## Тема №6 - Хранимые процедуры и функции.

Хранимая процедура — это специальный вид процедуры, который выполняется сервером баз данных. Хранимые процедуры 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 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. Хранимые процедуры *–* это откомпилированные SQL - инструкции, которые хранятся на сервере. При их использовании следует принимать во внимание следующее:

* 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 выполняться любыми приложениями, что облегчает контроль над целостностью данных.
* Процедуры запускаются приложением, а не SQL Server.
* 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.
* Процедуры создаются как для постоянного, так и для временного использования (в течение текущего сеанса работы с SQL Server).

**Системные хранимые процедуры.** В процессе работы многие системные хранимые процедуры необходимы как быстрое средство манипулирования информацией из системных таблиц. Многие задачи администрирования SQL Server выполняются с помощью этих заранее определенных системных процедур, но помните, что вы тоже можете создать системные хранимые процедуры. По умолчанию системная хранимая процедура начинается с префикса sp\_. Для создания своих собственных системных хранимых процедур используйте тот же префикс. Системные процедуры могут исполняться в любой базе данных.

**Расширенные хранимые процедуры.** Для прямого доступа к системным ресурсам Windows NT в SQL Server встроены специальные средства – *расширенные хранимые процедуры* (extended stored procedure). Они позволяют обращаться к функциям, написанным в виде динамических библиотек Windows – DLL, что существенно повышает скорость их выполнения. Расширенные хранимые процедуры отличаются тем, что их имя начинается с символов хр\_. Эти процедуры используются для поддержки интегрированной модели безопасности и системы оповещения SQL Mail. Кроме того, есть пользовательские расширенные процедуры и процедуры общего назначения. Пользовательские расширенные хранимые процедуры могут быть написаны программистом с помощью Microsoft Open Data Services (MODS). MODS – это специальный 32-разрядный интерфейс программирования для разработки приложений доступа клиентов SQL Server к другим источникам данных.

Хранимые процедуры создаются с помощью команды CREATE PROCEDURE. Процедуры можно создавать только в текущей базе данных. Команда CREATE PROCEDURE не может смешиваться с другими SQL -командами в одном пакете. Локальные и глобальные временные хранимые процедуры похожи на временные таблицы. Для того чтобы процедура стала временной, необходимо добавить знак # перед ее именем (локальная процедура) или два знака # (глобальная процедура). Полное имя, включая знаки # и ##, не может превышать 20 символов. Хранимая процедура представляет собой сценарий, который хранится в базе данных и может принимать и передавать параметры, которые фактически не могут быть использованы в обычных сценариях.

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!'

)

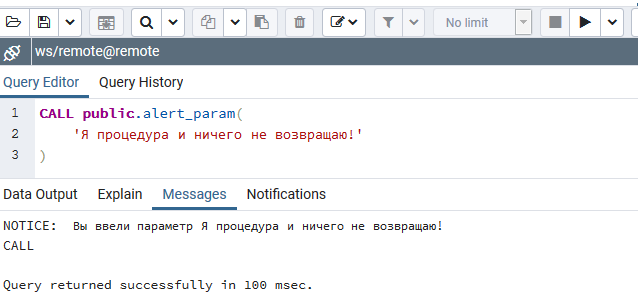


Рисунок куку – Уведомление процедуры

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.

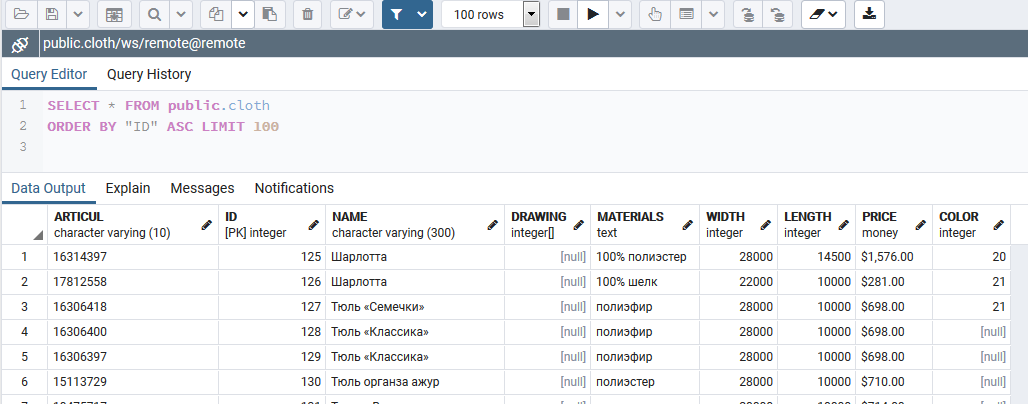


Рисунок 53 – Результаты работы хранимой процедуры

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

Определение функции подобно процедуре, однако нужно обязательно указать тип возвращаемого значения.

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.

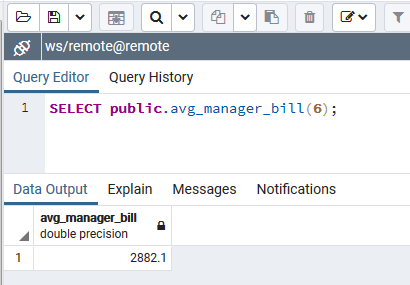


Рисунок нене – Результат работы функции