**ЛАБОРАТОРНАЯ РАБОТА № 4. Хранимые процедуры**

Хранимая процедура − это отдельная программа, написанная на процедурном языке используемого сервера БД. Существует две разновидности хранимых процедур:

Синтаксис, применяемый при создании хранимых процедур, варьируется в зависимости от SQL-сервера. Один из допустимых стандартов - создания хранимых процедур при помощи PL/pgSQL.

PL/pgSQL (Procedural Language/PostGres Structured Query Language - процедурное расширение языка SQL, используемое в СУБД PostgreSQL. Этот язык предназначен для написания функций, триггеров.

Программы PL/pgSQL состоят из блоков. Такой метод организации программного кода обычно называется блочной структурой.

Программные блоки вводятся в командах SQL CREATE FUNCTION, которые используются для определения функций PL/pgSQL в базах данных PostgreSQL. Команда CREATE FUNCTION определяет имя функции, типы ее аргументов и возвращаемого значения. После определения функции (после AS) необходимо указать, что за этим следует тело функции. Символ ' или $BODY$ как раз указывают на это. После ключевого слова END также должен быть либо символ ', либо $BODY$.

Основной блок функции начинается с секции объявлений. Секция объявлений обозна-чается ключевым словом DECLARE, а каждое объявление завершается символом точки с запятой ‘;’. Все переменные объявляются (а также могут инициализироваться значениями по умолчанию) в секции объявлений программного блока. В объявлении указывается имя и тип переменной. После объявления переменных следует ключевое слово BEGIN, обозначающее начало основного программного блока. За ключевым словом BEGIN находятся команды, входящие в блок.

Конец программного блока обозначается ключевым словом END. Основной блок функции PL/pgSQL должен вернуть значение заданного типа, а все вложенные блоки (блоки, начинающиеся внутри других блоков) должны быть завершены до достижения ключевого слова END.

Программный блок PL/pgSQL может содержать неограниченное количество вложенных блоков, которые читаются и интерпретируются по тем же правилам, что и обычные блоки. В свою очередь, они могут содержать свои вложенные блоки.

**Примеры хранимых процедур**

CREATE OR REPLACE FUNCTION bootsshop.bad\_workers()

RETURNS character varying AS

$BODY$

declare

row bootsshop.sluzhaschie%rowtype;

res text := '';

begin

FOR row IN SELECT \*

FROM bootsshop.sluzhaschie LOOP

IF row.sales - row.plan < 0 then

res := res || row.slzh\_family || ' ' || row.slzh\_name || E'\n';

END IF;

END LOOP;

RETURN res;

end;

$BODY$

LANGUAGE plpgsql VOLATILE

COST 100;

ALTER FUNCTION bootsshop.bad\_workers()

OWNER TO postgres;

CREATE OR REPLACE FUNCTION bootsshop.count\_clients()

RETURNS integer AS

$BODY$

declare

num INTEGER;

begin

select into num count(cln\_family) from bootsshop.clienty;

RETURN num;

end;

$BODY$

LANGUAGE plpgsql VOLATILE

COST 100;

ALTER FUNCTION bootsshop.count\_clients()

OWNER TO postgres;

CREATE OR REPLACE FUNCTION bootsshop.delete\_product(id\_product1 character varying)

RETURNS integer AS

$BODY$

DECLARE

num\_product integer := 0;

BEGIN

SELECT INTO num\_product COUNT(id\_prd)

FROM bootsshop.tovary

WHERE id\_prd = $1;

IF num\_product = 0 THEN RETURN -1;

ELSE

DELETE FROM bootsshop.tovary

WHERE id\_prd = $1;

RETURN 1;

END IF;

END;

$BODY$

LANGUAGE plpgsql VOLATILE

COST 100;

ALTER FUNCTION bootsshop.delete\_product(character varying)

OWNER TO postgres;

CREATE OR REPLACE FUNCTION bootsshop.get\_client\_name(id\_klient1 integer)

RETURNS character varying AS

$BODY$

DECLARE

id\_klient1 ALIAS FOR $1;

fio\_klient1 text;

name\_klient1 text;

BEGIN

SELECT INTO fio\_klient1, name\_klient1

cln\_family , cln\_name

FROM bootsshop.clienty

WHERE clienty.id\_cln = id\_klient1;

RETURN fio\_klient1 || ' ' || name\_klient1;

END;

$BODY$

LANGUAGE plpgsql VOLATILE

COST 100;

ALTER FUNCTION bootsshop.get\_client\_name(integer)

OWNER TO postgres;