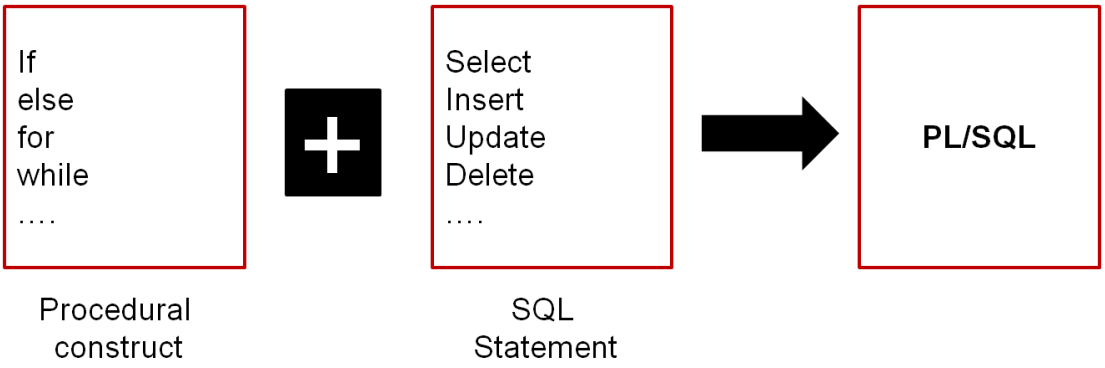
What is PL/SQL?



- PL/SQL is a collection of programming construct and SQL statement.
- PL stands for procedural language.

Disadvantage of SQL

- At a time only one query is executed.
- Query are not saved in database, we need to rewrite query again and again.
- Programming construct are not available i.e. loop, branching statement and conditional statement.
- Reduce network performance.

Advantages of PL/SQL

- We can execute multiple queries in the form of programs.
- Reusability
- Programming constructs are available.
- Improve network performance.

Procedure

- It is a named PL/SQL block which is used to solve some particular task.
- Permanently stored in database, that's why these procedures are also called as stored procedure.

Types of procedure

- Static procedure

    - Procedure created without argument
- Dynamic procedure
    - Procedure created with argument

Note

- Every procedure has two parts
    - Procedure Specification
        - In procedure specification we specify name of the procedure and type of the parameters.
    - Procedure body
        - In procedure body we solve actual task

Example#1

Create Procedure

```
DELIMITER //
CREATE PROCEDURE welcome()
BEGIN
        SELECT "welcome to PLSQL" as message;
END //
DELIMITER ;
```

Call Procedure

```
CALL welcome();
```

Example#2

Write a PL SQL procedure to take a number and display cube of it.

```
DELIMITER //
CREATE PROCEDURE display_cube (IN x INT)
BEGIN
   DECLARE x_cube INT;
   SET x_cube = x * x * x;
   SELECT CONCAT('The cube of ', x, ' is: ', x_cube) AS result;
END //
```

```
DELIMITER ;

CALL display_cube(3);
```

Example#3

Write a procedure to read age and display eligibility for voting.

```
DELIMITER //

CREATE PROCEDURE check_eligibility(age INT)
BEGIN
   DECLARE message VARCHAR(100);
   IF age >= 18 THEN
      SET message = 'You are eligible for voting';
   ELSE
      SET message = 'You are not eligible for voting';
   END IF;

   SELECT message AS result;
END //
DELIMITER ;

CALL check_eligibility(20);
```

Example#4

Write a procedure to read number and display it is even or odd.

```
DELIMITER //

CREATE PROCEDURE check_even_odd (IN p_number INT)
BEGIN
   IF  MOD(p_number, 2) = 0 THEN
      SELECT CONCAT(p_number, ' is even') AS result;
   ELSE
      SELECT CONCAT(p_number, ' is odd') AS result;
```

```
   END IF;
END //

DELIMITER ;

CALL check_even_odd(2);
```

Example#5

Write a procedure to read 2 numbers and display the greater number.

```
DELIMITER //
CREATE PROCEDURE check_greater_number(IN p_number1 INT, IN p_number2 INT)
BEGIN
   IF p_number1 > p_number2 THEN
      SELECT p_number1 AS greater_number;
   ELSE
      SELECT p_number2 AS greater_number;
   END IF;
END //

DELIMITER ;

CALL check_greater_number(30,20);
```

Example#6

Write a PL/SQL stored procedure for passing employee_id as a parameter, display employee details.

```
DELIMITER //

CREATE PROCEDURE get_employee(IN p_employee_id INT)
BEGIN
   SELECT * FROM employees WHERE employee_id = p_employee_id;
END //
```

```
DELIMITER ;
```

Example#7

Write a PL/SQL stored procedure for passing employee_id as a parameter, display employee and department details.

```
DELIMITER //

CREATE PROCEDURE get_employee(IN p_employee_id INT)
BEGIN
   SELECT emp.*,dept.department_name FROM employees emp, departments dept
   WHERE emp.employee_id = p_employee_id
   and emp.department_id = dept.department_id;
END //

DELIMITER ;
```

Example#8

Write a PL/SQL procedure to read employee_id as input and return employee_name as output parameter.

```
DELIMITER //
CREATE PROCEDURE get_employee (
   IN p_employee_id INT,
   OUT p_employee_name VARCHAR(100)
)
BEGIN
   SELECT employee_name INTO p_employee_name FROM employees
   WHERE employee_id = p_employee_id;
END //

CALL get_employee(101, @employee_name);
```

```
SELECT @employee_name;
```

Example#9

Write a PL/SQL procedure to read employee_id as input and return employee_name and salary as output parameter.

```
DELIMITER //
CREATE PROCEDURE get_employee (
   IN p_employee_id INT,
   OUT p_employee_name VARCHAR(100),
   OUT p_salary INT
)
BEGIN
   SELECT employee_name, salary INTO p_employee_name, p_salary  FROM employees
   WHERE employee_id = p_employee_id;
END //

CALL get_employee(101, @employee_name, @salary);

SELECT @employee_name,  @salary;
```