PL/SQL

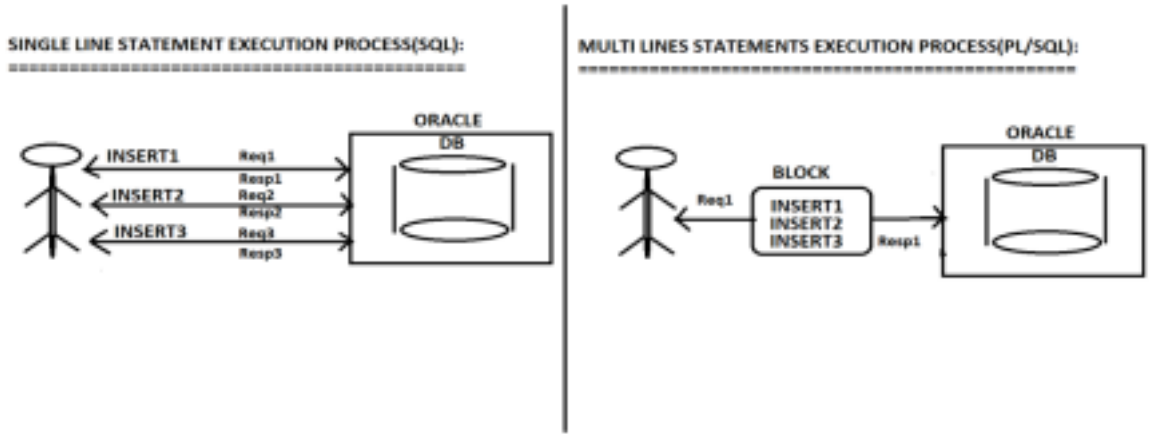
INTRODUCTION TO PL/SQL:

PL/SQL STANDS FOR PROCEDURAL LANGUAGE WHICH IS AN EXTENSION OF SQL.PL/SQL WAS INTRODUCED IN ORACLE 6.0 VERSION.

SQL IS A NON-PROCEDURAL LANGUAGE WHEREAS PL/SQL IS A PROCEDURAL LANGUAGE.

SQL SUPPORTS A SINGLE LINE STATEMENT (QUERY) EXECUTION PROCESS WHEREAS PL/SQL SUPPORTS MULTI LINES STATEMENTS(PROGRAM) EXECUTION PROCESS.

IN SQL EVERY QUERY STATEMENT IS COMPILING AND EXECUTING INDIVIDUALLY.SO THAT NO. OF COMPILATIONS ARE INCRESED AND REDUCE PERFORMANCE OF DATABASE.



IN PL/SQL ALL SQL QUERIES ARE GROUPED INTO A SINGLE BLOCK AND WHICH WILL COMPILE AND EXECUTE ONLY ONE TIME.SO THAT IT WILL REDUCE NO. OF COMPILATIONS AND IMPROVE PERFORMANCE OF DATABASE.

FEATURES OF PL/SQL:

1. TO IMPROVES PERFORMANCE.

2. SUPPORTING CONDITIONAL & LOOPING STATEMENTS. 3.

SUPPORTING REUSABILITY.

4. PROVIDING SECURITY BECAUSE ALL PROGRAMS ARE SAVED IN

DATABASE AND AUTHORIZED USER CAN ONLY ACCESS THE

PROGRAMS.

5. SUPPORTING PORTABILITY I.E PL/SQL PROGRAMS CAN BE MOVED FROM ONE

PLATFORM TO ANOTHER PLATFORM WITHOUT ANY CHANGES. 6.

SUPPORTING EXCEPTION HANDLING MECHANISM.

7. SUPPORTING MODULAR PROGRAMMING I.E IN A PL/SQL A BIG PROGRAM CAN BE DIVIDED INTO SMALL MODULES WHICH ARE CALLED AS STORED PROCEDURE AND

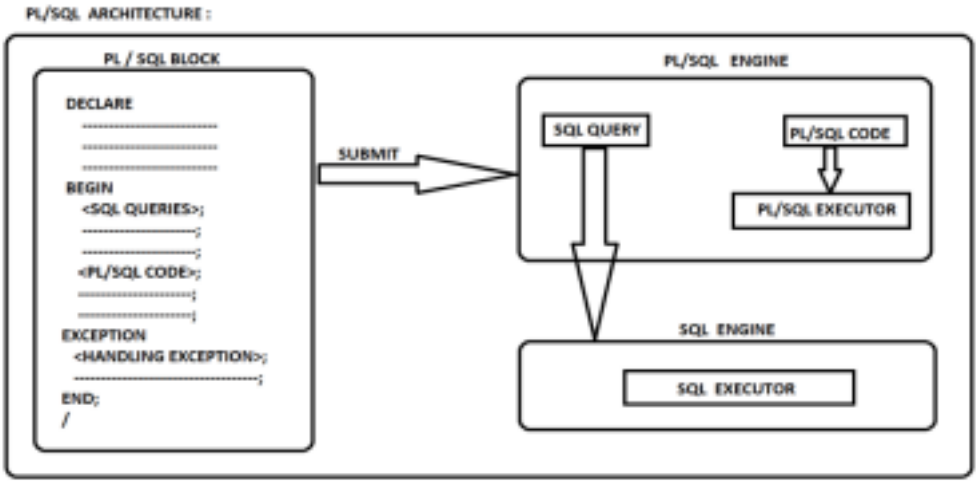
STORED FUNCTIONS.

PL/SQL ARCHITECTURE:

PL/SQL IS BLOCK STRUCTURE PROGRAMMING LANGUAGE.WHICH IS HAVING THE FOLLOWING TWO ENGINES THOSE ARE

1. SQL ENGINE

2. PL/SQL ENGINE



WHENEVER WE ARE SUBMITING A PL/SQL BLOCK INTO ORACLE SERVER THEN ALL SQL STATEMENTS(QUERIES) ARE SEPERATED AND EXECUTING BY SQLQUERY EXECUTOR WITH IN SQL ENGINE.WHERE AS ALL PL/SQL STATEMENTS(CODE) ARE SEPERATED AND EXECUTING BY PL/SQL CODE EXECUTOR WITH IN PL/SQL ENGINE.

WHAT IS BLOCK:

A BLOCK IS A SET OF STATEMENTS WHICH ARE COMPILE & EXECUTED BY ORACLE AS A SINGLE UNIT. PL/SQL SUPPORTING THE FOLLOWING TWO TYPES OF BLOCKS THOSE ARE,

1. ANONYMOUS BLOCK

2. SUB BLOCK

DIFF. B/W ANONYMOUS & SUB BLOCK:

ANONYMOUS BLOCK SUB BLOCK

1. UNNAMED BLOCK 1. NAMED BLOCK

2. THIS BLOCK CODE IS NOT 2. THIS BLOCK CODE IS SAVED IN DB. SAVED IN DB AUTOMATICALLY. 3. IT CANNOT REUSABLE. 3. IT CAN BE REUSABLE. 4. EVERY TIME COMPILATION 4. PRE - COMPILED CODE OF CODE. (FIRST TIME COMPILATION ONLY) 5. ARE USING IN "DB TESTING".5. ARE USING IN APPLICATION DEVELOPMENT LIKE "JAVA", ".NET" & "DB APPLICATIONS ".

ANONYMOUS BLOCKS:

THESE ARE UNNAMED BLOCKS IN PL/SQL.WHICH CONTAINS THREE MORE BLOCKS THOSE ARE,

I) DECLARATION BLOCK

II) EXECUTION BLOCK

III) EXCEPTION BLOCK

I) DECLARATION BLOCK:

> THIS BLOCK STARTS WITH " DECLARE " STATEMENT.

> DECLARING VARIABLES, CURSORS, USER DEFINE EXCEPTIONS.

> IT IS OPTIONAL BLOCK.

II) EXECUTION BLOCK:

> THIS BLOCK STARTS WITH " BEGIN " STATEMENT & ENDS WITH "END" STATEMENT.

> IMPLEMENTING SQL STATEMENTS(SQL) & LOGICAL CODE OF A PROGRAM (PL/SQL).

> IT IS MANDATORY BLOCK.

III) EXCEPTION BLOCK:

> THIS BLOCK STARTS WITH "EXCEPTION" STATEMENT. >

HANDLING EXCEPTIONS.

> IT IS An OPTIONAL BLOCK.

STRUCTURE OF PL/SQL BLOCK:

```
DECLARE
< VARIABLES, CURSOR, UD EXCEPTIONS>; BEGIN
< WRITING SQL STATEMENTS>;
< PL/SQL LOGICAL CODE>;
EXCEPTION
< HANDLING EXCEPTIONS>;
END;
/
```

VARIABLES IN PL/SQL:

STEP1: DECLARING VARIABLES:

SYNTAX:

```
DECLARE
<VARIABLE NAME> <DT>[SIZE];
```

EX:

```
DECLARE
A NUMBER (10) (OR) A INT;
B VARCHAR2(10);
```

STEP2: ASSIGNING / STORING A VALUE INTO VARIABLE: **SYNTAX:**

```
<VARIABLE NAME>: = <VALUE>;
```

EX:

```
A: = 1021;
B: = 'SAI';
```

HERE,

: = - ASSIGNMENT OPERATOR IN PL/SQL

= - COMPARISION OPERATOS IN PL/SQL

STEP3: PRINTING VARIABLES VALUES:

SYNTAX:

```
DBMS_OUTPUT.PUT_LINE (<VARIABLE NAME > (OR) '<UD
MESSAGE>');
```

EX:

```
DBMS_OUTPUT.PUT_LINE(A);

DBMS_OUTPUT.PUT_LINE(B);

DBMS_OUTPUT.PUT_LINE ('WELCOME TO PL/SQL');
```

EX1:

TO PRINT "WELCOME TO PL/SQL" STATEMENT.

SOL:

```
SQL> BEGIN
 DBMS_OUTPUT.PUT_LINE ('WELCOME TO PL/SQL');  END;
 /
```

PL/SQL PROCEDURE SUCCESSFULLY COMPLETED.

NOTE:

THE ABOVE PROGRAM WILL NOT DISPLAY THE OUTPUT OF A PL/SQL PROGRAM.IF ORACLE SERVER WANT TO DISPLAY OUTPUT OF A PL/SQL PROGRAM THEN WE USE THE FOLLOWING SYNTAX,

SYNTAX:

```
SET SERVEROUTPUT OFF / ON;
```

HERE,

OFF: IT IS DEFAULT.OUTPUT IS NOT DISPLAY

ON: OUTPUT IS DISPLAY

```
SQL> SET SERVEROUTPUT ON;
SQL> /
WELCOME TO PL/SQL
```

EX2:

TO PRINT VARIABLES VALUES?

SOL:

```
SQL> DECLARE
 X NUMBER (10);
 Y NUMBER (10);
 BEGIN
 X: =100;
 Y: =200;
  DBMS_OUTPUT.PUT_LINE ('VARIABLES VALUES ARE:'||X||','||Y);  END;
 /
VARIABLES VALUES ARE:100,200
```

EX3:

TO PRINT SUM OF TWO NUMBERS AT RUNTIME?

SOL:

```
DECLARE
        X NUMBER (2);
        Y NUMBER (2);
        Z NUMBER (10);
BEGIN
        X: =&X;
        Y: =&Y;
```

```
        Z: =X+Y;
        DBMS_OUTPUT.PUT_LINE(Z);
END;
/
```

OUTPUT:

```
ENTER VALUE FOR X: 10
OLD 6: X: =&X;
NEW 6: X: =10;
ENTER VALUE FOR Y: 20
OLD 7: Y: =&Y;
NEW 7: Y: =20;
30
```

VERIFY:

ON = DISPLAY OLD, NEW BIND VARIABELE STATEMENTS

OFF = DOESNOT DISPLAY OLD, NEW BIND VARIABLES STATEEMTNS

SYNTAX:

SET VERIFY ON / OFF

EX:

```
SQL> SET VERIFY OFF;
SQL> /
ENTER VALUE FOR X: 10
ENTER VALUE FOR Y: 20
30
```

SELECT..... INTO STATEMENT:

STORING A TABLE COLUMNS VALUES INTO VARIABELS.

RETURNS A SINGLE ROW (OR) A SINGLE VALUE.CAN USE IN EXECUTION BLOCK.

SYNTAX:

SELECT <COLUMN NAME1>, <COLUMN NAME2>, ...INTO <VARIABLE NAME1>, <VARIABLE NAME2>……. FROM <TN> [WHERE <CONDITION>];

EX1:

WA PL/SQL PRG. TO DISPLAY ENAME, SALARY DETAILS FROM EMP TABLE AS PER THE GIVEN EMPNO BY USING SELECT .....INTO STATEMENT?

SOL:

```
DECLARE
V_ENAME VARCHAR2(10);
V_SAL NUMBER (10);
BEGIN
SELECT ENAME, SAL INTO V_ENAME, V_SAL FROM EMP WHERE EMPNO=&EMPNO;
DBMS_OUTPUT.PUT_LINE(V_ENAME||','||V_SAL);
END;
/
```

OUTPUT:

```
ENTER VALUE FOR EMPNO: 7788
SCOTT,3000
```

EX:

WA PL/SQL PRG. TO FETCH MAX.SALARY OF EMP TABLE BY USING "SELECT INTO" STATEMENT?

SOL:

```
DECLARE
V_MAXSAL NUMBER (10);
BEGIN
SELECT MAX(SAL) INTO V_MAXSAL FROM EMP;
```

```
DBMS_OUTPUT.PUT_LINE(V_MAXSAL);

END;

/
```

OUTPUT:

5000

VARIABLES ATTRIBUTES (OR) ANCHOR NOTATIONS:

VARIABLES ATTRIBUTES ARE USED IN PLACE OF DATATYPES AT VARIABLE DECLARATION.

WHENEVER WE ARE USING VARIABLES ATTRIBUTES INTERNALLY ORACLE SERVER IS ALLOCATE SOME MEMORY FOR THESE VARIABLES ATTRIBUTES FOR STORING THE CORRESPONDING VARIABLE COLUMN DATATYPE WHICH WAS ASSIGNED AT THE TIME OF TABLE CREATION.

VARIABLES ATTRIBUTES ARE ALSO CALLED AS "ANCHOR NOTATIONS".

THE ADVANTAGE OF VARIABLES ATTRIBUTES ARE WHENEVER WE WANT TO CHANGE

A PARTICULAR COLUMN DATATYPE IN A TABLE THEN THE CORRESPONDING COLUMN VARIABLE DATATYPE ALSO CHANGED IN VARIABLE ATTRIBUTE MEMORY AUTOMATICALLY.

PL/SQL SUPPORTS THE FOLLOWING TWO TYPE VARIABLES ATTRIBUTES ARE,

1. **COLUMN LEVEL ATTRIBUTES**
2. **ROW LEVEL ATTRIBUTES**

1. COLUMN LEVEL ATTRIBUTES:

IN THIS LEVEL WE ARE DEFINING VARIABLES ATTRIBUTES FOR INDIVIDUAL COLUMNS.IT IS REPRESENTING WITH "%TYPE" STATEMENT.

SYNTAX:

<VARIABLE NAME> <TN>. <COLUMN NAME>%TYPE;

EX:

V_ENAME EMP.ENAME%TYPE;

V_SAL EMP.SAL%TYPE;

PROGRAM1:

```
DECLARE
V_ENAME EMP.ENAME%TYPE;
V_SAL EMP.SAL%TYPE;
BEGIN
SELECT ENAME, SAL INTO V_ENAME, V_SAL FROM EMP WHERE EMPNO=&EMPNO;
DBMS_OUTPUT.PUT_LINE(V_ENAME||','||V_SAL); END;
/
```

OUTPUT:

```
ENTER VALUE FOR EMPNO: 7788
SCOTT,3000
```

2. ROW LEVEL ATTRIBUTES:

IN THIS LEVEL WE ARE DECLARING A SINGLE VARIABLE WILL REPRESENT ALL DIFFERENT DATATYPES OF COLUMNS IN A TABLE.IT REPRESENT WITH "%ROWTYPE ".

SYNTAX:

<VARIABLE NAME> <TABLE NAME>%ROWTYPE; EX: I

EMP%ROWTYPE;

PROGRAM2:

```
DECLARE
I EMP%ROWTYPE;
BEGIN
```

```
SELECT ENAME, SAL INTO I. ENAME, I.SAL FROM EMP WHERE
EMPNO=&EMPNO;
DBMS_OUTPUT.PUT_LINE (I. ENAME||','||I.SAL); END;
/
```

(OR)

```
DECLARE
I EMP%ROWTYPE;
BEGIN
SELECT * INTO I FROM EMP WHERE EMPNO=&EMPNO;
DBMS_OUTPUT.PUT_LINE (I. ENAME||','||I.SAL||','||I.DEPTNO); END;
/
```