```
##########################################################################
## << PLSQL 코스 소개 >>                                                 ##
##########################################################################
```

(참고) PL/SQL 학습 시에 참고할 Reference들

1. 참고할 수 있는 Web 사이트 주소입니다.

   http://asktom.oracle.com/

   http://asktheoracle.net/

   http://www.oracle.com/technetwork/index.html
   (미국 OTN 주소)

2. 참고할 Oracle®Database 11g Release 2(11.2) 메뉴얼.

[[ 필수 메뉴얼 ]]

- PL/SQL User's Guide and Reference

[[ 참고 메뉴얼 ]]

- SecureFiles and Large Objects Developer's Guide
  11g Release 2 (11.2)
- PL/SQL Packages and Types Reference
- Reference
- Administrator's Guide
- SQL*Plus® User’s Guide and Reference

<< 과정을 위한 실습환경 구성 >>

실습을 원활하게 하기 위하여 hr 스키마의
테이블을 서브쿼리를 이용하여 새로 생성 후에 실습을 바랍니다.

(주의) 교재에 나오는 "테이블명"을
"테이블명2" 로 고쳐서 실습을 부탁드립니다.

1> SQL*Developer를 이용하여 hr 계정으로 데이터베이스에 원격접속합니다.

2> 다음의 DDL 문장을 실행합니다.

2-1> 이 전 실습에서 사용한 동일한 이름의 테이블을 삭제합니다.

[참고] 아래의 문장실행 시에 테이블이 없는 경우에는
아래처럼 오류가 발생됩니다.

```
오류 보고 -
ORA-00942: 테이블 또는 뷰가 존재하지 않습니다
00942. 00000 -  "table or view does not exist"
*Cause:
*Action:
```

아래의 테이블들을 생성한 적이 없으면,
[2-1] 부분의 문장들을 실행하지 말고
[2-2] 부분을 수행합니다.

```
SQL> DROP TABLE HR.employees2 PURGE ;

SQL> DROP TABLE HR.jobs2 PURGE ;

SQL> DROP TABLE HR.job_history2 PURGE ;

SQL> DROP TABLE HR.departments2 PURGE ;

SQL> DROP TABLE HR.locations2 PURGE ;

SQL> DROP TABLE HR.COUNTRIES2 PURGE ;
```

2-2> 아래의 DDL 문장을 실행하여 테이블을 생성합니다.

[참고] 아래의 문장들을 실행했을 때,
'Table 테이블이름이(가) 생성되었습니다.' 라는
메시지가 표시되어야 정상적으로 실습을 위한
테이블이 생성된 것입니다.

```
SQL> CREATE TABLE hr.employees2
     TABLESPACE example
     AS
       SELECT *
       FROM hr.employees ;


SQL> CREATE TABLE hr.departments2
     TABLESPACE example
     AS
       SELECT *
       FROM hr.departments ;

Table created.

SQL>
SQL> CREATE TABLE hr.locations2
     TABLESPACE example
     AS
       SELECT *
       FROM hr.locations ;

Table created.

SQL>
SQL> CREATE TABLE hr.COUNTRIES2
     TABLESPACE example
     AS
       SELECT *
       FROM hr.COUNTRIES ;

Table created.

SQL>
SQL> CREATE TABLE hr.jobs2
     TABLESPACE example
     AS
       SELECT *
       FROM hr.jobs ;
```

```
Table created.

SQL>
SQL> CREATE TABLE hr.job_history2
     TABLESPACE example
     AS
       SELECT *
       FROM hr.job_history ;

Table created.

SQL>
```

==========

<< Less 01 : 오라클 PL/SQL 언어 소개 >>

==========

중요 학습 내용:

- PL/SQL Block 구조를 이해합니다.
- 오라클 사가 제공하는 오라클 데이터베이스 서버의
  내장 PL/SQL packages 중에
  DBMS_OUTPUT.PUT_LINE 프로시저 사용법 연습

• 본 문서의 실습은 별도의 언급이 없는 한, SQL*Developer를 이용하여
  hr 계정으로 데이터베이스에 원격하여 수행합니다.

<< SQL 과 PL/SQL 의 차이 >>

- SQL : 데이터베이스 서버에 요구하여
  "데이터를 처리"를 수행 명령어.
  ------------
  |-> "데이터 입력, 수정, 삭제, 조회"
- PL/SQL: C, 또는 Java, ASP, JSP, PHP,..와 기능이 같습니다.

  위의 프로그램 언어(Java, ASP, JSP,..)들을 이용하여
  "데이터를 처리(Process)하는 실행 로직이 구현된 프로그램"을
  작성합니다.

  위의 언어로 작성된 프로그램들은 처리하는 데이터를
  다음처럼 확보합니다.

  - Web Server: 프로그램이 자체적으로 데이터를 가지고 있음.
  - WAS : 처리 대상이 되는 데이터를 데이터베이스 서버로
    부터 가져와서 처리.

  많은 데이터를 집계 처리를 수행하여 집계결과를
  표시하는 Web Page라면, 위와 같은 경우에 데이터가 Client쪽에서
  프로그램 의해 처리됩니다. 즉, 많은 데이터의 전송이 발생됩니다.

  만약, Database 서버에 데이터 처리 로직을 구현한다면,
  많은 데이터의 집계 처리 결과만 Client에 전송하면 됨으로
  처리 효율을 높일 수 있습니다.

  이 때 Oracle 데이터베이스 서버 내에서
  데이터 처리 로직이 구현된 프로그램을 생성하기 위하여
  사용하는 언어가 PL/SQL 입니다.

<< PL/SQL 이란 ? (1-3, 1-4) >>

- 오라클의 Procedural Language extension to SQL. 의 약자 입니다.
- SQL문장에서 변수정의, 조건처리( IF), 반복처리(LOOP, WHILE, FOR)등을 지원하며, 오라클 자체에 내장되어 있는 Procedure Language입니다
- PL/SQL 문은 블록 구조로 되어 있고 오라클 엔진이 자체적으로 PL/SQL 컴파일 엔진을 가지고 있습니다.

<< PL/SQL 사용 시의 잇점 >>

- PL/SQL은 BLOCK 구조로 다수의 SQL 문을 한 번에 ORACLE DB 로 보내서 처리하므로 수행속도를 향상 시킬 수 있습니다.
- PL/SQL의 모든 요소는 하나 또는 두 개 이상의 블록으로 구성하여 모듈화가 가능합니다.
- 보다 강력한 프로그램을 작성하기 위해서 큰 블록안에 소블럭을 위치시킬 수 있습니다.
- Variable, Constant, Cursor, Exception을 정의하고, SQL문장과 Procedural 문장에서 사용합니다.
- 단순, 복잡한 데이터형태의 변수를 선언합니다.
- Exception 처리 루틴을 이용하여 Oracle Server Error를 처리합니다.
- Portability : OS 또는 H/W (Flatform)에 관계 없이 Oracle Server나 Oracle 개발툴이 설치된 시스템에서 같은 모듈을 실행시킬 수 있습니다.

<< PL/SQL 블록 구조(BLOCK-STRUCTURE, 1-9) >>

1) Declarative Section(선언부, Optional):

   - 변수, 상수, CURSOR, USER_DEFINE Exception 선언

2) Executable Section(실행부, Mandatory)

   - SQL, 반복분, 조건문 실행
   - 실행부는 BEGIN으로 시작하고 END; 로 끝납니다.
   - 실행문은 필수적으로 사용되어야 합니다.

3) Exception Handling Section( 예외처리, Optional)

   - 예외에 대한 처리.
   - 일반적으로 오류를 정의하고 처리하는 부분.

<< PL/SQL 프로그램 코드 작성 시 주의점 >>

- PL/SQL 블록내에서는 한 문장 종료 시마다 세미콜론(;) 명시.
- END 뒤에 반드시 ; 을 명시하여 하나의 블록이 끝났음을 선언.

• PL/SQL 블록은 행에 어디서든 / 가 있으면 종결됨.

(참고) PL/SQL 내에서 주석처리

• 단일행 주석 : --

• 여러행 또는 일부 구간주석 : /*
주석
*/

<< PL/SQL 블록의 타입(BLOCK-TYPE) >>

(1) 익명-블록(ANONYMOUS-BLOCK)

• 이름이 없는 블록.

• 실행하기 위해 프로그램 안에서 선언되고,

• 프로그램 실행 시에 PL/SQL 엔진으로 전달되어 실행됨.

(2) 프로시저(PROCEDURE)

• 특정 작업을 수행할 수 있는 이름이 있는 PL/SQL 블록.

• 보통 연속 실행 또는 구현이 복잡한 트랜잭션을
수행하는 PL/SQL블록을 데이터베이스에 저장해놓고
편하게 사용하기 위해 이용.

(3) 함수(FUNCTION)

• 보통 값을 계산하고 결과값을 반환.

• 반드시 반환될 값의 데이터 타입을 RETURN문에 선언해야 함.

<< 익명블록 및 간단한 함수 생성 실습 >>

1> SQL*Developer 에서 DBMS_OUTPUT.put_line 프로시저의
실행 결과를 표시하기 위하여 출력 옵션 설정

```
SQL> show all  --SQL*Developer 의 모든 설정값이 표시됩니다.
               --SERVEROUTPUT 설정은 OFF가 기본값입니다.

SQL> SET SERVEROUTPUT ON --SQL*Developer 의 설정이고 이 설정을 이용해서
SQL>                     DBMS_OUTPUT.put_line 처리 결과가 화면에
                         표시되독록 활성화합니다.
```

2> 익명블록을 작성하여 실행합니다.

```
SQL>
DECLARE --선언부분, 실행부분에서 사용할 변수가 없으면,
        --이부분은 필요없습니다,
    f_name varchar2(30);        --변수선언
    l_name varchar2(15);

BEGIN                           --실행부 시작
```

```
        SELECT first_name INTO f_name     --데이터조회문
        FROM hr.employees
        WHERE employee_id = 100;

        DBMS_OUTPUT.PUT_LINE(
                'The First Name of the Employee is '|| f_name);

        f_name := f_name||'--a' ;

        DBMS_OUTPUT.PUT_LINE(f_name);

END; --실행부 종료
/
The First Name of the Employee is Steven
Steven--a

PL/SQL procedure successfully completed.

SQL>
```

3> 다음의 함수를 작성하여 생성합니다.

```
SQL>
CREATE FUNCTION HR.COMPUTE_TAX(sal in number)
     return number
IS
BEGIN
   if sal < 5000 then
        return sal*0.15 ;
   else return sal*0.33 ;
   end if;
END ;
/

Function created.

SQL>
```

4> 생성된 함수의 사용

```
SQL>
col last_name for a15

SELECT last_name, salary, COMPUTE_TAX(salary) as tax_amount
FROM hr.employees
WHERE department_id < 30 ;

LAST_NAME           SALARY TAX_AMOUNT
--------------- ---------- ----------
Whalen                4400        660
Hartstein            13000       4290
Fay                   6000       1980

SQL>
```