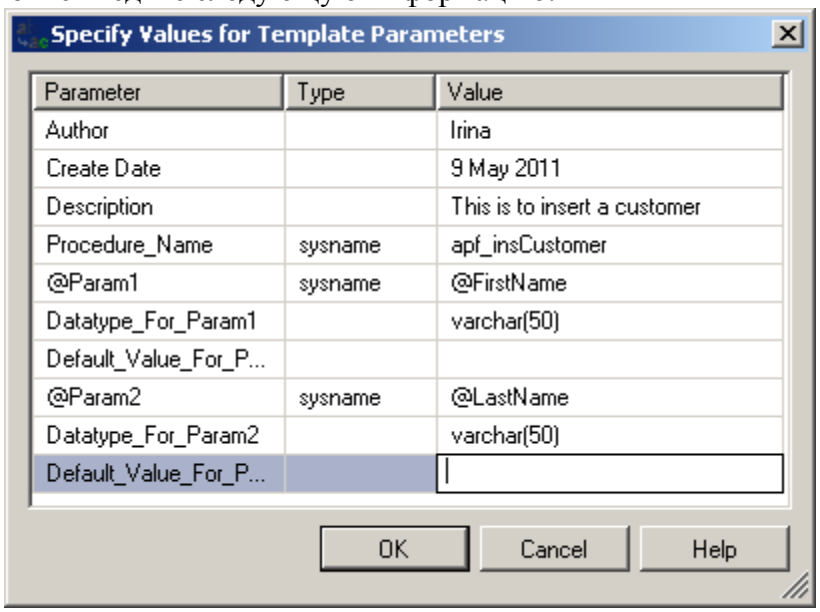
Лабораторная работа 7:

Реализация пользовательских функций и хранимых процедур

Создание хранимой процедуры в SSMS

В этом задании вы создадите хранимую процедуру, предназначенную для вставки клиента в таблицу **CustomerDetails.Customers** из переданной в процедуру информации.

1. Запустите SSMS:
   Start → Programs → Microsoft SQL Server 20XX **→ SQL Server Management Studio**
2. Раскройте БД **ApressFinancial** до узла **Programmability** (Программирование), раскройте его и щелкните 1ПКМ по **Stored Procedures** (Хранимые процедуры). Из контекстного меню выберите **New Stored Procedure…** (Создать хранимую процедуру).
3. Откроется панель **Query Editor** (Редактор запросов) с кодом основного шаблона хранимой процедуры (шаблон **Create Stored Procedures (New Menu)**). Откройте ОД **Specify Values for Template Parameters (**из меню **Query** (Запрос) → **Specify Values for Template Parameters,** или нажав <**Ctrl**>+<**Shift**>+<**M**>, или кнопкой на ПИ **SQL Editor** (Редактор SQL)).
4. В открывшемся окне введите следующую информацию:



   Первые три параметра не являются частью синтаксиса, поэтому введите свое имя и фамилию, дату создания и описание – что выполняет хранимая процедура. Эти параметры как описание должны входить в каждую хранимую процедуру.
   Имя процедуры имеет префикс – это принадлежность БД **ApressFinancial**, а само имя должно отражать выполняемые процедурой действия.
   Первые два входных параметра будут использоваться для заполнения столбцов **CustomerFirstName** и **CustomerLastName** в таблице. Удалите значения по умолчанию.
   Нажмите **OK.**
5. В окне **Query Editor** (Редактор запросов) вы увидите следующий код:

```
-- ============================================
-- Template generated from Template Explorer using:
-- Create Procedure (New Menu).SQL
--
-- Use the Specify Values for Template Parameters
-- command (Ctrl-Shift-M) to fill in the parameter
-- values below.
--
-- This block of comments will not be included in
-- the definition of the procedure.
```

```
-- ============================================
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- ============================================
-- Author:      Irina
-- Create date: 9 May 2011
-- Description: This is to insert a customer
-- ============================================
CREATE PROCEDURE apf_insCustomer
    -- Add the parameters for the stored procedure here
    @FirstName varchar(50) = ,
    @LastName varchar(50) =
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    -- Insert statements for procedure here
    SELECT @FirstName, @LastName
END
GO
```

6. Теперь добавьте к имени процедуры имя схемы можно определить оставшиеся параметры. Несмотря на то, что параметры могут идти в любом порядке, их все же следует пытаться сгруппировать. А процедура будет состоять из одной инструкции INSERT. Теперь код должен выглядеть так:

```
CREATE PROCEDURE CustomerDetails.apf_insCustomer
    -- Add the parameters for the stored procedure here
    @FirstName varchar(50),
    @LastName varchar(50),
    @CustTitle int,
    @CustInitials varchar(10),
    @AddressId int,
    @AccountNumber nvarchar(15),
    @AccountTypeId int
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;

    INSERT INTO CustomerDetails.Customers
          (CustomerTitleId, CustomerFirstName, CustomerOtherInitials,
           CustomerLastName, AddressId, AccountNumber, AccountTypeId,
           ClearBalance, UnclearBalance)
    VALUES (@CustTitle,@FirstName,@CustInitials,@LastName,
                @AddressId,@AccountNumber,@AccountTypeId, 0, 0)
END
GO
```

7. После выполнения кода, при условии, что при наборе не было ошибок, должна появиться строка: `Command(s) completed successfully.`
8. *Проверьте работу созданной хранимой процедуры*. Для этого вызовите эту процедуру: первый раз передайте ей входные параметры в том же самом порядке, в котором они определены в хранимой процедуре, а второй раз, перечисляя имена параметров и их значений:

```
CustomerDetails.apf_insCustomer 'Henry','Williams',1,NULL,431,'22067531',1;

EXEC CustomerDetails.apf_insCustomer @CustTitle=1, @FirstName='Julie',
   @CustInitials='A', @LastName='Dewson',  @AddressId=643,
   @AccountNumber='SS865', @AccountTypeId=7;
```

Должно появиться сообщение : `Command(s) completed successfully.`
Убедитесь, что оба клиента были добавлены в таблицу.

Создание хранимой процедуры с помощью Query Editor

В окне **Query Editor** (Редактор запросов) создайте хранимую процедуру с тремя входными параметрами, которая проверяет – имел ли клиент положительный или отрицательный баланс на своем счету в определенный период.

```
CREATE PROCEDURE CustomerDetails.apf_CusMovement
      @CustID bigint, @FromDate datetime, @ToDate datetime
AS
BEGIN
/* нам нужны три внутренние переменные:
   идентификатор транзакции из таблицы будем сохранять в @LastTran
   @StillCalc используется для проверки цикла WHILE */
      DECLARE @RunningBal money, @StillCalc bit, @LastTran bigint
      SELECT @StillCalc = 1, @LastTran = 0, @RunningBal = 0
-- выполнение цикла WHILE продолжается, пока инструкция возвращает строку
-- если строка не получена, значит в диапазоне дат больше нет транзакций
      WHILE @StillCalc = 1
      BEGIN
-- инструкция SELECT возвращает одну строку, в которой ( WHERE ):
-- идентификатор TransactionId больше предыдущего возвращенного идентификатора,
-- транзакция оказывает влияние на баланс клиента  и
-- транзакция находится в переданном диапазоне дат
            SELECT TOP 1  @RunningBal = @RunningBal + CASE
                              WHEN tt.CreditType = 1 THEN t.Amount
                              ELSE t.Amount* -1 END,
                        @LastTran = t.TransactionId
            FROM  CustomerDetails.Customers AS c
              JOIN TransactionDetails.Transactions AS t
                        ON t.CustomerId = c.CustomerId
              JOIN TransactionDetails.TransactionTypes AS tt
                        ON tt.TransactionTypesId = t.TransactionType
            WHERE t.TransactionId > @LastTran AND
                    tt.AffectCashBalance = 1  AND
                    DateEntered BETWEEN @FromDate AND @ToDate
            ORDER BY DateEntered
-- если строка возвращена, то выполнение цикла продолжается
            IF @@ROWCOUNT > 0
                  -- здесь следует выполнить расчет процента
                  CONTINUE
            ELSE
                  BREAK
      END
      SELECT @RunningBal AS 'End Balance'
END;
```

Для тестирования процедуры добавьте следующие данные в таблицы:

```
INSERT INTO TransactionDetails.Transactions
          (CustomerId, TransactionType, DateEntered, Amount,
           RelatedProductId)
VALUES    (1, 1, '1 Aug 2008', 100.00, 1),
          (1, 1, '3 Aug 2008', 75.67, 1),
          (1, 2, '5 Aug 2008', 35.20, 1),
          (1, 2, '6 Aug 2008', 20.00, 1);

INSERT INTO TransactionDetails.TransactionTypes
          (TransactionDescription, CreditType, AffectCashBalance)
VALUES    ( 'proc+', 1, 1),
          ( 'proc-', 0, 1);
```

Выполните процедуру:

```
EXECUTE CustomerDetails.apf_CusMovement 1,'1 Aug 2008','31 Aug 2008';
```

Код должен вернуть следующее значение:

	End Balance
1	120,47

Создание скалярной пользовательской функции

В окне **Query Editor** (Редактор запросов) создайте функцию, которая вычисляет проценты из указанной процентной ставки за количество дней, определяемое двумя датами. Процентная ставка по молчанию равна 10 (что означает 10%).

```
CREATE FUNCTION TransactionDetails.ufn_IntCalc
(@InterestRate decimal(6,3)=10, @Amount decimal(18,5),
 @FromDate date, @ToDate date)
RETURNS decimal(18,5)
WITH EXECUTE AS CALLER
AS
BEGIN
DECLARE @IntCalculated decimal(18,5)
SELECT @IntCalculated = @Amount *
 (@InterestRate / 100.00) * ( DATEDIFF(d,@FromDate,@ToDate)/365.00)
RETURN (ISNULL(@IntCalculated,0))
END;
```

Протестируйте функцию на наборе значений:

```
SELECT TransactionDetails.ufn_IntCalc (DEFAULT, 2000, 'Mar 1 2011', 'Mar 10 2011')
```

Результат должен быть следующий:

	(No column name)
1	4.93140

Функция, возвращающая табличное значение

В этом задании вы создадите функцию, которая принимает в качестве входного параметра **CustomerId** и возвращает таблицу строк **TransactionDetails.Transactions.**

```
CREATE FUNCTION TransactionDetails.ufn_ReturnTransactions (@CustID bigint)
RETURNS @Trans TABLE   (TransactionId bigint,
                                  CustomerId bigint,
                                  TransactionDescription nvarchar(30),
                                  DateEntered datetime,
                                  Amount money)
AS
BEGIN
    INSERT INTO @Trans
    SELECT TransactionId, CustomerId,TransactionDescription,
                DateEntered, Amount
    FROM TransactionDetails.Transactions AS t
          JOIN TransactionDetails.TransactionTypes AS tt
    ON tt.TransactionTypesId = t.TransactionType
    WHERE CustomerId = @CustID
    RETURN
END;
```

Протестируйте функцию на двух наборах значений:

```
SELECT  c.CustomerFirstName, c.CustomerLastName,
          Trans.TransactionId, TransactionDescription,
          DateEntered, Amount
FROM CustomerDetails.Customers AS c
    CROSS APPLY
    TransactionDetails.ufn_ReturnTransactions (c.CustomerId) AS Trans;

SELECT  c.CustomerFirstName, c.CustomerLastName,
          Trans.TransactionId, TransactionDescription,
          DateEntered, Amount
FROM CustomerDetails.Customers AS c
    OUTER APPLY
    TransactionDetails.ufn_ReturnTransactions (c.CustomerId) AS Trans;
```

Сравните результаты.