

Тестовое задание Backend-разработчик (LinkVideo)

Выполнил: Чернавский А.Ю.

Образцы кода на GitHub:

https://github.com/leha404/linkvideo_test

1. Классическая задача: есть 2 переменных a и b . Необходимо поменять местами их значения. Предложить не менее 3 вариантов.
2. Написать функцию, которая принимает на вход целое число $n > 0$, создает двумерный массив размера $n \times n$ и последовательно заполняет его $1, 2, \dots, n^2$ по спирали по часовой стрелке, начиная с первого элемента массива. Например, для $n = 4$ результат должен быть таким:

1 2 3 4

12 13 14 5

11 16 15 6

10 9 8 7

Задача 1:

В данной задаче можно рассмотреть стандартный пример – swap. Обмен значениями через буферную переменную.

```
// 1 - classic swap
c := a
a = b
b = c
fmt.Printf("\nSwap:\t\t a = %d, b = %d", a, b)
```

Допустимый для Go вариант – переприсваивание средствами языка (так же используется для объектов обменом по ссылкам)

```
// 2 – вариант, допустимый в go: переприсваивание
a, b = 1, 2
a, b = b, a
fmt.Printf("\nReAssign:\t a = %d, b = %d", a, b)
```

```
// 5 – swap objects через указатели
obj1 := &MyStruct{Value: 1}
obj2 := &MyStruct{Value: 2}
fmt.Printf("\nInit:\t\t obj1 = %o, obj2 = %o", obj1, obj2)

*obj1, *obj2 = *obj2, *obj1
fmt.Printf("\nSwapObjets:\t obj1 = %o, obj2 = %o", obj1, obj2)
fmt.Println()
```

И так же несколько вариантов, в зависимости от типа переменных – для целочисленных или для строк.

Варианты выше – универсальные

```
// 3 – только для int: арифметикой
a, b = 1, 2
a = a + b
b = a - b
a = a - b
fmt.Printf("\nMath:\t\t a = %d, b = %d", a, b)

fmt.Print("\n---")

// 4 – для string: через slices
str1, str2 := "string1", "str2"
fmt.Printf("\nInit str:\t str1 = %s, str2 = %s", str1, str2)

str1 = str1 + str2
str2 = str1[:len(str1)-len(str2)]
str1 = str1[len(str2):]
fmt.Printf("\nSlices Swap:\t str1 = %s, str2 = %s", str1, str2)

fmt.Print("\n---")
```

Задача 2:

Данная задача предполагает собой некий алгоритм. Поскольку у нас заполнение массива происходит по спирали, то я предположил, что должны быть некоторые «границы», которые ограничат движение заполнения.

Т.е. мы должны упираться не в границу нашего «квадрата», а в «границу движения»

Условно, если квадрат 5 на 5, то при первом движении направо мы идем до 5, НО при повторном во время спирали – до 4.

И это правило можно расширить на все направления, для чего был создан объект, хранящий это состояние.

```
You, 2 часа назад | 1 author (You)
type DirectionStruct struct {
    // "right", "down", "up", "left"
    // Init - "right"
    direction string

    // Boundaries
    // Индексы границ для каждого направления, чтобы указатель шел по спирали, а не до конца массива
    upBoundary    int
    downBoundary  int
    leftBoundary  int
    rightBoundary int
}
```

А далее вся реализация происходит через сравнение направления движения итератора, который заполняет массив и выполнения определенных действий по его сдвигу. (Вправо, вниз, влево, вверх и т.д. по спирали)

```

for val := 1; val <= n*n; val++ {
    switch directionObj.direction {
    case "right":
        if j < directionObj.rightBoundary-1 {
            // step right
            j++
            arr[i][j] = val
        } else {
            // step down
            i++
            directionObj.rightBoundary--
            directionObj.direction = "down"
            arr[i][j] = val
        }
    case "down":
        if i < directionObj.downBoundary-1 {
            // step down

```

Детальная реализация – в runTask2 функции.

Результаты выполнения работы:

```

Hello, World!

Task1
---
Init:      a = 1, b = 2
Swap:      a = 2, b = 1
ReAssign:  a = 2, b = 1
Math:      a = 2, b = 1
---
Init str:  str1 = string1, str2 = str2
Slices Swap: str1 = str2, str2 = string1
---
Init:      obj1 = &{1}, obj2 = &{2}
SwapObjets: obj1 = &{2}, obj2 = &{1}

Task2
---
n = 4
Output:
1      2      3      4
12     13     14     5
11     16     15     6
10     9      8      7

Task2
---
n = 5
Output:
1      2      3      4      5
16     17     18     19     6
15     24     25     20     7
14     23     22     21     8
13     12     11     10     9
Process 96615 has exited with status 0
Detaching

```

SQL Секция:

Для проверки скриптов использовал онлайн песочницу:

- <https://sqlfiddle.com/>

Скрипты можно посмотреть в папке SQL на GitHub

Задача 1:

1. Дана таблица

```
create table employees (  
    id number not null,  
    name varchar2(100) not null,  
    department_id number not null,  
    hire_date date not null  
    ...  
)
```

И дан анонимный PL/SQL блок:

```
declare

  v_result varchar2(1000);

begin

  select listagg(name, ', ')

    within group (order by hire_date)

  into v_result

  from employees

  where department_id = 2;

end;
```

Какие проблемы есть в этом коде? Предложите варианты решения этих проблем

Проверить код можно тут:

<https://sqlfiddle.com/oracle-plsql/online-compiler?id=058479db-b91c-41c1-b2e0-64bdd56248da>

Проблема тут видна невооруженным взглядом – при большом количестве записей в employees, соответствующих условию, наш ответ не поместится в ответ функции.

```
100  INSERT INTO employees(id, name, department_id, hire_date) VALUES (4, 'EmployeeerX', 2, CURRENT_DATE)
101  INSERT INTO employees(id, name, department_id, hire_date) VALUES (4, 'EmployeeerX', 2, CURRENT_DATE)
102  INSERT INTO employees(id, name, department_id, hire_date) VALUES (4, 'EmployeeerX', 2, CURRENT_DATE)
103
104  -- Если в pl/sql v_result = varchar2(1000), то эта строка переполнит длину склейки
105  INSERT INTO employees(id, name, department_id, hire_date) VALUES
106  (4, 'EmployeeerLAST', 2, CURRENT_DATE + 1);
107
```

Функция listagg склеивает name через запятую, сортируя по дате найма сотрудников.

Возможные варианты решения: расширить размер в типе, использовать другой тип или возвращать частями или массивом

```
10
11 declare
12     -- Нерабочий вариант
13     -- v_result varchar2(1000);
14
15     -- Один из возможных простых вариантов
16     -- (при условии, что мы доверяем данным и функция не будет меняться по логике)
17     -- Просто увеличиваем допустимое значение с запасом, например, в несколько раз
18     -- v_result varchar2(3000);
19
20     -- Так же можно изменить тип данных, если есть возможность
21     -- Если функция не анонимная, то ее тип может использоваться где-то еще
22     -- В местах, где вызывается функция
23     v_result clob;
24
25     -- И можно дополнительно возвращать чанками (кусочками), если varchar2 критично
26     -- 32767 - максимальный размер varchar2
27     v_chunk_size constant pls_integer := 32767;
28     v_pos pls_integer := 1;
29 begin
30     select listagg(name, ', ')
31     within group (order by hire_date)
32     into v_result
33     from employees
34     where department_id = 2;
35
36     while v_pos <= dbms_lob.getlength(v_result) loop
37         dbms_output.put_line(dbms_lob.substr(v_result, v_chunk_size, v_pos));
38         v_pos := v_pos + v_chunk_size;
39     end loop;
40 end;
```

Последний вариант (если функция будет не анонимной) надо использовать с осторожностью, поскольку на типе ответа данной функции может быть завязано что-то еще: другая процедура или функция.

Пример ответа (так же есть скрины на GitHub):

Execute

Share

Oracle PLSQL

Задача 2:

2. Дана таблица из предыдущего задания, и дан текстовый документ формата:

Иванов Иван Иванович

Петров Петр Петрович

...

Жмышенко Валерий Альбертович

Необходимо сделать список ФИО из тестового документа, которые отсутствуют в таблице employees.

Поскольку в песочнице нет доступа к файловой системе, то данные были предзаполнены, как будто из файла, но по документации Oracle это бы выглядело примерно как на скрине ниже:

```
-- Заполнение временной таблицы данными
insert into temp_fio (name) values ('ivan');
insert into temp_fio (name) values ('John');
insert into temp_fio (name) values ('sidr');

-- НО ПО ЗАДАНИЮ У НАС ФАЙЛ
-- В песочнице нет доступа к файловой системе, но примерный код выглядел бы так:
/*
declare
    F1 utl_file.file_type;
    L1 varchar2(32767);
begin
    -- Открытие файла для чтения
    F1 := utl_file.fopen('DIRECTORY', 'example.txt', 'r');

    -- Чтение данных из файла и вставка в таблицу temp_fio
    loop
        begin
            utl_file.get_line(F1, L1);
            insert into temp_fio (name) values (L1);
        exception
            when no_data_found then
                exit;
        end;
    end loop;

    -- Закрытие файла
    utl_file.fclose(F1);
end;
*/
```

Подход заключается в том, чтобы минимизировать обращение к базе. И если в худшем случае мы могли бы читать строку из файла и сравнивать сразу с таблицей – это было бы крайне неэффективно.

Мы можем создать временную таблицу, а потом по разнице данных получить ответ:

```
-- Создание временной таблицы
drop table if exists temp_fio;

create table temp_fio (
  name varchar2(100)
);

-- Заполнение временной таблицы данными
insert into temp_fio (name) values ('ivan');
insert into temp_fio (name) values ('John');
insert into temp_fio (name) values ('sidr');
```

```
drop table if exists employees;

create table employees (
  id number not null,
  name varchar2(100) not null,
  department_id number not null,
  hire_date date not null
);

-- Insert Section
INSERT INTO employees(id, name, department_id, hire_date) VALUES (1, 'John', 1, CURRENT_DATE);
INSERT INTO employees(id, name, department_id, hire_date) VALUES (2, 'Wick', 1, CURRENT_DATE);
INSERT INTO employees(id, name, department_id, hire_date) VALUES (3, 'Wayn', 1, CURRENT_DATE);

-- Запрос для нахождения ФИО, которых нет в таблице employees
select name
from temp_fio
where name not in (select name from employees);
```

Пример ответа:

```
51 INSERT INTO employees(id, name, department_id, hire_date) VALUES (3, 'Wayn', 1, CURRENT_DATE)
52
53 -- Запрос для нахождения ФИО, которых нет в таблице employees
54 select name
55 from temp_fio
56 where name not in (select name from employees);
```

Execute

Share

Oracle PLSQL ▼

NAME

ivan

sidr

Задача 3:

3. Дана таблица

```
create table events (  
  id_event number not null,  
  description varchar2(100) not null,  
  type number not null,  
  start_date date not null,  
  expiration_date date not null  
)  
  
partition by range (expiration_dt)  
interval (numtodsinterval(1,'DAY'));
```

В таблице содержится 100 млн записей. Необходимо обновить значение столбца start_date у всех записей в таблице, по условию:

если type = 1, то прибавить 1 день

если type = 2, то отнять 1 час

для остальных значений type - не изменять

Какие проблемы могут возникнуть при такой операции? Предложить наиболее быстрый и безопасный вариант обновления.

Поскольку моделировать миллион записей довольно проблематично, сделаем условный пример на 2 записях

```

drop table if exists events;

✓ create table events (
    id_event number not null,
    description varchar2(100) not null,
    type number not null,
    start_date date not null,
    expiration_date date not null
)
PARTITION BY RANGE (expiration_date)
-- Fix: minimum 1 date is needed
INTERVAL (NUMTODSINTERVAL(1, 'DAY'))
✓ (
    PARTITION p0 VALUES LESS THAN (TO_DATE('2023-01-01', 'YYYY-MM-DD'))
);

insert into events(id_event, description, type, start_date, expiration_date)
values(1, '123', 1, CURRENT_DATE - 1, CURRENT_DATE + 1);

insert into events(id_event, description, type, start_date, expiration_date)
values(2, '456', 2, CURRENT_DATE - 1, CURRENT_DATE + 1);

```

Основная проблема при большом количестве записей - медленное обновление. Так же оно может блокировать всю базу на время обновления.

Как вариант (не самый удачный) - запускать обновление как есть, но ночью или по событиям, чтобы обновлялось при изменении.

Как более адекватный вариант - разбить обновление на части (пакеты).

С процедурами и циклами я был знаком, но пришлось сделать небольшое исследование и посмотреть варианты решения.

Для нашей задач нам может помочь специальный тип в Oracle:

Курсор (CURSOR) - набор строк, возвращаемых запросом.

Итоговая функция обновления могла бы выглядеть так:

```
1 DECLARE
2   -- Определяем курсор
3   CURSOR curs
4   IS
5     SELECT id_event, start_date, type
6     FROM events
7     WHERE type IN (1, 2)
8     FOR UPDATE;
9
10  TYPE temp_type IS TABLE OF curs%ROWTYPE;
11  temp_table temp_type;
12
13  -- Размер пакета обновления
14  batch_limit PLS_INTEGER := 10000;
15
16 BEGIN
17   OPEN curs;
18   LOOP
19     -- BULK COLLECT, чтобы получить все строки одновременно
20     FETCH curs BULK COLLECT INTO temp_table LIMIT batch_limit;
21     EXIT WHEN temp_table.COUNT = 0;
22
23     FORALL i IN 1..temp_table.COUNT
24       UPDATE events
25       SET start_date = CASE
26         WHEN temp_table(i).type = 1 THEN temp_table(i).start_date + INTERVAL '1' DAY
27         WHEN temp_table(i).type = 2 THEN temp_table(i).start_date - INTERVAL '1' HOUR
28       END
29       WHERE id_event = temp_table(i).id_event;
30
31     -- Коммит после каждого пакета
32     COMMIT;
33   END LOOP;
34
35   -- Закрыть курсор
36   CLOSE curs;
37 END;
```

Пример ответа:

```
67
68  -- Коммит после каждого пакета
69  COMMIT;
70  END LOOP;
71
72  -- Закрыть курсор
73  CLOSE curs;
74  END;
75  /
76
77  -- Проверяем после обновления
78  select id_event, description, type,
79         TO_CHAR(start_date, 'YYYY-MM-DD HH24:MI:SS') AS start_date,
80         TO_CHAR(expiration_date, 'YYYY-MM-DD HH24:MI:SS') AS expiration_date
81  from events;
```

Execute

Share

Oracle PLSQL

ID_EVENT	DESCRIPTION	TYPE	START_DATE	EXPIRATION_DATE
1	123	1	2024-07-23 23:12:54	2024-07-25 23:12:54
2	456	2	2024-07-23 23:12:54	2024-07-25 23:12:54

ID_EVENT	DESCRIPTION	TYPE	START_DATE	EXPIRATION_DATE
1	123	1	2024-07-24 23:12:54	2024-07-25 23:12:54
2	456	2	2024-07-23 22:12:54	2024-07-25 23:12:54

Как можем видеть, для типа 1 дата увеличилась на 1 день в start_date, а для типа 2 время уменьшилось на 1 час (что соответствует условиям задачи)