СОЗДАНИЕ ХРАНИМЫХ ПРОЦЕДУР В MICROSOFT SQL SERVER

Хранимые процедуры

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

Системные хранимые процедуры предназначены для выполнения различных 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 действия. Хранимые процедуры – полноценный объект базы данных. Вследствие этого каждая хранимая процедура располагается в конкретной базе данных, где и выполняется.

Временные хранимые процедуры существуют лишь некоторое время, после чего автоматически уничтожаются сервером. Они делятся на локальные и глобальные. Локальные временные хранимые процедуры могут быть вызваны только из того соединения, в котором созданы. При создании 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

Создание хранимой процедуры предполагает решение следующих задач: 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:

```
{CREATE | ALTER } PROC[EDURE] имя_процедуры [;номер]
[{@имя_параметра тип_данных } [VARYING ] [=DEFAULT][OUTPUT]
][,...n]
[WITH { RECOMPILE | ENCRYPTION | RECOMPILE,
ENCRYPTION }]
[FOR REPLICATION]
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 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.

Хранимые процедуры позволяют ограничить доступ к данным в таблицах и тем самым уменьшить вероятность преднамеренных или неосознанных нежелательных действий в отношении этих данных.

И еще один важный аспект - производительность. Хранимые процедуры обычно выполняются быстрее, чем обычные SQL-инструкции. Все потому что код процедур компилируется один раз при первом ее запуске, а затем сохраняется в скомпилированной форме.

Для создания хранимой процедуры применяется команда CREATE PROCEDURE или CREATE PROC.

Таким образом, хранимая процедура имеет три ключевых особенности: упрощение кода, безопасность и производительность.

Например, пусть в базе данных есть таблица, которая хранит данные о товарах:

```
CREATE TABLE Products
(
   Id INT IDENTITY PRIMARY KEY,
   ProductName NVARCHAR(30) NOT NULL,
   Manufacturer NVARCHAR(20) NOT NULL,
   ProductCount INT DEFAULT 0,
   Price MONEY NOT NULL
);
```

Создадим хранимую процедуру для извлечения данных из этой таблицы:

```
USE productsdb;
GO
CREATE PROCEDURE ProductSummary AS
SELECT ProductName AS Product, Manufacturer, Price
FROM Products
```

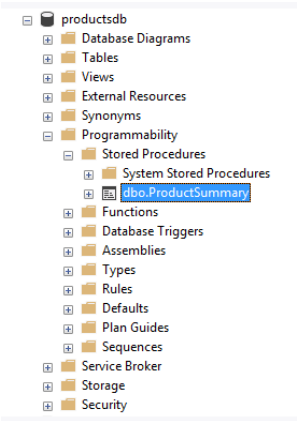
Поскольку команда CREATE PROCEDURE должна вызываться в отдельном пакете, то после команды USE, которая устанавливает текущую базу данных, используется команда GO для определения нового пакета.

После имени процедуры должно идти ключевое слово AS.

Для отделения тела процедуры от остальной части скрипта код процедуры нередко помещается в блок BEGIN...END:

```
USE productsdb;
GO
CREATE PROCEDURE ProductSummary AS
BEGIN
   SELECT ProductName AS Product, Manufacturer, Price
   FROM Products
END;
```

После добавления процедуры мы ее можем увидеть в узле базы данных в SQL Server Management Studio в подузле Programmability → Stored Procedures:



И мы сможем управлять процедурой также и через визуальный интерфейс.

Пример инструкции SQL хранимой процедуры, которая выбирает клиентов из определенного города с определенным почтовым кодом из таблицы "Customers":

```
CREATE PROCEDURE SelectAllCustomers @City nvarchar(30), @PostalCode
nvarchar(10)
AS
```

```
SELECT * FROM Customers WHERE City = @City AND PostalCode =
@PostalCode
GO;
```

Удаление хранимой процедуры

```
DROP PROCEDURE {имя_процедуры} [,...n]
```

Выполнение хранимой процедуры

Для выполнения хранимой процедуры используется команда:

```
[[ EXEC [ UTE] имя_процедуры [;номер]
[[@имя_параметра=]{значение | @имя_переменной}
[OUTPUT ]|[DEFAULT ]][,...n]
```

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 разрешается 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. Естественно, указанное слово DEFAULT разрешается только для тех параметров, для которых определено значение по умолчанию.

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 умолчанию, 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 параметров 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.

Рассмотрим другие примеры хранимых процедур

Пример создания процедуры без параметров:

```
CREATE PROCEDURE Count_Books AS
SELECT COUNT(Code_book) FROM Books
GO
```

Задание 1. Создайте данную процедуру в разделе Stored Procedures базы данных DB_Books через утилиту SQL server Management Studio. Запустите ее с помощью команды

```
EXEC Count_Books
```

Проверьте результат.

Пример создания процедуры с входным параметром:

```
CREATE PROCEDURE Count_Books_Pages @Count_pages AS INT
AS
SELECT COUNT(Code_book) FROM Books WHERE Pages>=@Count_pages
GO
```

Задание 2. Создайте данную процедуру в разделе Stored Procedures базы данных DB_Books через утилиту SQL server Management Studio. Запустите ее с помощью команды

```
EXEC Count_Books_Pages 100
```

Проверьте результат.

Пример создания процедуры с входными параметрами:

```
CREATE PROCEDURE Count_Books_Title @Count_pages AS INT, @Title AS
CHAR(10)
AS
SELECT COUNT(Code_book) FROM Books WHERE Pages>=@Count_pages
AND Title_book LIKE @Title
GO
```

Задание 3. Создайте данную процедуру в разделе Stored Procedures базы данных DB_Books через утилиту SQL server Management Studio. Запустите ее с помощью команды

```
EXEC Count_Books_Title 100, 'П%'
```

Проверьте результат.

Пример создания процедуры с входными параметрами и выходным параметром:

```
CREATE PROCEDURE Count_Books_Itogo @Count_pages INT, @Title
CHAR(10) , @Itogo INT OUTPUT
AS
SELECT @Itogo = COUNT(Code_book) FROM Books WHERE
Pages>=@Count_pages AND Title_book LIKE @Title
GO
```

Задание 4. Создайте данную процедуру в разделе Stored Procedures базы данных DB_Books через утилиту SQL server Management Studio. Запустите с помощью набора команд:

```
sql>
Declare @q As int
EXEC Count_Books_Itogo 100, 'П%', @q output
select @q
```

Проверьте результат.

Пример создания процедуры с входными параметрами и RETURN:

```
CREATE PROCEDURE checkname @param INT
AS
IF (SELECT Name_author FROM authors WHERE Code_author = @param) =
'Пушкин А.С.'
RETURN 1
ELSE
RETURN 2
```

Задание 5. Создайте данную процедуру в разделе Stored Procedures базы данных DB_Books через утилиту SQL server Management Studio. Запустите ее с помощью команд:

```
DECLARE @return_status INT
EXEC @return_status = checkname 1
SELECT 'Return Status' = @return_status
```

Пример создания процедуры без параметров для увеличения значения ключевого поля в таблице Purchases в 2 раза:

```
CREATE PROC update_proc
AS
UPDATE Purchases SET Code_purchase = Code_purchase*2
```

Процедура не возвращает никаких данных.

Задание 6. Создайте данную процедуру в разделе Stored Procedures базы данных DB_Books через утилиту SQL server Management Studio. Запустите ее с помощью команды

EXEC update_proc

Пример процедуры с входным параметром для получения всей информации о конкретном авторе:

```
CREATE PROC select_author @k CHAR(30)
AS
SELECT * FROM Authors WHERE name_author=@k
```

Задание 7. Создайте данную процедуру в разделе Stored Procedures базы данных DB_Books через утилиту SQL server Management Studio. Запустите ее с помощью команд:

```
EXEC select_author 'Пушкин А.С.' или
select_author @k='Пушкин А.С.' или
EXEC select_author @k='Пушкин А.С.'
```

Пример создания процедуры с входным параметром и значением по умолчанию для увеличения значения ключевого поля в таблице Purchases в заданное количество раза (по умолчанию в 2 раза):

```
CREATE PROC update_proc @p INT = 2
AS
UPDATE Purchases SET Code_purchase = Code_purchase *@p
```

Процедура не возвращает никаких данных.

Задание 8. Создайте данную процедуру в разделе Stored Procedures базы данных DB_Books через утилиту SQL server Management Studio. Запустите ее с помощью команд:

```
EXEC update_proc 4 или
EXEC update_proc @p = 4 или
EXEC update_proc --будет использовано значение по умолчанию.
```

Пример создания процедуры с входным и выходным параметрами. Создать процедуру для определения количества заказов, совершенных за указанный период:

```
CREATE PROC count_purchases
@d1 SMALLDATETIME, @d2 SMALLDATETIME,
```

```
@c INT OUTPUT
AS
SELECT @c=COUNT(Code_purchase) FROM Purchases WHERE Date_order
BETWEEN @d1 AND @d2
SET @c = ISNULL(@c,0)
```

Задание 9. Создайте данную процедуру в разделе Stored Procedures базы данных DB_Books через утилиту SQL server Management Studio. Запустите ее с помощью команд:

```
DECLARE @c2 INT
EXEC count_purchases ’01-jun-2006’, ’01-jul-2006’, @c2 OUTPUT
SELECT @c2
```

Список источников

https://metanit.com/sql/sqlserver/11.1.php
http://wiki.vspu.ru/workroom/pib22-2013/comp/index/lab4
https://schoolsw3.com/sql/sql_stored_procedures.php