ch17. 레코드와 컬렉션

17-1. 자료형이 다른 여러 데이터를 저장하는 레코드

레코드란?

레코드 *record* 는 자료형이 각기 다른 데이터를 하나의 변수에 저장하는 데 사용한다.

```
TYPE 레코드 이름 IS RECORD(
  변수 이름 자료형 NOT NULL := (또는 DEFAULT) 값 또는 값이 도출되는 여러 표현식
)
```

- 정의한 레코드는 지금까지 다룬 변수와 마찬가지로 기존 자료형처럼 사용할 수 있다.
- 레코드에 포함된 변수는 레코드 이름과 마침표로 사용할 수 있다.

- 레코드 정의해서 사용하기

```
DECLARE
   TYPE REC_DEPT IS RECORD(
      deptno NUMBER(2) NOT NULL := 99,
      dname DEPT.DNAME%TYPE,
      loc DEPT.LOC%TYPE
   );
   dept_rec REC_DEPT;
BEGIN
   dept_rec.deptno := 99;
   dept_rec.dname := 'DATABASE';
   dept_rec.loc := 'SEOUL';
   DBMS_OUTPUT.PUT_LINE('DEPTNO : ' || dept_rec.deptno);
   DBMS_OUTPUT.PUT_LINE('DNAME : ' || dept_rec.dname);
   DBMS_OUTPUT.PUT_LINE('LOC : ' || dept_rec.loc);
END;
/
```

레코드를 사용한 INSERT

PL/SQL문에서는 테이블에 데이터를 삽입하거나 수정하는 INSERT, UPDATE문에도 레코드를 사용할 수 있다.

- DEPT_RECORD 테이블 생성하기

```
CREATE TABLE DEPT_RECORD
    AS SELECT * FROM DEPT;
```

- DEPT_RECORD 테이블 생성하기(생성된 테이블 조회)

```
SELECT * FROM DEPT_RECORD;
```

- 기존 INSERT문에서는 삽입할 데이터를 VALUES절에 하나씩 명시하였는데, INSERT 문에 레코드를 사용하면 VALUES절에 레코드 이름만 명시해도 된다.
    - 선언한 레코드와 INSERT 대상이 되는 테이블의 데이터 개수, 자료형, 순서를 맞추어야 한다.

- 레코드를 사용하여 INSERT하기

```
DECLARE
   TYPE REC_DEPT IS RECORD(
      deptno NUMBER(2) NOT NULL := 99,
      dname DEPT.DNAME%TYPE,
      loc DEPT.LOC%TYPE
   );
   dept_rec REC_DEPT;
BEGIN
   dept_rec.deptno := 99;
   dept_rec.dname := 'DATABASE';
   dept_rec.loc := 'SEOUL';

   INSERT INTO DEPT_RECORD
   VALUES dept_rec;
END;
/
```

- 레코드를 사용하여 INSERT하기 (테이블 조회)

```
SELECT * FROM DEPT_RECORD;
```

레코드를 사용한 UPDATE

- 레코드는 UPDATE문에도 사용할 수 있다.
- SET절은 ROW 키워드와 함께 레코드 이름을 명시한다.
- 기존 UPDATE 문에서 SET절을 통해 변경할 열을 하나하나 지정한 것과 달리, 레코드에 저장된 데이터를 사용하면 행 전체의 데이터를 바꿔줄 수 있다.

- 레코드를 사용하여 UPDATE하기

```
DECLARE
   TYPE REC_DEPT IS RECORD(
      deptno NUMBER(2) NOT NULL := 99,
      dname DEPT.DNAME%TYPE,
      loc DEPT.LOC%TYPE
   );
   dept_rec REC_DEPT;
BEGIN
   dept_rec.deptno := 50;
   dept_rec.dname := 'DB';
   dept_rec.loc := 'SEOUL';

   UPDATE DEPT_RECORD
      SET ROW = dept_rec
    WHERE DEPTNO = 99;
END;
/
```

- 레코드를 사용하여 UPDATE 하기 (테이블 조회)

```
SELECT * FROM DEPT_RECORD;
```

레코드를 포함하는 레코드

- 레코드에 다른 레코드 포함하기

```
DECLARE
   TYPE REC_DEPT IS RECORD(
      deptno DEPT.DEPTNO%TYPE,
      dname DEPT.DNAME%TYPE,
      loc DEPT.LOC%TYPE
   );
   TYPE REC_EMP IS RECORD(
      empno EMP.EMPNO%TYPE,
      ename EMP.ENAME%TYPE,
      dinfo REC_DEPT
   );
   emp_rec REC_EMP;
BEGIN
   SELECT E.EMPNO, E.ENAME, D.DEPTNO, D.DNAME, D.LOC
     INTO emp_rec.empno, emp_rec.ename,
          emp_rec.dinfo.deptno,
          emp_rec.dinfo.dname,
          emp_rec.dinfo.loc
     FROM EMP E, DEPT D
    WHERE E.DEPTNO = D.DEPTNO
      AND E.EMPNO = 7788;

   DBMS_OUTPUT.PUT_LINE('EMPNO : ' || emp_rec.empno);
   DBMS_OUTPUT.PUT_LINE('ENAME : ' || emp_rec.ename);
   DBMS_OUTPUT.PUT_LINE('DEPTNO : ' || emp_rec.dinfo.deptno);
   DBMS_OUTPUT.PUT_LINE('DNAME : ' || emp_rec.dinfo.dname);
   DBMS_OUTPUT.PUT_LINE('LOC : ' || emp_rec.dinfo.loc);
END;
/
```

- 레코드에 포함된 변수의 자료형을 지정할 때 다른 레코드를 지정할 수도 있다.
- REC_EMP 레코드는 dinfo 변수에 REC_DEPT 레코드를 자료형으로 지정한다.
- 변수에 레코드형을 적용했으므로 두 개의 마침표(.)로 값을 사용한다.
- 중첩 레코드 *nested record* : 레코드 안에 또 다른 레코드를 포함한 형태

17-2. 자료형이 같은 여러 데이터를 저장하는 컬렉션

- 컬렉션 : 특정 자료형의 데이터를 여러 개 저장하는 복합 자료형
- 여러 종류의 데이터를 하나로 묶어 사용하는 레코드를 테이블의 한 행처럼 사용한다면, 컬렉션은 열 또는 테이블과 같은 형태로 사용 가능.

- PL/SQL에서 사용할 수 있는 컬렉션
  - 연관 배열 *associative array (index by table)*
  - 중첩 테이블 *nested table*
  - VARRY *varriable-size array*

연관 배열

연관 배열 : 인덱스라고도 불리는 키 *key*, 값 *value*으로 구성되는 컬렉션

- 중복되지 않은 유일한 키를 통해 값을 저장하고 불러오는 방식을 사용

```
TYPE 연관 배열 이름 IS TABLE OF 자료형 [NOT NULL]
INDEX BY 인덱스형;
```

자료형 ⇒ 단일 자료형 , 참조 자료형

인덱스형 ⇒ 키로 사용할 인덱스 자료형 지정 / 정수, 문자 자료형

- 정의한 연관 배열은 레코드와 마찬가지로 특정 변수의 자료형으로서 사용할 수 있다.

- 연관배열 사용하기

```
DECLARE
   TYPE ITAB_EX IS TABLE OF VARCHAR2(20)
      INDEX BY PLS_INTEGER;

   text_arr ITAB_EX;

BEGIN
   text_arr(1) := '1st data';
   text_arr(2) := '2nd data';
   text_arr(3) := '3rd data';
   text_arr(4) := '4th data';

   DBMS_OUTPUT.PUT_LINE('text_arr(1) : ' || text_arr(1));
   DBMS_OUTPUT.PUT_LINE('text_arr(2) : ' || text_arr(2));
   DBMS_OUTPUT.PUT_LINE('text_arr(3) : ' || text_arr(3));
   DBMS_OUTPUT.PUT_LINE('text_arr(4) : ' || text_arr(4));
```

```
END;
/
```

레코드를 활용한 연관 배열

- 연관 배열의 자료형에는 레코드를 사용할 수 있다.
  - 다양한 자료형을 포함한 레코드를 여러 개 사용할 수 있으므로 테이블과 같은 데이터 사용과 저장이 가능하다.

- 연관 배열 자료형에 레코드 사용하기

```
DECLARE
   TYPE REC_DEPT IS RECORD(
      deptno DEPT.DEPTNO%TYPE,
      dname DEPT.DNAME%TYPE
   );

   TYPE ITAB_DEPT IS TABLE OF REC_DEPT
      INDEX BY PLS_INTEGER;

   dept_arr ITAB_DEPT;
   idx PLS_INTEGER := 0;

BEGIN
   FOR i IN (SELECT DEPTNO, DNAME FROM DEPT) LOOP
      idx := idx + 1;
      dept_arr(idx).deptno := i.DEPTNO;
      dept_arr(idx).dname := i.DNAME;

      DBMS_OUTPUT.PUT_LINE(
         dept_arr(idx).deptno || ' : ' || dept_arr(idx).dname);
   END LOOP;
END;
/
```

- 만약 특정 테이블의 전체 열과 같은 구성을 가진 연관 배열을 제작한다면, 밑의 예시와 같이 %ROWTYPE을 사용하는 것이 레코드를 정의하는 것보다 간편하다.

```
DECLARE
   TYPE ITAB_DEPT IS TABLE OF DEPT%ROWTYPE
      INDEX BY PLS_INTEGER;

   dept_arr ITAB_DEPT;
   idx PLS_INTEGER := 0;
```

```
BEGIN
   FOR i IN(SELECT * FROM DEPT) LOOP
      idx := idx + 1;
      dept_arr(idx).deptno := i.DEPTNO;
      dept_arr(idx).dname := i.DNAME;
      dept_arr(idx).loc := i.LOC;

      DBMS_OUTPUT.PUT_LINE(
      dept_arr(idx).deptno || ' : ' ||
      dept_arr(idx).dname || ' : ' ||
      dept_arr(idx).loc);
   END LOOP;
END;
/
```

컬렉션 메서드

- 오라클에서는 컬렉션 사용상의 편의를 위해 서브프로그램을 재공한다 = 컬렉션 메서드
- 컬렉션 메서드는 컬렉션과 관련된 다양한 정보 조회 기능을 제공.
- 컬렉션 내의 데이터 삭제나 컬렉션 크기 조절을 위한 특정 조작도 가능.
- 컬렉션 메서드는 컬렉션형으로 선언한 변수에 마침표와 함께 작성하여 사용.

Q

1.

```
CREATE TABLE EMP_RECORD
    AS SELECT *
         FROM EMP
        WHERE 1<>1;
```

```
DECLARE
   TYPE REC_EMP IS RECORD (
      empno    EMP.EMPNO%TYPE NOT NULL := 9999,
      ename    EMP.ENAME%TYPE,
      job      EMP.JOB%TYPE,
      mgr      EMP.MGR%TYPE,
      hiredate EMP.HIREDATE%TYPE,
```

```
      sal      EMP.SAL%TYPE,
      comm     EMP.COMM%TYPE,
      deptno   EMP.DEPTNO%TYPE
   );
   emp_rec REC_EMP;
BEGIN
   emp_rec.empno    := 1111;
   emp_rec.ename    := 'TEST_USER';
   emp_rec.job      := 'TEST_JOB';
   emp_rec.mgr      := null;
   emp_rec.hiredate := TO_DATE('20180301','YYYYMMDD');
   emp_rec.sal      := 3000;
   emp_rec.comm     := null;
   emp_rec.deptno   := 40;

   INSERT INTO EMP_RECORD
   VALUES emp_rec;
END;
/
```

2.

```
DECLARE
   TYPE ITAB_EMP IS TABLE OF EMP%ROWTYPE
      INDEX BY PLS_INTEGER;
   emp_arr ITAB_EMP;
   idx PLS_INTEGER := 0;
BEGIN
   FOR i IN (SELECT * FROM EMP) LOOP
      idx := idx + 1;
      emp_arr(idx).empno    := i.EMPNO;
      emp_arr(idx).ename    := i.ENAME;
      emp_arr(idx).job      := i.JOB;
      emp_arr(idx).mgr      := i.MGR;
      emp_arr(idx).hiredate := i.HIREDATE;
      emp_arr(idx).sal      := i.SAL;
      emp_arr(idx).comm     := i.COMM;
      emp_arr(idx).deptno   := i.DEPTNO;

      DBMS_OUTPUT.PUT_LINE(
         emp_arr(idx).empno     || ' : ' ||
         emp_arr(idx).ename     || ' : ' ||
         emp_arr(idx).job       || ' : ' ||
         emp_arr(idx).mgr       || ' : ' ||
         emp_arr(idx).hiredate  || ' : ' ||
         emp_arr(idx).sal       || ' : ' ||
         emp_arr(idx).comm      || ' : ' ||
         emp_arr(idx).deptno);

   END LOOP;
END;
/
```