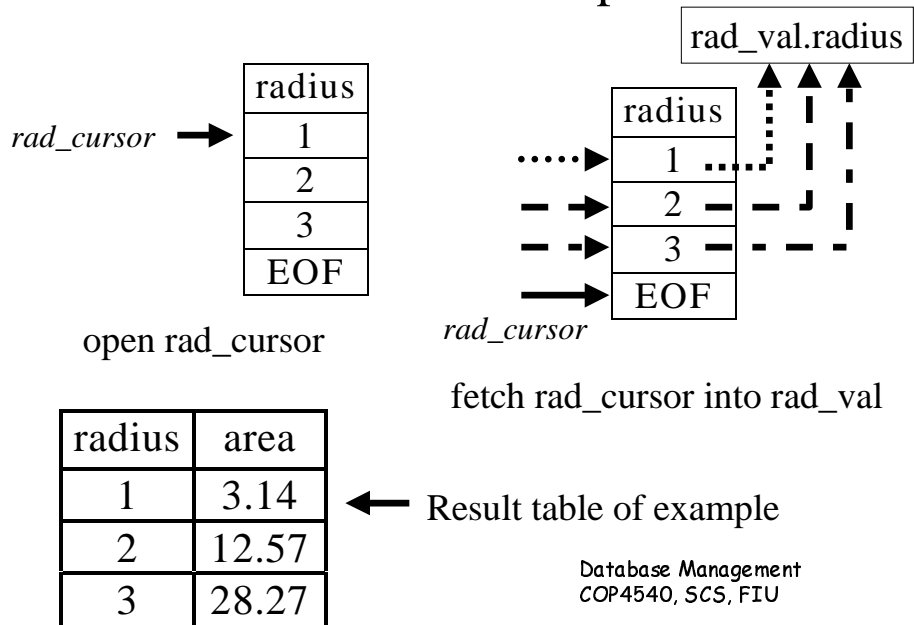
```

declare
  pi          constant Number(9,7) := 3.1415926;
  area NUMBER(14,2);
  cursor      rad_cursor is
    select * from RADISU_VALUS;
  rad_val     rad_cursor%ROWTYPE
begin
  open rad_cursor;
  loop
    fetch rad_cursor into rad_val;
    exit when rad_cursor%NOTFOUND;
    area := pi * power(rad_val.radius, 2);
    insert into AREAS values (rad_val.radius, area);
  end loop;
  close rad_cursor;
end;

```

Database Management
COP4540, SCS, FIU

Results of Example



Exception Handling Section

- The exception handling section of a PL/SQL program is optional.
- The except handling section starts with the keyword `exception`.
- When user-defined or system-related exception (errors) are encountered, the control of the PL/SQL program shifts to the exception handling section.

Database Management
COP4540, SCS, FIU

Example

```
declare
  pi          constant NUMBER(9,7) := 3.1415926;
  radius      INTEGER(5);
  area        NUMBER(14,2);
  var_test    NUMBER(14,2);
begin
  radius := 3;
  loop
    var_test := 1 / (radius -4);
    area := pi * power(radius, 2);
    insert into AREAS values (radius, area);
    radius := radius + 1;
    exit      when area > 100;
  end loop;
exception
  when ZERO_DIVIDE
  then insert into area values (0, 0);
end;
```

Database Management
COP4540, SCS, FIU

Example

```
DECLARE
    v_fname employee.fname%type;
    v_minit employee.minit%type;
    v_lname employee.lname%type;
    v_address employee.address%type;
    v_salary employee.salary%type;

BEGIN
    SELECT fname, minit, lname, address, salary
    INTO v_fname, v_minit, v_lname, v_address, v_salary
    FROM employee
    WHERE salary = (SELECT max(salary) FROM employee);
    DBMS_OUTPUT.PUT_LINE(v_fname, v_minit, v_lname, v_address, v_salary);

EXCEPTION
    WHEN OTHERS
    DBMS_OUTPUT.PUT_LINE('Error Detected');

END;
```

Database Management
COP4540, SCS, FIU

Example

```
DECLARE
    avg_salary NUMBER;

BEGIN
    SELECT avg(salary) INTO avg_salary FROM employee;

    UPDATE employee SET salary = salary * 1.1
    WHERE salary < avg_salary;

    SELECT avg(salary) INTO avg_salary FROM employee;

    COMMIT;

EXCEPTION
    WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('ERROR in Salary Update');
    ROLLBACK;

END;
```

Database Management
COP4540, SCS, FIU