LAB # 13: PL/SQL – SERVER SIDE PROGRAMMING

- Benefits

- More powerful than pure SQL because it combines the power of SQL and
  - Iteration (loops)
  - Selection (Ifs)
  - Block Structures
  - Stored Procedures
  - etc.

- Basic Constructs

- Basic Structure
- Running a program
- Variables
- SELECT INTO
- Comments
- IFs
- LOOPs
- Output

-- Basic Structure

DECLARE

BEGIN

EXCEPTION

END;
.
/

- OUTPUT ...

```
SET SERVEROUTPUT ON;

BEGIN
    DBMS_OUTPUT.PUT_LINE('This is my first program');
END;
/
```

Before executing code that contains DBMS_OUTPUT.PUT_LINE, must run (in a current session) at SQL prompt: set serveroutput on.

System Package DBMS_OUTPUT exists in Oracle Dictionary. It accepts and return SINGLE argument.

--- Basic Structure: Example

```
DECLARE
    v_id                    INTEGER;
    v_empno           NUMBER;
BEGIN
   v_id := 1234567;  --This id does not exist

   SELECT EMPNO
     INTO V_EMPNO
     FROM  EMP
     WHERE empno = v_id;

DBMS_OUTPUT.PUT_LINE('Value is '||v_empno);

EXCEPTION
   WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('No record exists');

 END;
/
```


--- Basic Structure: Example

```
DECLARE
    v_id              INTEGER;

BEGIN
   v_id := 1234567; --This id does not exist

   DELETE
   FROM  EMP
   WHERE id = v_id;

END;
/
```

-- Running a Program

```
DECLARE
        [redacted]
BEGIN
        [redacted]
EXCEPTION
        [redacted]
END;
/
```


-- Variables

- Common Data Types
  - NUMBER
  - DATE
  - INTEGER
  - VARCHAR2
  - CHAR
  - BOOLEAN
- Declaration
  - V_salary NUMBER(9,2);
  - V_id INTEGER;
  - V_dob DATE;
  - V_name VARCHAR2(35);
  - V_gender CHAR;
  - V_salary emp.salary%TYPE;

-- SELECT INTO

```
DECLARE

    v_job               emp.job%TYPE;
    v_sal               emp.sal%TYPE;
    v_empno             emp.empno%TYPE;

BEGIN

    v_empno := 1234567; --This id does not exist

    SELECT job, sal
     INTO v_job,v_sal
     FROM emp
      WHERE  empno = v_empno;

END;
/
```


-- Another Example

```
declare
  v_ename    emp.ename%TYPE;
  v_sal                    emp.sal%TYPE;
begin
   select ename, sal
   into v_ename, v_sal
    from emp
    where empno = '7844'; --This id exists
 DBMS_OUTPUT.PUT_LINE('solution is:'||v_ename||'---'||v_sal);

exception
  When others then
   DBMS_OUTPUT.PUT_LINE('no record');
end;
/
```

Before executing code that contains DBMS_OUTPUT.PUT_LINE,
must run at SQL prompt: set serveroutput

-- Comments

```
DECLARE
    /*
       This block will do
        so and so and so.
  */
    v_job                   emp.job%TYPE;
    v_sal                   emp.sal%TYPE;
    v_empno                 emp.empno%TYPE;

BEGIN

     v_empno := 1234567;  -- The use of this sentence is so and so.

     -- The following SELECT INTO statement will do so and so.

     SELECT job, sal
      INTO v_job,v_sal
      FROM emp
      WHERE  empno = v_empno;

END;
/
```


-- IFs

- IF – END IF
- IF – ELSE – END IF
- IF – ELSIF – ELSE – END IF

--- IF – END IF

```
DECLARE
    ...
    ...
BEGIN
    ...
    ...
    v_commison := 7500;

    IF v_dept = 10 THEN
        v_commision := 5000;
     END IF;
     ...
     ...
END;
/
```


--- IF – ELSE – END IF

```
DECLARE
    ...
    ...
BEGIN
    ...
    ...

    IF v_dept = 10 THEN
        v_commision := 5000;
     ELSE
        v_commision := 7500;
     END IF;
     ...
     ...
END;
/
```

--- IF – ELSIF – ELSE – END IF

```
DECLARE
    ...
    ...
BEGIN
    ...
    ...

    IF v_dept = 10 THEN
        v_commision := 5000;
    ELSIF v_dept = 20 THEN
        v_commison := 5500;
        DBMS_OUTPUT.PUT_LINE(v_commison);
    ELSIF v_dept = 30  THEN
        v_commison := 6200;
    ELSE
        v_commision := 7500;
    END IF;
    ...
    ...
END;
/
```


-- LOOPs

- LOOP – EXIT WHEN – END LOOP
- FOR – LOOP – END LOOP
- WHILE – LOOP – END LOOP

--- LOOP – EXIT WHEN – END LOOP

```
DECLARE
    ...
    v_deptno   dept.deptno%TYPE  := 50;
    v_counter  integer := 1;
    ...
BEGIN
    ...
    LOOP
      INSERT INTO dept(deptno)
             VALUES(v_deptno);
      v_counter := v_counter + 1;
      v_deptno := v_deptno + 10;
      EXIT WHEN v_counter > 5;
   END LOOP;
   ...
END;
/
```


--- FOR – LOOP - END LOOP

```
DECLARE
    ...
    v_deptno   dept.deptno%TYPE  := 50;
    v_counter  integer;
    ...
BEGIN
    ...
    FOR v_counter IN 1..5 LOOP
        INSERT INTO dept(deptno)
               VALUES(v_deptno);
        v_deptno := v_deptno + 10;
    END LOOP;
   ...
END;
/
```

--- WHILE – LOOP - END LOOP

```
DECLARE
     ...
     v_deptno   dept.deptno%TYPE  := 50;
     v_counter  integer;
     ...
BEGIN
     ...
 v_counter := 1;
     WHILE v_counter <= 5 LOOP
          INSERT INTO dept(deptno)
                VALUES(v_deptno);
          v_deptno := v_deptno + 10;
     END LOOP;
     ...
END;
/
```

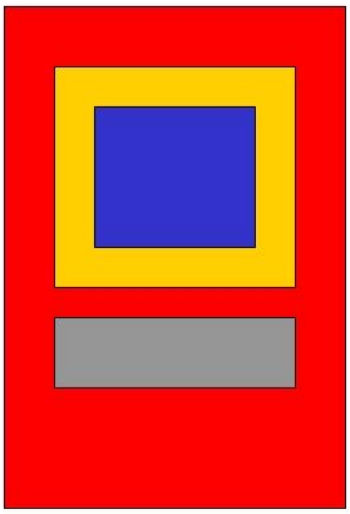

… - OUTPUT

```
SET SERVEROUTPUT ON;

DECLARE

   v_sum_sal    emp.sal%TYPE;
   v_deptno     emp.deptno%TYPE := 10;

BEGIN

     SELECT SUM(sal)
     INTO v_sum_sal
     FROM emp
     WHERE deptno = v_deptno;

     DBMS_OUTPUT.PUT_LINE('The sum is ‘ || TO_CHAR(v_sum_sal));

END;
/
```

- Anonymous Block

```
DECLARE
    v_id                INTEGER;

BEGIN
   v_id := 1234567;

   DELETE
   FROM  EMP
   WHERE id = v_id;

EXCEPTION
   WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE('No record exists');

 END;
/
```


-- Nesting Anonymous Blocks



Exceptions

Examples are

NO_DATA_FOUND, ZERO_DIVIDE, OTHERS

Raise some exception

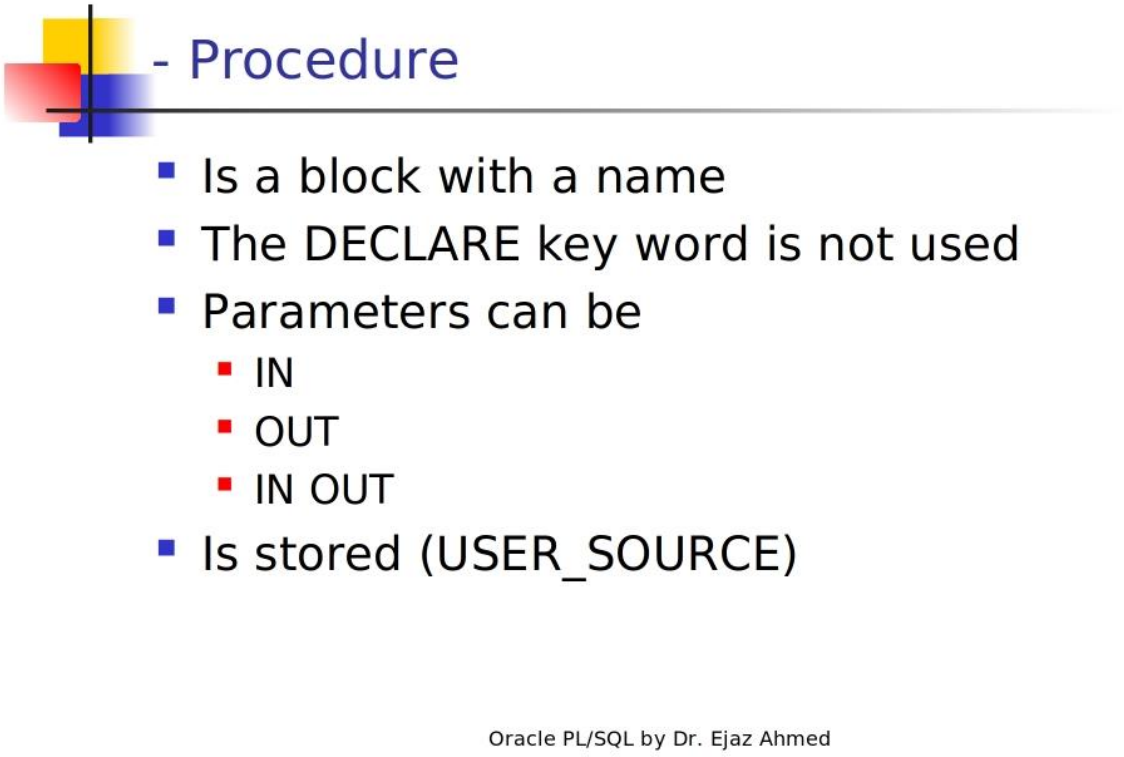
```
WHEN NO_DATA_FOUND THEN
raise_application_error(-20011,'Invalid FK value');
```

To display details of oracle standard error message

- EXCEPTION
  - WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Error detail is: '|| SQLERRM)

Transaction DONE or NOT DONE

```
BEGIN
 DELETE FROM EMP
 WHERE ENAME='ALL';

 UPDATE EMP
 SET SAL=200
 WHERE ENAME='SMITH';
 COMMIT;
EXCEPTION
 WHEN OTHERS THEN
  ROLLBACK;
END;
/
```



1) IN type parameter: These types of parameters are used to send values to stored procedures.

2) OUT type parameter: These types of parameters are used to get values from stored procedures. This is similar to a return type in functions.

3) IN OUT parameter: These types of parameters are used to send values and get values from stored procedures

… -- Creating or Replacing a Procedure

```
SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE  proc_test(p_empno IN VARCHAR2)  IS

     v_job          EMP.job%TYPE;
     v_sal          EMP.sal%TYPE;

BEGIN

      SELECT job, sal
      INTO  v_job,v_sal
      FROM  emp
     WHERE  empno = p_empno;
      DBMS_OUTPUT.PUT_LINE('job is '||v_job);

EXCEPTION

   WHEN OTHERS THEN
   DBMS_OUTPUT.PUT_LINE('ERROR...');

END;
/
SQL> Show errors
SQL> execute proc_test(5893);
```


-- Invoking a Procedure

```
DECLARE
    …
BEGIN
    …
    proc_test('23');
    …
END;
/
```

Or

```
SQL> exec proc_test('1123')
```

Another Example

```
SQL> ed
Wrote file afiedt.buf

  1  create or replace procedure test_proc is
  2       v_id  INTEGER;
  3      v_empno emp.empno%TYPE;
  4  BEGIN
  5      v_id := 1234567;
  6      select empno into v_empno
  7      FROM  EMP
  8      WHERE empno = v_id;
  9  EXCEPTION
 10      WHEN NO_DATA_FOUND THEN
 11       DBMS_OUTPUT.PUT_LINE('No record exists');
 12*   END;
SQL> /

Procedure created.

SQL> exec test_proc
No record exists

PL/SQL procedure successfully completed.

SQL> show errors                (to see errors for procedures, functions)
```


Oracle PL/SQL Trigger

Syntax:

```
CREATE [ OR REPLACE ] TRIGGER <trigger_name>
[ BEFORE | AFTER | INSTEAD OF ]          Trigger Timing
[ INSERT | UPDATE | DELETE.....]         Event
ON <name of underlying object>
[ FOR EACH ROW ]                         Row Level
[ WHEN <condition for trigger to get execute> ]   Conditional Clause
DECLARE
   <Declaration part>
BEGIN
   <Execution part>
EXCEPTION
   <Exception handling part>
END;
```

Example of Trigger

```
CREATE TABLE superheroes (
  sh_name VARCHAR2 (15)
);
```

Example: Before Insert Trigger

```
CREATE OR REPLACE TRIGGER bi_Superheroes
BEFORE INSERT ON superheroes
FOR EACH ROW
ENABLE
DECLARE
  v_user  VARCHAR2 (15);
BEGIN
 SELECT user INTO v_user FROM dual;
 DBMS_OUTPUT.PUT_LINE('You Just Inserted a Row Mr.'|| v_user);
END;
/
```

Output:
You Just Inserted a Row Mr. Scott

```
INSERT INTO superheroes VALUES ('Ironman');
```


Example of Trigger (**INSERT, UPDATE, DELETE**)

```
CREATE OR REPLACE TRIGGER tr_superheroes
AFTER INSERT OR DELETE OR UPDATE ON superheroes
FOR EACH ROW
ENABLE
DECLARE
  v_user  VARCHAR2(15);
BEGIN

  SELECT
    user INTO v_user FROM dual;
  IF INSERTING THEN
    DBMS_OUTPUT.PUT_LINE('one line inserted by '||v_user);
  ELSIF DELETING THEN
    DBMS_OUTPUT.PUT_LINE('one line Deleted by '||v_user);
  ELSIF UPDATING THEN
    DBMS_OUTPUT.PUT_LINE('New Value:  '||:NEW.sh_name|| ' Old value:  '||:OLD.sh_name);
  END IF;
END;
/
```

```
update superheroes set sh_name ='Superman';
```

Lab Work

Q # 1: Update Job_ID against Employee_ID "205". If his salary >1000 then promote him to "Manager" otherwise promote him to the "Analyst" using PLSQL.

Q#2: Write a PL/SQL procedure to update the commission of employees given a specific target otherwise a general commission to be paid. If an employee's monthly sale is more than the target sale then 25% of (monthly_sale - target sale) will be the commission of employee. If an employee's monthly sale is less than the target sale then $50 will be the commission. Procedure prototype is given:

update_commission(monthly_sale NUMBER, target_sale NUMBER, emp_id NUMBER)

Q#3: Write a PL/SQL trigger that display the new as well as old salary of the employee whenever the record is been updated.