Создание и применение пользовательских функций

Задание:

1. Изучить материал.
2. Сделать примеры.
3. Сделать отчет.

Пользовательские функции очень похожи на хранимые процедуры. Так же в них можно передавать параметры и они выполняют некоторые действия, однако их главным отличием от хранимых процедур является то, что они выводят (возвращают) какой то результат.

Все пользовательские функции делятся на 2 вида:

1) скалярные функции - функции, которые возвращают число или текст, то есть одно или несколько значений;

2) табличные функции - функции, которые выводят результат в виде таблицы.

Теперь рассмотрим создание и применение пользовательских функций. В БД Microsoft SQL Server все пользовательские функции находятся в папке **"Functions"** расположенной в папке **"Programmability"** в обозревателе объектов (рис. 1).

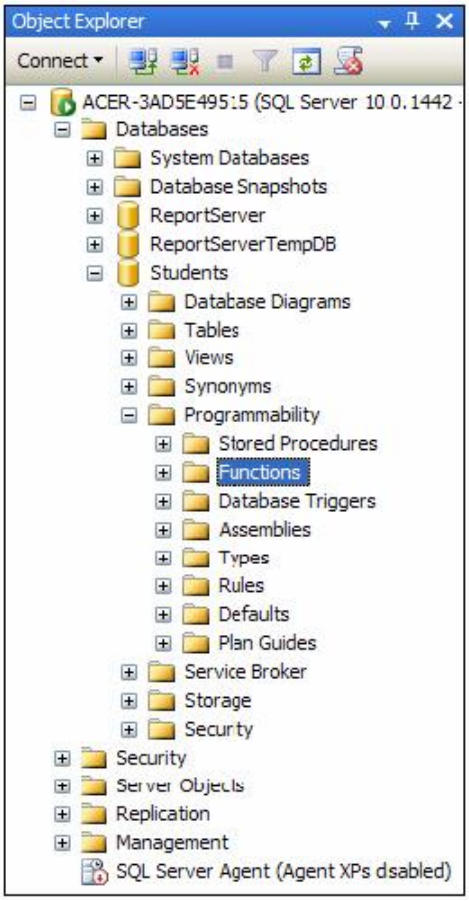


Рисунок 1

Начнем с создания скалярных пользовательских функций. Для создания новой скалярной пользовательской функции в обозревателе объектов, в БД **"Students"**, в папке **"Programmability"**, щелкните **ПКМ** по папке **"Functions"** и в появившемся меню выберите пункт **"New/Scalar-valued Function"**. Появится окно новой скалярной пользовательской функции (рис. 2)

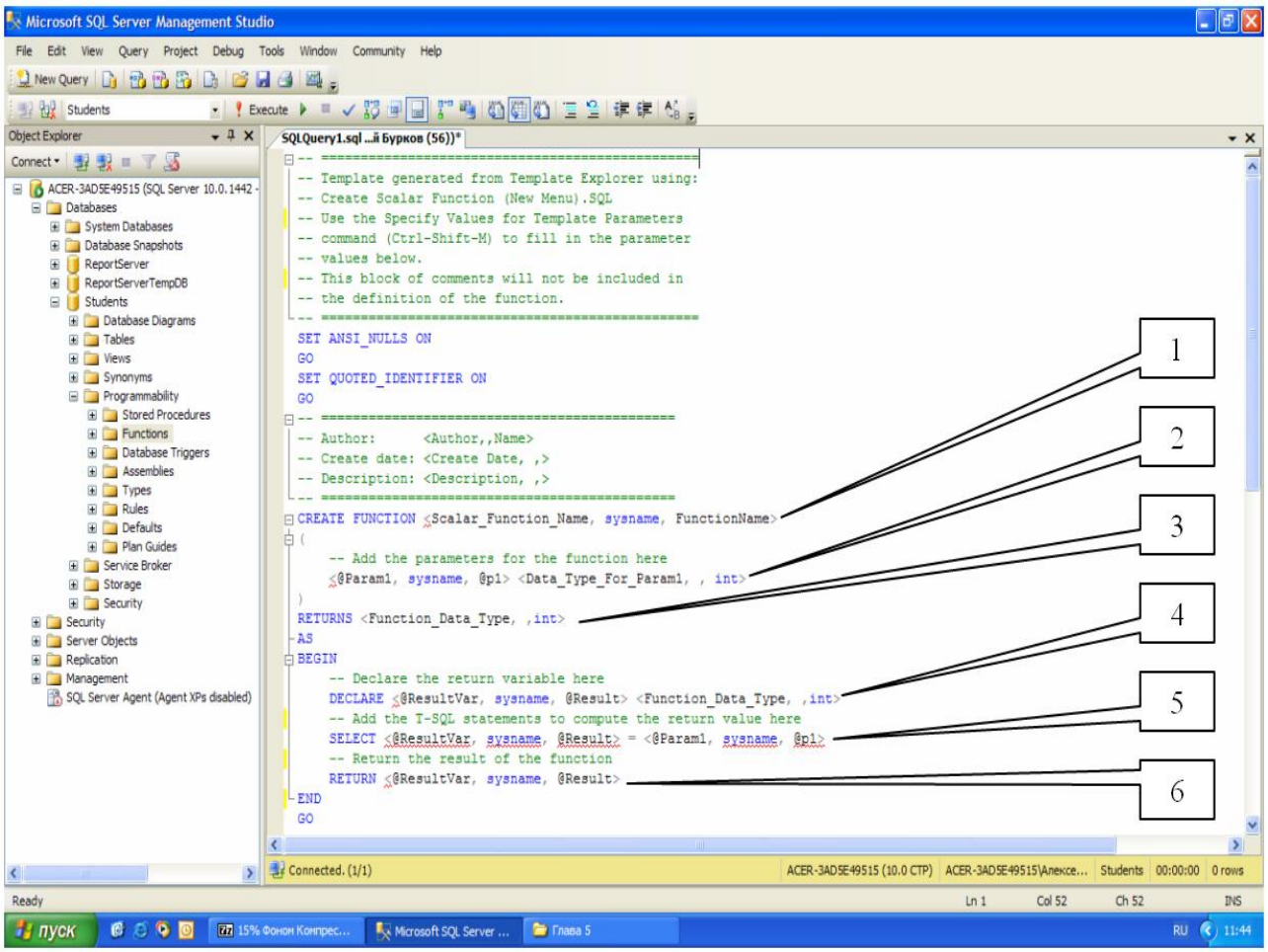


Рисунок 2

Синтаксис скалярной пользовательской функции похож на синтаксис хранимой процедуры. Однако имеется ряд существенных отличий (рис. 2):

1. Область определения имени функции (Inline_Function_Name);
2. Параметры, передаваемые в процедуру **(@Param1)**. Определение параметров аналогично определению параметров в хранимой процедуре;
3. Тип данных значения возвращаемого процедурой;
4. Область объявления переменных, используемых внутри функции. Объявление переменных имеет следующий синтаксис:
5. DECLARE @<Имя переменной> <Тип данных>
6. Тело самой пользовательской функции, содержит команды языка программирования запросов T-SQL;
7. Команда RETURN возвращающая результат выполнения функции. Имеет следующий синтаксис:
8. RETURN @<Имя переменной с результатом>

Переменная должна быть того же типа данных, который был указан в пункте 3.

Создадим скалярную пользовательскую функцию, вычисляющую среднее трех величин. В окне новой пользовательской функции наберите код представленный на рис. 3.

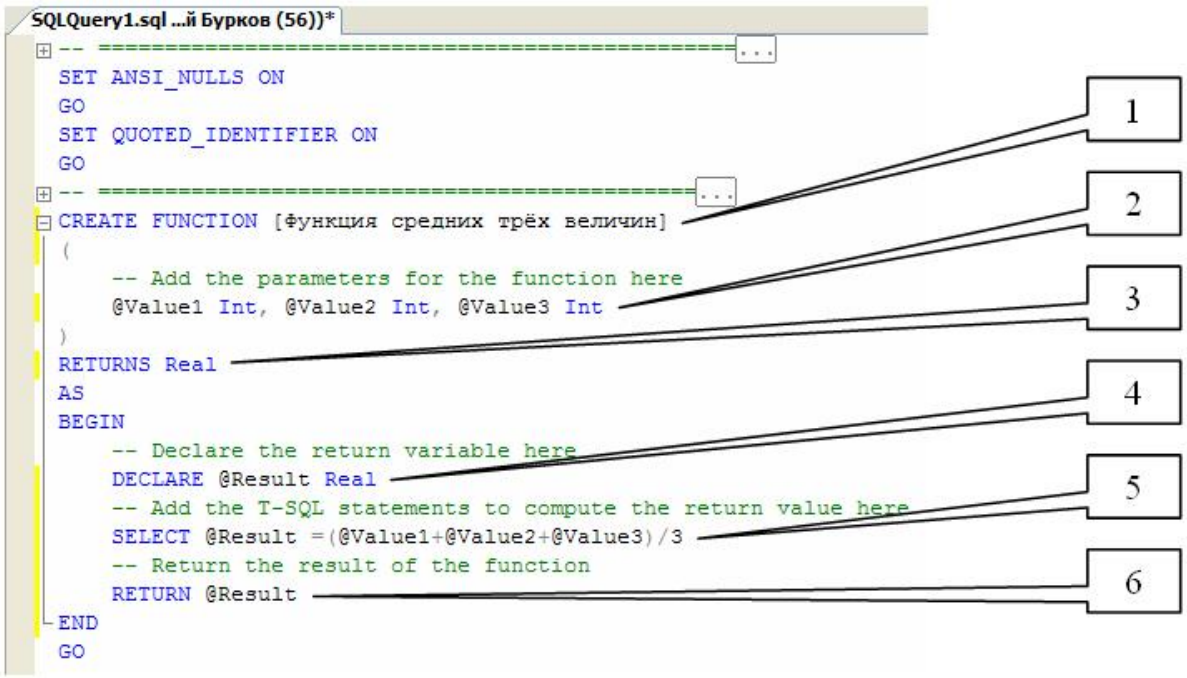
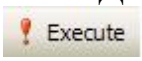


Рисунок 3

Рассмотрим более подробно код данной скалярной пользовательской функции (рис. 3):

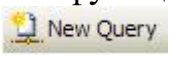
1. CREATE FUNCTION [Функция средних трех величин] - определяет имя создаваемой функции как "Функция средних трех величин";
2. @Value1 Real, @Value2, @Value3 - определяют три параметра процедуры **Value1**, **Value2** и **Value3**. Данным параметрам можно присвоить целые числа (Тип данных Int);
3. RETURNS Real - показывает, что функция возвращает дробные числа (Тип данных Real);
4. DECLARE @Result Real - объявляется переменная @Result для хранения результата работы функции, то есть дробного числа (Тип данных Real);
5. SELECT @Result=(@Value1+@Value2+@Value3)/3 - вычисляет среднее и помещает результат в переменную @Result;
6. RETURN @Result - возвращает значение переменной @Result.

Остальные фрагменты кода рассмотрены выше.

Для создания функции, выполним вышеописанный код, нажав кнопку 

. (Выполнить) на панели инструментов. В нижней части окна с кодом появиться сообщение "Command(s) completed successfully.". Закройте

окно с кодом, щелкнув мышью по кнопке закрытия  расположенной в верхнем правом углу окна с кодом функции.

Проверим работу созданной скалярной пользовательской функции. Для запуска пользовательской функции необходимо создать новый пустой запрос, нажав на кнопку 

. (Новый запрос) на панели инструментов. В появившемся окне с пустым запросом наберите команду SELECT dbo.[Функция средних трех величин] (3, 5, 4) и нажмите кнопку 

на панели инструментов (рис. 4).

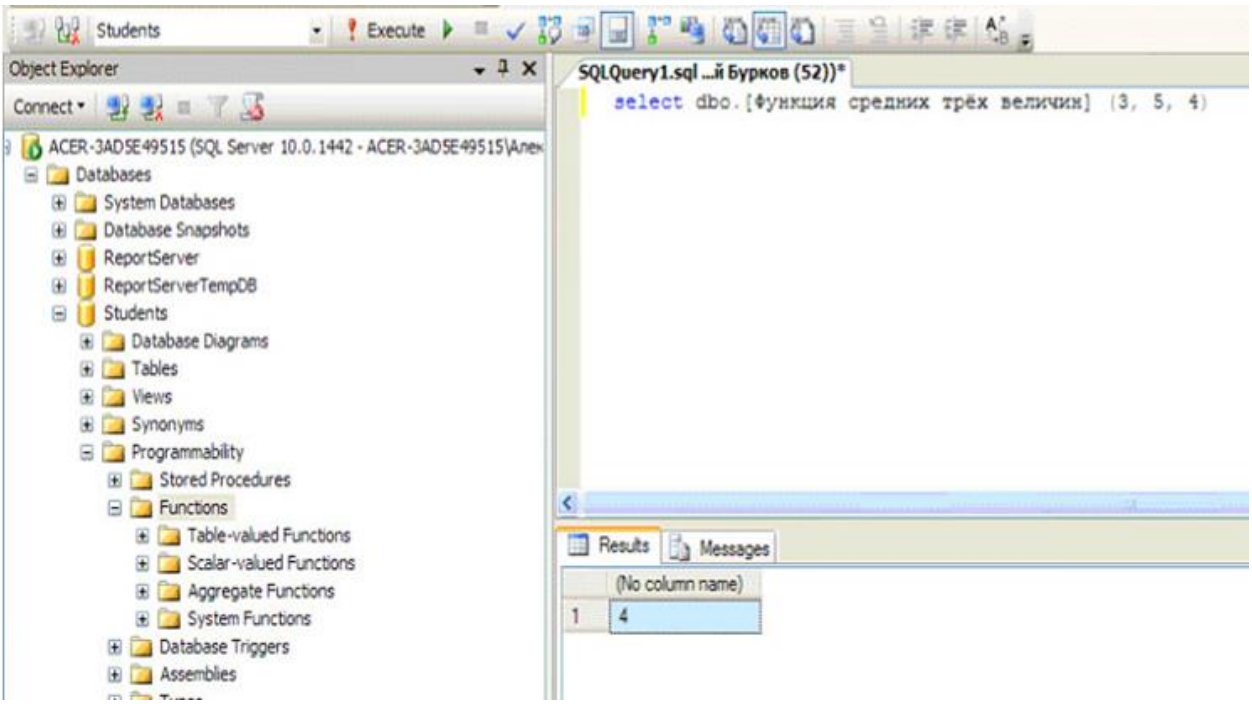


Рисунок 4

В нижней части окна с кодом появиться результат выполнения новой скалярной пользовательской функции: **4** (рис. 4).

Теперь создадим более сложную скалярную пользовательскую функцию, предназначенную для определения последнего дня месяца введенной даты.

Создайте новую скалярную пользовательскую функцию, так как об этом сказано выше. В окне новой пользовательской функции наберите следующий код (рис. 5):

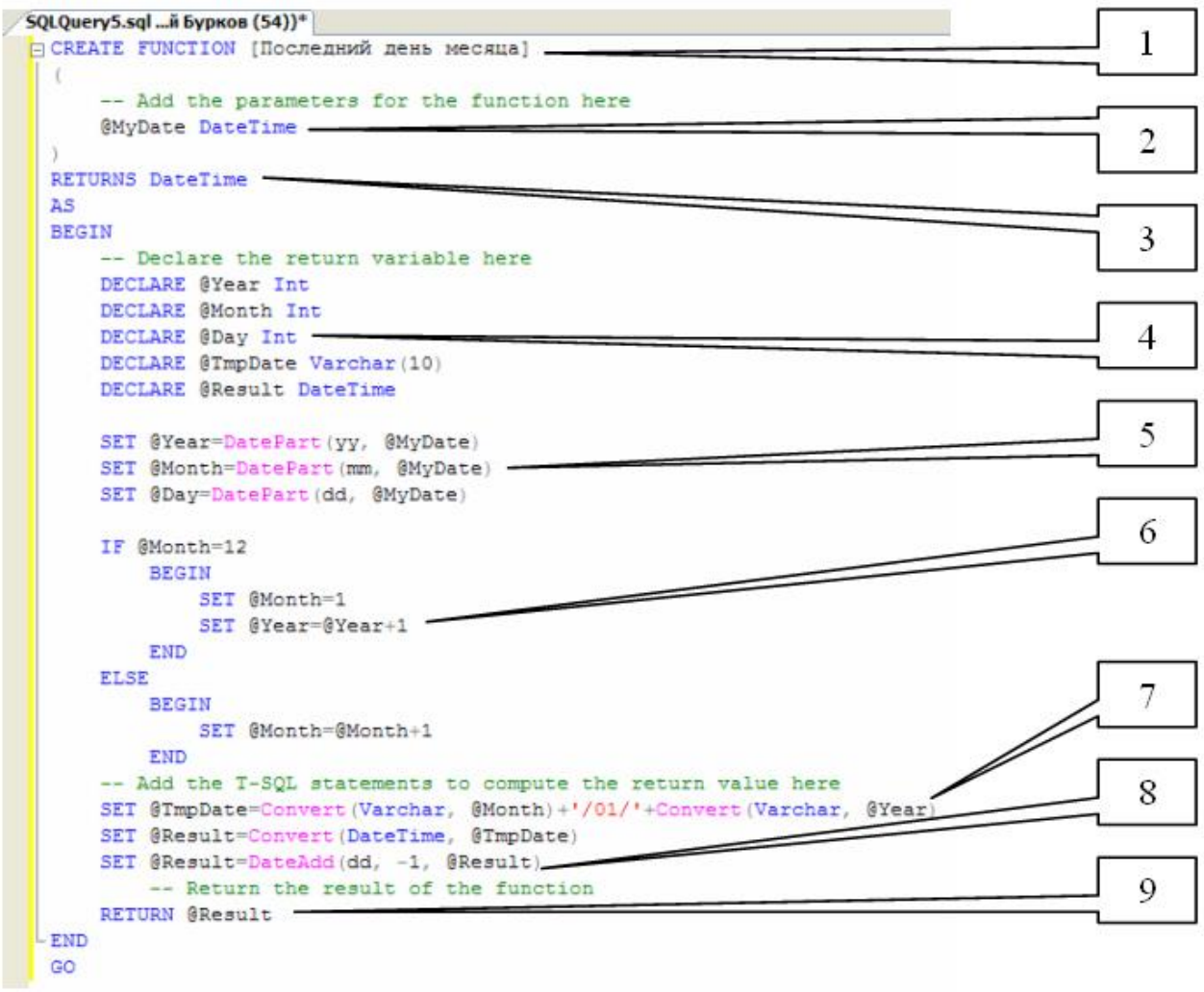


Рисунок 5

Перейдем к рассмотрению вышеприведенного кода (рис. 5). Код состоит из следующих групп команд:

1. CREATE FUNCTION [Последний день месяца] - определяет имя создаваемой функции как "Последний день месяца";

2. @MyDate - определяют параметр процедуры **MyDate**. Параметру можно присвоить значения дат или времени (Тип данных DateTime);

3. RETURNS DateTime - показывает, что функция возвращает дату или время (Тип данных DateTime);

4. DECLARE @Year Int, DECLARE @Month Int, DECLARE @Day Int - объявляются переменные @Year, @Month и @Day для хранения целочисленных значений года, месяца и дня введенной даты (Тип данных Int).

DECLARE @TmpDate VarChar(10) объявляет переменную "TmpDate" для хранения промежуточного значения даты в строке длинной до 10 символов (Тип данных VarChar(10)).

DECLARE @Result DateTime объявляет переменную "Result" для хранения результата - даты последнего дня месяца (Тип данных DateTime).

5. SET @Year=DatePart(yy, @MyDate), SET @Month=DatePart(mm, @MyDate), SET @Day=DatePart(dd, @MyDate) - определяются части

введенной даты и помещаются в переменные @Year, @Month и @Day. Для определения частей даты используется функция **DatePart**, имеющая следующий синтаксис: DatePart(<часть даты>, <дата>). Здесь **"часть даты"** - это закодированная специальными символами определяемая часть даты (yy - год, mm - месяц, dd - день), **"дата"** - это дата, части которой определяем.

```
6. IF @Month=12
7.         BEGIN
8.                 SET @Month=1
9.                 SET @Year=@Year+1
10.                END
11.        ELSE
12.                BEGIN
13.                        SET @Month=@Month+1
14.                END
15.
```

Выше приведенный фрагмент кода выполняет следующие действия: Если номер месяца равен 12 то установить номер месяца (@Month) равным 1 и увеличить год (@Year) на 1, иначе увеличить месяц на 1.

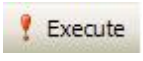
16. SET @TmpDate=Convert(Varchar, @Month)+'/01/'+Convert(Varchar, @Year), SET @Result=Convert(DateTime, @TmpDate) - переводит числовые значения даты в дату в строковом формате и записывает ее в переменную @TmpDate, затем переводит дату в строковом формате в тип данных даты и времени и помещает ее в переменную @Result. Для конвертации используется функция **Convert**, имеющая следующий синтаксис:

Convert(<тип данных>, <значение>), здесь "тип данных" это тип данных в который переводится "значение".

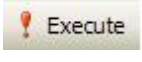
17. SET @Result=DateAdd(dd, -1, @Result) - из даты, хранимой в перменной @Result вычитается 1 день, для этого используется функция **DateAdd**, имеющая следующий синтаксис:

DateAdd(<часть даты>, <количество периодов>, <дата>) - здесь "часть даты" - это закодированная специальными символами определяемая часть даты (см. функцию **DatePart**), "количество периодов" - это количество частей даты прибавляемой к введенной дате (параметр **"дата"**).

18. RETURN @Result - возвращает значение, хранимое в переменной @Result.

Для создания функции, выполним вышеописанный код, как и в случае с предыдущей функцией, нажав кнопку 

. После появления сообщения "Command(s) completed successfully." закройте окно с кодом.

Проверим работу функции **"Последний день месяца"** выполнив ее. Создайте новый пустой запрос, затем в окне с пустым запросом наберите команду SELECT dbo.[Последний день месяца] ('12/07/08') и нажмите кнопку 

на панели инструментов (рис. 6).

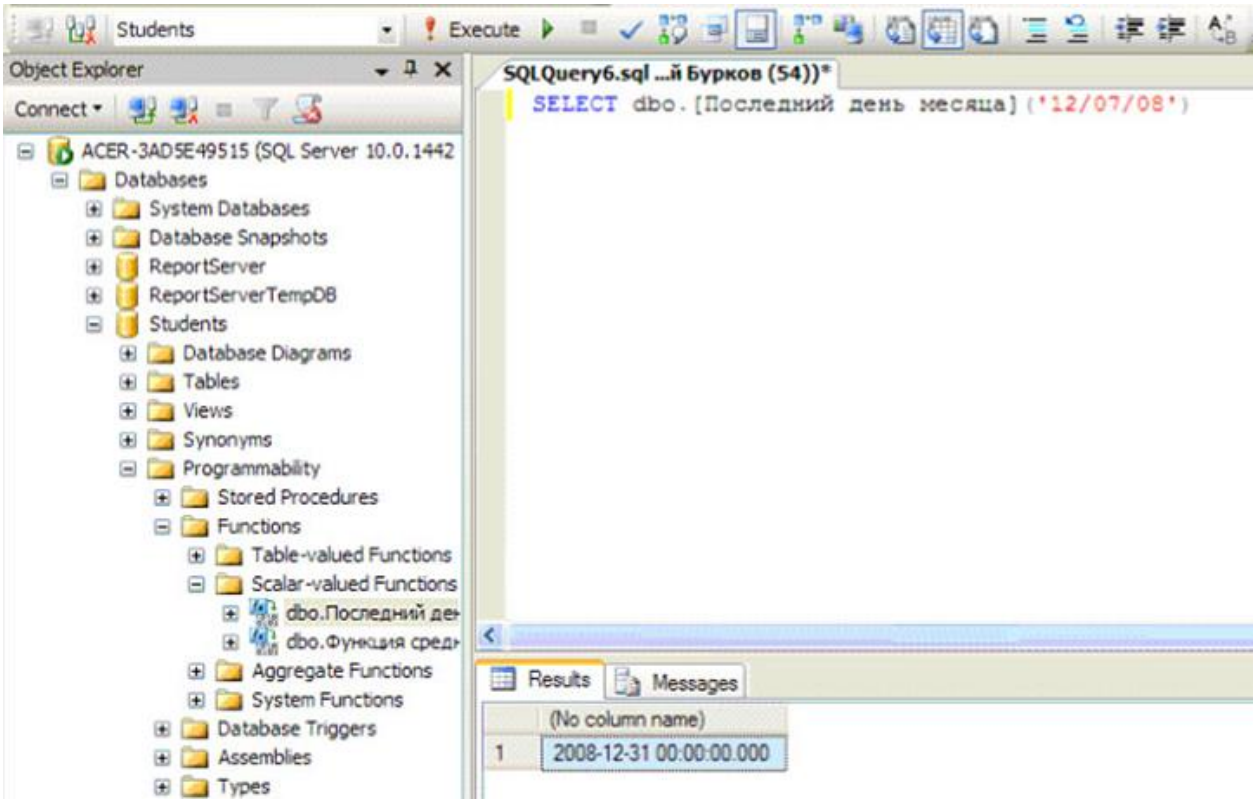


Рисунок 6

Появится результат выполнения новой скалярной пользовательской функции: **2008-12-31** (рис. 6).

Теперь перейдем к созданию табличных пользовательских функций. Для создания табличной пользовательской функции в обозревателе объектов, в БД **"Students"**, в папке **"Programmability"**, щелкните **ПКМ** по папке **"Functions"** и в появившемся меню выберите пункт **"New/Table-valued Function"**. Появится окно новой табличной пользовательской функции (рис. 6)

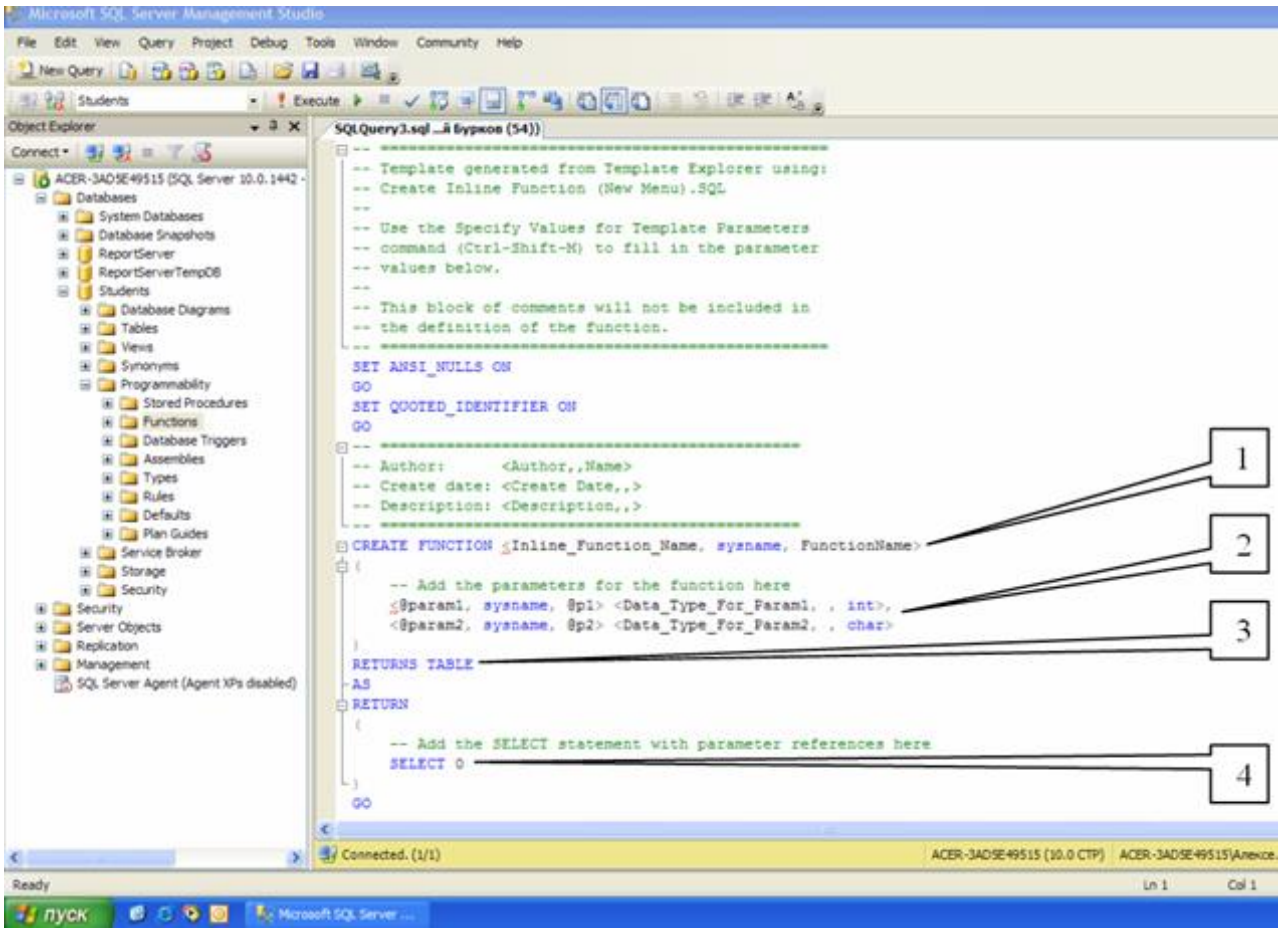


Рисунок 7

Рассмотрим структуру кода табличной пользовательской функции. Табличная пользовательская функция состоит из следующих разделов:

1. Область определения имени функции (Inline_Function_Name);
2. Параметры, передаваемые в процедуру (@Param1, @Param2);
3. RETURNS TABLE показывает что функция является табличной, то есть возвращает таблицу;

4. Тело самой пользовательской функции, состоит из команды SELECT языка программирования запросов T-SQL.

Остальные разделы табличной пользовательской функции аналогичны таким же разделам хранимых процедур и скалярных пользовательских функций.

В заключение рассмотрим создание табличной пользовательской функции **"Функция отбора по возрасту"**, вычисляющих текущий возраст студентов в зависимости от их даты рождения. В окне новой пользовательской функции (рис. 7) наберите следующий код (рис. 8):

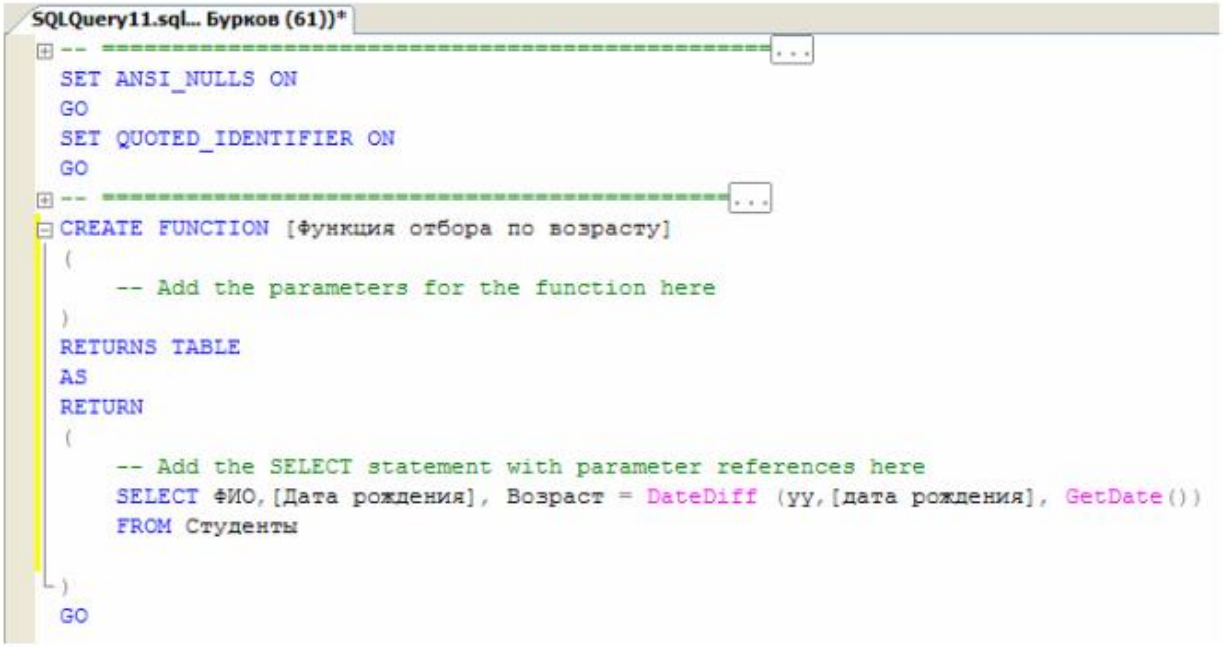


Рисунок 8

Из кода, представленного на рис. 8 видно, что данная табличная функция не имеет параметров и реализуется командой

SELECT ФИО, [Дата рождения], Возраст = DateDiff(yy, [Дата рождения], GetDate())

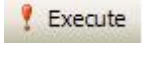
FROM Студенты

Из выше представленной команды видно, что из таблицы **"Студенты"** отображаются поля **"ФИО"** и **"Дата рождения"**, а также вычислимое поле **"Возраст"**. Поле **"Возраст"** вычисляется при помощи встроенной функции **DateDiff** вычисляющей различие между датами в определенных единицах измерения (частях даты) и имеющей следующий синтаксис:

DateDiff(<часть даты>, <начальная дата>, <конечная дата>).

Здесь `"часть даты"` - это закодированные специальными символами единицы измерения (часть даты) (yy - год, mm - месяц, dd - день), `"начальная дата"` - дата начала периода и `"конечная дата"` - дата конца периода. В нашем случае в качестве начальной даты берем дату рождения студента, а в качестве конечной даты берем текущую дату (функция **GetDate()**).

Для создания функции, выполним вышеописанный код, как и в случае с предыдущей функцией. После появления сообщения **"Command(s) completed successfully."** закройте окно с кодом.

Проверим работоспособность новой табличной пользовательской функции. Создайте новый пустой запрос, затем в окне с пустым запросом наберите команду `SELECT * FROM dbo.[Функция отбора по возрасту]()` и нажмите кнопку  на панели инструментов (рис. 9).

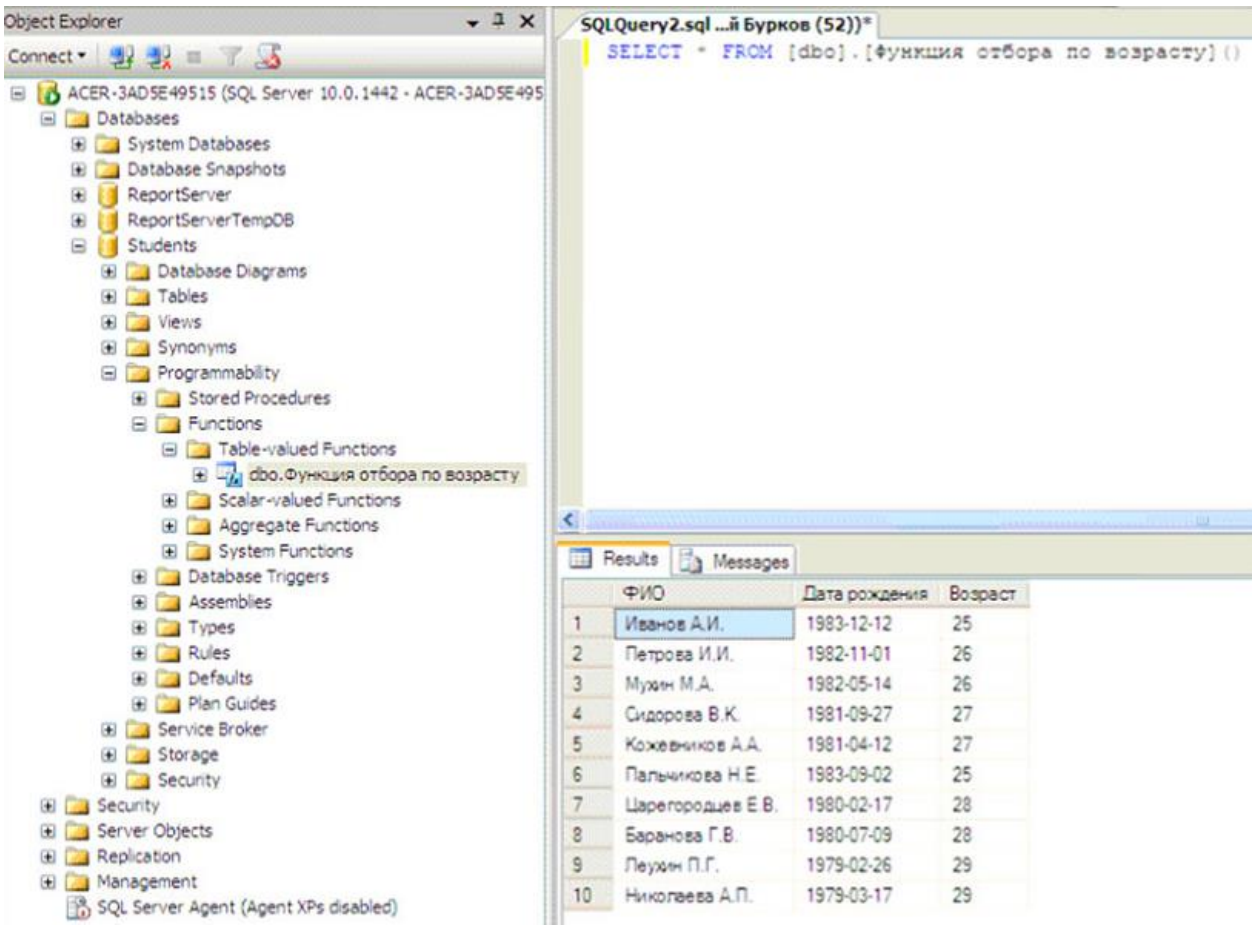


Рисунок 9

В нижней части окна появиться таблица с фамилиями, датами рождения и возрастом студентов на данный момент времени (рис. 9).

Обратите внимание на тот факт, что мы работаем с табличной функцией как с обыкновенной таблицей.