

0. PL/SQL 목차

목 차

<ul style="list-style-type: none">1. PL/SQL<ul style="list-style-type: none">1.1 PL/SQL란 ?1.2 pl/sql 구조1.3 연산자1.4 예제2. 변수<ul style="list-style-type: none">2.1 사용법2.2 변수 타입3. 조합데이터 (복합데이터)<ul style="list-style-type: none">3.1 PL/SQL 레코드3.2 %ROWTYPE3.3 PL/SQL 컬렉션4. DML 문장<ul style="list-style-type: none">4.1 예제5. if문<ul style="list-style-type: none">5.1 if문의 구조5.2 예제6. loop문<ul style="list-style-type: none">6.1 사용법6.2 예제7. while loop문<ul style="list-style-type: none">7.1 예제	<ul style="list-style-type: none">8. for loop문<ul style="list-style-type: none">8.1 예제9. case문 / continue문10. 암시적 커서문 (Implicit Cursor)<ul style="list-style-type: none">10.1 암시적 커서의 속성10.2 예제11. 명시적 커서문 (Cursor)<ul style="list-style-type: none">11.1 사용법11.2 예제12. 예외처리<ul style="list-style-type: none">12.1 사용법12.2 예제13. 함수<ul style="list-style-type: none">13.1 사용법13.2 예제14. 프로시저<ul style="list-style-type: none">14.1 예제
--	---

1. PL/SQL

2018년 4월 27일 금요일 오전 9:53

■ PL/SQL 을 배워야 하는 이유 ?

1. SQL로 반복해서 해야되는 작업을 PL/SQL로 단순하게 처리할 수 있다.
2. 데이터 분석을 위해 DB의 데이터를 R로 로드하지 않고 바로 분석을 할 수 있다. (회귀분석, 의사결정트리 분석)
3. SQL튜닝방법으로 PL/SQL을 사용할 수 있다.

1.1 PL/SQL란 ?

PL/SQL은 Procedure Language의 약자로 SQL과 프로그래밍언어를 혼합해서 구현하는 오라클 프로그램 언어이다.

- PL/SQL은 SQL의 확장으로 사용되는 프로그래밍 기능을 갖춘 언어.
- PL/SQL 프로그래밍 생성자는 비절차적 언어인 SQL을 절차적 언어로 만든다.
- PL/SQL 응용 프로그램은 Oracle 서버가 실행되는 모든 플랫폼이나 운영 체제에서 실행할 수 있다.

PL/SQL:

- "SQL을 확장한 절차적 언어(Procedural Language)"를 나타냅니다.
- 관계형 데이터베이스에서 사용되는 Oracle의 표준 데이터 액세스 언어입니다.
- 프로시저 생성자를 SQL과 완벽하게 통합합니다.



1.1.1 프로시저 생성자 제공

- 변수, 상수, 데이터 타입 제공
- 조건문 및 루프와 같은 제어 구조
- 한 번 작성하면 여러 번 실행 가능 -> 재사용 가능한 프로그램 단위

1.1.2 PL/SQL의 이점

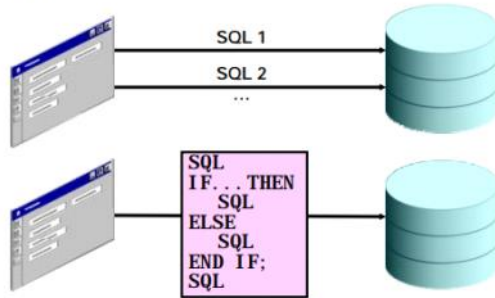
반복작업을 단순화 시킬 수 있어서 성능을 향상 시킨다.

PL/SQL의 이점

- 프로시저 생성자와 SQL의 통합
- 성능 향상

PL/SQL의 이점

- 프로시저 생성자와 SQL의 통합
- 성능 향상



1.2 pl/sql 구조

■ PL/SQL의 기본 block 구조

Declare절 -- 선언 절

= 변수, 상수, 예외 등을 선언한다.

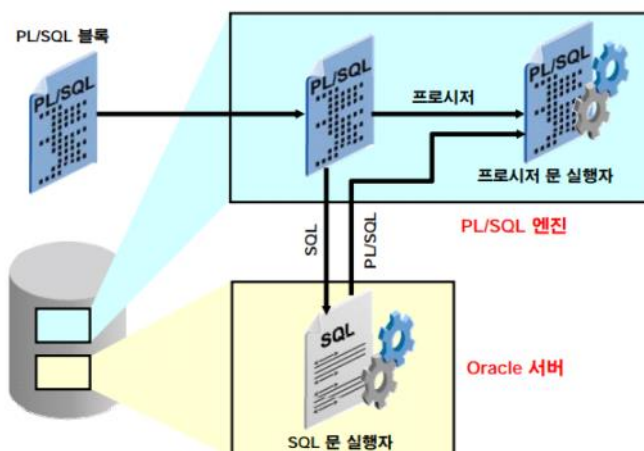
Begin절 -- 실행 절

= 실행할 SQL문, IF문, Loop문을 실행한다.

End; -- 종료 절

/ -- PL/SQL을 종료하겠다. (제일 왼쪽에 붙여줘야 한다)

PL/SQL 런타임 구조



PL/SQL 블록 구조

- **DECLARE (선택 사항)**
 - 변수, 커서, 유저 정의 예외
- **BEGIN (필수)**
 - SQL 문
 - PL/SQL 문
- **EXCEPTION (선택 사항)**
 - 예외 발생 시 수행할 작업
- **END; (필수)**



PL/SQL 블록 구조

이 슬라이드는 기본 PL/SQL 블록을 보여줍니다. PL/SQL 블록은 다음 네 개의 섹션으로 구성됩니다.

- **선언(선택 사항):** 선언 섹션은 DECLARE 키워드로 시작하며 실행 섹션 시작 부분에서 끝납니다.
- **시작(필수):** 실행 섹션은 BEGIN 키워드로 시작합니다. 이 섹션에는 명령문이 하나 이상 있어야 합니다. 그러나 PL/SQL 블록의 실행 섹션에는 PL/SQL 블록이 무제한 포함될 수 있습니다.
- **예외 처리(선택 사항):** 예외 섹션은 실행 섹션 내에 중첩됩니다. 이 섹션은 EXCEPTION 키워드로 시작됩니다.
- **종료(필수):** 모든 PL/SQL 블록은 END 문으로 끝나야 합니다. END는 세미콜론으로 종료됩니다.

PL/SQL 블록 구조 (계속)

PL/SQL 블록에서 DECLARE, BEGIN 및 EXCEPTION 키워드는 세미콜론으로 종료되지 않습니다. 그러나 END 키워드, 모든 SQL 문 및 PL/SQL 문은 세미콜론으로 종료해야 합니다.

섹션	설명	포함
선언(DECLARE)	실행 및 예외 섹션에서 참조하는 모든 변수, 상수, 커서 및 유저 정의 예외 선언을 포함합니다.	선택 사항
실행(BEGIN ... END)	데이터베이스에서 데이터를 검색하는 SQL 문과 블록에서 데이터를 조작하는 PL/SQL 문을 포함합니다.	필수
예외(EXCEPTION)	실행 섹션에서 오류 또는 비정상적인 상황이 발생할 경우에 수행할 조치를 지정합니다.	선택 사항

■ 블록 유형

PL/SQL 프로그램은 하나 이상의 블록으로 구성됩니다. 이러한 블록은 완전히 별개이거나 다른 블록 내에 중첩될 수 있습니다. PL/SQL 프로그램을 구성하는 다음 세 가지 유형의 블록이 있습니다.

1. 프로시저

프로시저는 SQL 및/또는 PL/SQL 문이 포함되어 있는 명명된 객체입니다.

2. 함수

함수는 SQL 및/또는 PL/SQL 문이 포함되어 있는 명명된 객체입니다. 프로시저와 달리 함수는 지정된 데이터 유형의 값을 반환합니다.

3. 익명 블록

익명 블록은 이름이 지정되지 않은 블록이다. 응용 프로그램에서 해당 블록을 실행할 지점에 인라인으로 선언되고 **응용 프로그램이 실행될 때마다 컴파일됩니다**. 이러한 블록은 **데이터베이스에 저장되지 않는다**. 런타임에 **PL/SQL 엔진으로 전달되어 실행됩니다**.

■ 중첩 블록

프로시저가 될 경우 PL/SQL은 명령문을 중첩할 수 있다. 실행문에서 허용되는 범위까지 무제한 블록을 중첩할 수 있다. 여러 업무 요구사항을 지원하기 위해 실행 섹션에 논리적으로 관련된 많은 기능들이 포함된 코드가 있는 경우 실행 섹션을 더 작은 블록으로 나눌 수 있다. 예외 섹션도 중첩 블록을 포함할 수 있다.

중첩 블록 예제

```
DECLARE
  v_outer_variable VARCHAR2(20):=' GLOBAL VARIABLE' ;
BEGIN
  DECLARE
    v_inner_variable VARCHAR2(20):=' LOCAL VARIABLE' ;
  BEGIN
    DBMS_OUTPUT.PUT_LINE(v_inner_variable);
    DBMS_OUTPUT.PUT_LINE(v_outer_variable);
  END;
  DBMS_OUTPUT.PUT_LINE(v_outer_variable);
END;
```

```
anonymous block completed
LOCAL VARIABLE
GLOBAL VARIABLE
GLOBAL VARIABLE
```

■ 코드 주석 처리

- 단일 행 주석은 행의 맨 앞에 두 개의 하이픈(--)을 사용
- 블록 주석은 /* 기호와 */ 기호 사이에 배치

1.3 연산자

PL/SQL의 연산자

- 논리적
- 산술
- 연결
- 연산 순서를 제어하기 위한 괄호

SQL과 동일

- 지수 연산자(**)

PL/SQL의 연산자

표현식 내의 연산은 우선 순위에 따라 특정 순서로 수행됩니다. 다음 표는 높은 우선 순위에서 낮은 우선 순위 순서로 연산의 기본 순서를 보여줍니다.

연산자	연산
**	제곱
+, -	일치, 부정
*, /	곱하기, 나누기
+, -,	더하기, 빼기, 연결
=, <, >, <=, >=, <>, !=, ~=, ^=, IS NULL, LIKE, BETWEEN, IN	비교
NOT	논리 부정
AND	결합
OR	포함

1.4 예제

문제1. 부서번호, 부서번호별 토탈월급을 가로로 출력 하시오. (sum함수와 decode를 이용)

```
SELECT SUM(DECODE(deptno,'10',sal,0))"10",  
       SUM(DECODE(deptno,'20',sal,0)) "20",  
       SUM(DECODE(deptno,'30',sal,0)) "30"  
FROM EMP;
```

	10	20	30
1	8750	10875	9400

문제2. 아래의 결과를 출력 하시오.

```
SELECT JOB,SUM(DECODE(deptno,'10',sal,0))"10",  
       SUM(DECODE(deptno,'20',sal,0)) "20",  
       SUM(DECODE(deptno,'30',sal,0)) "30"  
FROM EMP  
GROUP BY job;
```

	JOB	10	20	30
1	SALESMAN	0	0	5600
2	CLERK	1300	1900	950
3	PRESIDENT	5000	0	0
4	MANAGER	2450	2975	2850
5	ANALYST	0	6000	0

문제3. 아래의 결과를 PL/SQL로 수행 하시오.

	JOB	10	20	30
1	SALESMAN	0	0	5600
2	CLERK	1300	1900	950
3	PRESIDENT	5000	0	0
4	MANAGER	2450	2975	2850
5	ANALYST	0	6000	0

create or replace procedure

get_data(p_x out sys_refcursor)

as

l_query varchar2(400) := 'select deptno ';

begin

for x in (select distinct job from emp order by 1)

loop

l_query := l_query || replace(' sum(decode(job, '\$X', sal)) as \$X ' , '\$X', x.job);

end loop;

l_query := l_query || ' from emp group by deptno ';

open p_x for l_query;

end;

/

variable x refcursor;

exec get_data(:x);

print x;

*약간 다르게 짤다. (sqlplus에서 실행해야 함)

DEPTNO	ANALYST	CLERK	MANAGER	PRESIDENT	SALESMAN
30		950	2850		5600
20	6000	1900	2975		
10		1300	2450	5000	

문제 4. 통신사, 나이, 통신사별 나이별 인원수를 출력 하시오. (통신사 : 가로 출력, 나이 : 세로 출력)

```
SELECT age, SUM(DECODE(telecom,'sk',1))"sk",SUM(DECODE(telecom,'kt',1))"kt",
        SUM(DECODE(telecom,'lg',1))"lg",SUM(DECODE(telecom,'cjh',1))"cjh"
FROM EMP2
GROUP BY age
ORDER BY age ASC;
```

	AGE	sk	kt	lg	cjh
1	24	1	2	(null)	(null)
2	25	2	2	1	(null)
3	26	1	2	2	(null)
4	27	2	3	2	(null)
5	28	2	3	2	1
6	29	(null)	(null)	1	(null)
7	32	(null)	(null)	1	(null)

문제 5. 사원번호를 물어보게하고 사원번호를 입력하면 해당 사원의 월급이 출력되게끔 PL/SQL을 작성 하시오.

1.ed p5.sql 을 치고

2. 메모장이 실행되면 아래 명령어 입력 후 저장

SET serveroutput ON -- serveroutput이 on 되어야 dbms_output 패키지가 작동가능해짐

ACCEPT p_empno PROMPT '사원번호를 입력 하세요' -- 외부에서 사용하는 변수는 p_을 앞에 써준다.
-- 받아드려라 --변수 -- prompt [메시지] = 야기시켜라 [메시지를]

DECLARE

v_sal NUMBER(10); -- 숫자만 담을 수 있는 열자리 변수

BEGIN

SELECT sal INTO v_sal -- 월급(sal) 을 v_sal 변수에 넣겠다. -- PL/SQL에서 SELECT 문은 SELECT 컬러명
inoto

FROM EMP

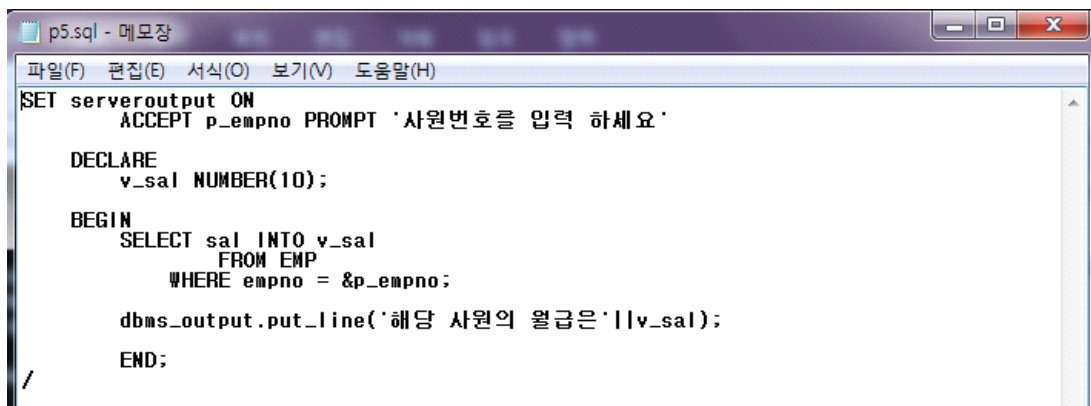
WHERE empno = &p_empno; -- & 치환 변수 : , 내부에서 사용하는 변수는 앞에 v_를 붙여준다.

dbms_output.put_line('해당 사원의 월급은'||v_sal);

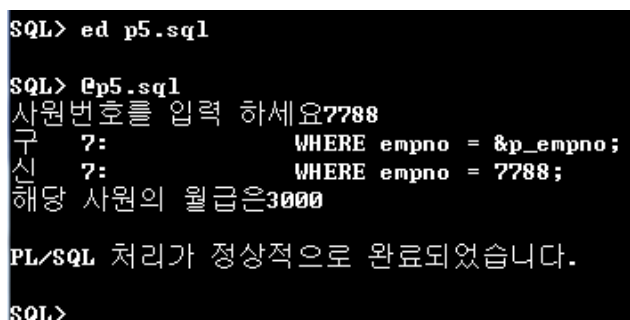
-- 괄호안의 메시지를 출력하는 패키지(패키지 기능을 수행하는 것을 모아논 것)

END;

/



3. @p5.sql 을 치고 사원번호를 입력하면 결과가 출력됨



* p5.sql 메모장이 저장되는 위치는 exit 쳤을 때 나오는 경로


```
SQL> exit
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options
C:\Users\Witwill>
```

문제 6. 사원번호를 물어보게 하고, 사원번호를 입력하면 해당 사원의 직업이 출력되게 하시오.

```
SET serveroutput ON
ACCEPT p_ename PROMPT '사원이름을 입력 하세요'

DECLARE
    v_job varchar2(20);

BEGIN
    SELECT job INTO v_job
    FROM EMP
    WHERE ename = '&p_ename';

    dbms_output.put_line('해당 사원의 직업은'||v_job);

END;
/
```

문제 7.(점심시간) 사원이름을 입력하면 해당 사원의 직업이 출력되는 PL/SQL을 작성하시오.

```
SET serveroutput ON
ACCEPT p_ename PROMPT '사원이름을 입력 하세요'

DECLARE
    v_job varchar2(20);

BEGIN
    SELECT job INTO v_job
    FROM EMP
    WHERE ename = '&p_ename';

    dbms_output.put_line('해당 사원의 직업은'||v_job);

END;
/
```

```
SQL> @p7.sql
사원이름을 입력 하세요SCOTT
구 7: WHERE ename = '&p_ename';
신 7: WHERE ename = 'SCOTT';
해당 사원의 직업은ANALYST
PL/SQL 처리가 정상적으로 완료되었습니다.
```

문제 8. (점심시간) 사원이름을 대소문자 상관없이 입력하면 해당 사원의 직업이 출력되게 하시오.

SET serveroutput ON

ACCEPT p_ename PROMPT '사원이름을 입력 하세요'

DECLARE

v_job varchar2(20);

BEGIN

SELECT job INTO v_job

FROM EMP

WHERE ename = upper('&p_ename');

dbms_output.put_line('해당 사원의 직업은'||v_job);

END;

/

```
SQL> @p8.sql
사원이름을 입력 하세요scott
구 7: WHERE ename = upper('&p_ename');
신 7: WHERE ename = upper('scott');
해당 사원의 직업은ANALYST
PL/SQL 처리가 정상적으로 완료되었습니다.
```

문제 9. 문제 8번을 다시 수행하는데 결과가 출력될 때 아래의 메시지가 안나오게 하시오.

```
SQL> @p8.sql
사원이름을 입력 하세요scott
구 7: WHERE ename = upper('&p_ename');
신 7: WHERE ename = upper('scott');
해당 사원의 직업은ANALYST
PL/SQL 처리가 정상적으로 완료되었습니다.
```

* 코드 맨위에 set verify off 를 쓰면 된다.

```
set verify off
SET serveroutput ON
ACCEPT p_ename PROMPT '사원이름을 입력 하세요'

DECLARE
v_job varchar2(20);

BEGIN
SELECT job INTO v_job
FROM EMP
WHERE ename = upper('&p_ename');

dbms_output.put_line('해당 사원의 직업은'||v_job);

END;
```

```
SQL> @p8.sql
사원이름을 입력 하세요scott
해당 사원의 직업은ANALYST
PL/SQL 처리가 정상적으로 완료되었습니다.
```

문제 10. 이름, 부서위치를 출력 하시오.

```
SELECT e.ename, d.loc
      FROM EMP e, DEPT d
      WHERE e.DEPTNO = d.deptno;
```

	ENAME	LOC
1	KING	NEW YORK
2	BLAKE	CHICAGO
3	CLARK	NEW YORK
4	JONES	DALLAS
5	MARTIN	CHICAGO
6	ALLEN	CHICAGO
7	TURNER	CHICAGO
8	JAMES	CHICAGO
9	WARD	CHICAGO
10	FORD	DALLAS
11	SMITH	DALLAS
12	SCOTT	DALLAS
13	ADAMS	DALLAS
14	MILLER	NEW YORK

문제 11. 이름을 물어보게 하고 이름을 입력하면 해당 사원의 부서위치가 출력되게 하는 PL/SQL을 작성 하시오.

```
SET serveroutput ON
ACCEPT p_ename PROMPT '사원이름을 입력 하세요'

DECLARE
    v_loc varchar2(10);

BEGIN
    SELECT d.loc INTO v_loc
      FROM EMP e, DEPT d
      WHERE e.ename = upper('&p_ename') and e.deptno = d.deptno;

    dbms_output.put_line('해당 사원의 부서위치는'||v_loc);
END;
```

```
SQL> @p11.sql
사원이름을 입력 하세요scott
구 ?:          WHERE e.ename = upper('&p_ename') and e.deptno = d.deptno;
신 ?:          WHERE e.ename = upper('scott') and e.deptno = d.deptno;
해당 사원의 부서위치는DALLAS
PL/SQL 처리가 정상적으로 완료되었습니다.
```

문제 12. 이름, 월급, 급여등급을 출력 하시오.

```
SELECT e.ename, e.sal, s.grade
      FROM EMP e, SALGRADE s
      WHERE e.sal BETWEEN s.LOSAL AND s.HISAL;
```

	ENAME	SAL	GRADE
1	SMITH	800	1
2	JAMES	950	1
3	ADAMS	1100	1
4	WARD	1250	2
5	MARTIN	1250	2
6	MILLER	1300	2
7	TURNER	1500	3
8	ALLEN	1600	3
9	CLARK	2450	4
10	BLAKE	2850	4
11	JONES	2975	4
12	FORD	3000	4
13	SCOTT	3000	4
14	KING	5000	5

문제 13. 이름을 물어보게하고 이름을 입력하면 해당 사원의 월급, 급여등급이 아래와 같이 출력되게 하시오.

이름을 입력하세요 ~ scott

월급은 3000 이고 등급은 4등급입니다.

```
SQL> @p12.sql
이름을 입력하세요 ~scott
구   8:          WHERE e.ename = upper('&p_ename') and e.sal between s.losal and s.hisal;
신   8:          WHERE e.ename = upper('scott') and e.sal between s.losal and s.hisal;
월급은 3000이고 등급은 4입니다.
PL/SQL 처리가 정상적으로 완료되었습니다.
```

SET serveroutput ON

ACCEPT p_ename PROMPT '이름을 입력하세요 ~'

DECLARE

v_grade number(10);

v_sal number(10);

BEGIN

SELECT e.sal, s.grade INTO v_sal, v_grade

FROM EMP e, salgrade s

WHERE e.ename = upper('&p_ename') and e.sal between s.losal and s.hisal;

dbms_output.put_line('월급은 '||v_sal||'이고 등급은 '|| v_grade||'입니다. ');

END;

/

문제 14. 이름, 월급, 월급의 순위를 출력 하시오.

```
SELECT ename, sal, RANK() OVER (ORDER BY sal desc) 순위
FROM EMP;
```

	ENAME	SAL	순위
1	KING	5000	1
2	SCOTT	3000	2
3	FORD	3000	2
4	JONES	2975	4
5	BLAKE	2850	5
6	CLARK	2450	6
7	ALLEN	1600	7
8	TURNER	1500	8
9	MILLER	1300	9
10	WARD	1250	10
11	MARTIN	1250	10
12	ADAMS	1100	12
13	JAMES	950	13
14	SMITH	800	14

문제 15. 순위를 입력하게 하고 해당 사원의 이름,월급이 출력되게 하시오.

```

set serveroutput on
accept p_ename prompt '이름을 입력하시오'

declare
    v_rank number(10);
begin

    select aa into v_rank
    from (
        select e.sal, e.ename, dense_rank() over (order by sal desc) aa
        from emp e
    )
    where ename = upper('&p_ename');

    dbms_output.put_line('이 사원의 월급 순위는 ' || v_rank || ' 입니다.');
```

end;

/

```

SQL> @p15.sql
이름을 입력하시오SCOTT
구 13:         where ename = '&p_ename';
신 13:         where ename = 'SCOTT';
이 사원의 월급 순위는 2 입니다.

PL/SQL 처리가 정상적으로 완료되었습니다.

SQL> @p15.sql
이름을 입력하시오JAMES
구 13:         where ename = '&p_ename';
신 13:         where ename = 'JAMES';
이 사원의 월급 순위는 11 입니다.

PL/SQL 처리가 정상적으로 완료되었습니다.
```

문제 16. 두 개의 숫자를 각각 물어보게 하고 숫자를 입력하면 합이 출력하게 하시오.

```

set serveroutput on
```

```

accept p_num1 prompt '첫 번째 숫자 입력 : '
accept p_num2 prompt '두 번째 숫자 입력 : '

declare
v_sum number(20);

begin
select &p_num1 + &p_num2 into v_sum
      from dual;

dbms_output.put_line('합 : ' || v_sum);

end;
/

```

```

SQL> @p16.sql
첫 번째 숫자 입력 : 10
두 번째 숫자 입력 : 12
구신 6: select &p_num1 + &p_num2 into v_sum
합    6: select 10 + 12 into v_sum
      : 22

PL/SQL 처리가 정상적으로 완료되었습니다.

```

이렇게 해도 되지만 할당 연산자(:=)를 사용하면 더 간단하다.

```

SQL> @p16_.sql
첫 번째 숫자 입력 : 10
두 번째 숫자 입력 : 11
구신 6:      v_sum := &p_num1 + &p_num2;
합    6:      v_sum := 10 + 11;
      : 21

PL/SQL 처리가 정상적으로 완료되었습니다.

```

```

set serveroutput on

```

```

accept p_num1 prompt '첫 번째 숫자 입력 : '
accept p_num2 prompt '두 번째 숫자 입력 : '

declare
v_sum number(20);

begin
      v_sum := &p_num1 + &p_num2;

dbms_output.put_line('합 : ' || v_sum);

end;
/

```

문제 17. 수학자 가우스가 초등학교 때 알아낸 방법을 사용해서 1 부터 10까지 다 더하면 몇인지 출력 하시오.

1 2 3 4 5 6 7 8 9 10

```
set serveroutput on
```

```
declare
```

```
    num1_v number(20);
```

```
begin
```

```
    select 10*(10+1)/2 into num1_v
    from dual;
```

```
    dbms_output.put_line( num1_v);
```

```
end;
```

```
/
```

```
SQL> @p18
55
PL/SQL 처리가 정상적으로 완료되었습니다.
```

문제 18. 첫 번째 숫자를 물어보게 하고 두 번째 숫자를 물어보게 해서 가우스가 알아낸 방법으로 숫자의 합을 출력하도록 PL/SQL 을 출력 하시오.

```
set serveroutput on
```

```
accept p_num1 prompt '첫 번째 숫자 입력 : '
```

```
accept p_num2 prompt '두 번째 숫자 입력 : '
```

```
declare
```

```
    v_rst number(10);
```

```
begin
```

```
    v_rst := (&p_num1 + &p_num2) * (&p_num2/2);
    dbms_output.put_line(' 합은 '||v_rst || ' 입니다.');
```

```
end;
```

```
/
```

```
SQL> @P17
첫 번째 숫자 입력 : 1
두 번째 숫자 입력 : 10
합은 55 입니다.
PL/SQL 처리가 정상적으로 완료되었습니다.
```

문제 19. log(2,10)이 무엇인지 SQL로 출력 하시오.

```
Select log(2,10)
```

From dual;

	LOG(2,10)
1	3.32192809488736234787031942948939017585

문제 20. LOG의 밑수와 진수를 입력 받아 값을 출력 하시오.

```
set serveroutput on
accept p_num1 prompt 'log의 밑수를 입력 하시오.'
accept p_num2 prompt 'log의 진수를 입력 하시오.'

declare
    v_rst number(20,4);          -- 소수 4자리 까지 출력한다.

begin

    v_rst := log(&p_num1,&p_num2);

    dbms_output.put_line('결과 : '|| v_rst);
end;
/
```

```
SQL> @p19.sql
log의 밑수를 입력 하시오.2
log의 진수를 입력 하시오.10
구   6:          v_rst := log(&p_num1,&p_num2);
신   6:          v_rst := log(2,10);
결과 : 3.3219

PL/SQL 처리가 정상적으로 완료되었습니다.
SQL> ED P19.SQL
```

문제 21. 수학책의 예제에는 $\frac{1}{3}\log_2 125$ 의 식을 PL/SQL로 구현되게 하시오.

```
SQL> @p21.sql
실수를 입력하세요.1/3
밑수를 입력하세요.2
진수를 입력하세요.125
구   7:          v_rst := &p_num1*log(&p_num2,&p_num3);
신   7:          v_rst := 1/3*log(2,125);
결과 : 2.3219281
```

```
set serveroutput on
accept p_num1 prompt '실수를 입력하세요.'
accept p_num2 prompt '밑수를 입력하세요.'
accept p_num3 prompt '진수를 입력하세요.'
```

```
declare

    v_rst number(10,7);

begin
```



```

v_rst := &p_num1*log(&p_num2,&p_num3);
dbms_output.put_line('결과 : ' || v_rst);
end;
/

```

문제 22. 수학책의 문제4번을 PL/SQL로 아래와 같이 구현 하시오.

수를 입력 하시오. : 1

수를 입력 하시오. : log(3,7)

수를 입력 하시오. : log(9,100)

결과 : log(9,100), log(3,7), 1

```

set serveroutput on

```

```

accept p_num1 prompt '수를 입력하시오.';

```

```

accept p_num2 prompt '수를 입력하시오.';

```

```

accept p_num3 prompt '수를 입력하시오.';

```

```

declare

```

```

    v_rst1 number(20,7);

```

```

    v_rst2 number(20,7);

```

```

    v_rst3 number(20,7);

```

```

begin

```

```

    if &p_num1 > &p_num2 then

```

```

        if &p_num1 > &p_num3 then v_rst1 := &p_num1;

```

```

            if &p_num2 > &p_num3 then v_rst2 := &p_num2; v_rst3 := &p_num3;

```

```

            else v_rst2 := &p_num3; v_rst3 := &p_num2;

```

```

        end if;

```

```

        else v_rst1 :=&p_num3; v_rst2 := &p_num1; v_rst3 := &p_num2;

```

```

        end if;

```

```

    end if;

```

```

    if &p_num2 > &p_num1 then

```

```

        if &p_num2 > &p_num3 then v_rst1 := &p_num2;

```

```

            if &p_num1 > &p_num3 then v_rst2 := &p_num1; v_rst3 := &p_num3;

```

```

            else v_rst2 := &p_num3; v_rst3 := &p_num1;

```

```

        end if;

```

```

        else v_rst1 :=&p_num3; v_rst2 := &p_num2; v_rst3 := &p_num1;

```

```

        end if;

```

```

    end if;

```

```

    if &p_num3 > &p_num1 then

```

```

        if &p_num3 > &p_num2 then v_rst1 := &p_num3;

```

```

            if &p_num1 > &p_num2 then v_rst2 := &p_num1; v_rst3 := &p_num2;

```

```

        else v_rst2 := &p_num2; v_rst3 := &p_num1;
        end if;
    else v_rst1 :=&p_num2; v_rst2 := &p_num3; v_rst3 := &p_num1;
    end if;

end if;

dbms_output.put_line(v_rst1 || ' > ' || v_rst2 || ' > ' || v_rst3);
end;
/

```

```

SQL> @p22
수를 입력하시오.1
수를 입력하시오.log(3,7)
수를 입력하시오.log(9,100)
2.0959033 > 1.7712437 > 1

PL/SQL 처리가 정상적으로 완료되었습니다.

```

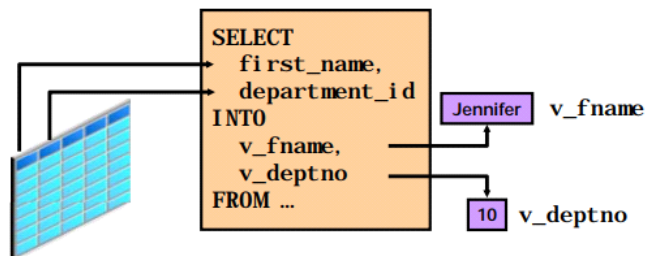
2. 변수

2018년 5월 3일 목요일 오후 6:29

변수 사용

변수는 다음 용도로 사용할 수 있습니다.

- 데이터의 임시 저장 영역
- 저장된 값 조작
- 재사용성



*PL/SQL에서 변수 처리

변수는 다음과 같이 처리됩니다.

- 선언 섹션에서 선언 및 선택적으로 초기화됨
- 실행 섹션에서 사용되고 새 값이 할당됨
- PL/SQL 서브 프로그램에 파라미터로 전달됨
- PL/SQL 서브 프로그램의 출력 결과를 보유하는 데 사용됨

2.1 사용법

PL/SQL 변수 선언 및 초기화

구문:

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

예제:

```
DECLARE
  v_hiredate    DATE;
  v_deptno      NUMBER(2) NOT NULL := 10;
  v_location    VARCHAR2(13) := 'Atlanta';
  c_comm        CONSTANT NUMBER := 1400;
```

■ 변수 유형

• PL/SQL 변수:

- 스칼라 : 스칼라 데이터 유형은 단일 값을 보유
- 참조 : 참조 데이터 유형은 저장 위치를 가리키는 포인터라는 값을 보유
- LOB(대형 객체) : LOB 데이터 유형은 대형 객체의 위치를 지정하는 위치자라는 값을 보유 (ex. 이미지)
- 조합 : 조합 데이터 유형은 PL/SQL 컬렉션과 레코드 변수를 통해 사용할 수 있다.

PL/SQL 컬렉션과 레코드에는 개별 변수로 처리할 수 있는 내부 요소가 포함

• 비PL/SQL 변수: 바인드 변수

■ 변수 이름 규칙

변수 이름에 대한 요구 사항

변수 이름:

- 문자로 시작해야 합니다.
- 문자나 숫자를 포함할 수 있습니다.
- 특수 문자(\$, _, # 등)를 포함할 수 있습니다.
- 30자 이하의 문자만 포함해야 합니다.
- 예약어를 포함하면 안됩니다.

이 과정에서 사용하는 PL/SQL 구조의 이름 지정 규칙

PL/SQL 구조	표기법	예제
변수	<i>v_variable_name</i>	v_rate
상수	<i>c_constant_name</i>	c_rate
서브 프로그램 파라미터	<i>p_parameter_name</i>	p_id
바인드(호스트) 변수	<i>b_bind_name</i>	b_salary
커서	<i>cur_cursor_name</i>	cur_emp
레코드	<i>rec_record_name</i>	rec_emp
유형	<i>type_name_type</i>	ename_table_type
예외	<i>e_exception_name</i>	e_products_invalid
파일 처리	<i>f_file_handle_name</i>	f_file

2.2 변수 타입

■ 기본 스칼라 데이터 유형(1)

- CHAR [(maximum_length)]
- VARCHAR2 (maximum_length)
- NUMBER [(precision, scale)]
- BINARY_INTEGER

- PLS_INTEGER
- BOOLEAN
- BINARY_FLOAT
- BINARY_DOUBLE

기본 스칼라 데이터 유형

데이터 유형	설명
CHAR [(<i>maximum_length</i>)]	고정 길이 문자 데이터의 기본 유형이며 최대 길이는 32,767바이트입니다. <i>maximum_length</i> 를 지정하지 않으면 기본 길이가 1로 설정됩니다.
VARCHAR2 (<i>maximum_length</i>)	가변 길이 문자 데이터의 기본 유형이며 최대 길이는 32,767바이트입니다. VARCHAR2 변수와 상수는 기본 크기가 없습니다.
NUMBER [(<i>precision</i> , <i>scale</i>)]	정밀도 <i>p</i> 와 배율 <i>s</i> 를 가진 숫자입니다. 정밀도 <i>p</i> 의 범위는 1부터 38까지, 배율 <i>s</i> 의 범위는 -84부터 127까지입니다.
BINARY_INTEGER	-2,147,483,647 - 2,147,483,647 사이의 정수에 대한 기본 유형입니다.

PLS_INTEGER	-2,147,483,647 - 2,147,483,647 사이의 부호 있는 정수에 대한 기본 유형입니다. PLS_INTEGER 값은 NUMBER 값보다 저장 공간이 적게 필요하고 연산 속도가 더 빠릅니다. Oracle Database 11g에서 PLS_INTEGER 및 BINARY_INTEGER 데이터 유형은 동일합니다. PLS_INTEGER 및 BINARY_INTEGER 값의 산술 연산은 NUMBER 값보다 빠릅니다.
BOOLEAN	논리적 계산에 사용 가능한 세 가지 값(TRUE, FALSE, NULL) 중 하나를 저장하는 기본 유형입니다.
BINARY_FLOAT	IEEE 754 형식의 부동 소수점 수를 나타냅니다. 값을 저장하기 위해 5바이트가 필요합니다.
BINARY_DOUBLE	IEEE 754 형식의 부동 소수점 수를 나타냅니다. 값을 저장하기 위해 9바이트가 필요합니다.

■ 기본 스칼라 데이터 유형(2) _날짜형

- DATE
- TIMESTAMP
- TIMESTAMP WITH TIME ZONE
- TIMESTAMP WITH LOCAL TIME ZONE
- INTERVAL YEAR TO MONTH
- INTERVAL DAY TO SECOND

데이터 유형	설명
DATE	날짜 및 시간에 대한 기본 유형입니다. DATE 값은 자정 이후 경과한 시간을 초 단위로 포함합니다. 날짜의 범위는 4712 B.C. - 9999 A.D 사이입니다.
TIMESTAMP	TIMESTAMP 데이터 유형은 DATE 데이터 유형을 확장하고 연도, 월, 일, 시, 분, 초 및 소수로 표시되는 초 단위를 저장합니다. 구문은 <code>TIMESTAMP[(precision)]</code> 이며 여기서 선택적 파라미터 <code>precision</code> 은 초 필드의 소수 부분 자릿수를 지정합니다. 자릿수를 지정하려면 0- 9 범위의 정수를 사용해야 합니다. 기본값은 6입니다.
TIMESTAMP WITH TIME ZONE	TIMESTAMP WITH TIME ZONE 데이터 유형은 TIMESTAMP 데이터 유형을 확장하고 시간대 변위를 포함합니다. 시간대 변위는 로컬 시간과 UTC(Coordinated Universal Time—이전의 그리니치 표준시)의 차이(시간과 분)입니다. 구문은 <code>TIMESTAMP[(precision)] WITH TIME ZONE</code> 이며 여기서 선택적 파라미터 <code>precision</code> 은 초 필드의 소수 부분 자릿수를 지정합니다. 자릿수를 지정하려면 0- 9 범위의 정수를 사용해야 합니다. 기본값은 6입니다.

데이터 유형	설명
TIMESTAMP WITH LOCAL TIME ZONE	TIMESTAMP WITH LOCAL TIME 데이터 유형은 TIMESTAMP 데이터 유형을 확장하고 시간대 변위를 포함합니다. 시간대 변위는 로컬 시간과 UTC(Coordinated Universal Time—이전의 그리니치 표준시)의 차이(시간과 분)입니다. 구문은 <code>TIMESTAMP[(precision)] WITH LOCAL TIME</code> 이며 여기서 선택적 파라미터 <code>precision</code> 은 초 필드의 소수 부분 자릿수를 지정합니다. 자릿수를 지정할 때 기호 상수 또는 변수는 사용할 수 없으며 0-9 범위의 정수 리터럴을 사용해야 합니다. 기본값은 6입니다. 이 데이터 유형은 데이터베이스 열에 값을 삽입하면 해당 값이 데이터베이스 시간대로 정규화되고 시간대 변위가 열에 저장되지 않는다는 점에서 TIMESTAMP WITH TIME ZONE과 다릅니다. 값을 검색할 때 Oracle 서버는 로컬 세션 시간대의 값을 반환합니다.
INTERVAL YEAR TO MONTH	INTERVAL YEAR TO MONTH 데이터 유형은 연도와 월의 간격을 저장하거나 조작하는 데 사용됩니다. 구문은 <code>INTERVAL YEAR[(precision)] TO MONTH</code> 이며 여기서 <code>precision</code> 은 연도 필드의 자릿수를 지정합니다. 자릿수를 지정할 때 기호 상수 또는 변수는 사용할 수 없으며 0-4 범위의 정수 리터럴을 사용해야 합니다. 기본값은 2입니다.
INTERVAL DAY TO SECOND	INTERVAL DAY TO SECOND 데이터 유형은 일, 시, 분, 초의 간격을 저장하거나 조작하는 데 사용됩니다. 구문은 <code>INTERVAL DAY[(precision1)] TO SECOND[(precision2)]</code> 이며 여기서 <code>precision1</code> 및 <code>precision2</code> 는 각각 일 필드와 초 필드의 자릿수를 지정합니다. 두 경우 모두 자릿수를 지정할 때 기호 상수 또는 변수는 사용할 수 없으며 0-9 범위의 정수 리터럴을 사용해야 합니다. 기본값은 각각 2와 6입니다.

■ %TYPE

사용법 : 변수 테이블명.컬럼명%TYPE;

- 데이터 유형 불일치나 잘못된 정밀도로 인해 발생하는 오류를 방지할 수 있다.
- 변수의 데이터 유형을 하드 코딩하는 것을 피할 수 있다.
- 열 정의가 변경된 경우 변수 선언을 변경할 필요가 없다

■ LOB 데이터 유형 변수

LOB(대형 객체)는 다량의 데이터를 저장하는 것을 의미한다. 데이터베이스 열은 LOB 범주에 포함될 수 있다.

LOB 범주의 데이터 유형(BLOB, CLOB 등)을 사용하여 데이터베이스 블록 크기에 따라 최대 128TB까지 텍스트, 그래픽 이미지, 비디오 클립, 사운드 웨이브 형식 등의 구조화되지 않은 데이터 블록을 저장할 수 있다.

■ 바인드 변수

바인드 변수는 생성한 후에는 다른 모든 SQL 문 또는 PL/SQL 프로그램에서 이 변수를 참조할 수 있다.

바인드 변수 참조

예제:

```
VARIABLE b_emp_salary NUMBER
BEGIN
  SELECT salary INTO :b_emp_salary
  FROM employees WHERE employee_id = 178;
END;
/
PRINT b_emp_salary
SELECT first_name, last_name
FROM employees
WHERE salary=:b_emp_salary;
```

출력 →

FIRST_NAME	LAST_NAME
Oliver	Tuvault
Sarah	Sewall
Kimberely	Grant

3 rows selected

*바인드 변수와 함께 AUTOPRINT 사용

성공적인 PL/SQL 블록에서 사용되는 바인드 변수를 자동으로 출력하려면 SET AUTOPRINT ON 명령을 사용

바인드 변수와 함께 AUTOPRINT 사용

The screenshot shows a SQL Worksheet with the following code:

```
VARIABLE b_emp_salary NUMBER
SET AUTOPRINT ON
DECLARE
  v_empno NUMBER(6):=&empno;
BEGIN
  SELECT salary INTO :b_emp_salary
  FROM employees WHERE employee_id = v_empno;
END;
/
```

A red box highlights the `SET AUTOPRINT ON` line. A red arrow points from the `&empno` substitution variable in the `DECLARE` section to an "Enter Substitution Variable" dialog box. The dialog box has the label "EMPNO:" and a text input field containing "178". Red arrows indicate the flow from the dialog box to the `PRINT` statement in the code and to the output window.

The output window shows the following results:

```
anonymous block completed
b_emp_salary
-----
7000
```

3. 조합데이터 (복합데이터)

2018년 5월 6일 일요일 오후 11:47

" 스칼라 유형과는 달리 다중 값을 보유할 수 있다. "

■ 조합 데이터 유형을 사용하는 이유

모든 관련 데이터를 단일 단위로 보유할 수 있기 때문에 데이터를 쉽게 액세스 및 수정할 수 있다.

3.1 PL/SQL 레코드

관련은 있지만 서로 다른 유형의 데이터를 논리적 단위로 처리하는데 사용되는 레코드. (C언어의 구조체와 비슷)

---> 서로 다른 데이터 유형의 값을 저장하려는 경우 한번에 하나씩만 저장할 때 사용


Ex) 사원 세부 정보를 포함할 레코드 정의

-사원번호 : number형

-사원이름 : varchar2형 으로 저장한다.

---> number형, varchar2형이 저장된 집합적 논리 단위가 생성된다. 따라서 데이터 액세스와 조작이 쉬워진다.

PL/SQL 레코드:

TRUE	23-DEC-98	ATLANTA	
------	-----------	---------	---

*레코드에 저장된 데이터는 서로 관련 있는 데이터이다.

■ 예제

```
TYPE emp_record IS RECORD
  (v_empno  NUMBER,
   v_ename  VARCHAR2(30),
   v_hiredate DATE );
emp_rec emp_record ;

BEGIN

DBMS_OUTPUT.ENABLE;

-- 레코드의 사용
SELECT empno, ename, hiredate
INTO emp_rec.v_empno, emp_rec.v_ename, emp_rec.v_hiredate
FROM emp
```



```

WHERE empno = p_empno;

DBMS_OUTPUT.PUT_LINE( '사원번호 : ' || emp_rec.v_empno );
DBMS_OUTPUT.PUT_LINE( '사원이름 : ' || emp_rec.v_ename );
DBMS_OUTPUT.PUT_LINE( '입 사 일 : ' || emp_rec.v_hiredate);

END;
/

```

3.2 %ROWTYPE

- 테이블이나 뷰 내부의 컬럼 데이터형, 크기, 속성등을 그대로 사용 할 수 있다.
- %ROWTYPE 앞에 오는 것은 데이터베이스 테이블 이름이다.
- 지정된 테이블의 구조와 동일한 구조를 갖는 변수를 선언 할 수 있다.
- 데이터베이스 컬럼들의 수나 DATATYPE을 알지 못할 때 편리 하다.
- 테이블의 데이터 컬럼의 DATATYPE이 변경 될 경우 프로그램을 재수정할 필요가 없다.

■ 예제

```

CREATE OR REPLACE PROCEDURE RowType_Test
( p_empno IN emp.empno%TYPE )
IS
-- %ROWTYPE 변수 선언,
-- emp테이블의 속성을 그대로 사용할 수 있다.
v_emp emp%ROWTYPE ;

BEGIN

DBMS_OUTPUT.ENABLE;

-- %ROWTYPE 변수 사용
SELECT empno, ename, hiredate
INTO v_emp.empno, v_emp.ename, v_emp.hiredate
FROM emp
WHERE empno = p_empno;

DBMS_OUTPUT.PUT_LINE( '사원번호 : ' || v_emp.empno );
DBMS_OUTPUT.PUT_LINE( '사원이름 : ' || v_emp.ename );
DBMS_OUTPUT.PUT_LINE( '입 사 일 : ' || v_emp.hiredate );

END;
/

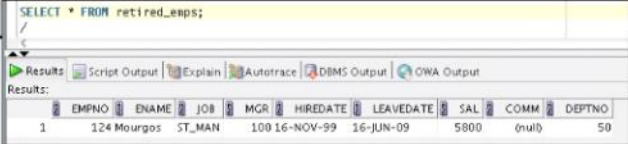
```

또 다른 %ROWTYPE 속성 예제

```

DECLARE
    v_employee_number number := 124;
    v_emp_rec employees%ROWTYPE;
BEGIN
    SELECT * INTO v_emp_rec FROM employees
    WHERE employee_id = v_employee_number;
    INSERT INTO retired_ems(empno, ename, job, mgr,
        hiredate, leavedate, sal, comm, deptno)
    VALUES (v_emp_rec.employee_id, v_emp_rec.last_name,
        v_emp_rec.job_id, v_emp_rec.manager_id,
        v_emp_rec.hire_date, SYSDATE,
        v_emp_rec.salary, v_emp_rec.commission_pct,
        v_emp_rec.department_id);
END;
/

```



EMPNO	ENAME	JOB	MGR	HIREDATE	LEAVEDATE	SAL	COMM	DEPTNO
1	124 Mourgos	ST_MAN	100	16-NOV-99	16-JUN-09	5800	(null)	50

3.3 PL/SQL 컬렉션

동일한 데이터 유형의 값을 저장하려는 경우 사용한다. (PL/SQL 컬렉션은 JAVA의 배열과 비슷하다.)

PL/SQL 컬렉션:

1	SMITH
2	JONES
3	BENNETT
4	KRAMER

└┐

PLS_INTEGER

VARCHAR2

0.3.1 컬렉션 유형

1. 연관 배열
2. 중첩 테이블
3. VARRAY

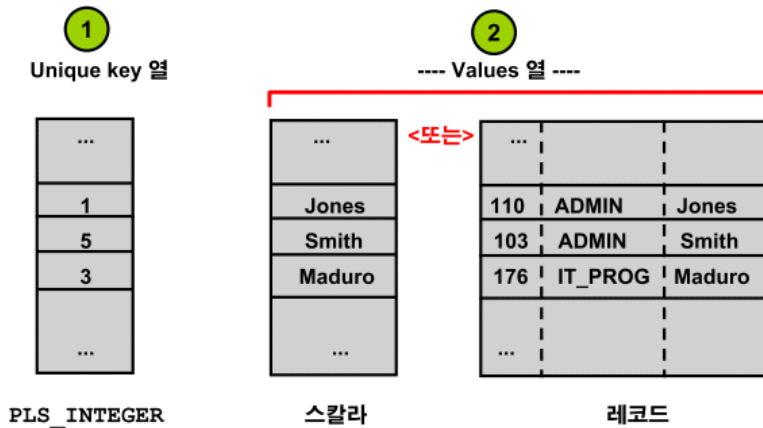
■ 연관 배열 (index by 테이블)

키-값 쌍의 집합이다. Primary key 값을 인덱스로 사용하여 데이터를 저장할 수 있다.

연관 배열은 두 열(키값 + 스칼라 or 레코드) 만 존재한다. 두 열 모두 이름을 지정할 수 없다.

키	값
1	JONES
2	HARDEY
3	MADURO
4	KRAMER

연관 배열 구조



연관 배열 생성 단계

구문:

```

1 TYPE type_name IS TABLE OF
  {column_type | variable%TYPE
  | table.column%TYPE} [NOT NULL]
  | table%ROWTYPE
  | INDEX BY PLS_INTEGER | BINARY_INTEGER
  | VARCHAR2(<size>);
2 identifier type_name;

```

예제:

```

...
TYPE ename_table_type IS TABLE OF
employees.last_name%TYPE
INDEX BY PLS_INTEGER;
...
ename_table ename_table_type;

```

연관 배열 생성 및 액세스

```

...
DECLARE
  TYPE ename_table_type IS TABLE OF
    employees.last_name%TYPE
    INDEX BY PLS_INTEGER;
  TYPE hiredate_table_type IS TABLE OF DATE
    INDEX BY PLS_INTEGER;
  ename_table      ename_table_type;
  hiredate_table   hiredate_table_type;
BEGIN
  ename_table(1)   := 'CAMERON';
  hiredate_table(8) := SYSDATE + 7;
  IF ename_table.EXISTS(1) THEN
    INSERT INTO ...
  ...
END;
/
...

```

anonymous block completed

ENAME	HIREDT
CAMERON	23-JUN-09

 1 rows selected

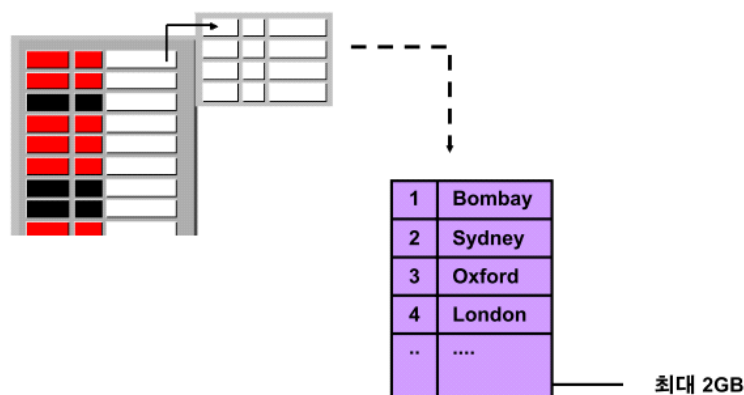
*관련 메소드

메소드	설명
EXISTS(<i>n</i>)	연관 배열에 <i>n</i> 번째 요소가 존재하면 TRUE를 반환합니다.
COUNT	현재 연관 배열에 포함된 요소 수를 반환합니다.
FIRST	<ul style="list-style-type: none"> 연관 배열에서 첫번째(가장 작은) 인덱스 번호를 반환합니다. 연관 배열이 비어 있으면 NULL을 반환합니다.
LAST	<ul style="list-style-type: none"> 연관 배열에서 마지막(가장 큰) 인덱스 번호를 반환합니다. 연관 배열이 비어 있으면 NULL을 반환합니다.
PRIOR(<i>n</i>)	연관 배열에서 인덱스 <i>n</i> 앞에 나오는 인덱스 번호를 반환합니다.
NEXT(<i>n</i>)	연관 배열에서 인덱스 <i>n</i> 뒤에 나오는 인덱스 번호를 반환합니다.
DELETE	<ul style="list-style-type: none"> DELETE 연관 배열에서 모든 요소를 제거합니다. DELETE(<i>n</i>) 연관 배열에서 <i>n</i>번째 요소를 제거합니다. DELETE(<i>m</i>, <i>n</i>) 연관 배열에서 <i>m</i> ... <i>n</i> 범위의 모든 요소를 제거합니다.

■ 중첩 테이블

- 스키마 레벨 테이블에 적합한 데이터 유형이다.
- 중첩 테이블의 크기는 동적으로 증가 가능 (최대 2GB)
- 연관 배열과 달리 키 값이 음수일 수 없다.
- 첫번째 열을 키로 참조하고 중첩 테이블에는 키가 없으며 숫자 열이 있다.

중첩 테이블



예제:

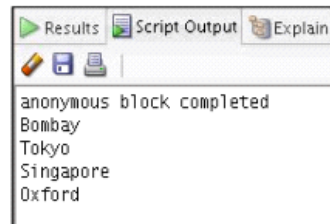
```
TYPE location_type IS TABLE OF locations.city%TYPE;  
offices location_type;
```

중첩 테이블을 초기화하지 않은 경우 자동으로 NULL로 초기화됩니다. 생성자를 사용하여 offices 중첩 테이블을 초기화할 수 있습니다.

```
offices := location_type('Bombay', 'Tokyo','Singapore',  
                          'Oxford');
```

전체 코드 예제와 출력은 다음과 같습니다.

```
SET SERVEROUTPUT ON;  
  
DECLARE  
    TYPE location_type IS TABLE OF locations.city%TYPE;  
    offices location_type;  
    table_count NUMBER;  
BEGIN  
    offices := location_type('Bombay', 'Tokyo','Singapore',  
                             'Oxford');  
    FOR i in 1.. offices.count() LOOP  
        DBMS_OUTPUT.PUT_LINE(offices(i));  
    END LOOP;  
END;  
/
```

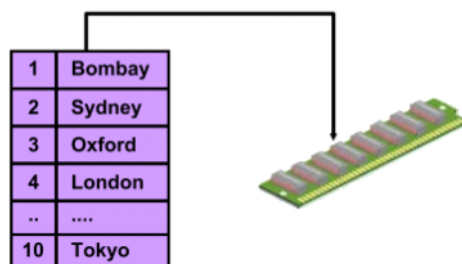


■ VARRAY

가변 크기 배열은 VARRAY 크기(2GB)에 제약이 있다는 점을 제외하고 연관 배열과 비슷하다.

중첩테이블과 차이점은 물리적 저장 모드이다. VARRAY의 크기가 4KB보다 크지 않으면 VARRAY의 요소는 테이블 데이터와 함께 인라인으로 저장된다.

VARRAY



예제:

```
TYPE location_type IS VARRAY(3) OF locations.city%TYPE;  
offices location_type;
```

VARRAY의 크기는 3으로 제한됩니다. 생성자를 사용하여 VARRAY를 초기화할 수 있습니다.

네 개 이상의 요소를 갖도록 VARRAY를 초기화하려고 하면 "Subscript outside of limit"라는 오류 메시지가 표시됩니다.

■ 컬렉션 요약

컬렉션 유형 요약



컬렉션 유형 요약

연관 배열

연관 배열은 키-값 쌍 집합입니다. 이 집합에서 각 키는 고유하며 배열에서 해당하는 값을 찾는 데 사용됩니다. 키는 정수 또는 문자 기반일 수 있습니다. 배열 값은 스칼라 데이터 유형(단일 값) 또는 레코드 데이터 유형(다중 값)일 수 있습니다.

연관 배열은 임시 데이터를 저장하기 위한 것이므로 INSERT 및 SELECT INTO와 같은 SQL 문에서는 사용할 수 없습니다.

중첩 테이블

중첩 테이블은 값 집합을 보유합니다. 달리 말하면 테이블 내에 테이블이 있는 것입니다. 중첩 테이블은 범위가 제한되지 않아 테이블 크기가 동적으로 증가할 수 있습니다. 중첩 테이블은 PL/SQL과 데이터베이스 모두에서 사용할 수 있습니다. PL/SQL에서 중첩 테이블은 크기가 동적으로 증가할 수 있는 1차원 배열과 같습니다.

varray

가변 크기의 배열, 즉 varray도 런타임에 요소 수를 변경할 수 있지만 고정 개수의 요소를 보유하는 동종 요소로 된 컬렉션입니다. varray는 일련 번호를 하위 스크립트로 사용합니다. 상응하는 SQL 유형을 정의하면 varray를 데이터베이스 테이블에 저장할 수 있습니다.

4. DML 문장

2018년 4월 30일 월요일 오전 10:12

■ DML 문의 종류 4가지

- 1.insert
- 2.update
- 3.delete
- 4.merge

■ 데이터 전처리를 위한 PL/SQL 사용법

[source table]						[target table]		
<input type="checkbox"/>	EMPNO	ENAME	JOB	MGR	HIREDATE	번호	부서번호	토탈 월급
1	7839	KING	PRESIDENT	(null)	1981-11-17 오전	1	10	8750
2	7698	BLAKE	MANAGER	7839	1981-05-01 오전	2	20	10875
3	7782	CLARK	MANAGER	7839	1981-05-09 오전	3	30	9400
4	7566	JONES	MANAGER	7839	1981-04-01 오전	4	10	0
5	7654	MARTIN	SALESMAN	7698	1981-09-10 오전			
6	7499	ALLEN	SALESMAN	7698	1981-02-11 오전			
7	7844	TURNER	SALESMAN	7698	1981-08-21 오전			
8	7900	JAMES	CLERK	7698	1981-12-11 오전			

Source table의 데이터를 가공해서 target table로 데이터를 이행

4.1 예제

문제 23. 부서번호, 부서번호별 토탈 월급을 담은 아래의 구조의 테이블을 생성 하시오.

번호	부서번호	토탈월급

```
CREATE TABLE p23
```

```
( 번호 number(20), 부서번호 NUMBER(10), 토탈월급 NUMBER(10));
```

☐ 번호 부서번호 토탈월급

표시할 데이터가 없습니다.

문제 24. 아래의 테이블에 10번 부서번호의 토탈월급을 아래와 같이 번호와 함께 입력되게 insert문을 작성 하시오.

```
CREATE SEQUENCE seq23
  START WITH 1
  INCREMENT BY 1;

INSERT INTO p23
SELECT seq23.nextval, deptno, SUM(Sal)
  FROM EMP
 WHERE deptno = 10
 GROUP BY deptno;
```

-오류 발생 : 이 위치에는 시퀀스를 사용할 수 없다. ---> PL/SQL을 사용해야 한다.

문제 25. 문제 24번을 PL/SQL로 구현 하시오.

```
SET serveroutput on
ACCEPT p_deptno PROMPT '번호를 입력 하시오.'

DECLARE
  v_deptno NUMBER(10);
  v_sumsal NUMBER(10);

BEGIN
  SELECT deptno, SUM(sal) INTO v_deptno, v_sumsal
    FROM EMP
   WHERE deptno = &p_deptno
  GROUP BY deptno;

  INSERT INTO p23
    VALUES(seq23.nextval, v_Deptno, v_sumsal);

  COMMIT;
END;
```

```
/
SQL> @p25
번호를 입력 하시오.10
구  8:      WHERE deptno = &p_deptno
신  8:      WHERE deptno = 10

PL/SQL 처리가 정상적으로 완료되었습니다.

SQL> select * from p23;

   번호   부서번호   토탈월급
-----
      1         10       8750
```

문제 26. 암시적 커서를 이용해서 PL/SQL 내에서 데이터를 입력할 때 입력된 건수가 화면에 출력되게 하시오.


```
SET serveroutput on
ACCEPT p_deptno PROMPT '번호를 입력 하시오.'
```

```
DECLARE
```

```
    v_deptno NUMBER(10);
    v_sumsal NUMBER(10);
```

```
BEGIN
```

```
    SELECT deptno, SUM(sal) INTO v_deptno, v_sumsal
    FROM EMP
    WHERE deptno = &p_deptno
    GROUP BY deptno;
```

```
    INSERT INTO p23
    VALUES(seq23.nextval, v_Deptno, v_sumsal);
```

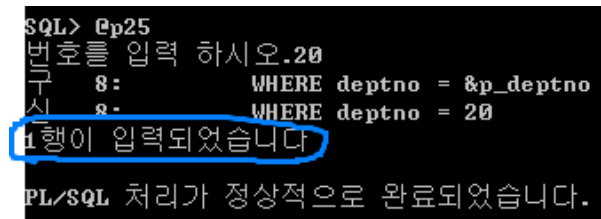
```
    dbms_output.put_line(SQL%rowcount || '행이 입력되었습니다'); -- SERVEROUTPUT 이 ON되어야 실행됨
```

```
COMMIT;
```

```
END;
```

```
/
```

***SQL%rowcount** : 암시적 커서로 앞의 DML 문의 처리된 건수(행)를 출력할 때 사용한다.



```
SQL> @p25
번호를 입력 하시오.20
구      8:      WHERE deptno = &p_deptno
시      8:      WHERE deptno = 20
1행이 입력되었습니다
PL/SQL 처리가 정상적으로 완료되었습니다.
```

문제 27. 통신사, 통신사별 학생 이름을 가로로 출력 하시오.

```
SELECT telecom, listagg(ename,',') WITHin GROUP (ORDER BY telecom)
FROM EMP2
GROUP BY telecom;
```

	TELECOM	LISTAGG(ENAME,',')WITHINGROUP(ORDERBYTELEC
1	gh	이근호
2	kt	김근아,김영토,김지우,방승준,백광훈,송윤호,신
3	lg	김광록,도웅,유혜린,은해찬,이광훈,이상민,이천
4	sk	김건태,김대경,김동윤,김원섭,윤동환,윤진민,이

문제 28. 통신사, 통신사별 학생이름, 통신사별 인원 수가 출력되게 하시오.

```
SELECT telecom, listagg(ename,',') WITHin GROUP (ORDER BY telecom), COUNT(*)
FROM EMP2
GROUP BY telecom;
```

	TELECOM	LISTAGG(ENAME,',')WITHINGROUP(ORDERBYTELEC	COUNT(*)
1	gh	이근호	1

	TELECOM	LISTAGG(ENAME,',')WITHINGROUP(ORDERBYTELEC	COUNT(*)
1	cjh	이근호	1
2	kt	김근아,김영토,김지우,방승준,백광훈,송윤호,신	12
3	lg	김광록,도웅,유혜린,은해찬,이광훈,이상민,이천	9
4	sk	김건태,김대경,김동윤,김원섭,윤동환,윤진민,이	8

문제 29. 아래의 데이터를 담은 테이블을 생성 하시오.

```
CREATE TABLE p29 (
    telecom VARCHAR2(10), NAME VARCHAR2(200), cnt NUMBER(10));

SELECT *
FROM p29;
```

	TELECOM	NAME	CNT
--	---------	------	-----

문제 30. 통신사를 물어보게 하고 통신사를 입력하면 해당 통신사와 통신사의 학생 이름과 인원수가 p29 테이블에 입력되게 하는 PL/SQL을 작성 하시오.

```
set serveroutput on

accept telecom_p prompt '통신사를 입력 하시오.'

declare
    telecom_v varchar2(10);
    name_v varchar2(200);
    cnt_v number(20);

begin

select telecom, listagg (ename,' ') within group (order by telecom),
    count(*) into telecom_v, name_v, cnt_v
from emp2
where telecom = '&telecom_p'
group by telecom;

insert into p29
values(telecom_v, name_v, cnt_v);

dbms_output.put_line(SQL%rowcount || '행이 입력되었습니다.');
```

```
commit;
end;
/
```

```

SQL> @p30.sql
통신사를 입력 하시오.sk
구 12:      where telecom = '&telecom_p'
신 12:      where telecom = 'sk'
1행이 입력되었습니다.

PL/SQL 처리가 정상적으로 완료되었습니다.

SQL> select *
      2  from p29;

TELECOM
-----
NAME
-----
CNT
-----
sk
김건태 ,김대경 ,김동윤 ,김원섭 ,윤동환 ,윤진민 ,이유진 ,한지윤
      8

```

문제 31. (점심시간 문제) 가우스 공식을 적용할 테이블을 생성하고 아래의 세번 째 컬럼이 자동 갱신되는 PL/SQL문을 작성 하시오.

```

SET verify OFF
SET serveroutput ON
ACCEPT p_num PROMPT '번호입력 : '
BEGIN
    update gauss
    set total = (num1 + num2) * (num2 / 2)
    where num = &p_num;

    dbms_output.put_line(SQL%rowcount || '행이 입력 되었습니다.');
```

commit; -- lock이 걸릴 수 있으니 commit을 해주자.

```

END;
/

```

```

SQL> @p31_.sql
번호를 입력해 주세요.5
구 11:      where num = &num_p>
신 11:      where num = 5>
구 12:      where num = &num_p;
신 12:      where num = 5;

PL/SQL 처리가 정상적으로 완료되었습니다.

SQL> select * from gauss;

      NUM      NUM1      NUM2      TOTAL
-----
      1         1      10         55
      2         1      20        210
      3         1      30        465
      4         1      40
      5         1      50        1275
      6         1      60

```

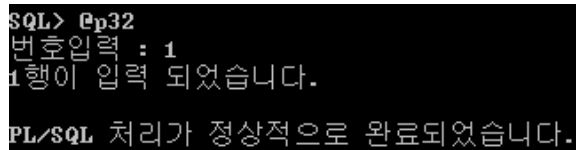
6 개의 행이 선택되었습니다.

문제 32. 문제 31번을 다시 수행하는데 몇 개의 행이 갱신 되었다는 메시지도 같이 출력되게 하시오.

```
SET verify OFF
SET serveroutput ON
ACCEPT p_num PROMPT '번호입력 : '
BEGIN
    update gauss
    set total = (num1 + num2) * (num2 / 2)
    where num = &p_num;

    dbms_output.put_line(SQL%rowcount || '행이 입력 되었습니다.');
```

```
commit;
END;
/
```



```
SQL> @p32
번호입력 : 1
1행이 입력 되었습니다.
PL/SQL 처리가 정상적으로 완료되었습니다.
```

문제 33. 부서번호를 물어보게 하고 부서번호를 입력하면 해당 부서번호의 직원들이 삭제되게 하는 PL/SQL을 작성 하시오.

```
set serveroutput on
accept deptno_p prompt '부서번호를 입력 하시오. : '

begin

    delete from emp
        where deptno = '&deptno_p';
    dbms_output.put_line(SQL%rowcount || '행이 삭제 되었습니다.');
```

```
commit;
END;
/
```

```

SQL> EP33.SQL
부서번호를 입력 하시오. : 10
3행이 삭제 되었습니다.

PL/SQL 처리가 정상적으로 완료되었습니다.

SQL> EP33.SQL
부서번호를 입력 하시오. : 20
5행이 삭제 되었습니다.

PL/SQL 처리가 정상적으로 완료되었습니다.

SQL> EP33.SQL
부서번호를 입력 하시오. : 30
6행이 삭제 되었습니다.

PL/SQL 처리가 정상적으로 완료되었습니다.

```

문제 34. 사원 테이블의 LOC 컬럼을 추가하고 해당 사원의 부서 위치로 값을 갱신 하시오.

```

ALTER TABLE EMP
ADD LOC VARCHAR2(10);

```

```

MERGE INTO EMP e
USING DEPT d
ON (e.deptno = d.deptno)
WHEN matched THEN
UPDATE
SET e.loc = d.loc;

```

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	LOC
1	7839	KING	PRESIDENT	(null)	1981-11-17 오전 12:00:00	5000	(null)	10	NEW YORK
2	7698	BLAKE	MANAGER	7839	1981-05-01 오전 12:00:00	2850	(null)	30	CHICAGO
3	7782	CLARK	MANAGER	7839	1981-05-09 오전 12:00:00	2450	(null)	10	NEW YORK
4	7566	JONES	MANAGER	7839	1981-04-01 오전 12:00:00	2975	(null)	20	DALLAS
5	7654	MARTIN	SALESMAN	7698	1981-09-10 오전 12:00:00	1250	1400	30	CHICAGO
6	7499	ALLEN	SALESMAN	7698	1981-02-11 오전 12:00:00	1600	300	30	CHICAGO
7	7844	TURNER	SALESMAN	7698	1981-08-21 오전 12:00:00	1500	0	30	CHICAGO
8	7900	JAMES	CLERK	7698	1981-12-11 오전 12:00:00	950	(null)	30	CHICAGO
9	7521	WARD	SALESMAN	7698	1981-02-23 오전 12:00:00	1250	500	30	CHICAGO
10	7902	FORD	ANALYST	7566	1981-12-11 오전 12:00:00	3000	(null)	20	DALLAS
11	7369	SMITH	CLERK	7902	1980-12-09 오전 12:00:00	800	(null)	20	DALLAS
12	7788	SCOTT	ANALYST	7566	1982-12-22 오전 12:00:00	3000	(null)	20	DALLAS
13	7876	ADAMS	CLERK	7788	1983-01-15 오전 12:00:00	1100	(null)	20	DALLAS
14	7934	MILLER	CLERK	7782	1982-01-11 오전 12:00:00	1300	(null)	10	NEW YORK

문제 35. 이름을 물어보게하고 이름을 입력하면 해당 사원의 부서위치로 loc 컬럼이 갱신되게 하는 PL/SQL을 작성 하시오.

```

set serveroutput on

```

```

accept ename_p prompt '이름을 입력 하세요 : '

```

```
begin
```

```
update emp e
  set e.loc = (select d.loc
               from dept d
               where d.deptno = e.deptno and e.ename = upper('&ename_p'))
  where e.ename = upper('&ename_p');
```

```
dbms_output.put_line(SQL%rowcount || '행을 입력하였습니다.');
```

```
commit;
```

```
end;
```

```
/
```

```
SQL> Ep35
이름을 입력 하세요 : scott
1행을 입력하였습니다.
```

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	LOC
1	7839	KING	PRESIDENT	(null)	1981-11-17 오전 12:00:00	5000	(null)	10	(null)
2	7698	BLAKE	MANAGER	7839	1981-05-01 오전 12:00:00	2850	(null)	30	(null)
3	7782	CLARK	MANAGER	7839	1981-05-09 오전 12:00:00	2450	(null)	10	(null)
4	7566	JONES	MANAGER	7839	1981-04-01 오전 12:00:00	2975	(null)	20	(null)
5	7654	MARTIN	SALESMAN	7698	1981-09-10 오전 12:00:00	1250	1400	30	(null)
6	7499	ALLEN	SALESMAN	7698	1981-02-11 오전 12:00:00	1600	300	30	(null)
7	7844	TURNER	SALESMAN	7698	1981-08-21 오전 12:00:00	1500	0	30	(null)
8	7900	JAMES	CLERK	7698	1981-12-11 오전 12:00:00	950	(null)	30	(null)
9	7521	WARD	SALESMAN	7698	1981-02-23 오전 12:00:00	1250	500	30	(null)
10	7902	FORD	ANALYST	7566	1981-12-11 오전 12:00:00	3000	(null)	20	(null)
11	7369	SMITH	CLERK	7902	1980-12-09 오전 12:00:00	800	(null)	20	(null)
12	7788	SCOTT	ANALYST	7566	1982-12-22 오전 12:00:00	3000	(null)	20	DALLAS
13	7876	ADAMS	CLERK	7788	1983-01-15 오전 12:00:00	1100	(null)	20	(null)
14	7934	MILLER	CLERK	7782	1982-01-11 오전 12:00:00	1300	(null)	10	(null)

문제 36. 부서 테이블에 `sumsal` 컬럼을 추가하고 해당 부서번호의 토탈 월급으로 값이 갱신되게 하는데 토탈 월급이 7000 이상인 경우만 갱신되게 하고 7000보다 작으면 갱신되지 않도록 PL/SQL을 작성 하시오.

```
ALTER TABLE DEPT
```

```
  ADD sumsal NUMBER(10);
```

```
accept p_deptno prompt '부서번호 : '
```

```
declare
```

```
  v_deptno number(10);
```

```
  v_sumsal number(10);
```

```
begin
```

```
  select deptno, sum(sal) into v_deptno, v_sumsal
```

```
  from emp
```

```
  where deptno = &p_deptno
```

```
  group by deptno;
```

```
  if v_sumsal >= 9000 then
```

```
    update dept
```

```

        set sumsal = v_sumsal
        where deptno = v_deptno;
    commit;
end if;
end;
/

```

```

SQL> @p36.sql
부서번호 : 20
구   8:      where deptno = &p_deptno
신   8:      where deptno = 20

PL/SQL 처리가 정상적으로 완료되었습니다.

SQL> select * from dept;

```

DEPTNO	DNAME	LOC	SUMSAL
10	ACCOUNTING	NEW YORK	
20	RESEARCH	DALLAS	10875
30	SALES	CHICAGO	
40	OPERATIONS	BOSTON	

5. if문

2018년 4월 30일 월요일 오후 3:10

5.1 if문의 구조

IF 문

Syntax:

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

If 조건 then
 실행문;

Elsif 조건 then
 실행문;

Else
 실행문;

End if;

5.2 예제

문제 37. 이름을 물어보게 하고 이름을 입력했을 때 해당 사원이 고소득자인지 저소득자인지 출력되게 하시오.

```
set serveroutput on
```

```
accept ename_p prompt '이름을 입력 하시오.'
```

```
declare
```

```
    sal_v number(10);
```

```
    -- sal_v varchar2(10) := ename_p; -- 바로 할당 가능 , := 할당 연산자
```

```
begin
```

```
    select sal into sal_v
```

```
        from emp
```

```
        where ename = upper('&ename_p'); -- 치환변수 대신 sal_v 로 대체 가능
```



```

if sal_v >= 3000 then
    dbms_output.put_line ('고소득자입니다.');
```

```

elsif sal_v >=2000 then
    dbms_output.put_line('적당합니다.');
```

```

else
    dbms_output.put_line('저소득자입니다.');
```

```

end if;
end;
/
```

```

SQL> @p37
이름을 입력 하시오.scott
구   8:      where ename = upper('&ename_p');
신   8:      where ename = upper('scott');
고소득자입니다.

PL/SQL 처리가 정상적으로 완료되었습니다.

SQL> @p37
이름을 입력 하시오.blake
구   8:      where ename = upper('&ename_p');
신   8:      where ename = upper('blake');
적당합니다.

PL/SQL 처리가 정상적으로 완료되었습니다.

SQL> @p37
이름을 입력 하시오.martin
구   8:      where ename = upper('&ename_p');
신   8:      where ename = upper('martin');
저소득자입니다.

PL/SQL 처리가 정상적으로 완료되었습니다.
```

문제 38. 숫자를 물어보게 하고 숫자를 입력하면 해당 숫자가 짝수 인지 홀수 인지 출력되게 하시오.

```

set verify off
set serveroutput on

accept num_p prompt '숫자를 입력 하시오.'

declare
    num_v number(10) := &num_p;
begin
    if mod(num_v,2) != 0 then
        dbms_output.put_line ('홀수 입니다.');
```

```

    else
        dbms_output.put_line('짝수 입니다.');
```

```

    end if;
```

```
end;  
/
```

```
SQL> Ep38.sql  
숫자를 입력 하시오.2  
짝수 입니다.  
  
PL/SQL 처리가 정상적으로 완료되었습니다.  
  
SQL> Ep38.sql  
숫자를 입력 하시오.5  
홀수 입니다.  
  
PL/SQL 처리가 정상적으로 완료되었습니다.
```

문제 39. 피타고라스 직각 삼각형 공식을 PL/SQL로 구현 하시오.

```
set verify off  
set serveroutput on  
  
accept g_p prompt '가로의 길이를 입력하시오.'  
accept s_p prompt '세로의 길이를 입력하시오.'  
accept b_p prompt '빗변의 길이를 입력하시오.'  
  
begin  
  
    if power(&g_p,2)+power(&s_p,2) = power(&b_p,2) then  
        dbms_output.put_line ('직각삼각형이 맞습니다.');    else  
        dbms_output.put_line('직각삼각형이 아닙니다.');    end if;  
end;  
/
```

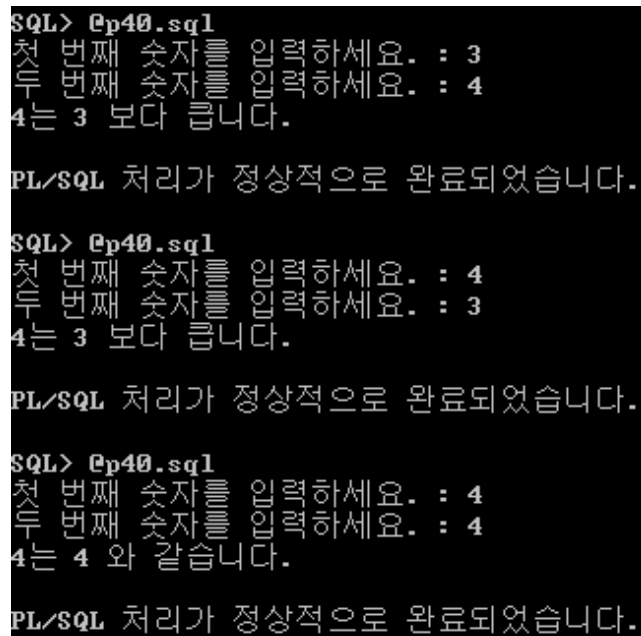
```
SQL> Ep39.sql  
가로의 길이를 입력하시오.3  
세로의 길이를 입력하시오.4  
빗변의 길이를 입력하시오.5  
직각삼각형이 맞습니다.  
  
PL/SQL 처리가 정상적으로 완료되었습니다.  
  
SQL> Ep39.sql  
가로의 길이를 입력하시오.3  
세로의 길이를 입력하시오.4  
빗변의 길이를 입력하시오.6  
직각삼각형이 아닙니다.  
  
PL/SQL 처리가 정상적으로 완료되었습니다.
```

문제 40. 두 개의 숫자를 입력 받아서 아래의 메시지가 출력되게 하시오.

```
set verify off
set serveroutput on

accept num1_p prompt '첫 번째 숫자를 입력하세요. : '
accept num2_p prompt '두 번째 숫자를 입력하세요. : '

declare
    num1_v number(10) := &num1_p;
    num2_v number(10) := &num2_p;
begin
    if num1_v > num2_v then
        dbms_output.put_line( num1_v || '는 ' || num2_v || ' 보다 큼니다. ');
    elsif num1_v < num2_v then
        dbms_output.put_line( num2_v || '는 ' || num1_v || ' 보다 큼니다. ');
    else
        dbms_output.put_line( num2_v || '는 ' || num1_v || ' 와 같습니다. ');
    end if;
end;
/
```



The screenshot shows three separate executions of the SQL script. Each execution starts with the prompt 'SQL> Ep40.sql'. The first execution shows inputs 3 and 4, resulting in the message '4는 3 보다 큼니다.' (4 is greater than 3). The second execution shows inputs 4 and 3, resulting in the message '4는 3 보다 큼니다.' (4 is greater than 3). The third execution shows inputs 4 and 4, resulting in the message '4는 4 와 같습니다.' (4 is equal to 4). Each execution ends with the prompt 'PL/SQL 처리가 정상적으로 완료되었습니다.' (PL/SQL processing completed normally).

```
SQL> Ep40.sql
첫 번째 숫자를 입력하세요. : 3
두 번째 숫자를 입력하세요. : 4
4는 3 보다 큼니다.

PL/SQL 처리가 정상적으로 완료되었습니다.

SQL> Ep40.sql
첫 번째 숫자를 입력하세요. : 4
두 번째 숫자를 입력하세요. : 3
4는 3 보다 큼니다.

PL/SQL 처리가 정상적으로 완료되었습니다.

SQL> Ep40.sql
첫 번째 숫자를 입력하세요. : 4
두 번째 숫자를 입력하세요. : 4
4는 4 와 같습니다.

PL/SQL 처리가 정상적으로 완료되었습니다.
```

6. loop문

2018년 5월 1일 화요일 오전 9:42

■ Loop문을 언제 사용하는가?

특정 실행문을 반복 실행하고자 할 때 사용

■ loop문의 종류

1.basic loop문

2.while loop문

3.for loop문

■ 이중 loop문

Loop

실행문;

Main loop 문을 종료하기 위한 코드;

Loop

실행문;

Inner loop문을 종료하기 위한 코드;

End loop;

End loop;

6.1 사용법

Begin

Loop

실행문;

루프문을 종료시키는 문법; -- 가장 중요, 부재 시 무한루프 (루프문 안에서 위치는 자유)

End loop;

End;

6.2 예제

set serveroutput on

declare

v_count number(10) := 0;

begin

loop

v_count := v_count+1;

dbms_output.put_line (v_count);

```

        exit when v_count = 1000;
    end loop;
end;
/

```

```

981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
PL/SQL 처리가 정상적으로 완료되었습니다.

```

문제 41. basic loop문으로 구구단 2단을 출력 하시오.

2 x 1 = 2

2 x 2 = 4

2 x 3 = 6

2 x 4 = 8

...

set serveroutput on

declare

 v_num number(10) := 1;

begin

 loop

 dbms_output.put_line ('2 x ' || v_num || ' = ' || 2*v_num);

 v_num := v_num+1;

 exit when v_num = 10;

 end loop;

end;

/

```

SQL> @p41.sql
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
PL/SQL 처리가 정상적으로 완료되었습니다.

```

문제 42. 아래와 같이 결과를 출력 하시오. (줄 안바꿔고)

1 2 3 4 5 6 7 8 9 10

set serveroutput on

declare

 v_num number(10) := 1;

begin

 loop

 dbms_output.put(v_num || ' '); -- 괄호 안의 내용을 버퍼에 올려놓겠다.

 v_num := v_num + 1;

 exit when v_num = 11;

 end loop;

 dbms_output.new_line; -- 버퍼에 있는 내용을 출력하겠다.

end;

/

```

SQL> @p42.sql
1 2 3 4 5 6 7 8 9 10
PL/SQL 처리가 정상적으로 완료되었습니다.

```

문제 43. 아래의 결과를 출력 하시오.

2 x 1 = 2 3 x 1 = 3 4 x 1 = 4 5 x 1 = 5

set serveroutput on

declare

 v_num number(10) := 2;

begin

```

        loop
        dbms_output.put( v_num || ' x 1 = ' || v_num || ' ');
        v_num := v_num+1;
        exit when v_num = 10;
        end loop;

        dbms_output.new_line;
    end;
/

```

```

SQL> @p43.sql
2 x 1 = 2  3 x 1 = 3  4 x 1 = 4  5 x 1 = 5  6 x 1 = 6  7 x 1 = 7  8 x 1 = 8  9 x 1 = 9
PL/SQL 처리가 정상적으로 완료되었습니다.

```

문제 44. 아래와 같이 결과를 출력 하시오.

```

2 x 1 = 2  3 x 1 = 3  4 x 1 = 4 ...
2 x 2 = 4  3 x 2 = 6  4 x 2 = 8 ...

```

```

set serveroutput on

```

```

declare
    v_num1 number(10) := 2;
    v_num2 number(10) := 1;

begin

    loop

        loop
            dbms_output.put( v_num1 || ' x ' || v_num2 || ' = ' || v_num1*v_num2 || ' ');
            v_num1 := v_num1+1;
            exit when v_num1 = 10;
        end loop;
        dbms_output.new_line;

        v_num1:=2;
        v_num2 := v_num2+1;
        exit when v_num2 = 10;
    end loop;

end;
/

```



```

v_num number(10) := 11;
v_star varchar2(10) := '★';
v_num2 number(10) := 1;
begin
  loop
    loop
      v_num := v_num -1;
      dbms_output.put(v_star);
      exit when v_num2 = v_num;
    end loop;

    v_num := 11;
    v_num2 := v_num2+1;
    dbms_output.new_line;
    exit when v_num2 = 11;
  end loop;
end;
/

```



+) 또 다른 방법

```

set verify off
set serveroutput on
declare
  v_star varchar2(100) := '★★★★★★★★★★';
begin
  loop
    v_star := substr(v_star, 1, length(v_star)-1 );
    dbms_output.put_line(v_star);
    exit when length(v_star) = 1;
  end loop;
end;
/

```

문제 47. 숫자를 물어보게 하고 숫자를 입력하면 아래와 같이 별이 출력되게 하시오.

```

set serveroutput on

```

```

set verify off

accept num_p prompt '숫자를 입력하시오 : '

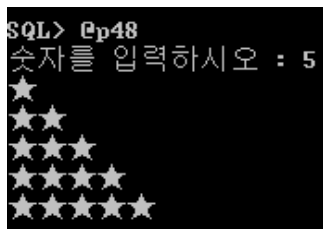
declare
    v_num number(10) := 0;
    v_star varchar2(10) := '★';
    v_num2 number(10) := 1;
begin
    loop
        loop

            v_num := v_num + 1;
            dbms_output.put(v_star);

            exit when v_num2 = v_num;
        end loop;

        v_num := 0;
        v_num2 := v_num2+1;
        dbms_output.new_line;
        exit when v_num2 = &num_p+1;
    end loop;
end;
/

```



문제 48. 숫자를 물어보고 숫자를 입력하면 해당 숫자부터 별을 찍어서 하나씩 깎이게 하시오.

```

set verify off
set serveroutput on

accept num_p prompt '숫자를 입력하시오. : '

declare
    v_num number(10) := &num_p+1;
    v_star varchar2(10) := '★';
    v_num2 number(10) := 1;

begin
    loop
        loop
            v_num := v_num - 1;
            dbms_output.put(v_star);

```

```

        exit when v_num2 = v_num;
    end loop;

    v_num := &num_p+1;
    v_num2 := v_num2+1;
    dbms_output.new_line;
    exit when v_num2 = &num_p+1;
end loop;

end;

/

```



■ 이중 loop문

문제 49. 구구단을 2단과 3단을 출력 하시오. (루프문을 중첩해서)

```

2 x 1 = 2
2 x 2 = 4
...
3 x 1 = 3
3 x 2 = 6
....
3 x 9 = 27

```

```

set serveroutput on
declare
    v_num1 number(20) :=2;
    v_num2 number(20) :=1;
begin
    loop
        dbms_output.put_line(v_num1 || ' x ' || v_num2 || ' = ' || v_num1*v_num2 );
        v_num2 :=v_num2+1;
    exit when v_num2 = 10;
    end loop;

    v_num1:=3;
    v_num2:=1;

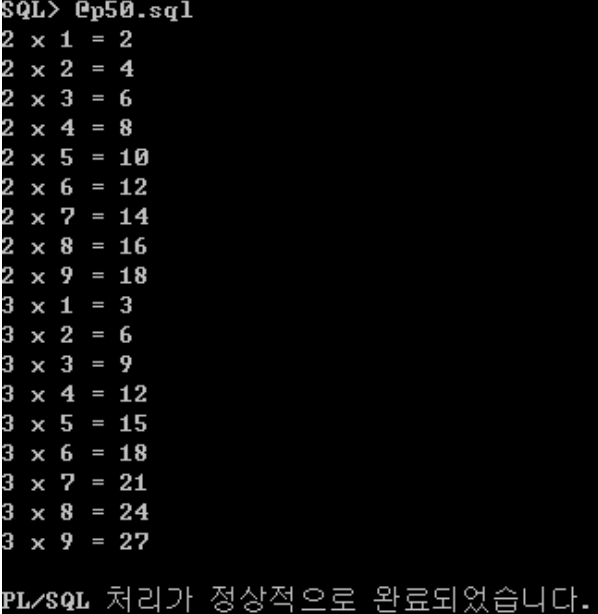
    loop
        dbms_output.put_line(v_num1 || ' x ' || v_num2 || ' = ' || v_num1*v_num2 );
        v_num2 :=v_num2+1;
    
```

```

exit when v_num2 = 10;
end loop;

end;
/

```



```

SQL> @p50.sql
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
PL/SQL 처리가 정상적으로 완료되었습니다.

```

문제 50. 구구단을 2단과 3단을 출력 하시오. (루프문을 중첩해서)

```

2 x 1 = 2
2 x 2 = 4
...
3 x 1 = 3
3 x 2 = 6
....
3 x 9 = 27

```

```

set serveroutput on

```

```

declare

```

```

    v_num1 number(20) :=2;
    v_num2 number(20) :=1;

```

```

begin

```

```

    loop

```

```

        loop

```

```

            dbms_output.put_line(v_num1 || ' x ' || v_num2 || ' = ' || v_num1*v_num2 );

```

```

            v_num2 :=v_num2+1;

```

```

            exit when v_num2 = 10;

```

```

        end loop;

```

```

        v_num1 := v_num1+1;

```

```

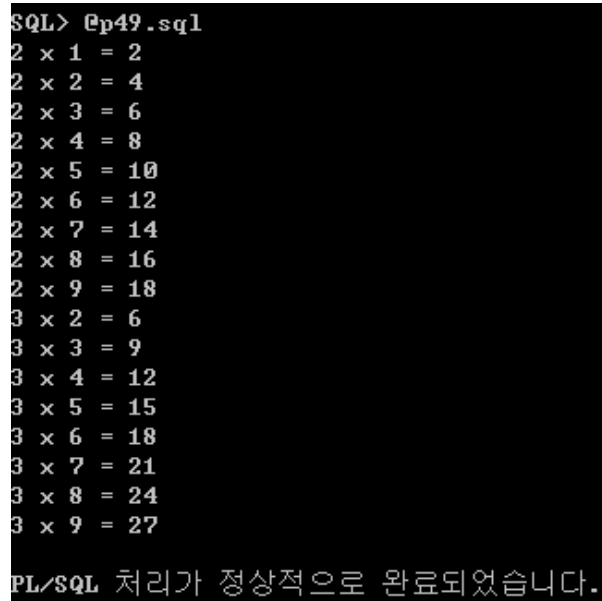
    v_num2 := 2;

```

```

        exit when v_num1 = 4;
    end loop;
end;
/

```



```

SQL> ep49.sql
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
PL/SQL 처리가 정상적으로 완료되었습니다.

```

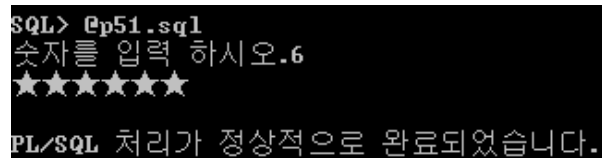
문제 52. 숫자를 물어보게 하고 숫자를 입력하면 해당 숫자만큼 ★가 가로로 출력되게 하시오.

```

set serveroutput on
set verify off
accept num_p prompt '숫자를 입력 하시오.'

declare
    num1_v number(20) := 0;
begin
    loop
        dbms_output.put('★');
        num1_v:= num1_v+1;
        exit when &num_p = num1_v;
    end loop;
    dbms_output.new_line;
end;
/

```



```

SQL> ep51.sql
숫자를 입력 하시오.6
★★★★★★
PL/SQL 처리가 정상적으로 완료되었습니다.

```

문제 53. factorial을 pl/sql로 구현 하시오.

```

set serveroutput on

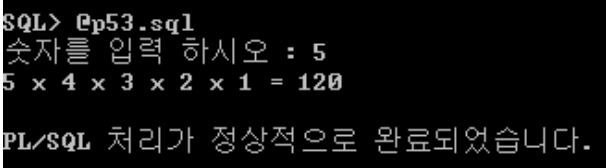
```

```

set verify off
accept num_p prompt '숫자를 입력 하시오 : '
declare
    num2_v number(10) := &num_p;
    rst_v number(20) :=1;
begin
    loop
        dbms_output.put(num2_v || ' x ');
        rst_v := rst_v*num2_v;
        num2_v := num2_v-1;
        exit when 1 = num2_v;
    end loop;

    dbms_output.put(num2_v || ' = ' || rst_v* num2_v );
    dbms_output.new_line;
end;
/

```



```

SQL> @p53.sql
숫자를 입력 하시오 : 5
5 x 4 x 3 x 2 x 1 = 120

PL/SQL 처리가 정상적으로 완료되었습니다.

```

문제 54. power 함수를 pl/sql로 구현 하시오.

```

set serveroutput on
set verify off

accept num1_p prompt '밑수를 입력하세요 : '
accept num2_p prompt '지수를 입력하세요 : '

declare
    num1_v number(20):= 1;
    rst_v number(20):=&num1_p;
begin
    loop
        rst_v := rst_v*&num1_p;
        num1_v := num1_v+1;
        exit when &num2_p = num1_v;
    end loop;

    dbms_output.put_line(rst_v);
end;
/

```

```
SQL> @p54.sql
밑수를 입력하세요 : 2
지수를 입력하세요 : 3
8
PL/SQL 처리가 정상적으로 완료되었습니다.
```

문제 55. mod 함수를 pl/sql로 구현 하시오.

```
set serveroutput on
```

```
set verify off
```

```
accept num1_p prompt '첫 번째 수를 입력 : '
```

```
accept num2_p prompt '두 번째 수를 입력 : '
```

```
declare
```

```
    rst_v number(20) := &num1_p;
```

```
begin
```

```
    loop
```

```
        rst_v := rst_v - &num2_p;
```

```
        exit when rst_v < &num2_p;
```

```
    end loop;
```

```
    dbms_output.put_line(&num1_p || '을 ' || &num2_p || '으로 나눈 나머지 값은 ' || rst_v || '입니다.');
```

```
end;
```

```
/
```

```
SQL> @p55.sql
첫 번째 수를 입력 : 10
두 번째 수를 입력 : 3
10을 3으로 나눈 나머지 값은 1입니다.
PL/SQL 처리가 정상적으로 완료되었습니다.
```

7. while loop문

2018년 5월 2일 수요일 오전 9:49

■ loop문의 종류

- 1.basic loop문
- 2.while loop문
- 3.for loop문

■ while loop문과 loop문 비교

* loop문	* while loop문
<pre>Loop 실행문; Exit when 조건; End loop;</pre>	<pre>While 조건 loop 실행문; End loop;</pre>

■ 이중 while loop문

<pre>*이중 while loop문 While 조건1 loop 실행문1; While 조건2 loop 실행문2; End loop; End loop;</pre>

7.1 예제

문제 56. 구구단 2단을 while loop문으로 구현 하시오.

```
set serveroutput on
```

```
declare
```

```
  num_v number(10) := 1;
```

```
begin
```

```
  while num_v != 10 loop
```



```

        dbms_output.put_line( '2 x ' || num_v || ' = ' || 2*num_v);
        num_v := num_v+1;
    end loop;

end;
/

```

```

SQL> @p56.sql
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
PL/SQL 처리가 정상적으로 완료되었습니다.

```

문제 57 while loop문을 사용해서 아래의 결과를 출력 하시오.

```
1 2 3 4 5 6 7 8 9
```

```
set serveroutput on
```

```
declare
```

```
    v_num number(10) := 1;
```

```
begin
```

```
    while v_num <11 loop
```

```
        dbms_output.put(v_num || ' ');
```

```
        v_num := v_num+1;
```

```
    end loop;
```

```
    dbms_output.new_line;
```

```
end;
```

```
/
```

```

SQL> @p57.sql
1 2 3 4 5 6 7 8 9 10

```

문제 58. 구구단을 2단부터 9단까지 세로로 출력 하시오.

```
set serveroutput on
```

```
declare
```

```
    v_num1 number(10) :=2;
```

```

v_num2 number(10) :=1;

begin

    while v_num2 < 10 loop
        while v_num1 < 10 loop
            dbms_output.put_line(rpad(v_num1 || ' x ' || v_num2 || ' = ' || v_num1*v_num2,15,' '));
            v_num1 := v_num1 +1;
        end loop;
        v_num1 :=2;
        v_num2 := v_num2 +1;
    end loop;

end;
/

```

```

SQL> @p59.sql
2 x 1 = 2
3 x 1 = 3
4 x 1 = 4
5 x 1 = 5
6 x 1 = 6
7 x 1 = 7
8 x 1 = 8
9 x 1 = 9
2 x 2 = 4
3 x 2 = 6
4 x 2 = 8
5 x 2 = 10
6 x 2 = 12
7 x 2 = 14
8 x 2 = 16
9 x 2 = 18
2 x 3 = 6
3 x 3 = 9
4 x 3 = 12
5 x 3 = 15
6 x 3 = 18
7 x 3 = 21
8 x 3 = 24
9 x 3 = 27

```

문제 59. 구구단을 2단부터 9단까지 중첩 while loop문으로 출력 하시오.

```

set serveroutput on

declare

    v_num1 number(10) :=2;
    v_num2 number(10) :=1;

begin

    while v_num2 < 10 loop
        while v_num1 < 10 loop
            dbms_output.put(rpad(v_num1 || ' x ' || v_num2 || ' = ' || v_num1*v_num2,15,' '));
            v_num1 := v_num1 +1;
        end loop;
    end loop;
end;

```

```

        dbms_output.new_line;
        v_num1 :=2;
        v_num2 := v_num2 +1;
    end loop;

end;

/

```

```

SQL> set lines 400
SQL> @58.sql
2 x 1 = 2      3 x 1 = 3      4 x 1 = 4      5 x 1 = 5      6 x 1 = 6      7 x 1 = 7      8 x 1 = 8      9 x 1 = 9
2 x 2 = 4      3 x 2 = 6      4 x 2 = 8      5 x 2 = 10     6 x 2 = 12     7 x 2 = 14     8 x 2 = 16     9 x 2 = 18
2 x 3 = 6      3 x 3 = 9      4 x 3 = 12     5 x 3 = 15     6 x 3 = 18     7 x 3 = 21     8 x 3 = 24     9 x 3 = 27
2 x 4 = 8      3 x 4 = 12     4 x 4 = 16     5 x 4 = 20     6 x 4 = 24     7 x 4 = 28     8 x 4 = 32     9 x 4 = 36
2 x 5 = 10     3 x 5 = 15     4 x 5 = 20     5 x 5 = 25     6 x 5 = 30     7 x 5 = 35     8 x 5 = 40     9 x 5 = 45
2 x 6 = 12     3 x 6 = 18     4 x 6 = 24     5 x 6 = 30     6 x 6 = 36     7 x 6 = 42     8 x 6 = 48     9 x 6 = 54
2 x 7 = 14     3 x 7 = 21     4 x 7 = 28     5 x 7 = 35     6 x 7 = 42     7 x 7 = 49     8 x 7 = 56     9 x 7 = 63
2 x 8 = 16     3 x 8 = 24     4 x 8 = 32     5 x 8 = 40     6 x 8 = 48     7 x 8 = 56     8 x 8 = 64     9 x 8 = 72
2 x 9 = 18     3 x 9 = 27     4 x 9 = 36     5 x 9 = 45     6 x 9 = 54     7 x 9 = 63     8 x 9 = 72     9 x 9 = 81
PL/SQL 처리가 정상적으로 완료되었습니다.

```

문제 60. while loop를 이용해서 아래의 결과를 출력 하시오.

```

set serveroutput on

declare
v_num number(10) := 1;
v_num2 number(10) := 1;

begin
    while v_num2 < 11 loop

        while v_num <= v_num2 loop
            dbms_output.put('★');
            v_num := v_num + 1;
        end loop;

        dbms_output.new_line;
        v_num := 1;
        v_num2 := v_num2 +1;
    end loop;

end;

/

```

```
SQL> @p60.sql

PL/SQL 처리가 정상적으로 완료되었습니다.
```

문제 61. 숫자를 물어보게 하고 숫자를 입력하면 ★를 출력 하시오. (while loop 문으로)

```
set serveroutput on
set verify off

accept p_num prompt ' 숫자 입력 : '

declare
    v_num number(10) := 1;
    v_num2 number(10) := 1;

begin
    while v_num2 <= &p_num loop

        while v_num <= v_num2 loop
            dbms_output.put('★');
            v_num := v_num + 1;
        end loop;

        dbms_output.new_line;
        v_num := 1;
        v_num2 := v_num2 + 1;
    end loop;
end;
```

```
SQL> @p61.sql
 숫자 입력 : 5

PL/SQL 처리가 정상적으로 완료되었습니다.
```

문제 62. 숫자를 물어보게 하고 숫자를 입력하면 ★을 출력 하시오. (거꾸로)

```
set serveroutput on
```

```

set verify off

accept p_num prompt '숫자 입력 : '

declare
    v_num1 number(10) := &p_num;
    v_num2 number(10) := 1;

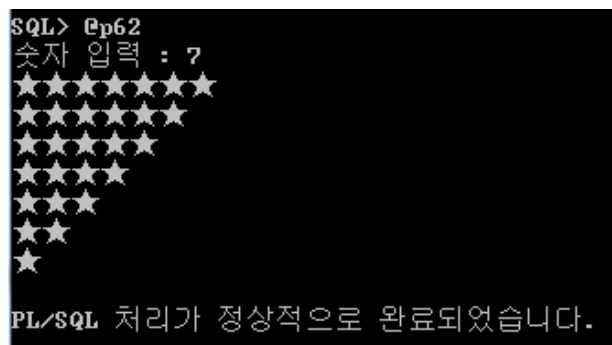
begin
    while &p_num >= v_num2 loop

        v_num1 := &p_num;

        while v_num2 <= v_num1 loop
            dbms_output.put('★');
            v_num1 := v_num1 - 1;
        end loop;

        v_num2 := v_num2 + 1;
        dbms_output.new_line;
    end loop;
end;
/

```



문제 63. while loop문으로 마름모를 출력 하시오.

```

set serveroutput on

declare
    v_star varchar2(100) := '★';

    v_num1 number(10) := 5;
    v_num2 number(10) := 0;

begin
    while v_num1 > 0 loop
        dbms_output.put_line(rpad(chr(7),v_num1,' ')|| v_star);
        v_star := v_star || '★';
    end loop;
end;
/

```

```

        v_num1 := v_num1-1;
end loop;

while v_num2 < 5 loop
    v_num2 := v_num2+1;
    dbms_output.put_line(rpad(chr(7),v_num2+1,' ')||substr(v_star,1,length(v_star)-1-v_num2));

end loop;
end;
/

```



8. for loop문

2018년 5월 2일 수요일 오후 1:57

■ while loop문과 loop문 비교

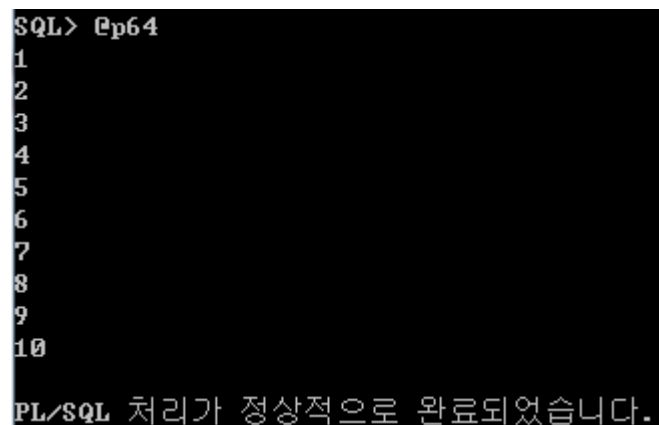
* loop문	* for loop문
<pre>Loop 실행문; Exit when 조건; End loop;</pre>	<pre>For i in 1..10 loop 실행문; End loop;</pre>

8.1 예제

문제 64. for loop문을 사용해서 1~10을 출력 하시오.

```
set verify off
set serveroutput on

begin
    for i in 1..10 loop
        dbms_output.put_line(i);
    end loop;
end;
/
```

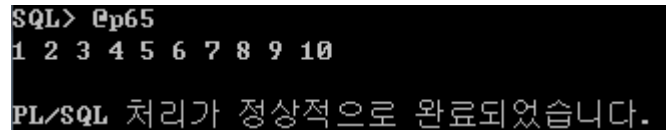


```
SQL> @p64
1
2
3
4
5
6
7
8
9
10
PL/SQL 처리가 정상적으로 완료되었습니다.
```

문제 65. for loop문을 이용해서 1부터 10까지 출력 하시오.(가로로 출력)

```
set serveroutput on
```

```
begin
    for i in 1..10 loop
        dbms_output.put(i || ' ');
    end loop;
    dbms_output.new_line;
end;
/
```

A screenshot of a SQL command window showing the execution of a PL/SQL block. The prompt is 'SQL> @p65'. The output is '1 2 3 4 5 6 7 8 9 10' on one line, followed by a new line and the message 'PL/SQL 처리가 정상적으로 완료되었습니다.' on the next line.

```
SQL> @p65
1 2 3 4 5 6 7 8 9 10
PL/SQL 처리가 정상적으로 완료되었습니다.
```

문제 66. for loop문을 이용해서 구구단 전체를 세로로 출력 하시오.

```
set verify off
set serveroutput on
```

```
begin

    for i in 2..9 loop
        for j in 1..9 loop
            dbms_output.put_line(i||' x '||j||' = '|| i*j);
        end loop;
    end loop;

end;
/
```



```

SQL> Ep66
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28

```

문제 67. 숫자를 물어보게 하고 숫자를 입력하면 ★을 출력하게 하시오.

```

set verify off
set serveroutput on

accept p_num prompt '숫자 입력 : '

begin

    for i in 1..&p_num loop
        for j in 1..i loop
            dbms_output.put('★');
        end loop;
        dbms_output.new_line;
    end loop;

end;
/

```

```

SQL> Ep67
숫자 입력 : 4
★
★★
★★★
★★★★

```

문제 68. 67번을 거꾸로 출력 하시오.

```
set verify off
set serveroutput on
accept p_num prompt '숫자 입력 : '
declare
    v_star varchar2(20) := '';

begin

    for i in 1..&p_num loop
        v_star := v_star||'★';
    end loop;

    for j in 0..&p_num-1 loop
        dbms_output.put_line(substr(v_star,1,length(v_star)-j));
    end loop;

end;

/
```



문제 69. 아래와 같이 가로,세로를 각각 물어보게 하고 ★ 모양이 출력되게 하시오.

```
set verify off
set serveroutput on

accept p_num1 prompt '가로 입력 : '
accept p_num2 prompt '세로 입력 : '

declare
    v_star varchar2(20) := '';

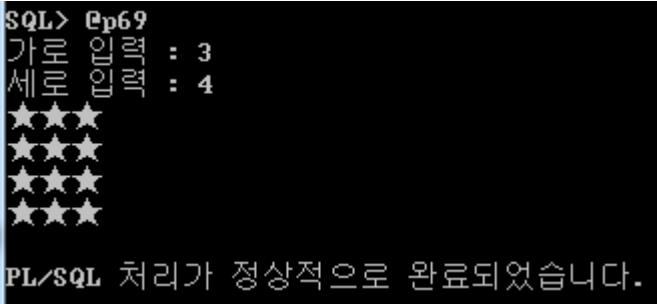
begin

    for i in 1..&p_num2 loop
        for j in 1..&p_num1 loop
            dbms_output.put('★');
        end loop;

    end loop;
```

```
        dbms_output.new_line;  
    end loop;  
end;  
/  

```



```
SQL> @p69  
가로 입력 : 3  
세로 입력 : 4  
***  
***  
***  
***  
PL/SQL 처리가 정상적으로 완료되었습니다.
```

9. case문 / continue문

2018년 5월 3일 목요일 오후 7:59

CASE 식

- CASE 식은 결과를 선택하여 반환합니다.
- 결과를 선택하기 위해 CASE 식은 표현식을 사용합니다. 이러한 표현식에서 반환되는 값은 여러 대안 중 하나를 선택하는 데 사용됩니다.

```
CASE selector
  WHEN expression1 THEN result1
  WHEN expression2 THEN result2
  ...
  WHEN expressionN THEN resultN
  [ELSE resultN+1]
END;
```

CASE 식: 예제

```
SET VERIFY OFF
DECLARE
  v_grade CHAR(1) := UPPER(' &grade' );
  v_appraisal VARCHAR2(20);
BEGIN
  v_appraisal := CASE v_grade
    WHEN 'A' THEN 'Excellent'
    WHEN 'B' THEN 'Very Good'
    WHEN 'C' THEN 'Good'
    ELSE 'No such grade'
  END;
  DBMS_OUTPUT.PUT_LINE (' Grade: ' || v_grade || '
                        Appraisal ' || v_appraisal );
END;
/
```

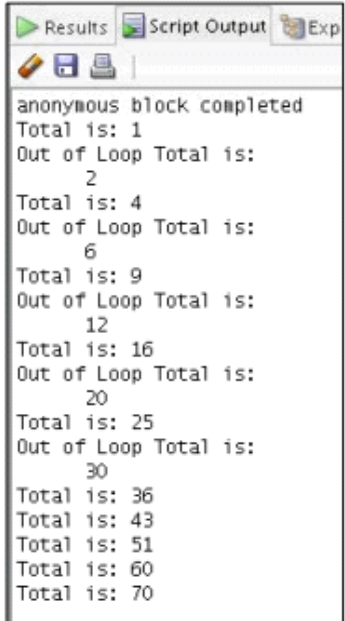
■ PL/SQL CONTINUE 문

Loop문 안에서 사용 : continue when 조건 ;

CONTINUE 문을 사용하면 루프 내의 제어를 새 반복으로 다시 전이하거나 루프를 벗어날 수 있다.

PL/SQL CONTINUE 문: 예제 1

```
DECLARE
  v_total SIMPLE_INTEGER := 0;
BEGIN
  FOR i IN 1..10 LOOP
    ① v_total := v_total + i;
      dbms_output.put_line
        ('Total is: ' || v_total);
      CONTINUE WHEN i > 5;
      v_total := v_total + i;
    ② dbms_output.put_line
        ('Out of Loop Total is:
        ' || v_total);
      END LOOP;
  END;
  /
```



Results Script Output Exp

anonymous block completed
Total is: 1
Out of Loop Total is:
2
Total is: 4
Out of Loop Total is:
6
Total is: 9
Out of Loop Total is:
12
Total is: 16
Out of Loop Total is:
20
Total is: 25
Out of Loop Total is:
30
Total is: 36
Total is: 43
Total is: 51
Total is: 60
Total is: 70

ORACLE

Copyright © 2009, Oracle. All rights reserved.

PL/SQL CONTINUE 문: 예제 1

예제에는 v_total 변수를 사용하는 할당이 두 개 있습니다.

1. 첫번째 할당은 루프의 10회 반복 각각에 대해 실행됩니다.
2. 두번째 할당은 루프의 처음 5회 반복에 대해 실행됩니다. CONTINUE 문이 루프 내의 제어를 새 반복으로 다시 전이하므로 루프의 마지막 5회 반복에 대해서는 두번째 TOTAL 할당이 실행되지 않습니다.

TOTAL 변수의 최종 결과는 70입니다.

10. 암시적 커서문 (Implicit Cursor)

2018년 5월 7일 월요일 오전 1:42

암시적 커서(Implicit Cursor)란?

암시적인 커서는 오라클이나 PL/SQL 실행 메커니즘에 의해 처리되는 SQL문장이 처리되는 곳에 대한 익명의 주소이다.

오라클 데이터베이스에서 실행되는 모든 SQL문장은 암시적인 커서가 생성되며, 커서 속성을 사용할 수 있다.

암시적 커서는 SQL 문이 실행되는 순간 자동으로 OPEN과 CLOSE를 실행 한다.

10.1 암시적 커서의 속성

- - **SQL%ROWCOUNT** : 해당 SQL 문에 영향을 받는 행의 수
- - **SQL%FOUND** : 해당 SQL 영향을 받는 행의 수가 한 개 이상일 경우 TRUE
- - **SQL%NOTFOUND** : 해당 SQL 문에 영향을 받는 행의 수가 없을 경우 TRUE
- - **SQL%ISOPEN** : 항상 FALSE, 암시적 커서가 열려 있는지의 여부 검색

10.2 예제

```
SQL> CREATE OR REPLACE PROCEDURE Implicit_Cursor
      (p_empno IN emp.empno%TYPE)
IS
  v_sal emp.sal%TYPE;
  v_update_row NUMBER;

BEGIN
  SELECT sal
    INTO v_sal
   FROM emp
  WHERE empno = p_empno;

  -- 검색된 데이터가 있을경우
  IF SQL%FOUND THEN
    DBMS_OUTPUT.PUT_LINE('검색한 데이터가 존재합니다 : '||v_sal);
  END IF;

  UPDATE emp
    SET sal = sal*1.1
   WHERE empno = p_empno;
```

```
-- 수정한 데이터의 카운트를 변수에 저장
v_update_row := SQL%ROWCOUNT;
DBMS_OUTPUT.PUT_LINE('급여가 인상된 사원 수 : '|| v_update_row);
```

EXCEPTION

```
WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE(' 검색한 데이터가 없네요... ');
```

```
END;
```

```
/
```

-- DBMS_OUTPUT.PUT_LINE을 출력하기 위해 사용

```
SQL> SET SERVEROUTPUT ON ;
```

-- 프로시저 실행

```
SQL> EXECUTE Implicit_Cursor(7369);
```

검색한 데이터가 존재합니다 : 880

급여가 인상된 사원 수 : 1

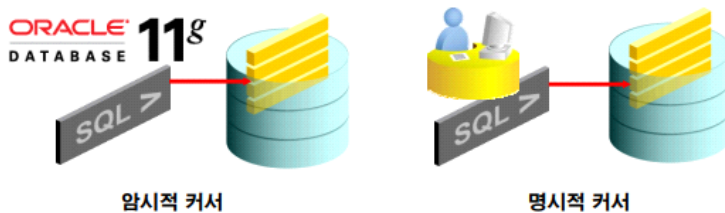
11. 명시적 커서문 (Cursor)

2018년 5월 2일 수요일 오후 3:10

커서

Oracle 서버에서 실행되는 모든 SQL 문에는 연관된 개별 커서가 있습니다.

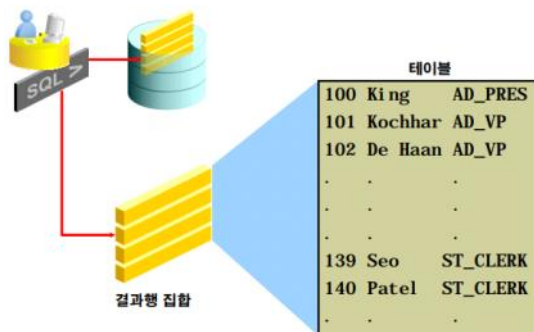
- **암시적 커서:** 모든 DML 및 PL/SQL SELECT 문에 대해 PL/SQL에서 선언하고 관리합니다.
- **명시적 커서:** 프로그래머가 선언하고 관리합니다.



■ 커서문을 사용하는 이유?

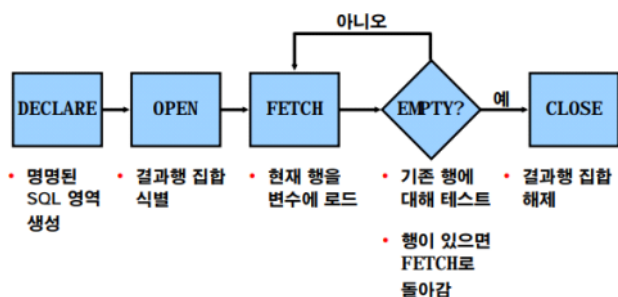
처리해야 할 데이터를 메모리에 올려 놓고 작업할 수 있기 때문에 **한번에 여러 작업을** 할 수 있다.

명시적 커서 작업



■ 명시적 커서의 제어 순서

명시적 커서 제어



11.1 사용법

set serveroutput on

declare

cursor emp_rnk is

select ename, dense_rank() over (order by sal desc) rnk
from emp;

begin

open emp_rnk;

loop

fetch emp_rnk into v_ename, v_rnk;

exit when **emp_rnk%notfound;**

update emp

set rnk = v_rnk

where ename = v_ename;

end loop;

close emp_rnk;

commit;

end;

/

1.Declare에 **cursor** 선언 : **cursor** 커서명 **is** select문

2.begin 에서 커서를 사용한다고 선언 : **open** 커서명;

3.커서값을 한 로우씩 가져옴 : **fetch** 커서명 **into** 변수1, 변수2, ... --(한 로우씩 컬럼이 변수에 담김)

4.커서값이 더 있는지 없는지 확인 할 때 : 커서명**%notfound** -- 불러올 커서 값이 없으면 true 반환 (when 에 사용)

5.커서사용이 끝나면 닫아 줘야함 (메모리에서 커서 삭제) : **close** 커서명;

■ for loop문을 이용한 커서문

*loop문	*for loop문
Declare 커서 선언;	Declare 커서 선언;
Begin 커서 오픈;	Begin For 레코드명 in 커서이름 loop
Loop	실행문;
커서 fetch;	End loop;
Exit 조건비교	End

End loop	
커서 닫기	*커서의 컬럼 개수 만큼 레코드에 공간이 있음.
End	만약 커서의 컬럼이 2개면 레코드도 컬럼을 2개씩 저장

암시적 커서에 대한 SQL 커서 속성

SQL 커서 속성을 사용하면 SQL 문의 결과를 테스트할 수 있습니다.

SQL%FOUND	가장 최근의 SQL 문이 한 행 이상에 영향을 미친 경우 TRUE로 평가되는 부울 속성
SQL%NOTFOUND	가장 최근의 SQL 문이 한 행에도 영향을 미치지 않은 경우 TRUE로 평가되는 부울 속성
SQL%ROWCOUNT	가장 최근의 SQL 문에 의해 영향을 받은 행 수를 나타내는 정수 값

11.2 예제

문제 70. 사원 테이블의 grade라는 컬럼을 추가하고 grade 컬럼의 데이터를 아래의 내용으로 갱신되게 하시오.

월급 >= 3000 A
 월급 >= 2500 B
 월급 >= 2000 C
 월급 >= 1000 D
 나머지 F

```
set serveroutput on
set verify off
```

```
accept p_ename prompt '이름을 입력 하세요 : '
```

```
declare
```

```
  v_sal emp.sal%type;      -- emp테이블의 sal 컬럼의 데이터 타입과 같게 한다.
  v_grade varchar2(10);    -- 테이블의 타입이 변경되도 소스를 수정 안해도 된다.
```

```
begin
```

```
  select sal into v_sal
    from emp
   where ename = upper('&p_ename');
```

```
  v_grade := case when v_sal >= 3000 then 'A'
                  when v_sal >= 2500 then 'B'
                  when v_sal >= 2000 then 'C'
                  when v_sal >= 1000 then 'D'
                  else 'F' end;
```

```
  update emp
     set grade = v_grade
```

```

        where ename = upper('&p_ename');
commit;
end;
/

```

```

SQL> @p70
이름을 입력 하세요 : scott
PL/SQL 처리가 정상적으로 완료되었습니다.

```

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	LOC	GRADE
8	7900	JAMES	CLERK	7698	1981-12-11 오전 12:00:00	950	(null)	30	(null)	(null)
9	7521	WARD	SALESMAN	7698	1981-02-23 오전 12:00:00	1250	500	30	(null)	(null)
10	7902	FORD	ANALYST	7566	1981-12-11 오전 12:00:00	3000	(null)	20	(null)	(null)
11	7369	SMITH	CLERK	7902	1980-12-09 오전 12:00:00	800	(null)	20	(null)	(null)
12	7788	SCOTT	ANALYST	7566	1982-12-22 오전 12:00:00	3000	(null)	20	DALLAS	A
13	7876	ADAMS	CLERK	7788	1983-01-15 오전 12:00:00	1100	(null)	20	(null)	(null)
14	7934	MILLER	CLERK	7782	1982-01-11 오전 12:00:00	1300	(null)	10	(null)	(null)

문제 71. 문제 70번을 이름을 물어보지 않고 14명의 직원들의 grade 가 동시에 갱신되게 기존 코드에서 loop문만 추가해서 모두 작성 하시오.

```

declare
    v_sal emp.sal%type;
    v_grade varchar2(20);
    v_ename emp.ename%type;
    v_num number(10);
    totnum number(10);

begin
    select count(*) into totnum
    from emp;

    for i in 1..totnum loop

        select ename, sal into v_ename,v_sal
        from(
            select ename,sal,rank() over (order by empno) rk
            from emp)
        where rk=i;

        v_grade := case when v_sal >= 3000 then 'A'
                        when v_sal >= 2500 then 'B'
                        when v_sal >= 2000 then 'C'
                        when v_sal >= 1000 then 'D'
                        else 'F' end;

        update emp
        set grade=v_grade
        where ename=v_ename;
        commit;
    end loop;
end;

```

```

        end loop;
        commit;

    end;

/

```

문제 72. 문제 70번을 커서문을 이용해서 수행 하시오.

```

set serveroutput on
set verify off

declare
    v_sal emp.sal%type;
    v_ename emp.ename%type;
    v_grade emp.grade%type;

    cursor emp_cursor is
        select ename, sal
            from emp;          -- 커서에 emp테이블의 ename, sal 컬럼 값들이 순서대로 저장된다.

begin
    open emp_cursor;          -- 커서를 사용하기위해 메모리에 적재

    loop
        fetch emp_cursor into v_ename, v_sal;
        -- 커서에 저장된 컬럼 값들을 순차적으로 불러와 v_ename, v_sal 변수에 담는다.

        exit when emp_cursor%notfound; -- 커서에서 데이터가 발견되지 않을 때 true가 됨 --> 루프문을 종료.

        v_grade := case when v_sal >= 3000 then 'A'
                        when v_sal >= 2500 then 'B'
                        when v_sal >= 2000 then 'C'
                        when v_sal >= 1000 then 'D'
                        else 'F' end;

        update emp
            set grade = v_grade
            where ename = v_ename;

    end loop;

    commit;

    close emp_cursor;          -- 메모리에 적재된 커서를 해제한다.
end;

/

```

```
SQL> @p71
```

```
PL/SQL 처리가 정상적으로 완료되었습니다.
```

1	7839	KING	PRESIDENT	(null)	1981-11-17 오전 12:00:00	5000	(null)	10	(null)	A
2	7698	BLAKE	MANAGER	7839	1981-05-01 오전 12:00:00	2850	(null)	30	(null)	B
3	7782	CLARK	MANAGER	7839	1981-05-09 오전 12:00:00	2450	(null)	10	(null)	C
4	7566	JONES	MANAGER	7839	1981-04-01 오전 12:00:00	2975	(null)	20	(null)	B
5	7654	MARTIN	SALESMAN	7698	1981-09-10 오전 12:00:00	1250	1400	30	(null)	D
6	7499	ALLEN	SALESMAN	7698	1981-02-11 오전 12:00:00	1600	300	30	(null)	D
7	7844	TURNER	SALESMAN	7698	1981-08-21 오전 12:00:00	1500	0	30	(null)	D
8	7900	JAMES	CLERK	7698	1981-12-11 오전 12:00:00	950	(null)	30	(null)	F
9	7521	WARD	SALESMAN	7698	1981-02-23 오전 12:00:00	1250	500	30	(null)	D
10	7902	FORD	ANALYST	7566	1981-12-11 오전 12:00:00	3000	(null)	20	(null)	A
11	7369	SMITH	CLERK	7902	1980-12-09 오전 12:00:00	800	(null)	20	(null)	F
12	7788	SCOTT	ANALYST	7566	1982-12-22 오전 12:00:00	3000	(null)	20	DALLAS	A
13	7876	ADAMS	CLERK	7788	1983-01-15 오전 12:00:00	1100	(null)	20	(null)	D
14	7934	MILLER	CLERK	7782	1982-01-11 오전 12:00:00	1300	(null)	10	(null)	D

문제 73. 우리반 테이블에 **grade** 라는 컬럼을 추가하고 **grade** 라는 컬럼의 데이터가 아래의 내용으로 자동 갱신되게 하시오.

```
set serveroutput on
```

```
declare
```

```
    v_ename emp2.ename%type;
```

```
    v_age emp2.age%type;
```

```
    v_grade emp2.grade%type;
```

```
cursor emp2_cursor is
```

```
    select ename, age
```

```
    from emp2;
```

```
begin
```

```
open emp2_cursor;
```

```
loop
```

```
    fetch emp2_cursor into v_ename, v_age;
```

```
exit when emp2_cursor%notfound;
```

```
v_grade := case when v_age >= 30 then 'A'
```

```
              when v_age >= 28 then 'B'
```

```
              when v_age >= 26 then 'C'
```

```
              when v_age >= 25 then 'D'
```

```
              else 'F' end;
```

```
update emp2
```

```
    set grade = v_grade
```

```
    where ename = v_ename;
```

```
end loop;
```

```
close emp2_cursor;
```

```
commit;
```

```
end;
```

/

	EMPNO	ENAME	AGE	BIRTH	MAJOR	EMAIL	MOBILE	ADDRESS	TELECOM	BIRTH_DAY	GRADE
1	10	김원섭	26	1994-02-08 오전 12:00:00	컴퓨터공학	rladnjstjkek@naver.com	010-3308-0492	경기도 남양주시 와부읍	sk	TUESDAY	C
2	11	김광록	27	1992-12-16 오전 12:00:00	경영정보학	kwangrok92@naver.com	010-7554-5473	경기도 용인시 기흥구 하갈동	lg	WEDNESDAY	C
3	20	유혜린	26	1993-04-22 오전 12:00:00	기술융합공학	yoohr22@gmail.com	010-8220-8283	서울시 송파구 방이동	lg	THURSDAY	C
4	8	이유진	25	1994-12-06 오전 12:00:00	시스템경영공	yjin1435@gmail.com	010-5920-3050	서울시 동작구 사당동	sk	TUESDAY	D
5	1	윤진민	27	1992-07-15 오전 12:00:00	경제학	yjmm92@gmail.com	010-4844-4224	경기도 구리시 교문동	sk	WEDNESDAY	C
6	9	백광홀	26	1993-06-18 오전 12:00:00	컴퓨터공학	bkh974@naver.com	010-7604-5060	서울시 노원구 공릉동	kt	FRIDAY	C
7	12	장은희	24	1996-01-07 오전 12:00:00	경영정보학	ja5772@naver.com	010-5423-4327	서울시 광진구 군자동	kt	SUNDAY	F
8	4	김영토	28	1991-10-13 오전 12:00:00	도시계획학	dudxh1@naver.com	010-3307-2231	경기도 안양시 동안구 평촌동	kt	SUNDAY	B
9	28	차호성	28	1991-07-01 오전 12:00:00	경제학	hoseong0701@naver.com	010-4946-9144	서울시 관악구 신림동	kt	MONDAY	B
10	17	방승준	26	1993-12-13 오전 12:00:00	컴퓨터공학	squazze@naver.com	010-6241-9137	서울시 관악구 신림동	kt	MONDAY	C
11	16	이한새	25	1994-01-19 오전 12:00:00	경영학	handol0@hotmail.com	010-8662-5709	경기도 용인시 기흥구 상하동	kt	WEDNESDAY	D

문제 74. 이름, 월급, 순위를 출력하는데 순위가 월급이 높은 사원순으로 순위를 부여 하시오.

```
SELECT ename, sal, dense_RANK() OVER(ORDER BY sal desc)
FROM EMP;
```

	ENAME	SAL	DENSE_RANK()OVER(ORDERBYSALDESC)
1	KING	5000	1
2	SCOTT	3000	2
3	FORD	3000	2
4	JONES	2975	3
5	BLAKE	2850	4
6	CLARK	2450	5
7	ALLEN	1600	6
8	TURNER	1500	7
9	MILLER	1300	8
10	WARD	1250	9
11	MARTIN	1250	9
12	ADAMS	1100	10
13	JAMES	950	11
14	SMITH	800	12

문제 75. 사원 테이블에 rnk라는 컬럼을 추가하고 rnk 컬럼에 월급에 대한 순위로 값을 갱신하는 PL/SQL 프로그램을 작성 하시오.

```
set serveroutput on
```

```
declare
```

```
    v_ename emp.ename%type;
```

```
    v_rnk number(10);
```

```
cursor emp_rnk is
```

```
select ename, dense_rank() over (order by sal desc) rnk
from emp;
```

```
begin
```

```
open emp_rnk;
```

```
loop
```

```
    fetch emp_rnk into v_ename, v_rnk;
```

```

exit when emp_rnk%notfound;

update emp
  set rnk = v_rnk
  where ename = v_ename;
end loop;

close emp_rnk;
commit;
end;
/

```

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	LOC	GRADE	RNK
1	7839	KING	PRESIDENT	(null)	1981-11-17 오전 12:00:00	5000	(null)	10	(null)	A	1
2	7698	BLAKE	MANAGER	7839	1981-05-01 오전 12:00:00	2850	(null)	30	(null)	B	4
3	7782	CLARK	MANAGER	7839	1981-05-09 오전 12:00:00	2450	(null)	10	(null)	C	5
4	7566	JONES	MANAGER	7839	1981-04-01 오전 12:00:00	2975	(null)	20	(null)	B	3
5	7654	MARTIN	SALESMAN	7698	1981-09-10 오전 12:00:00	1250	1400	30	(null)	D	9
6	7499	ALLEN	SALESMAN	7698	1981-02-11 오전 12:00:00	1600	300	30	(null)	D	6
7	7844	TURNER	SALESMAN	7698	1981-08-21 오전 12:00:00	1500	0	30	(null)	D	7
8	7900	JAMES	CLERK	7698	1981-12-11 오전 12:00:00	950	(null)	30	(null)	F	11
9	7521	WARD	SALESMAN	7698	1981-02-23 오전 12:00:00	1250	500	30	(null)	D	9
10	7902	FORD	ANALYST	7566	1981-12-11 오전 12:00:00	3000	(null)	20	(null)	A	2
11	7369	SMITH	CLERK	7902	1980-12-09 오전 12:00:00	800	(null)	20	(null)	F	12
12	7788	SCOTT	ANALYST	7566	1982-12-22 오전 12:00:00	3000	(null)	20	DALLAS	A	2
13	7876	ADAMS	CLERK	7788	1983-01-15 오전 12:00:00	1100	(null)	20	(null)	D	10
14	7934	MILLER	CLERK	7782	1982-01-11 오전 12:00:00	1300	(null)	10	(null)	D	8

문제 76. 사원 테이블에 loc 컬럼을 추가하고 해당 사원의 부서위치로 값을 갱신하는 PL/SQL 프로그램을 작성 하시오.

```

set serveroutput on

declare

  cursor emp_loc is
    select e.empno, d.loc
      from emp e, dept d
     where e.deptno = d.deptno;

  v_empno emp.empno%type;
  v_loc dept.loc%type;

begin

  open emp_loc;

  loop

    fetch emp_loc into v_empno, v_loc;

    exit when emp_loc%notfound;
    update emp

```

```

        set loc = v_loc
        where empno = v_empno;

    end loop;
    commit;

    close emp_loc;
end;
/

```

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	LOC
1	7839	KING	PRESIDENT	(null)	1981-11-17 오전 12:00:00	5000	(null)	10	NEW YORK
2	7698	BLAKE	MANAGER	7839	1981-05-01 오전 12:00:00	2850	(null)	30	CHICAGO
3	7782	CLARK	MANAGER	7839	1981-05-09 오전 12:00:00	2450	(null)	10	NEW YORK
4	7566	JONES	MANAGER	7839	1981-04-01 오전 12:00:00	2975	(null)	20	DALLAS
5	7654	MARTIN	SALESMAN	7698	1981-09-10 오전 12:00:00	1250	1400	30	CHICAGO
6	7499	ALLEN	SALESMAN	7698	1981-02-11 오전 12:00:00	1600	300	30	CHICAGO
7	7844	TURNER	SALESMAN	7698	1981-08-21 오전 12:00:00	1500	0	30	CHICAGO
8	7900	JAMES	CLERK	7698	1981-12-11 오전 12:00:00	950	(null)	30	CHICAGO
9	7521	WARD	SALESMAN	7698	1981-02-23 오전 12:00:00	1250	500	30	CHICAGO
10	7902	FORD	ANALYST	7566	1981-12-11 오전 12:00:00	3000	(null)	20	DALLAS
11	7369	SMITH	CLERK	7902	1980-12-09 오전 12:00:00	800	(null)	20	DALLAS
12	7788	SCOTT	ANALYST	7566	1982-12-22 오전 12:00:00	3000	(null)	20	DALLAS
13	7876	ADAMS	CLERK	7788	1983-01-15 오전 12:00:00	1100	(null)	20	DALLAS
14	7934	MILLER	CLERK	7782	1982-01-11 오전 12:00:00	1300	(null)	10	NEW YORK

문제 77. 학생이름, 생일, 태어난 날짜의 요일을 출력 하시오.

```

SELECT ename, birth, TO_char(birth,'day')
FROM EMP2;

```

	ENAME	BIRTH	TO_CHAR(BIRTH,'DAY')
1	김원섭	1994-02-08 오전 12:00:00	tuesday
2	김광록	1992-12-16 오전 12:00:00	wednesday
3	유혜린	1993-04-22 오전 12:00:00	thursday
4	이유진	1994-12-06 오전 12:00:00	tuesday
5	윤진민	1992-07-15 오전 12:00:00	wednesday
6	백광훈	1993-06-18 오전 12:00:00	friday
7	장은희	1996-01-07 오전 12:00:00	sunday
8	김영토	1991-10-13 오전 12:00:00	sunday
9	차호성	1991-07-01 오전 12:00:00	monday
10	방승준	1993-12-13 오전 12:00:00	monday
11	미한새	1994-01-19 오전 12:00:00	wednesday
12	이근호	1991-04-24 오전 12:00:00	wednesday
13	신영근	1994-02-03 오전 12:00:00	thursday

문제 78. 우리반 테이블에 birth_day라는 컬럼을 추가하고 해당 학생이 태어난 요일로 값을 갱신하는 PL/SQL 프로그램을 작성 하시오.

```

set serveroutput on

declare

cursor emp_day is

```



```

        select empno, to_char(birth,'day') day
        from emp2;
v_empno emp2.empno%type;
v_day emp2.birth_day%type;

begin

    open emp_day;

    loop
        fetch emp_day into v_empno, v_day;
        exit when emp_day%notfound;

        update emp2
            set birth_day = v_day
            where empno = v_empno;
    end loop;
    close emp_day;
    commit;

end;
/

```

	EMPNO	ENAME	AGE	BIRTH	MAJOR	EMAIL	MOBILE	ADDRESS	TELECOM	GRADE	BIRTH_DAY
1	10	김원섭	26	1994-02-08 오전 12:00:00	컴퓨터공학	rladnjstjqek@naver.com	010-3308-0492	경기도 남양주시 와부읍	sk	C	화요일
2	11	김광록	27	1992-12-16 오전 12:00:00	경영정보학	kwangrok92@naver.com	010-7554-5473	경기도 용인시 기흥구 하갈동	lg	C	수요일
3	20	유혜린	26	1993-04-22 오전 12:00:00	기술융합공학	yoohr22@gmail.com	010-8220-8283	서울시 송파구 방이동	lg	C	목요일
4	8	이유진	25	1994-12-06 오전 12:00:00	시스템경영공	yjin1435@gmail.com	010-5920-3050	서울시 동작구 사당동	sk	D	화요일
5	1	윤진민	27	1992-07-15 오전 12:00:00	경제학	yjimm92@gmail.com	010-4844-4224	경기도 구리시 교문동	sk	C	수요일
6	9	백광훈	26	1993-06-18 오전 12:00:00	컴퓨터공학	bkh974@naver.com	010-7604-5060	서울시 노원구 공릉동	kt	C	금요일
7	12	장은희	24	1996-01-07 오전 12:00:00	경영정보학	ja5772@naver.com	010-5423-4327	서울시 광진구 군자동	kt	F	일요일
8	4	김영토	28	1991-10-13 오전 12:00:00	도시계획학	dudxh1@naver.com	010-3307-2231	경기도 안양시 동안구 평촌동	kt	B	일요일
9	28	차호성	28	1991-07-01 오전 12:00:00	경제학	hoseong0701@naver.com	010-4946-9144	서울시 관악구 신림동	kt	B	월요일
10	17	방승준	26	1993-12-13 오전 12:00:00	컴퓨터공학	squazze@naver.com	010-6241-9137	서울시 관악구 신림동	kt	C	월요일
11	16	이한새	25	1994-01-19 오전 12:00:00	경영학	handol0@hotmail.com	010-8662-5709	경기도 용인시 기흥구 상하동	kt	D	수요일
12	23	이근호	28	1991-04-24 오전 12:00:00	통계학	hjku1245@naver.com	010-9484-8608	인천시 부평구 청천동	cjh	B	수요일
13	18	신영근	25	1994-02-03 오전 12:00:00	컴퓨터공학	dudrmsek@naver.com	010-2805-8258	경기도 안양시 만안구 석수동	kt	D	목요일

- 문제 78번을 for loop문으로 바꿔서 수행 하시오.

```
set serveroutput on
```

```
declare
```

```
cursor emp_day is
```

```

    select empno, to_char(birth,'day') day
    from emp2;

```

```
begin
```

```
for emp_record in emp_day loop
```

-- emp_record 에는 커서의 컬럼 수 만큼의 공간이 있다.

```
    update emp2
```

```
        set birth_day = emp_record.day
```

```
        where empno = emp_record.empno;
```

```
end loop;
```

```

commit;
end;
/

```

문제 79. 이름, 월급, 급여등급을 출력 하시오.

```

SELECT e.ename, e.sal, s.grade
FROM EMP e, SALGRADE s
WHERE e.sal BETWEEN S.LOsal AND s.hisal;

```

	ENAME	SAL	GRADE
1	SMITH	800	1
2	JAMES	950	1
3	ADAMS	1100	1
4	WARD	1250	2
5	MARTIN	1250	2
6	MILLER	1300	2
7	TURNER	1500	3
8	ALLEN	1600	3
9	CLARK	2450	4
10	BLAKE	2850	4
11	JONES	2975	4
12	FORD	3000	4
13	SCOTT	3000	4
14	KING	5000	5

문제 80. 사원 테이블의 grade라는 컬럼을 추가하고 해당 사원의 급여등급으로 값을 갱신하는 PL/SQL 프로그래밍을 작성 하시오.

```

begin
    for rcd in (
        select e.empno, s.grade
        from emp e, salgrade s
        where e.sal between s.losal and s.hisal) loop -- 커서 선언 안하고 하는 방법

        update emp
        set grade = rcd.grade
        where empno = rcd.empno;

    end loop

end;
/

```

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	LOC	RNK	GRADE
1	7839	KING	PRESIDENT	(null)	1981-11-17 오전 12:00:00	5000	(null)	10	NEW YORK	1	5
2	7698	BLAKE	MANAGER	7839	1981-05-01 오전 12:00:00	2850	(null)	30	CHICAGO	4	4
3	7782	CLARK	MANAGER	7839	1981-05-09 오전 12:00:00	2450	(null)	10	NEW YORK	3	4
4	7566	JONES	MANAGER	7839	1981-04-01 오전 12:00:00	2975	(null)	20	DALLAS	3	4
5	7654	MARTIN	SALESMAN	7698	1981-09-10 오전 12:00:00	1250	1400	30	CHICAGO	9	2
6	7499	ALLEN	SALESMAN	7698	1981-02-11 오전 12:00:00	1600	300	30	CHICAGO	6	3
7	7844	TURNER	SALESMAN	7698	1981-08-21 오전 12:00:00	1500	0	30	CHICAGO	7	3

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	LOC	RNK	GRADE
1	7839	KING	PRESIDENT	(null)	1981-11-17 오전 12:00:00	5000	(null)	10	NEW YORK	1	5
2	7698	BLAKE	MANAGER	7839	1981-05-01 오전 12:00:00	2850	(null)	30	CHICAGO	4	4
3	7782	CLARK	MANAGER	7839	1981-05-09 오전 12:00:00	2450	(null)	10	NEW YORK	3	4
4	7566	JONES	MANAGER	7839	1981-04-01 오전 12:00:00	2975	(null)	20	DALLAS	3	4
5	7654	MARTIN	SALESMAN	7698	1981-09-10 오전 12:00:00	1250	1400	30	CHICAGO	9	2
6	7499	ALLEN	SALESMAN	7698	1981-02-11 오전 12:00:00	1600	300	30	CHICAGO	6	3
7	7844	TURNER	SALESMAN	7698	1981-08-21 오전 12:00:00	1500	0	30	CHICAGO	7	3
8	7900	JAMES	CLERK	7698	1981-12-11 오전 12:00:00	950	(null)	30	CHICAGO	11	1
9	7521	WARD	SALESMAN	7698	1981-02-23 오전 12:00:00	1250	500	30	CHICAGO	9	2
10	7902	FORD	ANALYST	7566	1981-12-11 오전 12:00:00	3000	(null)	20	DALLAS	2	4
11	7369	SMITH	CLERK	7902	1980-12-09 오전 12:00:00	800	(null)	20	DALLAS	12	1
12	7788	SCOTT	ANALYST	7566	1982-12-22 오전 12:00:00	3000	(null)	20	DALLAS	2	4
13	7876	ADAMS	CLERK	7788	1983-01-15 오전 12:00:00	1100	(null)	20	DALLAS	10	1
14	7934	MILLER	CLERK	7782	1982-01-11 오전 12:00:00	1300	(null)	10	NEW YORK	8	2

문제 81. 우리반 테이블의 rnk라는 컬럼을 추가하고 나이 순서대로 rnk의 순위를 갱신하는 PL/SQL 프로그램을 작성 하시오.
(커서 사용안하고)

```

begin
    for rcd in (select empno, dense_rank() over (order by age desc) rnk
               from emp2) loop
        update emp2
            set rnk = rcd.rnk
            where empno = rcd.empno;
    end loop;
    commit;
end;
/

```

	EMPNO	ENAME	AGE	BIRTH	MAJOR	EMAIL	MOBILE	ADDRESS	TELECOM	GRADE	BIRTH_DAY	RNK
1	10	김원섭	26	1994-02-08 오전 12:00:00	컴퓨터공학	rladnjstjqek@naver.com	010-3308-0492	경기도 남양주시 와부읍	sk	C	화요일	5
2	11	김광복	27	1992-12-16 오전 12:00:00	경영정보학	kwangrok92@naver.com	010-7554-5473	경기도 용인시 기흥구 하갈동	lg	C	수요일	4
3	20	유해린	26	1993-04-22 오전 12:00:00	기술융합공학	yoohr22@gmail.com	010-8220-8283	서울시 송파구 방이동	lg	C	목요일	5
4	8	이유진	25	1994-12-06 오전 12:00:00	시스템경영공	yjin1435@gmail.com	010-5920-3050	서울시 동작구 사당동	sk	D	화요일	6
5	1	윤진민	27	1992-07-15 오전 12:00:00	경제학	yjimm92@gmail.com	010-4844-4224	경기도 구리시 교문동	sk	C	수요일	4
6	9	백광훈	26	1993-06-18 오전 12:00:00	컴퓨터공학	bkh974@naver.com	010-7604-5060	서울시 노원구 공릉동	kt	C	금요일	5
7	12	장은희	24	1996-01-07 오전 12:00:00	경영정보학	ja5772@naver.com	010-5423-4327	서울시 광진구 교자동	kt	F	일요일	7
8	4	김영토	28	1991-10-13 오전 12:00:00	도시계획학	dudxh1@naver.com	010-3307-2231	경기도 안양시 동안구 평촌동	kt	B	일요일	3
9	28	차호성	28	1991-07-01 오전 12:00:00	경제학	hoseong0701@naver.com	010-4946-9144	서울시 관악구 신림동	kt	B	월요일	3
10	17	방승준	26	1993-12-13 오전 12:00:00	컴퓨터공학	squazze@naver.com	010-6241-9137	서울시 관악구 신림동	kt	C	월요일	5
11	16	미한세	25	1994-01-19 오전 12:00:00	경영학	handol0@hotmail.com	010-8662-5709	경기도 용인시 기흥구 상하동	kt	D	수요일	6
12	23	미근호	28	1991-04-24 오전 12:00:00	통계학	hku1245@naver.com	010-9484-8608	인천시 부평구 청천동	cjh	B	수요일	3
13	18	신영근	25	1994-02-03 오전 12:00:00	컴퓨터공학	dudrmsek@naver.com	010-2805-8258	경기도 안양시 만안구 석수동	kt	D	목요일	6
14	22	신현수	27	1992-05-04 오전 12:00:00	경제학	shyuns0504@gmail.com	010-2079-6714	서울시 관악구 삼성동	kt	C	월요일	4
15	27	미한준	28	1991-10-26 오전 12:00:00	신학	lcjpaul@naver.com	010-4047-8810	서울시 영등포구 영등포동	lg	B	토요일	3
16	26	도웅	28	1991-01-12 오전 12:00:00	산업공학과	gunusa@naver.com	010-4422-4274	안양시 동안구 평촌동	lg	B	토요일	3

문제 82. 아래와 같이 통신사를 물어보게 하고 통신사를 입력하면 해당 통신사인 학생들의 정보가 출력되게 하시오.

```

set serveroutput on
set verify off

accept p_telecom prompt '통신사 입력 : '

```

```

begin
    for rcd in (select ename, age, major
                from emp2
                where telecom = lower('&p_telecom')) loop

        dbms_output.put_line ( rcd.ename || ' - ' || rcd.age || ' - ' || rcd.major );
    end loop;
end;
/

```

```

SQL> Ep82
신사 입력 : sk
이원섭 - 26 - 컴퓨터공학
유진미 - 25 - 시스템경영공학부
이진미 - 27 - 경제학
대경윤 - 27 - 통계학
홍영환 - 28 - 경영학
이동환 - 25 - 컴퓨터정보시스템과
이건태 - 28 - 환경공학
한지훈 - 24 - 경영학

PL/SQL 처리가 정상적으로 완료되었습니다.

```

문제 83. 부서번호를 물어보게 하고 부서번호를 입력하면 해당 부서번호의 직원들의 이름, 월급, 부서번호가 아래와 같이 출력되게 하시오.

부서번호를 입력하세요 : 20

```

set serveroutput
set verify off
accept p_deptno prompt '부서번호를 입력 하세요 : '

```

```

begin
    for rcd in (select ename, sal, deptno
                from emp
                where deptno = &p_deptno) loop

        dbms_output.put_line(rcd.ename || ' - ' || rcd.sal || ' - ' || rcd.deptno );
    end loop;
end;
/

```

```

SQL> Ep83
부서번호를 입력 하세요 : 20
JONES - 2975 - 20
FORD - 3000 - 20
SMITH - 800 - 20
SCOTT - 3000 - 20
ADAMS - 1100 - 20

PL/SQL 처리가 정상적으로 완료되었습니다.

```

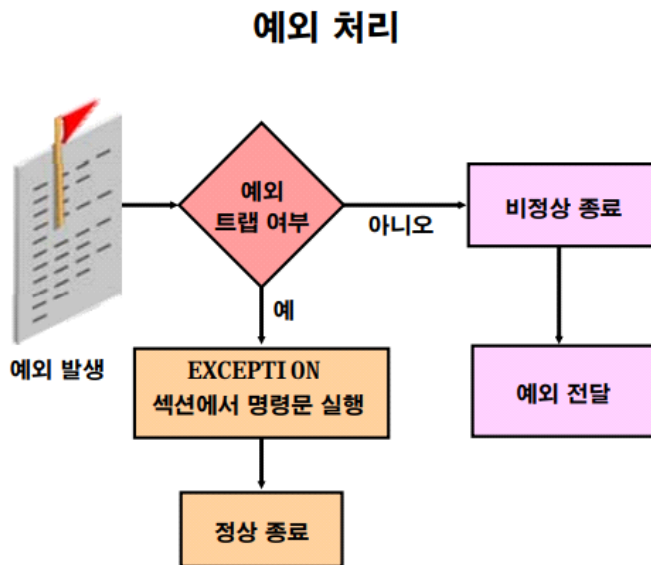
12. 예외처리

2018년 5월 3일 목요일 오후 2:13

■ 예외처리를 해야 하는 이유?

프로그램이 비정상적으로 처리되어서 잘못된 결과를 출력하지 않도록 하기 위해서

-- 예외가 발생하면 바로 exception으로 건너 뛴다.



■ 예외 종류 3가지

- 1.오라클에서 미리 정의한 예외
- 2.오라클에서 미리 정의하지 않은 예외
- 3.사용자 정의 예외

* 예외는 실행 절 (begin) 에 있어야만 catch 할 수 있다.

12.1 사용법

■ 오라클에서 미리 정의한 예외

1.exception 블록에 예외처리

Ex) exception

When 정의된 예외종류 then

실행문;

-- begin 안에서 발생한 예외에 대해서만 catch 한다!!!

■ 사용자 정의 예외

1. 예외 선언 : e_emp_exception exception;

2. if 문에 예외 조건 설정 : ex) if v_cnt=0 then raise e_emp_exception;

3. exception 블록에 예외처리 :

Ex) Exception

```
When e_emp_exception[사용자정의예외명] then  
    실행문;
```

미리 정의된 Oracle 서버 오류 트랩

- 예외 처리 루틴에서 미리 정의된 이름을 참조합니다.
- 미리 정의된 예외 예제:
 - NO_DATA_FOUND
 - TOO_MANY_ROWS
 - INVALID_CURSOR
 - ZERO_DIVIDE
 - DUP_VAL_ON_INDEX

12.2 예제

문제 85. 부서번호가 10번인 사원의 토탈 월급을 출력 하시오.

```
SELECT SUM(sal)  
FROM EMP  
WHERE deptno = 10;
```

	SUM(SAL)
1	8750

문제 86. 부서번호를 물어보게 하고 부서번호를 입력하면 해당 부서번호의 토탈 월급이 출력되는 PL/SQL 프로그램을 작성하는데 없는 부서번호를 입력하면 " 해당 부서번호는 없습니다 " 라는 메시지가 출력되게 하시오.

```
set serveroutput on  
set verify off  
  
accept p_deptno prompt '부서번호를 입력하세요 : '  
  
declare  
    v_sal emp.sal%type;  
  
begin  
    select sum(sal) into v_sal  
    from emp  
    where deptno = &p_deptno  
    group by deptno;  
  
    dbms_output.put_line(v_sal);  
  
exception  
    when no_data_found then          -- 선택된 레코드가 없을 때만 !!  
        dbms_output.put_line('해당 부서번호는 없습니다.');
```

```
end;
/
```

```
SQL> @p86
부서번호를 입력하세요 : 3
해당 부서번호는 없습니다.

PL/SQL 처리가 정상적으로 완료되었습니다.
```

문제 87. 숫자 2개를 아래와 같이 각각 물어보게 하고 첫 번째 숫자를 2번째 숫자가 나눈 값이 출력되게 하는 PL/SQL을 작성 하시오.

```
set serveroutput on
set verify off

accept p_num1 prompt '첫 번째 숫자를 입력하세요 : '
accept p_num2 prompt '두 번째 숫자를 입력하세요 : '

begin
    dbms_output.put_line('나눈 값 : ' || &p_num1/&p_num2);
end;
/
```

```
SQL> @p87
첫 번째 숫자를 입력하세요 : 3
두 번째 숫자를 입력하세요 : 0
          begin
*
1 행에 오류:
ORA-01476: 제수가 0 입니다
ORA-06512: 3행
```

문제 88. 문제 87번을 다시 수행하는데 숫자 0으로 나누면 예러가 나는게 아니라 0으로 나눌 수 없습니다 라는 메시지가 나오게 하시오.

```
set serveroutput on
set verify off

accept p_num1 prompt '첫 번째 숫자를 입력하세요 : '
accept p_num2 prompt '두 번째 숫자를 입력하세요 : '

begin
    dbms_output.put_line('나눈 값 : ' || &p_num1/&p_num2);

exception
    when zero_divide then
        dbms_output.put_line('0으로 나눌 수 없습니다.');
```

```
end;
/
```

```
SQL> @p87
첫 번째 숫자를 입력하세요 : 3
두 번째 숫자를 입력하세요 : 0
```

```
SQL> @p87
첫 번째 숫자를 입력하세요 : 3
두 번째 숫자를 입력하세요 : 0
0으로 나눌 수 없습니다.
```

문제 89. 문제 83번에 예외처리를 추가해서 부서번호가 없으면 메시지가 출력되게 하시오.

```
set serveroutput on
set verify off

accept p_deptno prompt '부서번호를 입력 하세요 : '

begin
    for rcd in (select ename, sal, deptno
                from emp
                where deptno = &p_deptno
                ) loop

        dbms_output.put_line(rcd.ename || ' - ' || rcd.sal || ' - ' || rcd.deptno );
    end loop;

exception
    when invalid_cursor then
        dbms_output.put_line('해당 데이터는 없습니다.');
```

end;

/

```
SQL> @p89
부서번호를 입력 하세요 : 4

PL/SQL 처리가 정상적으로 완료되었습니다.
```

*예외에 catch 되지 않는다. ---> 사용자 정의 예외를 사용 해야된다.

문제 90. 통신사를 물어보게 하고, 통신사를 입력하면 해당 통신사인 학생들의 정보가 출력되게 하는데 없는 통신사를 넣으면 메시지가 출력되게 하시오.

```
set serveroutput on
set verify off

accept p_deptno prompt '부서번호를 입력 하세요 : '

declare

    v_deptno emp.deptno%type := &p_deptno;
    v_cnt number(10);
    v_exception exception;

begin
    select count(*) into v_cnt
    from emp
```



```

        where deptno = v_deptno;

    if v_cnt=0 then
        raise v_exception;
    else
        for rcd in (select ename, sal, deptno
                    from emp
                    where deptno = &p_deptno) loop

            dbms_output.put_line(rcd.ename || ' - ' || rcd.sal || ' - ' || rcd.deptno );
        end loop;
    end if;

exception

    when v_exception then
        dbms_output.put_line('해당 부서번호는 없습니다.');
```

end;
/

```

SQL> @p90
부서번호를 입력 하세요 : 3
해당 부서번호는 없습니다.

PL/SQL 처리가 정상적으로 완료되었습니다.
```

문제 91. 통신사를 입력 받아 해당 통신사인 학생들의 정보가 출력되게, 통신사가 존재하지 않으면 메시지를 출력 하시오.

```

set serveroutput on
set verify off

accept p_telecom prompt '통신사 입력 : '

declare
    v_cnt number(10);
    v_exception exception;

begin
    select count(*) into v_cnt
    from emp2
    where telecom = '&p_telecom';

    if v_cnt = 0 then
        raise v_exception;
    else
        for rcd in (select ename, age, telecom
                    from emp2
                    where telecom = '&p_telecom') loop

            dbms_output.put_line(rcd.ename || ' - ' || rcd.age || ' - ' || rcd.telecom );
        end loop;
    end if;
end;
```

```

        end loop;
    end if;

    exception
        when v_exception then
            dbms_output.put_line ('존재하지 않는 통신사 입니다.');
```

end;

/

```

SQL> @p91
통신사 입력 : rr
존재하지 않는 통신사 입니다.
PL/SQL 처리가 정상적으로 완료되었습니다.
```

13. 함수

2018년 5월 3일 목요일 오후 4:14

■ 함수의 종류 2가지

1. 오라클 내장 함수 :

문자, 숫자, 날짜, 변환, 일반, 그룹함수, 분석함수

2. 사용자 정의 함수 :

사용자가 필요한 함수를 직접 만들어서 필요할 때 사용하는 함수

13.1 사용법

Create [or replace] **function** 함수명 (매개변수 **in** 매개변수 타입, [....]) **return** 반환 타입

Is -- is가 declare 역할을 한다

[변수 선언]

begin

실행문;

--- 함수에선 & 를 사용하면 안된다!!

Return 리턴값;

End;

/

*매개변수에 바이트를 지정해주면 안된다

*함수는 select 절이랑 where 절에서 주로 호출된다.

13.2 예제

문제 91. 부서번호를 입력하면 부서위치가 출력되는 함수를 생성 하시오.

```
CREATE OR REPLACE FUNCTION get_loc (p_Deptno IN dept.deptno%type) RETURN dept.loc%TYPE
```

```
IS
```

```
v_loc dept.loc%TYPE;
```

```
BEGIN
```

```
    SELECT loc INTO v_loc
```

```
    FROM DEPT
```

```
    WHERE deptno = p_deptno;
```

```
    RETURN v_loc;
```

```
END;
```

```
/
```

```
SQL> @p91
```

```
함수가 생성되었습니다.
```

```
SELECT ename, deptno, get_loc(Deptno)
FROM EMP;
```

	ENAME	DEPTNO	GET_LOC(DEPTNO)
1	KING	10	NEW YORK
2	BLAKE	30	CHICAGO
3	CLARK	10	NEW YORK
4	JONES	20	DALLAS
5	MARTIN	30	CHICAGO
6	ALLEN	30	CHICAGO
7	TURNER	30	CHICAGO
8	JAMES	30	CHICAGO
9	WARD	30	CHICAGO
10	FORD	20	DALLAS
11	SMITH	20	DALLAS
12	SCOTT	20	DALLAS
13	ADAMS	20	DALLAS
14	MILLER	10	NEW YORK

문제 92. 부서번호를 입력하면 해당 부서번호의 토탈 월급이 출력되는 함수를 생성하고 아래와 같이 실행 하시오.

```
create or replace function deptno_sum (v_deptno in emp.deptno%type) return emp.sal%type
is
```

```
v_sumsal emp.sal%type;
```

```
begin
```

```
select sum(sal) into v_sumsal
```

```
from emp
```

```
where deptno = v_deptno;
```

```
return v_sumsal;
```

```
end;
```

```
/
```

```
SQL> @p92
```

```
함수가 생성되었습니다.
```

```
SELECT ename, deptno, deptno_sum(deptno)
```

```
FROM EMP;
```

	ENAME	DEPTNO	DEPTNO_SUM(DEPTNO)
1	KING	10	8750
2	BLAKE	30	9400
3	CLARK	10	8750
4	JONES	20	10875
5	MARTIN	30	9400
6	ALLEN	30	9400
7	TURNER	30	9400
8	JAMES	30	9400
9	WARD	30	9400
10	FORD	20	10875
11	SMITH	20	10875
12	SCOTT	20	10875
13	ADAMS	20	10875
14	MILLER	10	8750

문제 93. gauss 테이블의 데이터를 아래와 같이 검색해서 아래의 결과가 출력되는 함수를 생성 하시오.

```
create or replace function func_gauss(v_num1 in number,v_num2 in number) return number
is
v_rst number(10);

begin
    select ((v_num1+v_num2)*(v_num2/2)) into v_rst
    from dual;
    return v_rst;
end;
/
```

SQL> Ep93
함수가 생성되었습니다.

```
SELECT num1,num2, func_gauss(num1, num2)
FROM GAUSS;
```

	NUM1	NUM2	FUNC_GAUSS(NUM1,NUM2)
1	1	10	55
2	1	20	210
3	1	30	465
4	1	40	820
5	1	50	1275
6	1	60	1830

문제 94. log_Example 테이블을 아래와 같이 생성하고 num1이라는 컬럼에 loop문을 이용해서 데이터를 입력 하시오.

```
set serveroutput on

begin
    for i in 2..10 loop
        insert into log_Example(num1)
        values(i);
    end loop;
    commit;
end;
/
```

	NUM1	NUM2
1	2 (null)	
2	3 (null)	
3	4 (null)	
4	5 (null)	
5	6 (null)	
6	7 (null)	
7	8 (null)	
8	9 (null)	
9	10 (null)	

문제 95. 수학문제 (19페이지) 예제 3번 구간 [4,10] 에서 $y = \log_3(x-1)$ 의 최대값과 최소값을 각각 구하시오.
위의 수학문제를 풀기 위한 함수를 생성 하시오.

```
create or replace function log_value (v_num1 in log_example.num1%type) return number
is
```

```
v_rst number(10,7);
```

```
begin
```

```
    select log(3,v_num1-1) into v_rst
    from dual;
```

```
    return v_rst;
```

```
end;
```

```
/
```

	NUM1	LOG_VALUE(NUM1)
1	2	0
2	3	0.6309298
3	4	1
4	5	1.2618595
5	6	1.4649735
6	7	1.6309298
7	8	1.7712437
8	9	1.8927893
9	10	2

문제 96. 문제 95번에서 만든 log_value 함수를 이용해서 log_example 테이블에 num2에 값을 갱신 하는 update 문을 작성 하시오.

함수

```
create or replace function log_value (v_num1 in log_example.num1%type) return number
is
```

```
v_rst number(10,7);
```

```
begin
```

```
    select log(3,v_num1-1) into v_rst
```

```
    from dual;          -- dual이 아니라 log_example 테이블이였으면 update 시 오류 발생!!!
```

```
    -- a테이블에서 호출한 함수에 a테이블의 컬럼 값을 가져와 가공해서 update 하면 오류 발생
```

```
    return v_rst;
```

```
end;
```

```
/
```

```
UPDATE log_example
```

```
    SET num2 = log_value(num1);
```

```
SELECT *
```

```
    FROM log_example;
```

	NUM1	NUM2
1	2	0
2	3	0.63093
3	4	1
4	5	1.26186
5	6	1.46497
6	7	1.63093
7	8	1.77124
8	9	1.89279
9	10	2

문제 97. 수학문제 19페이지에 문제 4번인 다음 수를 작은 것 부터 차례대로 나열하는 함수를 만들기 위해 아래의 테이블을 생성하고 데이터를 입력 하시오.

1 , log₃7 , log₉100

```
CREATE TABLE t_p97 (num1 varchar2(10), num2 NUMBER(10));
```

```
INSERT INTO t_p97 (num1)
VALUES ('1');
```

```
INSERT INTO t_p97 (num1)
VALUES ('log(3,7)');
```

```
INSERT INTO t_p97 (num1)
VALUES ('log(9,100)');
```

	NUM1	NUM2
1	1	(null)
2	log(3,7)	(null)
3	log(9,100)	(null)

문제 98. 문제 97번의 테이블의 num1의 값이 계산되는 함수를 생성 하시오.

```
create or replace function log_value (p_num1 in t_p97.num1%type) return number
is
    v_stmt varchar2(100);
    v_value number(10,8);

begin
    v_stmt := 'select ' || p_num1 || ' from dual';
    execute immediate v_stmt into v_value; -- execute immediate : 즉시 실행해라
                                           -- v_stmt 변수에 담겨있는 문자열 (sql문)을
                                           -- 그 결과를 v_value에 담는다.

    return v_value;
end;
/
```

```
SELECT num1, log_value(num1)
FROM t_p97;
```

	NUM1	LOG_VALUE(NUM1)
1	1	1
2	log(3,7)	1.77124375
3	log(9,100)	2.09590327

문제 99. 아래의 SQL을 다시 수행하는데 log_value(num1)이 높은 순서대로 순위가 부여되게 순위 컬럼을 출력 하시오.

```
SELECT num1, log_value(num1)
FROM t_p97;
```

```
SELECT num1, log_value(num1) AS 로그 , RANK() OVER (ORDER BY log_value(num1) desc) rn
```

FROM t_p97;

	NUM1	로그	RNK
1	log(9,100)	2.09590327	1
2	log(3,7)	1.77124375	2
3	1	1	3

문제 100. 아래의 테이블을 생성 하시오.

```
CREATE TABLE emp100
( COL1 VARCHAR2(20), col2 NUMBER(10,7));
```

```
INSERT INTO emp100 (col1)
VALUES ('max(sal)');
```

```
INSERT INTO emp100 (col1)
VALUES ('min(sal)');
```

```
INSERT INTO emp100 (col1)
VALUES ('sum(sal)');
```

```
INSERT INTO emp100 (col1)
VALUES ('avg(sal)');
```

```
SELECT *
FROM emp100;
```

	COL1	COL2
1	max(sal)	(null)
2	min(sal)	(null)
3	sum(sal)	(null)
4	avg(sal)	(null)

문제 101. emp100 테이블을 select 해서 아래의 결과가 출력되게 하는 함수를 생성 하시오.

```
create or replace function func_100 (v_col1 in emp100.col1%type) return number
is
```

```
v_stmt varchar2(30);
v_rst emp100.col2%type;
```

```
begin
```

```
    v_stmt := 'select ' || v_col1 || ' from emp';
    execute immediate v_stmt into v_rst;
    return v_rst;
```

```
end;
/
```

```
SELECT col1,func_100(col1)
FROM emp100;
```

	COL1	FUNC_100(COL1)
1	max(sal)	5000
2	min(sal)	800
3	sum(sal)	29025
4	avg(sal)	2073

문제 102. emp100 테이블에 col2 컬럼의 데이터가 아래와 같이 갱신되게 하시오.

```
UPDATE emp100
SET col2 = func_100(col1);
```

	COL1	COL2
1	max(sal)	5000
2	min(sal)	800
3	sum(sal)	29025
4	avg(sal)	2073

문제 103. SCOTT 유저가 가지고 있는 함수 리스트를 조회 하시오.

```
SELECT *
FROM user_objects
WHERE OBJECT_type = 'FUNCTION';
```

	OBJECT_NAME	SUBOBJECT_NAME	OBJECT_ID	DATA_OBJECT_ID	OBJECT_TYPE	CREATED	LAST_DDL_TIME	TIMESTAMP	STATUS	TEMPORARY	GENERATED	SECONDARY	NAMESPACE	EDITION_NAME
1	LOG_VALUE	(null)	74956	(null)	FUNCTION	2018-05-04 오전 9:59:51	2018-05-04 오전 11:19:46	2018-05-04 11:19:46	VALID	N	N	N	1	(null)
2	GET_LOC	(null)	74949	(null)	FUNCTION	2018-05-03 오후 4:20:56	2018-05-03 오후 4:20:56	2018-05-03 16:20:56	VALID	N	N	N	1	(null)
3	FUNC_GAUSS	(null)	74951	(null)	FUNCTION	2018-05-03 오후 4:47:23	2018-05-03 오후 4:54:27	2018-05-03 16:54:27	VALID	N	N	N	1	(null)
4	FUNC_100	(null)	74966	(null)	FUNCTION	2018-05-04 오전 11:35:3	2018-05-04 오전 11:48:59	2018-05-04 11:48:59	VALID	N	N	N	1	(null)
5	DEPTNO_SUM	(null)	74950	(null)	FUNCTION	2018-05-03 오후 4:35:43	2018-05-03 오후 4:36:49	2018-05-03 16:36:49	VALID	N	N	N	1	(null)

문제 104. 생성한 함수의 스크립트를 확인 하시오.

```
SELECT *
FROM user_objects
WHERE OBJECT_type = 'LOG_VALUE';
```

	NAME	TYPE	LINE	TEXT
1	LOG_VALUE	FUNCTION	1	function log_value (p_num1 in t_p97.num1%type) r
2	LOG_VALUE	FUNCTION	2	is
3	LOG_VALUE	FUNCTION	3	v_stmt varchar2(100);
4	LOG_VALUE	FUNCTION	4	v_value number(10,8);
5	LOG_VALUE	FUNCTION	5	
6	LOG_VALUE	FUNCTION	6	begin
7	LOG_VALUE	FUNCTION	7	v_stmt := 'select ' p_num1 ' from dual';
8	LOG_VALUE	FUNCTION	8	execute immediate v_stmt into v_value;
9	LOG_VALUE	FUNCTION	9	
10	LOG_VALUE	FUNCTION	10	return v_value;
11	LOG_VALUE	FUNCTION	11	end;

문제 105. SCOTT 유저가 생성한 함수들을 모두 삭제 하시오.

```
DROP FUNCTION GET_LOC;
DROP FUNCTION LOG_VALUE;
DROP FUNCTION FUNC_GAUSS;
```

...

14. 프로시저

2018년 5월 3일 목요일 오후 7:35

■ PL/SQL의 프로시저

1. anonymous PL/SQL -- 이름이 없다.
2. 함수 -- 이름이 있다.
3. 프로시저 -- 이름이 있다.

data base에 PL/STR코드를 저장하느냐 저장 하지 않느냐의 차이이다.

■ 프로시저와 함수의 차이

프로시저	함수
RETURN 절이 없다	RETURN 절이 있다.
결과를 출력 안해도 된다.	결과가 반드시 출력 되어야 한다.

14.1 예제

문제 106. Anonymous PL/SQL 문제 6번을 프로시저로 만드시오.

```
create or replace procedure pro106 (p_empno in emp.empno%type)
is
    v_job varchar2(20);

begin
    select job into v_job
    from emp

    where empno = p_empno;

    dbms_output.put_line('해당 사원의 직업은 ' || v_job);
end;
/
```

```
SQL> create procedure pro106
프로시저가 생성되었습니다.
```

```
SQL> exec pro106(7788);
해당 사원의 직업은 ANALYST
PL/SQL 처리가 정상적으로 완료되었습니다.
```

문제 107. 아래와 같이 이름을 입력해서 프로시저를 실행하면 해당 사원의 월급이 출력되게 하는 프로시저를 작성 하시오.

```
SQL> exec get_sal('scott');
```

3000

```
create or replace procedure get_sal(v_ename in emp.ename%type)
is
```

```
    v_rst emp.sal%type;
```

```
begin
```

```
    select sal into v_rst
        from emp
        where ename = upper(v_ename);
```

```
    dbms_output.put_line(v_rst);
end;
/
```

```
SQL> Cp107
프로시저가 생성되었습니다.
SQL> exec get_sal('SCOTT');
3000
PL/SQL 처리가 정상적으로 완료되었습니다.
```

문제 108. 아래와 같이 이름을 입력해서 프로시저를 실행하면 해당 사원의 월급에 대한 순위가 출력되는 프로시저를 생성 하시오.

```
create or replace procedure get_sal_rank(v_ename in emp.ename%type)
is
```

```
    v_rst number(10);
```

```
begin
```

```
    select rn into v_rst
        from(select ename, dense_rank() over (order by sal desc) rn
             from emp)
        where ename = upper(v_ename);
```

```
    dbms_output.put_line(v_rst || '등 입니다. ');
end;
/
```

```
SQL> Cp108
프로시저가 생성되었습니다.
SQL> exec get_sal_rank('scott');
2등 입니다.
PL/SQL 처리가 정상적으로 완료되었습니다.
```

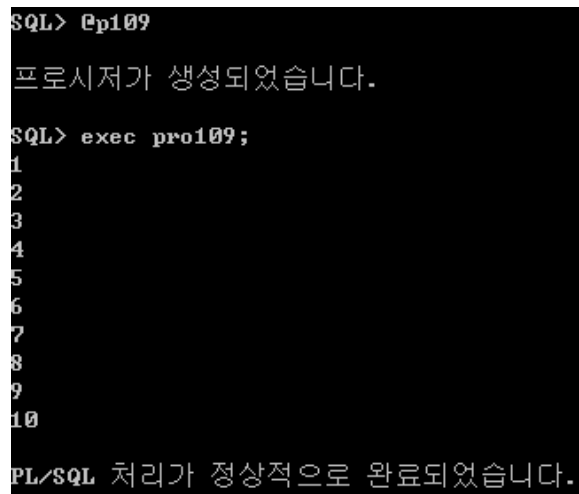
문제 109. 아래와 같이 프로시저를 수행하면 1부터 10까지 출력 되게하는 프로시저를 생성 하시오.

```
SQL> exec pro109;
1
2
3
4
5
...
10
create or replace procedure pro109
is

begin
    for i in 1.. 10 loop

        dbms_output.put_line(i);
    end loop;

end;
/
```



The screenshot shows a SQL*Plus session. The prompt is SQL>. The user enters 'ep109' (likely a typo for 'pro109'). The output is '프로시저가 생성되었습니다.' (The procedure was created successfully). Then the user enters 'SQL> exec pro109;'. The output is a list of numbers from 1 to 10, each on a new line. At the bottom, the prompt changes to PL/SQL and the output is '처리가 정상적으로 완료되었습니다.' (The processing was completed normally).

문제 110. 아래와 같이 구구단의 해당 단이 출력되는 프로시저를 생성 하시오.

```
set serveroutput on

create or replace procedure gugu_dan(v_dan in number)
is

begin

    for i in 1..9 loop
```

```

        dbms_output.put_line(v_dan || ' x ' || i || ' = ' || i*v_dan);
    end loop;
end;
/

```

프로시저가 생성되었습니다.

```

SQL> exec gugu_dan(3);
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
PL/SQL 처리가 정상적으로 완료되었습니다.

```

문제 111. demobld를 돌리고 grade 컬럼을 추가한 후에 해당 사원의 급여등급으로 값을 갱신하는 PL/SQL 코드인 문제 72번을 프로시저로 생성해서 수행 하시오.

```

create or replace procedure pro111    --- 기존의 소스에 declare 을 지우고 써줬다.
is

v_sal emp.sal%type;
v_ename emp.ename%type;
v_grade emp.grade%type;

cursor emp_cursor is
select ename, sal
      from emp;

begin

open emp_cursor;

loop
    fetch emp_cursor into v_ename, v_sal;
    exit when emp_cursor%notfound;

    v_grade := case when v_sal >= 3000 then 'A'
                    when v_sal >= 2000 then 'B'
                    when v_sal >= 1500 then 'C'
                    when v_sal >= 1000 then 'D'
                    else 'F' end;

    update emp
        set grade = v_grade
        where ename = v_ename;

end loop;
commit;

```

```
close emp_cursor;
```

```
end;
```

```
/
```

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	GRADE
1	7839	KING	PRESIDENT	(null)	1981-11-17 오전 12:00:00	5000	(null)	10	A
2	7698	BLAKE	MANAGER	7839	1981-05-01 오전 12:00:00	2850	(null)	30	B
3	7782	CLARK	MANAGER	7839	1981-05-09 오전 12:00:00	2450	(null)	10	B
4	7566	JONES	MANAGER	7839	1981-04-01 오전 12:00:00	2975	(null)	20	B
5	7654	MARTIN	SALESMAN	7698	1981-09-10 오전 12:00:00	1250	1400	30	D
6	7499	ALLEN	SALESMAN	7698	1981-02-11 오전 12:00:00	1600	300	30	C
7	7844	TURNER	SALESMAN	7698	1981-08-21 오전 12:00:00	1500	0	30	C
8	7900	JAMES	CLERK	7698	1981-12-11 오전 12:00:00	950	(null)	30	F
9	7521	WARD	SALESMAN	7698	1981-02-23 오전 12:00:00	1250	500	30	D
10	7902	FORD	ANALYST	7566	1981-12-11 오전 12:00:00	3000	(null)	20	A
11	7369	SMITH	CLERK	7902	1980-12-09 오전 12:00:00	800	(null)	20	F
12	7788	SCOTT	ANALYST	7566	1982-12-22 오전 12:00:00	3000	(null)	20	A
13	7876	ADAMS	CLERK	7788	1983-01-15 오전 12:00:00	1100	(null)	20	D
14	7934	MILLER	CLERK	7782	1982-01-11 오전 12:00:00	1300	(null)	10	D

문제 112. 직원번호를 물어보게 하고 직원번호를 입력 했을 때 해당 직원의 월급이 출력되는 코드인데 없는 직원번호를 넣었을 때 해당 직원은 없습니다 라는 메시지를 출력하도록 하자. (예외 사용)

```
create or replace procedure pro112 (v_empno in emp.empno%type )
```

```
is
```

```
    v_rst emp.sal%type;
```

```
begin
```

```
    select sal into v_rst
    from emp
    where empno = v_empno;
```

```
    dbms_output.put_line('월급 : ' ||v_rst);
```

```
exception
```

```
    when no_data_found then
        dbms_output.put_line('해당 직원은 없습니다.');
```

```
end;
```

```
/
```

```

SQL> exec pro112<7788>;
월급 : 30000

PL/SQL 처리가 정상적으로 완료되었습니다.

SQL> exec pro112<1234>;
해당 사원은 없습니다.

PL/SQL 처리가 정상적으로 완료되었습니다.

```

문제 113. 아래의 두개의 함수를 각각 만들고

$$f(x) = \log_a(x-m) + n$$

$$g(x) = a^{x-n} + m$$

Select f(g(1)) from dual; 의 결과가 1인지 확인해 보시오.

Fx함수

```

create or replace function fx
(v_num in number) return number
is
v_rst number(20,5);

begin

    select log(3,v_num-3)+3 into v_rst
        from dual;
    return v_rst;
end;
/

```

Gx함수

```

create or replace function gx
(x in number) return number
is
v_rst number(20,5);

begin

    select power(3,x-3)+3 into v_rst
        from dual;
    return v_rst;
end;
/

```

문제 114. 직업, 부서번호, 직업별 부서번호별 토달 월급을 출력 하시오.

```
SELECT job,SUM(DECODE(Deptno,10,sal,0)) "10", SUM(DECODE(deptno,20,sal,0))"20",
SUM(DECODE(Deptno,30,sal,0))"30"
      FROM EMP
      GROUP BY job;
```

	JOB	10	20	30
1	SALESMAN	0	0	5600
2	CLERK	1300	1900	950
3	PRESIDENT	5000	0	0
4	MANAGER	2450	2975	2850
5	ANALYST	0	6000	0

문제 115. 아래의 SQL문이 출력되는 프로시저를 생성 하시오.

```
create or replace procedure pro115
is
v_sum varchar2(100);

begin

      v_sum := ' sum(Decode(deptno,10,sal,0) ';
      dbms_output.put_line(v_sum);
end;
/
```

```
SQL> exec pro115
sum<Decode<deptno,10,sal,0>sum<Decode<deptno,20,sal,0>sum<Decode<deptno,30,sal,0
>
PL/SQL 처리가 정상적으로 완료되었습니다.
```

문제 116. 문제 115번 프로시저에 for loop문을 추가해서 아래의 SQL문이 출력되는 프로시저를 생성 하시오.

```
create or replace procedure pro116
is
v_sum varchar2(100);
v_deptno number(10) := 10;
begin

      loop
            v_sum := v_sum || 'sum(Decode(deptno,'|| v_deptno ||',sal,0), ';
            v_deptno := v_deptno + 10;
            exit when v_deptno = 40;
      end loop;

      v_sum := rtrim(v_sum,', ');

      dbms_output.put_line(v_sum);
```



```
end;  
/
```

```
SQL> exec pro116  
sum(Decode(deptno,10,sal,0), sum(Decode(deptno,20,sal,0),  
sum(Decode(deptno,30,sal,0)  
PL/SQL 처리가 정상적으로 완료되었습니다.
```

문제 117. 문제 116번 프로시저가 아래와 같이 출력될 수 있도록 코드를 수정 하시오. (select 추가)

```
create or replace procedure pro117  
is  
  
    v_sum varchar2(100) := 'select '  
    v_deptno number(10) := 10;  
    begin  
  
        loop  
            v_sum := v_sum || 'sum(Decode(deptno,'|| v_deptno ||',sal,0)), '  
            v_deptno := v_deptno + 10;  
            exit when v_deptno = 40;  
        end loop;  
  
        v_sum := rtrim(v_sum,', ');  
  
        dbms_output.put_line(v_sum);  
    end;  
/
```

```
SQL> exec pro117  
select sum(Decode(deptno,10,sal,0), sum(Decode(deptno,20,sal,0),  
sum(Decode(deptno,30,sal,0)  
PL/SQL 처리가 정상적으로 완료되었습니다.
```

문제 118. 컬럼의 맨 끝의 ', ' 를 제거 하시오.

```
create or replace procedure pro116  
is  
  
    v_sum varchar2(100);  
    v_deptno number(10) := 10;  
    begin  
  
        loop  
            v_sum := v_sum || 'sum(Decode(deptno,'|| v_deptno ||',sal,0)), '  
            v_deptno := v_deptno + 10;  
            exit when v_deptno = 40;
```

```

end loop;

v_sum := rtrim(v_sum,', ');

dbms_output.put_line(v_sum);
end;
/

```

```

SQL> exec pro116
sum(Decode(deptno,10,sal,0), sum(Decode(deptno,20,sal,0),
sum(Decode(deptno,30,sal,0)
PL/SQL 처리가 정상적으로 완료되었습니다.

```

문제 119. 문제 118번 프로시저가 아래와 같이 출력될 수 있도록 코드를 수정 하시오.

```

create or replace procedure pro118
is

v_sum varchar2(200) := 'select ';
v_deptno number(10) := 10;
begin

loop
    v_sum := v_sum || 'sum(Decode(deptno,'|| v_deptno ||',sal,0)), ';
    v_deptno := v_deptno + 10;
    exit when v_deptno = 40;
end loop;

v_sum := rtrim(v_sum,', ');
v_sum := v_sum|| ' from emp';

dbms_output.put_line(v_sum);
end;
/

```

```

SQL> exec pro118
select sum(Decode(deptno,10,sal,0), sum(Decode(deptno,20,sal,0),
sum(Decode(deptno,30,sal,0) from emp
PL/SQL 처리가 정상적으로 완료되었습니다.

```

문제 120. 문제 119번 프로시저가 아래와 같이 출력될 수 있도록 코드를 수정 하시오.

```

create or replace procedure pro120
is

v_sum varchar2(200) := 'select ';

```

```

v_deptno number(10) := 10;
begin

loop
    v_sum := v_sum || 'sum(Decode(deptno,|| v_deptno ||',sal,0)) ' || ""||v_deptno||"" ', ';
    v_deptno := v_deptno + 10;
    exit when v_deptno = 40;
end loop;

    v_sum := rtrim(v_sum,', ');
    v_sum := v_sum||' ' from emp';

    dbms_output.put_line(v_sum);
end;
/

```

```

SQL> exec pro120
select sum(Decode(deptno,10,sal,0) "10", sum(Decode(deptno,20,sal,0) "20",
sum(Decode(deptno,30,sal,0) "30" from emp
PL/SQL 처리가 정상적으로 완료되었습니다.

```

문제 121. 문제 120번에서 완성한 아래의 sql을 프로시저에서 실행되게 하시오.

```

create or replace procedure pp121
(v_result out sys_refcursor)--프로시저로 인해서 실행된 결과를 이 변수에 담을 것
is
    v_stmt varchar(200) := 'select ';
    v_dept number(10) := 0 ;

begin

    loop
        v_dept := v_dept+10;
        exit when v_dept = 40;
        v_stmt:= v_stmt||' sum(decode(deptno,||v_dept||',sal,0) )||""||v_dept||""';
    end loop;

    v_stmt := rtrim(v_stmt,',');
    v_stmt := v_stmt||' from emp';

    dbms_output.put_line(v_stmt);
    open v_result for v_stmt; --SQL 문장의 결과메모리를 담기 위해 v_result를 열겠다.

end;
/

```

```
SQL> variable v_rst2 sys_refcursor
사용법: VAR[iable] [ <변수> [ NUMBER | CHAR | CHAR <n [CHAR|BYTE]> |
          VARCHAR2 <n [CHAR|BYTE]> | NCHAR | NCHAR <n> |
          NVARCHAR2 <n> | CLOB | NCLOB | BLOB | BFILE
          REFCURSOR | BINARY_FLOAT | BINARY_DOUBLE ] ]
SQL> variable v_result2 refcursor;
```

```
SQL> exec ppl21(:v_result2);

PL/SQL 처리가 정상적으로 완료되었습니다.

SQL> print;
```

10	20	30
8750	10875	9400

문제 122. 아래의 결과를 출력하는 프로시저를 생성 하시오.

create or replace procedure pro122

(v_result out sys_refcursor)--프로시저로 인해서 실행된 결과를 이 변수에 담을 것
is

v_stmt varchar(200) := 'select job,;
v_dept number(10) := 0 ;

begin

loop
v_dept := v_dept+10;
exit when v_dept = 40;
v_stmt:= v_stmt||' sum(decode(deptno,'||v_dept||',sal,0))'||''||v_dept||''';
end loop;

v_stmt := rtrim(v_stmt,',');
v_stmt := v_stmt||' from emp group by job';
dbms_output.put_line(v_stmt);
open v_result for v_stmt; --SQL 문장의 결과메모리를 담기 위해 v_result를 열겠다.

end;
/

```
SQL> variable v_rst2 sys_refcursor
사용법: VAR[iable] [ <변수> [ NUMBER | CHAR | CHAR <n [CHAR|BYTE]> |
          VARCHAR2 <n [CHAR|BYTE]> | NCHAR | NCHAR <n> |
          NVARCHAR2 <n> | CLOB | NCLOB | BLOB | BFILE
          REFCURSOR | BINARY_FLOAT | BINARY_DOUBLE ] ]
SQL> variable v_result2 refcursor;
```

```
SQL> @p122
프로시저가 생성되었습니다.

SQL> exec pro122<:v_result2>;

PL/SQL 처리가 정상적으로 완료되었습니다.

SQL> print;
```

JOB	10	20	30
SALESMAN	0	0	5600
CLERK	1300	1900	950
PRESIDENT	5000	0	0
MANAGER	2450	2975	2850
ANALYST	0	6000	0

문제 123. 통신사를 중복없이 출력 하시오.

```
SELECT DISTINCT telecom
FROM EMP2;
```

	TELECOM
1	sk
2	lg
3	cjh
4	kt

문제 124. 아래의 결과를 출력하는 프로시저를 생성 하시오.

```
create or replace procedure pro123
(v_result out sys_refcursor)
is
    v_stmt varchar2(300):= 'select age,';
begin
    for v_cnt in (select distinct telecom from emp2) loop
        v_stmt := v_stmt || ' sum(decode(telecom,'" || v_cnt.telecom || "',1,0)) as '" || v_cnt.telecom || ' ' ;
    end loop;

    v_stmt := rtrim(v_stmt,', ');
    v_stmt := v_stmt || ' from emp2 group by age order by age asc';

    dbms_output.put_line(v_stmt);

    open v_result for v_stmt;

end;
/
```

```
SQL> variable v_result refcursor;
SQL> exec pro123(:v_result);
select age, sum(decode(telecom,'sk',1,0)) as sk, sum(decode(telecom,'lg',1,0))
as lg, sum(decode(telecom,'cjh',1,0)) as cjh, sum(decode(telecom,'kt',1,0)) as
kt from emp2 group by age order by age asc
```

PL/SQL 처리가 정상적으로 완료되었습니다.

```
SQL> print v_result;
```

AGE	SK	LG	CJH	KT
24	1	0	0	2
25	2	1	0	2
26	1	2	0	2
27	2	2	0	3
28	2	2	1	3
29	0	1	0	0
32	0	1	0	0

7 개의 행이 선택되었습니다.