Tipuri de date compuse

Tipul de date RECORD

Tipul de date înregistrare sau RECORD ne permite să lucrăm mult mai ușor cu tabelele din baza de date, fapt datorat structurilor similare dintre aceste obiecte. Un obiect de tipul record este format dintr-un număr variabil de câmpuri care au și ele la rândul lor câte un tip de date specific. Astfel printr-un obiect record putem reține întru-totul o înregistrare dintr-un anumit tabel.

Un tip de date record trebuie declarat explicit în secțiunea declare a blocului PL/SQL, cu tot cu tipurile de date ale câmpurilor din componența sa. Mai apoi se pot declara variabile cu acest tip și ele pot fi utilizate pentru a salva o anumită intrare dintr-un tabel (prin comenzi SQL de tipul SELECT INTO) sau pentru a insera sau actualiza o intrare deja existentă în tabel.

```
DECLARE
  TYPE intrare IS RECORD
  ( nume VARCHAR2(50),
  email VARCHAR2(50),
  varsta NUMBER(3)
);
```

Tipurile de date COLLECTION

Tipurile de date collection permit stocarea de informații sub forma unor mulțimi care au elemente de tip omogen prestabilit în faza de declarare. Aceste mulțimi pot avea proprietăți diferite implementate în PL/SQL sub forma a 3 subtipuri: Index-by Tables, Nested Tables și Variable-size Arrays (numite și vectori).

Subtipurile derivate din tipul collection beneficiază de diverse metode care permit programatorului să manipuleze mai ușor datele din cadrul acestora. Acestea trebuie însă să fie utilizate cu atenție întrucât nu toate metodele pot fi aplicate oricărui subtip de date. O scurtă prezentare a câtorva metode:

- EXISTS(n), verifică dacă la indexul n se află un element nenul
- COUNT, dă numărul de elemente din colecție
- FIRST, LAST, dau indexul elementelor aflate pe pozițiile de început și sfârșit
- DELETE(n,m), șterge elementele dintre pozițiile n și m, sau elementul de la n dacă m nu este specificat, sau toate elementele dacă nu sunt indicați parametrii. Se poate folosi pe varrays doar fără parametrii
- EXTEND(n,i), adaugă n copii ale elementului de pe poziția i la finalul colecției. Dacă i nu este specificat se adauga elemente null. Nu poate fi folosită pe tablourile indexate
- NEXT(n), PRIOR(n), furnizează elementele de dinaintea sau de după elementul de pe poziția n

Tipul INDEX-BY TABLE

Tabloul indexat reprezintă implementarea în PL/SQL a conceptului de hash table prezent în alte limbaje de programare. Un tablou indexat este, mai exact, o mulțime de perechi cheie-valoare, cu cheia unică, care permite stocarea elementelor la anumiți indici. Indexarea se poate face cu aproape orice tip de date din PL/SQL, exceptând cel referențial.

Adăugarea elementelor în tabloul indexat se realizează prin atribuirea unei valori la o cheie, moment în care perechea este adăugată în tabloul existent. Colecția începe cu 0 elemente și dimensiunea ei crește la adăugarea fiecărei noi perechi. În cazul în care este accesat un element a cărui cheie nu există în tablou, se lansează eroarea NO_DATA_FOUND.

```
DECLARE
  TYPE tab_ind IS TABLE OF DATE INDEX BY VARCHAR2(50);
  zile_de_nastere tab_ind;
  v_elev VARCHAR2(50);
BEGIN
  FOR i IN 1..100 LOOP
    SELECT nume
    INTO v_elev
    FROM elevi
    WHERE id = i;
    SELECT data_de_nastere
    INTO zile_de_nastere(v_elev)
    FROM elevi
    WHERE id = i;
  END LOOP;
  DBMS_OUTPUT.PUT_LINE(zile_de_nastere.EXISTS('Popescu'));
  FOR i IN zile_de_nastere.FIRST..zile_de_nastere.LAST LOOP
    DBMS_OUTPUT.PUT_LINE(zile_de_nastere(i));
  END LOOP;
END;
```

Tipul de date NESTED TABLE

Tabloul imbricat este implementat în PL/SQL prin obiectele de tip NESTED TABLE și este structurat ca un multiset. Elementele unui astfel de tablou sunt stocate la anumiți indecși care pot fi generați automat (atunci când preluăm informații dintr-un tabel SQL) sau stabiliți de programator.

Declararea unui astfel de tablou se face, ca la record și index table, cu specificarea tipului în secțiunea declare. Mai apoi este necesară și utilizaera unui constructor la inițializarea fiecărei variabile de acest tip unde se pot adăuga primele elemente. Tablourile îmbricate trebuie extinse la adăgarea fiecărui nou element care duce la depășirea dimensiunii stabilite în constructor (deobicei se inițializează aceste obiecte cu constructor fără parametrii deci dimensiunea inițială va fi 0).

```
DECLARE
  TYPE tabel IS TABLE OF NUMBER(10);
  v_fibo tabel := tabel(1,1);
BEGIN
  FOR i IN 2..100 LOOP
    v_fibo.EXTEND
    v_fibo(i) := v_fibo(i – 1) + v_fibo(i – 2);
  END LOOP;
  DBMS_OUTPUT.PUT_LINE(v_fibo(99));
END;
```

Tipul de date VARRAY

Obiectele de tip varray, denumite și vectori, sunt similare cu cele de tip nested tables, însă diferă prin restricția plasată pe numărul de elemente stocate de acestea. La declarare se specifică dimensiunea maximă a unui varray, iar aceasta nu poate fi depășită în blocul de cod nici prin utilizarea funcției EXTEND asupra obiectului.

Declararea vectorilor se face ca la nested tables, cu inițializarea mai întâi a tipului și apoi a variabilelor cu ajutorul unui constructor. De asemenea este necesară utilizarea funcției EXTEND dacă vrem să inserăm elemente ce depășesc dimensiunea dată de constructor, chiar dacă aceasta este mai mică decât cea declarată ca limită a tipului.

```
DECLARE
  TYPE vect IS VARRAY(15) OF elevi.nume%TYPE;
  v_elevi vect := vect();
BEGIN
  v_elevi.EXTEND(15);
  SELECT nume
  BULK COLLECT INTO v_elevi
  FROM elevi
  WHERE ROWNUM <= 15;
  v_elevi.DELETE(1,9);
  FOR i IN v_elevi.FIRST..v_elevi.LAST LOOP
```

```
    DBMS_OUTPUT.PUT_LINE(v_elevi(i) + ` admis`);
  END LOOP;
END;
```

Exerciții

a) Să se citească de la tastatură id-ul unui manager. Dacă acesta este manager la Administration, Marketing sau IT se va afișa numele complet al managerului. Dacă acesta este la Human Resources se va afișa salariul. Dacă este la Accounting se va afișa data angajării.

b) Să se determine cel mai bine plătit angajat născut după anul 2000 și departamentul al cărui manager are cel mai mic salariu. Să se afișeze dacă angajatul determinat aparține departamentului determinat.

c) Să se determine cele 2 orașe în care lucrează angajații cu media salariului cea mai mare respectiv cea mai mică. Să se afișeze cel care are dimensiunea mai mare (numărul de caractere din nume).

d) Să se rețină toate informațiile despre angajații care lucrează într-un oraș cu numele dat de la tastatură. Să se afișeze pe ecran doar cele pentru primii 10 angajați în ordine alfabetică.

e) Se dau 10 id-uri de angajați de la tastatură. Să se afle care este departamentul în care lucrează cei mai mulți dintre ei.

f) Pentru toate departamentele care nu au angajați să se rețină toate câmpurile din locația lor. Să se afișeze toate adresele reținute separate prin virgulă.