Коллекции и курсоры

Содержание

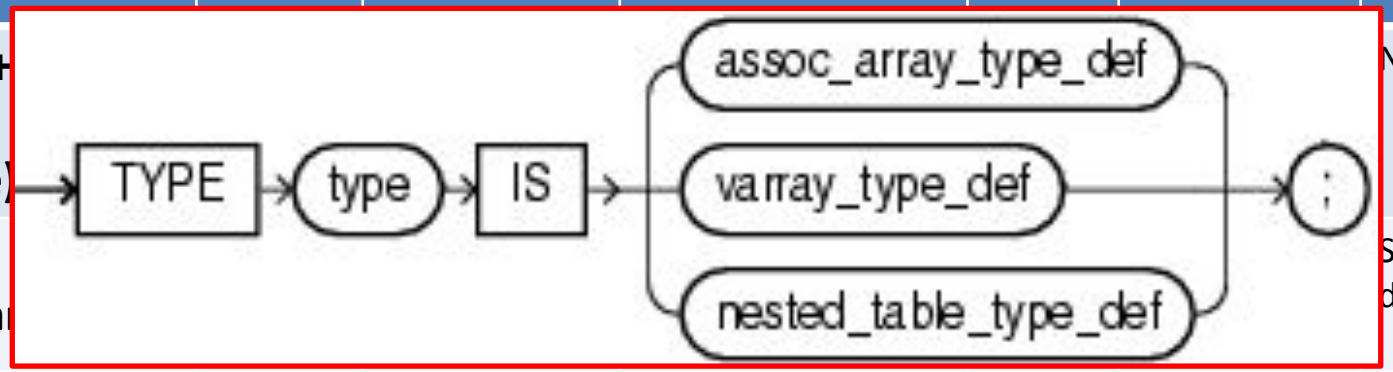
- ✓ Типы коллекций
- ✓ Ассоциативный массив (index by table)
- ✓ Varray
- ✓ Nested table
- ✓ Set Operations
- ✓ Логические операторы
- ✓ Методы коллекций
- ✓ Bulk Collect
- ✓ Forall

Коллекции в Oracle

- Создание коллекции
  1. Определить тип(type) коллекции
  2. Создать переменную этого типа
- Обращение к элементу коллекции: variable_name(index)
- Могут принимать значение NULL
- Возможны многомерные коллекции (коллекции коллекций)

Типы коллекций

Тип коллекции	Кол-во элементов	Тип индекса	Плотная или разреженная	Без инициализации	Где объявляется	Использование в SQL
Ассоциативный массив (index by table)						No
Varray (variable-size a						Schema level defined only
Nested table	Не задано	INTEGER	Starts dense Can become sparse	Null	PL/SQL block Package Schema level	Schema level defined only



Ассоциативный массив (index by table)

- ✓ На
- ✓ Да
- ✓ Не
- ✓ Пр
  фу
- ✓ По
  за
- ✓ Не
- ✓ Не

Испо

- ✓ Дл
- ✓ Дл

ован

ексом

```
CREATE OR REPLACE PACKAGE My_Types AUTHID DEFINER
IS
  TYPE My_AA IS TABLE OF VARCHAR2(20) INDEX BY
PLS_INTEGER;
  FUNCTION Init_My_AA RETURN My_AA;
END My_Types;
/
CREATE OR REPLACE PACKAGE BODY My_Types IS
  FUNCTION Init_My_AA RETURN My_AA IS
    Ret My_AA;
  BEGIN
    Ret(-10) := '-ten';
    Ret(0) := 'zero';
    Ret(1) := 'one';
    Ret(2) := 'two';
    Ret(3) := 'three';
    Ret(4) := 'four';
    Ret(9) := 'nine';
    RETURN Ret;
  END Init_My_AA;
END My_Types;
```

Varray

✓ Размер задается при создании

```
DECLARE
  TYPE Foursome IS VARRAY(4) OF VARCHAR2(15);   -

  -- varray variable initialized with constructo

  team Foursome := Foursome('John', 'Mary', 'Alb
'Juanita');

BEGIN
  team(3) := 'Pierre';  -- Change values of two
  team(4) := 'Yvonne';

  -- Invoke constructor to assign new values to
  team := Foursome('Arun', 'Amitha', 'Allan', 'M
END;
```

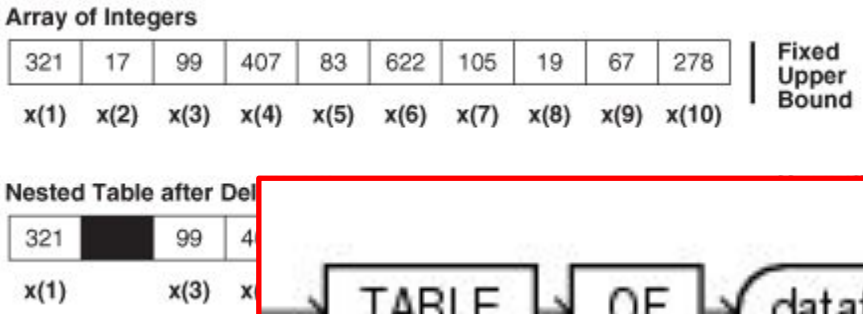
```
2001 Team:
1.John
2.Mary
3.Alberto
4.Juanita
---
2005 Team:
1.John
2.Mary
3.Pierre
4.Yvonne
---
2009 Team:
1.Arun
2.Amitha
3.Allan
4.Mae
---
```

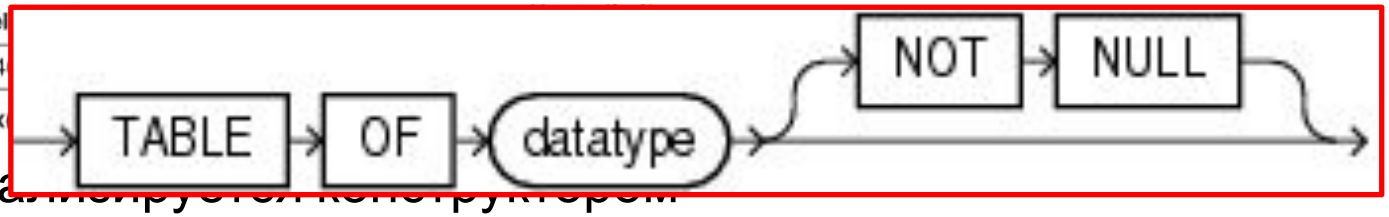
2. Доступ к элементам последовательный

Nested table

✓ Размер коллекции изменяется динамически

✓ Может быть в разряженном состоянии, как показано на картинке





✓ Инициализируется конструктором

collection_type ([*value* [, *value*]...])

✓ Если содержит только одно скалярное значение, то имя колонки – Column_Value

```
SELECT column_value
FROM   TABLE(nested_table)
```

Set Operations with Nested Tables

```
DECLARE
    TYPE nested_typ IS TABLE OF NUMBER;
    nt1 nested_typ := nested_typ(1,2,3);
    nt2 nested_typ := nested_typ(3,2,1);
    nt3 nested_typ := nested_typ(2,3,1,3);
    nt4 nested_typ := nested_typ(1,2,4);
    answer nested_typ;
BEGIN
    a
    a
    a
    a
    a                                                    nt3;
    a
    a
    answer := nt3 MULTISET EXCEPT DISTINCT nt2;
END;
```

```
nt1 MULTISET UNION nt4: 1 2 3 1 2 4
nt1 MULTISET UNION nt3: 1 2 3 2 3 1 3
nt1 MULTISET UNION DISTINCT nt3: 1 2 3
nt2 MULTISET INTERSECT nt3: 3 2 1
nt2 MULTISET INTERSECT DISTINCT nt3: 3 2 1
SET(nt3): 2 3 1
nt3 MULTISET EXCEPT nt2: 3
nt3 MULTISET EXCEPT DISTINCT nt2: empty set
```

```
DECLARE
    TYPE nested_typ IS TABLE OF NUMBER;
    nt1 nested_typ := nested_typ(1, 2, 3);
    nt2 nested_typ := nested_typ(3, 2, 1);
    nt3 nested_typ := nested_typ(2, 3, 1, 3);
    nt4 nested_typ := nested_typ();
BEGIN
    IF nt1 = nt2 THEN
        DBMS_OUTPUT.PUT_LINE('nt1 = nt2');
    END IF;

    IF (nt1 IN (nt2, nt3, nt4)) THEN
        DBMS_OUTPUT.PUT_LINE('nt1 IN (nt2,nt3,nt4)');
    END IF;

    IF (nt1 SUBMULTISET OF nt3) THEN
        DBMS_OUTPUT.PUT_LINE('nt1 SUBMULTISET OF nt3');
    END IF;

    IF (3 MEMBER OF nt3) THEN
        DBMS_OUTPUT.PUT_LINE('3 MEMBER OF nt3');
    END IF;

    IF (nt3 IS NOT A SET) THEN
        DBMS_OUTPUT.PUT_LINE('nt3 IS NOT A SET');
    END IF;

    IF (nt4 IS EMPTY) THEN
        DBMS_OUTPUT.PUT_LINE('nt4 IS EMPTY');
    END IF;
END;
```

nt1 = nt2
nt1 IN (nt2,nt3,nt4)
nt1 SUBMULTISET OF nt3
3 MEMBER OF nt3
nt3 IS NOT A SET
nt4 IS EMPTY

Методы коллекций

Метод	Тип	Описание
DELETE	Procedure	Удаляет элементы из коллекции (не работает с varray)
TRIM	Procedure	Удаляет элементы с конца varray или nested table
EXTEND	Procedure	Добавляет элементы в конец varray или nested table.
EXISTS	Function	Возвращает TRUE, если элемент присутствует в varray или nested table
FIRST	Function	Возвращает первый индекс коллекции
LAST	Function	Возвращает последний индекс коллекции
COUNT	Function	Возвращает количество элементов в коллекции
LIMIT	Function	Возвращает максимальное количество элементов, которые может хранить коллекция
PRIOR	Function	Возвращает индекс предыдущего элемента коллекции
NEXT	Function	Возвращает индекс следующего элемента коллекции

Синтаксис вызова методов:
collection_name.method

Delete

```
DECLARE
    TYPE nt_type IS TABLE OF NUMBER;
    nt nt_type := nt_type(11, 22, 33, 44, 55, 66);
BEGIN
    nt.DELETE(2); -- Удаляет второй элемент
    nt(2) := 2222; -- Восстанавливает 2-й элемент
    nt.DELETE(2, 4); -- Удаляет элементы со 2-го по 4-й
    nt(3) := 3333; -- Восстанавливает 3-й элемент
    nt.DELETE; -- Удаляет все элементы
END;
```

```
beginning: 11 22 33 44 55 66
after delete(2): 11 33 44 55 66
after nt(2) := 2222: 11 2222 33 44 55 66
after delete(2, 4): 11 55 66
after nt(3) := 3333: 11 3333 55 66
after delete: : empty set
```

Trim

```
DECLARE
    TYPE nt_type IS TABLE OF NUMBER;
    nt nt_type := nt_type(11, 22, 33, 44, 55, 66);
BEGIN
    nt.TRIM; -- Trim last element
    nt.DELETE(4); -- Delete fourth element
    nt.TRIM(2); -- Trim last two elements
END;
```

```
beginning: 11 22 33 44 55 66
after TRIM: 11 22 33 44 55
after DELETE(4): 11 22 33 55
after TRIM(2): 11 22 33
```

Extend

```
DECLARE
    TYPE nt_type IS TABLE OF NUMBER;
    nt nt_type := nt_type(11, 22, 33);
BEGIN
    nt.EXTEND(2, 1); -- Append two copies of first element
    nt.DELETE(5); -- Delete fifth element
    nt.EXTEND; -- Append one null element
END;
```

```
beginning: 11 22 33
after EXTEND(2,1): 11 22 33 11 11
after DELETE(5): 11 22 33 11
after EXTEND: 11 22 33 11
```

Exists

```
DECLARE
    TYPE NumList IS TABLE OF INTEGER;
    n NumList := NumList(1, 3, 5, 7);
BEGIN
    n.DELETE(2); -- Delete second element
    FOR i IN 1 .. 6
    LOOP
        IF n.EXISTS(i)
        THEN
            DBMS_OUTPUT.PUT_LINE('n(`||i||') = ' || n(i));
        ELSE
            DBMS_OUTPUT.PUT_LINE('n(`||i||') does not
  exist');
        END IF;
    END LOOP;
END;
```

First и Last

```
DECLARE
    TYPE aa_type_str IS TABLE OF INTEGER INDEX BY VARCHAR2(10);
    aa_str aa_type_str;
BEGIN
    aa_str('Z') := 26;
    aa_str('A') := 1;
    aa_str('K') := 11;
    aa_str('R') := 18;

    DBMS_OUTPUT.PUT_LINE('Before deletions:');
    DBMS_OUTPUT.PUT_LINE('FIRST = ' || aa_str.FIRST);
    DBMS_OUTPUT.PUT_LINE('LAST = ' || aa_str.LAST);

    aa_str.DELETE('A');
    aa_str.DELETE('Z');

    DBMS_OUTPUT.PUT_LINE('After deletions:');
    DBMS_OUTPUT.PUT_LINE('FIRST = ' || aa_str.FIRST);
    DBMS_OUTPUT.PUT_LINE('LAST = ' || aa_str.LAST);
END;
```

```
Before deletions:
FIRST = A
LAST = Z
After deletions:
FIRST = K
LAST = R
```

Count

```
DECLARE
    TYPE NumList IS VARRAY(10) OF INTEGER;
    n NumList := NumList(1, 3, 5, 7);

BEGIN
    DBMS_OUTPUT.PUT('n.COUNT = ' || n.COUNT || ', ');
    DBMS_OUTPUT.PUT_LINE('n.LAST = ' || n.LAST);

    n.EXTEND(2);
    DBMS_OUT                                  COUNT || ', ');
    DBMS_OUT                                  | n.LAST);

    n.TRIM(5);
    DBMS_OUTPUT.PUT('n.COUNT = ' || n.COUNT || ', ');
    DBMS_OUTPUT.PUT_LINE('n.LAST = ' || n.LAST);
END;
```

```
n.COUNT = 4, n.LAST = 4
n.COUNT = 7, n.LAST = 7
n.COUNT = 2, n.LAST = 2
```

Limit

```
DECLARE
    TYPE aa_type IS TABLE OF INTEGER INDEX BY PLS_INTEGER;
    aa aa_type; -- associative array

    TYPE va_type IS VARRAY(4) OF INTEGER;
    va va_type := va_type(2, 4); -- varray

    TYPE nt_type IS TABLE OF INTEGER;
    nt nt_type := nt_type(1, 3, 5); -- nested table

BEGIN
    aa(1) := 3;
    aa(2) := 6;
    aa(3) := 9;
    aa(4) := 12;
    DBMS_OUTPUT.PUT_LINE('aa.COUNT = ' || aa.count);
    DBMS_OUTPUT.PUT_LINE('aa.LIMIT = ' || aa.limit);

    DBMS_OUTPUT.PUT_LINE('va.COUNT = ' || va.count);
    DBMS_OUTPUT.PUT_LINE('va.LIMIT = ' || va.limit);

    DBMS_OUTPUT.PUT_LINE('nt.COUNT = ' || nt.count);
    DBMS_OUTPUT.PUT_LINE('nt.LIMIT = ' || nt.limit);
END;
```

```
aa.COUNT = 4
aa.LIMIT =
va.COUNT = 2
va.LIMIT = 4
nt.COUNT = 3
nt.LIMIT =
```

Prior и Next

✓ Позволяют перемещаться по коллекции

✓ Возвращают индекс предыдущего/следующего элемента (или null, если элемента нет)

```
DECLARE
    TYPE nt_type IS TABLE OF NUMBER;
    nt nt_type := nt_type(18, NULL, 36, 45, 5

BEGIN
    nt.DELETE(4);
    DBMS_OUTPUT.PUT_LINE('nt(4) was deleted.'

    FOR i IN 1 .. 7
    LOOP
        DBMS_OUTPUT.PUT('nt.PRIOR(' || i || '
        print(nt.PRIOR(i));
        DBMS_OUTPUT.PUT('nt.NEXT(' || i || ')
        print(nt.NEXT(i));
    END LOOP;
END;
```

```
nt(4) was deleted.
nt.PRIOR(1) =
nt.NEXT(1)  = 2
nt.PRIOR(2) = 1
nt.NEXT(2)  = 3
nt.PRIOR(3) = 2
nt.NEXT(3)  = 5
nt.PRIOR(4) = 3
nt.NEXT(4)  = 5
nt.PRIOR(5) = 3
nt.NEXT(5)  = 6
nt.PRIOR(6) = 5
nt.NEXT(6)  =
nt.PRIOR(7) = 6
nt.NEXT(7)  =
```

```
DECLARE
    TYPE NumTab IS TABLE OF employees.employee_id%TYPE;
    TYPE NameTab IS TABLE OF employees.last_name%TYPE;

    CURSOR c1 IS SELECT employee_id,last_name
        FROM   employees
        WHERE  salary > 10000
        ORDER  BY last_name;

    enums NumTab;
    names NameTab;
BEGIN
    SELECT employee_id, last_name
    BULK COLLECT INTO   enums, names
    FROM   employees
    ORDER  BY employee_id;

    OPEN c1;
    LOOP
        FETCH c1 BULK COLLECT INTO enums, names LIMIT 10;
        EXIT WHEN names.COUNT = 0;
        do_something();
    END LOOP;
    CLOSE c1;

    DELETE FROM emp_temp WHERE  department_id = 30
    RETURNING employee_id, last_name BULK COLLECT INTO enums, names;
END;
```

Forall

✓ посылает DML операторы из PL/SQL в SQL пачками, а не по одному

✓ может содержать только один DML оператор

```
DECLARE
    TYPE NumList IS TABLE OF NUMBER;
    depts NumList := NumList(10, 20, 30);

    TYPE enum_t IS TABLE OF employees.employee_id%TYPE;
    e_ids enum_t;

    TYPE dept_t IS TABLE OF employees.department_id%TYPE;
    d_ids dept_t;
BEGIN
    FORALL j IN depts.FIRST .. depts.LAST
        DELETE FROM emp_temp
        WHERE  department_id = depts(j)
        RETURNING employee_id, department_id BULK COLLECT INTO e_ids, d_ids;
END;
```

✓ SQL%ROWCOUNT

Exceptions in forall

- при возникновении исключения в любом из dml-операторов в цикле, транзакция полностью откатывается

- если описать обработчик ошибок, в нем можно зафиксировать успешно выполнившиеся операторы dml (это те операторы, которые выполнились до возникновения исключения).

- FORALL j IN collection.FIRST.. collection.LAST **SAVE EXCEPTIONS**

Генерит ORA-24381 в конце, если в цикле возникали исключения

- SQL%BULK_EXCEPTIONS

  .Count

  .ERROR_INDEX

  .ERROR_CODE -> SQLERRM(-(SQL%BULK_EXCEPTIONS(i).ERROR_CODE))

Collection exceptions

```
DECLARE
    TYPE NumList IS TABLE OF NUMBER;
    nums NumList;
BEGIN
    nums(1) := 1; -- raises COLLECTION_IS_NULL
    nums := NumList(1, 2);
    nums(NULL) := 3; -- raises VALUE_ERROR
    nums(0) := 3; -- raises SUBSCRIPT_BEYOND_COUNT
    nums(3) := 3; --raises SUBSCRIPT_OUTSIDE_LIMIT
    nums.Delete(1);
    IF nums(1) = 1 THEN ... -- raises NO_DATA_FOUND
END;
```

DBMS_SESSION.FREE_UNUSED_USER_MEMORY

- ✓ Процедура DBMS_SESSION.FREE_UNUSED_USER_MEMORY возвращает неиспользуемую более память системе
- ✓ В документации Oracle процедуру советуют использовать «редко и благоразумно».
- ✓ В случае подключения в режиме **Dedicated Server** вызов этой процедуры возвращает неиспользуемую PGA память операционной системе
- ✓ В случае подключения в режиме **Shared Server** вызов этой процедуры возвращает неиспользуемую память в **Shared Pool**

В каких случаях нужно освобождать память:

- ✓ Большие сортировки, когда используется вся область sort_area_size
- ✓ Компиляция больших PL/SQL пакетов, процедур или функций
- ✓ Хранение больших объемов данных в индексных таблицах PL/SQL

Summarizing

- ✓ Типы коллекций
- ✓ Ассоциативный массив (index by table)
- ✓ Varray
- ✓ Nested table
- ✓ Set Operations
- ✓ Логические операторы
- ✓ Методы коллекций
- ✓ Bulk Collect
- ✓ Forall