ОБРАЩАЮ ВНИМАНИЕ!!! Результатом работы процедуры является или какой-то изменение в базе, или набор значений предназначенных для вывода в интерфейс. Ситуация, когда процедура просто что-то печатает на экране в SQL Server

Management Studio - в корне неверна! Это значит, что работа процедуры никому не нужна, её никто не увидит. Пользователь не знает, не видит, никогда не использует Management Studio!

Примеры таких процедур: расчет суммарной стоимости платежей, по контрагентам, расчет прибыли гостиницы за заданный период, расчет пробега автомобилей данной марки в данном сезоне и т.д.

Запрещается использовать в процедурах "странные" константы или условия, которые не могут быть изменены без переписывания кода процедуры.

Процедуры должны иметь входные, выходные параметры. В процедурах должны быть использованы условные и циклические инструкции, курсоры.

Хранимые процедуры представляют собой набор команд, состоящий из одного или нескольких операторов SQL или функций и сохраняемый в базе данных в откомпилированном виде.

Типы хранимых процедур

Системные хранимые процедуры предназначены для выполнения раз- личных административных действий. Практически все действия по администрированию сервера выполняются с их помощью. Можно сказать, что системные хранимые процедуры являются интерфейсом, обеспечивающим работу с системными таблицами. Системные хранимые процедуры имеют префикс sp\_, хранятся в системной базе данных и могут быть вызваны в контексте любой другой базы данных.

Пользовательские хранимые процедуры реализуют те или иные дейст- вия. Хранимые процедуры – полноценный объект базы данных. Вследствие этого каждая хранимая процедура располагается в конкретной базе дан- ных, где и выполняется.

Временные хранимые процедуры существуют лишь некоторое время, после чего автоматически уничтожаются сервером. Они делятся на ло- кальные и глобальные. Локальные временные хранимые процедуры могут быть вызваны только из того соединения, в котором созданы. При созда- нии такой процедуры ей необходимо дать имя, начинающееся с одного символа #. Как и все временные объекты, хранимые процедуры этого типа автоматически удаляются при отключении пользователя, перезапуске или остановке сервера. Глобальные временные хранимые процедуры доступны для любых соединений сервера, на котором имеется такая же процедура. Для ее определения достаточно дать ей имя, начинающееся с символов ##. Удаляются эти процедуры при перезапуске или остановке сервера, а также при закрытии соединения, в контексте которого они были созданы.

Создание, изменение хранимых процедур

page11image62446912page11image62300496

Создание хранимой процедуры предполагает решение следующих за- дач: планирование прав доступа. При создании хранимой процедуры следует учитывать, что она будет иметь те же права доступа к объектам базы данных, что и создавший ее пользователь; определение параметров храни- мой процедуры, хранимые процедуры могут обладать входными и выход- ными параметрами; разработка кода хранимой процедуры. Код процедуры может содержать последовательность любых команд SQL, включая вызов других хранимых процедур.

Синтаксис оператора создания новой или изменения имеющейся хранимой процедуры в обозначениях MS SQL Server:

AS

sql\_оператор [...n]  
Рассмотрим параметры данной команды.

Используя префиксы sp\_, #, ##, создаваемую процедуру можно определить в качестве системной или временной. Как видно из синтаксиса команды, не допускается указывать имя владельца, которому будет принадлежать создаваемая процедура, а также имя базы данных, где она должна быть размещена. Таким образом, чтобы разместить создаваемую хранимую процедуру в конкретной базе данных, необходимо выполнить команду CREATE PROCEDURE в контексте этой базы данных. При обращении из тела хранимой процедуры к объектам той же базы данных можно использовать укороченные имена, т. е. без указания имени базы данных. Когда же требуется обратиться к объектам, расположенным в других базах данных, указание имени базы данных обязательно.

Для передачи входных и выходных данных в создаваемой хранимой процедуре имена параметров должны начинаться с символа @. В одной хранимой процедуре можно задать множество параметров, разделенных запятыми. В теле процедуры не должны применяться локальные переменные, чьи имена совпадают с именами параметров этой процедуры. Для определения типа данных параметров хранимой процедуры подходят любые типы данных SQL, включая определенные пользователем. Однако тип данных CURSOR может быть использован только как выходной параметр хранимой процедуры, т.е. с указанием ключевого слова OUTPUT.

Наличие ключевого слова OUTPUT означает, что соответствующий параметр предназначен для возвращения данных из хранимой процедуры. Однако это вовсе не означает, что параметр не подходит для передачи значений в хранимую процедуру. Указание ключевого слова OUTPUT предписывает серверу при выходе из хранимой процедуры присвоить текущее значение параметра локальной переменной, которая была указана при вызове процедуры в качестве значения параметра. Отметим, что при указании ключевого слова OUTPUT значение соответствующего параметра при вызове процедуры может быть задано только с помощью локальной переменной. Не разрешается использование любых выражений или констант, допустимое для обычных

page12image62798256

{CREATE | ALTER } PROC[EDURE] имя\_процедуры [;номер]

page12image62798464

[{@имя\_параметра тип\_данных } [VARYING ] [=DEFAULT][OUTPUT] ][,...n]

page12image62798672

[WITH { RECOMPILE | ENCRYPTION | RECOMPILE,

page12image62798880

ENCRYPTION }]

page12image62799088

[FOR REPLICATION]

параметров. Ключевое слово VARYING применяется совместно с параметром OUTPUT, имеющим тип CURSOR. Оно определяет, что выходным параметром будет результирующее множество.

Ключевое слово DEFAULT представляет собой значение, которое будет принимать соответствующий параметр по умолчанию. Таким образом, при вызове процедуры можно не указывать явно значение соответствующего параметра.

Так как сервер кэширует план исполнения запроса и компилированный код, при последующем вызове процедуры будут использоваться уже готовые значения. Однако в некоторых случаях все же требуется выполнять перекомпиляцию кода процедуры. Указание ключевого слова RECOMPILE предписывает системе создавать план выполнения хранимой процедуры при каждом ее вызове.

Параметр FOR REPLICATION востребован при репликации данных и включении создаваемой хранимой процедуры в качестве статьи в публикацию. Ключевое слово ENCRYPTION предписывает серверу выполнить шифрование кода хранимой процедуры, что может обеспечить защиту от использования авторских алгоритмов, реализующих работу хранимой процедуры. Ключевое слово AS размещается в начале собственно тела хранимой процедуры. В теле процедуры могут применяться практически все команды SQL, объявляться транзакции, устанавливаться блокировки и вызываться другие хранимые процедуры. Выход из хранимой процедуры можно осуществить посредством команды RETURN.

Удаление хранимой процедуры

DROP PROCEDURE {имя\_процедуры} [,...n] Выполнение хранимой процедуры

Если вызов хранимой процедуры не является единственной командой в пакете, то присутствие команды EXECUTE обязательно. Более того, эта команда требуется для вызова процедуры из тела другой процедуры или триггера.

Использование ключевого слова OUTPUT при вызове процедуры раз- решается только для параметров, которые были объявлены при создании процедуры с ключевым словом OUTPUT.

Когда же при вызове процедуры для параметра указывается ключевое слово DEFAULT, то будет использовано значение по умолчанию. Указанное слово DEFAULT разрешается только для тех параметров, для которых определено значение по умолчанию.

page13image61954976page13image61957264page13image61955184

Для выполнения хранимой процедуры используется команда:

page13image61956848

[[ EXEC [ UTE] имя\_процедуры [;номер]

page13image61949776

[[@имя\_параметра=]{значение | @имя\_переменной}

page13image61952896

[OUTPUT ]|[DEFAULT ]][,...n]

Из синтаксиса команды EXECUTE видно, что имена параметров могут быть опущены при вызове процедуры. Однако в этом случае пользователь должен указывать значения для параметров в том же порядке, в каком они перечислялись при создании процедуры. Присвоить параметру значение по умолчанию, просто пропустив его при перечислении, нельзя. Если же требуется опустить параметры, для которых определено значение по умолча- нию, достаточно явного указания имен параметров при вызове хранимой процедуры. Более того, таким способом можно перечислять параметры и их значения в произвольном порядке.

Отметим, что при вызове процедуры указываются либо имена пара- метров со значениями, либо только значения без имени параметра. Их комбинирование не допускается.

Использование RETURN в хранимой процедуре

Позволяет выйти из процедуры в любой точке по указанному условию, а также позволяет передать результат выполнения процедуры числом, по которому можно судить о качестве и правильности выполнения процедуры

Пример создания процедуры без параметров:

GO

Пример создания процедуры c входным параметром:

CREATE PROCEDURE Count\_Books\_Pages @Count\_pages AS INT  
AS  
SELECT COUNT(Code\_book) FROM Books WHERE Pages>=@Count\_pages GO

Пример создания процедуры c входными параметрами:

AS GO

Пример создания процедуры c входными параметрами и выходным параметром:

AS

GO

Пример создания процедуры c входными параметрами и RETURN:

page14image62368112page14image62368320

CREATE PROCEDURE Count\_Books AS

page14image62368528

SELECT COUNT(Code\_book) FROM Books

page14image62369568

CREATE PROCEDURE Count\_Books\_Title @Count\_pages AS INT, @Title AS

page14image62369776

CHAR(10)

page14image62369984

SELECT COUNT(Code\_book) FROM Books WHERE Pages>=@Count\_pages

page14image62370192

AND Title\_book LIKE @Title

page14image62370400

CREATE PROCEDURE Count\_Books\_Itogo @Count\_pages INT, @Title

page14image62370608

CHAR(10) , @Itogo INT OUTPUT

page14image62370816

SELECT @Itogo = COUNT(Code\_book) FROM Books WHERE

page14image62371024

Pages>=@Count\_pages AND Title\_book LIKE @Title

CREATE PROCEDURE checkname @param INT AS

ELSE

RETURN 2  
Использование курсоров

Например, набор строк, возвращаемый инструкцией SELECT, содержит все строки, которые удовлетворяют условиям, указанным в предложении WHERE инструкции.Такой полный набор строк, возвращаемых инструкцией, называется результирующим набором.Приложения, особенно интерактивные, не всегда эффективно работают с результирующим набором как с единым целым.Им нужен механизм, позволяющий обрабатывать одну строку или небольшое их число за один раз.Курсоры являются расширением результирующих наборов, которые предоставляют такой механизм.  
Курсоры позволяют усовершенствовать обработку результатов:

* ●  позиционируясь на отдельные строки результирующего набора;
* ●  получая одну или несколько строк от текущей позиции в результирующем

наборе;

* ●  поддерживая изменение данных в строках в текущей позиции

результирующего набора;

* ●  поддерживая разные уровни видимости изменений, сделанных другими

пользователями для данных, представленных в результирующем наборе;

* ●  предоставляя инструкциям Transact-SQL в скриптах, хранимых процедурах и

триггерах доступ к данным результирующего набора

Использование параметра-курсора OUTPUT

В следующем примере используется параметр курсора OUTPUT для возврата курсора, локального относительно процедуры, в вызывающий пакет, процедуру или триггер.  
Сначала следует создать процедуру, объявляющую и открывающую курсор для таблицы Currency:

IF OBJECT\_ID ( 'dbo.uspCurrencyCursor', 'P' ) IS NOT NULL

DROP PROCEDURE dbo.uspCurrencyCursor;

GO

CREATE PROCEDURE dbo.uspCurrencyCursor

@CurrencyCursor CURSOR VARYING OUTPUT

AS

SET NOCOUNT ON;

SET @CurrencyCursor = CURSOR

FORWARD\_ONLY STATIC FOR

SELECT CurrencyCode, Name

FROM Sales.Currency;

page15image58879568

IF (SELECT Name\_author FROM authors WHERE Code\_author = @param) =

page15image58879776

'Пушкин А.С.'

page15image58879984

RETURN 1

page15image58880400

Операции в реляционной базе данных выполняются над множеством

page15image58880608

строк.

OPEN @CurrencyCursor;

GO

Затем выполним пакет, в котором объявляется локальная переменная-курсор, выполняется процедура, назначающая курсор локальной переменной, и извлекаются строки из курсора.

DECLARE @MyCursor CURSOR;

EXEC dbo.uspCurrencyCursor @CurrencyCursor = @MyCursor OUTPUT;

WHILE (@@FETCH\_STATUS = 0)

BEGIN;

FETCH NEXT FROM @MyCursor;

END;

CLOSE @MyCursor;

DEALLOCATE @MyCursor;

GO