Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Национальный исследовательский университет ИТМО»

(Университет ИТМО)

Факультет **Инфокоммуникационных технологий**

Образовательная программа **11.03.02 Программирование в инфокоммуникационных системах**

ОТЧЕТ
по лабораторной работе 6
по дисциплине **«Разработка баз данных»**

Выполнил: **студент группы K33211 Швалов Д. А.**

Проверил: **ст. преподаватель Осетрова И.С.**

Санкт-Петербург

2024

Лабораторная работа №6 «Создание функций и процедур»

1. Цель работы

Создание функций и процедур.

2. Задачи, решаемые при выполнении работы

1. Создание хранимой процедуры в SSMS.
2. Создание хранимой процедуры с помощью Query Editor.
3. Создание скалярной пользовательской функции.
4. Создание функции, возвращающей табличное значение.
5. Создание хранимой процедуры, возвращающей информацию об акции.

3. Объект исследования

Создание функций и процедур в СУБД Microsoft SQL Server с помощью Microsoft SQL Server Management Studio (SSMS).

4. Исходные данные

— методические указания к лабораторной работе;
— СУБД Microsoft SQL Server;
— Microsoft SQL Server Management Studio;
— база данных ApressFinancial.

5. Выполнение работы

5.1. Первая задача

5.1.1. Создание хранимой процедуры с помощью SSMS

С помощью меню, показанного на рисунке 1, было открыто окно создания хранимой процедуры (рисунок 2). В данном окне был сгенерирован шаблонный запрос. С помощью кнопке на панели инструментов (рисунок 3) было открыто окно настроек переменных шаблона (рисунок 4), в котором были указаны значения переменных соответственно заданию. После сохранения изменений переменных, а даже дополнительного внесения изменений в код в соответствии с заданием, получился запрос, показанный на рисунке 5. После выполнения данного запроса новая процедура появилась в интерфейсе (рисунок 6).

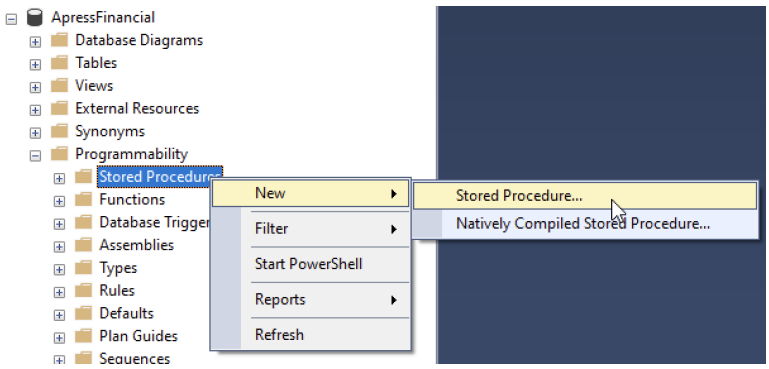


Рисунок 1 – Меню открытия окна создания хранимых процедур

```
-- ============================================
-- 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:      <Author,,Name>
-- Create date: <Create Date,,>
-- Description: <Description,,>
-- =============================================
CREATE PROCEDURE <Procedure_Name, sysname, ProcedureName>
    -- Add the parameters for the stored procedure here
    <@Param1, sysname, @p1> <Datatype_For_Param1, , int> = <Default_Value_For_Param1, , 0>,
    <@Param2, sysname, @p2> <Datatype_For_Param2, , int> = <Default_Value_For_Param2, , 0>
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 <@Param1, sysname, @p1>, <@Param2, sysname, @p2>
END
GO
```

Рисунок 2 – Окно создания хранимой процедуры

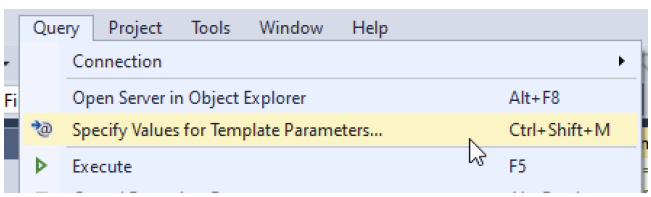


Рисунок 3 – Меню открытия окна настроек переменных шаблона

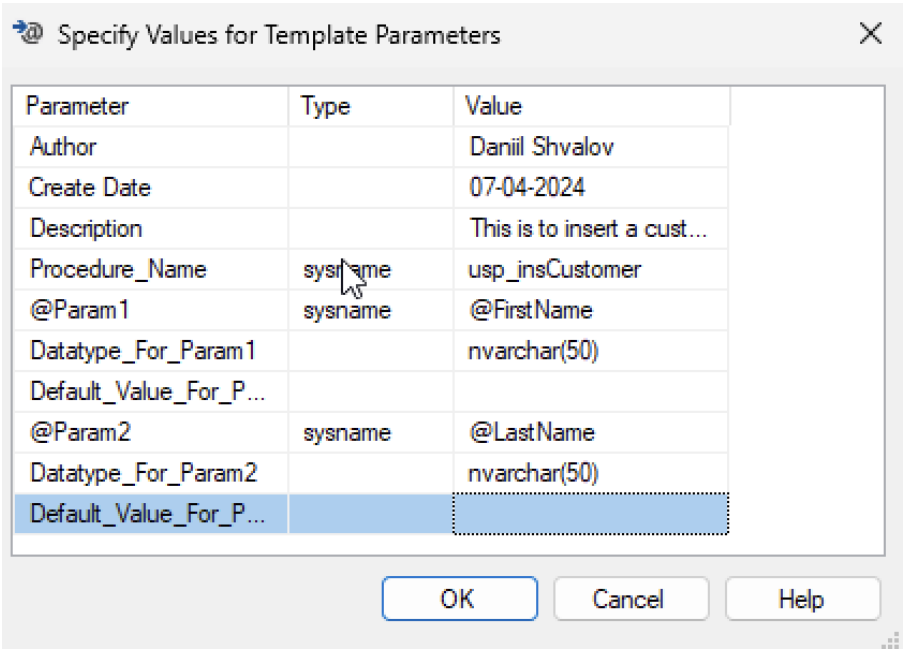


Рисунок 4 – Окно настроек переменных шаблона

```
CREATE PROCEDURE CustomerDetails.usp_InsertCustomer
    @CustTitle tinyint,
    @FirstName nvarchar(50),
    @CustInitials nvarchar(10),
    @LastName nvarchar(50),
    @AddressLine1 nvarchar(100),
    @AddressLine2 nvarchar(100),
    @AddressLine3 nvarchar(300),
    @TownOrCity int,
    @ZipCode nvarchar(20),
    @USState tinyint,
    @AccountType tinyint,
    @ClearedBalance money
AS
BEGIN
    SET NOCOUNT ON;
    INSERT INTO CustomerDetails.Customers (
        Title, FirstName, OtherInitials, LastName,
        AddressLine1, AddressLine2, AddressLine3,
        TownOrCity, ZipCode, USState, AccountType,
        ClearedBalance
    ) VALUES (
        @CustTitle, @FirstName, @CustInitials, @LastName,
        @AddressLine1, @AddressLine2, @AddressLine3,
        @TownOrCity, @ZipCode, @USState, @AccountType,
        @ClearedBalance
    )
END
GO
```

Рисунок 5 – Запрос для создания хранимой процедуры «usp_InsertCustomer»

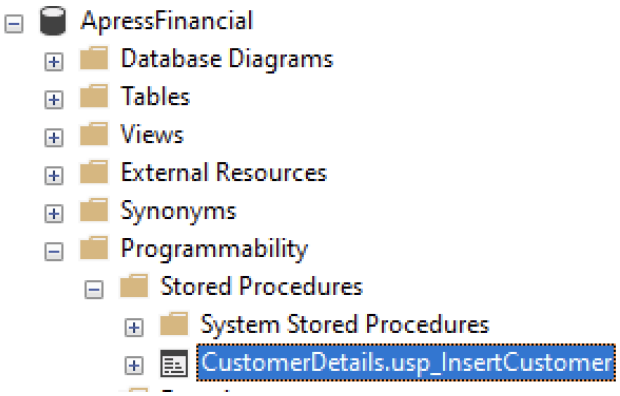


Рисунок 6 – Созданная процедура в интерфейсе SSMS

5.1.2. Тестирование хранимой процедуры

Для тестирования созданной процедуры был использован запрос, показанный на рисунке 7. С помощью данного запроса в таблицу «Customers» вставляются новые строки. С помощью запроса, изображенном на рисунке 8, были выведены созданные процедурой строки. Как видно на рисунке 9, новые строки соответствуют ожидаемым.

```
CustomerDetails.usp_InsertCustomer 1, 'Steve', NULL, 'Nicklin',
    '645 Hyde Park', NULL, NULL, 7545, 'W1A 1AA', 0, 1, 0
GO

EXEC CustomerDetails.usp_InsertCustomer @CustTitle=1, @FirstName= 'James',
    @CustInitials='A', @LastName='McVey', @AddressLine1='127 Rothsbury Avenue',
    @AddressLine2='Teguise Coast', @AddressLine3=NULL, @TownOrCity=94511,
    @ZipCode='121100', @USState=0, @AccountType=2, @ClearedBalance=0
GO
```

Рисунок 7 – Вызов созданной процедуры

```
SELECT * FROM CustomerDetails.Customers
ORDER BY CustomerId
OFFSET 1000 ROWS
GO
```

Рисунок 8 – Запрос для проверки работы процедуры

100 %

Results | Messages

	CustomerId	Title	FirstName	OtherInitials	LastName	AddressLine1	AddressLine2	AddressLine3	TownOrCity	ZipCode	USState	AccountType	ClearedBalance	UnclearedBalance	DateOpened	DateClosed
1	1001	1	James	A	McVey	127 Rothsbury Avenue	Teguise Coast	NULL	94511	121100	0	2	0.00	NULL	2024-04-07	NULL
2	1002	1	Steve	NULL	Nicklin	645 Hyde Park	NULL	NULL	7545	W1A 1AA	0	1	0.00	NULL	2024-04-07	NULL

Рисунок 9 – Созданные строки

5.2. Вторая задача

5.2.1. Создание хранимой процедуры с помощью Query Editor

С помощью запроса, показанного на рисунке 10, была создана процедура «usp_CustMovement». После выполнения данного запроса новая процедура появилась в интерфейсе SSMS (рисунок 11).

```
CREATE PROCEDURE CustomerDetails.usp_CustMovement
    @CustId int,
    @FromDate datetime,
    @ToDate datetime
AS
BEGIN
    DECLARE @RunningBal money, @StillCalc bit, @LastTrans int
    SELECT @StillCalc = 1, @LastTrans = 0, @RunningBal = 0
    WHILE @StillCalc = 1
    BEGIN
        SELECT TOP 1
            @RunningBal = @RunningBal +
                CASE
                    WHEN tt.CreditType = 1 THEN t.Amount
                    ELSE t.Amount * -1
                END,
            @LastTrans = t.TransactionId
        FROM CustomerDetails.Customers AS c
        JOIN TransactionDetails.Transactions AS t
            ON t.CustomerId = c.CustomerId
        JOIN TransactionDetails.TransactionTypes AS tt
            ON tt.TransactionTypeId = t.TransactionTypeId
        WHERE
            t.TransactionId > @LastTrans
            AND tt.AffectCashBalance = 1
            AND DateEntered BETWEEN @FromDate AND @ToDate
        ORDER BY DateEntered

        IF @@ROWCOUNT > 0
            CONTINUE
        ELSE
            BREAK

    END
    SELECT @RunningBal AS 'End Balance'
END
GO
```

Рисунок 10 – Запрос для создания процедуры «usp_CustMovement»

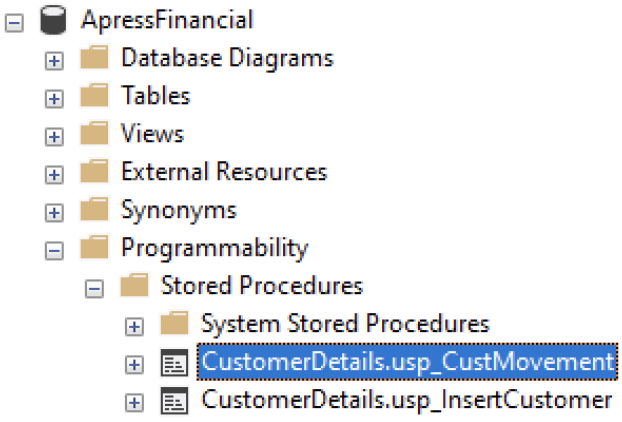


Рисунок 11 – Созданная процедура

5.2.2. Тестирование хранимой процедуры

С помощью запроса, показанного на рисунке 12, были добавлены тестовые данные в таблицу «Transactions». Затем, с помощью запроса, изображенном на рисунке 13, была вызвана созданная хранимая процедура. Как видно на рисунке 14, результат совпадает с ожидаемым.

```
INSERT INTO TransactionDetails.Transactions (
    CustomerId,
    TransactionTypeId,
    DateEntered,
    Amount,
    RelatedProductId
) VALUES
(1, 1, '01-08-2015', 100.0, 1),
(1, 1, '03-08-2015', 75.67, 1),
(1, 2, '05-08-2015', 35.20, 1),
(1, 2, '06-08-2015', 20.20, 1)
GO
```

Рисунок 12 – Запрос для добавления тестовых данных

```
EXEC CustomerDetails.usp_CustMovement 1, '01-08-2015', '31-08-2015'
GO
```

Рисунок 13 – Запрос для вызова созданной процедуры

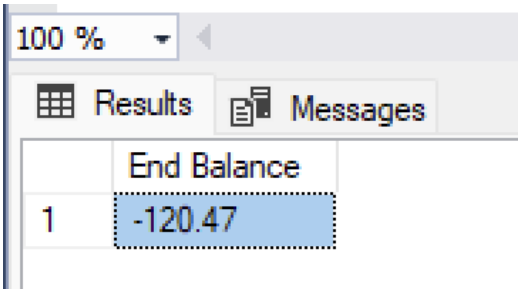


Рисунок 14 – Результат вызова процедуры

5.3. Третья задача

5.3.1. Создание скалярной функции

С помощью запроса, показанного на рисунке 15, была создана скалярная функция «ufn_IntCalc», которая рассчитывает сумму начислений на заданную сумму денежных средств за определенный период времени. После выполнения данного запроса, новая функция появилась в интерфейсе SSMS (рисунок 16).

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

Рисунок 15 – Запрос для создания скалярной функции «ufn_IntCalc»

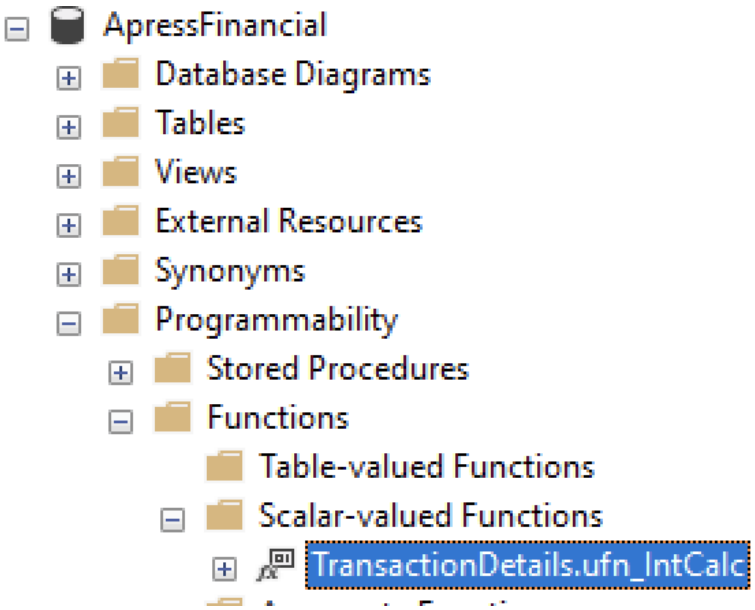


Рисунок 16 – Созданная процедура

5.3.2. Тестирование функции

С помощью запроса, показанного на рисунке 17, была протестирована созданная функция. Как видно на рисунке 18, результат совпал с ожидаемым значением.

```
SELECT TransactionDetails.ufn_IntCalc (10, 2000, '01-03-2015', '10-03-2015')
GO
```

Рисунок 17 – Запрос для тестирования созданной функции

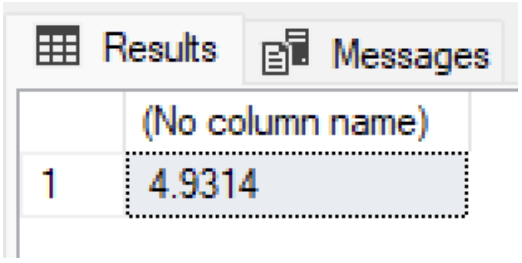


Рисунок 18 – Результат вызова функции

5.4. Четвертая задача

5.4.1. Создание функции, возвращающей табличное значение

С помощью запроса, показанного на рисунке 19, была создана функция, возвращающая информацию о транзакциях клиента. После выполнения данного запроса новая функция появилась в интерфейсе SSMS (рисунок 20).

```
CREATE FUNCTION TransactionDetails.ufn_ReturnTransactions(@CustId int)
RETURNS @Trans TABLE
(   TransactionId int,
    CustomerId int,
    TransactionDescription nvarchar(50),
    DateEntered datetime,
    Amount money    )
AS
BEGIN
    INSERT INTO @Trans
    SELECT t.TransactionId, t.CustomerId,
        tt.[Description], t.DateEntered, t.Amount
    FROM TransactionDetails.Transactions AS t
    JOIN TransactionDetails.TransactionTypes AS tt
        ON tt.TransactionTypeId = t.TransactionTypeId
    WHERE CustomerId = @CustId
    RETURN
END
GO
```

Рисунок 19 – Запрос для создания функции «ufn_ReturnTransactions», возвращающей табличное значение

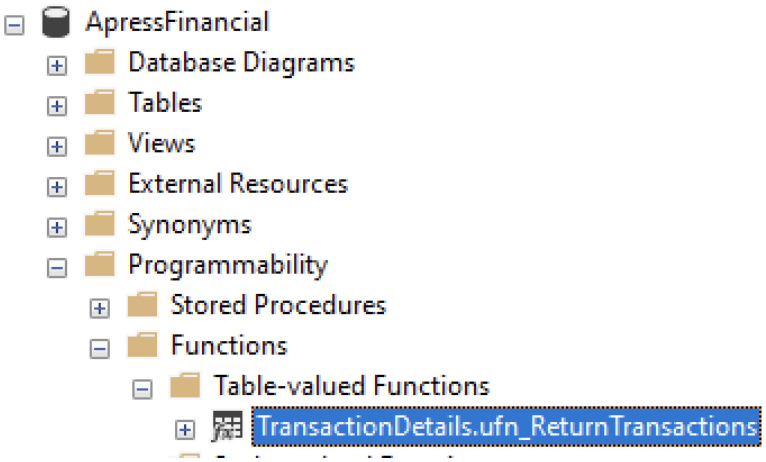


Рисунок 20 – Созданная функция

5.4.2. Тестирование созданной функции

С помощью запроса, показанного на рисунке 21, была протестирована новая функция. В данном есть два подзапроса: один использует «CROSS APPLY», другой — «OUTER APPLY». Разница между этими методами в том, что «CROSS APPLY» возвращает только те строки, которые есть как в таблице, так

и в результате вызова функции, а «OUTER APPLY» возвращает строки, которые есть в таблице, но необязательно есть в результате вызова функции. Как видно на рисунке 22, во втором случае есть строки с неопределенным значением.

```
SELECT
     c.FirstName,
     c.LastName,
     t.TransactionId,
     t.TransactionDescription,
     t.DateEntered,
     t.Amount
FROM CustomerDetails.Customers AS c
CROSS APPLY TransactionDetails.ufn_ReturnTransactions(c.CustomerId) AS t
GO

SELECT
     c.FirstName,
     c.LastName,
     t.TransactionId,
     t.TransactionDescription,
     t.DateEntered,
     t.Amount
FROM CustomerDetails.Customers AS c
OUTER APPLY TransactionDetails.ufn_ReturnTransactions(c.CustomerId) AS t
GO
```

Рисунок 21 – Запрос для тестирования созданной функции

Results Messages

	FirstName	LastName	TransactionId	TransactionDescription	DateEntered	Amount
1	Noel	Morgala	1801	Buy	2015-01-08 00:00:00.000	100.00
2	Noel	Morgala	1802	Buy	2015-03-08 00:00:00.000	75.67
3	Noel	Morgala	1803	Sell	2015-05-08 00:00:00.000	35.20
4	Noel	Morgala	1804	Sell	2015-06-08 00:00:00.000	20.20
5	Noel	Morgala	1809	Buy	2015-08-01 00:00:00.000	100.00
6	Noel	Morgala	1810	Buy	2015-08-03 00:00:00.000	75.67
7	Noel	Morgala	1811	Sell	2015-08-05 00:00:00.000	35.20
8	Noel	Morgala	1812	Sell	2015-08-06 00:00:00.000	20.00
9	Noel	Morgala	509	Buy	2011-12-05 13:54:54.880	62521.5101
10	Noel	Morgala	836	Cash Deposit	2012-03-10 16:38:08.960	48033.0788
11	Aubrey	Lomas	244	Buy	2012-03-22 22:04:05.440	14149.7153
12	Aubrey	Lomas	607	Sell	2012-03-06 14:20:18.770	34215.274
13	Bernie	McGee	926	Sell	2011-11-10 02:01:26.640	75316.5828
14	Jane	Harper	6	Sell	2012-01-21 16:14:49.450	41124.6195
15	Terence	Madden	121	Cash Withdrawal	2012-01-29 09:22:33.250	52477.0824
16	Terence	Madden	511	Buy	2012-01-11 22:57:03.670	1027.0005

	FirstName	LastName	TransactionId	TransactionDescription	DateEntered	Amount
1	Noel	Morgala	1801	Buy	2015-01-08 00:00:00.000	100.00
2	Noel	Morgala	1802	Buy	2015-03-08 00:00:00.000	75.67
3	Noel	Morgala	1803	Sell	2015-05-08 00:00:00.000	35.20
4	Noel	Morgala	1804	Sell	2015-06-08 00:00:00.000	20.20
5	Noel	Morgala	1809	Buy	2015-08-01 00:00:00.000	100.00
6	Noel	Morgala	1810	Buy	2015-08-03 00:00:00.000	75.67
7	Noel	Morgala	1811	Sell	2015-08-05 00:00:00.000	35.20
8	Noel	Morgala	1812	Sell	2015-08-06 00:00:00.000	20.00
9	Noel	Morgala	509	Buy	2011-12-05 13:54:54.880	62521.5101
10	Noel	Morgala	836	Cash Deposit	2012-03-10 16:38:08.960	48033.0788
11	Aubrey	Lomas	244	Buy	2012-03-22 22:04:05.440	14149.7153
12	Aubrey	Lomas	607	Sell	2012-03-06 14:20:18.770	34215.274
13	Bernie	McGee	926	Sell	2011-11-10 02:01:26.640	75316.5828
14	Jane	Harper	6	Sell	2012-01-21 16:14:49.450	41124.6195
15	Mark	Vernon-...	NULL	NULL	NULL	NULL
16	Terence	Madden	121	Cash Withdrawal	2012-01-29 09:22:33.250	52477.0824

Рисунок 22 – Результат выполнения запроса

5.5. Пятая задача

5.5.1. Создание хранимой процедуры для вывода информации об акциях

С помощью запроса, показанного на рисунке 23 была создана процедура «usp_AllShareDetails», которая принимает ID акции и выводит для него:

— выборку из таблицы Shares, содержащую ID акции, название и номер для фондового рынка;

— отсортированную выборку из таблицы SharePrices по дате цены в убывающем порядке, содержащую ID цены, ID акции, цену акции и дату этой цены.

В данной процедуре используется два запроса: первый выводит информацию об определенной акции, а второй — информацию о ценах данной акции. После выполнения данного запроса процедура появилась в интерфейсе SSMS (рисунок 24).

При вызове данной процедуры, например, для ID акции, равной 1, выводятся результаты, показанные на рисунке 25.

```
CREATE PROCEDURE ShareDetails.usp_AllShareDetails(@ShareId int) AS
BEGIN
    SELECT ShareId, [Description], StockExchangeTicker
    FROM ShareDetails.Shares
    WHERE ShareId = @ShareId

    SELECT SharePriceId, ShareId, Price, PriceDate
    FROM ShareDetails.SharePrices
    WHERE ShareId = @ShareId
    ORDER BY PriceDate DESC
END
```

Рисунок 23 – Запрос для создания процедуры «usp_AllShareDetails»

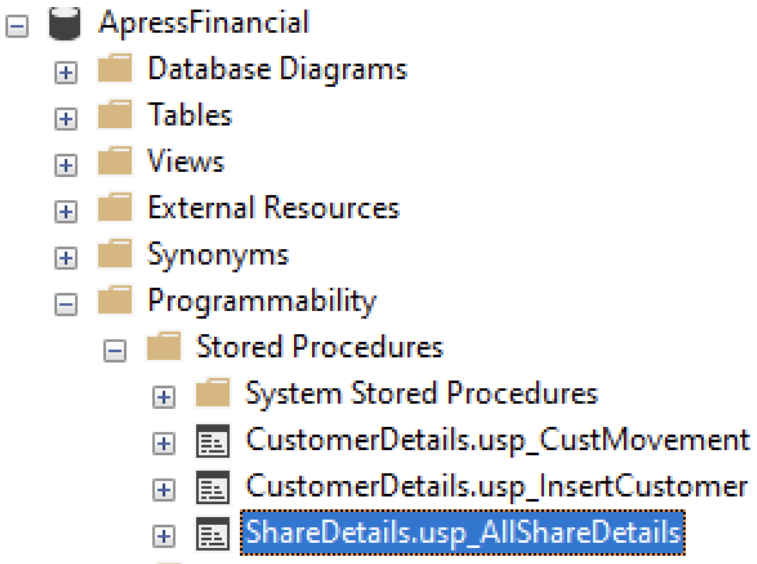
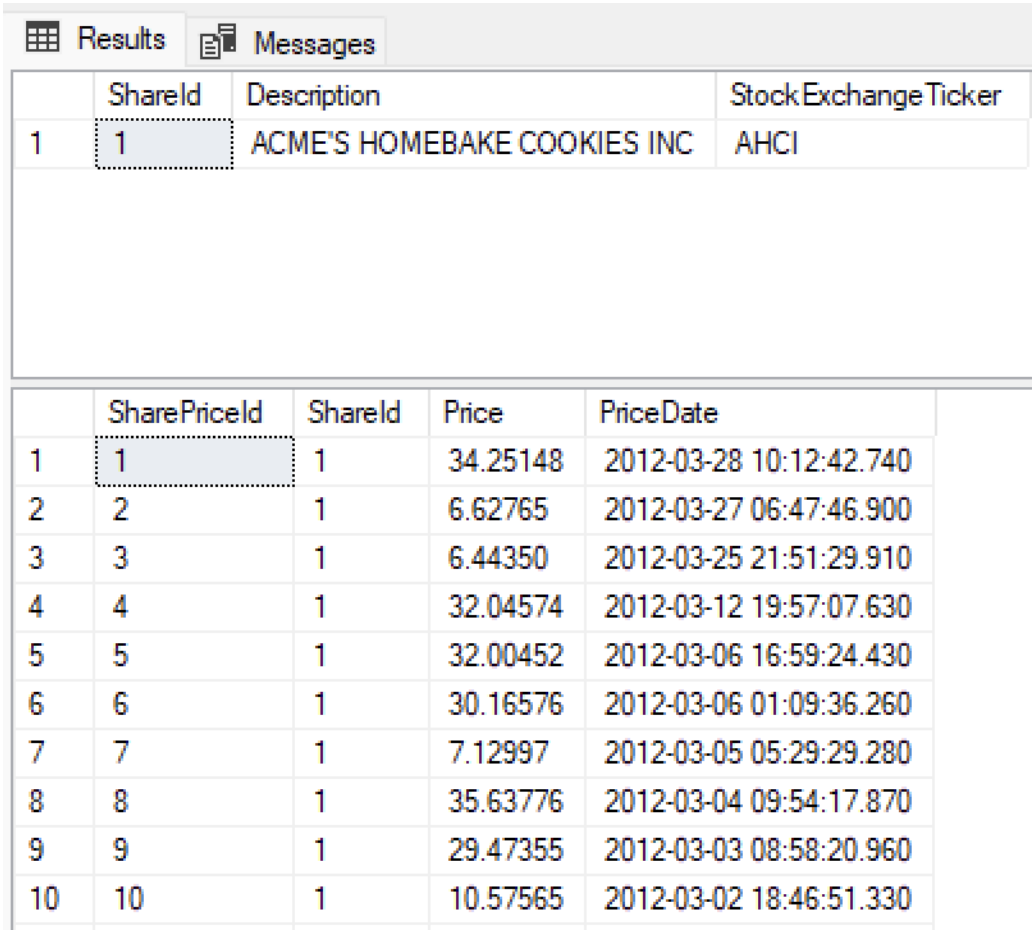


Рисунок 24 – Созданная процедура



Results Messages

	ShareId	Description	StockExchangeTicker
1	1	ACME'S HOMEBAKE COOKIES INC	AHCI

	SharePriceId	ShareId	Price	PriceDate
1	1	1	34.25148	2012-03-28 10:12:42.740
2	2	1	6.62765	2012-03-27 06:47:46.900
3	3	1	6.44350	2012-03-25 21:51:29.910
4	4	1	32.04574	2012-03-12 19:57:07.630
5	5	1	32.00452	2012-03-06 16:59:24.430
6	6	1	30.16576	2012-03-06 01:09:36.260
7	7	1	7.12997	2012-03-05 05:29:29.280
8	8	1	35.63776	2012-03-04 09:54:17.870
9	9	1	29.47355	2012-03-03 08:58:20.960
10	10	1	10.57565	2012-03-02 18:46:51.330

Рисунок 25 – Пример работы процедуры

6. Выводы и анализ результатов работы

В данной лабораторной работе изучены способы создания хранимых процедур и функций в SSMS. В ходе выполнения лабораторной работы были созданы несколько хранимых процедур и функций. Как было показано, процедуры и функции являются очень полезным инструментом, позволяющим автоматизировать различные процессы при работе с базой данных. При этом SSMS предоставляет различные способы, по-своему удобные, для создания хранимых процедур и функций. Все это делает работу с процедурами и функциями не только полезной, но и удобной.

Цель, поставленная в начале работы, достигнута, задачи выполнены.