

# PL/SQL (Procedural Language extension to SQL)

## PL/SQL 이란?

SQL만으로는 구현이 어렵거나 구현 불가능한 작업을 수행하기 위해 오라클에서 제공하는 프로그래밍 언어이다.

변수, 조건, 반복 처리 등 다양한 기능을 사용할 수 있다.

## 블록(Block) 이란?

PL/SQL은 데이터베이스 관련 특정 작업을 수행하는 명령어와 실행에 필요한 여러 요소를 정의하는 명령어 등으로 구성된다. 이러한 명령어들을 모아 둔 PL/SQL 프로그램의 기본 단위를 블록이라고 한다.

### [형식]

#### DECLARE

- 선언부 - 변수, 상수, 커서 등을 선언
- 생략 가능

#### BEGIN

실행하는 명령어

#### EXCEPTION

- 예외처리 - 생략 가능

#### END;

- DECLARE, BEGIN, EXCEPTION 에는 ;(세미콜론)을 사용하지 않는다.
- 주석으로는 --, /\* \*/ 를 사용한다.

### [실습]

#### ① CMD

SQL> SET SERVEROUTPUT ON; -- PL/SQL 결과를 화면에 출력

SQL> BEGIN

2 DBMS\_OUTPUT.PUT\_LINE('Hello PL/SQL!!');

3 END;

4 / -- CMD 창에서 실행할 때

Hello PL/SQL!!

PL/SQL 처리가 정상적으로 완료되었습니다.

#### ② SQLDEVELOPER

SET SERVEROUTPUT ON;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Hello PL/SQL!!');

END;

## 변수

- 데이터를 일시적으로 저장하는 요소이다.
- 선언부에 작성한다.
- 선언한 변수에 값을 할당하기 위해서는 **:=**를 사용한다.

### [형식]

변수명 자료형 := 값 또는 표현식

## 변수명

- 같은 블록 안에서는 중복될 수 없다.
- 대소문자를 구별하지 않는다.
- 이름은 문자로 시작해야 한다. (한글도 가능)
- 30 byte (영어는 30자, 한글은 15자)
- 영문자, 한글, 숫자, 특수문자 (\$, #, \_) 사용할 수 있다.

## 변수의 자료형

- 변수에 저장할 데이터가 어떤 종류인지를 정하기 위해 사용된다
- 스칼라(scalar), 복합(composite), 참조(reference), LOB(Large Object)로 구분된다.

### 1. 스칼라형

- 숫자, 문자열, 날짜 등과 같이 오라클에서 기본으로 정의해 놓은 자료형

NUMBER : 소수점을 포함할 수 있는 최대 38자리 숫자 데이터

CHAR / VARCHAR2

DATE

BOOLEAN (PL/SQL에서만 사용할 수 있는 논리 자료형, true, false, NULL을 포함)

### 2. 참조형

- 오라클 데이터베이스에 존재하는 특정 테이블 열의 자료형이나 하나의 행 구조를 참조하는 자료형이다.
- **열을 참조할 때는 %TYPE, 행을 참조할 때는 %ROWTYPE**을 사용한다

### [형식]

변수명 테이블명.열이름**%TYPE**;

변수명 테이블명**%ROWTYPE**;

### 3. 복합형

- 여러 종류 및 개수의 데이터를 저장하기 위해 사용자가 직접 정의하는 자료형이다.
- 컬렉션, 레코드로 구분된다.

### 4. LOB형

- 대용량의 텍스트, 이미지, 동영상, 사운드 데이터 등 대용량 데이터를 저장하기 위한 자료형이다.
- BLOB, CLOB 등이 있다.

## [실습]

```
SQL> DECLARE
  2     NAME VARCHAR2(10);
  3     AGE NUMBER(4) := 25;
  4 BEGIN
  5     NAME := '홍길동';
  6     DBMS_OUTPUT.PUT_LINE('NAME : ' || NAME);
  7     DBMS_OUTPUT.PUT_LINE('AGE : ' || AGE);
  8 END;
  9 /
```

NAME : 홍길동

AGE : 7788

PL/SQL 처리가 정상적으로 완료되었습니다.

## [실습]

```
SQL> DECLARE
  2     V_NO DEPARTMENTS.DEPARTMENT_ID%TYPE := 50;
  3 BEGIN
  4     DBMS_OUTPUT.PUT_LINE('V_NO : ' || V_NO);
  5 END;
  6 /
```

V\_NO : 50

PL/SQL 처리가 정상적으로 완료되었습니다.

\* V\_NO 변수는 DEPARTMENT\_ID와 같은 자료형 및 크기가 지정된다

## [실습]

```
SQL> DECLARE
  2     V_DEPT DEPARTMENTS%ROWTYPE;
      -- V_DEPT 변수를 DEPARTMENTS 테이블의 행 구조로 선언
  3 BEGIN
  4     SELECT DEPARTMENT_ID, DEPARTMENT_NAME, MANAGER_ID, LOCATION_ID INTO
V_DEPT FROM DEPARTMENTS WHERE DEPARTMENT_ID = 50;
  5     -- INTO V_DEPT를 사용하여 SELECT 한 결과를 V_DEPT에 대입한다.
  6     -- V_DEPT가 가지고 있는 필드 개수, 자료형은 SELECT한 결과의 개수와 자료형이 같아야 한다.
  7
  8     DBMS_OUTPUT.PUT_LINE('DEPARTMENT_ID : ' || V_DEPT.DEPARTMENT_ID);
  9     DBMS_OUTPUT.PUT_LINE('DEPARTMENT_NAME : ' || V_DEPT.DEPARTMENT_NAME);
 10     DBMS_OUTPUT.PUT_LINE('MANAGER_ID : ' || V_DEPT.MANAGER_ID);
 11     DBMS_OUTPUT.PUT_LINE('LOCATION_ID : ' || V_DEPT.LOCATION_ID);
 12 END;
 13 /
```

DEPARTMENT\_ID : 50  
DEPARTMENT\_NAME : Shipping  
MANAGER\_ID : 121  
LOCATION\_ID : 1500

PL/SQL 처리가 정상적으로 완료되었습니다.

## 상수

- 한번 저장한 값이 프로그램이 종료될 때까지 변하지를 않는다

### [형식]

변수명 **CONSTANT** 자료형 := 값 또는 표현식

## 변수의 기본값 지정

### [형식]

변수명 자료형 **DEFAULT** 값 또는 표현식

## 변수에 NULL 값 막기

### [형식]

변수명 자료형 **NOT NULL** := 값 또는 표현식

### [실습]

```
SQL> DECLARE
  2     PI CONSTANT NUMBER := 3.141592;
  3     APPLE VARCHAR2(10) DEFAULT '사과';
  4     ID VARCHAR2(10) NOT NULL := 'hong';
  5     PWD VARCHAR2(10) NOT NULL DEFAULT '1234';
  6 BEGIN
  7     DBMS_OUTPUT.PUT_LINE('PI : ' || PI);
  8     DBMS_OUTPUT.PUT_LINE('APPLE : ' || APPLE);
  9     DBMS_OUTPUT.PUT_LINE('ID : ' || ID);
 10     DBMS_OUTPUT.PUT_LINE('PWD : ' || PWD);
 11 END;
 12 /
PI : 3.141592
APPLE : 사과
ID : hong
PWD : 1234
```

PL/SQL 처리가 정상적으로 완료되었습니다.

# 조건 제어문

## 1. IF문

### [형식]

- ① IF - THEN
- ② IF - THEN - ELSE
- ③ IF - THEN - ELSIF

## 2. CASE문

- ① 단순 CASE문  
: 비교 기준이 되는 변수 또는 식을 명시한다
- ② 검색 CASE문  
: 비교기준을 명시하지 않고 각각의 WHEN 절에서 조건식을 명시한다  
: IF-THEN-ELSIF와 큰 차이가 없다.

### [형식]

```
CASE 비교기준
  WHEN 값1 THEN
    명령어;
  WHEN 값2 THEN
    명령어;
  ELSE
    명령어;
END CASE;
```

### [형식]

```
CASE
  WHEN 조건1 THEN
    명령어;
  WHEN 조건2 THEN
    명령어;
  ELSE
    명령어;
END CASE;
```

오라클 SQL문에 사용하는 CASE 문은 조건에 따라 특정 결과값을 반환하는 것이면,  
PL/SQL 문의 CASE 조건문은 조건에 따라 수행할 작업을 지정할 수 있다는 차이가 있다.

SQL문의 CASE는 END로 종료되지만, PL/SQL 문의 CASE 조건문은 END CASE로 종료된다.

## [실습]

```
SQL> DECLARE
  2     NUM NUMBER := 15;
  3 BEGIN
  4     IF MOD(NUM, 2) = 1 THEN
  5         DBMS_OUTPUT.PUT_LINE(NUM || '는 홀수이다');
  6     END IF;
  7 END;
  8 /
15는 홀수이다
```

PL/SQL 처리가 정상적으로 완료되었습니다.

## [실습]

```
SQL> DECLARE
  2     NUM NUMBER := 16;
  3 BEGIN
  4     IF MOD(NUM, 2) = 0 THEN
  5         DBMS_OUTPUT.PUT_LINE(NUM || '는 짝수이다');
  6     ELSE
  7         DBMS_OUTPUT.PUT_LINE(NUM || '는 홀수이다');
  8     END IF;
  9 END;
10 /
16는 짝수이다
```

PL/SQL 처리가 정상적으로 완료되었습니다.

## [실습]

```
SQL> DECLARE
  2     SCORE NUMBER := 87;
  3 BEGIN
  4     IF SCORE >= 90 THEN
  5         DBMS_OUTPUT.PUT_LINE('A 학점');
  6     ELSIF SCORE >= 80 THEN
  7         DBMS_OUTPUT.PUT_LINE('B 학점');
  8     ELSIF SCORE >= 70 THEN
  9         DBMS_OUTPUT.PUT_LINE('C 학점');
10     ELSIF SCORE >= 60 THEN
11         DBMS_OUTPUT.PUT_LINE('D 학점');
12     ELSE
13         DBMS_OUTPUT.PUT_LINE('F 학점');
14     END IF;
15 END;
16 /
```

B 학점

PL/SQL 처리가 정상적으로 완료되었습니다.

### [실습]

```
SQL> DECLARE
  2     SCORE NUMBER := 87;
  3 BEGIN
  4     CASE TRUNC(SCORE/10)
  5         WHEN 9 THEN DBMS_OUTPUT.PUT_LINE('A 학점');
  6         WHEN 8 THEN DBMS_OUTPUT.PUT_LINE('B 학점');
  7         WHEN 7 THEN DBMS_OUTPUT.PUT_LINE('C 학점');
  8         WHEN 6 THEN DBMS_OUTPUT.PUT_LINE('D 학점');
  9         ELSE DBMS_OUTPUT.PUT_LINE('F 학점');
 10     END CASE;
 11 END;
 12 /
```

B 학점

PL/SQL 처리가 정상적으로 완료되었습니다.

### [실습]

```
SQL> DECLARE
  2     SCORE NUMBER := 87;
  3 BEGIN
  4     CASE
  5         WHEN SCORE >= 90 THEN DBMS_OUTPUT.PUT_LINE('A 학점');
  6         WHEN SCORE >= 80 THEN DBMS_OUTPUT.PUT_LINE('B 학점');
  7         WHEN SCORE >= 70 THEN DBMS_OUTPUT.PUT_LINE('C 학점');
  8         WHEN SCORE >= 60 THEN DBMS_OUTPUT.PUT_LINE('D 학점');
  9         ELSE DBMS_OUTPUT.PUT_LINE('F 학점');
 10     END CASE;
 11 END;
 12 /
```

B 학점

PL/SQL 처리가 정상적으로 완료되었습니다.

## 반복 제어문

### [형식]

#### ① 기본 LOOP

### [형식]

LOOP

    명령어;

END LOOP;

#### ② WHILE LOOP

- 조건이 참이면 반복하고, 거짓이면 반복문을 벗어난다.

### [형식]

WHILE 조건 LOOP

    명령어;

END LOOP;

#### ③ FOR LOOP

- 시작값부터 종료값까지 1씩 증가한다.

### [형식]

FOR i IN 시작값 .. 종료값 LOOP

    명령어;

END LOOP;

FOR i IN REVERSE 시작값 .. 종료값 LOOP -- 역순 (0 .. 4 라고 쓰는 것은 변하지 않는다)

    명령어;

END LOOP;

#### ④ Cusor FOR LOOP

반복문을 중단 / 특정 반복 주기를 건너뛰는 명령어

① **EXIT** : 수행 중인 반복 종료

② **EXIT WHEN** : 반복 종료를 위한 조건을 지정

③ **CONTINUE** : 수행 중인 반복의 현재 주기를 건너뛰다

④ **CONTINUE WHEN** : 조건식을 지정하고 조건식을 만족하면 현재 반복 주기를 건너뛰다



## [실습] 기본 LOOP

```
SQL> DECLARE
  2     NUM NUMBER := 0;
  3 BEGIN
  4     LOOP
  5         DBMS_OUTPUT.PUT_LINE(NUM);
  6         NUM := NUM + 1;
  7         EXIT WHEN NUM > 4;
  8     END LOOP;
  9 END;
10 /
0
1
2
3
4
```

PL/SQL 처리가 정상적으로 완료되었습니다.

## [실습] WHILE LOOP

```
SQL> DECLARE
  2     NUM NUMBER := 0;
  3 BEGIN
  4     WHILE NUM < 4 LOOP
  5         DBMS_OUTPUT.PUT_LINE(NUM);
  6         NUM := NUM + 1;
  7         EXIT WHEN NUM > 4;
  8     END LOOP;
  9 END;
10 /
0
1
2
3
```

PL/SQL 처리가 정상적으로 완료되었습니다.

## [실습] FOR LOOP

```
SQL> BEGIN
  2     FOR i IN 0..4 LOOP -- i는 선언부에서 정의하는 것이 아니라 FOR LOOP에서 정의한다.
  3         DBMS_OUTPUT.PUT_LINE(i);
  4     END LOOP;
  5 END;
  6 /
0
```

1  
2  
3  
4

PL/SQL 처리가 정상적으로 완료되었습니다.

## [실습] FOR LOOP

- 역순으로 출력할 때는 **REVERSE**를 사용한다.
- 시작..끝의 위치는 변하지 않는다.

```
SQL> BEGIN
  2   FOR i IN REVERSE 0..4 LOOP
  3       DBMS_OUTPUT.PUT_LINE(i);
  4   END LOOP;
  5 END;
  6 /
4
3
2
1
0
```

PL/SQL 처리가 정상적으로 완료되었습니다.

## [실습] CONTINUE WHEN

```
SQL> BEGIN
  2   FOR i IN 0..4 LOOP
  3       CONTINUE WHEN MOD(i, 2) = 1;
  4       DBMS_OUTPUT.PUT_LINE(i);
  5   END LOOP;
  6 END;
  7 /
0
2
4
```

PL/SQL 처리가 정상적으로 완료되었습니다.

## [문제]

1~10까지의 숫자 중에서 홀수만 출력하고, 홀수의 합을 구하시오