# Лабораторная работа № 7

**Создание пользовательских функций в PostgreSql**

**Синтаксис команды**

**CREATE**[ OR REPLACE ] **FUNCTION**  
имя\_функции**([ [** метод\_аргумента **][**имя\_аргумента **]** тип\_ аргумента **[,…] ])**  
**RETURNS**тип\_возвращаемого\_значения   
**AS  '**определение**'   
LANGUAGE  '**язык**'  
 [ WITH (**атрибут**[…])]**

* **CREATE FUNCTION** *имя\_функции* ([[ *метод\_аргумента* ] [*имя\_аргумента* ] *тип\_ аргумента* [,…] ]) - после ключевых слов CREATE FUNCTION указывается имя создаваемой функции, после чего в круглых скобках перечисляются аргументы, разделенные запятыми. Для каждого аргумента достаточно указать только тип, но при желании можно задать метод (in, out, inout; по умолчанию in) и  имя.  Если список в круглых скобках пуст, функция вызывается без аргументов (хотя сами круглые скобки обязательно должны присутствовать как в определении функции, так и при ее использовании). Ключевые слова  OR REPLACE  используются для изменения уже существующей функции.

* **RETURNS** тип\_возвращаемого\_значения   -  тип данных, возвращаемый функцией.

* **AS** '*определение*'  -  программное определение функции. В процедурных языках (таких, как PL/pgSQL) оно состоит из кода функции. Для откомпилированных функций С указывается абсолютный системный путь к файлу, содержащему объектный код.
* **LANGUAGE** '*язык*'. Название языка, на котором написана функция. В аргументе может передаваться имя любого процедурного языка (такого, как plpgsql или plperl, если соответствующая поддержка была установлена при компиляции),  С или SQL.   
  **[**

**WITH** ( *атрибут* [. ...] ) ] - *атрибут* может принимать два значения: iscachable  и  isstrict.  
      **iscachable**. Оптимизатор может использовать предыдущие вызовы функций для ускоренной обработки будущих вызовов с тем же набором аргументов. Кэширование обычно применяется при работе с функциями, сопряженны­ми с большими затратами ресурсов, но возвращающими один и тот же ре­зультат при одинаковых значениях аргументов.  
       **isstrict**. Функция всегда возвращает NULL в случае, если хотя бы один из ее аргументов равен NULL. При передаче атрибута isstrict результат возвраща­ется сразу, без фактического выполнения функции.

**1).**

--Создание функции, возвращающей Hello World

CREATE FUNCTION fun\_hello\_world()

RETURNS text

LANGUAGE 'sql'

AS $BODY$

SELECT 'Hello World!';

$BODY$;

-- ВЫЗОВ ФУНКЦИИ

select fun\_hello\_world();

**2).**

CREATE FUNCTION sum\_n\_product(x int, y int, OUT sum int, OUT prod int) AS $$

BEGIN

sum := x + y;

prod := x \* y;

END;

$$ LANGUAGE plpgsql;

-- ВЫЗОВ ФУНКЦИИ

select sum\_n\_product(10,20);

### **3). Объявление параметров функции**

Переданные в функцию параметры именуются идентификаторами $1, $2 и т. д. Либо псевдоним, либо цифровой идентификатор используются для обозначения параметра.

Создать псевдоним можно двумя способами. Предпочтительный способ это дать имя параметру в команде CREATE FUNCTION, например:

CREATE FUNCTION sales\_tax(subtotal real) RETURNS real AS $$

BEGIN

RETURN subtotal \* 0.06;

END;

$$ LANGUAGE plpgsql;

select sales\_tax(100);

Другой способ - это явное объявление псевдонима при помощи синтаксиса:

***имя*** ALIAS FOR $***n***;

Предыдущий пример для этого стиля выглядит так:

DROP FUNCTION sales\_tax;

CREATE OR REPLACE FUNCTION sales\_tax(real) RETURNS real AS $$

DECLARE

subtotal ALIAS FOR $1;

BEGIN

RETURN subtotal \* 0.06;

END;

$$ LANGUAGE plpgsql;

select sales\_tax(100);

CREATE OR REPLACE FUNCTION concat\_selected\_fields(in\_t sometablename) RETURNS text AS $$

BEGIN

RETURN in\_t.f1 || in\_t.f3 || in\_t.f5 || in\_t.f7;

END;

$$ LANGUAGE plpgsql;

**4).**Структура программного блока PL/pgSQL.

CREATE FUNCTION a\_function () RETURNS int4 AS $$

DECLARE

an\_integer int4;

BEGIN

an\_integer:=10\*10;

return an\_integer;

END;

$$ LANGUAGE plpgsql;

Вызов функции

SELECT a\_function() AS output;

5).Создание функции, возвращающей  целое значение (использование select into)

CREATE OR REPLACE FUNCTION apf (integer) RETURNS integer AS '

DECLARE

i integer;

-- Функция вычисляет количество работников в отделе $1

BEGIN

select count(\*) from emp into i where deptno = $1;

-- возвращение результата

return i;

END;

' LANGUAGE plpgsql;

Вызов функции

SELECT apf(10) AS output;

select \* from emp where deptno = 10;

6).Создание функции, возвращающей  вещественное значение (использование select into)

CREATE OR REPLACE FUNCTION avgves (integer) RETURNS real AS '

DECLARE

aves real;

-- Функция вычисляет среднюю з/п в отделе $1

BEGIN

select avg(sal) from emp into aves where deptno=$1;

-- возвращение результата

return aves;

END;

' LANGUAGE plpgsql;

SELECT distinct deptno, apf(10) AS kol, avgves(10) as sves

FROM emp where deptno = 10 ;

7). Создание функции, возвращающей столбец текстовых значений

CREATE OR REPLACE FUNCTION onef (integer) RETURNS SETOF character AS '  
--  Функция возвращает имена работников с зарплатой больше  $1  
select ename from emp where sal>$1;  
' LANGUAGE sql;

Использование функции, возвращающей столбец текстовых значений

SELECT  onef(3000) AS sname ;

8). Создание функции, возвращающей столбец записей

CREATE OR REPLACE FUNCTION manyf (integer) RETURNS SETOF record AS '

-- Функция возвращает информацию о работниках с з.п. больше $1

select ename, deptno, sal from emp where sal>$1 order by deptno;

' LANGUAGE sql;

SELECT manyf(3000) AS sname ;

9). Создание функции, возвращающей курсор

CREATE OR REPLACE FUNCTION fun1 (integer) RETURNS refcursor AS '

-- если входной параметр - 0 - функция возвращает список сотрудников отдела

-- если входной параметр - 1 - функция возвращает список отделов

DECLARE

ref refcursor;

BEGIN

if $1=0 then

OPEN ref FOR SELECT \* FROM emp ;

else

OPEN ref FOR SELECT \* FROM dept ;

end if;

RETURN ref;

END;

' LANGUAGE plpgsql;

select \* from fun1(0) ;

FETCH ALL From "<unnamed portal 1>"

select \* from fun1(1) ;

FETCH ALL From "<unnamed portal 2>"