Лабораторная работа № 4

Создание хранимых процедур

Цель работы:

1. Работа в среде Visual Studio.Net
2. Работа с языком Transact-SQL
3. Написание процедур с переменными параметрами

Хранимые процедуры позволяют выполнять массовые операции с записями, в частности, обновлять, вставлять и удалять данные из многих записей. В настоящей работе показаны способы создания хранимых процедур, а также их проверки в программе Server Explorer.
В приложении нужно не только иметь возможность просматривать данные, но и выполнять такие операции, как добавление, обновление и удаление записей. Для этой цели хорошо подходят хранимые процедуры, поэтому они часто используются в приложениях. Ниже описаны способы создания хранимых процедур в среде VS.

Общее описание

Хранимые процедуры могут применяться, если есть необходимость добавлять, удалять или обновлять записи, особенно если область действия таких операций должна быть указана с помощью параметров. В частности, если в приложении должна часто выполняться задача, требующая проведения нескольких действий, включая создание временных таблиц, такая возможность может быть реализована с использованием хранимых процедур. Хранимые процедуры являются мощным и удобным программным средством.

Для создания хранимых процедур применяется язык Transact-SQL (сокращение T-SQL). В этой работе используются только некоторые простые операторы для демонстрации интерактивного интерфейса среды VS .NET, применяемого для создания хранимых процедур, а более подробное описание операторов этого языка приведено в [ ].
Для создания хранимых процедур используется окно `Designer` программы `Visual Studio`. Но в отличие от окна `Views Designer`, окно проектировщика хранимых процедур вначале предоставляет не графический, а алфавитно-цифровой интерфейс. Тем не менее после перехода в текстовое окно проектировщика можно вызвать на экран графическое окно проектировщика.
Для создания новой хранимой процедуры щелкните правой кнопкой мыши на узле Stored Procedures в иерархическом меню базы данных, в которую необходимо ввести хранимую процедуру, а затем выберите во всплывающем меню команду New Stored Procedure. Для редактирования существующей хранимой процедуры отметьте ее на экране, щелкните правой кнопкой мыши

и выберите во всплывающем меню команду Edit Stored Procedure.
После того как откроется хранимая процедура, на экране появится оператор `Select` или целый ряд других операторов языка T-SQL. Если это — новая хранимая процедура, щелкните правой кнопкой мыши на окне и выберите во всплывающем меню команду Insert SQL. Откроется окно построителя запросов Query Builder, которое выглядит как окно View Designer. Если это — существующая хранимая процедура, вы можете поместить курсор в блок кода SQL, который обведен синей линией, и, щелкнув правой кнопкой мыши, выбрать команду Design SQL Block во всплывающем меню, как показано на рис.1. В окне появится текст

```
ALTER PROCEDURE dbo.StoredProcedure1
/*
      (
            @parameter1 datatype = default value,
            @parameter2 datatype OUTPUT
      )
*/
AS
      /* SET NOCOUNT ON */
      RETURN
```

РИС. 1 Всплывающее меню, которое позволяет разрабатывать хранимые процедуры

После этого в окне Query Builder снова появится блок SQL.
При указании параметров, которые могут использоваться в качестве критериев отбора в хранимых процедурах, необходимо ввести символ @ перед именем каждого параметра и объявить все параметры в начале хранимой процедуры. Пример подобного объявления также показан на рис. 1.

Порядок действий

В этом упражнении для получения перечня авторов, проживающих в указанном штате, будет создана хранимая процедура, состоящая из простого оператора `Select` с параметром. Если это еще не сделано, откройте окно `Server Explorer` и разверните иерархическое меню базы данных `pubs`.

1. Щелкните правой кнопкой мыши на узле Stored Procedures, а затем выберите во всплывающем меню команду New Stored Procedure. Откроется новая страница, которая представляет собой шаблон для ввода кода хранимой процедуры. На этой странице находится следующий текст:

```
CREATE  PROCEDURE dbo.StoredProcedurel /*
l
®parameterl  datatype  =  default  value,  ®parameter2
datatype OUTPUT
```

```
*/ AS
/* SET NOCOUNT ON */ RETURN
```

2. Замените весь этот фрагмент кодом, приведенным ниже.

```
CREATE  PROCEDURE  dbo.SearchState
@parState char(2)
AS
SELECT  au_lname, au_fname, state FROM authors
WHERE  (state = @parState)
RETURN
```

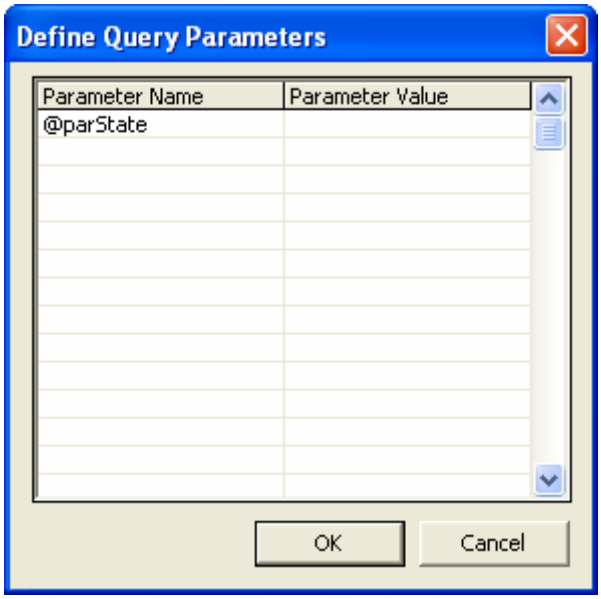
В этом коде демонстрируется использование параметра.

3. Сохраните хранимую процедуру.

`SearchState` – это имя хранимой процедуры

Описание полученных результатов

Для проверки только что созданной хранимой процедуры щелкните правой кнопк мыши на блоке кода и выберите во всплывающем меню команду Design SQL Block. Затем можно щелкнуть на кнопке Run Query панели инструментов и, после вывода на экран диалогового окна, ввести в поле параметра `parState` значение CA в окне `Parameter Value`.

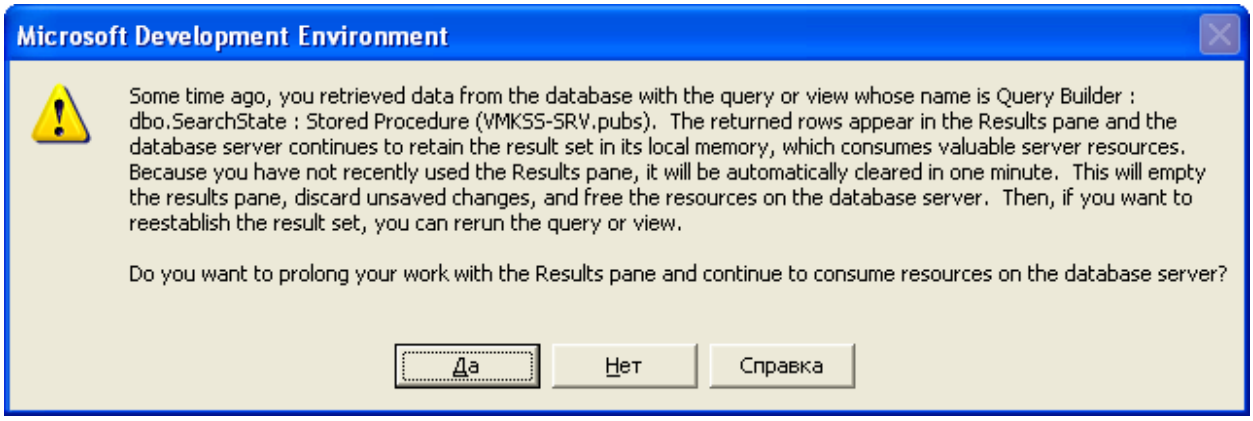


После этого в области результатов отображается полученная информация. Примеры использования хранимых процедур приведены в следующей разделе, в котором описано применение технологии `ADO .NET` в сочетании с объектами `SQL Server`.

Комментарий

Хранимые процедуры могут создаваться динамически и не записываться в базу данных, но иногда необходимо хранить их в базе данных постоянно,

чтобы иметь возможность использовать один и тот же код в разных местах приложения.
Результат хранится только определенное время. После этого появляется сообщение:



Выполнение параметризованных хранимых процедур в среде ADO.NET

Важным преимуществом хранимых процедур является возможность передавать им параметры и определять таким образом конкретные критерии отбора данных. В этом упражнении показано, как обеспечить формирование параметров в объекте `OleDbCommand` для их передачи в базу данных `SQL Server`.
Параметризованные хранимые процедуры могут найти широкое применение в приложениях. Ниже приведен пример использования такой процедуры на основе средств `Visual Basic .NET и ADO.NET`.

Общее описание

В технологии ADO.NET имеется объект Command, предназначенный для выполнения хранимых процедур. В настоящем работе применяется не только объект `Command`, но и другие объекты. Общий перечень этих объектов приведен в табл. 1.

Таблица 1. Объекты, применяемые в настоящей работе, а также их свойства и методы

Объект	Свойство или метод	Описание
Connection	ConnectionString	Содержит применяемую строку подключения
Connection	Open	Открывает соединение, которое используется объектом Command
Command	cmdText	Указывает применяемый оператор SQL. Может содержать оператор SQL или имена таких объектов, как таблицы или хранимые процедуры

Command	Connection	Использует объект Connection
Command	CommandType	Указывает тип команды, которая должна быть выполнена на сервере. В качестве значения этого свойства может быть указан один из следующих типов команд: StoredProcedure, DirectTable или Text
Command	Parameters	Параметры, передаваемые хранимой процедуре
Command	ExecuteReader	Создает объект DataReader с данными, указанными в объекте команды
DataReader	Read	Считывает следующую запись в объекте DataReader, а также проверяет, достигнут ли конец возвращаемых данных
DataReader	GetString	Возвращает текущую запись, выполняет выборку значения из указанного столбца и возвращает его в виде строки
DataReader	GetInt32	Возвращает текущую запись, выполняет выборку значения из указанного столбца и возвращает его в виде 32-битового целого числа

Примеры использования этих объектов, а также их свойств и методов приведены ниже.

Порядок действий

Чтобы получить список заказов для указанного идентификатора заказчика из базы данных `NorthWind`, по умолчанию применяется идентификатор ALFKI. После этого в нижней части формы отображается элемент управления `TextBox`. Рассматриваемая форма представлена на рис. 2.

1. Создайте новую форму Windows.
2. Введите следующие элементы управления и присвойте их свойствам значение представленные в табл. 2.

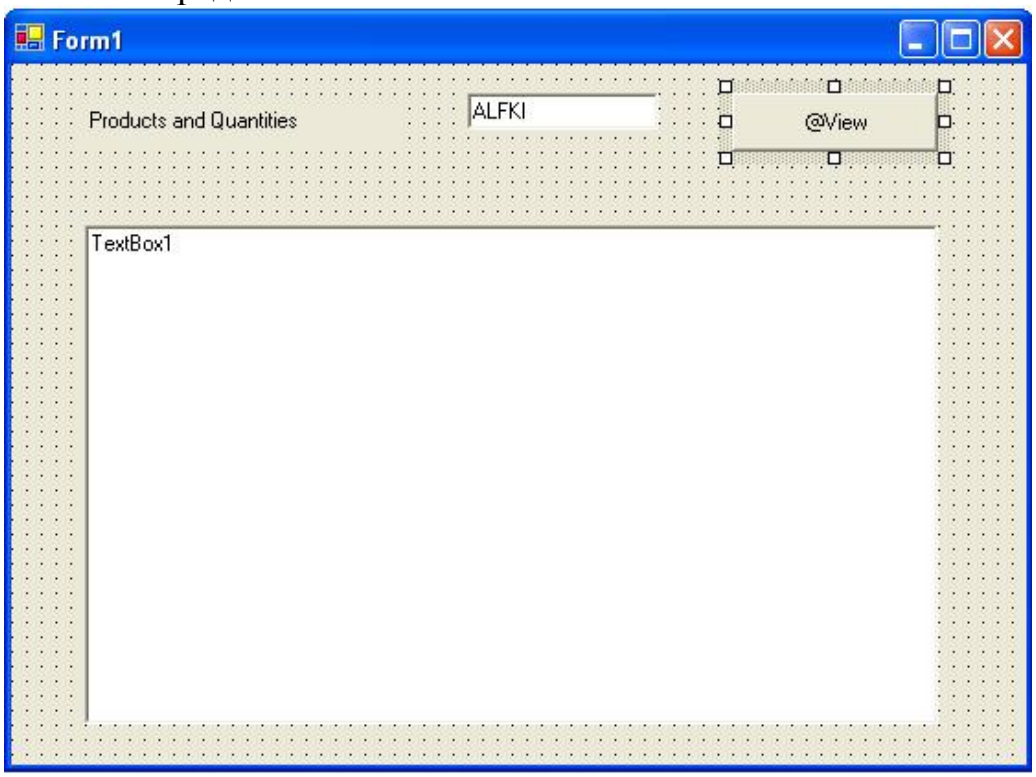


РИС. 2. Форма, в которой для заполнения элемента управления TextBox применяется объект Command с хранимой процедурой

Таблица 2. Значения свойств элементов управления

Объект	Свойство	Значение
Label	Name	Label1
	Caption	Products and Quantities Ordered By:
TextBox	Name	txtCustID
	Text	ALFKI
Button	Name	btnView
	Text	&View
TextBox	Name	txtResults
	MultiLine	True

3. Введите следующий код в обработчик событий Click кнопки btnView. В этом коде создается объект команды с учетом информации о конкретном соединении. В базу данных передается имя хранимой процедуры с указанием типа команды, который представляет собой значение CommandType. StoredProcedure. Затем формируются параметры, и создается объект DataReader. На последнем этапе выполняется обработка данных в цикле и размещение их в текстовых полях, отображаемых на экране.

```
Private Sub btnView_Click(ByVal  sender As
System.Object,
ByVal e As System.EventArgs)   Handles btnView.Click
Dim BuildCnnStr As Array
        Dim ocnn As New
OleDb.OleDbConnection(BuildCnnStr("(local)", "Northwind"))
        Dim ocmdCustHist As New
OleDb.OleDbCommand("CustOrderHist", ocnn)
        Dim odrCustHist As OleDb.OleDbDataReader

        Try
            ' Указать имя хранимой процедуры
    ocmdCustHist.CommandType = CommandType.StoredProcedure
            ' Задать параметры
    cmdCustHist.Parameters.Add("@CustomerID", Me.txtCustID.Text)
            ' Открыть объект соединения
            ocnn.Open()
            ' Создать объект чтения данных
  odrCustHist =
ocmdCustHist.ExecuteReader(CommandBehavior.SequentialAccess)
     ' Обработать в цикле загруженные данные и сформировать
            ' строку результатов
       Do While odrCustHist.Read
```

```
        Me.txtResults.Text &= odrCustHist.GetString(0) & ", " &
odrCustHist.GetInt32(1) & vbCrLf
        Loop
        Catch excpData As Exception
        MessageBox.Show("Error Occurred: " & excpData.Message)
        End Try
End Sub
```

Описание полученных результатов

После указания идентификатора заказчика `CustomerID` и щелчка на кнопке `View` заполняется расположенное ниже текстовое поле с заказами выбранного заказчика