Записи и коллекции в PL/SQL

Записи и коллекции являются *составными структурами данных*. Записи очень похожи на строки в таблицах баз данных (БД). Коллекция – это структура данных, похожая на одномерный массив. В PL/SQL используются три типа коллекций: ассоциативные массивы, вложенные таблицы и массивы VARRAY.

1. Записи в PL/SQL

Организация данных в виде записей позволяет выполнять различные операции не только над их отдельными элементами, но и над записью как единым целым.

1.1. Объявление записей

- **на основе курсора**
- **на основе таблицы базы данных**
- **вручную (явно), используя оператор TYPE . . . RECORD**

1.1.1. Объявление записей на основе курсора

```
DECLARE
   CURSOR c_stan IS
       SELECT stname, nod FROM stations WHERE nod=2;
   r_stan   c_stan%ROWTYPE;
```

```
DECLARE
   TYPE t_c_stan IS REF CURSOR RETURN stations%ROWTYPE;
   c_stan t_c_stan;
   r_stan   c_stan%ROWTYPE;
```

```
BEGIN
   FOR  r_stan IN (SELECT * FROM stations WHERE nod=2)  LOOP
            DBMS_OUTPUT.LINE_PUT('Station is '|| r_stan.stname);
   END LOOP;
END;
```

1.1.2. Объявление записей на основе таблицы

```
DECLARE
   r_sal     study.sal%ROWTYPE;
```

1.1.3. Записи, определяемые программистом

```
TYPE имя_типа     IS RECORD (
   имя_поля1        тип_данных1,
   имя_поля2        тип_данных2,
   ...
   имя_поляN        тип_данныхN ) ;
```

- **Скалярный тип данных (VARCHAR2, NUMBER и т.д.)**
- **Подтип, определяемый программистом**
- **Тип, на основе уже определенной структуры данных (%TYPE, %ROWTYPE)**
- **Тип коллекции PL/SQL**
- **Тип REF CURSOR**

имя_записи тип_записи;

```
DECLARE
  TYPE r_sal_totl  IS RECORD (
        sal_att      study.sal%ROWTYPE,
        curr_date    DATE      :=  sysdate,
        ords_cnt     NUMBER(4),
        ords_sum     NUMBER(9,2) )  DEFAULT  null ) ;
rSal      r_sal_totl;
```

1.2. Обработка записей

Независимо от метода определения записи приемы работы с ней всегда одинаковы. Обрабатываться может либо вся запись целиком, либо каждое отдельное поле.

1.2.1. Операции над записями

- **копирование содержимого одной записи в другую (записи должны быть однотипными)**
- **присваивание записи значения NULL**
- **передача записи процедуре или функции в качестве аргумента**
- **возврат записи из функции (тип записи определяется явно)**

```
DECLARE
   rSal_tab   study.sal%ROWTYPE;
   CURSOR c_sal IS  SELECT * FROM study.sal;
   rSal_cur   c_sal %ROWTYPE;
BEGIN
   …
   rSal_tab  := rSal_cur;
END;
```

```
PROCEDURE output_ord(rOrd_in IN ord%ROWTYPE) IS
   rOrd   ord %ROWTYPE  :=  rOrd_in;
BEGIN
   DBMS_OUTPUT.PUT_LINE('Date order is '|| rOrd.odate);
END;
```

- **Нельзя использовать синтаксис IS NULL для проверки, все ли поля записи имеют значение NULL. Это можно сделать лишь проверкой каждого поля**
- **Невозможно сравнить две записи единственной операцией**
- **Только в Oracle9i Release 2 появилась возможность добавлять новые записи в таблицу БД целиком**

1.2.2. Операции над отдельными полями записи

[*имя_схемы*.][*имя_пакета*.]*имя_записи.имя_поля*

```
DECLARE
   newCm NUMBER;
BEGIN
   FOR v_sal IN (select * from sal) LOOP
      newCm:= v_sa.comm * 1.15;
      DBMS_OUTPUT.PUT_LINE(v_sa.sname||'   '|| newCm);
   END LOOP;
END;
```

1.3. Сравнение записей

Сравнить две записи как обычные скалярные величины невозможно.

```
IF (rStSal.city= rMySal.city OR
   (rStSal.city IS NULL  AND rMySal.city IS NULL)  )
  AND (rStSal.comm= rMySal.comm. OR
   (rStSal.comm. IS NULL  AND rMySal.comm IS NULL)  ) . . . THEN
```

2. Коллекции в PL/SQL

Коллекция – это структура данных, имеющая составной тип и предназначенная для хранения одномерных массивов. Коллекции используются для хранения множества однотипных элементов в коде PL/SQL или в столбцах таблиц базы данных.

- **Эмуляция двунаправленных курсоров и курсоров с произвольным доступом**
- **Хранение списков подчиненной информации непосредственно в столбцах таблицы – в виде вложенной таблицы или массива VARRAY**
- **Кэширование статичной информации базы данных, которая часто указывается в запросах**

2.1. Типы коллекций и объявление

Oracle9i поддерживает три типа коллекций.

- ***Ассоциативные массивы*. Это одномерные неограниченные разреженные коллекции, которые используются только в PL/SQL.**
- ***Вложенные таблицы*. Коллекции, которые первоначально заполняются полностью, но вследствие последующего удаления элементов могут стать разреженными. Используются и в PL/SQL, и в базах данных.**
- ***Массивы VARRAY*. Коллекции, размер которых всегда ограничены, и они не могут быть разреженными. Используются и в PL/SQL, и в базах данных.**

Коллекции называются *ограниченными*, если заранее определены границы возможных значений индексов ее элементов. Для VARRAY индексы изменяются от 1 до 2^{31}-1.

Для *неограниченных* коллекций верхняя или нижняя границы номеров элементов не указываются. Для ассоциативных массивов индексы изменяются от -2^{31}+1 до 2^{31}-1. Для вложенных таблиц индексы изменяются от 1 до 2^{31}-1.

***Плотными* коллекциями называются такие, все элементы которых определены, и каждому элементу присвоено некоторое значение (или NULL).**

Коллекция *разреженная*, если отдельные ее элементы отсутствуют.

Коллекции всех типов *индексируются целочисленными значениями*.

В Oracle9i Release 2 появилась дополнительная возможность индексации ассоциативных массивов – *индексация строками*.

Как задать тип коллекции.

- **с использование команды CREATE TYPE. Так определяется тип вложенной таблицы и тип массива VARRAY в базе данных. Это позволяет использовать объявленный тип при определении столбцов таблицы БД, переменных в программах PL/SQL, а также атрибутов объектных типов**
- **с применением синтаксиса TYPE . . . IS. Этот способ используется при определении типа коллекции в программе PL/SQL**

2.1.1. Объявление ассоциативных массивов

TYPE *имя_типа_таблицы* IS TABLE OF *тип_данных* [NOT NULL]
INDEX BY {BINARY_INTEGER | *подтип_тип* BINARY_INTEGER | VARCHAR2(*размер*)};

- **скалярный тип данных – любой поддерживаемый PL/SQL скалярный тип**
- **тип данных с привязкой – определяется ранее объявленной переменной, таблицей или курсором**

```
DECLARE
  TYPE t_ord IS TABLE OF ord %ROWTYPE
    INDEX BY VARCHAR2(20);
  TYPE t_sal_nm IS TABLE OF sal.sname%TYPE  NOT NULL
    INDEX BY BINARY_INTEGER;
  ...
BEGIN  . . .
```

До Oracle9i Release 2

INDEX BY BINARY_INTEGER

Теперь появились дополнительные возможности:

INDEX BY PLS_INTEGER
INDEX BY NATURAL
INDEX BY POSITIVE
INDEX BY VARCHAR2(*размер*)
INDEX BY *таблица.столбец*%TYPE
INDEX BY *пакет.переменная*%TYPE
INDEX BY *подтип*

***имя_коллекции тип_коллекции*;**

```
CREATE OR REPLACE PACKAGE sal_pkg  IS
   TYPE t_sal_nm IS TABLE OF sal.sname%TYPE  NOT NULL
     INDEX BY BINARY_INTEGER;
   t_salNm  t_sal_nm;
END sal_pkg;
```

2.1.2. Объявление вложенной таблицы или массива VARRAY

Синтаксис задания в базе данных:

```
CREATE  [OR REPLACE]  TYPE имя_типа  AS | IS
TABLE OF тип_данных_элемента  [NOT NULL];

CREATE  [OR REPLACE]  TYPE имя_типа  AS | IS
VARRAY  (максимальный_индекс)  OF  тип_данных_элемента
[NOT NULL];
```

Для задания типов в программе PL/SQL

```
TYPE имя_типа  IS TABLE OF тип_данных_элемента  [NOT NULL];
TYPE имя_типа  IS  VARRAY (максимальный_индекс)
        OF тип_данных_элемента  [NOT NULL];


CREATE  TYPE Sal_city_t  AS TABLE OF VARCHAR2(30);
DECLARE
    TYPE Number_t  IS ARRAY(5) OF NUMBER;
    my_city          Sal_city_t ;
    my_number      Number_t  :=  Number_t(7,19);
    . . .
END;
```

2.2. Использование коллекций

2.2.1. Коллекция как компонент записи

```
DECLARE
    TYPE Number_r  IS RECORD (
    Id_rec             NUMBER(4),
    my_number      Number_t );
    . . .
END;
```

2.2.2. Коллекция как параметр программы

```
CREATE PROCEDURE show_color1(p_Color IN Color_t) IS
    tCol   Color_t := p_Color;
BEGIN
    IF (tCol.count>0  AND  tCol.exists(1)) THEN
        DBMS_OUTPUT.PUT_LINE('Color(1) is '|| tCol(1));
    END IF;
END;
```

2.2.3. Коллекция как тип данных, возвращаемый функцией

```
CREATE or replace FUNCTION get_ch  RETURN  Ch_Name_t  IS
 tChName   Ch_Name_t;
BEGIN
 SELECT Ch_Name  INTO tChName  FROM employees where id=1202;
 RETURN  tChName;
EXCEPTION WHEN others THEN
 RETURN  NULL;
END;
```

```
DECLARE
  my_tChName   Ch_Name_t;
BEGIN
  my_tChName   :=  get_ch;
END;
```

2.2.4. Коллекция как столбец таблицы базы данных

Id NUMBER	Name VARCHAR2(20)	Ch_Name VARCHAR2(20)
1000	Molly Smith	Kate
1007	Teddy Mlime	
1202	Nick Borrow	Tom
		Bill

```
CREATE  TYPE Ch_Name_t  AS VARRAY(10) OF VARCHAR2(20);

CREATE  TABLE employees  (
    Id           NUMBER,
    Name       VARCHAR2(20),
    Ch_Name  Ch_Name_t
);
```

Используем Ch_Name_t как конструктор

```
INSERT INTO employees  (Id,Name, Ch_Name)
     VALUES(1000, 'Molly Smith', Ch_Name_t('Kate'));
INSERT INTO employees  (Id,Name, Ch_Name)
     VALUES(1007, 'Teddy  Mlime', NULL);
INSERT INTO employees  (Id,Name, Ch_Name)
     VALUES(1202, 'Nick  Borrow', Ch_Name_t('Tom','Bill'));
```

2.3. Встроенные методы коллекций

Метод	Описание
Функция COUNT	Возвращает текущее количество элементов в коллекции
Процедура DELETE	Удаляет из один или несколько элементов коллекции
Функция EXISTS	Возвращает значение TRUE или FALSE, определяющее, существует ли в коллекции заданный элемент
Процедура EXTEND	Увеличивает количество элементов в коллекции
Функции FIRST, LAST	Возвращают индексы первого (FIRST) и последнего (LAST) элементов коллекции
Функция LIMIT	Возвращает максимальное количество элементов в массиве VARRAY
Функции PRIOR, NEXT	Возвращают индексы элементов, предшествующих заданному (PRIOR) и следующему за ним (NEXT)
Функция TRIM	Удаляет элементы, начиная с конца коллекции

имя_коллекции.метод

***имя_коллекции.метод*(*индекс* [, *индекс*])**

2.3.1. Метод COUNT

Коллекция.COUNT

```
DECLARE
    my_List   List_t;
BEGIN
    DBMS_OUTPUT.PUT_LINE('my_List count is '|| my_List.COUNT);
END;
```

2.3.2. Метод DELETE

Коллекция.DELETE [(n1 [,n2])]

```
CREATE PROCEDURE my_del (p_List  IN OUT List_t) IS
    l_ot    NUMBER := p_List.FIRST;
    l_do   NUMBER := p_List.NEXT(l_ot);
BEGIN
    IF (p_List.COUNT>0) then
        IF (l_do IS NOT NULL) THEN
            p_List.DELETE(l_ot, l_do);
        ELSE
            p_List.DELETE;
        END IF;
    END IF;
END;
```

2.3.3. Метод EXISTS

Коллекция.EXISTS (n)

```
CREATE PROCEDURE my_empty (p_List  IN OUT List_t) IS
BEGIN
    IF (p_List.EXISTS(2)) then
            p_List(2)  :=  NULL;
    END IF;
END;
```

2.3.4. Метод EXTEND

Коллекция.EXTEND [(n1 [,n2])]

```
CREATE PROCEDURE my_value (p_List  IN OUT List_t) IS
BEGIN
    p_List. EXTEND;
    p_List(p_List.LAST)  :=  '-';
END;
```

2.3.5. Методы FIRST и LAST

Коллекция.FIRST и Коллекция.LAST

```
CREATE PROCEDURE my_ send (p_List  IN OUT List_t) IS
BEGIN
    FOR idx  IN  p_List. FIRST .. p_List. LAST   LOOP
        send_email(idx);
    END LOOP;
END;
```

2.3.6. Метод LIMIT

Коллекция.LIMIT

```
IF my_List.LAST < my_List. LIMIT  THEN
        my_List.EXTEND;
END IF;
```

2.3.7. Методы PRIOR и NEXT

Коллекция.PRIOR (n) и Коллекция.NEXT (n)

```
CREATE FUNCTION cnt_Totl (p_List  IN List_t) RETURN NUMBER IS
    idx   NUMBER :=  p_List.FIRST;
    Ttl   NUMBER :=  0;
BEGIN
    LOOP
        EXIT WHEN idx IS NULL;
        Ttl  :=  Ttl  +  p_List(idx);
        Idx  :=  p_List.NEXT (idx);
    END LOOP;
    RETURN  Ttl;
END;
```

2.3.8. Метод TRIM

Коллекция.TRIM [(n)]

```
CREATE FUNCTION val_last (p_List  IN List_t) RETURN NUMBER IS
    l_val   NUMBER;
BEGIN
    IF p_List.COUNT>0 THEN
        l_val  :=  p_List(p_List.LAST);
        p_List.TRIM
    END IF;
    RETURN  l_val;
END;
```

2.4. Работа с коллекциями

2.4.1. Инициализация коллекций

Ассоциативные массивы не требуют явной инициализации.
Инициализация вложенных таблиц и массивов VARRAY:

- **явно, с помощью конструктора**
- **неявно, путем непосредственного присваивания содержимого другой переменной-коллекции**
- **неявно, путем выборки записей из базы данных**

```
DECLARE
   my_List   List_t  :=  List_t('RED','GREEN');
BEGIN
   . . .
END;
```

```
DECLARE
   my_List   List_t  :=  List_t('RED','GREEN');
   other_List   List_t;
BEGIN
   other_List  :=  my_List;
   other_List(2)  :=  'BLUE';
END;
```

```
DECLARE
   my_List   List_t;
BEGIN
   SELECT color INTO my_List FROM color_models
   WHERE model_type='RGB'
   . . .
END;
```

2.4.2. Присваивание значений элементам коллекции

```
my_List(4)  :=  'RED';

other_List  :=  my_List;
```

2.4.3. Считывание значений элементов коллекции

```
IF (p_List.EXISTS(2)) then
      DBMS_OUTPUT.PUT_LINE('p_List(2)='|| p_List(2));
END IF;
```

2.4.4. Работа с коллекциями составных элементов

- **коллекции записей (Oracle 7.3.4 и выше) – при условии, что записи не содержат в качестве полей другие записи или коллекции**
- **коллекции объектов (Oracle8 и выше)**
- **коллекции коллекций (Oracle9i и выше)**

2.4.5. Использование ассоциативных массивов типа VARCHAR2

```
DECLARE
    TYPE name_t IS TABLE emp%ROWTYPE INDEX BY VARCHAR2(20);
    TYPE id_t IS TABLE emp%ROWTYPE INDEX BY BINARY_INTEGER;
    by_name  name_t;
    by_tabN    name_t;
    by_id        id_t;
BEGIN
    FOR v_emp IN (SELECT * FROM emp) LOOP
        by_name(v_emp.name) := v_emp;
        by_tabN (v_emp.tabN)   := v_emp;
        by_id (v_emp.id)           := v_emp;
    END LOOP;
    IF by_id.EXISTS(27) THEN
       -- обработка записи
    END IF;
    IF by_name.EXISTS('Петров П.П.') THEN
       -- обработка записи
    END IF;
    IF by_tabN.EXISTS('052') THEN
       -- обработка записи
    END IF;
END;
```

2.4.6. Работа с массивом как с таблицей

```
 create or replace type t_z_sal is object (
    snum   NUMBER(4,0),
    sname  VARCHAR2(10 BYTE),
    city   VARCHAR2(10 BYTE),
    comm   NUMBER(7,2));
 create type t_tab_sal is table of dispark.t_z_sal;

declare
  cursor c_tst is
    select * from sal order by snum;

  cursor c_arr(p_City varchar2) is
    select snum,sname,comm from
      table ( tTab )
--      table ( cast (tTab as t_tab_sal) )
      where city=p_City
      order by snum desc;
  type t_t_arr is table of c_arr%rowtype index by binary_integer;

  tTab t_tab_sal := t_tab_sal();
  v_tst t_z_sal := t_z_sal(null,null,null,null);
  tArr t_t_arr;

begin
  open c_tst;
  loop
    fetch c_tst into v_tst.snum,v_tst.sname,v_tst.city,v_tst.comm;
      exit when(c_tst%notfound);
    tTab.extend;
    tTab(tTab.last) := v_tst;
  end loop;
  close c_tst;
  if (tTab.count>0) then
    for v_tst in c_arr('Barcelona') loop --с курсором
      tArr(c_arr%rowcount) := v_tst;
      dbms_output.put_line(' sname='||v_tst.sname);
    end loop;
  end if;
  select * bulk collect into tArr from    --без курсора
    table ( tTab )
    where city='Barcelona'
    order by snum desc;
end;
```