РЕАЛИЗАЦИЯ ХРАНИМЫХ ПРОЦЕДУР И ФУНКЦИЙ

Виды хранимых процедур

Хранимая процедура — это группа инструкций Transact-SQL, собранных в одном плане выполнения. С помощью хранимых процедур разработчики достигают согласованной реализации алгоритма в нескольких приложениях.

Назначение хранимых процедур:

- позволяет уменьшить размер запроса, посылаемого по сети от клиента на сервер;
- повышает общую производительность системы;
- упрощает сопровождение программных комплексов и внесение изменений в исходный текст модулей.

Хранимые процедуры являются самостоятельными объектами базы данных, к которым можно разрешать или запрещать доступ командами GRANT и DENY. Например, выполнение следующей команды запретит выполнение команд хранимой процедуры hello для пользователя mng:

DENY exec ON hello TO mng.

Хранимые процедуры схожи с процедурами других языков программирования и позволяют:

- включать различные операторы и вызывать другие хранимые процедуры;
- принимать входные параметры и возвращать значения в виде выходных параметров.

MS SQL Server поддерживает следующие виды хранимых процедур:

- системные процедуры – хранятся в системной базе данных master, их имена начинаются с символов sp_. Используются для решения специализированных системных задач: администрирования, безопасности и др.;
- пользовательские процедуры – создаются, хранятся и выполняются пользователями в контексте только той базы данных, для которой были созданы;
- временные процедуры – доступны только в активном соединении, после закрытия соединения удаляются автоматически. Имена таких процедур должны начинаться с символа #.

Для работы с хранимыми процедурами предназначены системные хранимые процедуры:

- sp_helptext ИмяПроцедуры – выводит код указанной хранимой процедуры;

- sp_help ИмяПроцедуры – выводит список параметров и их типов данных для указанной процедуры;
- sp_stored_procedures – возвращает список сохраненных процедур текущей базы данных.

Использование хранимых процедур

Синтаксис создания хранимых процедур приведен на рисунке 1.

```
CREATE { PROC | PROCEDURE } [schema_name.] procedure_name [ ;
number ]
        [{ @parameter [ type_schema_name. ] data_type }
         [ VARYING ] [ = default ] [ OUT | OUTPUT ] [READONLY]
        ] [ ,...n ]
[ WITH <procedure_option> [ ,...n ] ]
[ FOR REPLICATION ]
AS { <sql_statement> [;][ ...n ] | <method_specifier> }
[;]
<procedure_option> ::=
        [ ENCRYPTION ]
        [ RECOMPILE ]
        [ EXECUTE_AS_Clause ]

<sql_statement> ::=
{ [ BEGIN ] statements [ END ] }
<method_specifier> ::=
EXTERNAL NAME assembly_name.class_name.method_name
```

Рисунок 1 - Синтаксис создания хранимых процедур

Как и большинство объектов MS SQL Server хранимая процедура может быть создана с помощью средств Transact-SQL или с применением графического интерфейса *Management Studio*.

При создании хранимой процедуры необходимо учитывать следующие особенности:

- процедура может содержать неограниченно количество операторов, кроме операторов создания следующих объектов: процедуры, представления, правила, умолчания;
- создание процедуры может выполнить пользователь роли *sysadmin*, *db_owner* или *db_ddladmin*, а также имеющий право на выполнение команды CREATE PROC;
- количество параметров не должно превышать 2100.

Для создания хранимых процедур используется инструкция CREATE PROCEDURE. Хранимые процедуры могут создаваться только в текущей базе данных, кроме временных хранимых процедур, которые всегда создаются в базе данных **tempdb**.

Чтобы изменить существующую хранимую процедуру и сохранить назначение разрешений, используйте инструкцию ALTER PROCEDURE**.**

Для удаления пользовательских хранимых процедур из текущей базы данных используется инструкция DROP PROCEDURE**.**

Хранимые процедуры могут быть локальными и глобальными:

Локальные – должны иметь имя и начинаться с символа #. Могут быть вызваны только из того соединения, в котором они были созданы. Автоматически удаляются при отключении пользователя, перезапуске или остановке сервера.

Глобальные – должны иметь имя и начинаться с символа ##. Доступны для любых соединений с экземпляром сервера, на котором они были созданы. Они удаляются либо при закрытии соединения, где были созданы, либо автоматически при запуске или остановке сервера.

Функции и хранимые процедуры могут быть вызваны:

- клиентскими программами,
- другими функциями или хранимыми процедурами,
- триггерами.

Хранимые процедуры могут вызываться только командой EXECUTE, или сокращенно EXEC. За этой командой должны быть указаны имя процедуры и через пробел список аргументов, если вызывается процедура с параметрами. Список аргументов разделяется запятой.

Пример: Создание хранимой процедуры, который выводит имя студентов, у которых средний балл больше заданной величины:

```
CREATE PROCEDURE СрБАЛЛ
@X Real
AS
SELECT *
FROM Студенты
WHERE
(Оценка1+ Оценка2+ Оценка3)/3>@X
```

Команда вызова приведенной выше процедуры выглядит следующим образом:

```
EXEC СрБАЛЛ 4
```

Команда выводит всех студентов, у которых средний балл больше 4.

Можно привести следующие рекомендации по работе с хранимыми процедурами:

- Разрабатывайте каждую хранимую процедуру для выполнения одной задачи.
- Создайте, протестируйте и устраните неполадки процедуры на сервере, затем протестируйте ее со стороны клиента.
- Избегайте использования префикса **sp_** в именах локальных хранимых процедур, чтобы можно было легко отличать системные хранимые процедуры.

- Сведите к минимуму использование временных хранимых процедур

Хранимые процедуры могут принимать информацию с помощью задания входных параметров. Эти параметры определяются при создании хранимой процедуры с помощью выражения CREATE PROCEDURE.

При использовании входных параметров старайтесь придерживаться следующих рекомендаций:

- По возможности определяйте для входных параметров значения по умолчанию. Для этого в определении хранимой процедуры используйте ключевое слово DEFAULT.
- Проверяйте значения входных параметров в начале хранимой процедуры, чтобы определить отсутствующие или недопустимые значения.

В SQL Server 2008 появилась новая возможность — передавать в качестве параметров хранимых процедур табличные значения. Табличные параметры определяются на основе табличных пользовательских типов данных и могут быть использованы для передачи набора строк в хранимую процедуру без создания временных таблиц.

Работа с функциями

Функции — это подпрограммы, состоящие из одной или нескольких инструкций Transact-SQL, которые могут использоваться для сохранения кода для повторного использования. Пользовательская функция не может выполнять действия, изменяющие состояние базы данных.

SQL Server поддерживает 4 типа функций:

1. **Встроенные функции** — большой набор предопределенных функций, которые не могут быть изменены.
2. **Скалярные функции** — возвращают единственное значение с типом, определенным в операторе RETURNS.

```
CREATE FUNCTION [ schema_name. ] function_name
( [ { @parameter_name [ AS ][ type_schema_name. ] parameter_data_type
   [ = default ] [ READONLY ] }
   [ ,...n ]
 ]
)
RETURNS return_data_type
   [ WITH <function_option> [ ,...n ] ]
   [ AS ]
   BEGIN
            function_body
      RETURN scalar_expression
   END
[ ; ]
```

3. **Табличные функции с одним запросом** — возвращают таблицу, которая является результатом одной инструкции SELECT.

```
CREATE FUNCTION [ schema_name. ] function_name
( [ { @parameter_name [ AS ] [ type_schema_name. ] parameter_data_type
   [ = default ] [ READONLY ] }
   [ ,...n ]
 ]
)
RETURNS TABLE
   [ WITH <function_option> [ ,...n ] ]
   [ AS ]
   RETURN [ ( ] select_stmt [ ) ]
[ ; ]
```

4. **Многооператорные табличные функции** — возвращают таблицу, созданную одной или несколькими инструкциями Transact-SQL и сходны с хранимыми процедурами. В отличие от хранимой процедуры, на многооператорную табличную функцию можно ссылаться в предложении FROM инструкции SELECT.

```
CREATE FUNCTION [ schema_name. ] function_name
( [ { @parameter_name [ AS ] [ type_schema_name. ] parameter_data_type
   [ = default ] [READONLY] }
   [ ,...n ]
 ]
)
RETURNS @return_variable TABLE <table_type_definition>
   [ WITH <function_option> [ ,...n ] ]
   [ AS ]
   BEGIN
           function_body
     RETURN
   END
[ ; ]
```

Изменение функции осуществляется командой Transact-SQL ALTER FUNCTION, удаление — DROP FUNCTION.

Функции в SQL Server можно разделить на 2 категории:

- **Детерминированные функции** возвращают один и тот же результат, если предоставлять им один и тот же набор входных значений и использовать одно и то же состояние базы данных.
- **Недетерминированные функции** могут возвращать каждый раз разные результаты, если предоставлять им один и тот же набор входных значений и даже если использовать одно и то же состояние базы данных.

Пример: Создать функцию для выполнения четырех арифметических операций “+”, “- ”, “*” и “/” над целыми операндами типа bigint, выполнив кодирование и проверку:

1. Кодирование

```
CREATE FUNCTION Calculator
    (@ Opd1 bigint,
    @ Opd2 bigint,
    @ Oprt char(1) = “*”)
RETURNS bigint
    AS
        BEGINT
            DECLARE @ Result bigint
            SET @ Result =
                CASE @ Oprt
                    WHEN “+” THEN @ Opd1 + @ Opd2
                    WHEN “-” THEN @ Opd1 - @ Opd2
                    WHEN “*” THEN @ Opd1 * @ Opd2
                    WHEN “/” THEN @ Opd1 / @ Opd2
                    ELSE 0
    END
    Return @ Result
    END
```

2. Тестирование

```
    SELECT dbo.Calculator(4,5, ‘+’),
            dbo. Calculator(3,7, ‘*’) – dbo.Calculator(64,4,‘/’)*2.
    9           -11
    (1 row (s) affected)
```

Основные рекомендации по созданию функций:

- Задавайте полное имя для пользовательской функции — имя_схемы.имя_функции.
- Избегайте ошибок TransactSQL, которые отменяют выполнение оператора и осуществляют переход на следующий оператор в этом модуле. Это приводит к прекращению выполнения функции.