PL/SQL

Lecture 3 :
Procedures and functions

Procedures and functions (01/11)

- Why ?
  - Sometimes, you need to get a result from a complex calculus
  - Sometimes, you also need to reuse parts of your program
  - Like tables, they can be used by anyone who has the appropriate rights
- What ?
  - Subprograms that use a set of *parameters*
  - Can be created, modified and dropped

Procedures and functions (02/11)

- Procedure
  - Subprogram that does not return any value
  - Mainly used to perform a series of actions
- Syntax

```
CREATE OR REPLACE PROCEDURE
   procedure_name  (  parameters )
      {AS | IS}
    /* local variables */
BEGIN
    /* procedure_body */
END procedure_name;
```

Procedures and functions (03/11)

- Create *"or replace"* ?
  - Allows the modification of the procedure
- Parameters ?
  - What the procedure needs to be executed
  - Syntax : parameter_name1 PARAMETER_TYPE1, parameter_name2 PARAMETER_TYPE2, ...
  - Example : a INT, b VARCHAR2, ...
- "As" or "Is" ?
  - AS : in case of a standalone procedure
  - IS : most often (there aren't many differences)

Procedures and functions (04/11)

- Procedure : Example 1 (standalone)

```
CREATE OR REPLACE PROCEDURE greetings AS
BEGIN
    DBMS_OUTPUT.put_line('Hello World');
END;
/
```

- Invoking a standalone procedure
  - In SQL*PLUS : command CALL or EXECUTE

```
EXECUTE greetings;
```

  - In a PL/SQL program : just call the procedure

```
BEGIN greetings; END;
/
```

Procedures and functions (05/11)

- Procedure : Example 2 (inside a program)

```
DECLARE
    message VARCHAR2(30);

PROCEDURE writeText (mess VARCHAR2) IS
    BEGIN
        DBMS_OUTPUT.put_line(mess);
    END;

BEGIN
    message := 'Goodbye';
    writeText(message);
END;
/
```

Procedures and functions (06/11)

- "In" and "Out" parameters
  - Sometimes, parameters are modified inside the code of the procedure
  - You can decide whether the modifications are saved or not
  - Syntax : name {IN | OUT | IN OUT} mode
- Three options
  - IN : value is read only, eventual changes are unsaved (by default option)
  - OUT : no input value, changes are saved
  - IN OUT : value is read, changes are saved

Procedures and functions (07/11)

- Example

```
DECLARE
    value NUMBER;
PROCEDURE incr (val IN OUT NUMBER) IS
    BEGIN
        val := val + 1;
    END;
BEGIN
    value := 5;
    incr(value);
    DBMS_OUTPUT.put_line(value);
END;
/
```


- What is the result ? And if IN OUT is removed ?

Procedures and functions (08/11)

- Function
  - Subprogram that returns one (and only one) value
  - Mainly used to perform a calculus
- Syntax

```
CREATE OR REPLACE FUNCTION
    function_name ( parameters )
RETURN /* datatype */ {AS | IS}
    /* local variables */
BEGIN
    /* procedure_body */
    RETURN returned_value_name;
END procedure_name;
```

Procedures and functions (09/11)

- Function : example

```
CREATE OR REPLACE FUNCTION addition(a Number, b Number)
RETURN Number IS
        c Number;
    BEGIN
        c := a+b;
        RETURN c;
    END;
    /
```

- Parameters are always IN

Procedures and functions (10/11)

- How to invoke
  - In a PL/SQL program

    VARIABLE variable_name TypeName

EXECUTE :variable_name := function_name(parameters);

Print variable_name

Example :

Variable result Number;

Execute:result := addition(3,5);

Print result;

Procedures and functions (10/11)

- On SQL*Plus prompt
  - We use a SQL pseudo-table named DUAL
  - SELECT function_name(parameters) FROM DUAL
  - Example : SELECT addition(3,5) FROM DUAL;
- Recursive functions
  - Subprograms can call other subprograms...
  - ... but they can also call themselves => recursion