# Тема №6 - Хранимые процедуры и функции.

**Хранимая процедура** — это объект базы данных, представляющий собой набор SQL- инструкций, который компилируется один раз и хранится на сервере.

**Хранимая функция** отличается от хранимой процедуры, тем что хранимая функция всегда возвращает значение.

Хранимые процедуры и функции пишутся на процедурном языке, который зависит от конкретной СУБД. Они могут вызывать друг друга, читать и изменять данные в таблицах, и их можно вызвать из клиентского приложения, работающего с базой данных. Хранимые процедуры и функции обычно используются при выполнении часто встречающихся задач (например, сведение бухгалтерского баланса). Они могут иметь аргументы, возвращать значения, коды ошибок и иногда наборы строк и колонок. Однако последний тип процедур поддерживается не всеми СУБД. Укажем основные моменты которые стоит иметь в виду при работе с функциями и процедурами в SQL.

* В связи с тем, что процедурный кэш работает по принципу хранения либо самых ранних использовавшихся данных либо недавно использовавшихся данных, хранимая процедура остается в кэше до тех пор, пока не будет вытеснена оттуда другой часто исполняемой процедурой.
* Проверка синтаксических ошибок и компиляция происходят при первом запуске хранимой процедуры на исполнение.
* Процедуры могут либо выбирать данные, либо модифицировать их, но не то и другое одновременно.
* Хранимые процедуры могут использоваться как механизм безопасности. Пользователю предоставляется право запускать хранимую процедуру, но не право непосредственного доступа к данным таблицы.
* Процедуры могут принимать и возвращать параметры.

Далее опишем особенности хранимых функции в PostgreSQL, которые также справедливы для хранимых процедур.

SQL-функции выполняют произвольный список операторов SQL и возвращают результат последнего запроса в списке. В простом случае (не с множеством) будет возвращена первая строка результата последнего запроса. (Помните, что понятие "первая строка" в наборе результатов с несколькими строками определено точно, только если присутствует ORDER BY.) Если последний запрос вообще не вернёт строки, будет возвращено значение NULL.

Кроме того, можно объявить SQL-функцию как возвращающую множество (то есть, несколько строк), указав в качестве возвращаемого типа функции SETOF некий\_тип, либо объявив её с указанием RETURNS TABLE(колонки). В этом случае будут возвращены все строки результата последнего запроса.

Тело SQL-функции должно представлять собой список SQL-операторов, разделённых точкой с запятой. Точка с запятой после последнего оператора может отсутствовать. Если только функция не объявлена как возвращающая void, последним оператором должен быть SELECT, либо INSERT, UPDATE или DELETE с предложением RETURNING.

Любой набор команд на языке SQL можно скомпоновать вместе и обозначить как функцию. Помимо запросов SELECT, эти команды могут включать запросы, изменяющие данные (INSERT, UPDATE и DELETE), а также другие SQL-команды. Однако последней командой должна быть SELECT или команда с предложением RETURNING, возвращающая результат с типом возврата функции. Если же вы хотите определить функцию SQL, выполняющую действия, но не возвращающую полезное значение, вы можете объявить её как возвращающую тип void, что по сути является процедурой, поэтому намного логичнее использовать именно ей.

Простейший синтаксис операторов создания хранимых процедур:

CREATE [OR REPLACE] PROCEDURE procedure\_name(parameter\_list)

LANGUAGE language\_name

AS $$

stored\_procedure\_body;

$$;

Проще всего описать синтаксис хранимой процедуры на примере, показывающей уведомление.

CREATE OR REPLACE PROCEDURE alert\_param(IN text)

LANGUAGE 'plpgsql'

AS $$

BEGIN

RAISE NOTICE 'Вы ввели параметр %', $1;

END;

$$;

После создания хранимой процедуры вызовем её на исполнение, чтобы ознакомиться с полученными результатами, показанными на рисунке куку:

CALL public.alert\_param(

'Я процедура и ничего не возвращаю!'

)

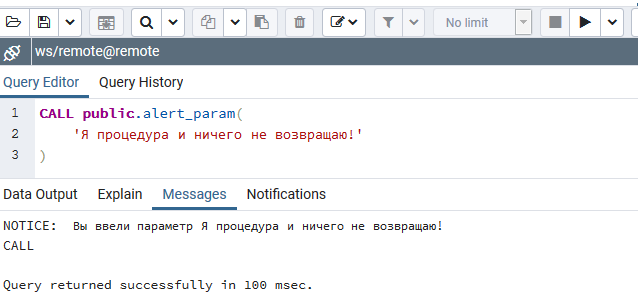


Рисунок куку – Уведомление процедуры

Хранимые процедуры поддерживают входные и выходные параметры.

Для объявления параметра необходимо знать следующее:

* Тип данных
* Заданное по умолчанию значение
* Обозначение входного параметра
* Имя

Объявление параметров происходит следующим образом:

[default (IN)|OUT] [TYPE] [NAME = ""]

Запись выше означает, что по умолчанию параметры являются входными, но лучше указывать это явно. Далее следует тип значения, можно указать имя. Если имя параметра не задано, то внутри процедуры к нему можно обратиться через ${num}, где num – порядковый номер параметра в списке, начиная с 1.

Для примера создадим процедуру, которая будет устанавливать цвет ткани по его названию:

CREATE OR REPLACE PROCEDURE public."insertClothColor"(IN integer, IN character varying)

LANGUAGE 'plpgsql'

AS $$

DECLARE

cnt INTEGER := 1;

BEGIN

SELECT

COALESCE(SUM("ID"), 0)

FROM colors

WHERE "NAME" = $2

INTO cnt

;

IF cnt = 0 THEN

INSERT INTO colors("NAME") VALUES ($2) RETURNING "ID" INTO cnt;

END IF;

UPDATE

cloth

SET

"COLOR" = cnt

WHERE

"ID" = $1

;

END;

$$;

Теперь проверим, как работает наша хранимая процедура – запустим её с помощью команды CALL:

CALL "insertClothColor"(125, 'синий');

CALL "insertClothColor"(126, 'бежевый');

CALL "insertClothColor"(127, 'бежевый');

Результат работы хранимой процедуры изображен на рисунке 53.

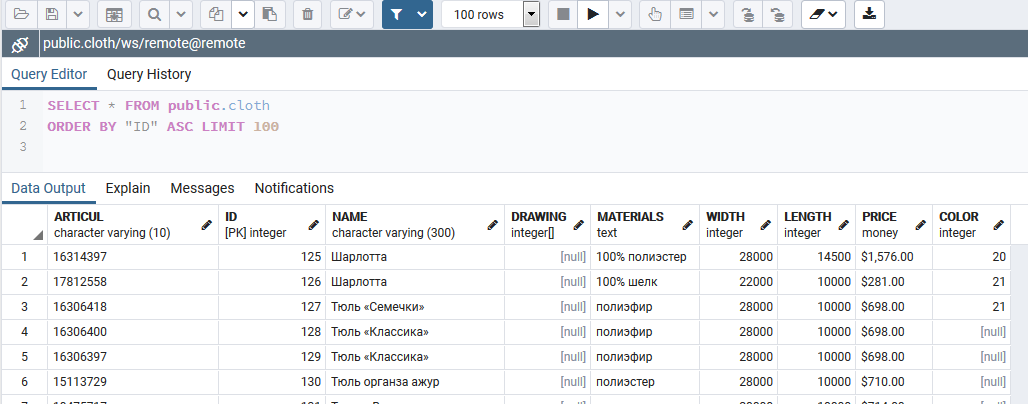


Рисунок 53 – Результаты работы хранимой процедуры

В общем случае синтаксис команды CREATE PROCEDURE бедет выглядеть следующим образом:

CREATE [ OR REPLACE ] PROCEDURE

name ( [ [ argmode ] [ argname ] argtype [ { DEFAULT | = } default\_expr ] [, ...] ] )

{ LANGUAGE lang\_name

| TRANSFORM { FOR TYPE type\_name } [, ... ]

| [ EXTERNAL ] SECURITY INVOKER | [ EXTERNAL ] SECURITY DEFINER

| SET configuration\_parameter { TO value | = value | FROM CURRENT }

| AS 'definition'

| AS 'obj\_file', 'link\_symbol'

} ..

## Функции

Синтаксис функции практические идентичен синтаксису процедуры, единственным отличием является возвращаемое значение.

CREATE [OR REPLACE] FUNCTION function\_name(parameter\_list)

RETURNS(value)

LANGUAGE language\_name

AS $$

stored\_function\_body;

$$;

Для примера, напишем функцию, возвращающую средний чек клиентов для менеджера, ID которого укажем в параметре.

CREATE OR REPLACE FUNCTION public.avg\_manager\_bill(mgr\_id integer)

RETURNS double precision

LANGUAGE 'plpgsql'

VOLATILE

PARALLEL UNSAFE

COST 100

AS

$BODY$

DECLARE

res double precision;

BEGIN

SELECT

round(AVG(CAST("BASE\_COST" AS NUMERIC)), 2)

FROM

"order" as "o"

WHERE

"MANAGER" = mgr\_id

INTO

res

;

RETURN res;

END;

$BODY$;

Вызвать функцию можно при помощи оператора SELECT. Результат показан на рисунке нене.

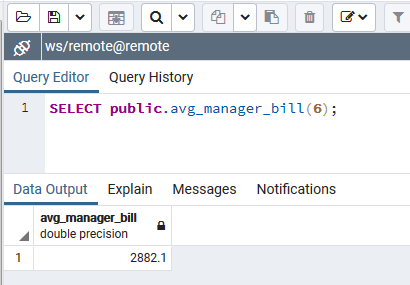


Рисунок нене – Результат работы функции

## Создание процедур и функций в pgAdmin

Созданные процедуры и функции можно найти в соответствующих разделах элементов текущей БД, как показано на рисунке ккк. Там же при помощи меню можно определить новые элементы.

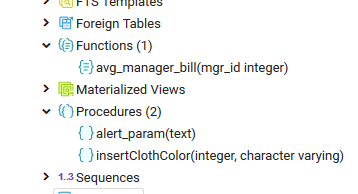


Рисунок ккк – Список процедур и функций в pgAdmin

При создании объектов в основных настройках указывается имя сущности, на вкладке «Определения» указываются язык и аргументы, на вкладке «Код» - основной код процедуры или функции, как например на рисунке изи. Итоговый код можно увидеть на вкладке «SQL».



Рисунок изи – Правильное заполнение вкладки «Код»