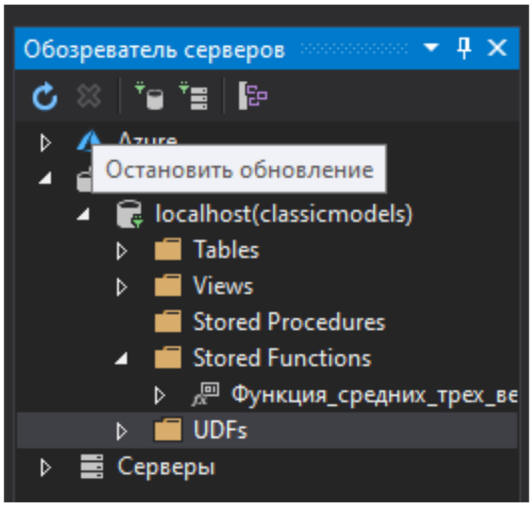
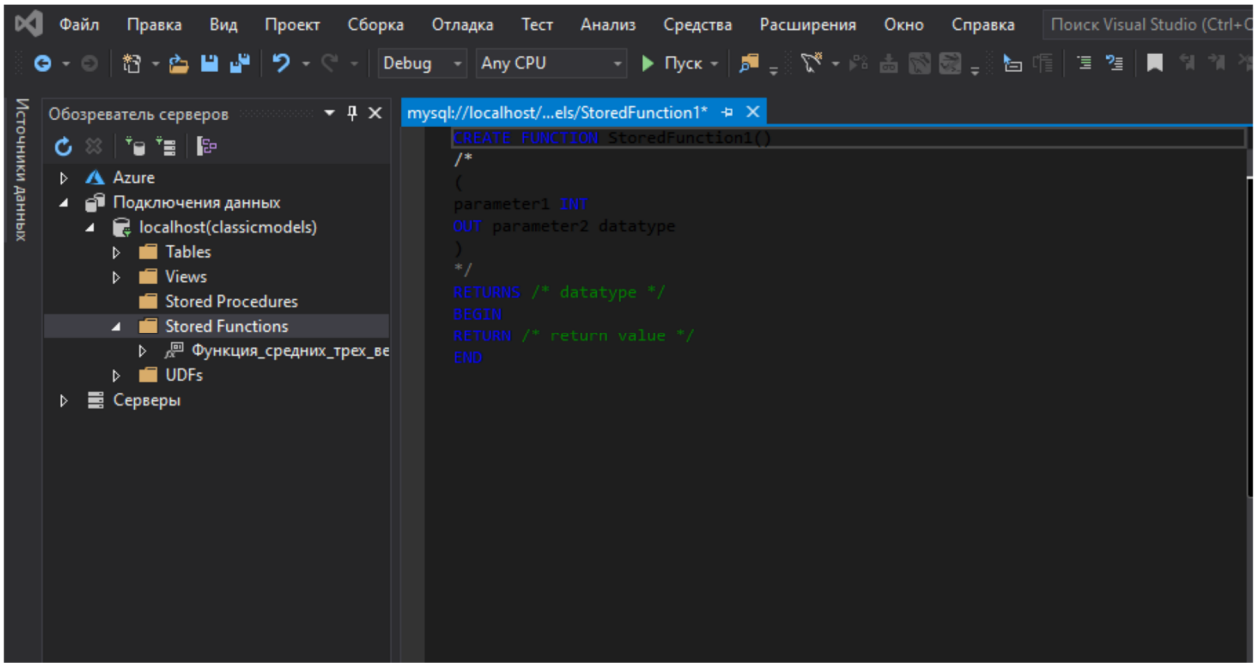
Пользовательские функции

Рассмотрим создание и применение пользовательских функций. В БД все пользовательские функции находятся в папке «Stored Functions».



Начнём с создания хранимой функций. Для создания новой хранимой функции в обозревателе объектов, в БД, щёлкните правой кнопкой мыши по папке «Stored Functions» и в появившемся меню выберите пункт «Create Stored Functions».



Синтаксис хранимой функции похож на синтаксис хранимой процедуры. Однако имеется ряд существенных отличий:

- Область определения имени функции (Scalar_Function_Name);
- Параметры, передаваемые в процедуру. Определение параметров аналогично определению параметров в хранимой процедуре;
- Тип данных значения возвращаемого процедурой;
- Область объявления переменных, используемых внутри функции. Объявление переменных имеет следующий синтаксис:
- DECLARE <Имя переменной> <Тип данных>

- Тело самой пользовательской функции, содержит команды языка программирования запросов SQL;
- Команда RETURN возвращающая результат выполнения функции. Имеет следующий синтаксис:
- RETURN <Имя переменной с результатом>

Так же можно указать дополнительные параметры:

- Deterministic - детерминированная функция всегда возвращает один и тот же результат при одинаковых входных параметрах иначе она является не детерминированной.
- Not deterministic – в противном случае
- NO SQL — не содержит sql.
- Contains SQL — содержит встроенные sql функции или операторы, которые не читают, не пишут и не изменяют данные в базе данных. Например, установка значения переменной: SET name = значение;
- READS SQL DATA — только чтение данных, без любой модификации данных, указывается для запроса SELECT.
- MODIFIES SQL DATA — изменение или внесение данных, в базу данных, указывается для запросов: INSERT, UPDATE, но при этом не должен присутствовать запрос SELECT.

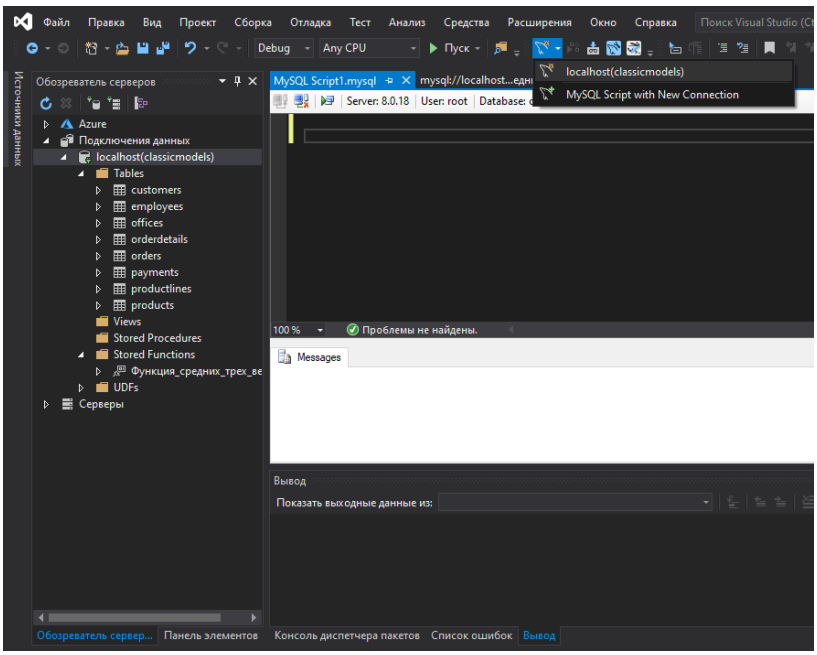
Создадим хранимую функцию, вычисляющую среднее трех величин. В окне новой пользовательской функции наберите код представленный на рисунке.

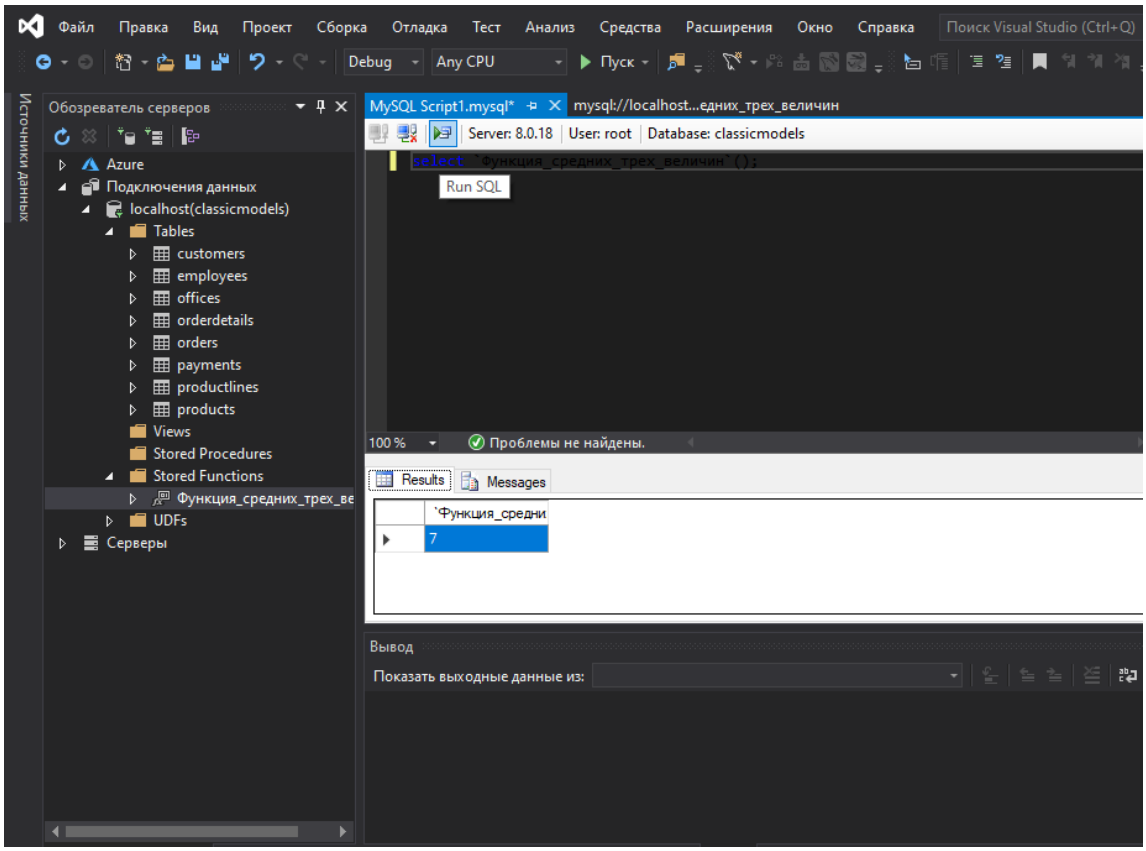
```
CREATE DEFINER=`root`@`localhost` FUNCTION `Функция_средних_трех_величин`() RETURNS
double
    DETERMINISTIC
BEGIN
DECLARE Value1 INT DEFAULT 10;
DECLARE Value2 INT DEFAULT 7;
DECLARE Value3 INT DEFAULT 4;
DECLARE Result Real;
SELECT (Value1+Value2+Value3)/3 INTO  Result;
RETURN (Result);
END
```

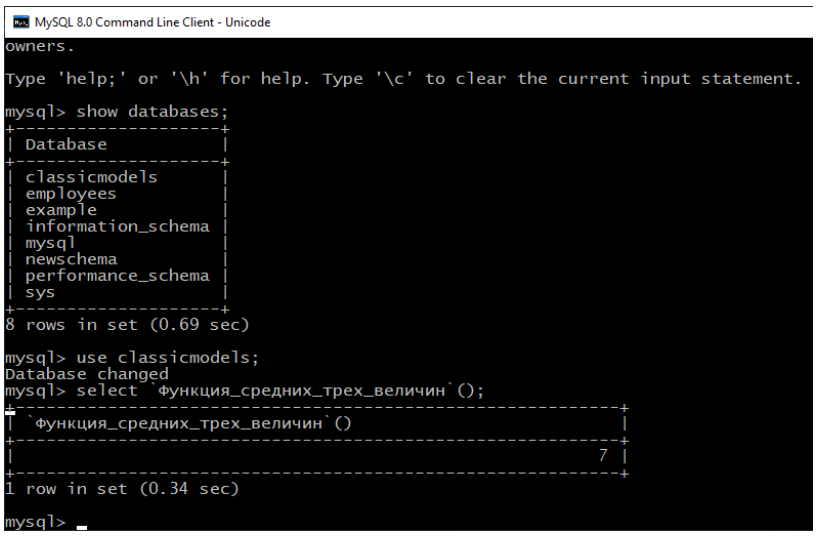
Рассмотрим более подробно код данной хранимые функции:

- CREATE FUNCTION [Функция_средних_трёх_величин] - определяет имя создаваемой функции как «Функция_средних_трёх_величии»;
- `DECLARE Value1 INT, DECLARE Value2 INT, DECLARE Value3 INT` - определяют три параметра процедуры Value1, Value2 и Value3. Данным параметрам можно присвоить целые числа (Тип данных Int);
- RETURNS Real - показывает, что функция возвращает дробные числа (Тип данных Real);
- DECLARE Result Real - объявляется переменная Result для хранения результата работы функции, то есть дробного числа (Тип данных Real);
    - `SELECT (Value1+Value2+Value3)/3 INTO  Result;` - вычисляет среднее и помещает результат в переменную Result;
- RETURN Resuit - возвращает значение переменной Result.

Для создания функции, выполним вышеописанный код, можно или из среды разработки с помощью MySql Script или из консоли MySqlServer.


Файл Правка Вид Проект Сборка Отладка Тест Анализ Средства Расширения Окно Справка
Debug
Any CPU
Пуск
Обозреватель серверов
Azure
Подключения данных
localhost(classicmodels)
Tables
customers
employees
offices
orderdetails
orders
payments
productlines
products
Views
Stored Procedures
Stored Functions
Функция_средних_трех_ве
UDFs
Серверы
MySQL Script1.mysql
mysql://localhost...едн
Server: 8.0.18
User: root
Database:
localhost(classicmodels)
MySQL Script with New Connection
100 %
Проблемы не найдены.
Messages
Вывод
Показать выходные данные из:
Обозреватель сервер...
Панель элементов
Консоль диспетчера пакетов
Список ошибок
Вывод

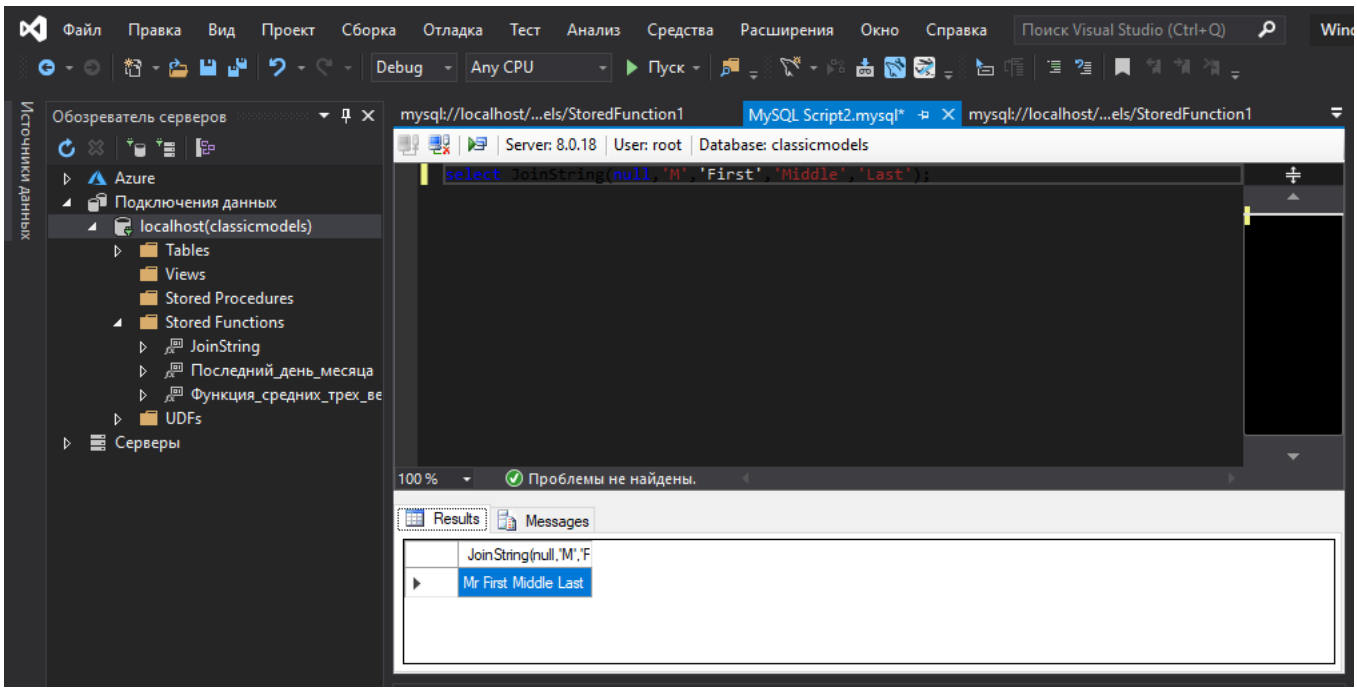

Файл Правка Вид Проект Сборка Отладка Тест Анализ Средства Расширения Окно Справка
Поиск Visual Studio (Ctrl+Q)
Debug
Any CPU
Пуск
Обозреватель серверов
Azure
Подключения данных
localhost(classicmodels)
Tables
customers
employees
offices
orderdetails
orders
payments
productlines
products
Views
Stored Procedures
Stored Functions
Функция_средних_трех_ве
UDFs
Серверы
MySQL Script1.mysql*
mysql://localhost...едних_трех_величин
Server: 8.0.18
User: root
Database: classicmodels
select `Функция_средних_трех_величин`();
Run SQL
100 %
Проблемы не найдены.
Results
Messages
'Функция_средни
7
Вывод
Показать выходные данные из:



Теперь создадим более сложную хранимую функцию, предназначенную для объединения строки по гендерному различию.

```
CREATE DEFINER=`root`@`localhost` FUNCTION `JoinString`(
            in_title          VARCHAR(4),
            in_gender         CHAR(1),
             in_firstname      CHAR(20),
             in_middle_initial CHAR(20),
             in_surname        CHAR(20)) RETURNS char(60) CHARSET latin1
    DETERMINISTIC
BEGIN
       DECLARE l_title               VARCHAR(4);
       DECLARE l_name_string         VARCHAR(60);
       IF ISNULL(in_title)  THEN
          IF in_gender='M' THEN
             SET l_title='Mr';
          ELSE
             SET l_title='Ms';
          END IF;
       END IF;
           IF ISNULL(in_middle_initial) THEN
          SET l_name_string=CONCAT(l_title," ",in_firstname," ",in_surname);
       ELSE
          SET l_name_string=CONCAT(l_title," ",in_firstname," ",in_middle_initial,"
",in_surname);
       END IF;

       RETURN(l_name_string);
            END
```



Теперь создадим более сложную хранимую функцию, предназначенную для определения последнего дня месяца введённой даты.

Задавайте дату в виде строки 'YYYY-MM-DD'. Проблема в том, что в указанном формате MySQL-сервер всегда однозначно интерпретирует дату, независимо от региональных и прочих настроек. Например, " ... WHERE SomeDateField<'2001-03-02' ... ". Эта дата при любом раскладе будет интерпретирована сервером как 2 марта 2001 года.

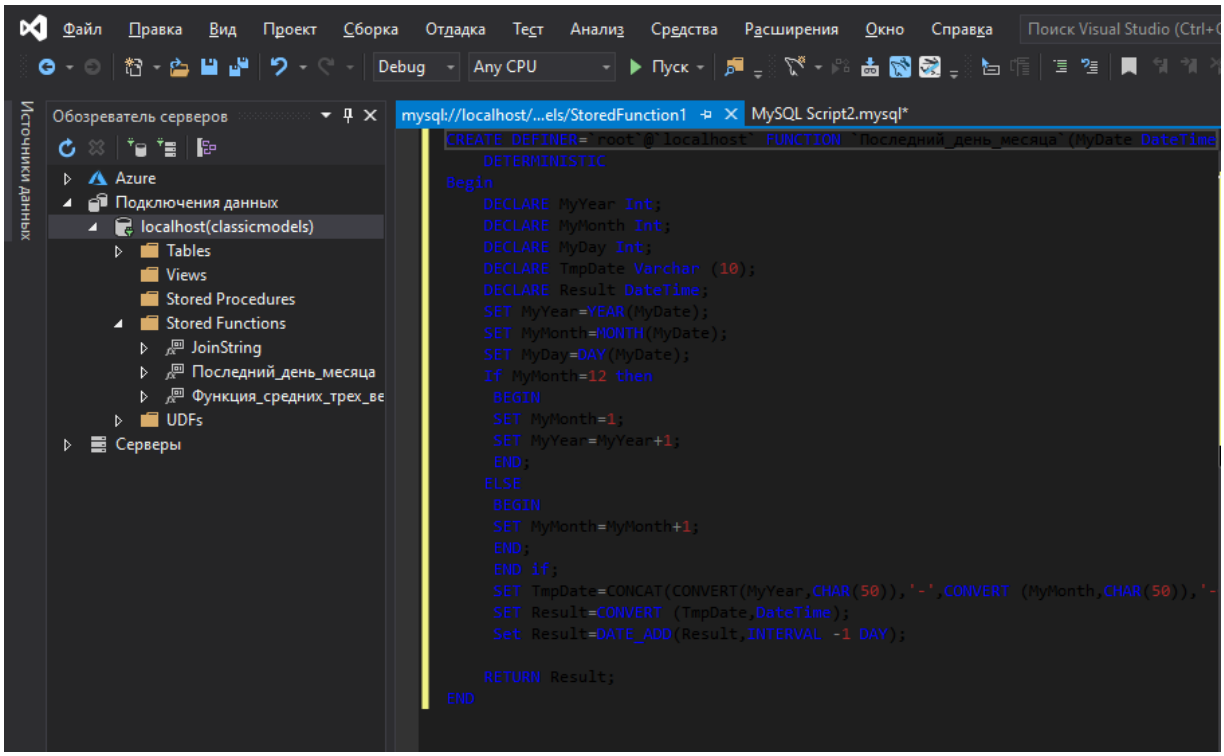
```
'20120810' – 10 августа 2012 года
'2012-08-10' – 8 октября 2012 года
```

Создайте новую хранимую функцию, так как об этом сказано выше. В окне новой пользовательской функции наберите следующий код:

```
CREATE DEFINER=`root`@`localhost` FUNCTION `Последний_день_месяца`(MyDate DateTime)
RETURNS datetime
    DETERMINISTIC
Begin
       DECLARE MyYear Int;
       DECLARE MyMonth Int;
       DECLARE MyDay Int;
       DECLARE TmpDate Varchar (10);
       DECLARE Result DateTime;
       SET MyYear=YEAR(MyDate);
       SET MyMonth=MONTH(MyDate);
       SET MyDay=DAY(MyDate);
       If MyMonth=12 then
        BEGIN
        SET MyMonth=1;
        SET MyYear=MyYear+1;
        END;
       ELSE
        BEGIN
```

```
        SET MyMonth=MyMonth+1;
        END;
        END if;
        SET TmpDate=CONCAT(CONVERT(MyYear,CHAR(50)),'-',CONVERT (MyMonth,CHAR(50)),'-
01');
        SET Result=CONVERT (TmpDate,DateTime);
        Set Result=DATE_ADD(Result,INTERVAL -1 DAY);

       RETURN Result;
END
```



Перейдём к рассмотрению вышеприведенного кода (Рисунок 9.5). Код состоит из следующих групп команд:

- CREATE FUNCTION [Последний день месяца] - определяет имя создаваемой функции как «Последний день месяца»;
- MyDate - определяют параметр процедуры MyDate. Параметру можно присвоить значения дат или времени (Тип данных DateTime);
- RETURNS DateTime - показывает, что функция возвращает дату или время (Тип данных DateTime);
- DECLARE MyYear Int, DECLARE MyMonth Int, DECLARE MyDay Int- объявляются переменные MyYear, MyMonth и MyDay для хранения целочисленных значений года, месяца и дня введённой 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 Resuit DateTime объявляет переменную «Result» для хранения результата - даты последнего дня месяца (Тип данных DateTime).

- SET MyYear=YEAR (MyDate), SET MyMonth=MONTH(MyDate) , SET MyDay=DAY(MyDate) - определяются части введённой даты и помещаются в переменныне MyYear, MyMonth и MyDay.

IF @Month=12

BEGIN

SET @Month=l SET @Year=@Year+l

END

ELSE

BEGIN

SET @Month=@Month+l

END

Выше приведённый фрагмент кода выполняет следующие действия: Если номер месяца равен 12 то установить номер месяца (MyMonth) равным 1 и увеличить год (MyYear) на 1, иначе увеличить месяц на 1.

□ SET TmpDate=CONCAT(CONVERT(MyYear,CHAR(50)),'-',CONVERT (MyMonth,CHAR(50)), '-01') SET Result=CONVERT (TmpDate, DateTime) - переводит числовые значения даты в дату в строковом формате и записывает сё в переменную @TmpDate, затем переводит дату в строковом формате в тип данных даты и времени и помещает её в переменную Result. Для конвертации используется функция Convert, имеющая следующий синтаксис:

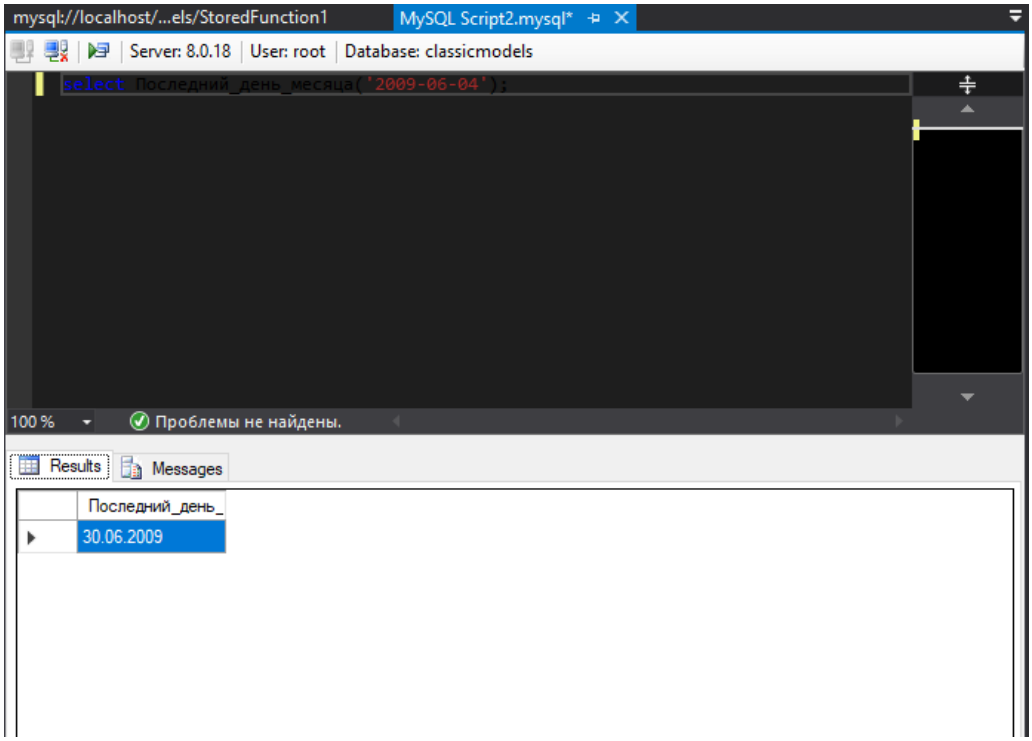
Convert (<значение>, <тип данных>), здесь «тип данных» это тип данных в который переводится «значение».

□ SET Result=DATE_ADD(Result,INTERVAL -1 DAY) - из даты, хранимой в переменной Result вычитается 1 день, для этого используется функция Convert, имеющая следующий синтаксис:

Date_Add(<дата>, <интервал добавления>).

RETURN Result - возвращает значение, хранимое в переменной Result. Для создания функции, выполним вышеописанный код, как и в случае с предыдущей функцией.

Проверим работу функции «Последний день месяца» выполнив её. Создайте новый пустой запрос, затем в окне с пустым запросом наберите команду



Создание объектов Command в Visual Studio .NET

Visual Studio .NET экономит вам время, усилия и избавляет от головной боли, позволяя легко и просто создавать и конфигурировать объекты Command. Рассмотрим возможности Visual Studio .NET периода разработки, предназначенные для работы с объектами Command.

Задание значения свойства Connection

Прежде чем добавить код, который выполнит наш объект Commands. Свойство CommandType упрощает написание текста запроса и может принимать константы из перечисления CommandType (доступно в пространстве имен SystemData}. Если задать свойству CommandType значение TableDirect, объект Command при выполнении запроса дополнит значение свойства Commaridlext префиксом «SELECT " FROM». Это означает, что при успешном выполнении запроса объект Command выберет все записи и столбцы таблицы.

Константа	Значение	Описание
Text	1	Объект Command не изменяет содержимое свойства CommandText
StoredProcedure	4	Объект Command создаст запрос для вызова хранимой процедуры, используя в качестве имени последней значение свойства CommandText
TableDirect	512	Объект Command дополнит значение свойства CommandText префиксом «SELECT ' FROM*

Наиболее часто используемые методы объекта Command

Метод	Описание
Cancel	Отменяет выполнение запроса
CrealeParameter	Создает для запроса новый параметр
ExecuteNonQuery	Предназначен для выполнения запросов, не возвращающих записей
ExecuteReader	Выполняет запрос и вставляет его результаты
ExecuteScalar	Выполняет запрос и получает первое поле первой записи, Предназначен для единичных запросов типа "SELECT COUNT(*) FROM HyTable WHERE..."
Prepare	Создает в хранилище данных подготовленную версию запроса
ResetCommandTimeout	Задает свойству CommandTimeout его значение по умолчанию — 30 секунд

Метод Cancel

Метод Cancel позволяет отменить выполнение запроса. Если вызвать метод Cancel объекта Command, не выполняющего в данный момент запрос, ничего не произойдет. Кроме того, при вызове метода Cancel объект Command отбрасывает все записи объекта DataReader, которые он не успел считать. Следующий фрагмент кода получает результаты простого запроса. Код выводит результаты, а после них — число полученных записей. В коде имеется закомментированный вызов метода Cancel. Удалив символ комментария и выполнив запрос, вы увидите, что метод Cancel отбрасывает его результаты.

Метод ExecuteNonQuery

Метод ExecuteNonQuery выполняет запрос, не создавая объект DataReader для приема его результатов. Используйте этот метод, если вам нужно выполнить командный запрос или не требуется просматривать возвращаемые запросом записи. Значения возвращаемых параметров и параметров вывода становятся доступными по завершении вызова метода ExecuteNonQuery. Метод ExecuteNonQuery возвращает целое число, соответствующее количеству записей,

затронутых вашим запросом. Подробнее о возвращаемом значении метода ExecuteNonQuery и пакетных запросах — в начале главы.

Метод ExecuteReader

Метод ExecuteReader объекта Command помещает ряды, возвращаемые запросом, в новый объект DataReader. Ранее мы уже обсуждали основы использования этого метода, но он предоставляет еще и некоторые интересные возможности, о которых я расскажу сейчас. Метод ExecuteReader объекта Command перегружен и может принимать значения из перечисления CommandBebavior.

Константа	Значение	Описание
CloseConnection	32	При закрытии объекта DataReader закрывается и соединение
KeyInfo	4	Объект DataReader получает сведения первичного ключа для столбцов, входящих в набор результатов
ScbemaOnly	2	Объект DataReader содержит только информацию о столбцах; запрос фактически не выполняется
SequentialAccess	16	Значения столбцов доступны только в последовательном порядке. Например, просмотрев содержимое третьего столбца, просмотреть содержимое первого и второго столбцов вы уже не сможете
SingleResult	1	Объект DataReader содержит результаты только первого запроса, возвращающего записи
SingteRow	8	Объект DataReader содержит только первую запись, возвращенную запросом

Метод ExecuteScalar

Метод ExecuteScalar аналогичен методу ExecuteReader за исключением того, что возвращает первое поле первой записи набора результатов, задавая его значению универсальный тип данных Object. Если запрос возвращает больше одной ячейки данных, вторая и последующие ячейки отбрасываются. Если ваш запрос по аналогии с приведенным далее возвращает одну ячейку данных, для повышения производительности кода можно воспользоваться методом

ExecuteScalar.

SELECT СОШ(-) FROM HyTable

Метод Prepare

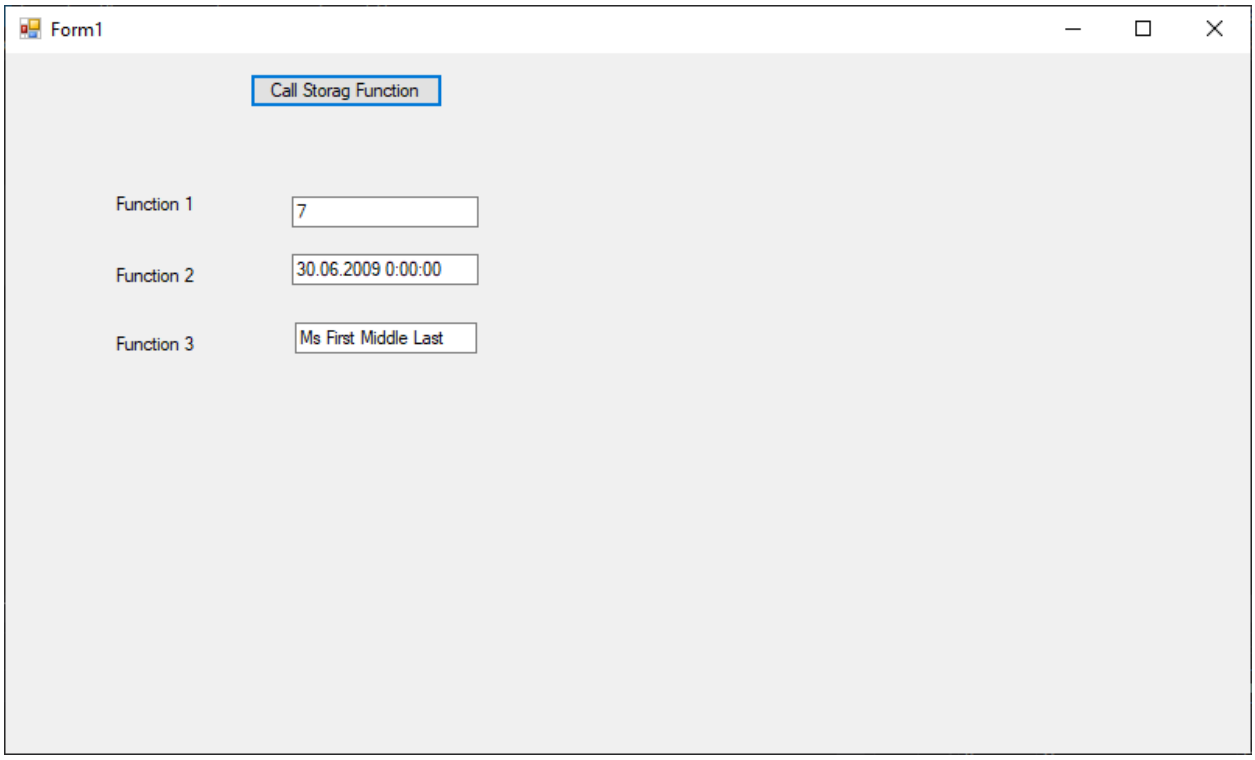
Одно из основных преимуществ хранимых процедур — то, что по сравнению с динамическими запросами они обычно выполняются быстрее. Это обусловлено тем, что СУБД способны заранее подготовить план выполнения процедуры. Такое отличие сравнимо с разницей между кодом сценария и скомпилированным кодом. Код сценария зачастую более гибок, поскольку его разрешается генерировать в период выполнения, но зато скомпилированный код выполняется быстрее. Большинство СУБД поддерживают понятие «подготовленного» запроса; такой запрос можно рассматривать как временную хранимую процедуру. Если вам придется многократно выполнять один и тот же запрос, для повышения производительности его стоит подготовить. Чтобы подготовить объект Command, достаточно вызвать его метод Prepare. При разработке многоуровневых приложений подготовленные запросы вряд ли обеспечат прирост производительности. Более того, я не рекомендую использовать их в таких ситуациях. Код серверных компонентов многоуровневого приложения обычно подключается к БД, выполняет один-два запроса и затем отключается. Многие многоуровневые приложения используют на промежуточном уровне преимущества пула соединений. Говоря по-простому, пул соединений в

течение краткого периода времени поддерживает соединение. Если вашему коду требуется соединение, параметры которого совпадают с параметрами соединения из пула, код не будет создавать новое соединение, воспользуется соединением из пула. Это значительно повышает производительность кода промежуточного уровня. Тем не менее, если соединения постоянно заимствуются из пула, а не закрываются, БД не сможет удалить все временные хранимые процедуры, созданные для ваших подготовленных запросов.

```
        private void Button2_Click(object sender, EventArgs e)
        {
            string cs = @"server=localhost;userid=root;
            password=urd741;database=classicmodels";

            using (MySqlConnection conn = new MySqlConnection(cs))
            {
                MySqlCommand command = new MySqlCommand("Функция_средних_трех_величин",
conn);
                command.CommandType = System.Data.CommandType.StoredProcedure;
                command.Parameters.Add("@ireturnvalue", MySqlDbType.Double);
                command.Parameters["@ireturnvalue"].Direction =
ParameterDirection.ReturnValue;
                conn.Open();
                command.ExecuteScalar();
                textBox1.Text =
((double)command.Parameters["@ireturnvalue"].Value).ToString();
                conn.Close();
            }
            using (MySqlConnection conn = new MySqlConnection(cs))
            {
                MySqlCommand command = new MySqlCommand("Последний_день_месяца", conn);
                command.CommandType = System.Data.CommandType.StoredProcedure;
                command.Parameters.AddWithValue("?Mydate", "2009-06-04");
                command.Parameters["?Mydate"].Direction = ParameterDirection.Input;
                command.Parameters.Add("@ireturnvalue", MySqlDbType.DateTime);
                command.Parameters["@ireturnvalue"].Direction =
ParameterDirection.ReturnValue;
                conn.Open();
                command.ExecuteScalar();
                textBox2.Text =
((DateTime)command.Parameters["@ireturnvalue"].Value).ToString();
                conn.Close();
            }
            using (MySqlConnection conn = new MySqlConnection(cs))
            {
                MySqlCommand command = new MySqlCommand("JoinString", conn);
                command.CommandType = System.Data.CommandType.StoredProcedure;
                command.Parameters.AddWithValue("?in_title", null);
                command.Parameters["?in_title"].Direction = ParameterDirection.Input;
                command.Parameters.AddWithValue("?in_gender", 'M');
                command.Parameters["?in_gender"].Direction = ParameterDirection.Input;
                command.Parameters.AddWithValue("?in_firstname", "First");
                command.Parameters["?in_firstname"].Direction = ParameterDirection.Input;
                command.Parameters.AddWithValue("?in_middle_initial", "Middle");
                command.Parameters["?in_middle_initial"].Direction =
ParameterDirection.Input;
                command.Parameters.AddWithValue("?in_surname", "Last");
                command.Parameters["?in_surname"].Direction = ParameterDirection.Input;
                command.Parameters.Add("@ireturnvalue", MySqlDbType.VarString);
                command.Parameters["@ireturnvalue"].Direction =
ParameterDirection.ReturnValue;
                conn.Open();
                command.ExecuteScalar();
```

```
                textBox3.Text =
((string)command.Parameters["@ireturnvalue"].Value).ToString();
                conn.Close();
            }
        }
```



Задание

Для выбранной базы данных создать 3 хранимых функции с выводом в оконном приложении с входными данными.