/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

레코드(Record)

- 복합형 구조의(서로다른) 데이터 타입. 하나의 행(Row) 값을 표현하는데 사용된다.

- 필드(Field)

- 레코드를 구성 요소로 실제 value를 가지고 있다.

- 레코드명.필드명 으로 접근한다.

- 기존 테이블을 이용한 레코드

- 사용자 정의 레코드

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

- 기존 테이블을 이용한 레코드

- %ROWTYPE 속성

- 테이블이름%ROWTYPE : 기존 테이블이나 View의 컬럼을 필드로 가지는 레코드를 생성

- 해당 테이블이나 VIEW의 모든 컬럼을 필드로 가진다.

- 구문

변수선언: 변수명 테이블명%rowtype;

- 조회나 변경대상은 레코드의 필드들의 값이다.

레코드의 필드 사용 : 변수명.필드명

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

declare

rec\_emp emp%rowtype; --레코드타입 변수 : rec

--필드의 순서는 테이블 만들때 지정한 컬럼순서

begin

select \*

into rec\_emp

from emp

where emp\_id = 100;

--레코드변수명.필드명

dbms\_output.put\_line(rec\_emp.emp\_id||' - '||rec\_emp.emp\_name||' - '||rec\_emp.salary);

end;

/

-- ROWTYPE을 이용한 DML

drop table emp\_copy;

create table emp\_copy as select \* from emp where dept\_id =100 ;

declare

rec\_emp emp%rowtype;

begin

select \*

into rec\_emp

from emp

where emp\_id = 140;

insert into emp\_copy values rec\_emp; --필드명-컬럼명이 일치하는 컬럼에 값을 insert

--레코드 필드의 값을 변경

rec\_emp.salary := 30000;

rec\_emp.comm\_pct := 0.3;

rec\_emp.hire\_date := '2019-07-22';

update emp\_copy

set row = rec\_emp --전체 컬럼의 값을 레코드가 가진 값으로 변경.

where emp\_id = rec\_emp.emp\_id;

end;

/

select \* from emp\_copy;

declare

rec\_emp emp%rowtype;

begin

select emp\_id, emp\_name, salary

into rec\_emp.emp\_id, rec\_emp.emp\_name, rec\_emp.salary

from emp

where emp\_id = 110;

dbms\_output.put\_line(rec\_emp.emp\_id||' - '||rec\_emp.emp\_name||' - '||rec\_emp.salary);

end;

/

--아래 TODO의 조회 결과를 담을 변수는 %ROWTYPE 레코드로 선언.

-- TODO : JOB 테이블에서 JOB\_ID가 'AD\_PRES' 인 행을 조회해 출력.

declare

rec\_job job%rowtype;

begin

select \*

into rec\_job

from job

where job\_id = 'AD\_PRES';

dbms\_output.put\_line(rec\_job.job\_id||' - '||rec\_job.job\_title);

end;

/

select \* from job;

-- TODO : CUSTOMERS 테이블에서 고객\_ID(cust\_id), 이름(cust\_name), 주소(address),

--이메일주소(cust\_email)을 조회하여 출력하는 코드 작성.

declare

rec\_cust customers%rowtype;

begin

select cust\_id, cust\_name, address, cust\_email

into rec\_cust.cust\_id, rec\_cust.cust\_name, rec\_cust.address, rec\_cust.cust\_email

from customers

where cust\_id = 100;

dbms\_output.put\_line(rec\_cust.cust\_id||' - '||rec\_cust.cust\_name||' - '||rec\_cust.address||' - '||rec\_cust.cust\_email);

end;

/

select \* from customers;

-- DEPT\_COPY 테이블을 생성 (모든 데이터 카피)

-- TODO : DEPT\_COPY에서 DEPT\_ID가 100인 부서 정보를 출력하는 코드 작성.

drop table dept\_copy;

create table dept\_copy

as select \* from dept;

declare

rec\_dept dept%rowtype;

begin

select \*

into rec\_dept

from dept

where dept\_id = 100;

dbms\_output.put\_line(rec\_dept.dept\_id||'-'||rec\_dept.dept\_name||'-'||rec\_dept.loc);

end;

/

create or replace procedure ex\_dept\_copy\_sp(p\_dept\_id dept\_copy.dept\_id%type)

is

rec\_dept\_copy dept\_copy%rowtype;

begin

select \*

into rec\_dept\_copy

from dept\_copy

where dept\_id = p\_dept\_id;

dbms\_output.put\_line(rec\_dept\_copy.dept\_id||'-'||rec\_dept\_copy.dept\_name||'-'||rec\_dept\_copy.loc);

end;

/

exec ex\_dept\_copy\_sp(&id);

-- TODO : DEPT\_COPY 의 ROWTYPE 레코드 변수를 선언 하고 그 변수의 필드에 적당한 값을 대입한 뒤

--INSERT 하는 코드를 작성.

declare

rec\_dept\_copy dept\_copy%rowtype;

begin

rec\_dept\_copy.dept\_id := 4701;

rec\_dept\_copy.dept\_name := '부서명';

rec\_dept\_copy.loc := '서울';

insert into dept\_copy values rec\_dept\_copy;

ex\_dept\_copy\_sp(rec\_dept\_copy.dept\_id);

rec\_dept\_copy.loc := '대전';

update dept\_copy

set row = rec\_dept\_copy

where dept\_id = rec\_dept\_copy.dept\_id;

ex\_dept\_copy\_sp(rec\_dept\_copy.dept\_id); --변경 결과 조회

end;

/

select \* from dept\_copy;

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

사용자 정의 레코드

- 레코드 타입을 직접 선언

- 변수 선언시 데이터 타입으로 사용한다.

- 구문

TYPE 레코드이름 IS RECORD (

필드명 필드타입 [NOT NULL] [ := 디폴트값] ,

...

);

-필드타입(데이터 타입)

-PL/SQL 데이터타입

-%type

-%rowtype

-사용

변수명 레코드이름;

type 이름 is 종류

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

declare

--부서id와 부서이름을 저장할 수 있는 record

type dept\_type is record( --레코드타입 이름: 이름\_type

-- id number(4),

-- name varchar2(100)

id dept.dept\_id%type,

name dept.dept\_name%type

);

rec\_dept dept\_type; --ID/NAME을 필드로 가지는 레코드

begin

select dept\_id, dept\_name

into rec\_dept

from dept

where dept\_id = 10;

dbms\_output.put\_line(rec\_dept.id||' - '||rec\_dept.name);

rec\_dept.id := 20;

rec\_dept.name := '영업기획부';

update dept

set dept\_name = rec\_dept.name

where dept\_id = rec\_dept.id;

end;

/

select \*from dept;

--TODO: 아래 select 문 실행결과를 저장할 사용자 정의 레코드를 만들고 SQL실행결과를 그 타입의 변수에 저장한뒤 출력하는 프로시저를 작성.

/\*

select e.emp\_id, e.emp\_name, d.dept\_id, d.dept\_name, j.job\_title

from emp e, dept d, job j

where e.dept\_id = d.dept\_id(+)

and e.job\_id = j.job\_id(+)

and e.emp\_id = 100;

\*/

declare

type todo\_type is record(

emp\_id emp.emp\_id%type,

emp\_name emp.emp\_name%type,

dept\_id dept.dept\_id%type,

dept\_name dept.dept\_name%type,

job\_title job.job\_title%type

);

rec\_todo todo\_type;

begin

select e.emp\_id, e.emp\_name, d.dept\_id, d.dept\_name, j.job\_title

into rec\_todo

from emp e, dept d, job j

where e.dept\_id = d.dept\_id(+)

and e.job\_id = j.job\_id(+)

and e.emp\_id = 100;

dbms\_output.put\_line(rec\_todo.emp\_id||' - '||rec\_todo.emp\_name||' - '||rec\_todo.dept\_id

||' - '||rec\_todo.dept\_name||' - '||rec\_todo.job\_title);

end;

/

declare

type join\_type is record(

rec\_emp emp%rowtype,

rec\_dept dept%rowtype,

rec\_job job%rowtype

);

rec\_join join\_type;

begin

select e.emp\_id, e.emp\_name, d.dept\_id, d.dept\_name, j.job\_title

into rec\_join.rec\_emp.emp\_id,

rec\_join.rec\_emp.emp\_name,

rec\_join.rec\_dept.dept\_id,

rec\_join.rec\_dept.dept\_name,

rec\_join.rec\_job.job\_title

from emp e, dept d, job j

where e.dept\_id = d.dept\_id(+)

and e.job\_id = j.job\_id(+)

and e.emp\_id = 100;

dbms\_output.put\_line(rec\_join.rec\_emp.emp\_id);

end;

/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

컬렉션

- 같은 타입의 값들을 여러개 저장하는 구조.(열)

- 레코드는 다양한 타입의 필드를 모으지만 테이블 처럼 여러 ROW를 가질 수 없음.

- 컬렉션은 여러 row를 가지는 형태

- 종류

1. 연관배열: 키와 값으로 구성된 컬렉션

2. VARRAY : 크기가 고정되 있는 배열구조

3. 중첩테이블 : 크기를 자유롭게 정할 수있는 배열구조

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

- 연관배열(Associative Array)

- 키-값 쌍으로 구성된 컬렉션

- 키를 INDEX로 부르기 때문에 Index-by 테이블이라고도 한다.

- 변수의 타입으로 사용된다.

- 구문

- 타입 정의

- TYPE 이름 IS TABLE OF 값타입 INDEX BY 키(인덱스)타입

- 값타입 : 모든 타입 가능

- 키(인덱스) 타입 : 문자형 또는 PLS\_INTEGER/BINARY\_INTEGER 타입만 가능.

- 선언부에 정의한다.

- 타입을 정의 한뒤 변수의 타입으로 사용할 수 있다.

- 값 대입 : 배열이름(INDEX):=값

- 값 조회 : 배열이름(INDEX)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

declare

-- key(index) 타입: 정수

-- 값(value) 타입: varchar2(100)

-- 연관배열 타입이름 : 이름\_table\_type

type my\_table\_type is table of varchar2(100)

index by binary\_integer;

--변수 선언

my\_table my\_table\_type;

begin

--연관배열에 값 대입

my\_table(10) := 'text1';

my\_table(20) := 'text2';

dbms\_output.put\_line(my\_table(10));

dbms\_output.put\_line(my\_table(20));

end;

/

declare

-- index(key) : dept\_id, value: dept\_name

type dept\_name\_table\_type is table of dept.dept\_name%type

index by binary\_integer; --index의 타입 유의! number타입은 올 수 없다.

dept\_name\_table dept\_name\_table\_type;

begin

select dept\_name

into dept\_name\_table(10) --select결과를 연관배열에 추가

from dept

where dept\_id = 10;

select dept\_name

into dept\_name\_table(20) --select결과를 연관배열에 추가

from dept

where dept\_id = 20;

dbms\_output.put\_line(dept\_name\_table(10));

dbms\_output.put\_line(dept\_name\_table(20));

end;

/

declare

--index: 정수(dept\_id), value: 부서레코드(한 행)

type dept\_table\_type is table of dept%rowtype

index by binary\_integer;

dept\_table dept\_table\_type;

begin

select \*

into dept\_table(10)

from dept

where dept\_id = 10;

select \*

into dept\_table(20)

from dept

where dept\_id = 20;

dbms\_output.put\_line(dept\_table(10).dept\_name||'-'||dept\_table(10).loc);

dbms\_output.put\_line(dept\_table(20).dept\_name||'-'||dept\_table(20).loc);

end;

/

--TODO

-- 1. index는 크기 10인 문자열로 값은 date를 가지는 연관배열 선언

-- 2. index는 정수, 값은 dept 테이블의 loc와 같은 타입인 연관배열 선언

declare

--1번 선언

type var1\_table\_type is table of date

index by varchar2(10);

--2번 선언

type var2\_table\_type is table of dept.loc%type

index by binary\_integer;

var1\_table var1\_table\_type;

var2\_table var2\_table\_type;

begin

--1번 타입의 변수에 값 대입 및 출력

var1\_table('홍길동') := '2010-10-20';

var1\_table('이순신') := to\_date('2000-10','yyyy-mm');

dbms\_output.put\_line(var1\_table('홍길동'));

dbms\_output.put\_line(var1\_table('이순신'));

--2번 타입의 변수에 값 대입 및 출력

var2\_table(10) := '서울';

var2\_table(20) := '부산';

dbms\_output.put\_line(var2\_table(10));

dbms\_output.put\_line(var2\_table(20));

end;

/

--TODO 부서 ID를 Index로 부서 레코드(%ROWTYPE)을 Value로 가지는 연관배열을 선언하고

-- dept\_id가 10인 부서와 dept\_id가 20 인 부서의 정보(dept\_id, dept\_name, loc)를 조회하여 연관배열 변수에 넣는 프로시저를 작성

declare

type dept\_table\_type is table of dept%rowtype

index by binary\_integer;

dept\_type dept\_table\_type; --부서 정보들을 모으는 컬렉션.

begin

select \*

into dept\_type(10)

from dept

where dept\_id = 10;

select \*

into dept\_type(20)

from dept

where dept\_id = 20;

dbms\_output.put\_line(dept\_type(10).dept\_id||'-'||dept\_type(10).dept\_name||'-'||dept\_type(10).loc);

dbms\_output.put\_line(dept\_type(20).dept\_id||'-'||dept\_type(20).dept\_name||'-'||dept\_type(20).loc);

end;

/

-- TODO: emp\_id 가 100 ~ 120번인 직원의 이름을 조회해 연관배열(index: emp\_id, value: emp\_name)에 저장하는 코드를 작성.

declare

type emp\_table\_type is table of emp.emp\_name%type

index by binary\_integer;

emp\_type emp\_table\_type;

begin

--반복문을 이용해 100 ~ 120 인 직원의 이름을 조회 및 연관배열에 저장가능

for idx in 100..120

loop

select emp\_name

into emp\_type(idx)

from emp

where emp\_id = idx;

end loop;

--연관배열내의 조회결과 출력

for idx in 100..120

loop

dbms\_output.put\_line(idx||emp\_type(idx));

end loop;

end;

/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

- VARRAY(Variable-Size Array)

- 고정길이 배열.

- 선언시 배열의 크기(최대요소개수)를 지정하면 그 크기 만큼의 요소만 저장가능

- 선언한 개수 이상 저장 할 수 없지만 적게 저장하는 것은 가능.

- Index는 1부터 1씩 증가하는 정수로 자동으로 설정된다.

- 생성자를 통한 초기화

- 반드시 생성자를 이용해 값들을 저장한 뒤에사용할 수 있다.

- 일반 적으로 연속적인 값을 저장해야 하는 경우 많이 사용됨(index가 순번 형태이므로)

- C 나 Java의 배열과 비슷한 형태이다.

- 구문

- 타입 정의

- TYPE 이름 IS VARRAY(최대크기) OF 요소값타입;

- 초기화

- 변수 := 이름(값 [, ...])

- 초기화 된 개수만큼만 변경 가능

- 값 대입(변경) : 이름(INDEX):=값

- 값 조회 : 이름(INDEX)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

declare

type va\_type is varray(5) of varchar2(100);

va\_arr va\_type;

begin

--초기화

va\_arr := va\_type('가', '나', '다', '라'); --위에서 최대크기를 5개로 지정하였어도 조기화를 4개를 하면 크기는 4로 고정된다.

va\_arr(1) := 'a';

dbms\_output.put\_line(va\_arr(1)||va\_arr(2)||va\_arr(3)||va\_arr(4));

end;

/

--dept\_id = 10, 50, 70인 부서의 정보를 조회 =>연관배열의

declare

--결과를 담을 연관배열타입

type dept\_table\_type is table of dept%rowtype

index by binary\_integer;

--조회할 부서 id들을 담을 varray

type dept\_id\_list is varray(3) of dept.dept\_id%type;

dept\_table dept\_table\_type;

dept\_ids dept\_id\_list;

begin

dept\_ids := dept\_id\_list(10,50,70); --조회할 부서id로 varray 초기화

for idx in 1..3

loop

select \*

into dept\_table(dept\_ids(idx))

from dept

where dept\_id = dept\_ids(idx);

end loop;

for idx in 1..3

loop

dbms\_output.put\_line(dept\_table(dept\_ids(idx)).dept\_name);

end loop;

end;

/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

- 중첩테이블(Nested Table)

- 선언시 크기를 지정하지 않고 생성자를 이용해 초기화 할때 대입된 원소 수에 맞춰 크기가 정해진다.

- 생성자를 사용한 초기화 후 사용

- index 는 1부터 1씩 자동증가하는 정수로 설정.

- 일반 테이블의 컬럼 타입으로 사용될 수 있다.

- 구문

- 타입 정의

- TYPE 이름 IS TABLE OF 값타입;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

declare

type nt\_table\_type is table of number;

nt\_table nt\_table\_type;

nt\_table2 nt\_table\_type;

begin

--생성자를 이용한 초기화

nt\_table := nt\_table\_type(10,20,30,40,50);

nt\_table2 := nt\_table\_type(1,2,3,4,5,6,7,8,9,10);

nt\_table(2):=2000; --대입(변경)

-- for idx in 1..5

for idx in 1..nt\_table.count

loop

dbms\_output.put\_line(nt\_table(idx));

end loop;

end;

/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

- 컬렉션 메소드

# DELETE : 모든 요소 삭제

# DELETE(n) : index가 n인 요소 삭제 (varray는 지원안함)

# DELETE(n, m) : index가 n ~ m 인 요소 삭제 (varray는 지원안함)

# EXISTS(index) : index가 있는 지 여부 boolean값으로 반환

# FIRST : 첫번째 IDNEX 반환

# LAST : 마지막 INDEX 반환

- FOR idx IN 컬렉션.FIRST..컬렉션.LAST

# PRIOR(index) : index 이전 INDEX 반환

# NEXT(index) : index 다음 INDEX 반환

# COUNT: 컬렉션내의 요소 개수 반환

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

declare

type varr\_type is varray(5) of varchar2(10);

varr varr\_type;

begin

varr := varr\_type('a','b','c','d');

dbms\_output.put\_line(varr.first);

dbms\_output.put\_line(varr.last);

dbms\_output.put\_line(varr.next(3));

dbms\_output.put\_line(varr.prior(3));

if varr.exists(3) then

dbms\_output.put\_line('index 있음');

end if;

end;

/

-- TODO: emp에서 emp\_id가 100 ~ 120인 직원들의 정보를 조회한 뒤 그 정보를 emp\_copy에 추가하는 코드를 작성.

DECLARE

-- 연관배열 타입 정의(조회한 직원정보들 저장): key - emp\_id타입, value:emp row

type emp\_table\_type is table of emp%rowtype

index by binary\_integer;

-- 연관배열 타입 변수 선언

emp\_table emp\_table\_type;

BEGIN

-- 반복문을 이용해 emp\_id가 100 ~ 120 인 직원 조회해서 연관배열에 저장

for idx in 100..120

loop

select \*

into emp\_table(idx)

from emp

where emp\_id = idx;

end loop;

-- 반복문을 이용해 연관배열내 저장된 조회결과들을 emp\_copy 테이블에 insert (FIRST, LAST 메소드 이용)

for idx in emp\_table.first..emp\_table.last

loop

insert into emp\_copy values emp\_table(idx);

end loop;

commit;

END;

/

drop table emp\_copy;

create table emp\_copy as select \* from emp where 1=0;

select \* from emp\_copy;