



20장 PL/SQL 기초

20.1 PL/SQL의 구조



- PL/SQL 은 Oracle's Procedural Language extension to SQL의 약자이다. SQL문장에서 변수정의, 조건처리(IF), 반복처리(LOOP, WHILE, FOR)등을 지원하며, 오라클 자체에 내장되어 있는 절차적 언어(Procedure Language)로서 SQL의 단점을 보완해준다.
- PL/SQL은 SQL에 없는 다음과 같은 기능이 제공된다.
 - 변수 선언을 할 수 있다.
 - 비교 처리를 할 수 있다.
 - 반복 처리를 할 수 있다.
- PL/SQL은 블록(BLOCK) 구조의 언어로서 크게 3 부분으로 나눌 수 있다.
 - 선언부(DECLARE SECTION): PL/SQL에서 사용하는 모든 변수나 상수를 선언하는 부분으로서 DECLARE로 시작한다.
 - 실행부(EXECUTABLE SECTION): 절차적 형식으로 SQL문을 실행할 수 있도록 절차적 언어의 요소인 제어문, 반복문, 함수 정의 등 로직을 기술할 수 있는 부분으로 BEGIN으로 시작한다.
 - 예외 처리(EXCEPTION SECTION): PL/SQL 문이 실행되는 중에 예외가 발생할 수 있는데 이를 예외 사항이라고 한다. 이러한 예외 사항이 발생했을 때 이를 해결하기 위한 문장을 기술할 수 있는 부분으로 EXCEPTION 으로 시작한다.
- PL/SQL 프로그램의 작성 요령은 다음과 같다.
 - PL/SQL 블록내에서는 한 문장이 종료할 때마다 세미콜론(;)을 사용한다.
 - END뒤에 ;을 사용하여 하나의 블록이 끝났다는 것을 명시한다.
 - PL/SQL 블록의 작성은 편집기를 통해 파일로 작성할 수도 있고, 프롬프트에서 바로 작성할 수도 있다.
 - SQL*PLUS환경에서는 DECLARE나 BEGIN이라는 키워드로 PL/SQL블록이 시작하는 것을 알 수 있다.
 - 단일행 주석은 --이고 여러행 주석 /* */이다.
 - 쿼리문을 수행하기 위해서 /가 반드시 입력되어야 하며, PL/SQL 블록은 행에 / 가 있으면 종결된 것으로 간주한다.

```
-- 형식
DECLARE
    변수 선언;
BEGIN
    실행문;
END;
/

-- 예
SET SERVEROUTPUT ON -- 출력해 주는 내용을 화면에 보여주도록 설정한다.
BEGIN
    DBMS_OUTPUT.PUT_LINE('Hello World!');
END;
/
```

20.2 변수 선언과 대입문

- PL/SQL의 선언부에서는 실행부에서 사용할 변수를 선언한다. 변수를 선언할 때 변수명 다음에 자료형을 기술해야 한다.
- PL/SQL에서 변수 선언할 때 사용되는 자료형은 SQL에서 사용하던 자료형과 거의 유사하다.

```
-- 형식
identifier [CONSTANT] datatype [NOT NULL]
[:= | DEFAULT expression];

-- identifier: 변수의 이름
-- CONSTANT: 변수의 값을 변경할 수 없도록 제약한다.
-- datatype: 자료형을 기술한다.
-- NOT NULL: 값을 반드시 포함하도록 하기 위해 변수를 제약한다.
-- Expression: Literal, 다른 변수, 연산자나 함수를 포함하는 표현식

-- 예
VEMPNO NUMBER(4);
VENAME VARCHAR2(10);
```

20.2.1 대입문으로 변수에 값 지정하기

- PL/SQL에서는 변수의 값을 지정하거나 재지정하기 위해서 :=를 사용한다. := 의 좌측에 새 값을 받기 위한 변수를 기술하고 우측에 저장할 값을 기술한다.

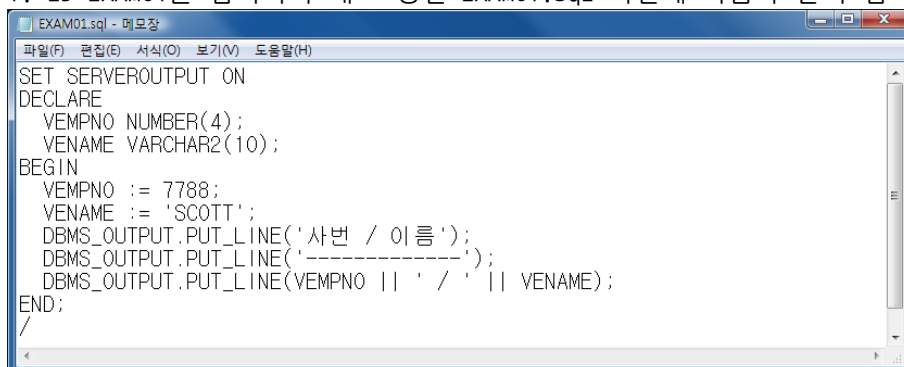
```
-- 형식
identifier := expression;

-- 예
VEMPNO := 7788;
VENAME := 'SCOTT';
```

[실습] 변수 사용하기

- 변수의 선언 및 할당을 하고 그 변수 값을 출력해 본다.

1. ED EXAM01을 입력하여 새로 생긴 EXAM01.sql 파일에 다음과 같이 입력하라.



```
SET SERVEROUTPUT ON
DECLARE
  VEMPNO NUMBER(4);
  VENAME VARCHAR2(10);
BEGIN
  VEMPNO := 7788;
  VENAME := 'SCOTT';
  DBMS_OUTPUT.PUT_LINE('사번 / 이름');
  DBMS_OUTPUT.PUT_LINE('-----');
  DBMS_OUTPUT.PUT_LINE(VEMPNO || ' / ' || VENAME);
END;
/
```

2. 작성이 완료한 후에 파일을 저장한다. SQL> 프롬프트에 @EXAM01을 입력하면 EXAM01.sql 파일

내부에 기술한 PL/SQL 이 실행된 후 결과가 출력된다.

```
C:\Windows\system32\cmd.exe - sqlplus scott/tiger

SQL> ED EXAM01

SQL> @EXAM01
사번 / 이름
-----
7788 / SCOTT

PL/SQL 처리가 정상적으로 완료되었습니다.

SQL>
```

20.2.2 스칼라 변수/레퍼런스 변수

- PL/SQL 에서 변수를 선언하기 위해 사용할 수 있는 데이터형은 크게 스칼라(Scalar)와 레퍼런스(Reference)로 나눌 수 있다.
- 스칼라: PL/SQL 에서 변수를 선언할 때 사용되는 자료형은 SQL 에서 사용하던 자료형과 거의 유사하다. 숫자를 저장하려면 NUMBER 를 사용하고 문자를 저장하려면 VARCHAR2 를 사용해서 선언한다.

```
VEMPNO NUMBER(4);
VENAME VARCHAR2(10);
```

- 레퍼런스: 이전에 선언된 다른 변수 또는 데이터베이스 컬럼에 맞추어 변수를 선언하기 위해 %TYPE 속성을 사용한다.

```
VEMPNO EMP.EMPNO%TYPE;
VENAME EMP.ENAME%TYPE;
```

20.2.3 PL/SQL에서 SELECT INTO문

- 데이터베이스에서 정보를 추출할 필요가 있을 때 또는 데이터베이스로 변경된 내용을 적용할 필요가 있을 때 SQL 을 사용한다.
- PL/SQL 은 SQL 에 있는 DML 명령을 지원한다. 테이블의 행에서 질의된 값을 변수에 할당시키기 위해 SELECT 문장을 사용한다.
- PL/SQL 의 SELECT 문은 INTO 절이 필요한데, INTO 절에는 데이터를 저장할 변수를 기술한다.
- SELECT 절에 있는 컬럼은 INTO 절에 있는 변수와 1 대 1 대응을 하기에 개수와 데이터의 형, 길이가 일치하여야 한다.
- SELECT 문은 INTO 절에 의해 하나의 행만을 저장할 수 있다.

```
-- 형식
SELECT select_list
INTO {variable_name1[, variable_name2,...] / record_name}
FROM table_name
WHERE condition;

-- 예
```



```
SELECT EMPNO, ENAME INTO VEMPNO, VENAME
FROM EMP
WHERE ENAME='SCOTT';
```

[실습] 사번과 이름 검색하기

- PL/SQL의 SELECT 문으로 EMP 테이블에서 사원번호와 이름을 조회한다.

1. ED 다음에 파일이름을 입력하여 새로 생긴 SQL 파일에 다음과 같이 입력하라.

[EXAM02.SQL]

```
01 SET SERVEROUTPUT ON
02 DECLARE
03 -- %TYPE 속성으로 컬럼 단위 레퍼런스 변수 선언
04 VEMPNO EMP.EMPNO%TYPE;
05 VENAME EMP.ENAME%TYPE;
06 BEGIN
07 DBMS_OUTPUT.PUT_LINE('사번 / 이름');
08 DBMS_OUTPUT.PUT_LINE('-----');
09
10 SELECT EMPNO, ENAME INTO VEMPNO, VENAME
11 FROM EMP
12 WHERE ENAME='SCOTT';
13
14 -- 레퍼런스 변수에 저장된 값을 출력한다.
15 DBMS_OUTPUT.PUT_LINE(VEMPNO || ' / ' || VENAME);
16 END;
17 /
```

2. 작성이 완료한 후에 파일을 저장한다. SQL> 프롬프트에 @파일명을 입력하면 SQL 파일 내부에 기술한 PL/SQL 이 실행된 후 결과가 출력된다.

C:\Temp>SQLPLUS SCOTT/TIGER

SQL> ed EXAM02.SQL

SQL> @EXAM02.SQL

사번 / 이름

7788 / SCOTT

PL/SQL 처리가 정상적으로 완료되었습니다.

SQL> quit

20.2.4 PL/SQL 테이블 TYPE

- PL/SQL 테이블은 로우에 대해 배열처럼 액세스하기 위해 기본키를 사용한다.
- 배열과 유사하고 PL/SQL 테이블을 액세스하기 위해 BINARY_INTEGER 데이터형의 기본키와 PL/SQL 테이블 요소를 저장하는 스칼라 또는 레코드 데이터형의 컬럼을 포함해야 한다.

Primary key	Column
.....
1	SMITH
2	ALLEN
3	WARD
.....
BINARY_INTEGER	스칼라

```
TYPE table_type_name IS TABLE OF
{column_type | variable%TYPE | table.column%TYPE} [NOT NULL]
[INDEX BY BINARY_INTEGER];
identifier table_type_name;
```

[실습] TABLE 변수 사용하기

■ TABLE 변수를 사용하여 EMP 테이블에서 이름과 업무를 출력해 본다.

1. ED 다음에 파일이름을 입력하여 새로 생긴 SQL 파일에 다음과 같이 입력하라.

[EXAM03.SQL]

```
01 SET SERVEROUTPUT ON
02 DECLARE
03   -- 테이블 타입을 정의
04   TYPE ENAME_TABLE_TYPE IS TABLE OF EMP.ENAME%TYPE
05     INDEX BY BINARY_INTEGER;
06   TYPE JOB_TABLE_TYPE IS TABLE OF EMP.JOB%TYPE
07     INDEX BY BINARY_INTEGER;
08
09   -- 테이블 타입으로 변수 선언
10   ENAME_TABLE ENAME_TABLE_TYPE;
11   JOB_TABLE JOB_TABLE_TYPE;
12
13   I BINARY_INTEGER := 0;
14
15 BEGIN
16   -- EMP 테이블에서 사원이름과 직급을 얻어옴
17   FOR K IN (SELECT ENAME, JOB FROM EMP) LOOP
18     I := I + 1;           --인덱스 증가
19     ENAME_TABLE(I) := K.ENAME; --사원이름과
20     JOB_TABLE(I) := K.JOB;   --직급을 저장.
21   END LOOP;
22
23   --테이블에 저장된 내용을 출력
24   FOR J IN 1..I LOOP
25     DBMS_OUTPUT.PUT_LINE(RPAD(ENAME_TABLE(J),12)
26       || ' / ' || RPAD(JOB_TABLE(J),9));
27   END LOOP;
28 END;
29 /
```

2. 작성을 완료한 후에 파일을 저장한다. SQL> 프롬프트에 @파일명을 입력하면 SQL 파일 내부에

기술한 PL/SQL이 실행된 후 결과가 출력된다.

```
C:\Temp>SQLPLUS SCOTT/TIGER
```

```
SQL> ed EXAM03.SQL
```

```
SQL> @EXAM03.SQL
```

```
SMITH      / CLERK
ALLEN      / SALESMAN
WARD       / SALESMAN
JONES      / MANAGER
MARTIN     / SALESMAN
BLAKE      / MANAGER
CLARK      / MANAGER
SCOTT      / ANALYST
KING       / PRESIDENT
TURNER     / SALESMAN
ADAMS      / CLERK
JAMES      / CLERK
FORD       / ANALYST
MILLER     / CLERK
```

PL/SQL 처리가 정상적으로 완료되었습니다.

```
SQL>
```

20.2.5 PL/SQL RECORD TYPE

- PL/SQL RECORD TYPE은 프로그램 언어의 구조체와 유사하다.
- PL/SQL RECORD는 FIELD(ITEM)들의 집합을 하나의 논리적 단위로 처리할 수 있게 해주므로 테이블의 ROW를 읽어올 때 편리하다.

[실습] RECORD TYPE 사용하기

- EMP 테이블에서 SCOTT 사원의 정보를 출력해 본다.

1. ED 다음에 파일이름을 입력하여 새로 생긴 SQL 파일에 다음과 같이 입력하라.

```
[EXAM04.SQL]
```

```
01  SET SERVEROUTPUT ON
02  DECLARE
03      -- 레코드 타입을 정의
04      TYPE emp_record_type IS RECORD(
05          v_empno    emp.empno%TYPE,
06          v_ename    emp.ename%TYPE,
07          v_job      emp.job%TYPE,
08          v_deptno   emp.deptno%TYPE);
09
10      -- 레코드로 변수 선언
11      emp_record    emp_record_type;
12  BEGIN
13      -- SCOTT 사원의 정보를 레코드 변수에 저장
14      SELECT empno,ename, job, deptno
15          INTO emp_record
16          FROM emp
```

```

17     WHERE ename = UPPER('SCOTT');
18
19     -- 레코드 변수에 저장된 사원 정보를 출력
20     DBMS_OUTPUT.PUT_LINE('사원번호 : ' || TO_CHAR(emp_record.v_empno));
21     DBMS_OUTPUT.PUT_LINE('이름 : ' || emp_record.v_ename);
22     DBMS_OUTPUT.PUT_LINE('담당업무 : ' || emp_record.v_job);
23     DBMS_OUTPUT.PUT_LINE('부서번호 : ' || TO_CHAR(emp_record.v_deptno));
24 END;
25 /

```

2. 작성을 완료한 후에 파일을 저장한다. SQL> 프롬프트에 @파일명을 입력하면 SQL 파일 내부에 기술한 PL/SQL이 실행된 후 결과가 출력된다.

```

C:\Temp>SQLPLUS SCOTT/TIGER

SQL> ed EXAM04.SQL

SQL> @EXAM04.SQL
사원번호 : 7788
이름 : SCOTT
담당업무 : ANALYST
부서번호 : 20

PL/SQL 처리가 정상적으로 완료되었습니다.

SQL>

```

20.3 PL/SQL의 제어문

- PL/SQL의 제어문은 어떤 조건에서 어떤 코드가 실행되어야 하는지를 제어하기 위한 문법으로, 절차적 언어의 구성요소를 포함함.

구문	의미	문법
BEGIN-END	<ul style="list-style-type: none"> • PL/SQL 문을 블록화시킴 • 중첩 가능 	BEGIN { SQL 문 } END
IF-ELSE	<ul style="list-style-type: none"> • 조건의 검사 결과에 따라 문장을 선택적으로 수행 	IF <조건> SQL 문 [ELSE SQL 문] END IF;
FOR	<ul style="list-style-type: none"> • counter 값이 범위 내에 있을 경우 FOR 문의 블록을 실행 	FOR counter IN <범위> {SQL 문} END LOOP
WHILE	<ul style="list-style-type: none"> • 조건이 참일 경우 WHILE 문의 블록을 실행 	WHILE <조건> { SQL 문 BREAK CONTINUE } END LOOP
RETURN	<ul style="list-style-type: none"> • 프로시저를 종료 • 상태값을 정수로 반환 가능 	RETURN [<정수>]

20.3.1 IF~THEN~END IF

```
IF condition THEN ..... 조건문
statements; ..... 조건에 만족할 경우 실행되는 문장
END IF
```

[실습] 부서 번호로 부서명 알아내기

- 다음은 사원 번호가 7788인 사원의 부서 번호를 얻어 와서 부서 번호에 따른 부서명을 구하는 예제이다. IF문이 끝났을 때에는 반드시 END IF를 기술해야 한다는 점에 주의해야 한다.

1. ED 다음에 파일이름을 입력하여 새로 생긴 SQL 파일에 다음과 같이 입력하라.

[EXAM05.SQL]

```
01  SET SERVEROUTPUT ON
02  DECLARE
03      VEMPNO          NUMBER(4);
04      VENAME          VARCHAR2(20);
05      VDEPTNO         EMP.DEPTNO%TYPE;
06      VDNNAME         VARCHAR2(20) := NULL;
07  BEGIN
08      SELECT EMPNO, ENAME, DEPTNO INTO VEMPNO, VENAME, VDEPTNO
09      FROM EMP
10      WHERE EMPNO=7788;
11
12      IF (VDEPTNO = 10) THEN
13          VDNNAME := 'ACCOUNTING';
14      END IF;
15      IF (VDEPTNO = 20) THEN
16          VDNNAME := 'RESEARCH';
17      END IF;
18      IF (VDEPTNO = 30) THEN
19          VDNNAME := 'SALES';
20      END IF;
21      IF (VDEPTNO = 40) THEN
22          VDNNAME := 'OPERATIONS';
23      END IF;
24
25      DBMS_OUTPUT.PUT_LINE('사번   이름   부서명');
26      DBMS_OUTPUT.PUT_LINE(VEMPNO||'   '||VENAME||'   '||VDNNAME);
27  END;
28  /
```

2. 작성을 완료한 후에 파일을 저장한다. SQL> 프롬프트에 @파일명을 입력하면 SQL 파일 내부에 기술한 PL/SQL이 실행된 후 결과가 출력된다.

```
C:\Temp>SQLPLUS SCOTT/TIGER
```

```
SQL> ed EXAM05.SQL
```

```
SQL> @EXAM05.SQL
```

```
사번   이름   부서명
7788   SCOTT   RESEARCH
```

```
PL/SQL 처리가 정상적으로 완료되었습니다.
```

```
SQL>
```


20.3.2 IF~THEN~ELSE~END IF

```
[문장1]
IF condition THEN ..... 조건문
statements; ..... 조건에 만족할 경우 실행되는 문장[문장2]
ELSE
statements; ..... 조건에 만족하지 않을 경우 실행되는 문장[문장3]
END IF
[문장4]
```

[실습하기] 직원의 연봉 구하기

- 다음은 연봉을 구하는 예제이다. 커미션을 받는 직원은 급여에 12를 곱한 후 커미션과 합산하여 연봉을 구하고 커미션을 받지 않는 직원은 급여에 12를 곱한 것으로만 연봉을 구한다.

1. ED 다음에 파일이름을 입력하여 새로 생긴 SQL 파일에 다음과 같이 입력하라.

```
[EXAM06.SQL]

01  SET SERVEROUTPUT ON
02  DECLARE
03      VEMP EMP%ROWTYPE;
04      ANNSAL NUMBER(7,2);
05  BEGIN
06      -- SCOTT 사원의 전체 정보를 로우 단위로 얻어와 VEMP에 저장한다.
07      SELECT * INTO VEMP
08      FROM EMP
09      WHERE ENAME='SCOTT';
10
11      IF (VEMP.COMM IS NULL) THEN      -- 커미션이 NULL 이면
12          ANNSAL:=VEMP.SAL*12;        -- 급여에 12를 곱한다.
13      ELSE                            -- 커미션이 NULL이 아니면
14          ANNSAL:=VEMP.SAL*12+VEMP.COMM;-- 급여에 12를 곱한 후 커미션과 합산
15      END IF;
16
17      DBMS_OUTPUT.PUT_LINE('사번 / 이름 / 연봉');
18      DBMS_OUTPUT.PUT_LINE('-----');
19      DBMS_OUTPUT.PUT_LINE(VEMP.EMPNO||'/'||VEMP.ENAME||'/'||ANNSAL);
20  END;
21  /
```

2. 작성을 완료한 후에 파일을 저장한다. SQL> 프롬프트에 @파일명을 입력하면 SQL 파일 내부에 기술한 PL/SQL이 실행된 후 결과가 출력된다.

```
C:\Temp>SQLPLUS SCOTT/TIGER

SQL> ed EXAM06.SQL

SQL> @EXAM06.SQL
사번 / 이름 / 연봉
-----
7788/SCOTT/36000

PL/SQL 처리가 정상적으로 완료되었습니다.
```

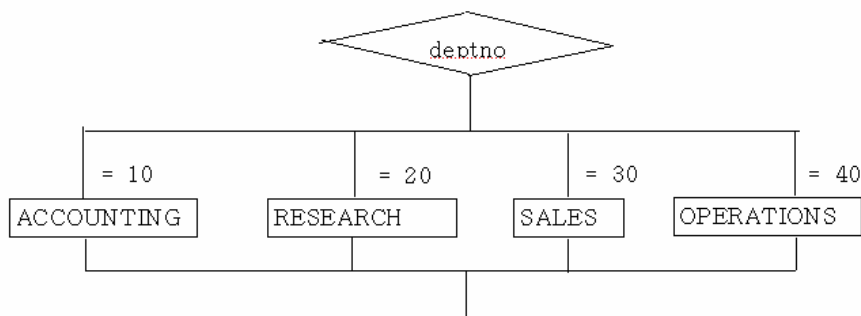
SQL>

20.3.3 IF~THEN~ELSEIF~ELSE~END IF

```
IF condition THEN
statements;
ELSIF condition THEN
statements;
ELSIF condition THEN
statements;
ELSE
statements;
END IF
```

[실습] 부서 번호로 부서명 알아내기

- SQL 함수에서 선택을 위한 DECODE 함수를 학습하면서 부서번호에 대해서 부서명을 지정해 보았다.



- 이곳 PL/SQL에서는 DECODE 함수 대신 IF ~ THEN ~ ELSEIF ~ ELSE ~ END IF 구문으로 부서번호에 대한 부서명을 구해 보자.

1. ED 다음에 파일이름을 입력하여 새로 생긴 SQL 파일에 다음과 같이 입력하라.

[EXAM07.SQL]

```
01 SET SERVEROUTPUT ON
02 DECLARE
03     VEMP EMP%ROWTYPE;
04     VDNAME VARCHAR2(14);
05 BEGIN
06     DBMS_OUTPUT.PUT_LINE('사번 / 이름 / 부서명');
07     DBMS_OUTPUT.PUT_LINE('-----');
08
09     SELECT * INTO VEMP
10     FROM EMP
11     WHERE ENAME='SCOTT';
12
13     IF (VEMP.DEPTNO = 10) THEN
14         VDNAME := 'ACCOUNTING';
15     ELSIF (VEMP.DEPTNO = 20) THEN
16         VDNAME := 'RESEARCH';
17     ELSIF (VEMP.DEPTNO = 30) THEN
18         VDNAME := 'SALES';
19     ELSIF (VEMP.DEPTNO = 40) THEN
```

```

20      VDDNAME := 'OPERATIONS';
21      END IF;
22
23      DBMS_OUTPUT.PUT_LINE(VEMP.EMPNO||'/'||VEMP.ENAME||'/'||VDDNAME);
24      END;
25      /

```

2. 작성을 완료한 후에 파일을 저장한다. SQL> 프롬프트에 @파일명을 입력하면 SQL 파일 내부에 기술한 PL/SQL이 실행된 후 결과가 출력된다.

```

C:\Temp>SQLPLUS SCOTT/TIGER

SQL> ed EXAM07.SQL

SQL> @EXAM07.SQL
사번 / 이름 / 부서명
-----
7788/SCOTT/RESEARCH

PL/SQL 처리가 정상적으로 완료되었습니다.

SQL>

```

20.3.4 BASIC LOOP문

- 지금 소개할 구문은 가장 간단한 루프로 구분 문자로 LOOP와 END LOOP가 사용된다.
- 실행 상의 흐름이 END LOOP에 도달할 때마다 그와 짝을 이루는 LOOP 문으로 제어가 되돌아간다.
- 이러한 루프를 무한 루프라 하며, 여기서 빠져나가려면 EXIT문을 사용한다.
- 기본 LOOP는 LOOP에 들어갈 때 조건이 이미 일치했다 할지라도 적어도 한번은 문장이 실행된다.

```

LOOP
statement1;
statement2;
. . . . .
EXIT [WHERE condition];
END LOOP

```

[실습] BASIC LOOP문으로 1부터 5까지 출력하기

- 다음은 BASIC LOOP 문으로 1부터 5까지 출력하는 예제이다.

1. ED 다음에 파일이름을 입력하여 새로 생긴 SQL 파일에 다음과 같이 입력한다.

```

[EXAM08.SQL]

01      SET SERVEROUTPUT ON
02      DECLARE

```

```

03      N  NUMBER := 1;
04      BEGIN
05          LOOP
06              DBMS_OUTPUT.PUT_LINE( N );
07              N := N + 1;
08              IF N > 5 THEN
09                  EXIT;
10              END IF;
11          END LOOP;
12      END;
13      /

```

2. 작성을 완료한 후에 파일을 저장한다. SQL> 프롬프트에 @파일명을 입력하면 SQL 파일 내부에 기술한 PL/SQL이 실행된 후 결과가 출력된다.

```

C:\Temp>SQLPLUS SCOTT/TIGER

SQL> ed EXAM08.SQL

SQL> @EXAM08.SQL
1
2
3
4
5

PL/SQL 처리가 정상적으로 완료되었습니다.

SQL>

```

20.3.5 FOR LOOP문

```

-- 형식
FOR index_counter
IN [REVERSE] lower_bound..upper_bound LOOP
    statement1;
    statement2;
    . . . . .
END LOOP

```

[실습] FOR LOOP문으로 1부터 5까지 출력하기

■ 다음은 FOR LOOP 문으로 1부터 5까지 출력하는 예제이다.

1. ED 다음에 파일이름을 입력하여 새로 생긴 SQL 파일에 다음과 같이 입력한다.

```

[EXAM09.SQL]

01      SET SERVEROUTPUT ON
02      DECLARE
03      BEGIN
04          FOR N IN 1..5 LOOP
05              DBMS_OUTPUT.PUT_LINE( N );
06          END LOOP;

```

```
07 END;
08 /
```

2. 작성을 완료한 후에 파일을 저장한다. SQL> 프롬프트에 @파일명을 입력하면 SQL 파일 내부에 기술한 PL/SQL이 실행된 후 결과가 출력된다.

```
C:\Temp>SQLPLUS SCOTT/TIGER
```

```
SQL> ed EXAM09.SQL
```

```
SQL> @EXAM09.SQL
```

```
1
2
3
4
5
```

```
PL/SQL 처리가 정상적으로 완료되었습니다.
```

```
SQL>
```

20.3.6 WHILE LOOP문

```
WHILE condition LOOP
  statement1;
  statement2;
  . . . . .
END LOOP
```

[실습] WHILE LOOP문으로 1부터 5까지 출력하기

■ 다음은 WHILE LOOP 문으로 1부터 5까지 출력하는 예제이다.

1. ED 다음에 파일이름을 입력하여 새로 생긴 SQL 파일에 다음과 같이 입력한다.

```
[EXAM10.SQL]
```

```
01 SET SERVEROUTPUT ON
02 DECLARE
03   N NUMBER := 1;
04 BEGIN
05   WHILE N <= 5 LOOP
06     DBMS_OUTPUT.PUT_LINE( N );
07     N := N + 1;
08   END LOOP;
09 END;
10 /
```

2. 작성을 완료한 후에 파일을 저장한다. SQL> 프롬프트에 @파일명을 입력하면 SQL 파일 내부에 기술한 PL/SQL이 실행된 후 결과가 출력된다.

```
C:\Temp>SQLPLUS SCOTT/TIGER
```

```
SQL> ed EXAM010.SQL
```

```
SQL> @EXAM10.SQL
```

```
1
```

```
2
```

```
3
```

```
4
```

```
5
```

```
PL/SQL 처리가 정상적으로 완료되었습니다.
```

```
SQL>
```