



**PL/SQL**

## ▶ PL/SQL

Procedural Language extension to SQL의 약자로 오라클 자체에 내장되어 있는 절차적 언어  
SQL의 단점을 보완하여 SQL문장 내에서 변수의 정의, 조건처리, 반복처리 등 지원

### ✓ 구조

구조	설명
DECLARE SECTION (선언부)	<b>DECLARE</b> 로 시작 변수나 상수를 선언하는 부분
EXECUTABLE SECTION (실행부)	<b>BEGIN</b> 으로 시작 제어문, 반복문, 함수 정의 등 로직 기술
EXCEPTION SECTION (예외처리부)	<b>EXCEPTION</b> 으로 시작 예외사항 발생 시 해결하기 위한 문장 기술

# ▶ PL/SQL

## ✓ 예시

**SET SERVEROUTPUT ON;**

\* 프로시저를 사용하여 출력하는 내용을 화면에 보여주도록 설정하는 환경변수로 기본 값은 OFF여서 ON으로 변경

**BEGIN**

DBMS\_OUTPUT.PUT\_LINE('HELLO WORLD');

**END;**

/

\* PUT\_LINE이라는 프로시저를 이용하여 출력(DBMS\_OTUPUT패키지에 속해있음)

PL/SQL 프로시저가 성공적으로 완료되었습니다.

HELLO WORLD

## ▶ PL/SQL

- ✓ 타입 변수 선언 : 변수의 선언과 초기화, 변수 값 출력

**DECLARE**

EMP\_ID NUMBER;

EMP\_NAME VARCHAR2(30);

**BEGIN**

EMP\_ID := 888;

EMP\_NAME := '배장남';

DBMS\_OUTPUT.PUT\_LINE('EMP\_ID : ' || EMP\_ID);

DBMS\_OUTPUT.PUT\_LINE('EMP\_NAME : ' || EMP\_NAME);

**END;**

/

PL/SQL 프로시저가 성공적으로 완료되었습니다.

EMP\_ID : 888

EMP\_NAME : 배장남

## ▶ PL/SQL

- ✓ 타입 변수 선언 : 레퍼런스 변수의 선언과 초기화, 변수 값 출력

**DECLARE**

```
EMP_ID EMPLOYEE.EMP_ID%TYPE;  
EMP_NAME EMPLOYEE.EMP_NAME%TYPE;
```

**BEGIN**

```
SELECT EMP_ID, EMP_NAME  
INTO EMP_ID, EMP_NAME  
FROM EMPLOYEE  
WHERE EMP_ID = '&EMP_ID';
```

```
DBMS_OUTPUT.PUT_LINE('EMP_ID : ' || EMP_ID);  
DBMS_OUTPUT.PUT_LINE('EMP_NAME : ' || EMP_NAME);
```

**END;**

/

대체 변수 입력

EMP\_ID에 대한 값 입력::

확인 취소

PL/SQL 프로시저가 성공적으로 완료되었습니다.

EMP\_ID : 214  
EMP\_NAME : 방명수

## ▶ PL/SQL

- ✓ 타입 변수 선언 : 한 행에 대한 ROWTYPE 변수의 선언과 초기화, 값 출력

**DECLARE**

E EMPLOYEE%ROWTYPE;

**BEGIN**

**SELECT \* INTO E**

**FROM** EMPLOYEE

**WHERE** EMP\_ID = '&EMP\_ID';

DBMS\_OUTPUT.PUT\_LINE('EMP\_ID : ' || E.EMP\_ID);

DBMS\_OUTPUT.PUT\_LINE('EMP\_NAME : ' || E.EMP\_NAME);

DBMS\_OUTPUT.PUT\_LINE('EMP\_NO : ' || E.EMP\_NO);

DBMS\_OUTPUT.PUT\_LINE('SALARY : ' || E.SALARY);

**END;**

/

PL/SQL 프로시저가 성공적으로 완료되었습니다.

EMP\_ID : 200  
EMP\_NAME : 선동일  
EMP\_NAME : 621235-1985634  
EMP\_NAME : 8000000

# ▶ PL/SQL

## ✓ 선택문 : IF ~ THEN ~ END IF

**DECLARE**

```
EMP_ID EMPLOYEE.EMP_ID%TYPE;  
EMP_NAME EMPLOYEE.EMP_NAME%TYPE;  
SALARY EMPLOYEE.SALARY%TYPE;  
BONUS EMPLOYEE.BONUS%TYPE;
```

**BEGIN**

```
SELECT EMP_ID, EMP_NAME, SALARY, NVL(BONUS, 0)  
INTO EMP_ID, EMP_NAME, SALARY, BONUS  
FROM EMPLOYEE  
WHERE EMP_ID = '&EMP_ID';
```

```
DBMS_OUTPUT.PUT_LINE('사번 : ' || EMP_ID);  
DBMS_OUTPUT.PUT_LINE('이름 : ' || EMP_NAME);  
DBMS_OUTPUT.PUT_LINE('급여 : ' || SALARY);
```

```
IF(BONUS = 0)  
    THEN DBMS_OUTPUT.PUT_LINE('보너스를 지급받지 않는 직원입니다.');
```

**END IF;**

```
DBMS_OUTPUT.PUT_LINE('보너스율 : ' || BONUS * 100 || '%');
```

**END;**

대체 변수 입력

×

EMP\_ID에 대한 값 입력::

200

확인

취소

PL/SQL 프로시저가 성공적으로 완료되었습니다.

사번 : 200

이름 : 선동일

급여 : 8000000

보너스율 : 30%

대체 변수 입력

×

EMP\_ID에 대한 값 입력::

201

확인

취소

PL/SQL 프로시저가 성공적으로 완료되었습니다.

사번 : 201

이름 : 송종기

급여 : 6000000

보너스를 지급받지 않는 직원입니다.

보너스율 : 0%

# ▶ PL/SQL

## ✓ 선택문 : IF ~ THEN ~ ELSE ~ END IF

**DECLARE**

```
EMP_ID EMPLOYEE.EMP_ID%TYPE;  
EMP_NAME EMPLOYEE.EMP_NAME%TYPE;  
DEPT_TITLE DEPARTMENT.DEPT_TITLE%TYPE;  
NATIONAL_CODE LOCATION.NATIONAL_CODE%TYPE;  
TEAM VARCHAR2(20);
```

**BEGIN**

```
SELECT EMP_ID, EMP_NAME, DEPT_TITLE, NATIONAL_CODE  
INTO EMP_ID, EMP_NAME, DEPT_TITLE, NATIONAL_CODE  
FROM EMPLOYEE E, DEPARTMENT D, LOCATION L  
WHERE E.DEPT_CODE = D. DEPT_ID AND D.LOCATION_ID = L.LOCAL_CODE AND EMP_ID = '&EMP_ID';
```

```
IF(NATIONAL_CODE = 'KO') THEN TEAM := '국내팀';  
ELSE TEAM := '해외팀';  
END IF;
```

```
DBMS_OUTPUT.PUT_LINE('사번 : ' || EMP_ID);  
DBMS_OUTPUT.PUT_LINE('이름 : ' || EMP_NAME);  
DBMS_OUTPUT.PUT_LINE('부서 : ' || DEPT_TITLE);  
DBMS_OUTPUT.PUT_LINE('소속 : ' || TEAM);
```

**END;**  
/

대체 변수 입력

EMP\_ID에 대한 값 입력::

확인 취소

×

PL/SQL 프로시저가 성공적으로 완료되었습니다.

사번 : 200  
이름 : 선동일  
부서 : 총무부  
소속 : 국내팀

대체 변수 입력

EMP\_ID에 대한 값 입력::

확인 취소

×

PL/SQL 프로시저가 성공적으로 완료되었습니다.

사번 : 203  
이름 : 송은희  
부서 : 해외영업2부  
소속 : 해외팀



# ▶ PL/SQL

## ✓ 선택문 : IF ~ THEN ~ ELSIF ~ ELSE ~ END IF

**DECLARE**

```
SCORE INT;  
GRADE VARCHAR2(2);
```

**BEGIN**

```
SCORE := '&SCORE';
```

```
IF SCORE >= 90 THEN GRADE := 'A';  
ELSIF SCORE >= 80 THEN GRADE := 'B';  
ELSIF SCORE >= 70 THEN GRADE := 'C';  
ELSIF SCORE >= 60 THEN GRADE := 'D';  
ELSE GRADE := 'F';  
END IF;
```

```
DBMS_OUTPUT.PUT_LINE('당신의 점수는 ' || SCORE || '점이고,  
학점은 ' || GRADE || '학점입니다.');
```

**END;**

/

PL/SQL 프로시저가 성공적으로 완료되었습니다.

당신의 점수는 95점이고, 학점은 A 학점 입니다.

PL/SQL 프로시저가 성공적으로 완료되었습니다.

당신의 점수는 65점이고, 학점은 D 학점 입니다.

# ▶ PL/SQL

## ✓ 반복문 : BASIC LOOP

```
DECLARE
    N NUMBER := 1;
BEGIN
    LOOP
        DBMS_OUTPUT.PUT_LINE(N);
        N := N + 1;

        IF N > 5 THEN EXIT;
        END IF;
    END LOOP;
END;
/
```

PL/SQL 프로시저가 성공적으로 완료되었습니다.

1  
2  
3  
4  
5

# ▶ PL/SQL

## ✓ 반복문 : FOR LOOP

```
BEGIN
    FOR N IN 1..5 LOOP
        DBMS_OUTPUT.PUT_LINE(N);
    END LOOP;
END;
/
```

PL/SQL 프로시저가 성공적으로 완료되었습니다.

1  
2  
3  
4  
5

```
BEGIN
    FOR N IN REVERSE 1..5 LOOP
        DBMS_OUTPUT.PUT_LINE(N);
    END LOOP;
END;
/
```

PL/SQL 프로시저가 성공적으로 완료되었습니다.

5  
4  
3  
2  
1

## ▶ PL/SQL

### ✓ 반복문 : WHILE LOOP

```
DECLARE
    N NUMBER := 1;
BEGIN
    WHILE N <= 5 LOOP
        DBMS_OUTPUT.PUT_LINE(N);
        N := N + 1;
    END LOOP;
END;
/
```

PL/SQL 프로시저가 성공적으로 완료되었습니다.

1  
2  
3  
4  
5

# ▶ PL/SQL

## ✓ 타입 변수 선언 : 테이블 타입의 변수 선언과 초기화, 변수 값 출력

**DECLARE**

```
TYPE EMP_ID_TABLE_TYPE IS TABLE OF EMPLOYEE.EMP_ID%TYPE  
INDEX BY BINARY_INTEGER;  
TYPE EMP_NAME_TABLE_TYPE IS TABLE OF EMPLOYEE.EMP_NAME%TYPE  
INDEX BY BINARY_INTEGER;
```

```
EMP_ID_TABLE EMP_ID_TABLE_TYPE;  
EMP_NAME_TABLE EMP_NAME_TABLE_TYPE;
```

```
I BINARY_INTEGER := 0;
```

**BEGIN**

```
FOR K IN (SELECT EMP_ID, EMP_NAME FROM EMPLOYEE) LOOP
```

```
    I := I + 1;
```

```
    EMP_ID_TABLE(I) := K.EMP_ID;
```

```
    EMP_NAME_TABLE(I) := K.EMP_NAME;
```

```
END LOOP;
```

```
FOR J IN 1..I LOOP
```

```
    DBMS_OUTPUT.PUT_LINE('EMP_ID : ' || EMP_ID_TABLE(J) || ', EMP_NAME : ' ||
```

```
EMP_NAME_TABLE(J));
```

```
END LOOP;
```

**END;**

/

PL/SQL 프로시저가 성공적으로 완료되었습니다.

```
EMP_ID : 200, EMP_NAME : 선동일  
EMP_ID : 201, EMP_NAME : 송종기  
EMP_ID : 202, EMP_NAME : 노용철  
EMP_ID : 203, EMP_NAME : 송은희  
EMP_ID : 204, EMP_NAME : 유재식  
EMP_ID : 205, EMP_NAME : 정중하  
EMP_ID : 206, EMP_NAME : 박나라  
EMP_ID : 207, EMP_NAME : 하이유  
EMP_ID : 208, EMP_NAME : 김해솔  
EMP_ID : 209, EMP_NAME : 심봉선  
EMP_ID : 210, EMP_NAME : 윤은혜  
EMP_ID : 211, EMP_NAME : 전형돈  
EMP_ID : 212, EMP_NAME : 장프위  
EMP_ID : 213, EMP_NAME : 하동운  
EMP_ID : 214, EMP_NAME : 방영수  
EMP_ID : 215, EMP_NAME : 대복훈  
EMP_ID : 216, EMP_NAME : 차태연  
EMP_ID : 217, EMP_NAME : 전지연  
EMP_ID : 218, EMP_NAME : 미오리  
EMP_ID : 219, EMP_NAME : 임시환  
EMP_ID : 220, EMP_NAME : 이종석  
EMP_ID : 221, EMP_NAME : 유하진  
EMP_ID : 222, EMP_NAME : 이태림  
EMP_ID : 900, EMP_NAME : 장채현
```

# ▶ PL/SQL

## ✓ 타입 변수 선언 : 레코드 타입의 변수 선언과 초기화, 변수 값 출력

**DECLARE**

```
TYPE EMP_RECORD_TYPE IS RECORD (  
    EMP_ID EMPLOYEE.EMP_ID%TYPE,  
    EMP_NAME EMPLOYEE.EMP_NAME%TYPE,  
    DEPT_TITLE DEPARTMENT.DEPT_TITLE%TITLE,  
    JOB_NAME JOB.JOB_NAME%TYPE  
);
```

```
EMP_RECORD EMP_RECORD_TYPE;
```

**BEGIN**

```
SELECT EMP_ID, EMP_NAME, DEPT_TITLE, JOB_NAME INTO EMP_RECORD  
FROM EMPLOYEE E, DEPARTMENT D, JOB J  
WHERE E.DEPT_CODE = D.DEPT_ID  
        AND E.JOB_CODE = J.JOB_CODE  
        AND EMP_NAME = '&EMP_NAME';
```

```
DBMS_OUTPUT.PUT_LINE('사번 : ' || EMP_RECORD.EMP_ID);  
DBMS_OUTPUT.PUT_LINE('이름 : ' || EMP_RECORD.EMP_NAME);  
DBMS_OUTPUT.PUT_LINE('부서 : ' || EMP_RECORD.DEPT_TITLE);  
DBMS_OUTPUT.PUT_LINE('직급 : ' || EMP_RECORD.JOB_NAME);
```

**END;**

대체 변수 입력

EMP\_NAME에 대한 값 입력::

노웅철

확인 취소

PL/SQL 프로시저가 성공적으로 완료되었습니다.

사번 : 202  
이름 : 노웅철  
부서 : 총무부  
직급 : 부사장

## ▶ PL/SQL

### ✓ 예외처리 : 미리 정의되지 않은 오라클 SERVER 에러 예외 처리

```
DECLARE
    DUP_EMPNO EXCEPTION;
    PRAGMA EXCEPTION_INIT(DUP_EMPNO, -00001);

BEGIN
    UPDATE EMPLOYEE
    SET EMP_ID = '&사번'
    WHERE EMP_ID = 200;

EXCEPTION
    WHEN DUP_EMPNO THEN DBMS_OUTPUT.PUT_LINE('이미 존재하는 사번입니다.');
```

END;  
/

대체 변수 입력

사번에 대한 값 입력::

확인 취소

PL/SQL 프로시저가 성공적으로 완료되었습니다.

DUP\_EMPID가 작동되었습니다.