МГТУ ИМ. Н.Э. БАУМАНА

Базы Данных

Семинар 6

Описание семинара: Функции и процедуры

Функции T-SQL

SQL Server содержит набор встроенных функций и предоставляет возможность создавать пользовательские функции (User Defined Function – UDF). Различают детерминированные и недетерминированные функции.

- Функция является **детерминированной**, если при одном и том же заданном входном значении она всегда возвращает один и тот же результат. Например, встроенная функция DATEADD является детерминированной; она возвращает новое значение даты, добавляя интервал к указанной части заданной даты.
- Функция является **недетерминированной**, если она может возвращать различные значения при одном и том же заданном входном значении. Например, встроенная функция GETDATE является недетерминированной; при каждом вызове она возвращает различные значения даты и времени компьютера, на котором запущен экземпляр SQL Server.

Все функции конфигурации, курсора, метаданных, безопасности и системные статистические – недетерминированные. Список этих функций приводится в справочнике. Функции, вызывающие недетерминированные функции и расширенные хранимые процедуры, также считаются недетерминированными.

Пользовательские функции в зависимости от типа данных возвращаемых ими значений могут быть скалярными и табличными. Табличные пользовательские функции бывают двух типов: подставляемые и многооператорные.

Скалярные пользовательские функции

Скалярные пользовательские функции обычно используются в списке столбцов инструкции SELECT и в предложении WHERE. Табличные пользовательские функции обычно используются в предложении FROM, и их можно соединять с другими таблицами и представлениями.

```
CREATE FUNCTION [ имя-схемы. ] имя-функции ( [ список-объявлений-параметров ] )
RETURNS скалярный-тип-данных
[ WITH список-опций-функций ]
[ AS ]
BEGIN
    тело-функции
    RETURN скалярное-выражение
END [ ; ]
```

Для создания скалярной функции используется инструкция CREATE FUNCTION, имеющая следующий синтаксис:

```
@имя-параметра [ AS ] [ имя-схемы. ] тип-данных [ = значение-по-умолчанию ] [ READONLY ]
```

- Тип данных параметра – любой тип данных, включая определяемые пользователем типы данных CLR и определяемые пользователем табличные типы, за исключением скалярного типа timestamp и нескалярных типов cursor и table.
- Значение по умолчанию
- Необязательное ключевое слово READONLY указывает, что параметр не может быть обновлен или изменен при определении функции. Если тип параметра является определяемым пользователем табличным типом, то должно быть указано ключевое слово READONLY.

где

- Список объявлений параметров является необязательным, но наличие скобок обязательно.
- Объявление параметра в списке объявлений параметров имеет вид:

- Возвращаемое значение может быть любого скалярного типа данных, включая определяемые пользователем типы данных CLR, за исключением типа данных timestamp.
- В качестве опций функции используются:
  - ENCRIPTION — SQL Server шифрует определение функции.
  - SCHEMABINDING - привязывает функцию к схеме базовой таблицы или таблиц. Если аргумент SCHEMABINDING указан, нельзя изменить базовую таблицу или таблицы таким способом, который может повлиять на определение функции.
    RETURNS NULL ON NULL INPUT или| CALLED ON NULL INPUT - ...

    - Предложение EXECUTE AS - указывает контекст безопасности, в котором может быть выполнена функция. Например, EXECUTE AS CALLER указывает, что функция будет выполнена в контексте пользователя, который ее вызывает. Также могут быть указаны параметры SELF, OWNER и имя-пользователя.
- Операторы BEGIN и END, которыми ограничивается тело функции, являются обязательными.
- Тело функции представляет собой ряд инструкций T-SQL, которые в совокупности вычисляют скалярное выражение.

Синтаксис вызова скалярных функций схож с синтаксисом, используемым для встроенных функций T-SQL:

имя-схемы.имя-функции ([список-параметров])
где
- Имя схемы (имя владельца) для скалярной функции является обязательным.
- Нельзя использовать синтаксис с именованными параметрами (@имя-параметра = значение).
- Нельзя не указывать (опускать) параметры, но можно применять ключевое слово DEFAULT для указания значения по умолчанию.

Для скалярной функции можно использовать инструкцию EXECUTE:

```
EXECUTE @возвращаемое-значение = имя-функции ( [ список-параметров ] )
```
где
- Не нужно указывать имя схемы (имя владельца).
- Можно использовать именованные параметры (@имя-параметра = значение).
- Если используются именованные параметры, они не обязательно должны следовать в том порядке, в котором указаны в объявлении функции, но необходимо указать все параметры; нельзя опускать ссылку на параметр для использования значения по умолчанию.

Примеры создания и вызова скалярных функций

```
CREATE FUNCTION dbo.AveragePrice() RETURNS smallmoney
WITH SCHEMABINDING AS
BEGIN
    RETURN (SELECT AVG(Price) FROM dbo.R)
END;

CREATE FUNCTION dbo.PriceDifference(@Price smallmoney) RETURNS smallmoney AS
BEGIN
    RETURN @Price - dbo.AveragePrice()
END;

— Вызов функции
SELECT Pname, Price, dbo.AveragePrice() AS Average, dbo.PriceDifference(Price) AS Difference FROM R
WHERE City='Смоленск'
```

Подставляемая табличная функция
Тело подставляемой табличной функции фактически состоит из единственной инструкции SELECT. Синтаксис создания подставляемой табличной функции выглядит так:

```
CREATE FUNCTION [ имя-схемы. ] имя-функции ( [ список-объявлений-параметров ] )
RETURNS TABLE
[ WITH список-опций-функций ]
[ AS ]
RETURN [ ( ] выражение-выборки [ ) ] END [ ; ]
```

Пример создания и вызова подставляемой табличной функции

```
CREATE FUNCTION dbo.FullSPJ()
RETURNS TABLE
AS
      RETURN (SELECT S.Sname, P.Pname, J.Jname, SPJ.Qty
                  FROM S INNER JOIN SPJ ON S.Sno=SPJ.Sno
                              INNER JOIN P ON P.Pno=SPJ.Pno
                              INNER JOIN J ON J.Jno=SPJ.Jno)

— Вызов функции
SELECT *
FROM dbo.FullSPJ()
```

Многооператорная табличная функция
Подобно скалярным функциям, в многооператорной табличной функции команды T-SQL располагаются внутри блока BEGIN-END. Поскольку блок может содержать несколько инструкций SELECT, в предложении RETURNS необходимо явно определить таблицу, которая будет возвращаться. Поскольку оператор RETURN в многооператорной табличной функции всегда возвращает таблицу, заданную в предложении RETURNS, он должен выполняться без аргументов. Синтаксис создания многооператорной табличной функции выглядит так:

```
CREATE FUNCTION [ имя-схемы. ] имя-функции ( [ список-объявлений-параметров ] )
RETURNS @имя-возвращаемой-переменной TABLE определение-таблицы
[ WITH список-опций-функций ]
[ AS ]
BEGIN
RETURN END [ ; ]
```

Пример создания и вызова многооператорной табличной функции

```
CREATE FUNCTION dbo.fnGetReports ( @EmployeeID AS int )
RETURNS @Reports TABLE ( EmployeeID int NOT NULL, ReportsToID int NULL )
AS BEGIN
      DECLARE @Employee AS int
      INSERT INTO @Reports
        SELECT EmployeeID, ReportsTo FROM Employees
        WHERE EmployeeID = @EmployeeID
SELECT @Employee = MIN(EmployeeID) FROM Employees
WHERE ReportsTo = @EmployeeID
WHILE @Employee IS NOT NULL
  BEGIN
    INSERT INTO @Reports
       SELECT *
END RETURN
END
```

Хранимые процедуры T-SQL

Хранимая процедура – именованный объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере. Хранимые процедуры похожи на обыкновенные процедуры языков высокого уровня, у них могут быть входные и выходные параметры и локальные переменные. В хранимых процедурах могут выполняться операторы DDL, DML, TCL, FCL. Процедуры можно создавать для постоянного использования, для временного использования в одном сеансе (локальная временная процедура), для временного использования во всех сеансах (глобальная временная

процедура). Хранимые процедуры могут выполняться автоматически при запуске экземпляра SQL Server.

Создание хранимых процедур
Для создания хранимой процедуры используется инструкция CREATE PROCEDURE, имеющая следующий синтаксис:

```
CREATE PROCEDURE [ имя-схемы. ] имя-процедуры [ список-объявлений-параметров ]
[ WITH список-опций-процедуры ]
[ FOR REPLICATION ]
AS
тело-процедуры
[ ;]
```

где
- Если имя схемы не указано при создании процедуры, то автоматически назначается схема по умолчанию для пользователя, который создает процедуру.
- Список объявлений параметров является необязательным.
- Объявление параметра в списке объявлений параметров имеет вид:

```
@имя-параметра тип-данных [ VARYING ] [ = значение-по-умолчанию ] [ OUT | OUTPUT ] [READONLY]
```

- Параметрами процедуры могут быть любые типы данных, за исключением table.
- Для создания параметров, возвращающих табличное значение, можно использовать определяемый пользователем табличный тип. Возвращающие табличное значение параметры могут быть только входными и должны сопровождаться ключевым словом READONLY.
- Тип данных cursor может быть использован только в качестве выходного параметра.
- VARYING применяется только к аргументам типа cursor.
- OUT или OUTPUT показывает, что параметр процедуры является выходным. Параметры типов text, ntext и image не могут быть выходными.
- READONLY указывает, что параметр не может быть обновлен или изменен в тексте процедуры.
- Опциями функции могут быть:
  - ENCRYPTION - SQL Server шифрует определение процедуры.
  - RECOMPILE - SQL Server перекомпилирует процедуру при каждом ее выполнении.
  - Предложение EXECUTE AS - определяет контекст безопасности, в котором должна быть выполнена процедура.
- Тело процедуры - одна или несколько инструкций T-SQL. Инструкции можно заключить в необязательные ключевые слова BEGIN и END.
- FOR REPLICATION указывает, что процедура создается для репликации.
- Тело процедуры может содержать оператор RETURN, возвращающий целочисленное значение вызывающей процедуре или приложению.

Примечания.
- Локальную временную процедуру можно создать, указав один символ номера (#) перед именем процедуры.
- Глобальную временную процедуру можно создать, указав два символа номера (##) перед именем процедуры.
- Временные процедуры создаются в базе данных tempdb.
- Рекомендуется начинать текст процедуры с инструкции SET NOCOUNT ON (она должна следовать сразу за ключевым словом AS). В этом случае отключаются сообщения, отправляемые SQL Server клиенту после выполнения любых инструкций SELECT, INSERT, UPDATE, MERGE и DELETE.

Выполнение хранимых процедур
При выполнении процедуры в первый раз она компилируется, при этом определяется оптимальный план получения данных. При последующих вызовах процедуры может быть повторно использован уже созданный план, если он еще находится в кэше планов компонента Database Engine.

Одна процедура может вызывать другую. Уровень вложенности увеличивается на 1, когда начинается выполнение вызванной процедуры, и уменьшается на 1, когда вызванная процедура завершается. Уровень вложенности процедур может достигать 32. Текущий уровень вложенности процедур можно получить при помощи функции @@NESTLEVEL.
Чтобы выполнить процедуру, надо использовать инструкцию EXECUTE. Также можно выполнить процедуру без использования ключевого слова EXECUTE, если процедура является первой инструкцией в пакете.
При выполнении процедуры (в пакете или внутри хранимой процедуры или функции) настоятельно рекомендуется уточнять имя хранимой процедуры указанием, по крайней мере, имени схемы.

Пример процедуры с входными и выходными параметрами

```
CREATE PROCEDURE dbo.Factorial @ValIn bigint, @ValOut bigint output
AS
BEGIN
      IF @ValIn > 20 BEGIN
            PRINT N'Входной параметр должен быть <= 20'
            RETURN -99
      END
      DECLARE @WorkValIn bigint, @WorkValOut bigint
      IF @ValIn != 1
      BEGIN
            SET @WorkValIn = @ValIn - 1
            PRINT @@NESTLEVEL
            EXEC dbo.Factorial @WorkValIn, @WorkValOut OUTPUT
            SET @ValOut = @WorkValOut * @ValIn
       END
       ELSE
             SET @ValOut = 1
 END

— Вызов процедуры
DECLARE @FactIn int, @FactOut int
SET @FactIn = 8
EXEC dbo.Factorial @FactIn, @FactOut OUTPUT

PRINT N'Факториал ' + CONVERT(varchar(3),@FactIn) +
          N' равен ' + CONVERT(varchar(20),@FactOut)
```

Изменение хранимых процедур

Если нужно изменить инструкции или параметры хранимой процедуры, можно или удалить (DROP PROCEDURE) и создать ее заново (CREATE PROCEDURE), или изменить ее за один шаг (ALTER PROCEDURE). При удалении и повторном создании хранимой процедуры все разрешения, связанные с ней, будут утеряны при восстановлении. При изменении хранимой процедуры ее определение или определение ее параметров меняются, но разрешения, связанные с ней, остаются и все зависящие от нее процедуры или триггеры не затрагиваются.