

# PL/SQL 기초

자바 강의실

## 목차

---

I. PL/SQL 구조

II. 변수 선언과 대입문

# 1. 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 보완
- 오라클 사에서는 데이터베이스 내의 데이터를 조작하기 위해서 SQL과 함께 PL/SQL을 제공해줍니다. C나 자바와 같은 프로그래밍 언어에 익숙한 사람이라면 절차적 언어인 PL/SQL을 쉽게 이해

# 1. PL/SQL 구조

- PL/SQL은 SQL에 없는 다음과 같은 기능이 제공

- 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 사용

# 1. 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)

# 1. 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.

# 1. PL/SQL 구조

- PL/SQL 프로그램의 작성 요령은 다음과 같습니다.

PL/SQL 블록내에서는 한 문장이 종료할 때마다 세미콜론(;)을 사용합니다.

END뒤에 ;을 사용하여 하나의 블록이 끝났다는 것을 명시합니다.

PL/SQL 블록의 작성은 편집기를 통해 파일로 작성할 수도 있고, 프롬프트에서 바로 작성할 수도 있습니다.

SQL\*PLUS환경에서는 DELCLARE나 BEGIN이라는 키워드로 PL/SQL블럭이 시작하는 것을 알 수 있습니다.

단일행 주석은 --이고 여러행 주석 /\* \*/입니다.

쿼리문을 수행하기 위해서 /가 반드시 입력되어야 PL/SQL 블록은 행에 / 가 있으면 종결된 것으로 간주합니다.

# 1. PL/SQL 구조

- 실습

## 예: 간단한 메시지 출력하기

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

오라클의 환경 변수 SERVEROUTPUT는 오라클에서 제공해주는 프로시저를 사용하여 출력해 주는 내용을 화면에 보여주도록 설정하는 환경 변수인데 디폴트값 OFF이기에 ON으로 변경해야만 합니다.

```
SET SERVEROUTPUT ON
```

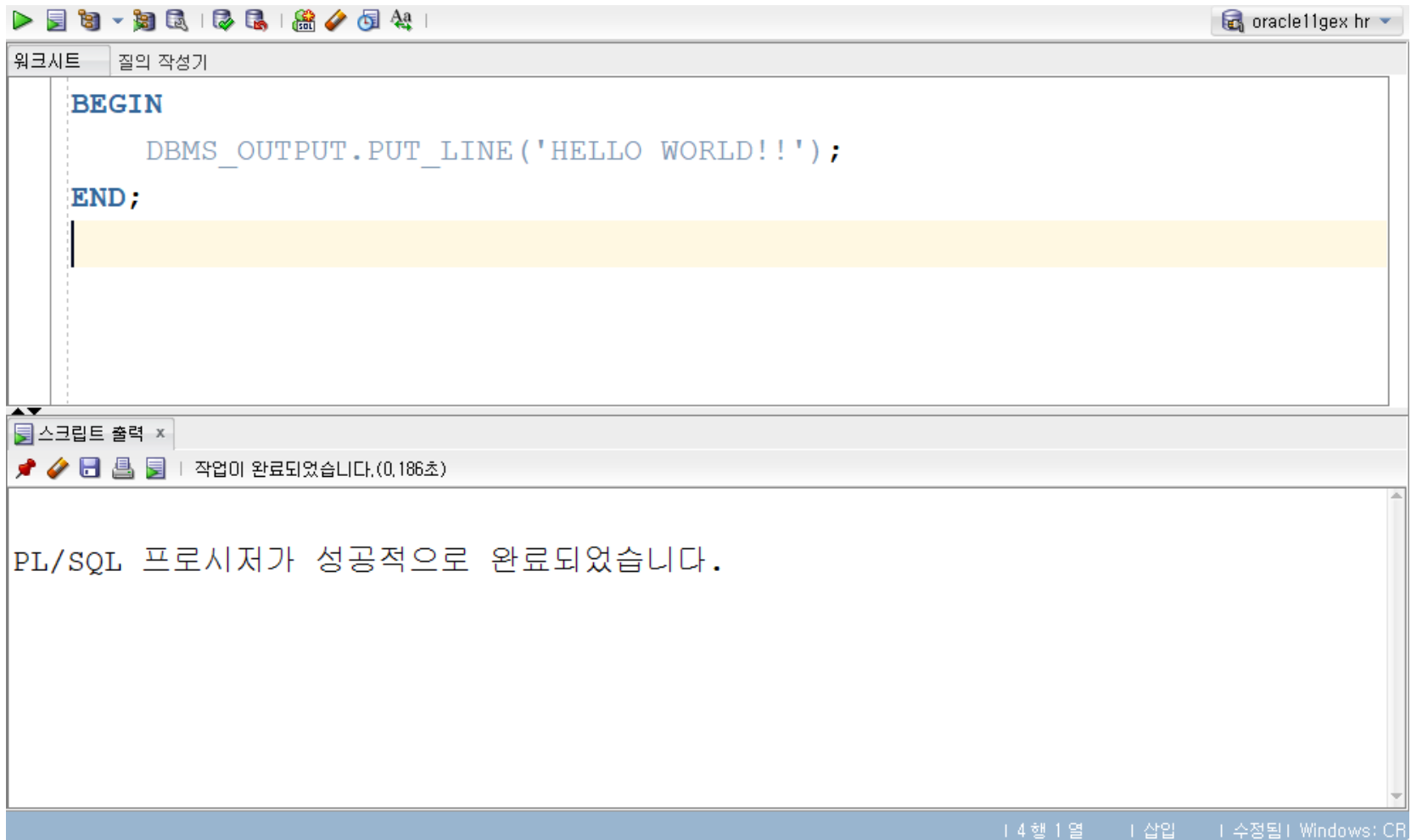
화면 출력을 위해서는 PUT\_LINE이란 프로시저를 이용합니다.  
PUT\_LINE은 오라클이 제공해주는 프로시저로  
DBMS\_OUTPUT 패키지에 묶여 있습니다.  
따라서 DBMS\_OUTPUT.PUT\_LINE과 같이 사용해야 합니다.

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



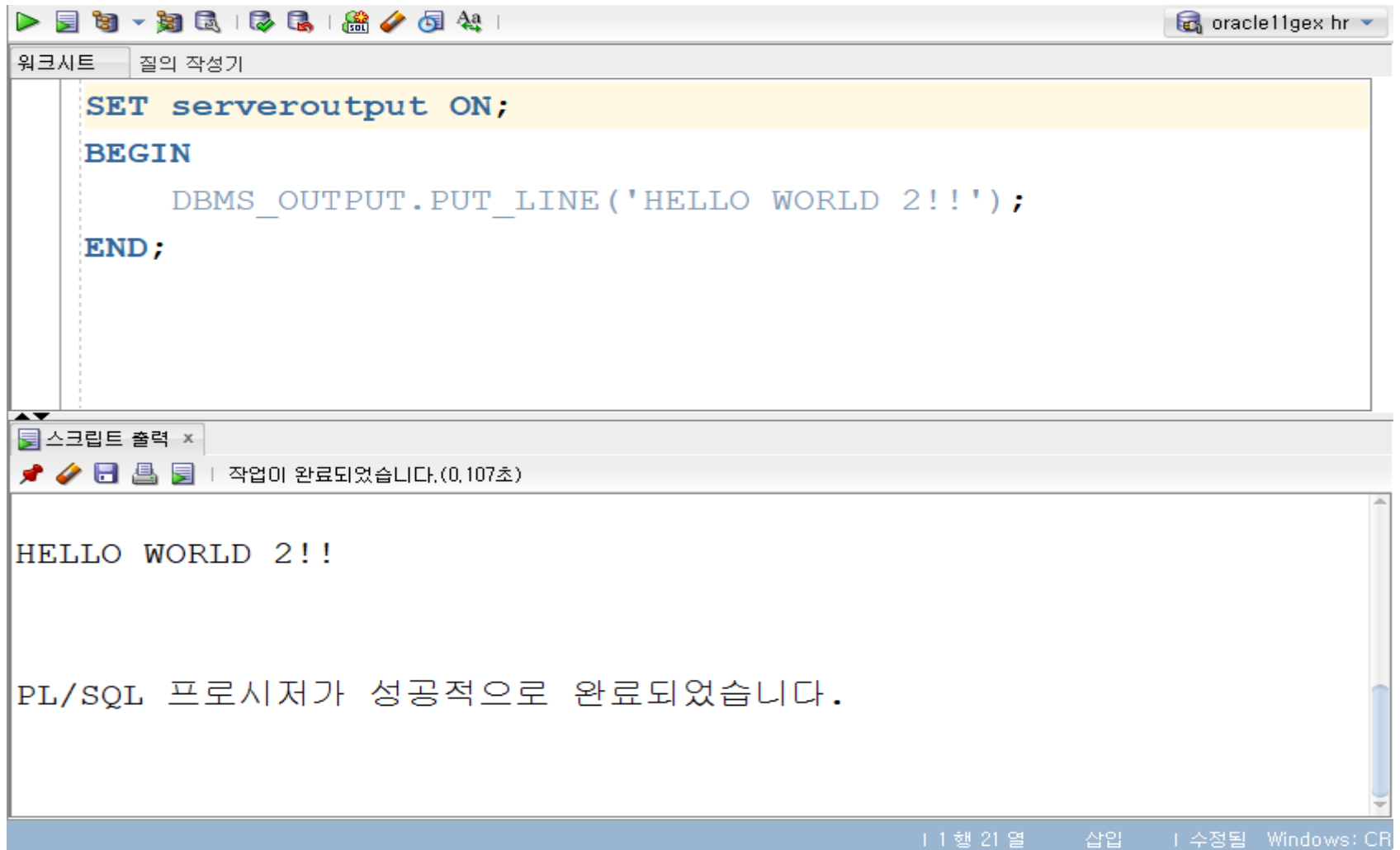
# 1. PL/SQL 구조

- 실습순서1



# 1. PL/SQL 구조

## • 실습순서2



The screenshot shows the Oracle SQL Developer interface. The main editor window contains the following PL/SQL code:

```
SET serveroutput ON;  
BEGIN  
    DBMS_OUTPUT.PUT_LINE('HELLO WORLD 2!!');  
END;
```

Below the editor, the 'Script Output' window is open, displaying the results of the script execution:

```
HELLO WORLD 2!!  
  
PL/SQL 프로시저가 성공적으로 완료되었습니다.
```

The status bar at the bottom indicates the cursor is at line 1, column 21, and the script was executed successfully.

# 1. PL/SQL 구조

## • 실습순서1

- Data Modeler ▶
- DB 객체 찾기(J)
- DBA(B)
- DBMS 출력(D)**
- Data Miner ▶
- OLAP ▶
- OWA 출력(A)
- REST 데이터 서비스 ▶
- SQL 내역(H) F8
- SSH
- Show Only Editor Ctrl+Shift-Enter
- TimesTen Grid
- 구성요소(N) Ctrl+Shift-P
- 단위 테스트(U)
- 디버거(E) ▶

The screenshot displays the Oracle SQL Developer environment. The main editor window, titled '작업기' (Workspace), contains the following PL/SQL code:

```
BEGIN
    DBMS_OUTPUT.PUT_LINE('HELLO WORLD!!');
END;
```

Below the editor, the '스크립트 출력' (Script Output) window shows the message: 'PL/SQL 프로시저가 성공적으로 완료되었습니다.' (PL/SQL procedure completed successfully.).

At the bottom, the 'DBMS 출력' (DBMS Output) window is visible, showing the output: 'HELLO WORLD!!'. The buffer size is set to 20000.

The status bar at the bottom indicates the cursor is at line 3, column 5, and the window title is 'oracle11gex hr'.

## 02. 변수선언과 대입문

- PL/SQL의 선언부에서는 실행부에서 사용할 변수를 선언합니다. 변수를 선언할 때 변수명 다음에 자료형을 기술해야 합니다.
- PL/SOL에서 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 선언.

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

- PL/SOL에서 변수를 선언할 때 위와 같이 SQL에서 사용하던 자료형과 유사하게 선언하는 것을 **스칼라(SCALAR) 변수**라고 함.
- 숫자를 저장하기 위해서 VEMPNO 변수는 NUMBER로 선언하고 VENAME 변수는 문자를 저장하려면 VARCHAR2를 사용해서 선언

## 02. 변수선언과 대입문

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

*identifier := expression;*

- 선언부 에서 선언한 변수에 값을 할당하기 위해서는 :=를 사용

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

## 02. 변수선언과 대입문

- 실습

### 예: 변수 사용하기

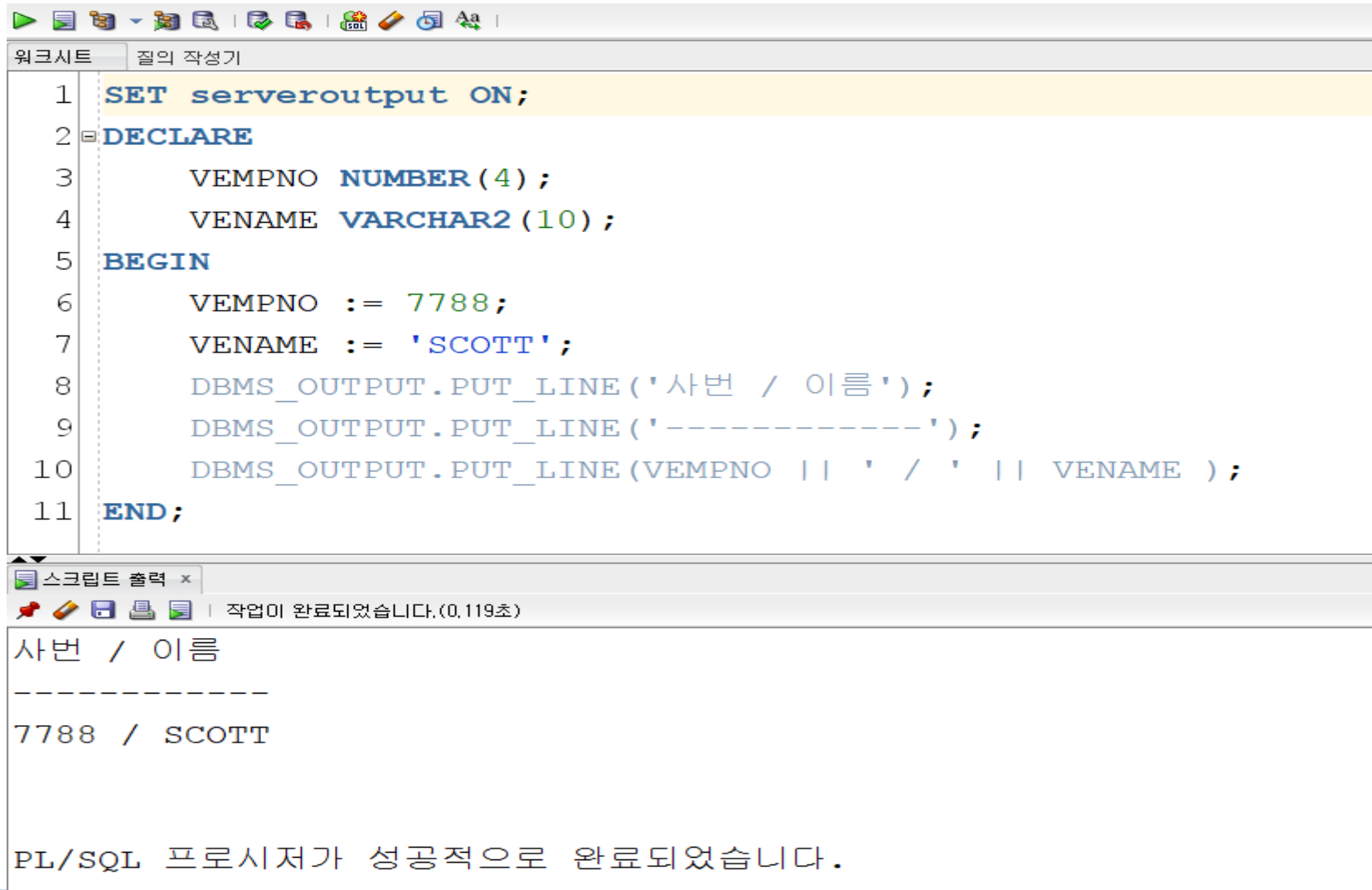
변수의 선언 및 할당을 하고 그 변수 값을 출력해 봅시다.

1. SQL 워크 시트에 다음 코드를 입력하고
2. exam\_01.sql 파일로 저장하시오

```
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;
```

## 02. 변수선언과 대입문

### • 실습순서1



The screenshot displays a PL/SQL development environment. The top window, titled '워크시트' (Worksheet) and '질의 작성기' (Query Builder), contains the following PL/SQL code:

```
1 SET serveroutput ON;
2 DECLARE
3     VEMPNO NUMBER(4);
4     VENAME VARCHAR2(10);
5 BEGIN
6     VEMPNO := 7788;
7     VENAME := 'SCOTT';
8     DBMS_OUTPUT.PUT_LINE('사번 / 이름');
9     DBMS_OUTPUT.PUT_LINE('-----');
10    DBMS_OUTPUT.PUT_LINE(VEMPNO || ' / ' || VENAME);
11 END;
```

The bottom window, titled '스크립트 출력' (Script Output), shows the execution results:

```
사번 / 이름
-----
7788 / SCOTT
```

Below the output, a status message reads: 'PL/SQL 프로시저가 성공적으로 완료되었습니다.'



## 02. 변수선언과 대입문 – 변수종류

- 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;
```

## 02. 변수선언과 대입문 – 변수종류

- 레퍼런스

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

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

## 02. 변수선언과 대입문 – 변수종류

- %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 모르더라도 지정할 수 있습니다.
- SELECT 문장으로 로우를 검색할 때 유리합니다.

## 02. 변수선언과 대입문 – 변수종류

- 예제 exam\_02.sql

```
1 SET serveroutput ON;
2 DECLARE
3     VEMP      EMPLOYEES%ROWTYPE ;
4 BEGIN
5     SELECT EMPLOYEE_ID, FIRST_NAME, HIRE_DATE
6     INTO VEMP.EMPLOYEE_ID, VEMP.FIRST_NAME, VEMP.HIRE_DATE
7     FROM EMPLOYEES
8     WHERE EMPLOYEE_ID = 102;
9
10    DBMS_OUTPUT.PUT_LINE( '사원번호 : ' || VEMP.EMPLOYEE_ID );
11    DBMS_OUTPUT.PUT_LINE( '사원이름 : ' || VEMP.FIRST_NAME );
12    DBMS_OUTPUT.PUT_LINE( '입사일 : ' || VEMP.HIRE_DATE );
13 END;
```

## 02. 변수선언과 대입문 – PLSQL에서 SELECT문

- 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.

## 02. 변수선언과 대입문 – PLSQL 에서 SELECT문

- **SELECT** *select\_list*
- **INTO** {*variable\_name1*[,*variable\_name2*,...] / *record\_name*}
- **FROM** *table\_name*
- **WHERE** *condition*;

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

## 02. 변수선언과 대입문 – PLSQL 에서 SELECT문

```
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.

## 02. 변수선언과 대입문 – PLSQL에서 SELECT문

- 실습

### 예: 사번과 이름 검색하기

PL/SQL의 SELECT 문으로 EMPLOYEES 테이블에서 사원번호와 이름을 조회하시오

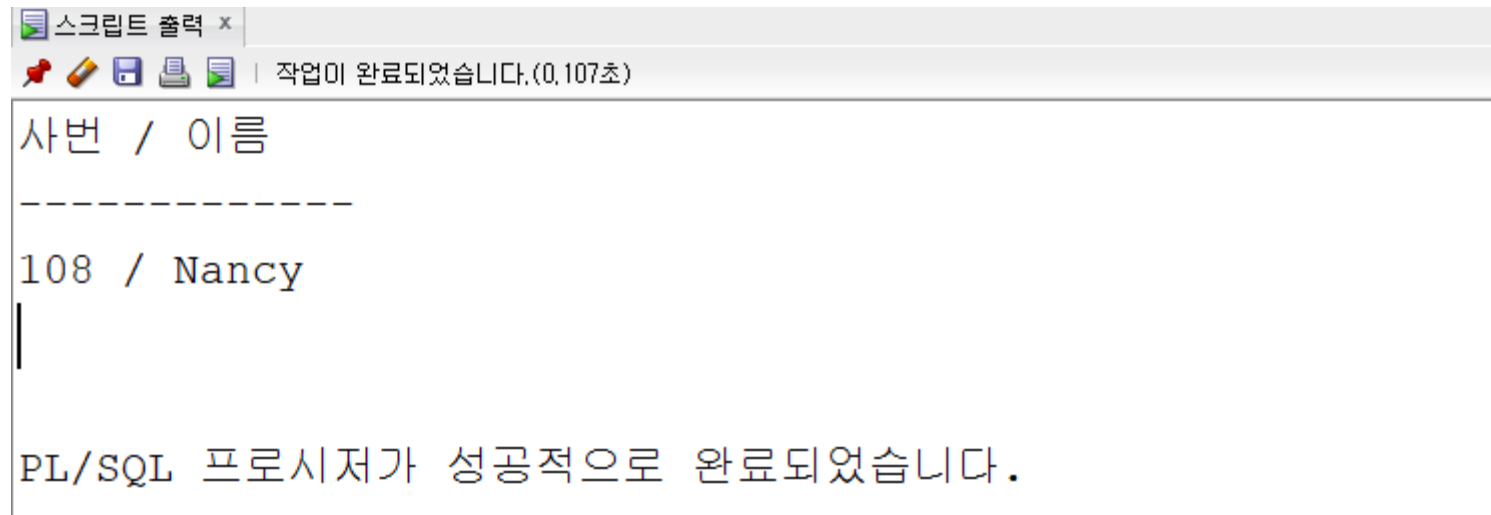
1. SQL 워크 시트에 다음 코드를 입력하고
2. exam\_03.sql 파일로 저장하시오

```
1 SET SERVEROUT ON
2 DECLARE
3     VEMPNO EMPLOYEES.EMPLOYEE_ID%TYPE;
4     VENAME EMPLOYEES.FIRST_NAME%TYPE;
5 BEGIN
6     DBMS_OUTPUT.PUT_LINE('사번 / 이름');
7     DBMS_OUTPUT.PUT_LINE('-----');
8
9     SELECT EMPLOYEE_ID, FIRST_NAME
10    INTO VEMPNO, VENAME FROM EMPLOYEES
11   WHERE FIRST_NAME='Nancy';
12
13     DBMS_OUTPUT.PUT_LINE(VEMPNO || ' / ' || VENAME);
14 END;
```



## 02. 변수선언과 대입문 – PLSQL 에서 SELECT문

- 실습결과1



The screenshot shows a window titled "스크립트 출력 x" (Script Output x). The status bar indicates "작업이 완료되었습니다.(0,107초)" (Work completed. (0,107 seconds)). The output text is as follows:

```
사번 / 이름
-----
108 / Nancy
|

PL/SQL 프로시저가 성공적으로 완료되었습니다.
```

## 02. 변수선언과 대입문 – 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.

## 02. 변수선언과 대입문 – 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.

## 02. 변수선언과 대입문 – FOR LOOP문

- 실습

예: 반복문으로 1~5 출력

다음은 FOR LOOP 문으로 1부터 5까지 출력하시오

1. SQL 워크 시트에 다음 코드를 입력하고
2. exam\_04.sql 파일로 저장하시오

```
1 SET SERVEROUTPUT ON
2 DECLARE
3 BEGIN
4 FOR N IN 1..5 LOOP
5 DBMS_OUTPUT.PUT_LINE( N );
6 END LOOP;
7 END;
```

## 02. 변수선언과 대입문 – IF THEN END IF문

- IF문은 조건값이 참이면 해당 조건의 처리문장을 실행함

```
IF 조건 THEN  
처리문  
ELSIF 조건2 THEN  
처리문  
.....  
ELSE  
처리문  
END IF;
```

### 주의사항

- THEN 다음에 처리할 문장
- ELSE IF 가 아니라 ELSIF
- END IF 로 끝낸다

## 02. 변수선언과 대입문 – IF THEN END IF문

### • 실습

#### 예: 점수 비교하여 학점 출력

SCORE 변수값을 비교하여 학점을 출력 하시오

1. 90점 이상 A등급, 80점 이상 B등급, 기타는 C등급
2. SQL 워크 시트에 다음 코드를 입력하고
3. exam\_05.sql 파일로 저장하시오

```
1 SET SERVEROUTPUT ON;
2 DECLARE
3     SCORE NUMBER :=80;
4 BEGIN
5     IF SCORE >= 90 THEN
6         DBMS_OUTPUT.PUT_LINE('A등급');
7     ELIF SCORE >=80 THEN
8         DBMS_OUTPUT.PUT_LINE('B등급');
9     ELSE
10        DBMS_OUTPUT.PUT_LINE('C등급');
11    END IF;
12 END;
```

## 02. 변수선언과 대입문 – PLSQL 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	Column
.....	.....
1	SMITH
2	ALLEN
3	WARD
.....	.....
BINARY_INTEGER	스칼라

## 02. 변수선언과 대입문 – PLSQL 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



## 02. 변수선언과 대입문 – PLSQL 테이블

- 실습

### 예: 사번과 이름 검색하기

TABLE 변수를 사용하여 EMP 테이블에서 이름과 업무를 출력하시오

1. SQL 워크 시트에 다음 코드를 입력하고
2. exam\_05.sql 파일로 저장하시오

```
1 SET SERVEROUTPUT ON
2 DECLARE
3 -- 테이블 타입을 정의
4 TYPE ENAME_TABLE_TYPE IS TABLE OF EMPLOYEES.FIRST_NAME%TYPE
5 INDEX BY BINARY_INTEGER;
6 TYPE JOB_TABLE_TYPE IS TABLE OF EMPLOYEES.JOB_ID%TYPE
7 INDEX BY BINARY_INTEGER;
8
9 -- 테이블 타입으로 변수 선언
10 ENAME_TABLE ENAME_TABLE_TYPE;
11 JOB_TABLE JOB_TABLE_TYPE;
12
13 I BINARY_INTEGER := 0;
```

## 02. 변수선언과 대입문 – PLSQL 테이블

- 실습

예: 사번과 이름 검색하기(계속)

```
15 BEGIN
16     -- EMP 테이블에서 사원이름과 직급을 얻어옴
17     FOR K IN (SELECT FIRST_NAME, JOB_ID FROM EMPLOYEES) LOOP
18         I := I + 1;                --인덱스 증가
19         ENAME_TABLE(I) := K.FIRST_NAME; --사원이름과
20         JOB_TABLE(I) := K.JOB_ID;      --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;
```

## 02. 변수선언과 대입문 – 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.

## 02. 변수선언과 대입문 – RECORD TYPE

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

`Record_name.field_name`