



## 20장 PL/SQL 기초

오라클사에서 SQL 언어에 절차적인 프로그래밍 언어를 가미해 만든 것이 PL/SQL에 대해서 학습하겠습니다.

## 이 장에서 다룰 내용



1 **PL/SQL 구조**

---

2 **변수선언과 대입문**

---

3 **선택문**

---

4 **반복문**

## 01. PL/SQL 구조



- ❖ PL/SQL 은 Oracle's Procedural Language extension to SQL의 약자입니다. SQL문장에서 변수정의, 조건처리(IF), 반복처리(LOOP, WHILE, FOR)등을 지원하며, 오라클 자체에 내장되어 있는 절차적 언어(Procedure Language)로서 SQL의 단점을 보완해줍니다.
- ❖ 오라클 사에서는 데이터베이스 내의 데이터를 조작하기 위해서 SQL과 함께 PL/SQL을 제공합니다. C나 자바와 같은 프로그래밍 언어에 익숙한 사람이라면 절차적 언어인 PL/SQL을 쉽게 이해할 수 있을 것입니다.

## 01. PL/SQL 구조



❖ PL/SQL은 SQL에 없는 다음과 같은 기능이 제공됩니다.

- 변수 선언을 할 수 있습니다.
- 비교 처리를 할 수 있습니다.
- 반복 처리를 할 수 있습니다.

❖ 위에서 언급한 기능은 절차적 언어에서 제공되는 것으로 효율적으로 SQL 문을 사용할 수 있게 해줍니다.

## 01. PL/SQL 구조



- ❖ PL/SQL은 PASCAL과 유사한 구조로서 DECLARE~BEGIN~EXCEPTION~END 순서를 갖습니다.
- ❖ PL/SQL은 다음과 같은 블록(BLOCK) 구조의 언어로서 크게 3 부분으로 나눌 수 있습니다.

DECLARE SECTION(선언부)

EXECUTABLE SECTION(실행부)

EXCEPTION SECTION(예외 처리부)

# 01. PL/SQL 구조



## ❖ 선언부(DECLARE SECTION)

- PL/SQL에서 사용하는 모든 변수나 상수를 선언하는 부분으로서 DECLARE로 시작합니다.

## ❖ 실행부(EXECUTABLE SECTION)

- 절차적 형식으로 SQL문을 실행할 수 있도록 절차적 언어의 요소인 제어문, 반복문, 함수 정의 등 로직을 기술할 수 있는 부분으로 BEGIN으로 시작합니다.

## ❖ 예외 처리(EXCEPTION SECTION)

- PL/SQL 문이 실행되는 중에 에러가 발생할 수 있는데 이를 예외 사항이라고 합니다. 이러한 예외 사항이 발생했을 때 이를 해결하기 위한 문장을 기술할 수 있는 부분으로 EXCEPTION 으로 시작합니다.

## 01. PL/SQL 구조

❖ PL/SQL 프로그램의 작성 요령은 다음과 같습니다.



1. PL/SQL 블록내에서는 한 문장이 종료할 때마다 세미콜론(;)을 사용합니다.
2. END뒤에 ;을 사용하여 하나의 블록이 끝났다는 것을 명시합니다.
3. PL/SQL 블록의 작성은 편집기를 통해 파일로 작성할 수도 있고, 프롬프트에서 바로 작성할 수도 있습니다.
4. SQL\*PLUS환경에서는 DELCLARE나 BEGIN이라는 키워드로 PL/SQL블록이 시작하는 것을 알 수 있습니다.
5. 단일행 주석은 --이고 여러행 주석 /\* \*/입니다.
6. 쿼리문을 수행하기 위해서 /가 반드시 입력되어야 PL/SQL 블록은 행에 / 가 있으면 종결된 것으로 간주합니다.

## 〈실습하기〉 간단한 메시지 출력하기

'Hello World!'를 출력하는 PL/SQL 문을 작성해 봅시다.

1. 오라클의 환경 변수 **SERVEROUTPUT**는 오라클에서 제공하는 프로시저를 사용하여 출력해 주는 내용을 화면에 보여주도록 설정하는 환경 변수인데 디폴트값 **OFF**이기에 **ON**으로 변경해야만 합니다.

**SET SERVEROUTPUT ON**

2. 화면 출력을 위해서는 **PUT\_LINE**이란 프로시저를 이용합니다. **PUT\_LINE**은 오라클이 제공하는 프로시저로 **DBMS\_OUTPUT** 패키지에 묶여 있습니다. 따라서 **DBMS\_OUTPUT.PUT\_LINE**과 같이 사용해야 합니다.

```
BEGIN
DBMS_OUTPUT.PUT_LINE('Hello world!');
END;
/
```



## 02. 변수선언과 대입문



- ❖ PL/SQL의 선언부에서는 실행부에서 사용할 변수를 선언합니다. 변수를 선언할 때 변수명 다음에 자료형을 기술해야 합니다.
- ❖ PL/SQL에서 변수 선언할 때 사용되는 자료형은 SQL에서 사용하던 자료형과 거의 유사합니다.

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

구문	설명
identifier	변수의 이름
CONSTANT	변수의 값을 변경할 수 없도록 제약합니다.
datatype	자료형을 기술합니다.
NOT NULL	값을 반드시 포함하도록 하기 위해 변수를 제약합니다.
Expression	Literal, 다른 변수, 연산자나 함수를 포함하는 표현식

## 02. 변수선언과 대입문



- ❖ 쿼리문을 수행하고 난 후에 얻어진 결과를 컬럼 단위로 변수에 저장할 경우 다음과 같이 선언합니다.

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

- ❖ PL/SOL에서 변수를 선언할 때 위와 같이 SQL에서 사용하던 자료형과 유사하게 선언하는 것을 스칼라(SCALAR) 변수라고 합니다.
- ❖ 숫자를 저장하기 위해서 VEMPNO 변수는 NUMBER로 선언하고 VENAME 변수는 문자를 저장하려면 VARCHAR2를 사용해서 선언하였습니다.

## 2.1 대입문으로 변수에 값 지정하기



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

```
identifier := expression;
```

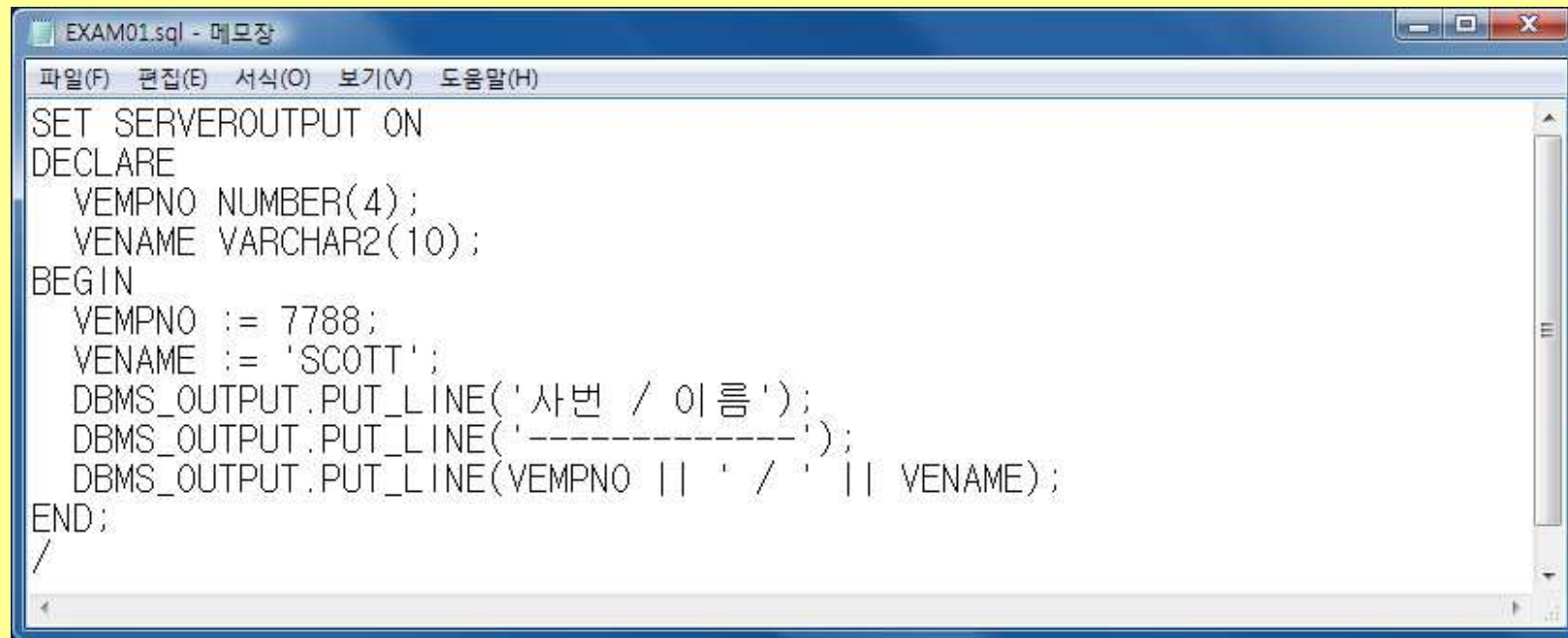
- ❖ 선언부에서 선언한 변수에 값을 할당하기 위해서는 :=를 사용해 봅시다.

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

## 〈실습하기〉 변수 사용하기

변수의 선언 및 할당을 하고 그 변수 값을 출력해 봅시다.

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

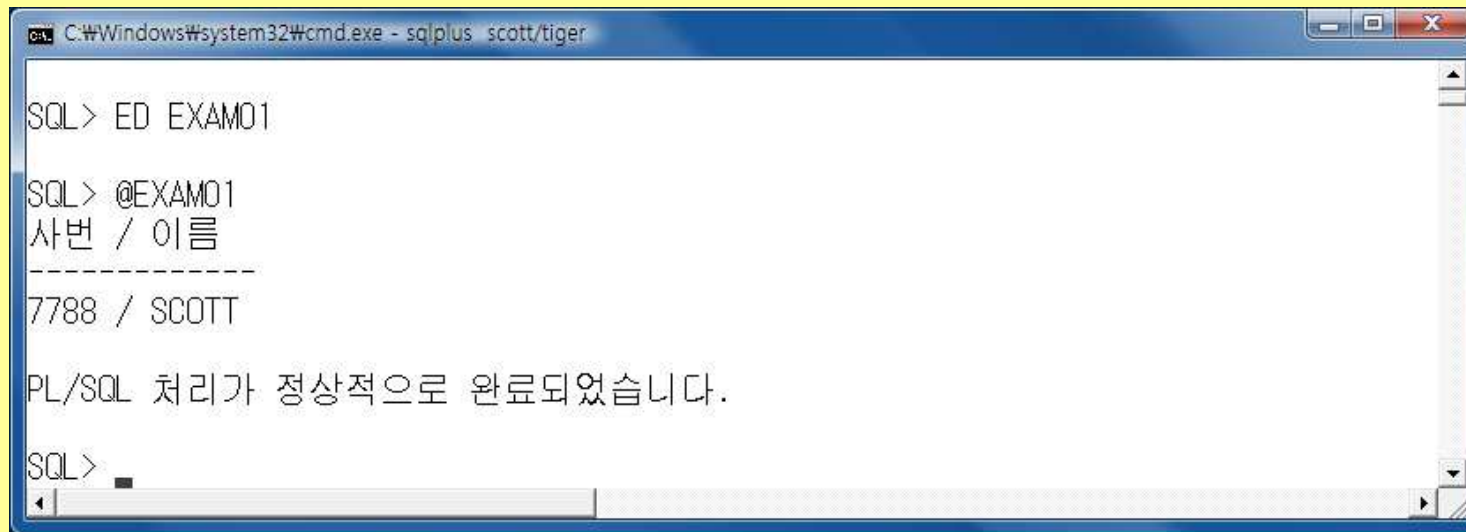


```
EXAM01.sql - 메모장
파일(F)  편집(E)  서식(O)  보기(V)  도움말(H)

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;
/
```

## 〈실습하기〉 변수 사용하기

2. 작성이 완료한 후에 파일을 저장합니다 SQL> 프롬프트에 @EXAM01을 입력하면 EXAM01.sql 파일 내부에 기술한 PL/SQL 이 실행된 후 결과가 출력됩니다.



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

SQL> ED EXAM01

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

PL/SQL 처리가 정상적으로 완료되었습니다.

SQL>
```

## 2.2 스칼라 변수/레퍼런스 변수



- ❖ PL/SQL에서 변수를 선언하기 위해 사용할 수 있는 데이터형은 크게 스칼라(Scalar)와 레퍼런스(Reference)로 나눌 수 있습니다.

- ❖ 스칼라

- PL/SQL에서 변수를 선언할 때 사용되는 자료형은 SQL에서 사용하던 자료형과 거의 유사하다. 숫자를 저장하려면 NUMBER를 사용하고 문자를 저장하려면 VARCHAR2를 사용해서 선언합니다.

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

- ❖ 레퍼런스

- 이전에 선언된 다른 변수 또는 데이터베이스 컬럼에 맞추어 변수를 선언하기 위해 %TYPE속성을 사용할 수 있습니다.

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

## 2.2 스칼라 변수/레퍼런스 변수



### ❖ 레퍼런스

```
VEMPNO EMP.EMPNO%TYPE;  
          ①      ②  
VENAME EMP.ENAME%TYPE;  
          ①      ②
```

- ❖ %TYPE속성을 사용하여 선언한 VEMPNO 변수는 해당 테이블(①EMP)의 해당 칼럼(①EMPNO 혹은 ②)의 자료형과 크기를 그대로 참조해서 정의합니다.
- ❖ 모든 개발자가 테이블에 정의된 칼럼의 자료형과 크기를 모두 파악하고 있다면 별 문제가 없겠지만, 대부분은 그렇지 못하기 때문에 오라클에서는 레퍼런스(REFERENCES) 변수를 제공합니다.
- ❖ 컬럼의 자료형이 변경되더라도 칼럼의 자료형과 크기를 그대로 참조하기 때문에 굳이 레퍼런스 변수 선언을 수정할 필요가 없다는 장점이 있습니다.

## 2.2 스칼라 변수/레퍼런스 변수



- ❖ %TYPE가 컬럼 단위로 참조한다면 로우(행) 단위로 참조하는 %ROWTYPE가 있습니다.
- ❖ 데이터베이스의 테이블 또는 VIEW의 일련의 컬럼을 RECORD로 선언하기 위하여 %ROWTYPE를 사용합니다.
- ❖ 데이터베이스 테이블 이름을 %ROWTYPE 앞에 접두어를 붙여 RECORD를 선언하고 FIELD는 테이블이나 VIEW의 COLUMN명과 데이터 타입과 LENGTH를 그대로 가져올 수 있습니다.

```
VEMP EMP%ROWTYPE;
```

- ❖ %ROWTYPE을 사용 시 장점은 특정 테이블의 컬럼의 개수와 데이터 형식을 모드더라도 지정할 수 있습니다.
- ❖ SELECT 문장으로 로우를 검색할 때 유리합니다.



## 2.3 PL/SQL에서 SELECT 문



- ❖ 데이터베이스에서 정보를 추출할 필요가 있을 때 또는 데이터베이스로 변경된 내용을 적용할 필요가 있을 때 SQL을 사용합니다.
- ❖ PL/SQL은 SQL에 있는 DML 명령을 지원합니다. 테이블의 행에서 질의된 값을 변수에 할당시키기 위해 SELECT문장을 사용합니다.
- ❖ PL/SQL의 SELECT 문은 INTO절이 필요한데, INTO절에는 데이터를 저장할 변수를 기술합니다.
- ❖ SELECT 절에 있는 컬럼은 INTO절에 있는 변수와 1대1대응을 하기에 개수와 데이터의 형, 길이가 일치하여야 합니다.
- ❖ SELECT 문은 INTO절에 의해 하나의 행만을 저장할 수 있습니다.

## 2.3 PL/SQL에서 SELECT 문



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

구문	설명
<b>select_list</b>	열의 목록이며 행 함수, 그룹 함수, 표현식을 기술할 수 있습니다.
<b>variable_name</b>	읽어들인 값을 저장하기 위한 스칼라 변수
<b>record_name</b>	읽어 들인 값을 저장하기 위한 PL/SQL RECORD 변수
<b>Condition</b>	PL/SQL 변수와 상수를 포함하여 열명,표현식,상수,비교 연산자로 구성되며 오직 하나의 값을 RETURN할 수 있는 조건이어야 합니다.

## 2.3 PL/SQL에서 SELECT 문



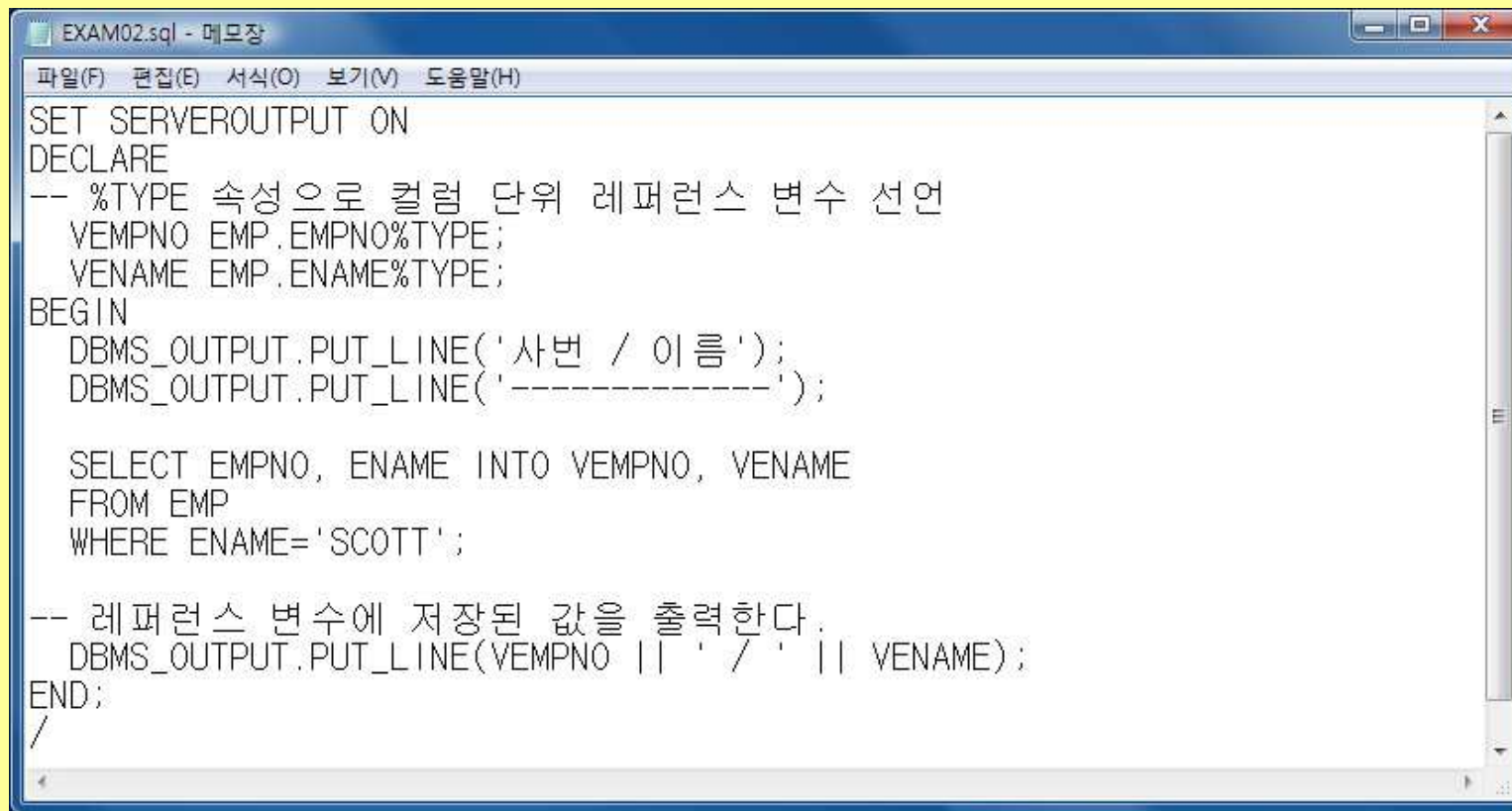
```
SELECT EMPNO, ENAME INTO VEMPNO, VENAME  
FROM EMP  
WHERE ENAME='SCOTT';
```

- ❖ VEMPNO, VENAME 변수는 컬럼(EMPNO, ENAME)과 동일한 데이터형을 갖도록 하기 위해서 %TYPE 속성을 사용합니다.
- ❖ INTO 절의 변수는 SELECT에서 기술한 컬럼의 데이터형뿐만 아니라 컬럼의 수와도 일치해야 합니다.

## 〈실습하기〉 사번과 이름 검색하기

PL/SQL의 **SELECT** 문으로 **EMP** 테이블에서 사원번호와 이름을 조회합니다.

1. ED 다음에 파일이름을 입력하여 새로 생긴 **SQL** 파일에 다음과 같이 입력하십시오.(파일이름:EXAM02.sql)



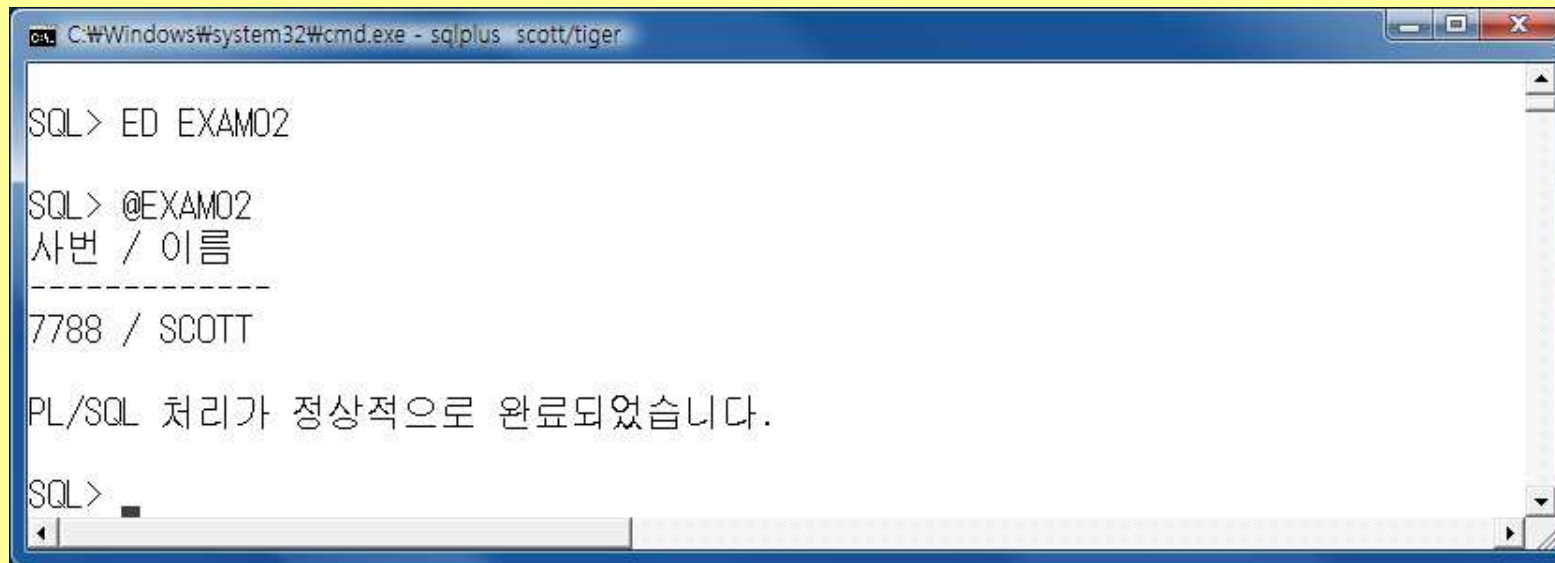
```
EXAM02.sql - 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
SET SERVEROUTPUT ON
DECLARE
-- %TYPE 속성으로 컬럼 단위 레퍼런스 변수 선언
  VEMPNO EMP.EMPNO%TYPE;
  VENAME EMP.ENAME%TYPE;
BEGIN
  DBMS_OUTPUT.PUT_LINE('사번 / 이름');
  DBMS_OUTPUT.PUT_LINE('-----');

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

-- 레퍼런스 변수에 저장된 값을 출력한다.
  DBMS_OUTPUT.PUT_LINE(VEMPNO || ' / ' || VENAME);
END;
/
```

## 〈실습하기〉 변수 사용하기

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



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

SQL> ED EXAM02

SQL> @EXAM02
사번 / 이름
-----
7788 / SCOTT

PL/SQL 처리가 정상적으로 완료되었습니다.

SQL>
```

## 2.4 PL/SQL 테이블



- ❖ PL/SQL 테이블은 로우에 대해 배열처럼 액세스하기 위해 기본키를 사용합니다.
- ❖ 배열과 유사하고 PL/SQL 테이블을 액세스하기 위해 BINARY\_INTEGER 데이터형의 기본키와 PL/SQL 테이블 요소를 저장하는 스칼라 또는 레코드 데이터형의 컬럼을 포함해야 합니다.
- ❖ 또한 이들은 동적으로 자유롭게 증가할 수 있습니다.

Primary key

.....
1
2
3
.....

BINARY\_INTEGER

Column

.....
SMITH
ALLEN
WARD
.....

스칼라

## 2.4 PL/SQL 테이블



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

구문	설명
<b>table_type_name</b>	테이블형의 이름
<b>column_type</b>	VARCHAR2,DATE,NUMBER과 같은 스칼라 데이터 형
<b>identifier</b>	전체 PL/SQL 테이블을 나타내는 식별자의 이름

## 〈실습하기〉 TABLE 변수 사용하기

TABLE 변수를 사용하여 EMP 테이블에서 이름과 업무를 출력해 봅시다.

1. ED 다음에 파일이름을 입력하여 새로 생긴 SQL 파일에 다음과 같이 입력하십시오.(파일이름:EXAM03.sql)

```
SET SERVEROUTPUT ON  
DECLARE
```

```
-- 테이블 타입을 정의
```

```
TYPE ENAME_TABLE_TYPE IS TABLE OF EMP.ENAME%TYPE  
INDEX BY BINARY_INTEGER;
```

```
TYPE JOB_TABLE_TYPE IS TABLE OF EMP.JOB%TYPE  
INDEX BY BINARY_INTEGER;
```

```
-- 테이블 타입으로 변수 선언
```

```
ENAME_TABLE ENAME_TABLE_TYPE;  
JOB_TABLE JOB_TABLE_TYPE;
```

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



## 〈실습하기〉 TABLE 변수 사용하기

```
BEGIN
  -- EMP 테이블에서 사원이름과 직급을 얻어옴
  FOR K IN (SELECT ENAME, JOB FROM EMP) LOOP
    I := I + 1;                --인덱스 증가
    ENAME_TABLE(I) := K.ENAME; --사원이름과
    JOB_TABLE(I) := K.JOB;     --직급을 저장.
  END LOOP;

  --테이블에 저장된 내용을 출력
  FOR J IN 1..I LOOP
    DBMS_OUTPUT.PUT_LINE(RPAD(ENAME_TABLE(J),12)
      || ' / ' || RPAD(JOB_TABLE(J),9));
  END LOOP;
END;
/
```

## 2.5 PL/SQL RECORD TYPE



- ❖ PL/SQL RECORD TYPE은 프로그램 언어의 구조체와 유사합니다.
- ❖ PL/SQL RECORD는 FIELD(ITEM)들의 집합을 하나의 논리적 단위로 처리할 수 있게 해 주므로 테이블의 ROW를 읽어올 때 편리합니다.

```
TYPE type_name IS RECORD  
(field_name1 {scalar_datatype/record_type}  
[NOT NULL] [{:= | DEFAULT} expr],  
field_name2 {scalar_datatype/record_type}  
[NOT NULL] [{:= | DEFAULT} expr],  
. . . . .);  
identiffee_name type_name;
```

type_name	RECODE 형의 이름, 이 식별자는 RECODE를 선언하기 위해 사용합니다.
field_name	RECODE내의 필드명입니다.

## 2.5 PL/SQL RECORD TYPE



- ❖ 개별 필드를 참조하거나 초기화하기 위해 "."을 사이에 두고 레코드 이름과 필드 이름을 기술합니다.

```
Record_name.field_name
```

## 〈실습하기〉 RECORD TYPE 사용하기

EMP 테이블에서 SCOTT 사원의 정보를 출력해 봅시다.

1. ED 다음에 파일이름을 입력하여 새로 생긴 SQL 파일에 다음과 같이 입력하십시오.(파일이름:EXAM04.sql)

```
SET SERVEROUTPUT ON
DECLARE
  -- 레코드 타입을 정의
  TYPE emp_record_type IS RECORD(
    v_empno   emp.empno%TYPE,
    v_ename   emp.ename%TYPE,
    v_job     emp.job%TYPE,
    v_deptno  emp.deptno%TYPE);

  -- 레코드로 변수 선언
  emp_record emp_record_type;
```

## 〈실습하기〉 RECORD TYPE 사용하기

```
BEGIN
  -- SCOTT 사원의 정보를 레코드 변수에 저장
  SELECT empno,ename, job, deptno
    INTO emp_record
   FROM emp
  WHERE ename = UPPER('SCOTT');

  -- 레코드 변수에 저장된 사원 정보를 출력
  DBMS_OUTPUT.PUT_LINE('사원번호 : ' ||
    TO_CHAR(emp_record.v_empno));
  DBMS_OUTPUT.PUT_LINE('이름: ' ||
    emp_record.v_ename);
  DBMS_OUTPUT.PUT_LINE('담당업무 : ' ||
    emp_record.v_job);
  DBMS_OUTPUT.PUT_LINE('부서번호 : ' ||
    TO_CHAR(emp_record.v_deptno));
END;
/
```

### 03. 선택문



- ❖ 기본적으로 모든 문장들은 나열된 순서대로 순차적으로 수행됩니다.
- ❖ 하지만 경우에 따라서는 문장의 흐름을 변경할 필요가 있습니다. 이때 사용하는 것이 IF 문입니다.
- ❖ IF 문은 조건을 제시해서 만족하느냐 하지 않느냐에 따라 문장을 선택적으로 수행하기 때문에 선택문이라고 합니다.
- ❖ 오라클에서는 3가지 형태의 선택문이 제공됩니다.

## 3.1 IF-THEN-END IF



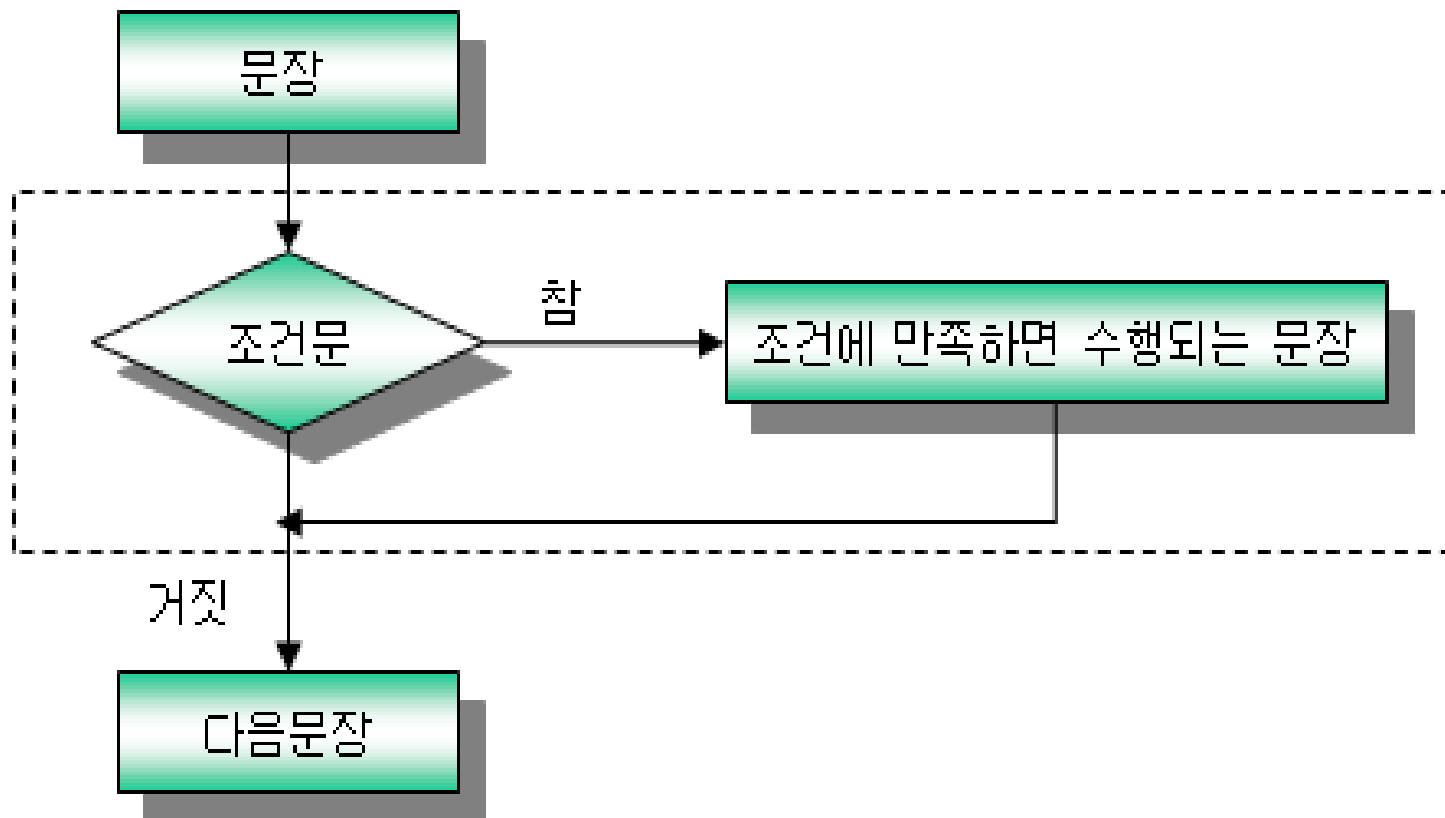
- ❖ if 라는 단어의 사전적인 의미는 “만약 ~ 라면” 입니다.
- ❖ 이러한 의미처럼 if문은 조건에 따라 어떤 명령을 선택적으로 처리하기 위해 사용하는 가장 대표적인 구문입니다.
- ❖ “어떤 경우에 어떤 행동을 해라!” 와 같은 간단한 처리를 할 때 사용합니다.

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

## 3.1 IF-THEN-END IF



- ❖ 조건이 TRUE이면 THEN이하의 문장을 실행하고 조건이 FALSE나 NULL이면 END IF다음 문장을 수행합니다.





## 〈실습하기〉 부서번호로 부서명 알아내기

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

1. ED 다음에 파일이름을 입력하여 새로 생긴 SQL 파일에 다음과 같이 입력하십시오.(파일이름:EXAM05.sql)

```
SET SERVEROUTPUT ON
DECLARE
  VEMPNO      NUMBER(4);
  VENAME      VARCHAR2(20);
  VDEPTNO     EMP.DEPTNO%TYPE;
  VDNAM      VARCHAR2(20) := NULL;
BEGIN
  SELECT EMPNO, ENAME, DEPTNO INTO VEMPNO,
    VENAME, VDEPTNO
  FROM EMP
  WHERE EMPNO=7788;
```

## 〈실습하기〉 RECORD TYPE 사용하기

```
IF (VDEPTNO = 10) THEN
    VDNAM := 'ACCOUNTING';
END IF;
IF (VDEPTNO = 20) THEN
    VDNAM := 'RESEARCH';
END IF;
IF (VDEPTNO = 30) THEN
    VDNAM := 'SALES';
END IF;
IF (VDEPTNO = 40) THEN
    VDNAM := 'OPERATIONS';
END IF;

DBMS_OUTPUT.PUT_LINE('사번   이름   부서명');
DBMS_OUTPUT.PUT_LINE(VEMPNO||'   '||VDNAM||'   '||VENAM);
END;
/
```

## 〈탄탄히 다지기〉

1. 다음은 연봉을 구하는 예제입니다. 연봉 계산을 위해 “급여\*12+커미션”이란 공식을 사용하겠습니다. (CH20\_QUIZ\_01.sql)

```
SET SERVEROUTPUT ON
DECLARE
-- %ROWTYPE 속성으로 로우 전체를 저장할 수 있는 레퍼런스 변수 선언
VEMP EMP%ROWTYPE;
ANNSAL NUMBER(7,2);
BEGIN
DBMS_OUTPUT.PUT_LINE('사번 / 이름 / 연봉');
DBMS_OUTPUT.PUT_LINE('-----');
-- SCOTT 사원의 전체 정보를 로우 단위로 얻어와 VEMP에 저장한다.
SELECT * INTO VEMP
FROM EMP
WHERE ENAME='SCOTT';
```

-- 커미션이 NULL일 경우 0으로 변경해야 올바른 연봉 계산

① \_\_\_\_\_

② \_\_\_\_\_

③ \_\_\_\_\_

-- 스칼라 변수에 연봉을 계산할 결과를 저장한다.

**ANNSAL:=VEMP.SAL\*12+VEMP.COMM ;**

-- 결과 출력

**DBMS\_OUTPUT.PUT\_LINE(VEMP.EMPNO||' / '**  
**||VEMP.ENAME||' /'||ANNSAL);**

**END;**

**/**

위 결과를 보면 연봉으로 아무런 내용도 출력되지 않음을 확인할 수 있습니다. 왜냐하면 **COMM** 칼럼 값이 **NULL**이기에 값과의 연산 결과인 연봉 역시 **NULL**이 구해지기 때문입니다. 따라서 커미션 칼럼이 **NULL** 값일 때는 **0** 값으로 바꿔줘야 올바른 연봉 계산을 할 수 있게 됩니다. 공란을 채워서 다음과 같이 올바른 결과를 얻을 수 있도록 하시오.

## 3.2 IF ~ THEN ~ ELSE ~ END IF



- ❖ if 문 계열 중 가장 일반적으로 많이 사용되는 형식이 IF ~ THEN ~ ELSE ~ END IF 문입니다.
- ❖ 이 문장은 참일 때와 거짓일 때 각각 다른 문장을 수행하도록 지정할 수 있습니다.

[문장1]

IF *condition* THEN ..... 조건문

*statements; ..... 조건에 만족할 경우 실행되는 문장*[문장2]

ELSE

*statements; ..... 조건에 만족하지 않을 경우 실행되는 문장*[문장3]

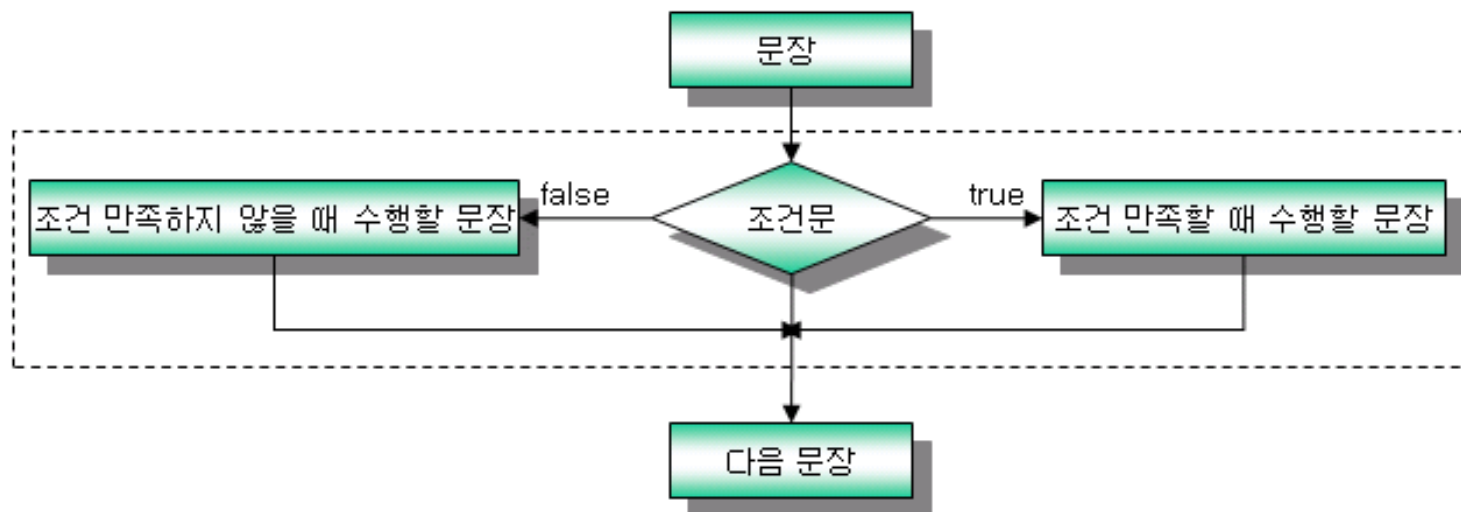
END IF

[문장4]

## 3.2 IF ~ THEN ~ ELSE ~ END IF



- ❖ [문장1]을 수행하고 if 문을 만나면 조건문을 검사합니다. 그리고 그 결과가 참이면 [문장2]를 수행하고, 거짓이면 [문장3]을 수행합니다. 그런 후에는 [문장4]를 수행하게 됩니다.



## 〈실습하기〉 연봉 구하는 예제

다음은 연봉을 구하는 예제입니다. 커미션을 받는 직원은 급여에 12를 곱한 후 커미션과 합산하여 연봉을 구하고 커미션을 받지 않는 직원은 급여에 12를 곱한 것으로만 연봉을 구합니다.

1. ED 다음에 파일이름을 입력하여 새로 생긴 SQL 파일에 다음과 같이 입력하십시오.(파일이름:EXAM06.sql)

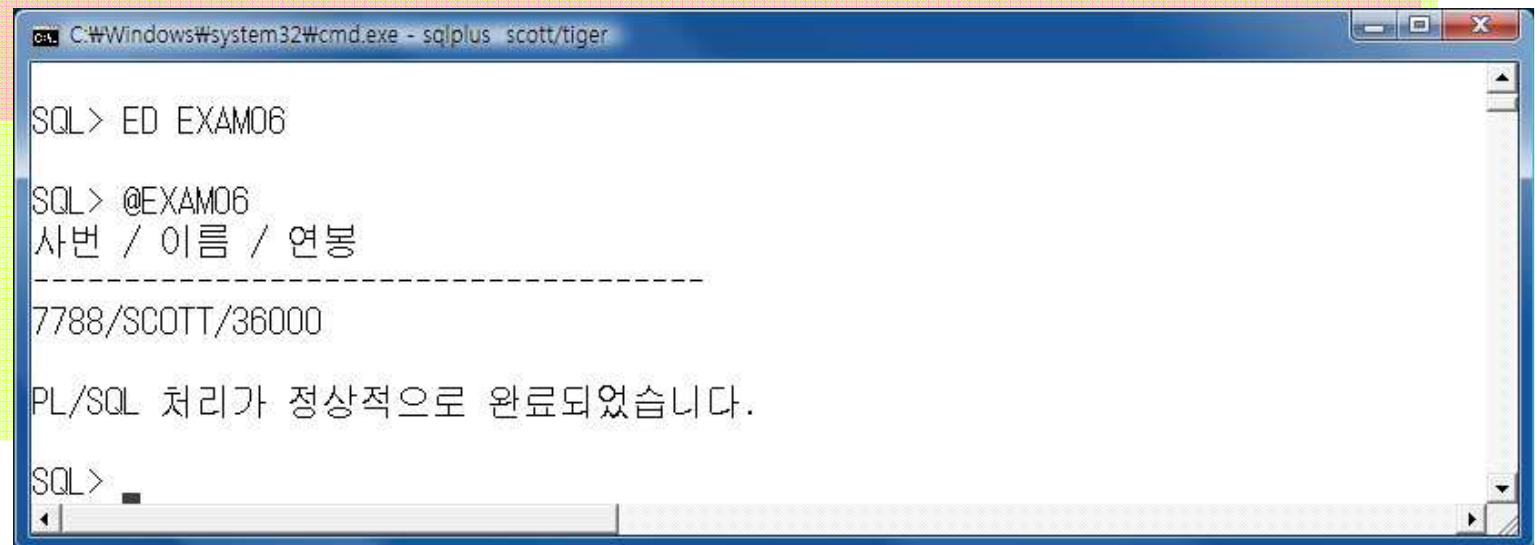
```
SET SERVEROUTPUT ON
DECLARE
  VEMP EMP%ROWTYPE;
  ANNSAL NUMBER(7,2);
BEGIN
  -- SCOTT 사원의 전체 정보를 로우 단위로 얻어와 VEMP에 저장한다.
  SELECT * INTO VEMP
  FROM EMP
  WHERE ENAME='SCOTT';
```

## 〈실습하기〉 연봉 구하는 예제

```
IF (VEMP.COMM IS NULL) THEN      -- 커미션이 NULL 이면
    ANNSAL:=VEMP.SAL*12;          -- 급여에 12를 곱한다.
ELSE                             -- 커미션이 NULL이 아니면
    ANNSAL:=VEMP.SAL*12+VEMP.COMM;
    -- 급여에 12를 곱한 후 커미션과 합산
END IF;

DBMS_OUTPUT.PUT_LINE('사번 / 이름 / 연봉');
DBMS_OUTPUT.PUT_LINE('-----');
DBMS_OUTPUT.PUT_LINE(VEMP.EMPNO||' / '
                      ||VEMP.ENAME||'/'||ANNSAL);

END;
/
```



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

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

PL/SQL 처리가 정상적으로 완료되었습니다.

SQL>
```

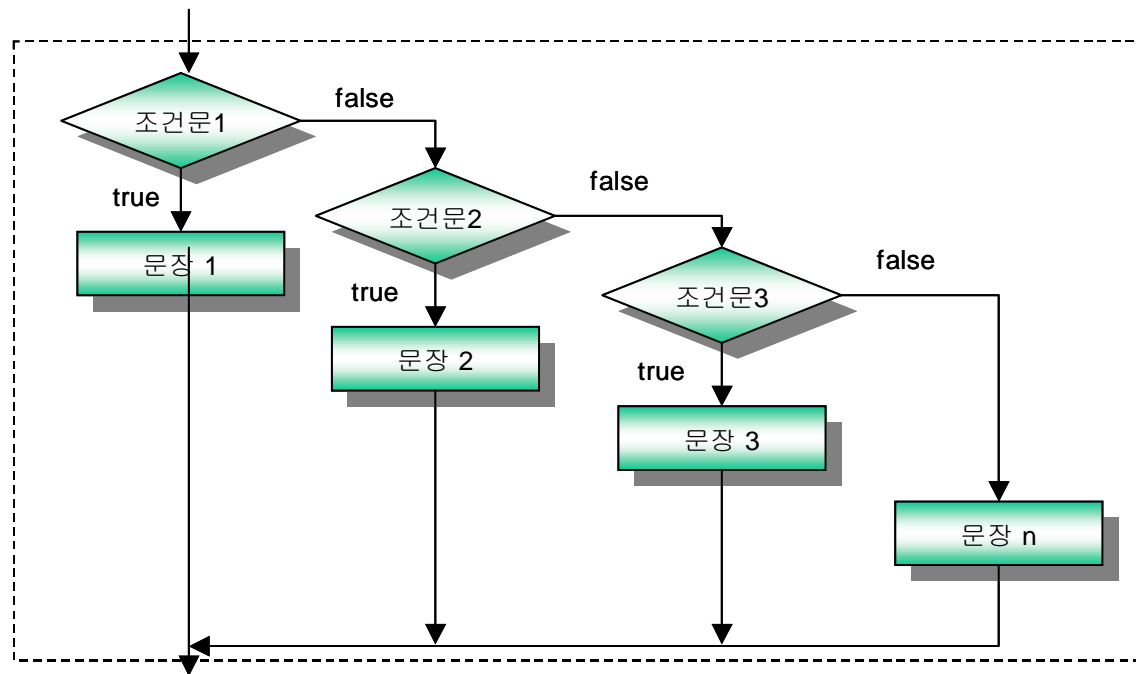


### 3.3 IF ~ THEN ~ ELSIF ~ ELSE ~ END IF



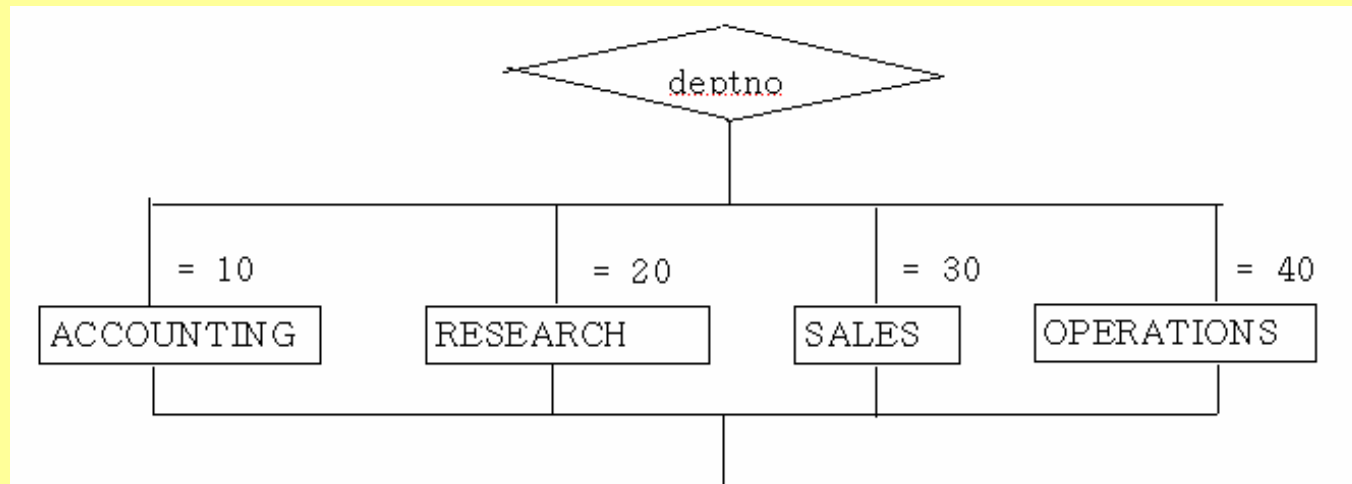
- ❖ IF ~ THEN ~ ELSE ~ END IF 문은 참 거짓을 선택하는 과정에서 한번만 사용되었지만, 이럴 경우 둘 중에 하나를 선택할 수 있습니다.
- ❖ 만일 그 경우의 수가 둘이 아닌 셋 이상에서 하나를 선택해야 할 경우에는 IF ~ THEN ~ ELSIF ~ ELSE ~ END IF 문을 사용해야 합니다.

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



## 〈실습하기〉 부서번호로 부서명 알아내기

SQL 함수에서 선택을 위한 **DECODE** 함수를 학습하면서 부서번호에 대해서 부서명을 지정해 보았습니다.



이곳 PL/SQL에서는 **DECODE** 함수 대신 **IF ~ THEN ~ ELSIF ~ ELSE ~ END IF** 구문으로 부서번호에 대한 부서명을 구해 봅시다.

1. **ED** 다음에 파일이름을 입력하여 새로 생긴 **SQL** 파일에 다음과 같이 입력하십시오.(파일이름:EXAM07.sql)

## 〈실습하기〉 부서번호로 부서명 알아내기

```
SET SERVEROUTPUT ON
DECLARE
  VEMP EMP%ROWTYPE;
  VDNAME VARCHAR2(14);
BEGIN
  DBMS_OUTPUT.PUT_LINE('사번 / 이름 / 부서명');
  DBMS_OUTPUT.PUT_LINE('-----');

  SELECT * INTO VEMP
  FROM EMP
  WHERE ENAME='SCOTT';

  IF (VEMP.DEPTNO = 10) THEN
    VDNAME := 'ACCOUNTING';
  ELSIF (VEMP.DEPTNO = 20) THEN
    VDNAME := 'RESEARCH';
```

## 〈실습하기〉 부서번호로 부서명 알아내기

```
ELSIF (VEMP.DEPTNO = 30) THEN
    VDNAME := 'SALES';
ELSIF (VEMP.DEPTNO = 40) THEN
    VDNAME := 'OPERATIONS';
END IF;

DBMS_OUTPUT.PUT_LINE(VEMP.EMPNO||'/'||
VEMP.ENAME||'/'||VDNAME);
END;
/
```

## 04. 반복문



- ❖ 반복문은 SQL 문을 반복적으로 여러 번 실행하고자 할 때 사용합니다.
- ❖ PL/SQL에서는 다음과 같이 다양한 반복문이 사용됩니다.

1. 조건 없이 반복 작업을 제공하기 위한 BASIC LOOP문
2. COUNT를 기본으로 작업의 반복 제어를 제공하는 FOR LOOP문
3. 조건을 기본으로 작업의 반복 제어를 제공하기 위한 WHILE LOOP문
4. LOOP를 종료하기 위한 EXIT문

## 4.1 BASIC LOOP문



- ❖ 지금 소개할 구문은 가장 간단한 루프로 구분 문자로 LOOP와 END LOOP가 사용됩니다.

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

- ❖ 실행 상의 흐름이 END LOOP에 도달할 때마다 그와 짝을 이루는 LOOP 문으로 제어가 되돌아갑니다.
- ❖ 이러한 루프를 무한 루프라 하며, 여기서 빠져나가려면 EXIT문을 사용합니다.
- ❖ 기본 LOOP는 LOOP에 들어갈 때 조건이 이미 일치했다 할지라도 적어도 한번은 문장이 실행됩니다.

## 〈실습하기〉 BASIC LOOP 문으로 1부터 5까지 출력하기

다음은 BASIC LOOP 문으로 1부터 5까지 출력하는 예제입니다.

1. ED 다음에 파일이름을 입력하여 새로 생긴 SQL 파일에 다음과 같이 입력하십시오.(파일이름:EXAM08.sql)

```
SET SERVEROUTPUT ON
DECLARE
N NUMBER := 1;
BEGIN
LOOP
DBMS_OUTPUT.PUT_LINE( N );
N := N + 1;
IF N > 5 THEN
EXIT;
END IF;
END LOOP;
END;
/
```

## <탄탄히 다지기>

다음은 BASIC LOOP 문으로 5 단을 출력하는 예제입니다. 공란을 채우시오  
(CH20\_QUIZ\_02.sql)

```
SET SERVEROUTPUT ON
```

```
DECLARE
```

```
DAN NUMBER := 5;
```

```
I NUMBER := 1;
```

```
BEGIN
```

```
LOOP
```

```
DBMS_OUTPUT.PUT_LINE(DAN || ' * ' || I || ' = ' || (DAN*I
```

① \_\_\_\_\_

```
IF ② _____ THEN
```

③ \_\_\_\_\_

```
END IF;
```

④ \_\_\_\_\_

```
END;
```

```
/
```



## 4.2 FOR LOOP문



- ❖ FOR LOOP는 반복되는 횟수가 정해진 반복문을 처리하기에 용이합니다.

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

구문	설명
<i>index_counter</i>	<i>upper_bound</i> 나 <i>lower_bound</i> 에 도달할 때까지 LOOP를 반복함으로써 1씩 자동적으로 증가하거나 감소되는 값을 가진 암시적으로 선언된 정수입니다.
REVERSE	<i>upper_bound</i> 에서 <i>lower_bound</i> 까지 반복함으로써 인덱스가 1씩 감소되도록 합니다.
<i>lower_bound</i>	<i>index_counter</i> 값의 범위에 대한 하단 바운드값을 지정합니다.
<i>upper_bound</i>	<i>index_counter</i> 값의 범위에 대한 상단 바운드값을 지정합니다.

## 4.2 FOR LOOP문



- ❖ FOR LOOP 문에서 사용되는 인덱스는 정수로 자동 선언되므로 따로 선언할 필요가 없다.
- ❖ FOR LOOP 문은 LOOP을 반복할 때마다 자동적으로 1씩 증가 또는 감소합니다. REVERSE는 1씩 감소함을 의미합니다.

## 〈실습하기〉 FOR LOOP 문으로 1부터 5까지 출력하기

다음은 FOR LOOP 문으로 1부터 5까지 출력하는 예제입니다.

1. ED 다음에 파일이름을 입력하여 새로 생긴 SQL 파일에 다음과 같이 입력하십시오.(파일이름:EXAM09.sql)

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

## <탄탄히 다지기>

3. 다음은 FOR 문으로 부서 번호를 계산한 후 이를 SELECT 문의 WHERE 절에 지정하여 부서 정보를 얻어오는 예입니다. 빈 공간을 채우시오  
(CH20\_QUIZ\_03.sql)

```
SET SERVEROUTPUT ON  
DECLARE
```

①

```
BEGIN
```

```
DBMS_OUTPUT.PUT_LINE('부서번호 / 부서명 / 지역명');
```

```
DBMS_OUTPUT.PUT_LINE('-----');
```

```
-- 변수 CNT는 1부터 1씩 증가하다가 4에 도달하면 반복문에서 벗어난다.
```

②

```
SELECT * INTO VDEPT
```

```
FROM DEPT
```

```
WHERE DEPTNO=10*CNT;
```

```
DBMS_OUTPUT.PUT_LINE(VDEPT.DEPTNO || ' / ' || VDEPT.DNAME  
|| ' / ' || VDEPT.LOC);
```

```
END LOOP;
```

```
END;
```

```
/
```

## 4.3 WHILE LOOP 문



- ❖ 제어 조건이 TRUE인 동안만 일련의 문장을 반복하기 위해 WHILE LOOP 문장을 사용합니다. 조건은 반복이 시작될 때 체크하게 되어 LOOP내의 문장이 한 번도 수행되지 않을 경우도 있습니다. LOOP을 시작할 때 조건이 FALSE이면 반복 문장을 탈출하게 됩니다.

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

## 〈실습하기〉 WHILE LOOP 문으로 1부터 5까지 출력하기

다음은 WHILE LOOP 문으로 1부터 5까지 출력하는 예제입니다.

1. ED 다음에 파일이름을 입력하여 새로 생긴 SQL 파일에 다음과 같이 입력하십시오.(파일이름:EXAM10.sql)

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

## <탄탄히 다지기>

4. 다음은 WHILE LOOP 문으로 별을 삼각형 구도로 출력하는 예입니다. 빈 공간을 채우시오.(CH20\_QUIZ\_04.sql)

```
SET SERVEROUTPUT ON
DECLARE
V_CNT NUMBER := 1;
V_STR VARCHAR2(10) := NULL;
BEGIN
WHILE ① _____ LOOP
V_STR := V_STR || '*';
DBMS_OUTPUT.PUT_LINE(V_STR);
② _____
③ _____
END;
/
```