Application

Other DBMS

* + PL/SQL is portable
  + You can declare variable
  + You can program with procedural language control structure
  + Can error handle

Benefits of subprograms (modulation)

* + Easy maintained
  + Improved data security and integrity
  + Improve performance
  + Improved code clarity

**PL/SQL Block structure**

DECLARE (optional)

Variable, cursors, user-defined exception

BEGIN (mandatory)

-SQL statement

-PL/SQL statement

EXCEPTION(optional)

Action to perform error occurs

END; (mandatory)

Note – DBMS\_OUTPUT.PUT\_LINE(variable)

We need to instruct to server to show output values

Prompt – SET SERVEROUTPUT ON

Inherit the columns data types

<table\_name>.<col\_name>%TYPE

<table\_name>%ROWTYPE

Block types

Anonymos

Procedure

Functions

SQL cursers

A curser is a private SQL work area

* Implicit curser-to parse and execute your SQL statement
* Explicit curser-declared by programmer

SQL%ROWCOUNT - # of rows affected by most recent SQL statemet

SQL%FOUND – Boolean attribute that evaluate to TRUE if the most recent SQL statement affected one or more rows

SQL%ISOPEN – always evaluate to FLASE because PL/SQL close implicit cursor immediate after they are executing

IF condition THEN

Statement;

[ELSIF condition THEN

Statemet;]

[ELSE

Statement;]

END IF;

Loops

Basic,for,while

Basic loop

LOOP

Statement 1;

…

EXIT [WHEN condition];

END LOOP;

DECLARE

v\_row number;

v\_col number;

BEGIN

v\_row := 0;

v\_col := 0;

LOOP

v\_col := 0;

v\_row := v\_row+1;

LOOP

v\_col := v\_col+1;

DBMS\_OUTPUT.PUT('\*');

EXIT WHEN v\_col = v\_row;

END LOOP;

dbms\_output.new\_line;

EXIT WHEN v\_row = 10;

END LOOP;

END;

While condition

Statement 1;

End while;

FOR counter IN [Reverse]

Lower\_bound … upper\_bound LOOP

Statement1;

Statement1;

END LOOP:

Record more than one row – index by table

Two component

Primary key of data type BINARY\_INTEGER

Column of scalar of record data type

TYPE type\_name IS TABLE OF

{column\_type | variable%TYPE}

EXCEPTION

NO\_DATA\_FOUND

TOO\_MANY\_ROWS

INVALID\_CURSOR

ZERO\_DIVIDE

DUP\_VAL\_ON\_INDEX

User defined exceptions

DECLARE

ex\_fk\_constraint EXCEPTION;

PRAGMA EXCEPTION\_INIT

(ex\_fk\_constraint, -02292);

v\_dID DEPARTMENTS.department\_id%TYPE;

BEGIN

v\_dID := 10;

DELETE FROM DEPARTMENTS

WHERE department\_id = &v\_dID;

EXCEPTION

WHEN ex\_fk\_constraint THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR');

DBMS\_OUTPUT.PUT\_LINE('ERROR CODE '|| SQLCODE);

DBMS\_OUTPUT.PUT\_LINE('ERROR MSG '|| SQLERRM);

DBMS\_OUTPUT.PUT\_LINE('TIME '|| TO\_CHAR(SYSDATE,'YYYY/MM/dd HH24:mm'));

END;

/

RAISE\_APPLICATION\_ERROR

Raise\_application\_error(error\_number, message[, {TRUE | FALSE}]);

# Create procedure

Database object

* 2 type(procedure and functions)

<header>

IS | AS

BEGIN

END;

* Easy maintain
* Improve data security and integrity
* improve performance
* Improve code clarity

<header>

CREATE [OR REPALCE ] PROCEDURE procedure\_name

[(para 1 [mode 1] datatype 1,

para 2 [mode 2] datatype 2,

…)]

CREATE OR REPLACE PROCEDURE proc\_get\_service

(p\_hd IN employees.hire\_date%TYPE, p\_service OUT NUMBER)

IS

v\_hd employees.hire\_date%TYPE;

v\_service NUMBER;

BEGIN

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PROCEDURE : PROC\_GET\_SERVICE

PURPOSE : TAKE HIRE\_DATE AND SEND THE SERVICE YEARS TO P\_SERVICE FORMAL PARAM

DATE BY DESC

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

v\_hd:= p\_hd;

SELECT TRUNC((SYSDATE - v\_hd)/365)

INTO v\_service

FROM dual;

p\_service:=v\_service;

END proc\_get\_service;

CREATE OR REPLACE PROCEDURE proc\_ret\_emprec(p\_emp IN OUT employees%ROWTYPE)

IS

v\_salary employees.salary%TYPE:=p\_emp.salary;

v\_hd employees.hire\_date%TYPE:=p\_emp.hire\_date;

v\_service NUMBER;

BEGIN

SELECT \*

INTO p\_emp

FROM employees

WHERE employee\_id = p\_emp.employee\_id;

proc\_get\_service(v\_hd, v\_service);

IF v\_service > 30 THEN

p\_emp.salary := 1.25\*p\_emp.salary;

ELSIF v\_service > 25 THEN

p\_emp.salary := 1.2\*p\_emp.salary;

ELSIF v\_service > 20 THEN

p\_emp.salary := 1.15\*p\_emp.salary;

ELSE

p\_emp.salary := 1.1\*p\_emp.salary;

END IF;

END proc\_ret\_emprec;

DECLARE

v\_emp employees%ROWTYPE;

CURSOR rec\_emp IS

SELECT \* FROM employees;

BEGIN

FOR one\_rec\_emp1111 IN rec\_emp LOOP

proc\_ret\_emprec(one\_rec\_emp);

DBMS\_OUTPUT.PUT\_LINE(one\_rec\_emp.first\_name||' '||one\_rec\_emp.employee\_id||' '||one\_rec\_emp.salary);

END LOOP;

END;

/

DECLARE

v\_emp employees%ROWTYPE;

CURSOR rec\_emp IS

SELECT employee\_id FROM employees;

BEGIN

v\_emp.employee\_id := 1500;

proc\_ret\_emprec(v\_emp);

DBMS\_OUTPUT.PUT\_LINE(v\_emp.first\_name||' '||v\_emp.employee\_id||' '||v\_emp.salary);

END;

Dynamic SQL

EXECUTE IMMEDIATE v\_sql;