10. Функции

Бывают

- скалярные – возвращают 1 значение, кроме image, text, cursor, timestamp, имеет тело (BEGIN, END)
- табличные – возвращают таблицу, не имеют тела
- многооператорные – возвращают таблицу посредством 1/n операторов, похожа на процедуру, но в отличие от процедур, на них можно ссылаться в WHERE.

Создание

- скалярная ф.

Для создания нужно определить имя, параметры (необязательно), возвращаемый тип, после идёт тело функции, где можно использовать операторы

```
CREATE FUNCTION [owner.]name
([@param_name [AS ] data_type [=default], …)
RETURNS data_type
[ AS ]
BEGIN
   function_body
   RETURN scalar_expression
END
```

- табличная

Для создания нужно определить имя, параметры (необязательно), возвращаемый тип, вместо тела функции – оператор выбора

```
CREATE FUNCTION [ owner_name. ] function_name
   ( [ @parameter_name [AS] scalar_parameter_data_type [ = default ]… )
RETURNS TABLE
[ WITH < function_option > [ [,] ...n ] ]
[ AS ]
RETURN [(] select-stmt [)]
```

- многооператорная

Для создания нужно определить имя, параметры (необязательно), возвращаемый тип, после идёт тело функции и доп.опции

```
CREATE FUNCTION [ owner_name. ] function_name
   ( [ @parameter_name [AS] scalar_parameter_data_type [ = default ]… )
RETURNS @return_variable TABLE < table_type_definition >
[ WITH < function_option > [ [,] ...n ] ]
[ AS ]
BEGIN
   function_body
   RETURN
END
< function_option > ::=
   { ENCRYPTION | SCHEMABINDING }
< table_type_definition > :: =
   ( { column_definition | table_constraint } [ ,...n ] )
```

Примеры:

1. Создадим скалярную функцию, вычисляющую стоимость товара (цена*количество) в определённый день из меющейся таблицы Товары

```
2.   CREATE FUNCTION GetSumm
3.    (@name varchar(50), @date datetime)
4.   RETURNS numeric(10,2)
5.    BEGIN
6.     DECLARE @Summ numeric(10,2)
7.     SELECT @Summ = Цена*Количество
8.     FROM Товары
9.     WHERE [Название товара]=@name
10.      AND Дата=@date;
11.    RETURN @Summ
12.   END
```

Для того, чтобы посчитать значение, была создана переменная, для сохранения этого значения и использовался оператор select, отделяемый в конце ;.
Возвращаемое значение должно быть скалярным.

Использование

```
SELECT dbo.GetSumm('Хлеб', '01.01.2005')
```

2. Табличная функция

Вернём таблицу с товарами и их общей стоимостью.

```
CREATE FUNCTION GetPrice()
RETURNS TABLE
AS
 RETURN
 (
  SELECT Дата, [Название товара], Цена,
    Количество, Цена*Количество AS Сумма
  FROM Товары
 )
```

Здесь параметров нет, но скобки нужны. После слова RETURNS должно быть TABLE. У столбцов таблицы должны быть названия, если их нет, нужно использовать AS.

3. Многооператорная функция

```
CREATE FUNCTION getFIO ()
RETURNS  @ret TABLE
  (idPeoples int primary key,
   vcFIO varchar(100))
AS
 BEGIN
  INSERT @ret
  SELECT idPeoples, vcFamil+' '+vcName+' '+vcSurName
  FROM tbPeoples;

  RETURN
 END
```

Здесь возвращается таблица с именем @ret и указанными полями. В теле ф-ции в таблицу заносятся данные не только из таблицы, но и изменённые.

Использование

```
SELECT *
FROM GetFIO()
```