# *Тема 6. Технология работы с данными в среде Visual Studio .NET*

[6.1. Создание приложений для обработки данных в среде Visual Studio .NET](#_5.6.1._Создание_приложений)

[6.2. Основные понятия модели доступа к данным – ADO .NET](#_5.6.2._Основные_понятия)

[6.3. Пример разработки приложения для работы с базами данных СУБД Access](#_5.6.3._Примеры_разработки)

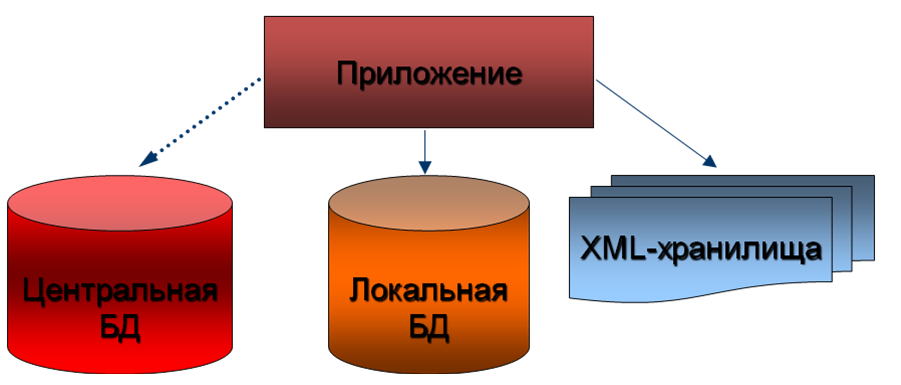
6.4. Задачи для самостоятельного решения

[6.5.](#_5.3.6._Тестовые_задания)  Лабораторная работа

6.6. Контрольные вопросы

## 6.1. Создание приложений для обработки данных в среде Visual Studio .NET

С самого своего рождения программирование решало задачи обработки данных, поэтому практически во всех приложениях данные в том или ином виде хранятся в некоторых хранилищах, а сами приложения предоставляют способы просмотра, редактирования, обновления и использования этих данных (рис. 6.1-1).



*Рис. 6.1-1. Хранилища данных*

Visual Studio .NET предоставляет множество средств разработки приложений, которые призваны помочь пользователю в процессе создания приложений, получающих доступ к данным. Кратко рассмотрим основные процессы, которые задействованы при создании приложений, работающих с данными.

При разработке приложений в VS, работающих с данными, к ним могут предъявляться различные требования. В некоторых случаях требуется только отобразить данные в форме, а в других может потребоваться разработать способ обмена информацией с другими приложениями или процессами.

Основной архитектурой данных, которая используется для создания приложений в Visual Studio .NET, является **ADO.NET**. Для всех операций, выполняемых с данными, существуют некоторые основные понятия, которые следует знать. Возможно, никогда не потребуется узнать некоторые детали обработки данных – например, может не потребоваться программно создавать базу данных. В то же время, очень полезно представлять структуру **ADO.NET**, так же как и использовать средства для работы с данными (мастера и конструкторы), доступные в VS.

Общие задачи по работе с данными в приложении можно разделить на несколько процессов верхнего уровня. Так, например, прежде чем отобразить данные пользователю на форме, необходимо сначала подключиться к источнику данных (к базе данных или к службе, предоставляющей данные), а затем выбрать данные, которые требуется отобразить. После импорта этих данных в приложение, могут потребоваться средства для их временного хранения.

В **клиент-серверных** приложениях традиционно используется технология доступа к источнику данных,  в которых ***соединение с базой поддерживается постоянно***. Однако после широкого распространения приложений, ориентированных на **Web**-приложения, выявились некоторые недостатки такого подхода.

Так, например, соединения с базой данных требуют выделения системных ресурсов, что может быть критично при большой нагрузке сервера. Хотя постоянное соединение позволяет несколько ускорить работу приложения, общий убыток от растраты системных ресурсов сводит преимущество к нулю.

Особенность  **Web**-приложений не позволяет серверу в каждый момент времени знать, что необходимо пользователю. То есть, до следующего запроса сервер не имеет представления, нужно ли еще поддерживать соединение.

Опыт разработчиков показал, что приложения, постоянно соединенные с источником данных, чрезвычайно трудно поддаются масштабированию. Хотя существуют и другие недостатки, приведенные выше наиболее существенны. Все эти проблемы порождаются постоянным соединением с базой данных и решаются в **ADO .NET** следующим образом:

* Во-первых, используется модель доступа, в которой соединение устанавливается лишь на то короткое время, когда необходимо проводить операции над базой данных. Такая модель работы с данными называется модель с ***отсоединенными источниками данных.***
* Во-вторых, для обмена текстовыми данными используется формат **XHL**.

Следует признать, что новая технология иногда все же проигрывает традиционной. Примерами таких приложений служат программы, проводящие частые и объемные изменения содержания записей базы данных.

Обычные приложения, работающие с данными, используют большинство из процессов (задач), показанных на рис. 6.1-2 [1].

Следует помнить, что вспомогательные программы (**Мастера** и **Конструкторы**) помогают упростить процессы работы с данными, отображенные на рис. 6.1-2. Например, запуск *Мастера настройки источника данных* предоставляет приложению достаточно сведений для подключения к данным, для создания типизированного набора данных, для получения данных и для переноса данных в приложение.

|  |
| --- |
| **Цикл данных** График цикла данных |

*Рис. 6.1-2. Процессы приложений, работающих с данными*

Кратко рассмотрим назначение процессов из диаграммы, приведенной на рис. 6.1-2.

***Подключение к данным***. Чтобы перенести данные в приложение (и отправить изменения обратно в источник данных), необходимо установить двустороннее соединение. Этот двусторонний обмен данными обычно обрабатывается подключением TableAdapter в приложениях, использующих наборы данных. VS предоставляет несколько средств для упрощения создания подключений, которые могут быть использованы в приложении.

***Подготовка приложения к получению данных***. Если приложение использует модельс ***отсоединенными источниками данных,*** необходимо временно где-то хранить данные в приложении во время работы с ним. VS .NET предоставляет средства, помогающие создавать объекты, которые приложение использует для временного хранения данных. Необходимо обратить внимание, что приложение, использующее модельс ***отсоединенными источниками данных*,** подключается к базе данных обычным образом, выполняет запрос, передающий данные в приложение, отключается от базы данных, а затем оперирует данными в автономном режиме до повторного подключения и обновления базы данных.

http://i.msdn.microsoft.com/Global/Images/clear.gif ***Выборка данных в приложение.*** Вне зависимости от того, использует ли приложение модель ***с*** ***отсоединенными источниками данных***, необходимо обеспечить загрузку данных в приложение. Обычно данные в приложение загружаются путем выполнения запроса над базой данных. Приложения, хранящие данные в наборах данных, выполняют запросы с помощью TableAdapter.

***Отображение данных на форме в приложении Windows.*** После переноса данных в приложение их обычно отображают на форму для просмотра или изменения. VS .NET предоставляет окно **Источники данных**, откуда можно перетащить элементы данных на форму, чтобы автоматически создать элементы управления с привязкой к данным, отображающие эти элементы данных.

***Редактирование данных в приложении.*** После того, как пользователям стали доступны данные, они могут изменять их путем добавления новых, редактирования и удаления существующих записей до их передачи обратно в базу данных. Эти изменения обычно выполняются через управление отдельными объектами, составляющими таблицы в наборе данных. Необходимо обратить внимание, что при перетаскивании элементов из окна   
**Источники данных** на форму, большинство изменений данных автоматически обрабатываются через элементы управления с привязкой к данным. При изменении значений в элементе управления, новые значения автоматически записываются в набор данных.

***Проверка данных***. Обычно при внесении изменений в данные необходимо проверить изменения перед разрешением внесения значений в набор данных или в базу данных. Проверка – это процесс, проверяющий эти новые значения на соответствие требованиям приложения. Для проверки значений в приложении по мере их изменения пользователь может добавить соответствующую логику. VS .NET предоставляет средства, помогающие добавлять программный код, проверяющий данные во время изменения значений столбцов и строк.

***Сохранение данных.*** После внесения изменений в приложении (и проверки этих изменений), обычно требуется отправить изменения обратно в базу данных.

## 6.2. Основные понятия модели доступа

## к данным – ADO.NET

**ADO .NET**  (ActiveX Data Objects .NET) – технология, предоставляющая доступ к [данным](https://ru.wikipedia.org/wiki/%D0%94%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5) для приложений, основанных на [Microsoft .NET](https://ru.wikipedia.org/wiki/Microsoft_.NET) (<https://ru.wikipedia.org/wiki/ADO.NET>), иначе - модель доступа приложений **.NET** к данным. В то же время, **ADO.NET** представляет собой набор библиотек классов, входящих в Microsoft .NET Framework**,** и предназначенных для взаимодействия с различными хранилищами данных из **.NET** приложений. Библиотеки **ADO.NET** включают все необходимые классы для подключения к источникам данных практически любого формата, выполнения запросов к этим источникам и получения результата. Кроме того, несомненным достоинством **ADO.NET** является возможность работы с ***отсоединенными источниками данных***, представляющих собой структуры, организующие данные в оперативной памяти компьютера. Таким образом, **ADO .NET** можно использовать в качестве надежного, иерархически организованного отсоединенного от источника элемента данных для автономной работы, что незаменимо при построении масштабируемых приложений, особенно ориентированных на **Web.**

На сегодняшний день, **ADO .NET** является наиболее развитой технологией доступа к данным.

Так как **ADO .NET** представляет собой библиотеку классов для организации взаимодействия клиентского приложения с базой данных, рассмотрим более подробно объектную модель **ADO .NET**.

На показаны классы, составляющие объектную модель **ADO.NET.**

|  |
| --- |
| **Connection**  DataSet  **Transaction**  **DataAdapter**  **Command**  **Parameter**  **DataReader**  **DataTable**  **DataRow**  **DataColumn**  **Constraint**  **DataRelation**  **DataView** |

Рис. 6.2-1. Иерархия классов объектной модели ADO.NET

Объекты, расположенные в левой части называются подсоединенными и необходимы для управления соединением, транзакциями, выборкой данных и передачей изменений данных в БД. Они непосредственно взаимодействуют с базой данных. Объекты, расположенные в правой части называются отсоединенными. Они позволяют работать с данными автономно. То есть, отсоединенные объекты не взаимодействуют непосредственно с данными из БД. Точнее, можно сказать, что они не взаимодействуют со всеми объектами левой части иерархии объектов. И, хотя вопросы заполнения данными отсоединенных объектов, будут обсуждаться позднее, можно сказать, что одной из форм взаимодействия этих двух групп является использованием объектом DataAdapter объекта DataSet для заполнения последнего набором данных, извлеченных непосредственно из БД.

Итак, структура объектной модели ADO.NET базируется на двух различных группах объектах:

* на объектах для хранения данных **DataSet** (справа на рис. 6.2-1);
* на объектах для доступа к данным – **поставщика данных** (слева на рис. 6.2-1).

Обратите внимание, что объект **DataSet** и наследуемые от него объекты (**DataTable**, **DataRow** и **DataColumn**) обеспечивают хранение и просмотр данных в формате XML-документа. Причем данные, представленные в объекте **DataSet,** организованы так же, как и в связанной с ним базе данных, в виде таблиц – объектов класса **DataSet**. Именно в объект **DataSet** библиотеки классов ADO загружаются данные из БД, после чего соединение с БД-источником может быть разорвано.

Одной из основных идей, лежащих в основе ADO.NET является наличие ***поставщиков данных***.

***Поставщик данных*** – это набор классов, предназначенных для взаимодействия с хранилищем данных определенного типа. За счет этого, модель ADO.NET является чрезвычайно гибкой и расширяемой. Рассмотрим уровни модели поставщиков ADO.NET ().

Как видно на рис. 6.2-2, приложение .NET взаимодействует с базой данных посредством поставщиков данных. Каждый поставщик данных может обеспечивать доступ только к базе данных определенного формата. Так, для доступа к БДMicrosoft SQL Server используется поставщик SQL Server для .NET, для доступа к БДOracle – поставщик Oracle для .NET и т.д. Названные базы данных являются одними из самых распространенных во всем мире, поэтому поставщики данных для них выделены в отдельные элементы модели ADO.NET, тем не менее, существует множество баз данных другого формата, к которым необходимо осуществлять доступ из приложения .NET. Для этого может применяться поставщик данных **OleDb** для .NET или **ODBC** для .NET, которые обеспечивают доступ к любым данным, для которых существует драйвер **OleDb**, либо **ODBC** соответственно.

Приложение .NET

Поставщик SQL Server для .NET

Поставщик OleDb (Access) для .NET

Поставщик Oracle для .NET

Поставщик OleDb

База данных SQL Server

Источник данных

**База** данных Oracle

Рис. 6.2-2. Уровни моделей поставщиков ADO.NET

При построении приложения, использующего доступ к данным, необходимо сначала попытаться найти «родного» поставщика данных для **.**NET. Если такового не существует, можно воспользоваться **OleDb**, если существует соответствующий драйвер для источника данных, к которому устанавливается подключение. Так как технология **OleDb** существует достаточно давно, то существует много драйверов для различных источников данных, поддерживающих ее. Если в системе не установлен соответствующий драйвер **OleDb**, его можно попытаться найти на сайте разработчика той базы данных, к которой осуществляется доступ, либо на сайте **Microsoft**. То же самое справедливо и для технологии **ODBC**. В ряде случаев, существует несколько альтернативных вариантов организации доступа к данным определенного формата. Например, доступ к источнику данных на основе **SQL Server** можно организовать, используя либо поставщика данных **SQL Server** для .NET, либо поставщика данных **OleDb**. Тем не менее, всегда предпочтительнее использовать тот поставщик данных, который специально предназначен для обеспечения доступа к данному источнику данных, так как он учитывает его особенности.

Каждый поставщик **.NET** реализует одинаковые базовые классы – **Connection, Transaction, DataAdapter, Command, Parameter, DataReader** имена которых зависят от поставщика. Например, у поставщика **SQL Server** существует объект **SqlDataAdapter**, у поставщика **OleDb – OleDbDataAdapter** и т.д.

У каждого поставщика данных существует собственное пространство имен. Хотя все поставщики относятся к пространству имен **System.Data**, каждый из них содержит свой подраздел этого пространства, который содержит объекты, специфичные для данного поставщика. Например, объект **SqlDataAdapter** находится в пространстве имен **System.Data.SqlClient**.

Все поставщики данных .NET реализуют одинаковые базовые функции, поэтому программный код, создаваемый для доступа к данным, выглядит приблизительно одинаково независимо от поставщика. Это означает, что, рассматривая использование интерфейсов отдельного поставщика, мы фактически рассматриваем возможность использования этих же интерфейсов применительно и к другим поставщикам данных.

При организации доступа к данным с помощью ADO.NET исключительно важную роль играют объекты, изображенные на рис. 6.2-2. Рассмотрим их назначение.

Объект **Connection** представляет соединение с источником данных. С его помощью можно задать тип источника данных, его местонахождения, параметры доступа и ряд других атрибутов. Перед тем, как начать взаимодействие с источником данных, необходимо установить подключение к нему с помощью объекта **Connection**.

Объект **Transaction** позволяет осуществлять группировку записей в логическую единицу работы, называемой транзакцией. Можно сказать, что транзакция – это набор команд, которые должны выполняться обязательно от начала до конца. Транзакция логически объединяет несколько различных действий, связанных с манипулированием данными в единое целое. В процессе выполнения действий, осуществляемых в рамках транзакции, СУБД обычно кэширует изменения, вносимые этими действиями в данные до момента завершения транзакции. Это позволяет производить отмену любых изменений, выполненных в рамках транзакции в случае, если хотя бы одно из действий транзакции завершилось неудачно.

Объект **Command** представляет запрос к источнику данных или прямой запрос на возврат содержимого конкретной таблицы. Как известно, существует несколько типов запросов. Часть из них возвращают данные, извлекаемые из источника данных, другие – изменяют записи, третьи – управляют структурой БД. С помощью объекта **Command** возможно выполнить любой из перечисленных типов запросов. Различия в поведении объекта **Command** начинают проявляться тогда, когда необходимо исполнить тот или иной запрос. Так, например, при необходимости исполнения запроса, не возвращающего записи, необходимо использовать метод **ExecuteNonQuery** объекта **Command.** При извлечении данных из БД – метод **ExecuteReader**, который в свою очередь возвращает объект **DataReader,** позволяющий просматривать полученные в результате запроса записи.

Объект **Parameter** позволяет вводить в запрос элемент, значение которого может быть задано непосредственно перед исполнением запроса. За счет этого отпадает необходимость каждый раз изменять текст самого запроса.

Объект **DataReader** предназначен для максимально быстрой выборки и просмотра возвращаемых запросом записей. Однако, он позволяет просматривать весь результирующий набор записей путем перемещения от одной записи к другой, и, таким образом, просматривать только одну запись за раз. Кроме того, **DataReader** не имеет возможности для обновления значений записей, вследствие чего имеет возможность работать в режиме только для чтения, за счет чего обладает высокой производительностью.

Объект **DataAdapter** представляет собой связующее звено между отсоединенными объектами **ADO.NET** и базой данных и используется для передачи данных между источником данных и объектом **DataSet**. С его помощью осуществляется заполнение объекта как **DataSet** значениями, полученными в результате выполнения запроса к базе данных для последующей автономной работы с ними. Помимо этого, **DataAdapter** реализует эффективный механизм выполнения обновления данных, хранимых в базе данных изменениями, внесенными в данные объекта **DataSet**.

Объект **DataSet** хранит в себе набор таблиц с дополнительной информацией об их структуре и отношениями межу ними, то есть представляет собой отсоединенный набор данных, который может рассматриваться как контейнер для объектов **DataTable**. Объект **DataSet** позволяет организовывать внутри себя ***структуру, полностью соответствующую реальной структуре таблиц и связей между ними в БД***. Это удобно в том случае, когда при работе с базой данных необходимы данные из разных таблиц. В этом случае, вместо того, чтобы многократно обращаться к серверу и выбирать данные из одной таблицы за раз, можно поместить все данные в объект **DataSet**, а затем передать его клиентскому приложению. Объект **DataSet** является очень мощным инструментом для работы с отсоединенными наборами данных. Все изменения, вносимые в данные, которые хранятся в **DataSet**, кэшируются в объектах **DataRow**. Когда возникает необходимость передачи изменений из **DataSet** в БД**,** возможно, осуществить передачу только измененных данных, вместо того, чтобы передавать все данные объекта, что значительно снижает объем данных, передаваемых между клиентским компьютером и сервером.

Объект **DataTable** позволяет просматривать данные в виде наборов записей (**DataRow**) и столбцов (**DataColumn**). Фактически, он представляет собой аналог таблицы БД**,** размещенный в памяти. Достоинством такой организации является возможность автономной работы с данными. Это означает, что после установления соединения с базой данных, чтения данных и заполнения ими объекта **DataTable** можно отключиться от источника данных и продолжать работать с ним в автономном режиме. Такая возможность чрезвычайно полезна при организации Web**-**приложений, которые должны быть хорошо масштабируемыми и ориентированы на многопользовательский режим работы. При этом, возникают и побочные эффекты, один из которых связан с тем, что при работе с автономным набором данных нельзя увидеть изменений, вносимых в данные в этот момент другими пользователями. Объект **DataSet** может включать в себя несколько объектов **DataTable**, каждый из которых может представлять логическую таблицу данных.

Объект **DataColumn** представляет собой столбец объекта **DataTable**. Набор же всех столбов объекта **DataTable** представляет собой коллекцию **Columns.** Посредством этого объекта можно получить доступ к значению ячейки соответствующего столбца.

Объект **DataRow** представляет собой строку объекта **DataTable.** Набор всех строк этого объекта представляет собой коллекцию **Rows**. Объект **DataRow** очень часто используется для доступа к значению конкретного поля определенной записи. При этом применяется свойство Item.

Объект **DataRelation** представляет собой описание связей между таблицами реляционной базы данных. Он предоставляется объектом **DataSet** и позволяет организовывать взаимосвязи между таблицами отсоединенного набора данных объекта **DataSet**. Объект **DataRelation** выполняет функцию, аналогичную той, которую выполняют связи, определяемые в СУБД между таблицами при создании структур данных. Это касается и соблюдения принципов ссылочной целостности информации. Например, **DataRelation** можно настроить таким образом, чтобы изменения значения первичного ключа родительской таблицы автоматически передавались (каскадировались) дочерним записям, а при удалении записи в родительской таблице, автоматически удалялись записи в дочерних таблицах, связанных с ней.

Объект **DataView** предназначен для организации возможности просмотра содержимого **DataTable** различными способами. Это относится к таким операциям, как сортировка и фильтрация записей. С помощью **DataView** допускается просматривать содержимое одного объекта **DataTable** с различными установками фильтрации и сортировки. Для этого необходимо использовать два различных объекта, связанных с одним объектом **DataTable**. Такая возможность исключает необходимость хранения одного набора данных в двух разных структурах.

Использования объектов модели ADO.NET рассмотрим на примерах.

## 6.3. Пример разработки приложения для работы с базами данных СУБД Access

Одним из наиболее распространенных сценариев в разработке приложений является отображение данных из БД на форме **Приложения** **Windows Forms**. Поэтому с учетом процессов, которые используются при работе с наборами данных, предлагаемых Microsoft **(**рис. 6.1-2**),** и с учетом того, что представленные задачи учебные, в примерах будем рассматривать следующие процессы:

1. Создание БД.
2. Разработка приложения Windows:
   1. Создание проекта.
   2. Создание интерфейса пользователей.
   3. Создание подключения к данным БД и отображение их на форме.
   4. Редактирование, проверка и сохранение данных в приложении.
   5. Разработка программного кода приложения.
3. Выполнение приложения.

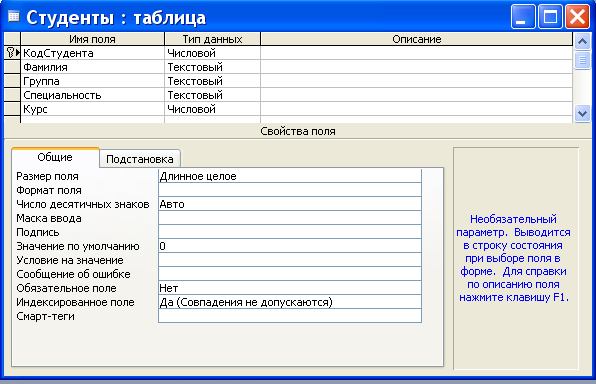
**Пример 6.3-1. Разработать проект ОценкиСтудентов для работы с БД DataBase2.**

**В базу данных c именем DataBase2 поступают сведения о студентах и оценках, полученные этими студентами по трем предметам на двух контрольных. Предположим, что данные хранятся в двух таблицах, однаиз которых называетсяСТУДЕНТЫ, а другая ОЦЕНКИ. Проект должен реализовать расчет сведений о тех студентах, которые получили пятерок на второй контрольной больше, чем на первой.**

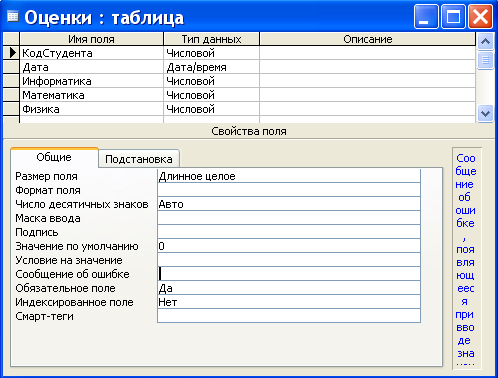
**Документ должен иметь заголовок, информацию о времени его создания, даты проведения контрольных, а также информацию обо всех студентах, повысивших успеваемость, с указанием групп и специальностей*.***

1. **Создание базы данных DataBase2.**

Для построения таблиц **СТУДЕНТЫ** и **ОЦЕНКИ** необходимо создать их структуры, вид которых приведен на рис. 6.3-1 и рис. 6.3-2.

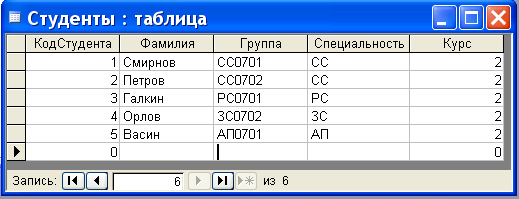


*Рис. 6.3-1. Структура таблицы* ***СТУДЕНТЫ***

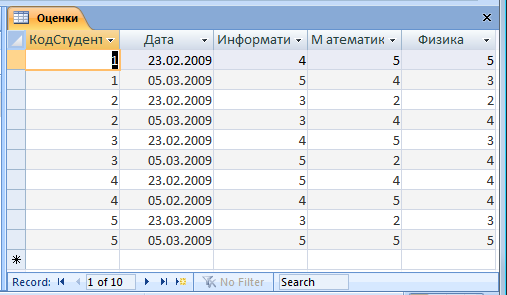


*Рис. 6.3-2. Структура таблицы* ***ОЦЕНКИ***

Заполнение таблиц данными можно выполнить в режиме таблицы. После заполнения таблиц **СТУДЕНТЫ** и **ОЦЕНКИ** они могут принимать вид, приведенный на рис. 6.3-3-6.3-4.

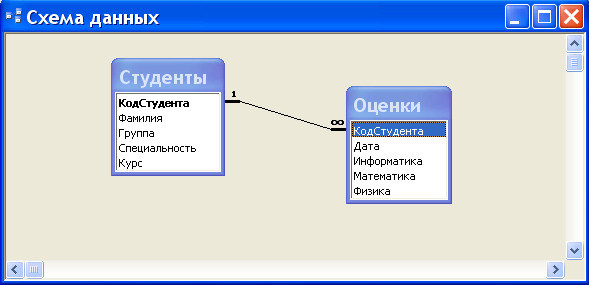


*Рис.6.3-3. Данные таблицы* ***СТУДЕНТЫ***



*Рис.6.3-4. Данные таблицы* ***ОЦЕНКИ***

Таблицы базы данных будут связаны по схеме «один ко многим» как на рис.6.3-5.



*Рис. 6.3-5. Схема данных базы* ***DataBase2***

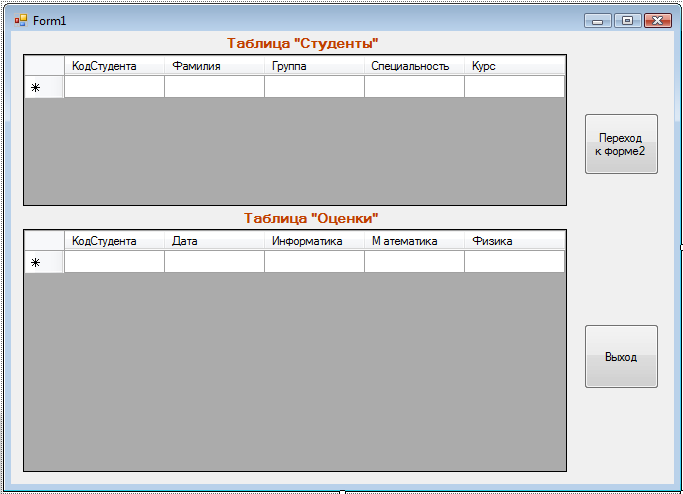
**Разработка Приложения Windows для работы с** БД **DataBase2.**

* 1. **Создание нового проекта.**

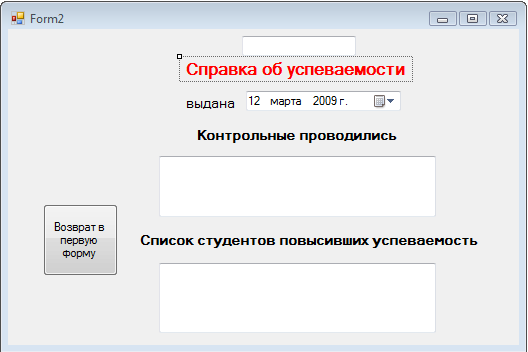
Создать проект **Оценки студентов**.

* 1. **Создание интерфейса пользователей.**

Создать интерфейс пользователей из двух форм **Form1** и **Form2**, которые изображены на рис. 6.3-6 и рис. 6.3-7.



*Рис. 6.3-6. Форма* ***Form1*** *проекта* ***Оценки студентов***



*Рис. 6.3-7. Форма* ***Form2*** *проекта* ***Оценки студентов***

* 1. **Подключение набора данных приложения к базе данных**

**DataBase2 СУБД Access и отображение их на форме Form1.**

Процедура подключения приложения VS к данным БД **DataBase2** может быть упрощена за счет использования[**Мастера Настройки источников данных**](http://msdn.microsoft.com/ru-ru/library/w4dd7z6t(printer).aspx). После завершения работы **Мастера** данные в окне **Источники данных** становятся доступными для перетаскивания на форму.

Для создания набора данных при помощи **Мастера настройки источника данных** необходимо:

* В элементе Главного меню **Проект** выполнить команду ***Добавить новый источник данных***.

В появившемся окне **Мастер настройки источника данных**, выбрать **Источник данных для приложения** – **База данных** и нажать кнопку **Далее.**

* В окне **Мастер настройки источника данных** выбрать тип источника – **Набор данных** и нажать **Далее**.

Мастер отобразит форму – **Выбор подключения базы данных**, которая помогает установить соединение с базой данных, создав связывающую строку.

* Нажать **Создать подключение**.
* В появившемся окне **Добавить подключение** ввести путь к файлу базы данных или щелкнуть кнопку **Обзор**, чтобы найти этот файл.
* Нажать кнопку ***Проверить подключение*.**

**Visual Studio** попробует открыть указанный файл базы данных, используя связывающую строку, созданную мастером. Если база данных имеет правильный формат, то будет выдано сообщение ***Проверка подключения выполнена***.

* Закрыть окно сообщения, а затем закрыть окно **Добавить подключение**. Мастер отобразит предупреждение, указывающее, что была выбрана новая локальная база данных, и спросит, следует ли скопировать базу данных в папку проекта. Это сообщение появляется только при установке первого соединения с файлом локальной базы данных. При повторном выполнении этого упражнения данного сообщения не будет. Чтобы избежать создания дополнительной копии базы данных, щелкните на **Нет**.
* В **Мастере настройки источника данных** включить опцию **Да**, сохранить подключение и нажать кнопку **Далее**.

Будет выдан запрос на выбор подмножества объектов базы данных, которые будут использоваться в данном проекте.

* Выберите соответствующие таблицы.

Чтобы добавить функциональность в приложение, то есть отобразить таблицы базы данных на форме необходимо в окне **Источники данных** выделить таблицу **СТУДЕНТЫ** и перетащить ее на форму. Становятся видны названия полей. Это происходит благодаря тому, что подключается элемент управления **DataGridView,** позволяющий иметь на форме элемент в виде таблицы с заголовками, делая тем самым удобным зрительное восприятие таблицы БД, а также навигационную панель (вверху формы) **BindingNavigator**, имеющую кнопки для перемещения по записям, добавления, удаления и сохранения записей в наборе **DataSet**.

Для отображения на форме таблицы **ОЦЕНКИ** необходимо проделать аналогичные шаги:

* Первая форма будет иметь вид как на рис. 6.3-6.
* Перетаскивание элементов из окна **Источники данных** в форму автоматически добавляет нужный код для заполнения набора данных в обработчик событий **Form1\_Load()** (рис. 6.3-8).

|  |
| --- |
| **Private Sub Form1\_Load(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) \_**  **Handles MyBase.Load**  **'TODO: Данная строка кода позволяет загрузить данные**  **'в таблицу "Database2DataSet.Оценки".**  **'При необходимости она может быть перемещена или удалена.**  **Me.ОценкиTableAdapter.Fill(Me.Database2DataSet.Оценки)**  **End Sub** |

*Рис. 6.3-8. Код обработчика событий* ***Form1\_Load()***

* 1. **Редактирование, проверка и сохранение данных в**

**приложении**

После заполнения набора данных обычно производится добавление, редактирование или удаление данных перед их отправкой обратно источнику данных, другому процессу или приложению. Поскольку каждая запись в наборе данных представляется объектом [**DataRow**](http://msdn.microsoft.com/ru-ru/library/system.data.datarow(printer).aspx), изменения набора данных выполняются посредством работы с отдельными строками.

В **Приложении** **Windows Forms** архитектура привязки к данным занимается отправкой изменений из элементов управления с привязкой к данным в набор данных, поэтому не требуется в явном виде обновлять набор данных при помощи собственного кода.

Наборы данных поддерживают множество версий строк данных для нахождения изначальных записей в источнике данных. Перед выполнением обновления источника данных, возможно, потребуется проверить отдельные строки.

Изменение данных в [**DataSet**](http://msdn.microsoft.com/ru-ru/library/system.data.dataset(printer).aspx)представляет собой процесс управления фактическими данными в отдельных объектах, которые составляют набор данных. Изменение данных в таблицах данных подобно изменению данных в таблице любой базы данных – процесс может включать вставку, обновление и удаление записей таблицы.

Кроме изменения фактических данных, также можно запрашивать [**DataTable**](http://msdn.microsoft.com/ru-ru/library/system.data.datatable(printer).aspx) для возвращения определенных строк данных, например, отдельных строк, определенных версий строк (исходные и предложенные), только измененных строк и строк, которые содержат ошибки.

* 1. **Разработка программного кода приложения.**

Реализация модуля **Modulе1** представлена на рис. 6.3.9.

|  |
| --- |
| **Module Module1**  **Public Sub vivod(ByVal t As String, ByVal B As TextBox)**  **B.Text = t**  **End Sub**  **Public Sub way(ByRef t As String, ByRef d As DataRowCollection, \_**  **ByVal mark As DataRowCollection)**  **Dim i, j, k, p, n As Integer**  **n = d.Count**  **For i = 0 To 2 \* (n - 1) Step 2**  **k = 0 : p = 0**  **For j = 2 To 4**  **If mark(i)(j) = 5 Then k = k + 1**  **If mark(i + 1)(j) = 5 Then p = p + 1**  **Next**  **If p > k Then**  **t = t & d(i \ 2)(1) & " группа " & d(i \ 2)(2) & \_**  **" специальность: " & d(i \ 2)(3) & vbCrLf**  **End If**  **Next**  **End Sub**  **End Module** |

*Рис. 6.3-9. Код модуля* ***Modulе1***

C первой формой будет связан код, приведенный на рис. 6.3-10.

|  |
| --- |
| **Public Class Form1**  **Private Sub Button1\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) Handles Button1.Click**  **Me.Hide()**  **Form2.Show() 'Переход ко второй форме**  **End Sub**  **Private Sub Button2\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) Handles Button2.Click**  **End**  **End Sub**    **Private Sub Form1\_Load(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) Handles MyBase.Load**  **'TODO: Данная строка кода позволяет загрузить данные**  **'в таблицу "Database2DataSet.Оценки".**  **'При необходимости она может быть перемещена или удалена.**  **Me.ОценкиTableAdapter.Fill(Me.Database2DataSet.Оценки)**  **End Sub**  **End Class** |

*Рис. 6.3-10. Код первой формы* ***Form1***

Со второй формой будет связан код, приведенный на рис. 6.3-11.

|  |
| --- |
| **Public Class Form2**  **Public t As String**  **Private Sub Form2\_Load(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) \_**  **Handles MyBase.Load**  **Dim d As DataRowCollection**  **Dim mark As DataRowCollection**  **d = Form1.Database2DataSet.Студенты.Rows**  **mark = Form1.Database2DataSet.Оценки.Rows**  **TextBox2.Text = "Контрольныe проводились: " & vbCrLf & \_**  **"Первая :" & mark(0)(1) & vbCrLf**  **TextBox2.Text = TextBox2.Text & "Вторая: " & mark(1)(1)**  **t = ""**  **way(t, d, mark)**  **vivod(t, TextBox3)**  **End Sub**  **Private Sub Button1\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) \_**  **Handles Button1.Click**  **Me.Hide() : Form1.Show()**  **End Sub**  **End Class** |

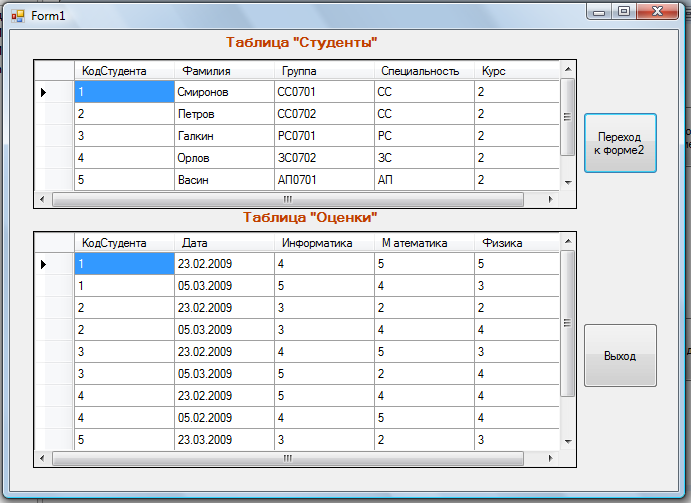
*Рис. 6.3-11.**Код второй формы* ***Form2***

Текущая дата в документе этой формы определяется с помощью элемента **DataTimePicker,** который выбирается из панели элементов.

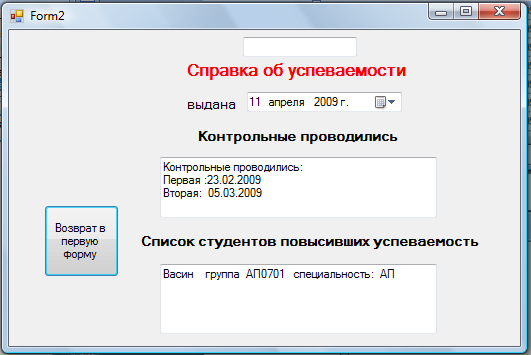
Процедура way() реализует алгоритм, позволяющий выводить на экран документы, о студентах, которые повысили свою успеваемость на второй контрольной по сравнению с первой.

1. **Выполнение приложения**

Результат выполнения программы представлены на рис.6.3-12 и   
 рис. 6.3-13.



*Рис.6.3-12. Результат на форме* ***Form1***



*Рис.6.3-13. Результат на форме* ***Form2***

## 6.4. Задачи для самостоятельного решения

Общее задание: Создать БД, содержащую сведения в двух таблицах. Разработать и создать проект, реализующий соответствующий запрос. Результат запроса должен отражаться в выходном документе.

Таблица 6.4.1

|  |  |
| --- | --- |
| **№** | **Структуры таблиц БД и запросы** |
| 1) | **СТУДЕНТЫ: Код, фамилия, группа, курс, специальность.**  **ОЦЕНКИ: Код, дата, информатика, математика**.  Выбрать тех студентов первого курса, у которых четверок больше, чем у студентов второго курса. |
| 2) | **СТУДЕНТЫ: Код, фамилия, группа, курс, специальность**.  **ОЦЕНКИ: Код, дата, информатика, математика.**  Выбрать тех студентов, у которых четверок больше, чем троек. |
| 3) | **СТУДЕНТЫ: Код, фамилия, группа, курс, специальность.**  **ОЦЕНКИ: Код, дата, информатика, математика.**  Выбрать тех студентов, у которых троек по информатике меньше чем по математике. |
| 4) | **ПРЕДПРИЯТИЕ: Код предприятия, название, количество работающих, объем выпускаемой продукции (штук).**  **СОТРУДНИКИ: Код предприятия, количество сотрудников без жилья, количество сотрудников, нуждающихся в улучшении жилья**.  Выбрать предприятия, объем выпуска которых превышает 1000 штук, а нуждающихся в жилье менее 100. |
| 5) | **ПРЕДПРИЯТИЕ: Код предприятия, название, количество работающих, объем выпускаемой продукции (штук).**  **СОТРУДНИКИ: Код предприятия, количество сотрудников без жилья, количество сотрудников, нуждающихся в улучшении жилья.**  Выбрать предприятия, объем выпуска которых не превышает заданного значения, а нуждающихся в жилье меньше числа людей, которые не имеют жилья. |
| 6) | **ПРЕДПРИЯТИЕ: Код предприятия, название, количество работающих, объем выпускаемой продукции (штук).**  **СОТРУДНИКИ: Код предприятия, количество сотрудников без жилья, количество сотрудников, нуждающихся в улучшении жилья.**  Выбрать предприятия, в которых процент не имеющих жилья не превышает 50 процентов. Упорядочить по убыванию значений процентов. |
| 7) | **ФИРМА: Название, адрес, водитель.**  **ВОДИТЕЛЬ: ФИО, дата вывоза, число рейсов**.  Выбрать водителей, которые сделали более 3-х поездок в заданный период. |
| 8) | **ФИРМА: Название, адрес, водитель.**  **ВОДИТЕЛЬ: ФИО, дата вывоза, число рейсов**  Выбрать водителей, которые непрерывно увеличивали число поездок в заданный период. |
| 9) | **ФИРМА: Название, адрес, водитель.**  **ВОДИТЕЛЬ: ФИО, дата вывоза, затраты на бензин**.  Выбрать фирмы и водителей, затраты на бензин которых не превышают заданной величины. |
| 10) | **СПОРТСМЕН: ФИО, возраст, спортобщество.**  **ОЦЕНКИ: ФИО и 5 оценок за выступления (по 10-бальной системе**).  Выбрать по каждому обществу ФИО спортсменов с наименьшим суммарным баллом. |

## 6.5. Лабораторная работа по теме «Средство Visual Studio.Net ADO.NET для работы с базами данных»

**Цель** данной работы состоит в изучении технологии ADO.NETпри разработке проекта средствами VB.NET с использованием БД реляционного типа, состоящей из одной таблицы.

### 6.5.1. Вопросы, подлежащие изучению

1. Основные понятия и определения БД.
2. БД реляционного типа.
3. Основные варианты работы с БД в VB.NETс использованием технологии ADO.NET.
4. Создание реляционной БД в среде СУБД Access.
5. Установка соединения с базой данных в проекте.
6. Создание приложения для работы с БД.
7. Использования коллекций при адресации к ячейкам таблицы.

### 6.5.2. Задание

1. ***Изучите основные понятия, определения и средства Visual Studio.Net ADO.NET для работы с базами данных (п. п. 6.1-6.3).***
2. ***Выберите вариант индивидуального задания*** *из таблицы 6.5-1.*
3. ***Уточнение задания****:*
   * *определите необходимые данные;*
   * *определитесь с названием проекта, БД и ее таблицы;*
   * *при необходимости приведите расчетные формулы, которые будут использоваться для преобразования исходных данных в результаты.*
4. ***Разработка БД:***
   * *разработайте* ***ER****-модель;*
   * *создайте БД средствами* ***СУБД*** *Access.*
5. ***Разработка приложения (проекта) средствами*** *VB.NET****:***
   * *разработайте интерфейс пользователя с элементами, предназначенными для отображения данных таблицы БД, для отображения управляющих элементов, для отображения результатов решения задачи);*
   * *создайте программный код проекта, который реализует поставленную задачу и использует стандартные модули VB. NET.*
6. ***Подготовьте тестовые наборы исходных данных****, п****олучите решение и подготовьте отчет.***

### 6.5.3.Индивидуальные варианты заданий

Общее задание: создать БД, содержащую сведения в одной таблице. Разработать и создать проект, реализующий соответствующий запрос-выборку. Результат запроса должен отражаться в выходном документе

Таблица 6.5-1

|  |  |  |
| --- | --- | --- |
| **Номер** | **Название и содержимое таблицы БД** | **Выборка** |
| 1) | **Студенты: дата проверки, группа, фамилия студента, число пропусков по математике, физике и физкультуре.** | Найти для заданной группы суммарное число пропусков. |
| 2) | **Студенты: дата проверки, группа, фамилия студента, число пропусков по математике, физике и физкультуре.** | Выбрать фамилии студентов, имеющих пропуски занятий в заданные даты. |
| 3) | **Вуз: название вуза, название специальности, год приема, план приема, количество принятых студентов.** | Выбрать список вузов, в которых имеется заданная специальность, и план приема не превосходит количество принятых. |
| 4) | **Студенты: дата проверки, группа, фамилия студента, число пропусков по математике, физике и физкультуре.** | Выбрать фамилии самых молодых студентов, имеющих суммарно наименьшее число пропусков. |
| 5) | **Магазин: название магазина, название товара, стоимость товара, дата продажи.** | Выбрать все товары, не превосходящие заданной стоимости, которые были проданы после заданной даты. |
| 6) | **Студенты: дата проверки, группа, фамилия студента, число пропусков по математике, физике и физкультуре.** | Выбрать даты, в которые пропуски по математике превышали пропуски по физкультуре с указанием фамилии студентов и их групп. |
| 7) | **Сотрудники: табельный номер, должность, год рождения, оклад, фамилия.** | Выбрать самых молодых сотрудников с максимальным окладом. |
| 8) | **Товар: наименование товара, количество на складе, дата поступления, изготовитель.** | Выбрать список товаров заданного изготовителя, количество которых находится в заданных пределах. |
| 9) | **Сотрудники: табельный номер, должность, оклад, фамилия.** | Выбрать сотрудников заданной должности с максимальным окладом. |
| 10) | **Аэропорт: марка самолета, количество мест, средняя стоимость билета, количество проданных билетов.** | Выбрать марки самолетов, для которых количество проданных билетов меньше половины всех мест самолета. |
| 11) | **Магазин: дата покупки, фамилия покупателя, название товара, цена.** | Выбрать покупателей, купивших самый дешевый товар. |
| 12) | **Детали: код детали, название, норма расхода материала, сорт.** | Выбрать детали первого и второго сорта. |
| 13) | **Предприятие: название предприятия, количество сотрудников предприятия, профессии.** | Выбрать предприятия, имеющие заданные профессии. |
| 14) | **Сотрудники: табельный номер, должность, оклад, фамилия.** | Выбрать сотрудников, средний заработок которых больше заданного. |
| 15) | **Сотрудники: табельный номер, профессия, оклад, фамилия.** | Выбрать сотрудников, сгруппировав их по профессиям. Фамилии для каждой профессии должны быть расположены по алфавиту. |
| 16) | **Автомобили: модель авто, стоимость, цвет кузова, наличие или отсутствие автоматической коробки передач.** | Выбрать автомобили с автоматической коробкой передач, имеющих заданный цвет и стоимость, не превосходящую заданную. |
| 17) | **Автомобили: модель, стоимость. год выпуска, страна сборки.** | Выбрать автомобили, упорядочив их по году выпуска. |
| 18) | **Изделия: код изделия, название предприятия, план выпуска, фактический выпуск.** | Выбрать предприятия, сгруппировав их по наименованиям изделий. Вывести предприятия перевыполнившие план. |
| 19) | **Предприятия: название, количество сотрудников, профессии.** | Выбрать предприятия, количество сотрудников которых находится в заданных пределах. |
| 20) | **Изделия: код изделия, наименование изделия, название предприятия, план выпуска, фактический выпуск.** | Выбрать изделия, напечатав их по названиям предприятиям, фактический выпуск которых превосходит среднее значение по всем предприятиям. |
| 21) | **Предприятия: название, количество сотрудников, объем выпускаемой продукции, профессии.** | Выбрать предприятия, имеющие минимальное количество сотрудников и наибольший объем выпуска. |
| 22) | **Радиолокационные станции: адрес станции, год ввода в эксплуатацию, тип используемой антенны, диапазон.** | Выбрать для каждого диапазона самые старые станции. |
| 23) | **Студенты: год рождения, адрес, дата зачисления, номер приказа, факультет, группа.** | Выбрать самых молодых студентов для каждого факультета. |
| 24) | **Радиолокационные станции: адрес станции, год ввода в эксплуатацию, тип используемой антенны, диапазон.** | Вывести станции, сгруппировав их по типам антенн. |
| 25) | **Преподаватели: год рождения, адрес, телефон, кафедра, должность, учёная степень.** | Выбрать самых молодых преподавателей, не имеющих ученой степени, и самых пожилых преподавателей с ученой степенью. |
| 26) | **Книги: издательство, авторы, название, раздел (техническая, общественно-политическая и т.п.), год издания, количество страниц, цена.** | Выбрать все книги, цена которых выше средней цены для всех книг данного издательства. |
| 27) | **Читатели: номер читательского билета, ФИО, год рождения, дата записи, вид читателя (студент, аспирант и т.п.), названия взятых книг и даты их выдачи.** | Определить общее количество книг по каждому виду читателя. |
| 28) | **Продажи: наименования товаров, поставка из магазина или со склада, количество и стоимость товара.** | Определить самый дорогой товар из магазина и самый дешевый со склада. |
| 29) | **Блюда: цена, название, категория (мясное, рыбное, салат и т.п.), является ли вегетарианским, время приготовления (в мин.), есть ли в наличии.** | Выбрать все вегетарианские блюда, время приготовления которых меньше заданного. |
| 30) | **Проекты: название, дата начала, дата окончания, размер финансирования, тип финансирования (периодический, разовый), исполнитель.** | Выбрать все проекты по каждому типу финансирования, размер финансирования которых не превышает заданную сумму. |

### 6.5.4. Содержание отчета

1. Тема и название работы.
2. Задание на разработку проекта и вариант задания.
3. Уточнение задания.
4. Разработка БД.

4.1.) Разработка ER-модели

4.2.) Создание БД средствами СУБД Access.

1. Элементы разрабатываемого приложения:
2. Графический интерфейс пользователя.
3. Результаты установки соединения с базой данных.
4. Результат выполнения проекта.
5. Результат изменения значений набора данных.
6. Укрупненная схема алгоритма решения задачи и схемы

алгоритмов процедур.

1. Программный код проекта с использованием процедур и

Стандартного модуля.

1. Результаты выполнения приложения.

### 6.5.5. Пример разработки приложений для работы с базой данных «Посещаемость студентов»

#### Тема и название работы:

Средства Visual Studio.Net – ADO.NET для работы с базами данных.

Разработка приложения для работы с **БД «Посещаемость студентов».**

#### Задание на разработку БД предметной области «Посещаемость студентов»:

В БД поступают результаты рубежного контроля о посещаемости занятий студентами по трем дисциплинам за определенный период. Необходимо создать проект, который, используя сведения из этой базы данных, вычисляет, в какой день и по какому предмету имелось наибольшее число пропусков (указать также фамилию студента, имеющего такое число пропусков). Если таких дней несколько, то перечислить их все.

#### Уточнение задания:

Данные о каждом студенте должны содержать:

* дату рубежного контроля;
* номер группы;
* фамилию студента;
* год рождения;
* число пропущенных часов по информатике;
* число пропущенных часов по математике;
* число пропущенных часов по физике.

Файл БД назовем **Stud.mdb**, а сущность этой БД - **СТУДЕНТЫ.**

#### Разработка БД:

* 1. **Разработка ER-модели:**

На языке инфологического моделирования получим:

|  |
| --- |
| **СТУДЕНТЫ (N,дата,группа,фамилия,год,информатика,математика,физика)** |

* 1. **Создание БД, содержащей одну таблицу, средствами СУБД Access**:
* Запустите приложение MS Access и создайте новую базу данных с именем **Stud**.**mdb**.
* Задайте структуру таблицы в конструкторе (имена, типы и свойства полей):
* **N *–*** Счетчик, Ключевое поле;
* **Дата** – Дата/время;
* **Группа** – Текстовый, Обязательное поле – Да, Индексированное – Да (Совпадения допускаются);
* **Фамилия** –Текстовый, Обязательное поле – Да, Индексированное – Да (Совпадения допускаются);
* **Год**, **Информатика**, **Математика**, **Физика** – все Числовой.
  + - * Закройте окно конструктора и сохраните таблицу под именем **СТУДЕНТЫ**.
      * Введите в таблицу **СТУДЕНТЫ** 10-15 записей и сохраните их в БД (Рис.6.5-1).

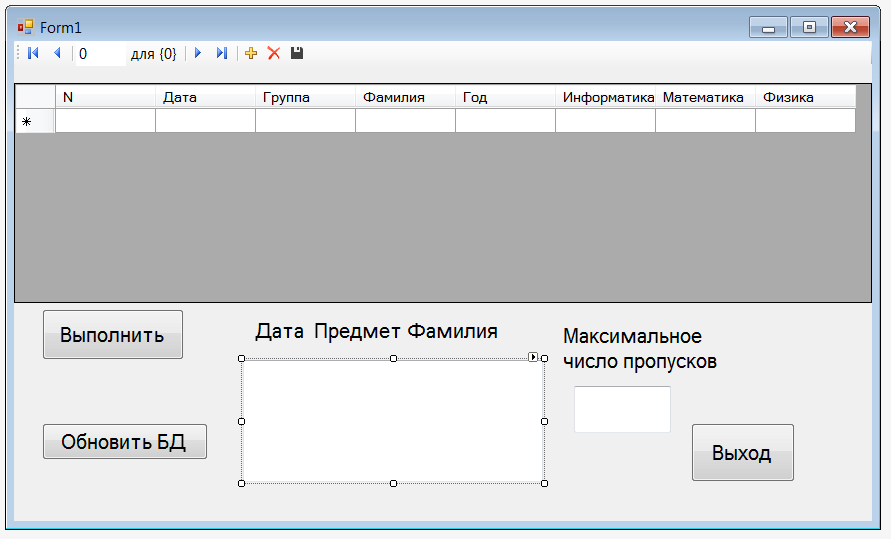


*Рис.6.5-1. Записи таблицы* ***СТУДЕНТЫ***

#### Разработка приложения

1. **Создайте новый проект в VS с именем ФАМИЛИЯ\_ГРУППА.**
2. **Разработка графического интерфейса пользователя.**

Графический интерфейс пользователя – на рис. 6.5-2.   
Обратите внимание на то, что макет таблицы **СТУДЕНТЫ** на форме появляется только после перетаскивания таблицы на форму (см. ниже).

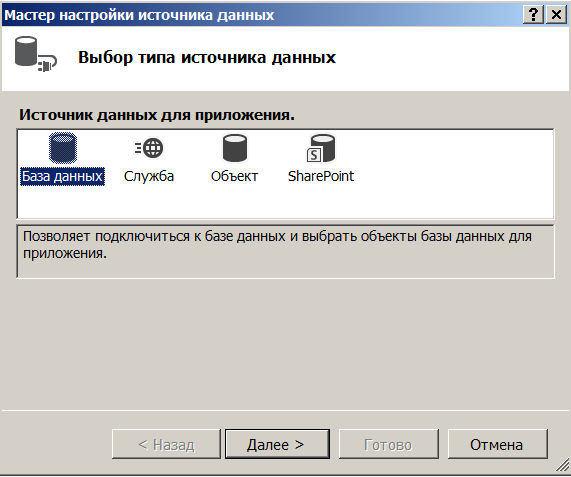


*Рис. 6.5-2. Графический интерфейс пользователя проекта* **ФАМИЛИЯ\_ГРУППА**

1. **Установление соединения с базой данных.**
   * + Подключитесь к данным базы данных **Stud** СУБД Access, для чего:

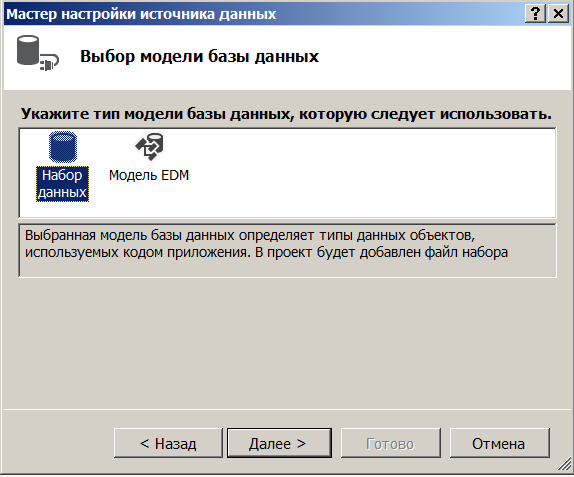
* выберите в элементе главного меню **Проект** команду   
  ***Добавит***ь ***новый источник данных***;

в появившемся окне **Мастер настройки источника данных** выберите **Источник данных для приложения – База данных** (рис. 6.5-3) и нажмите кнопку ***Далее***, откроется новое окно **Мастер настройки источника данных** (рис. 6.5-4);



*Рис. 6.5-3. Окно* ***Мастер настройки источника данных***

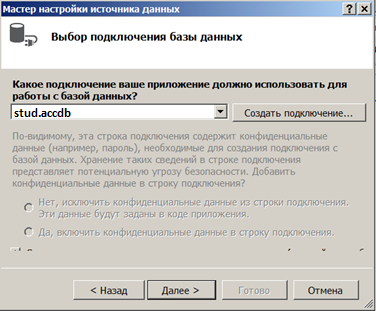
* в окне **Мастер настройки источника данных** выберите тип модели базы данных – **Набор данных** и нажмите кнопку ***Далее***.



*Рис. 6.5.-4. Выбор типа модели БД*

Мастер отобразит форму – **Выбор подключения базы данных**, которая помогает установить соединение с базой данных, создав связывающую строку (рис. 6.5-5);

* нажмите кнопку ***Создать подключение*** и нажмите кнопку ***Далее***;

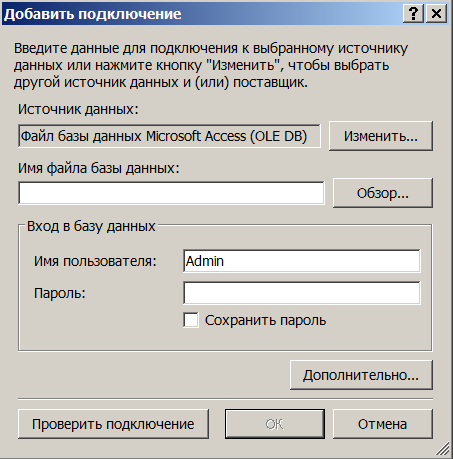


*Рис. 6.5-5. Выбор подключения*

* в появившемся окне **Добавить подключение** введите путь к файлу базы данных **Stud.mdb** или щелкните кнопку **Обзор**, чтобы найти этот файл (рис. 6.5-6);
* нажмите кнопку ***Проверить подключение***.

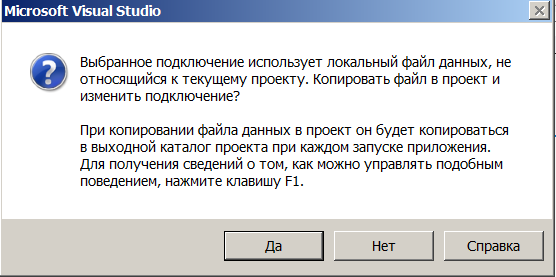
VS попробует открыть указанный файл базы данных, используя связывающую строку, созданную мастером. Если база данных имеет правильный формат, то будет выдано сообщение **Проверка подключения выполнена;**

* щелкните на кнопке ОК, чтобы закрыть окно сообщения, а затем на кнопке ОК, чтобы закрыть окно **Добавить подключение**.



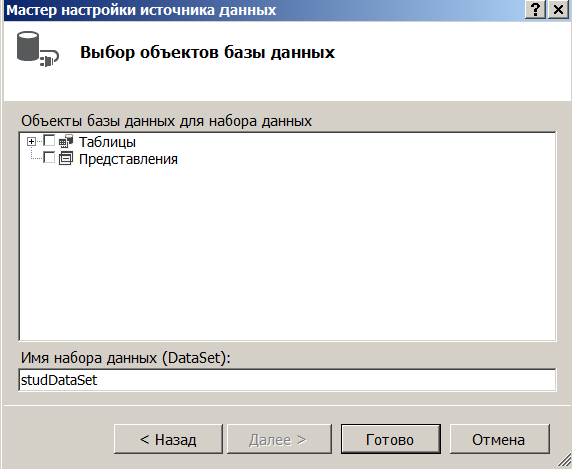
*Рис. 6.5-6. Добавление подключения*

* Мастер отобразит предупреждение (Рис. 6.5-7), указывающее, что была выбра­на новая локальная база данных, и спросит, следует ли скопировать базу данных в папку проекта. Это сообщение появляется только при установке первого соединения с файлом локальной базы данных. При повторном выполнении этого упражнения данного сообщения не бу­дет. Чтобы избежать создания дополнительной копии базы данных, щел­кните на ***Нет***.



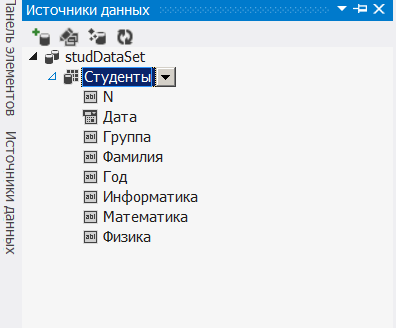
*Рис. 6.5-7. Сообщение - предупреждение*

* в **Мастере настройки источника данных** включите опцию ***Да***, сохраните подключение и нажмите кнопку ***Далее*;** будет выдан запрос на выбор подмножества объектов базы дан­ных, которые будут использоваться в данном проекте, как показано в окне **Мастер настройки источника данных (**рис. 6.5-8);



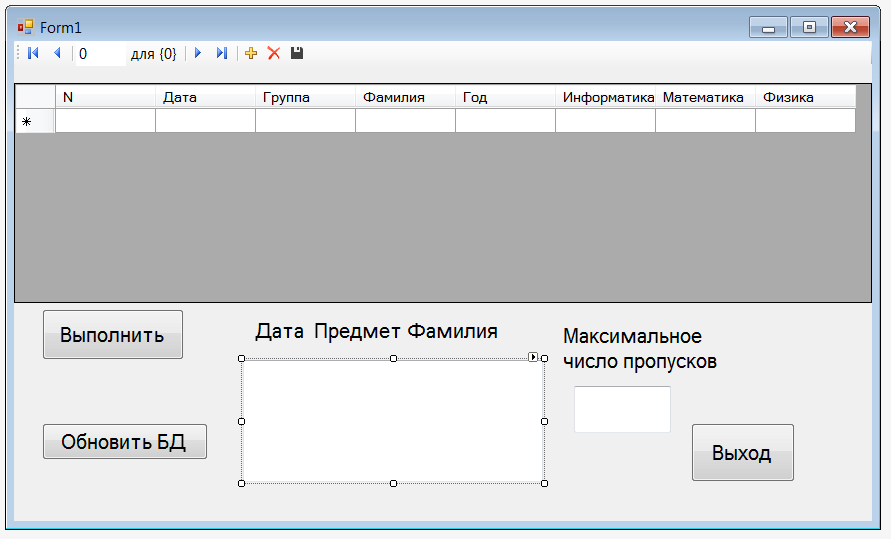
*Рис. 6.5-8. Выбор объектов БД*

* выберите таблицу **СТУДЕНТЫ** (Рис.6.5-9)



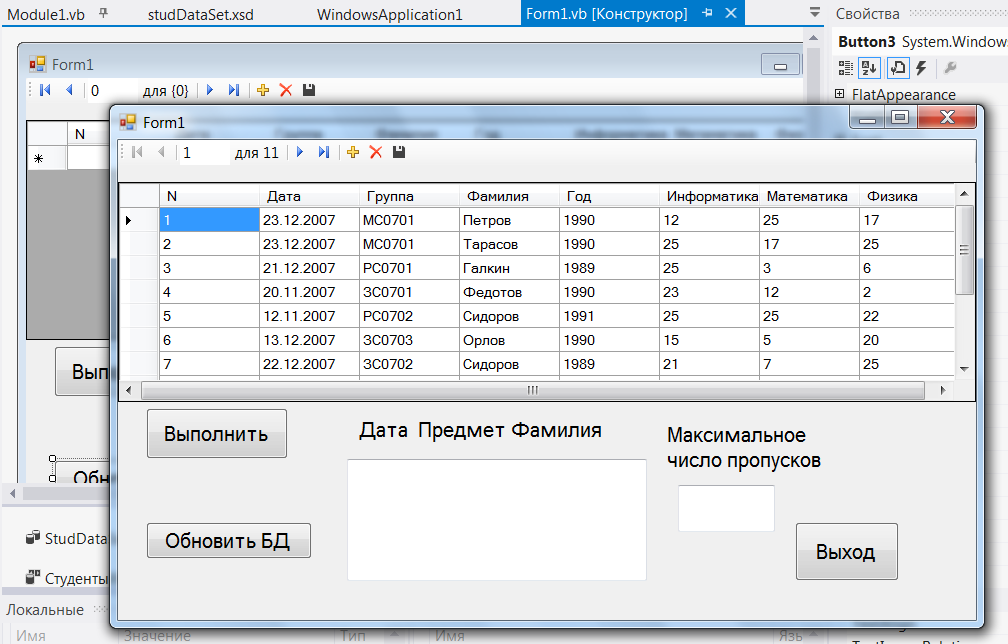
*Рис. 6.5-9. Выбор таблицы* ***СТУДЕНТЫ***

* выбранную таблицу **СТУДЕНТЫ** в окне **Источники данных** перетащите на форму; становятся видны названия полей (рис. 6.5-10). Это происходит благодаря тому, что подключается элемент управления **DataGridView**, позволяющий иметь на форме элемент в виде таблицы с заголовками, делая тем самым удобным зрительное восприятие таблицы БД, а также навигационную панель (вверху формы) **BindingNavigator,** имеющую кнопки для перемещения по записям, добавления, удаления и сохранения записей в наборе **DataSet**. Кроме того, в результате перетаскивания таблицы на форму в код программы добавляется процедура **СтуденыBindingNavigatorSaveItem\_Click( )** и обработчик событий **Form1\_Load( )**, который будет выполняться при загрузке формы.



*Рис. 6.5-10. Вид формы с таблицей*

1. **Запустите отладку проекта. Н**а форме отобразится заполненная таблица (рис. 6.5-11);



*Рис. 6.5-11. Заполненная таблица* **СТУДЕНТЫ** *на форме* **Form1**

1. Измените значения набора данных, для чего:

* проведите непосредственно в таблице, представленной на форме, простое изменение, например, измените данные об информатике для первого студента (обратите внимание на появляющийся значок карандаша слева от записи при ее изменении);.
* не закрывая проекта, откройте таблицу **СТУДЕНТЫ** в СУБД Access, и убедитесь в том, что никаких изменений в самой БД не произошло. Сами изменения проводились в наборе данных, т.е. в **StudDataSet**. Для сохранения же изменений в БД необходимо использовать метод **Update()**

|  |
| --- |
| **Me.СтудентыTableAdapter.Update(Me.StudDataSet.Студенты)** |

* перейдите в проект и остановите отладку;
* добавьте в проект процедуры **Button1**\_**Click()** (кнопка ***Обновить БД*)** и **Button2**\_**Click() (**кнопка **Выход) (**рис. 6.5-12):

|  |
| --- |
| **Private Sub Button1\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) Handles Button1.Click**  **Me.СтудентыTableAdapter.Update(Me.StudDataSet.Студенты)**  **End Sub**  **Private Sub Button2\_Click(ByVal sender As System.Object, \_**  **ByVal e As System.EventArgs) Handles Button2.Click**  **End**  **End Sub** |

*Рис. 6.5-12. Процедуры* ***Button1\_Click()*** *и* ***Button2\_Click()***

* Запустите отладку проекта, произведите изменения в одной записи таблицы в проекте и нажмите на кнопку **Обновить БД**, а затем на кнопку **Выход** в форме. Откройте таблицу **СТУДЕНТЫ** в БД СУБД Access и убедитесь в том, что изменения произошли непосредственно в самой БД.

1. Разработайте программный код проекта.

Программный код решения задачи приведен на рис. 6.5-13, с учетом того, что форма содержит заголовок (**Label1)**, 3 кнопки (**Button1, Button2, Button3**), текстовые поля (**TextBox1** и **TextBox2**) с заголовками (**Label2**и **Label3**):

|  |
| --- |
| **Option Strict On**  **Module Module1**  **Sub успеваемость(ByVal n As Integer, ByRef max As Integer, \_**  **ByRef t As String, \_**  **ByRef b As DataRowCollection)**  **Dim i, j As Integer**  **Dim learn() As String={"информатика","математика",физика"}**  **max = Integer.MinValue**  **For i = 0 To n**  **For j = 5 To 7**  **If CInt(b(i)(j)) > max Then**  **max = CInt(b(i)(j))**  **End If**  **Next**  **Next**  **For i = 0 To n**  **For j = 5 To 7**  **If CInt(b(i)(j)) = max Then**  **t = t & CStr(b(i)(1)) & " , " & learn(j - 5)\_**  **& " , " & CStr(b(i)(3)) & vbCrLf**  **End If**  **Next**  **Next**  **End Sub**  **Sub vivod(ByVal z As String, ByVal T As TextBox)**  **T.Text = z**  **End Sub**  **End Module**  **Public Class Form1**  **Private Sub СтудентыBindingNavigatorSaveItem\_Click \_**  **(sender As Object, e As EventArgs) \_**  **Handles СтудентыBindingNavigatorSaveItem.Click**  **Me.Validate()**  **Me.СтудентыBindingSource.EndEdit()**  **Me.TableAdapterManager.UpdateAll(Me.StudDataSet)**  **End Sub**  **Private Sub Form1\_Load(sender As Object, e As EventArgs) \_**  **Handles MyBase.Load**  *'TODO: данная строка кода позволяет загрузить данные в*  *'таблицу "StudDataSet.Студенты".*  *'При необходимости она может быть перемещена или удалена.*  **Me.СтудентыTableAdapter.Fill(Me.StudDataSet.Студенты)**  **End Sub**  **Private Sub Button1\_Click(sender As Object, e As EventArgs)\_**  **Handles Button1.Click**  **Dim t As String = ""**  **Dim nz, m As Integer**  **Dim b As DataRowCollection**  **TextBox1.Text = "" : TextBox1.Text = ""**  **b = StudDataSet.Студенты.Rows**  **nz = StudDataSet.Студенты.Rows().Count() - 1**  **'число записей в БД**  **успеваемость(nz, m, t, b)**  **vivod(Str(m), TextBox1) : vivod(t, TextBox2)**  **End Sub**  **Private Sub Button2\_Click(sender As Object, e As EventArgs)\_**  **Handles Button2.Click**  **End**  **End Sub**  **Private Sub Button3\_Click(sender As Object, e As EventArgs)\_**  **Handles Button3.Click**  **Me.СтудентыTableAdapter.Update(Me.StudDataSet.Студенты)**  **End Sub**  **End Class** |

*Рис. 6.5-13. Код программного проекта* ***ФАМИЛИЯ\_ГРУППА***

Для решения задачи в данной лабораторной работе нам необходимо знать, что объект **StudDataSet** включает в себя коллекцию **Tables**, состоящую из таблиц типа **DataTable**. Каждая таблица из этой коллекции включает в себя коллекцию **Columns** столбцов (полей) типа **DataColumn** и коллекцию **Rows** строк (записей) типа **DataRow**. Каждая строка к тому же позволяет обращаться к своим полям по индексу, как к элементам массива.

Коллекции представляют собой набор элементов одного или разного типов. Обычно коллекции составляют из элементов управления или каких-либо других объектов. Коллекция **Tables** не составляется, а уже имеется в VB.

Таким образом, мы имеем возможность обращаться к полям **БД**:

|  |
| --- |
| **StudentDataSet.Tables (“Студенты”).Rows(4)(3)** |

Это выражение есть объект, представляющий нашу таблицу   
**СТУДЕНТЫ**. У него есть свойство **Rows**, являющееся коллекцией строк таблицы. Строки пронумерованы с нуля, поэтому данное выражение есть запись о фамилии студента под номером 6.

Значения ячеек таблицы можно изменять банальным присваиванием:

|  |
| --- |
| **StudDataSet.Tables(“Студенты”).Rows(4)(3)=”Сахаров”** |

Чтобы избежать необходимости при обращении к ячейкам таблицы делать такие длинные записи, можно объявить короткую переменную типом коллекции строк таблицы:

|  |
| --- |
| **Dim d As DataRowsCollection** |

Затем, выполнить оператор, присваивающий этой переменной значение коллекции строк из конкретной таблицы:

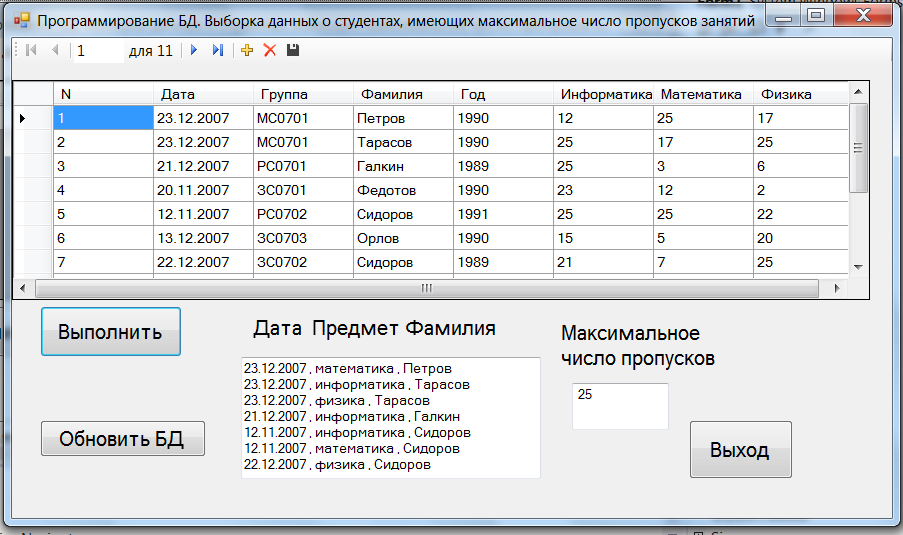
|  |
| --- |
| **d = StudDataSet.Студенты.Rows** |

Теперь к этой коллекции можно обращаться, например, так:

|  |
| --- |
| **d(4)(3) = ”Сахаров”** |

#### Результат выполнения приложения.

Результат выполнения приложения без использования **BindingNavigator** приведен на рис. 6.5-14, так как элемент управления **DataGridView** был подключен из панели инструментов.



*Рис. 6.5-14. Результат выполнения проекта* ***Посещаемость студентов***

**6.6. Контрольные вопросы**

1. Что такое база данных?
2. Что такое реляционная база данных?
3. Какие основные варианты работы с БД в VB.NET с использованием технологии ADO.NE**T** вы знаете?
4. ADO.NET – это набор библиотек или технология доступа к данным?
5. Что является основной архитектурой данных, которая используется для создания приложений в Visual Studio .NET?
6. Что является основным достоинством ADO.NET?
7. Что такое «Модель с присоединенными источниками данных»?
8. Какие проблемы БД порождаются постоянным соединением с базой данных?
9. Какие возможности предоставляет ***Мастер настройки источника данных?***
10. Перечислите основные процессы (задачи) при работе с данными.
11. Перечислите основные классы, составляющие объектную модель ADO.N**ET**.
12. Как создать реляционную БД в среде СУБД Access?
13. Как создать таблицу в БД?
14. Как установить соединение с базой данных в проекте?
15. Как использовать коллекции при адресации к ячейкам таблицы БД?