Программирование на стороне сервера

Циклы в PL/pgSQL

Циклы в PL/pgSQL

- Циклы предназначены для многократного выполнения одной или нескольких инструкций
- 3 разновидности простых циклов:
  - безусловный цикл
  - цикл FOR
  - цикл WHILE
- Любой цикл состоит из двух частей:
  - **ограничители** – ключевые слова, определяющие начало и завершение цикла, а также условие выхода
  - **тело цикла** – последовательность исполняемых операторов внутри границ цикла
- Циклы могут быть вложенными

Безусловный цикл

```
[<<метка>>] LOOP                                    --ограничитель
    инструкции;
    [оператор_выхода;]                              --EXIT
    [оператор_перехода_на_новую_итерацию;]          --CONTINUE
END LOOP [метка];                                   --ограничитель
```

- *Операторы **EXIT**, **CONTINUE** и другие инструкции могут располагаться в любой последовательности внутри тела цикла
- Условие выхода из цикла не обязательно, но
- **Без оператора выхода цикл может стать бесконечным!**

Пример

```
DO LANGUAGE plpgsql
$block_name$
DECLARE
    v_ordid int    := 1;
    v_counter int := 0;
BEGIN
    LOOP
        INSERT INTO item(ordid, itemid)
        VALUES (v_ordid, v_counter);
        v_counter := v_counter + 2;
        v_ordid   := v_ordid + 1;
        EXIT WHEN v_ordid > 10;
    END LOOP;
END
$block_name$;
```

```
SELECT *
FROM public.item;
```

	ordid	itemid
1	1	0
2	2	2
3	3	4
4	4	6
5	5	8
6	6	10
7	7	12
8	8	14
9	9	16
10	10	18

Цикл WHILE

```
[<<метка>>]
WHILE условие LOOP     --условие WHILE оценивается в начале каждой итерации
    инструкции;
    [оператор_выхода;]
    [оператор_перехода_на_новую_итерацию;]
END LOOP [метка];
```

- Выполнение тела цикла происходит только если истинно условие, заданное после ключевого слова **WHILE**

```
begin
    WHILE v_salary < 10000 LOOP
        v_salary := v_salary + 1000;
    END LOOP;
    RAISE NOTICE ' %', v_salary ;
end
```

Простой цикл FOR

- Применяется, когда точно известно необходимое количество итераций

```
[<<метка>>]
FOR счётчик IN [REVERSE] выражение_1 .. выражение_2
[BY выражение_шага] LOOP
    инструкции;
    [оператор_выхода;]
    [оператор_перехода_на_новую_итерацию;]
END LOOP [метка];
```

- Пример:

```
FOR i IN 1..10 LOOP
    INSERT INTO item(ordid, itemid)
    VALUES(v_ordid, i);
END LOOP;
```

Пример цикла по записям

```
DO $$
DECLARE
   employees RECORD;
   i int = 1;
BEGIN
 RAISE NOTICE 'Start... ';
  FOR employees IN  --переменная получает 1 запись на каждой итерации цикла
       SELECT e.lastname , e.firstname , e.title
       FROM "HR"."Employees" e
       order by 3
    LOOP
        RAISE NOTICE '%. %, %  %',
                 i, employees.title, employees.lastname, employees.firstname;
        i = i+1;
    END LOOP;
    RAISE NOTICE 'Finish... ';
END;
$$;
```

Простой цикл FOR. Замечания

- Не объявляйте счетчик цикла
- Область действия переменной-счётчика – цикл, в котором она неявно объявлена
  - В начале выполнения цикла счётчик инициализируется значением нижней границы
- Границы и шаг могут быть только числовыми или приводимыми к целому числу (дробные числа округляются)
  - По умолчанию шаг = 1
- Ни одна из границ не может быть **NULL**
- Обратный отсчёт обеспечивается с помощью ключевого слова **REVERSE**
  - Границы диапазона в этом случае указываются в обратном порядке

Операторы EXIT и CONTINUE

- Оператор [условного] выхода из [отмеченного] цикла
  EXIT [метка] [**WHEN** условие]
- Оператор [условного] перехода к следующей итерации [отмеченного] цикла
  CONTINUE [метка] [**WHEN** условие]

```
DO $$
DECLARE
v_x INT := 0;
BEGIN
    LOOP                    -- после выполнения условия CONTINUE переходим сюда
        v_x := v_x + 1;
        CONTINUE WHEN v_x < 3;
        RAISE NOTICE 'x=%', v_x;
        EXIT WHEN v_x = 5;
    END LOOP;               -- после выполнения условия EXIT переходим сюда
END; $$
```

Использование меток циклов

```
DO $$
DECLARE
    v_counter INT := -5;
BEGIN
    <<outer_loop>>
    LOOP
        v_counter := v_counter + 1;
        RAISE NOTICE 'v_counter во внешнем цикле: %', v_counter;
        EXIT WHEN v_counter > 5;
        <<inner_loop>>
        FOR i IN 1..2 LOOP
            v_counter := v_counter + 2;
            IF v_counter > 3 THEN
                CONTINUE outer_loop;
            END IF;
            RAISE NOTICE 'v_counter во внутреннем цикле: %', v_counter;
        END LOOP inner_loop;
    END LOOP outer_loop;
    RAISE NOTICE 'v_counter вне циклов: %', v_counter;
END; $$
```