Kapitola 5

Kolekcie, Recordy, Objekty1

5.1 Zadanie cvičenia 5

5.1.1 Recordy

- Doplňte do zadaného zdrojového kódu deklaráciu typu `t_st_rec`, tak aby ste nemenili ostatné príkazy. (Typy zadajte podľa schémy príslušných stĺpcov).

```
create or replace procedure vypis_studentov( rocnik IN student.rocnik%TYPE)
as
  cursor cur1 ( p_rocnik  student.rocnik%TYPE) IS
    select st.os_cislo, ou.meno, ou.priezvisko, st.st_sk_new
    from priklad_db2.student st JOIN priklad_db2.os_udaje ou USING (rod_cislo)
      where st.rocnik = p_rocnik
      order by st.st_sk_new, ou.priezvisko, ou.meno;

  TYPE t_st_rec ...
  st_rec t_st_rec;
begin
  OPEN cur1(rocnik);
  LOOP
    FETCH cur1 INTO st_rec;
    IF cur1%NOTFOUND THEN
      EXIT;
    ELSE
      dbms_output.put(st_rec.oc||' ');
      dbms_output.put(st_rec.meno||' '||st_rec.priezv||' ');
      dbms_output.put_line(st_rec.skupina);
    END IF;
  END LOOP;
  CLOSE cur1;
end;
/
```

- Upravte predchádzajúci zdrojový kód tak, aby ste nepoužili exaktnú deklaráciu typu `t_st_rec`, ale typ record na základe kurzora.

```
st_rec cur1%ROWTYPE;
```

5.1.2 Kolekcie

Vytvorte nepomenovaný blok príkazov:

- vytvorte typ `t_pole` ako pole celých čísel `varray` s maximálnou dĺžkou 10
- vytvorte premennú `pole` a naplňte ho 5 číslami
- vypíšte počet prvkov poľa
- vypíšte obsah poľa
- vložte ďalšie číslo do poľa
- opäť vypíšte obsah poľa
- pokúste sa vymazať 3 prvok poľa

- to isté spravte pre typ `nested table`
- okrem toho vymažte 3 a 4 prvok
- a opäť vypíšte obsah poľa pomocou nasledovných cyklov:

a)
```
FOR i IN 1 .. pole.count LOOP
...
END LOOP;
```

b)
```
FOR i IN pole.first .. pole.last LOOP
...
END LOOP;
```

c)
```
WHILE pole.exists(i) LOOP
...
END LOOP;
```

5.1.3 Objekty

- vytvorte typ `t_adresa` s:
  - atribútmi `ulica, psc, mesto`,
  - procedúrou `vypis` - formátovaný výpis adresy
- vytvorte typ `t_osoba` s:
  - atribútmi `meno, priezvisko, rod_cislo , adresa` (typu `t_adresa`)
  - procedúrou `vypis_adresu` - výpis adresy aj s menom a priezviskom. Pričom využite procedúru `vypis` atribútu `adresa`.
- V nepomenovanom bloku
  - vytvorte objekt `OSOBA` typu `t_osoba`
  - osobu naplňte údajmi,
  - vypíšte adresu

- vytvorte tabuľku `osoby` objektov typu `t_osoba`. (Nie tabuľku so stĺpcom objektového typu)
- vložte aspoň 3 osoby do tabuľky (použite rôzne formy insertu)

- vypíšte obsah tabuľky – `select *`
- vypíšte obsah tabuľky – `select value(p)`
- pomocou procedúry `vypis_adresu` vypíšte adresy všetkých ľudí z tabuľky `osoby`
  1. pomocou kurzora,
  2. pomocou `bulk collect`

Materiál 5

Recordy

5.1 Premenná typu RECORD

Record je možné definovať viacerými spôsobmi:

1. ako vlastný typ a následne premennú daného typu
   - Deklarácia typu

     ```
     TYPE nazov_typu  IS  RECORD(
       nazov_polozky1 typ_polozky1,
       nazov_polozky1 typ_polozky2,
          ...
       nazov_polozky1 typ_polozkyN);
     ```

   - deklarácia premennej

     ```
     nazov_premennej nazov_typu;
     ```

   pričom typ položky môže byť priamo konkrétny dátový typ, alebo typ podľa stĺpca tabuľky. (`tabulka.stlpec%TYPE`)
2. record s rovnakou štruktúrou ako tabuľka

   ```
   nazov_premennej nazov_tabulky%ROWTYPE;
   ```

3. record s rovnakou štruktúrou ako kurzor

   ```
   nazov_premennej nazov_kurzora%ROWTYPE;
   ```

Príklad 5.1 *- Record definovaný podľa kurzora*

```
Procedure uvod AS
  Cursor cur1 IS
    Select meno, priezvisko from os_udaje;

  ou_rec cur1%ROWTYPE ;                -- deklaracia premennej podla kurzora

  BEGIN
    OPEN cur1;                           -- 1. otvorenie kurzora
    LOOP                                 --    nekonecny cyklus
      FETCH cur1 INTO ou_rec;             -- 2. nacitanie riadku do premennej
```

```
      IF cur1%FOUND THEN                    -- 3. ak bolo nieco nacitane
        HTP.P('<TR><TD>'||ou_rec.meno);
        HTP.P('</TD><TD>'|| ou_rec.priezvisko );
        HTP.P('</TD></TR>');
      ELSE
        EXIT;             -- 4. ukoncenie cyklu, ak uz ziaden riadok nebol nacitany
      END IF;
    END LOOP;
    CLOSE cur1;           -- 5. zatvorenie kurzora
END uvod;
/
```

Kolekcia je usporiadaná množina elementov rovnakého typu. Každý element má unikátny index, ktorý určuje jeho pozíciu v kolekcii. PL/SQL ponúka nasledovné typy kolekcií – Varray (pole) a Nested Table (hniezdená tabuľka) a Index by table (hash tabuľka).

Index-by table – jedno dimenzionálna kolekcia homogénnych elementov, ktoré sú prístupné len v PL/SQL, ale nie v databáze. Veľkosť a index tejto kolekcie je vždy zhodný so zvoleným dátovým typom indexu (a to buď PLS_INTEGER, alebo BINARY_INTEGER).

Nested table - jedno dimenzionálna **neohraničená** kolekcia homogénnych elementov, ktoré sú prístupné aj v PL/SQL aj v databáze.

VARRAY – jedno dimenzionálne **ohraničené** pole homogénnych elementov s variabilnou dĺžkou. Táto kolekcia je dostupná aj v PL/SQL aj v databáze.

5.2 Vytvorenie kolekcií

- ***Syntax:*** Index-by table

```
TYPE type_name IS TABLE OF element_type [NOT NULL]
  INDEX BY  BINARY_INTEGER | PLS_INTEGER ;
```

!! ***Upozornenie***
Typ indexu môže byť len BINARY_INTEGER, alebo PLS_INTEGER.
Nie INTEGER, alebo CHAR.

- ***Syntax:*** Nested table

```
[CREATE [OR REPLACE]] TYPE type_name IS TABLE
  OF  element_type [NOT NULL];
```

- ***Syntax:*** Varray

```
[CREATE [OR REPLACE]] TYPE type_name IS VARRAY |
    VARYING ARRAY (max_elements)
  OF element_type [NOT NULL];
```

Príklad 5.2 *- Deklarácie typov a inicializácia premenných v nepomenovanom bloku*

```
DECLARE
  -- varray
  type t_pole1 IS VARRAY (15) OF integer;
  pole1 t_pole1;

  -- nested table
  type t_pole2 IS TABLE  OF integer;
  pole2 t_pole2;

  -- index by table
  type t_pole3 IS TABLE OF integer INDEX BY PLS_INTEGER ;
  pole3 t_pole3;

BEGIN
  -- inicializacia premennych pomocou konstruktora
  pole1:= t_pole1(10,30,60,24,67,90,7,14);
  pole2:= t_pole2(10,30,60,24,67,90,7,14);

  -- index by table - len priamy pristup k polozkam
  pole3(1):= 10;
  pole3(2):= 30;
  pole3(3):= 60;
  pole3(4):= 24;
  pole3(5):= 67;
  pole3(6):= 90;
  pole3(7):= 7;
  pole3(8):= 14;

...

END;
/
```

Príklad 5.3 *- Vytvorenie typu a deklarácia premenných*

```
  -- vytvorenie typov
  CREATE TYPE t_pole1 IS VARRAY (15) OF integer;
  /
  CREATE TYPE t_pole2 IS TABLE  OF integer;
  /

  -- index by table nie je možné natrvalo vytvoriť v DB

DECLARE
  -- deklarácia premenných
  pole1 t_pole1;
  pole2 t_pole2;

BEGIN
  pole1:= t_pole1(10,30,60,24,67,90,7,14);
  pole2:= t_pole2(10,30,60,24,67,90,7,14);
...
END;
/
```

5.3 Metódy kolekcií

Premenné typu Varray, Nested table a Index by table svojím menom poskytujú metódy, ktoré nám umožňujú manipuláciu s jednotlivými prvkami kolekcií.

Metóda	Popis
`function EXISTS(i)`	existuje položka s indexom i?
`function COUNT`	skutočný počet platných položiek
`function LIMIT`	maximálny možný počet položiek
`function FIRST, LAST`	absolútny pohyb
`function PRIOR(i), NEXT(i)`	relatívny pohyb po platných prvkoch
`procedure EXTEND[(n)]`	rozšír kolekciu o `n` počet prvkov
`procedure TRIM[(n)]`	usekni `n` elementov z konca kolekcie
`procedure DELETE[(i[,j])]`	vymaž elementy od `i` po `j`, prípadne koniec

Kde:

- EXTEND, TRIM – nemôžu byť použité s Index-by tabuľkami.
- EXTEND - rozširuje varray a nested table, aby mohli byť pridané ďalšie prvky. Varray môže byť rozšírené maximálne po limit zadaný pri vytvorení typu.
- EXISTS
  - Vráti hodnotu `TRUE` iba ak má daný prvok platnú hodnotu. Inak vráti `FALSE`.
  - Len funkcia `EXISTS` môže byť aplikovaná aj na null kolekcie. Ak aplikujeme inú metódu na null kolekciu, PL/SQL vyvolá výnimku `COLLECTION_IS_NULL`.
- LIMIT – nemá zmysel pre Nested table ani Index by table.
- COUNT, LAST – vo VARRAY sú vždy rovnaké, v nested table rozdiel tvoria vymazané položky.
- FIRST, LAST, PRIOR(i), NEXT(i) - vracajú index prvého, posledného elementu, prípadne predchádzajúceho, či nasledujúceho elementu voči i-temu elementu. (Pričom vymazané (neplatné) položky budú preskočené.)
- TRIM[(n)] – `n` počet udáva počet prvkov (default 1), ktoré budú odseknuté z konca, ak je `n` väčšie ako COUNT, potom vyvolá výnimku `SUBSCRIPT_BEYOND_COUNT`.
- DELETE[(i[,j])] - vymaže elementy od elementu `i` po element `j`. V prípade VARRAY je možné vymazať len celé pole pomocou `DELETE` bez parametrov.

5.3.1 Volanie metód kolekcií

Základná syntax:

```
nazov_kolekcie.nazov_metody[(parametre)]
```

Príklad 5.4 *- Vymazanie*

```
DECLARE
-- nie je možné DELETE(3) ak je typu VARRAY
-- delete pri VARRAY vymaze cele pole
  type t_pole is TABLE OF integer;
  pole t_pole;
BEGIN
  pole:=t_pole(10,20,30,40,50,60,70,80,90);
  pole.DELETE(3);   -- vymaze 3 prvok
  pole.DELETE(7,7); -- vymaze 7. prvok
```

```
  pole.DELETE(3,6); -- od 3. po 6. prvok
  pole.DELETE(6,3); -- nevymaže nič
  pole.DELETE;      -- vymaze vsetky prvky
END;
/
```

Príklad 5.5 *- Použitie funkcií*

```
DECLARE
  type t_pole IS TABLE OF integer;
  pole t_pole;
BEGIN
  pole:=t_pole(9,8,5,6,7,2,1,3,4);

  pole.DELETE(3);

  dbms_output.put_line (pole.COUNT);          -- 8
  dbms_output.put_line (pole.LAST);           -- 9

  dbms_output.put_line (pole.NEXT(2));        -- 4
  dbms_output.put_line (pole(pole.NEXT(2)));  -- 6

  if (pole.exists(3)) then
    dbms_output.put_line('existuje');
  else
    dbms_output.put_line('neexistuje');
  end if;                                     -- neexistuje

  pole.EXTEND;
  if (pole.exists(pole.last)) then
    dbms_output.put_line('existuje '|| pole.last);  -- existuje 10
    if (pole(pole.last) is null) then
      dbms_output.put_line('hodnota is null ');  -- hodnota is null
    else
      dbms_output.put_line('hodnota = '|| pole(pole.last));
    end if;
  else
    dbms_output.put_line('neexistuje');
  end if;

  pole(pole.last):= 1000;

  dbms_output.put_line (pole(pole.count));    -- 4
  dbms_output.put_line (pole(pole.last));     -- 1000

END;
/
```

Príklad 5.6 *- Funkcia Exists*

```
DECLARE
  -- varray
  type t_pole1 IS VARRAY (15) OF integer;
  pole1 t_pole1;
```

```
  -- nested table
  type t_pole2 IS TABLE  OF integer;
  pole2 t_pole2;

  -- index by table
  type t_pole3 IS TABLE OF integer INDEX BY PLS_INTEGER ;
  pole3 t_pole3;

BEGIN
  pole1:= t_pole1(10,30,60,24,67,90,7,14);
  pole2:= t_pole2(10,30,60,24,67,90,7,14);

  pole3(1):= 10;
  pole3(2):= 30;
  pole3(3):= 60;
  pole3(4):= 24;
  pole3(5):= 67;
  pole3(6):= 90;
  pole3(7):= 7;
  pole3(8):= 14;

  dbms_output.put_line('-- VARRAY ---');
  if (pole1.exists(16)) then dbms_output.put_line('18 - existuje');
    else dbms_output.put_line('16 - neexistuje');
  end if;

  if (pole1.exists(9)) then dbms_output.put_line('9 - existuje');
    else dbms_output.put_line('9 - neexistuje');
  end if;

  if (pole1.exists(3)) then dbms_output.put_line('3 - existuje');
    else dbms_output.put_line('3 - neexistuje');
  end if;

  dbms_output.put_line('-- NESTED TABLE ---');
  if (pole2.exists(9)) then dbms_output.put_line('9 - existuje');
    else dbms_output.put_line('9 - neexistuje');
  end if;

  pole2.delete(4,4);
  if (pole2.exists(4)) then dbms_output.put_line('4 - existuje');
    else dbms_output.put_line('4 - neexistuje');
  end if;

  if (pole2.exists(3)) then dbms_output.put_line('3 - existuje');
    else dbms_output.put_line('3 - neexistuje');
  end if;

  dbms_output.put_line('-- INDEX BY TABLE ---');
  if (pole3.exists(9)) then dbms_output.put_line('9 - existuje');
    else dbms_output.put_line('9 - neexistuje');
  end if;

  pole3.delete(4,4);
  if (pole3.exists(4)) then dbms_output.put_line('4 - existuje');
    else dbms_output.put_line('4 - neexistuje');
```

```
  end if;

  if (pole3.exists(3)) then dbms_output.put_line('3 - existuje');
    else dbms_output.put_line('3 - neexistuje');
  end if;
END;
/
```

Výstup:

```
-- VARRAY ---
16 - neexistuje
9 - neexistuje
3 - existuje
-- NESTED TABLE ---
9 - neexistuje
4 - neexistuje
3 - existuje
-- INDEX BY TABLE ---
9 - neexistuje
4 - neexistuje
3 - existuje
```

5.4 Tabuľky a kolekcie

Ako už bolo spomenuté kolekcie typu Varray a Nested table môžu byť použité aj pre dátové typy stĺpcov v relačných tabuľkách. V prípade Varray sa dáta ukladajú priamo v tabuľkovom priestore, avšak v prípade Nested table sa ukladajú mimo tabuľkového priestoru a meno tohto úložného priestoru je nutné zadať pri vytváraní tabuľky.

Syntax pre NESTED TABLE:

```
CREATE TABLE nazov_tabulky
(
  ...
)
NESTED TABLE nazov_stlpca STORE AS nazov_externej_tabulky
```

Príklad 5.7 *- Varray v tabuľke*

1. Najprv musíme vytvoriť typ, ktorý bude natrvalo uložený v databáze.

   ```
   CREATE OR REPLACE TYPE t_pole1 IS VARRAY(100) OF integer;
   /
   ```

2. Vytvorený typ môžeme použiť pre stĺpec tabuľky.

   ```
   create table tabx (
     stlpce t_pole1
   );
   ```

3. Na vloženie hodnôt do stĺpca je možné použiť konštruktor typu

```
insert into tabx values ( t_pole1(12,62,31,645,213));
```

alebo premennú daného typu.

```
declare
  pole t_pole1 := t_pole1(32,4,11,2);
begin
  insert into tabx values (pole);
end;
/
```

4. Potom obsah tabuľky bude nasledovný

```
SQL> select * from tabx;

STLPCE
--------------------------------------------------------------------------------
T_POLE1(12, 62, 31, 645, 213)
T_POLE1(32, 4, 11, 2)
```

Príklad 5.8 *- Nested table v tabuľke*

1. Najprv musíme vytvoriť typ, ktorý bude natrvalo uložený v databáze.

```
CREATE OR REPLACE TYPE t_pole2 IS TABLE OF integer;
/
```

2. Vytvorený typ môžeme použiť pre stĺpec tabuľky, ale keďže sa jedná o Nested table musíme špecifikovať kam ho má uložiť.

```
CREATE TABLE tabx (
    stlpec t_pole2
)
nested table stlpec store as tab_stlpec;
```

3. Na vloženie hodnôt do stĺpca je možné použiť konštruktor typu

```
insert into tabx values ( t_pole2(12, 34, 664, 12, 435));
```

alebo premennú daného typu.

```
declare
  pole t_pole1 := t_pole2(32,4,11,2);
begin
  insert into tabx values (pole);
end;
/
```

4.
```
SQL> select * from tabx;

STLPEC
--------------------------------------------------------------------------------
T_POLE2(12, 34, 664, 12, 435)
```

5. Spolu s tabuľkou vznikla pomocná štruktúra

```
SQL> desc tab_stlpec
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 COLUMN_VALUE                                       NUMBER(38)
```

z ktorej však nie je možné priamo selectovať.

```
SQL> select * from tab_stlpec;
select * from tab_stlpec
              *
ERROR at line 1:
ORA-22812: cannot reference nested table column's storage table
```