Практическая работа №27

Тема: Создание приложения с БД, хранимые процедуры

Цель работы: изучение механизма работы с базами данных в Visual Studio

Задачи:

– формирование компетенций при построении приложений по обработке информационных массивов с применением элементов управления: DataSet, DataGridView, BindingSourse, BindingNavigator

– создание пользовательского интерфейса и модулей для обработки кнопок ввода, редактирования и поиска данных

Материально-техническое обеспечение:

Место проведения: Компьютерный класс.

Время на выполнение работы: 2 часа.

Оборудование: ПК

Средства обучения: операционная система, текстовый процессор MS Word, программные средства определенного вида Visual Studio

Исходные данные:

1. Конспект занятия.
2. Задание для практической работы.

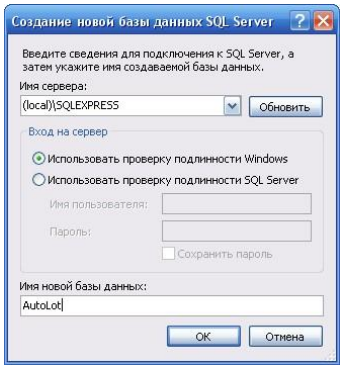
Перечень справочной литературы:

1) Программирование на языке высокого уровня. Программирование на языке С++: учеб. пособие / Т. И. Немцова, С. Ю. Голова, А. И. Терентьев ; под ред. Л. Г. Гагариной. – М. : ИД «ФОРУМ» : ИНФРА-М, 2018. – 512 с. – (Среднее профессиональное образование).

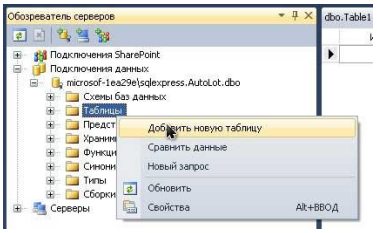
Краткие теоретические сведения:

Создание таблицы inventory

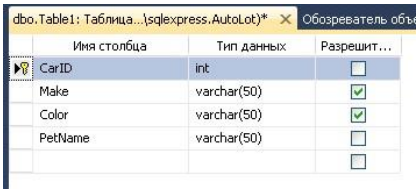
Чтобы приступить к созданию тестовой базы данных, запустите Visual Studio и откройте Server Explorer через меню View (Просмотр). Затем щелкните правой кнопкой мыши на узле Data Connections (Подключения к данным) и выберите в контекстном меню пункт Create New SQL Server Database (Создать новую базу данных SQL Server). В открывшемся диалоговом окне подключитесь к SQL Server, установленному на вашей локальной машине (с именем (local)), и укажите в поле имени базы данных AutoLot. Для наших целей можно оставить аутентификацию Windows:



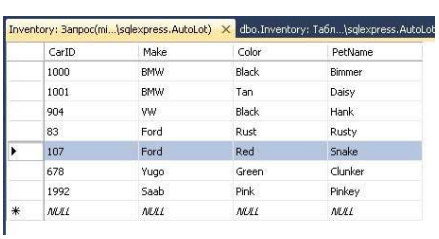
Сейчас база данных AutoLot совершенно пуста и не содержит никаких объектов (таблиц, хранимых процедур и т.п.). Для добавления новой таблицы щелкните правой кнопкой мыши на узле Tables (Таблицы) и выберите в контекстном меню пункт Add New Table:



С помощью редактора таблиц добавьте в таблицу четыре столбца данных: CarID (Идентификатор автомобиля), Make (Модель), Color (Цвет) и PetName (Дружественное имя). У столбца CarID должно быть установлено свойство Primary Key (первичный ключ) — для этого щелкните правой кнопкой мыши на строке CarID и выберите в контекстном меню пункт Set Primary Key (Установить первичный ключ). Окончательные параметры таблицы показаны на рисунке ниже. На панели Column Properties (Свойства столбца) ничего делать не надо, просто запомните типы данных для каждого столбца:



Сохраните и закройте новую таблицу; новый объект базы данных должен иметь имя Inventory. Теперь таблица Inventory должна быть видна под узлом Tables (Таблицы) в Server Explorer. Щелкните правой кнопкой мыши на ее значке и выберите в контекстном меню



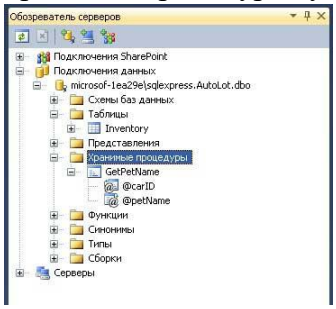
пункт Show Table Data (Просмотр данных таблицы). Введите информацию о нескольких новых автомобилях по своему усмотрению (чтобы было интереснее, пусть у некоторых автомобилей совпадают цвета и модели). Один из возможных вариантов списка товаров приведен на рисунке:

Создание хранимой процедуры GetPetName()

Хранимые процедуры — это подпрограммы, хранимые непосредственно в базе данных; обычно они работают с данными таблиц и возвращают какое-то значение. Мы добавим в базу данных одну хранимую процедуру, которая по идентификатору автомобиля будет возвращать его дружественное имя. Для этого щелкните правой кнопкой мыши на узле Stored Procedures (Хранимые процедуры) базы данных AutoLot в Server Explorer и выберите в контекстном меню пункт Add New Stored Procedure (Добавить новую хранимую процедуру). В появившемся окне редактора введите следующий текст:

```
ALTER PROCEDURE GetPetName
@carID int,
@petName char(10) output
AS
    SELECT @petName = PetName from Inventory
    where CarID = @carID
```

При сохранении, этой процедуре автоматически будет присвоено имя GetPetName, взятое из оператора **CREATE PROCEDURE** (учтите, что при первом сохранении Visual Studio автоматически изменяет имя SQL-сценария на "ALTER PROCEDURE..."). После этого новая хранимая процедура будет видна в Server Explorer:

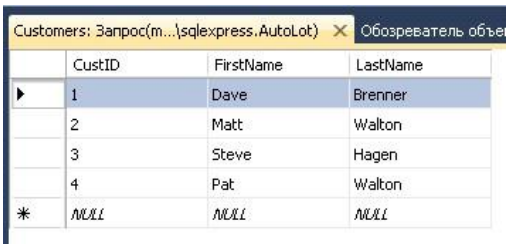


Хранимые процедуры не обязательно должны возвращать данные через выходные параметры, как это сделано здесь.

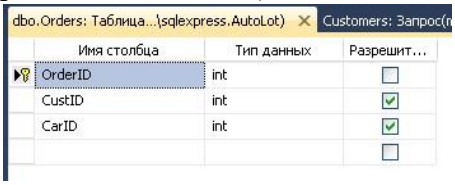
Создание таблиц Customers и Orders

В нашей тестовой базе данных должны быть еще две таблицы: Customers (Клиенты) и Orders (Заказы). Таблица Customers будет содержать список клиентов и состоять из трех столбцов: CustID (Идентификатор клиента; должен быть первичным ключом), FirstName (Имя) и LastName (Фамилия). Повторите шаги, которые были выполнены для создания таблицы Inventory, и создайте таблицу Customers, пользуясь схемой, приведенной на рисунке:

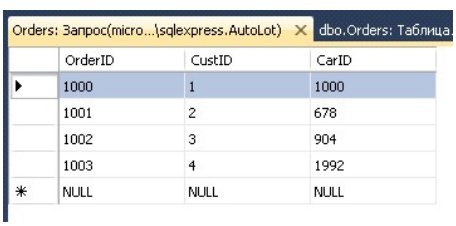
После сохранения этой таблицы добавьте в нее несколько записей:



Последняя наша таблица — Orders — предназначена для связи клиентов и интересующих их автомобилей. Для этого выполняется отображение значений OrderID на CarID/CustID. Ее структура показана ниже (здесь OrderID также является первичным ключом):



Теперь добавьте в таблицу Orders данные. Выберите для каждого значения CustID уникальное значение CarID (предположим, что значения OrderID начинаются с 1000):

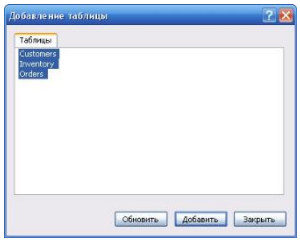


Например, в соответствии с информацией, приведенной на рисунках, видно, что Дэйв Бреннер (Dave Brenner, CustID = 1) мечтает о черном BMW (CarID = 1000), а Пэт Уолтон (Pat Walton, CustID = 4) приглянулся розовый Saab (CarID = 1992).

Визуальное создание отношений между таблицами

И, наконец, между таблицами Customers, Orders и Inventory нужно установить отношения "родительский-дочерний". В Visual Studio это выполняется очень просто, т.к. она позволяет вставить новую диаграмму базы данных на этапе проектирования.

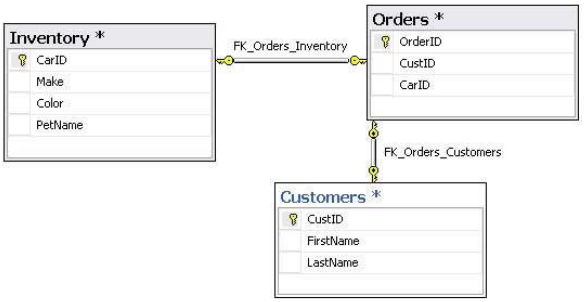
Для этого откройте Server Explorer, щелкните правой кнопкой мыши на узле Database Diagrams базы AutoLot и выберите пункт контекстного меню Add New Diagram (Добавить новое представление). Откроется диалоговое окно, в котором можно выбирать таблицы и добавлять их в диаграмму. Выберите все таблицы из базы данных AutoLot:



или в окне Обозреватель решений дважды щелкните по файлу с названиемБДDataSet.xsd, появится схема данных. Если схема данных отсутствует, то можно создать ее вручную. Для этого из панели Обозреватель серверов выделите таблицы и перенесите их в рабочую область конструирования

Далее чтобы начать устанавливать отношения между таблицами, щелкните на ключе CarID таблицы Inventory, а затем (не отпуская кнопку мыши) перетащите его на поле CarID таблицы Orders. Когда вы отпустите кнопку, появится диалоговое окно; согласитесь со всеми предложенными в нем значениями по умолчанию.

Теперь повторите те же действия для отображения ключа CustID таблицы Customers на поле CustID таблицы Orders. После этого вы увидите диалоговое окно классов, показанное на рисунке ниже (было включено отображение отношений между таблицами за счет щелчка правой кнопкой мыши на конструкторе и выбора в контекстном меню пункта Show Relationship Labels (Показывать метки взаимосвязи)).



Вот и все, база AutoLot полностью готова.

Ход работы:

Требования к содержанию отчета:

- Номер и название практической работы.
- Цель работы.
- По каждому заданию (задаче/примеру) экранные формы (при наличии) и листинг программного кода, показывающие порядок выполнения практической работы, и результаты, полученные в ходе её выполнения.
- Ответы на контрольные вопросы в тетради.

Порядок выполнения работы:

Все проекты практической работы размещать в своей сетевой в новой папке **Пр27_Фамилия.**

В начале каждого файла проекта установить комментарии: пр.р.№_____ (указать номер), свою Фамилию. Формулировку задания

Задание:

1. Создайте базу данных в соответствии с вашим вариантом предметной области и заполните её записями (не менее пяти)
2. Разработайте приложение к своей БД, соответствующее следующим требованиям:
   2.1. на главной форме расположить меню, в котором осуществить вызов модулей. Основное меню должно содержать вкладки: *База данных, Выходные документы, О программе, Выход*
   2.2. при создании интерфейса пользователя реализовать строку статуса, всплывающие подсказки, выполняющий необходимые действия
   2.3. в меню *База данных* реализовать пункт вывода данных в табличной форме и пункт вызова формы для добавления записей. Данные в просматриваемой таблице должны быть понятны пользователю
   2.4. добавление записей таблицы должна быть реализовано в новом окне – форма для добавления с понятной для пользователей структурой. Переход на данную форму должно осуществляться из формы списка (табличная форма) и при выборе соответствующего подпункта меню *База данных*. Для необходимых полей ввода использовать маски ввода, списки.

2.5. пользователь должен иметь возможность отсортировать данные (по возрастанию и убыванию) и отфильтровать данные по выбранному критерию (предусмотреть не менее трех критериев)
2.6. формы должны содержать необходимые элементы управления для просмотра, добавления, удаления и сохранения данных.
2.7. в пункте меню *Выходные документы* должна осуществляться выгрузка табличных данных в файл в Excel. Таблица Excel должна содержать заголовок таблицы - название таблицы, и соответствующие заголовки столбцов.
2.8. в пункте меню *Справка* реализовать вызов модулей подпунктов *Вызов справки* и *О программе* должен осуществляться вывод соответствующей формы

Варианты предметных областей:

14 Аптека. Объекты предметной области: лекарства, показания, поставщики, пациенты. Основные функции: учет заказов лекарств. Выходные документы: журнал заказов
15 Центр занятости. Объекты предметной области: вакансии, резюме, предприятия, организации, повышение квалификации. Выходные документы: сведения о вакансиях.
16 Магазин бытовой техники. Объекты предметной области: категории товаров, условия кредитов, условия доставки. Выходные документы: прайс-лист.
17 Ателье пошива одежды. Объекты предметной области: портные, услуги, материалы. Выходные документы: прайс-лист.
18 Театр. Объекты предметной области: актеры, спектакли. Выходные документы: афиша.
19 Зоопарк. Объекты предметной области: животные, мероприятия. Выходные документы: расписание мероприятий.
20 Станция техобслуживания. Объекты предметной области: оказываемые услуги, обслуживающий персонал. Выходные документы: прайс-лист.
21 Кредитный отдел банка. Основные объекты предметной области: клиенты, типы кредитов, счета, график платежей. Основные функции: учет выплат и начисление процентов по вкладам. Выходные документы: кредитные истории клиентов.
22 Пункт обмена валют. Объекты предметной области: курс валют, клиенты, заказ валют, наличие. Выходные документы: журнал заказа.
23 Отдел кадров. Основные объекты предметной области: должности, отделы, сотрудники, квалификации сотрудников. Основные функции: начисление зарплаты Выходные документы: ведомость сотрудников.
24 Учет компьютеров на предприятии. Основные объекты предметной области: техника, сотрудники, поставщики, ремонт. Выходные документы: ведомость компьютерной техники.
25 Страховая фирма. Основные объекты предметной области: клиенты, объекты и виды страхования, агенты. Основные функции: начисление зарплаты агентам, составление договора страхования. Выходные документы: ведомость страховок.

Приложение 1

Пространство имен System.Windows.Forms содержит классы для создания приложений Windows, которые позволяют наиболее эффективно использовать расширенные возможности пользовательского интерфейса, доступные в операционной системе MicrosoftWindows.

Категория класса	**Подробно**
Элементы управления, пользовательские элементы управления и формы	Большинство классов в пространстве имен System.Windows.Forms являются производными от класса Control. Класс Control предоставляет основные функциональные возможности для всех элементов управления, отображаемых в Form. Класс Form представляет окно в приложении. Оно включает диалоговые окна, немодальные окна, а также клиентские и родительские окна интерфейса MDI. Можно также создать собственные элементы управления путем наследования от класса UserControl.
Меню и панели инструментов	WindowsForms включает широкий набор классов, которые позволяют создавать пользовательские панели инструментов и меню, отличающиеся современным обликом и поведением. ToolStrip, MenuStrip, ContextMenuStrip и StatusStrip позволяют создавать панели инструментов, строки меню, контекстные меню и строки состояния, соответственно.
Элементы управления	Пространство имен System.Windows.Forms предоставляет большое количество классов элементов управления, которые позволяют создавать пользовательские интерфейсы с расширенными возможностями. Некоторые элементы управленияпредназначены для ввода данных в приложении, например, элементы TextBox и ComboBox. Другие элементы управления отображают данные приложений, например, Label и ListView. Это пространство имен также предоставляет элементы управления для вызова команд в приложении, например, Button. Элемент управления WebBrowser и такие классы управляемых HTML-страниц, как HtmlDocument, позволяют отображать HTML- страницы и выполнять с ними определенные действия в области управляемого приложения WindowsForms. Элемент управления MaskedTextBox представляет собой улучшенный элемент управления вводом данных, который позволяет создавать маску для принятия или отклонения введенных пользователем данных в автоматическом режиме. Кроме того, с помощью элемента управления PropertyGrid можно создать собственный конструктор WindowsForms, отображающий видимые конструктором свойства
Макет	Несколько важных классов в WindowsForms помогают контролировать расположение элементов управления на отображаемой поверхности, например, в форме или элементе управления. FlowLayoutPanel располагает все элементы управления, которые содержит в последовательном режиме, а TableLayoutPanel позволяет определять ячейки и строчки для расположения элементов управления в фиксированной сетке. SplitContainer разделяет поверхность отображения на две или более корректируемых части.
Данные и привязка данных	WindowsForms обеспечивает расширенную архитектуру для привязывания к таким источникам данных, как базы данных и XML-файлы. Элемент управления DataGridView предоставляет настраиваемую таблицу для отображения данных и позволяет настраивать формат ячеек, строк, столбцов и границ. Элемент управления BindingNavigator представляет стандартный способ навигации и работы с данными в форме; BindingNavigator часто используется в сочетании с элементом управления BindingSource для перемещения от одной записи к другой в форме, а также для выполнения операций с записями
Компоненты	Помимо элементов управления пространство имен System.Windows.Forms предоставляет другие классы, которые не являются производными от класса Control, но также обеспечивают визуальные функции для приложений Windows. Такие классы, как ToolTip и ErrorProvider, расширяют возможности или предоставляют сведения пользователям. Классы Help и HelpProvider позволяют отображать текст справки для пользователя, который работает с приложениями.

Общие диалоговые окна	Windows предоставляет несколько основных диалоговых окон, позволяющих обеспечить единообразие пользовательского интерфейса в приложениях Windows при выполнении таких операций как открытие и сохранение файлов, задание цвета шрифта или текста и печать. Классы OpenFileDialog и SaveFileDialog предоставляют возможность отображения диалогового окна, в котором пользователь может выполнить поиск файла, а также ввести имя файла, который необходимо открыть или сохранить. Класс FontDialog отображает диалоговое окно для изменения элементов Font, используемого приложением. Классы PageSetupDialog, PrintPreviewDialog и PrintDialog отображают диалоговые окна, позволяющие пользователю управлять параметрами

Элемент управления MasketTextBox использует маску, чтобы отличить допустимые данные, вводимые пользователем, от недопустимых.

СвойствоMask используется по умолчанию для класса MaskedTextBox. Маска **Mask** должна быть строкой, состоящей из одного или нескольких элементов маски, как показано в следующей таблице. Язык маски, используемый в элементе управления MaskedTextBox, определяется сопоставленным с ним объектом MaskedTextProvider.

Maskingelement	**Описание**
0	Цифра, необходимый символ. Данный элемент позволяет вводить любое число от 0 до 9
9	Цифра или пробел, необязательный символ.
#	Цифра или пробел, необязательный символ. Если эта позиция в маске пуста, в свойстве Text она будет представлена пробелом. Допускаются знаки плюса (+) и минуса (-).
L	Буква, необходимый символ. Ограничивает вводимые данные наборами букв ASCII от a до z и от A до Z. Этот элемент маски аналогичен регулярному выражению [a- zA-Z].
?	Буква, необязательный символ. Ограничивает вводимые данные наборами букв ASCII от a до z и от A до Z. Этот элемент маски эквивалентен выражению [a-zA-Z]? inregularexpressions.
&	Символ, необходимый. Если свойство AsciiOnly имеет значение true, этот элемент действует аналогично элементу "L".
C	Символ, необязательный. Любой неуправляющий символ. Если свойство AsciiOnly имеет значение true, этот элемент действует аналогично элементу "?".
a[x]	Буква или цифра, необходимый символ. Если свойство AsciiOnly имеет значение **true**, будут приниматься только буквы ASCII от a до z и от A до Z. Этот элемент маски действует аналогично элементу "A".
a[x]	Буква или цифра, необязательный символ. Если свойство AsciiOnly имеет значение **true**, будут приниматься только буквы ASCII от a до z и от A до Z. Этот элемент маски действует аналогично элементу "A".
.	Местозаполнитель для десятичного разделителя. Символ десятичного разделителя, который будет фактически отображаться, соответствует поставщику форматирования, определяемому свойством FormatProvider элемента управления.
,	Местозаполнитель для разделителя групп разрядов. Символ разделителя групп разрядов, который будет фактически отображаться, соответствует поставщику
	форматирования, определяемому свойством FormatProvider элемента управления.
:	Timeseparator. Символ разделителя компонентов времени, который будет фактически отображаться, соответствует поставщику форматирования, определяемому свойством FormatProvider элемента управления.
/	Dateseparator. Символ разделителя компонентов даты, который будет фактически отображаться, соответствует поставщику форматирования, определяемому свойством FormatProvider элемента управления.

$	Символ денежной единицы. Символ денежной единицы, который будет фактически отображаться, соответствует поставщику форматирования, определяемому свойством FormatProvider элемента управления.
<	Переключение в нижний регистр. Преобразует все последующие символы в нижний регистр.
>	Переключение в верхний регистр. Преобразует все последующие символы в верхний регистр.
|	Отмена предыдущего изменения регистра.
\	Escape-знак. Заменяет символ маски на литерал. " \\" — escape-последовательность для обратной косой черты.
Allothercharacters	Literals. Все элементы, не образующие маску, отображаются в элементе MaskedTextBox без изменений. Литералы всегда занимают статическую позицию в маске во время выполнения, и пользователь не может перемещать или удалять их.

Если изменить маску в момент, когда элемент управления MaskedTextBox содержит введённые данные, отфильтрованные предыдущей маской, элемент управления MaskedTextBox выполнит попытку переноса этих данных в новое определение маски.

В случае неудачи существующие данные будут сброшены. Если задать в качестве маски строку нулевой длины, будут сохранены все существующие данные в элементе управления. Элемент управления MaskedTextBox с маской нулевой длины работает аналогично однострочному элементу управления TextBox.

Десятичный разделитель (.), разделитель групп разрядов (,), разделитель компонентов времени (:), разделитель компонентов даты (/) и символы денежной единицы ($) по умолчанию отображаются в соответствии с языком и региональными параметрами приложения. Можно принудительно отображать символы для других региональных параметров, используя свойство FormatProvider.

Вставка символов в маску во время выполнения управляется свойством InsertKeyMode. Пользователи могут перемещаться по маске, используя клавиши со стрелками влево и вправо или курсор мыши, а ввод пробела позволяет пропустить необязательные позиции в маске.

В следующей таблице приведены примеры масок.

Mask	**Назначение**
00/00/0000	Дата (день, месяц в числовом формате, год) в международном формате. Знак "/" в маске является логическим разделителем компонентов даты. Пользователю будет отображаться разделитель компонентов даты, соответствующий текущему языку и региональным параметрам приложения.
00->L<LL-0000	Дата (день, сокращенное наименование месяца и год) в формате США, в котором трехбуквенное сокращенное наименование месяца отображается с первой буквой в верхнем регистре и двумя следующими — в нижнем.
(999)-000-0000	Номер телефона в США и код города (необязательно). Чтобы не вводить необязательные символы, пользователь может вводить вместо них пробелы или переместить указатель мыши в позицию маски, представленную первой цифрой 0.
$999,999.00	Значение денежных единиц лежит в диапазоне от 0 до 999999. Символы денежной единицы, разделителя групп разрядов и десятичного разделителя будут заменены во время выполнения на эквиваленты, соответствующие текущему языку и региональным параметрам.