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

**Цели занятия:** изучить назначение хранимых процедур, способы их создания.

**План занятия**

**Теоретические сведения**

# Хранимые процедуры

Представьте, что каждый день, по несколько раз вам приходится набирать один и тот же набор запросов, изменяя в нем несколько параметров, например критерий поиска. Конечно же, вы сохраните весь набор в текстовый файл, откуда и будете его копировать. SQL Server предоставляет вам очень мощный механизм хранимых процедур.

У вас может возникнуть вопрос, а почему бы не пользоваться Представлениями? Есть несколько основных положительных моментов у хранимых процедур:

1. После первого выполнения, хранимая процедура компилируется, и код ее выполнения хранится в кэше в оперативной памяти, что очень повышает быстродействие выполнения хранимой процедуры относительно обычных запросов и представлений.
2. Любой пользователь может получить доступ к хранимой процедуре, при этом он может не иметь права на работу с теми объектами, которые она в себя включает.

**Хранимая процедура** - это набор операторов T-SQL, который компилируется системой SQL Server в единый «план исполнения». Хранимые процедуры T-SQL аналогичны процедурам в других языках программирования в том смысле, что они допускают входные параметры и возвращают выходные значения в виде параметров или сообщения о состоянии (успешное или неуспешное завершение). Все операторы процедуры обрабатываются при вызове процедуры. Они могут использоваться различными пользователями для согласованного повторяемого выполнения одинаковых задач и даже в различных приложениях.

Хранимая процедура — это специальный тип пакета инструкций Transact-SQL, созданный, используя язык SQL и процедурные расширения. Основное различие между пакетом и хранимой процедурой состоит в том, что последняя сохраняется в виде объекта базы данных. Иными словами, хранимые процедуры сохраняются на стороне сервера, чтобы улучшить производительность и постоянство выполнения повторяемых задач.

Компонент Database Engine поддерживает хранимые процедуры и системные процедуры. Хранимые процедуры создаются таким же образом, как и все другие объекты баз данных, т. е. при помощи языка DDL. Системные процедуры предоставляются компонентом Database Engine и могут применяться для доступа к информации в системном каталоге и ее модификации.

При создании хранимой процедуры можно определить необязательный список параметров. Таким образом, процедура будет принимать соответствующие аргументы при каждом ее вызове. Хранимые процедуры могут возвращать значение, содержащее определенную пользователем информацию или, в случае ошибки, соответствующее сообщение об ошибке.

Хранимая процедура предварительно компилируется перед тем, как она сохраняется в виде объекта в базе данных. Предварительно компилированная форма процедуры сохраняется в базе данных и используется при каждом ее вызове. Это свойство хранимых процедур предоставляет важную выгоду, заключающуюся в устранении (почти во всех случаях) повторных компиляций процедуры и получении соответствующего улучшения производительности. Это свойство хранимых процедур также оказывает положительный эффект на объем данных, участвующих в обмене между системой баз данных и приложениями. В частности, для вызова хранимой процедуры объемом в несколько тысяч байтов может потребоваться меньше, чем 50 байтов. Когда множественные пользователи выполняют повторяющиеся задачи с применением хранимых процедур, накопительный эффект такой экономии может быть довольно значительным.

Хранимые процедуры можно также использовать для следующих целей:

управления авторизацией доступа;

создания аудиторского следа действий с таблицами баз данных.

Использование хранимых процедур предоставляет возможность управления безопасностью на уровне, значительно превышающем уровень безопасности, предоставляемый использованием инструкций GRANT и REVOKE, с помощью которых пользователям предоставляются разные привилегии доступа. Это возможно вследствие того, что авторизация на выполнение хранимой процедуры не зависит от авторизации на модифицирование объектов, содержащихся в данной хранимой процедуре.

Хранимые процедуры, которые выполняют аудит операций записи и/или чтения таблиц, предоставляют дополнительную возможность обеспечения безопасности базы данных. Используя такие процедуры, администратор базы данных может отслеживать модификации, вносимые в базу данных пользователями или прикладными программами.

Общий синтаксис:

Create Procedure имя\_хранимой\_процедуры

As

Запрос\_SQL

Чтобы потом ее вызвать, необходимо указать:

Execute имя\_хранимой\_процедуры

**Пример.** *Необходимо создать хранимую процедуру, выводящую на экран список студентов, не вернувших книги (база данных Library).*

Create Procedure Taken\_books

As

Select students.firstname,students.lastname,books.name

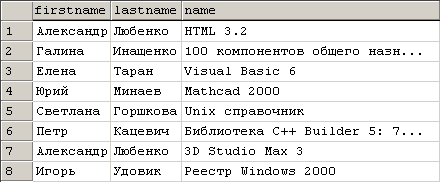
From students,s\_cards,books

Where students.id=s\_cards.id\_student and s\_cards.id\_book=books.id

and s\_cards.datein is null

Чтобы получить результат достаточно указать:

Execute Taken\_books



Существует *возможность передавать параметры в хранимую процедуру*.

**Например.** *Нужно показать книги определенного автора, при этом имя автора необходимо передать при вызове.*

Чтобы указать передаваемые параметры и возвращаемое значение используется следующий синтаксис:

Create Procedure имя\_хранимой\_процедуры

@аргумент1 тип\_данных[=значение\_по\_умолчанию][Output],

@аргумент2 тип\_данных[=значение\_по\_умолчанию][Output],

...

@аргументN тип\_данных[=значение\_по\_умолчанию][Output]

As

Запрос\_SQL

**Output** определяет, что этот аргумент является возвращаемым значением этой функции. Об этом чуть позже, пока разбираем передачу параметров.

**Пример.** *Нужно показать книги определенной тематики, при этом шаблон названия тематики необходимо передать при вызове (многотабличная books).*

Create Procedure Books\_By\_Theme

@Name varchar(25)

As

Select books.name, themes.name

from books,themes

where books.id\_theme=themes.id and

themes.name like @Name

Вызываем:

Execute Books\_By\_theme '[Г-М]%'

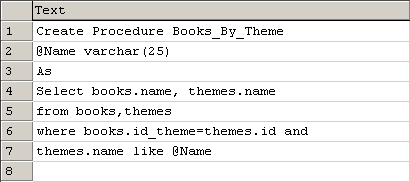
Получаем список книг, у которых первая буква в названии тематики лежит в диапазоне от Г до М.

Как видите, при вызове указывается передаваемый параметр. Если их несколько, то они перечисляются через запятую.

Существует служебная хранимая процедура, позволяющая просмотреть синтаксис запроса, который использовался для создания хранимой процедуры. Она называется **SP\_HELPTEXT.**

Например:

Execute sp\_helptext Books\_By\_Theme



Но иногда необходимо скрыть текст запроса от любопытных глаз. Для этого, при создании хранимой процедуры указывается **ключевое слово WITH ENCRYPTION**. Общий синтаксис:

Create Procedure имя\_хранимой\_процедуры

@аргумент1 тип\_данных[=значение\_по\_умолчанию][Output],

@аргумент2 тип\_данных[=значение\_по\_умолчанию][Output],

...

@аргументN тип\_данных[=значение\_по\_умолчанию][Output]

With Encryption

As

Запрос\_SQL

При указании With Encryption текст хранимой процедуры шифруется, и просмотреть его нельзя ни при помощи хранимой процедуры sp\_helptext ни при помощи Enterprise manager.

## Возвращаемые значения

Существует возможность вернуть какие-либо значения из функции. Для этого существует несколько вариантов.

**Первый.** Указать при определении параметров спецификатор OUTPUT у тех параметров, которые получат результат. Для начала необходимо научиться объявлять собственные переменные. Для этого используется следующий синтаксис:

Declare @имя\_переменной тип\_данных,

@имя\_переменной2 тип\_данных,

...

@имя\_переменнойN тип\_данных

Локальные переменные являются важным процедурным расширением языка Transact-SQL. Они применяются для хранения значений любого типа в пакетах и подпрограммах. Локальными они называются по той причине, что они могут бытьиспользованы только в том пакете, в котором они были объявлены.

Все локальные переменные пакета объявляются, используя инструкцию DECLARE. Определение переменной состоит из имени переменной и ее типа данных. Имена локальных переменных в пакете всегда начинаются с префикса @.

Присвоение значений локальной переменной осуществляется:

* используя специальную форму инструкции SELECT;
* используя инструкцию SET;
* непосредственно в инструкции DECLARE посредством знака = (например, @extra\_budget MONEY = 1500).

Чтобы записать значения в переменные используется следующий синтаксис:

ВНИМАНИЕ!!!

Select имя\_переменной1 = значение1, имя\_переменной2 = значение2, ... имя\_переменнойN = значениеN

Либо

Set имя\_переменной1 = значение1

Set имя\_переменной2 = значение2

...

Set имя\_переменнойN = значениеN

Чтобы вызвать хранимую процедуру и получить возвращаемые значения используется следующий синтаксис: Предположим, что у хранимой процедуры 4 параметра и последних два нужно получить.

ВНИМАНИЕ!!!

Execute имя\_хранимой\_процедуры значение1, значение2, имя\_переменной1 Output, имя\_переменной2 Output

Если не указать Output, то значения не сохраняться!!!

**Задание:** *Необходимо создать хранимую процедуру, которая вернет имя и фамилию студента, который взял наибольшее кол-во книг.*

Надеюсь, вы понимаете, что без представления здесь не обойтись.

Создаем представление, которое определяет студентов и кол-во взятых каждым из них книг.

Create View MyView1

as

Select students.firstname,students.lastname,count(s\_cards.id\_book)as c

from students,s\_cards

where students.id=s\_cards.id\_student

group by students.firstname,students.lastname

Создаем хранимую процедуру:

Create Procedure Best\_student @n varchar(25) output, @s varchar(25) output

/\*создаем хранимую процедуру с двумя параметрами, в которые будет записан результат.\*/

As

Select @n=firstname,@s=lastname /\*В переменные записываем интересующие нас значения.\*/

from MyView1

where c=(select max(c) from MyView1)

Вызываем хранимую процедуру и показываем результат:

Declare @name varchar(25), @surname varchar(25) /\*Объявляем две переменные для хранения

результата хранимой процедуры\*/

Execute Best\_Student @name output,@surname output /\*Вызываем хранимую процедуру и

получаем результат.\*/

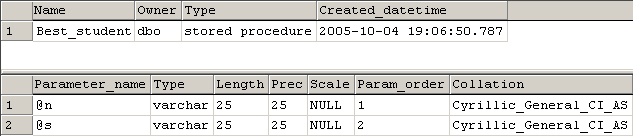
Select 'Best student name: ',@name+' '+@surname /\*Склеиваем две строки и показываем

на экран результат.\*/

Есть возможность узнать структуру хранимой процедуры, т.е. названия переменных и их типы данных, их порядок и пр. Для этого служит та же хранимая процедура SP\_HELP но по отношению к хранимой процедуре.

Пример:

execute sp\_help Best\_student



Второй вариант вернуть значение - использовать ключевое слово return. Тогда можно не указывать параметры, в которые будет записан результат. Но **при помощи return можно вернуть только одно целочисленное значение**.

Если вы хотите получить результат хранимой процедуры, возвращающей значение через return, необходимо использовать следующий синтаксис:

Declare @имя\_переменной

Execute @имя\_переменной = имя\_хранимой процедуры значение1, значение2 ... значениеN

Select 'Значение переменной = ', @имя\_переменной

Пример хранимой процедуры, складывающей 2 числа, переданных в качестве параметров.

Create Procedure MySum @a int,@b int

As

Declare @s int

Set @s=@a+@b

return @s

Вызов:

Declare @Summ int

Execute @Summ=MySum 1,25

Select 'Summa = ',@Summ

C:\Users\admin\Google Диск\Мои документы\ШАГ\MS SQL\MS SQL\Уроки ПСТ\Урок 6\program\img\pict5.jpg

Пример. Необходимо создать хранимую процедуру, которая вернет кол-во взятых с библиотеки книг и преподавателями и студентами.

Create Procedure Total\_books\_am --создаем хранимую процедуру

AS

Declare @st\_books\_am int,

@tch\_books\_am int,

@total int

Select @st\_books\_am=count(s\_cards.id\_book) from s\_cards --получаем кол-во книг, взятых студентами

Select @tch\_books\_am=count(t\_cards.id\_book) from t\_cards --получаем кол-во книг, взятых преподавателями

Set @total=@st\_books\_am+@tch\_books\_am --сумируем полученные данные

return @total --возвращаем полученную сумму

Вызов хранимой процедуры.

Declare @amm int

Execute @amm=Total\_books\_am

Select 'Total taken books ammount = ',@amm

Результат:

C:\Users\admin\Google Диск\Мои документы\ШАГ\MS SQL\MS SQL\Уроки ПСТ\Урок 6\program\img\pict4.jpg

При работе с хранимыми процедурами, существует набор ограничений:

1. В хранимой процедуре не может быть двух переменных с одним и тем же именем.
2. В хранимой процедуре не может быть больше 1024 параметров.
3. В хранимой процедуре нельзя использовать директивы:
   * Create View
   * Create Procedure
   * Create Trigger
   * Create Role
   * Create Rule
   * Create Default

## Операторы ветвления.

При написании хранимых процедур можно использовать операторы ветвления. Это известные вам if, else if, else. Блоки, которые должны быть выполнены, заключаются между ключевыми словами Begin и End.

Есть специальная функция, позволяющая, изначально, выдавать сообщение об ошибке. Но ее очень удобно использовать для компоновки выводимой на экран строки. Она называется **Raiserror**.

Raiserror ('строка со спецификаторами',степень\_тяжести\_ошибки,состояние\_ошибки\_на\_момент\_вызова, подставляемые\_переменные);

/\*Надеюсь вам это напомнило функцию printf.\*/

В качестве спецификаторов может указываться:

1. %d - целое число
2. %s - строка
3. %u - беззнаковое целое

Дробные, к сожалению, не поддерживаются.

Степень тяжести указывается в пределах от 0 до 25.

От 0 до 18 - могут указываться пользователями, а

от 19 до 25 - критические ошибки, которые могут указывать только члены группы sysadmin.

В случае таких ошибок подсоединение клиента с сервером разрывается

Состояние\_ошибки\_на\_момент\_вызова

Должно быть указано значение от 0 до 127.

Подставляемые\_переменные - те переменные, которые должны быть подставлены на место спецификаторов.

Пример. Создаем хранимую процедуру, возвращающую максимальное из двух чисел.

Create Procedure MyMax @a int, @b int

as

if(@a>@b)

return @a

else

return @b

Вызов:

Declare @max int

Execute @max=MyMax 8, 9

raiserror('Max from %d and %d is %d',0,1,8,9,@max)--используем функцию для компонования строки

**Результат:**

C:\Users\admin\Google Диск\Мои документы\ШАГ\MS SQL\MS SQL\Уроки ПСТ\Урок 6\program\img\pict6.jpg

Пример. Необходимо написать хранимую процедуру, выводящую на экран книги и цены по указанной категории, при этом необходимо указывать направление сортировки. 0 - сортировка результата по цене по убыванию, 1 - по возрастанию, любое другое - без сортировки. (многотабличная Books).

Создание хранимой процедуры:

Create Procedure MySort @i varchar(25), @c int

as

if(@c=0)

begin

select books.name,books.price

from books,press

where books.id\_press=press.id and press.name like @i

order by 2 desc

end

else if (@c=1)

begin

select books.name,books.price

from books,press

where books.id\_press=press.id and press.name like @i

order by 2 asc

end

else

begin

select books.name,books.price

from books,press

where books.id\_press=press.id and press.name like @i

end

Вызов.

Execute MySort 'BHV%' ,1

Получаем список книг, издательств BHV, отсортированный по цене по возрастанию

## Цикл.

Еще одна возможность которая дана - использовать цикл while.

while (логическое\_выражение)

begin

тело

end

Также внутри цикла можно использовать операторы Break и Continue, использующиеся в том случае, если необходимо внутри цикла перепроверить условие и относительно него либо закончить выполнение цикла (Break), либо его продолжить (Continue).

Пример на хранимую процедуру, которая считает сумму чисел в диапазоне, переданном ей в качестве параметров. (диапазон открытый).

Create Procedure DistSum @a int, @b int

as

Declare @s int

Set @s=@a

while @a<@b

begin

Set @s=@s+@a+1

Set @a=@a+1

end

return @s

Вызов:

Declare @s int

Execute @s=DistSum 2,4

raiserror('Summa chisel v diapazone = %d',0,1,@s)

Результат:

C:\Users\admin\Google Диск\Мои документы\ШАГ\MS SQL\MS SQL\Уроки ПСТ\Урок 6\program\img\pict7.jpg