**Модуль 2. Триггеры, хранимые процедуры и пользовательские функции (6 пар)**

**Цели занятия:** изучить назначение пользовательских функций.

**План занятия**

**Теоретические сведения**

# Пользовательские функции

# Пользовательские функции.

В языках программирования обычно имеется два типа подпрограмм:

хранимые процедуры;

определяемые пользователем функции (ОПФ).

Хранимые процедуры состоят из нескольких инструкций и имеют от нуля до нескольких входных параметров, но обычно не возвращают никаких параметров. В отличие от хранимых процедур, функции всегда возвращают одно значение.

Итак. *Смысл хранимых процедур* сводится к *определенным SQL запросам, заранее скомпилированным, для быстроты работы*. Но что делать, если необходимо получить значение, а точнее набор значений из нее, а потом еще и вывести их на экран. ВЫЗЫВАТЬ ЧЕРЕЗ SELECT? Но мы уже определили, что вызывать хранимую процедуру можно только через EXECUTE. Так вот именно **для обработки набора данных** предоставляется еще один очень мощный инструмент - **Функции**!

Функции можно поделить на 3 вида:

1. **Скалярные** - функции, возвращающие одно значение определенного типа.
2. **Однозапросные** - функции, имеющие внутри только один Select запрос.
3. **Многозапросные** - функции, внутрь которых помещен большой кусок кода, включающий операторы ветвления, циклы, запросы и пр.

Общий синтаксис:

Create Function имя\_функции

(

@имя\_параметра1[=значение\_по\_умолчанию],

@имя\_параметра2[=значение\_по\_умолчанию],

...

@имя\_параметраN[=значение\_по\_умолчанию]

)

Returns возвращаемый\_тип\_данных

[WITH ENCRYPTION]

AS

Begin

ТЕЛО

Return значение

End

Вызов:

Либо

Select имя\_базы\_данных.имя\_владельца.имя\_функции (передача\_параметров)

--параметры в круглых скобках

Либо

Declare @имя\_переменной тип\_данных

Execute @имя\_переменной = имя\_функции значение1, значение2 ... значениеN/\*достаточно указать только имя функции. Значения передаются без круглых скобок!!!\*/

raiserror('значение = спецификатор\_типа',0,1,@имя\_переменной)

***Схема*** представляет собой именованный контейнер для объектов базы данных, позволяющий группировать объекты по отдельным пространствам имен.

**Владельцы схемы и разрешения**

Владельцем схемы может быть любой участник базы данных, при этом одному участнику может принадлежать несколько схем. К схеме можно применить правила безопасности, которые наследуются всеми объектами схемы. После установки разрешений доступа для схемы они автоматически применяются по мере добавления к схеме новых объектов. Пользователь может назначить новую схему, а несколько пользователей базы данных могут совместно использовать одну одну схему.

По умолчанию, если разработчик создает объект в схеме, он принадлежит участнику безопасности, являющемуся владельцем схемы, а не разработчику. Владение объектом можно передать с помощью инструкции Transact-SQL ALTER AUTHORIZATION. Схема может также содержать объекты, принадлежащие другим пользователям и иметь более детализированные разрешения, чем назначенные схеме, хотя это не рекомендуется из-за увеличения сложности управления разрешениями. Объекты можно перемещать из одной схемы в другую, а принадлежность схемы передавать от одного участника другому. Пользователей базы данных можно удалять, не влияя этим на схемы.

**Встроенные схемы**

SQL Server поставляется с десятью предварительно определенными схемами, имеющими одинаковое имя со встроенными пользователями и ролями базы данных. Они существуют главным образом ради обратной совместимости. Схемы, имеющие одинаковые имена с предопределенными ролями базы данных, можно удалить, если они не нужны. Невозможно удалить следующие схемы:

dbo

guest

sys

INFORMATION\_SCHEMA

Если их удалить из шаблона базы данных, эти схемы не появятся в новых базах данных.

**Схема dbo**

Схема dbo представляет собой схему по умолчанию для вновь созданной базы данных. Владельцем схемы dbo является учетная запись пользователя dbo. Схемой по умолчанию для пользователей, созданных с помощью команды Transact-SQL CREATE USER, является dbo.

Пользователи, которым назначена схема dbo, не наследуют разрешения пользовательской учетной записи dbo. Пользователи не наследуют разрешения схемы, их наследуют объекты базы данных, содержащиеся в схеме.

## Скалярные функции.

*Изначально хранимые процедуры* возвращают значения, чтобы *уведомить пользователя о какой-либо ошибке*, к примеру. В основном же они используются *для администрирования и быстрого выполнения набора запросов*. Мы же их использовали **для получения значений**, которые нужны нам. Это **лучше делать при помощи функций**.

***Пример 1.*** *Например, необходимо создать функцию, возвращающую слово "Четное", если переданное число четное, "Ноль" - если передали ноль и "Нечетное" - если переданное число нечетное.*

Create function Chet (@a int)

returns varchar(25)

As

begin

Declare @res varchar(25)

if(@a=0)

begin

Set @res='Ноль'

end

else if(@a%2=0)

begin

Set @res='Четное'

end

else

begin

Set @res='Нечетное'

end

return @res

end

Вызов:

Select dbo.Chet(123)

Либо:

Declare @c varchar(25)

Execute @c=Chet 28

raiserror('Число %s',0,1,@c)

**Пример 2.** *Необходимо написать функцию, которая вернет кол-во подсоединений к серверу в текущий момент.* Для этого нам понадобится таблица SYSPROCESSES, которая находится в базе данных Master. Поле LOGINAME сохраняет имена пользователей, подсоединенных к серверу. Все что требуется - их посчитать.

Create function UsersNum ()/\*Если принимаемых параметров нет,

то круглые скобки все равно должны быть указаны.\*/

returns int

as

begin

Declare @c int

Select @c=count(Distinct(loginame)) from sysprocesses

return @c

end

Вызов:

Select dbo.UsersNum ()

## Однозапросные функции.

Здесь необходимо вам рассказать о еще одном типе данных. Это не скалярный тип, называемый TABLE. Он используется, когда из функции, к примеру, необходимо вернуть результат, представляющий набор строк и столбцов, что и представляет из себя таблицу. При этом вы обязательно должны помнить, что для каждого поля возвращаемой таблицы, должно указываться свое имя.

У возвращаемой таблицы должен быть следующий принцип создания: во **From указываются псевдонимы для используемых таблиц и во всем запросе используются эти псевдонимы. Результирующим полям также должны быть даны псевдонимы.**

Create function SpisokKnig2 ()

returns table

as

return (

Select b.name as book,a.firstname+' '+a.lastname as author,

c.name as Category,t.name as theme/\*Для каждого поля указан псевдоним!!!\*/

from books b,authors a,categories c,themes t/\*Для каждой таблицы указан псевдоним!!!\*/

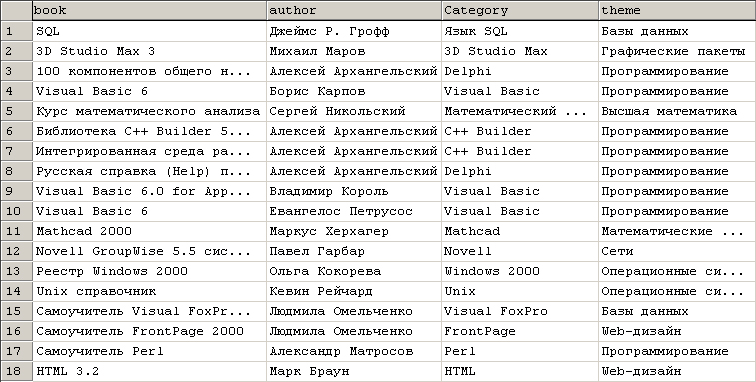
where b.id\_author=a.id and b.id\_category=c.id

and b.id\_themes=t.id

)

Вызов:

Select \* from dbo.SpisokKnig()



## Многозапросные функции.

Весь смысл этих функций состоит в том, что результирующая таблица не всегда создается при помощи одного запроса. Точнее, более ни менее нужные функции всегда создаются при помощи набора запросов.

Существует возможность создать переменную, типа Table. Для этого служит следующий синтаксис:

Declare @имя\_переменной (

имя\_поля1 тип\_данных спецификаторы,

имя\_поля2 тип\_данных спецификаторы,

...

имя\_поляN тип\_данных спецификаторы

)

Общий синтаксис создания многозапросной функции:

Create Function имя\_функции

(

@имя\_параметра1[=значение\_по\_умолчанию],

@имя\_параметра2[=значение\_по\_умолчанию],

...

@имя\_параметраN[=значение\_по\_умолчанию]

)

Returns @имя\_переменной TABLE {определение полей возвращаемой таблицы}

[WITH ENCRYPTION]

AS

Begin

ТЕЛО

Сохранение в возвращаемую переменную требуемых значений

RETURN

End

Create function Books\_By\_Authors ()

returns @Books\_Authors table (author varchar(25), amm int)/\*Определяем переменную и

тип возвращаемой таблицы!!!\*/

as

begin

Declare @temp\_books table (author1 varchar(25), amm1 int)/\*Объявляем временную таблицу!!!\*/

insert @temp\_Books/\*Записываем в нее список авторов и кол-во взятых студентами книг этих авторов!!!\*/

Select authors.firstname+' '+authors.lastname,count(s\_cards.id\_book)

from authors,books,s\_cards

where authors.id=books.id\_author and books.id=s\_cards.id\_book

group by authors.firstname,authors.lastname

insert @temp\_Books/\*Дописываем в нее список авторов и

кол-во взятых преподавателями книг этих авторов!!!\*/

Select authors.firstname+' '+authors.lastname,count(t\_cards.id\_book)

from authors,books,t\_cards

where authors.id=books.id\_author and books.id=t\_cards.id\_book

group by authors.firstname,authors.lastname

Declare @temp\_books2 table (author2 varchar(25), amm2 int)/\*Объявляем еще одну временную таблицу!!!\*/

insert @temp\_books2/\*Записываем в нее содержимое первой временной таблицы,

сумируя при этом кол-во книг одного автора!!!\*/

Select t.author1,sum(t.amm1)

from @temp\_books t

group by t.author1

insert @Books\_authors/\*Записываем содержимое в возвращаемую переменную!!!\*/

Select t.author2, t.amm2

from @temp\_books2 t

return

end

Вызов:

Select \* from Books\_By\_Authors()

Результат:

# C:\Users\admin\Google Диск\Мои документы\ШАГ\MS SQL\MS SQL\Уроки ПСТ\Урок 6\program\img\pict10.jpg