**Приложение 2**

**Методические указания по**

**проектированию диалоговых приложений**

для базы данных на примере **«База данных Кондитерской фабрики»**

**Содержание**

[Введение 11](#_Toc192146642)

[Шаг 1: Создание базы данных 11](#_Toc192146643)

[Шаг 2: Создание визуального интерфейса в Microsoft Visual Studio .NET 15](#_Toc192146644)

[Создание файла приложения. 15](#_Toc192146645)

[Подключение базы данных. 16](#_Toc192146646)

[Проектирование основной формы. 19](#_Toc192146647)

[Создание формы для работы с таблицами 22](#_Toc192146648)

[Обработка меню «Удалить текущую запись» 24](#_Toc192146649)

[Обработка меню «Сохранить изменения» 25](#_Toc192146650)

[Обработка меню «Вернуться» 26](#_Toc192146651)

[Меню «Экспорт» 26](#_Toc192146652)

[Создание экспорта в Excel 26](#_Toc192146653)

[Создание экспорта в HTML 30](#_Toc192146654)

[Обработка меню «Запросы» 33](#_Toc192146655)

[Обработка кнопки «Выполнение запроса» 34](#_Toc192146656)

[Очистка текстового окна для произвольных запросов 35](#_Toc192146657)

[Создание отчетов с помощью GrystaiReports, 36](#_Toc192146658)

[Приложение 45](#_Toc192146659)

[Заключение 51](#_Toc192146660)

# Введение

В ходе практики необходимосоздать интефейс в виде графических приложений для работы с базой данных. Он является, связующим звеном между конечными пользователями и базой данных.

В качестве инструментального средства разработки приложений демонстрируется среда Microsoft Visual [Studio.NET](file:///C:\Users\User\AppData\Local\Temp\AppData\Local\Opera\Opera\temporary_downloads\Studio.NET). Все приложения созданы на языке С# с использованием технологии доступа к данным [ADO.NET](file:///C:\Users\User\AppData\Local\Temp\AppData\Local\Opera\Opera\temporary_downloads\ADO.NET). Данный язык реализует объектно-ориентированную модель программирования, а также предоставляет наибольшую функциональность и удобство разработки.

Программа является диалоговым многооконным приложением. В ней реализована форма, предоставляющая доступ к остальным окнам программы. Просмотр таблиц осуществляется с помощью небольшой диалоговой формы, позволяющей удалять старые записи, добавлять новые и изменять существующие. Таким образом, работа со всеми таблицами реализована на одной форме.

База данных автоматически проверяет вводимые поля на целостность данных. В программе реализована возможность выполнения типовых запросов к базе данных, созданных в курсовой работе ранее в предыдущем учебном модуле. Ввод и просмотр запросов осуществляется в соответствующем диалоговом окне.

При работе с подобными приложениями следует выделить 3 логических этапа:

- проектирование базы данных в Erwin или другом Case-средстве;

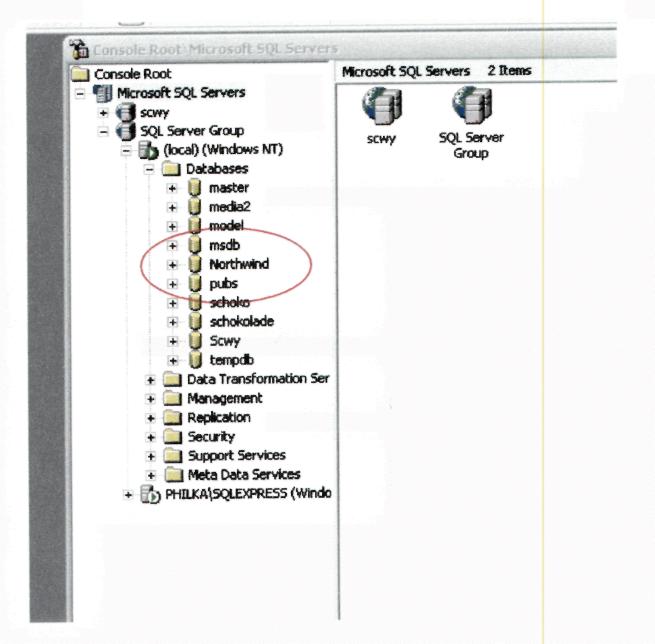
- перенос спроектированной базы в SQL Server и ее заполнение;

- создание интерфейса в среде программирования Microsoft Visual Studio 2005

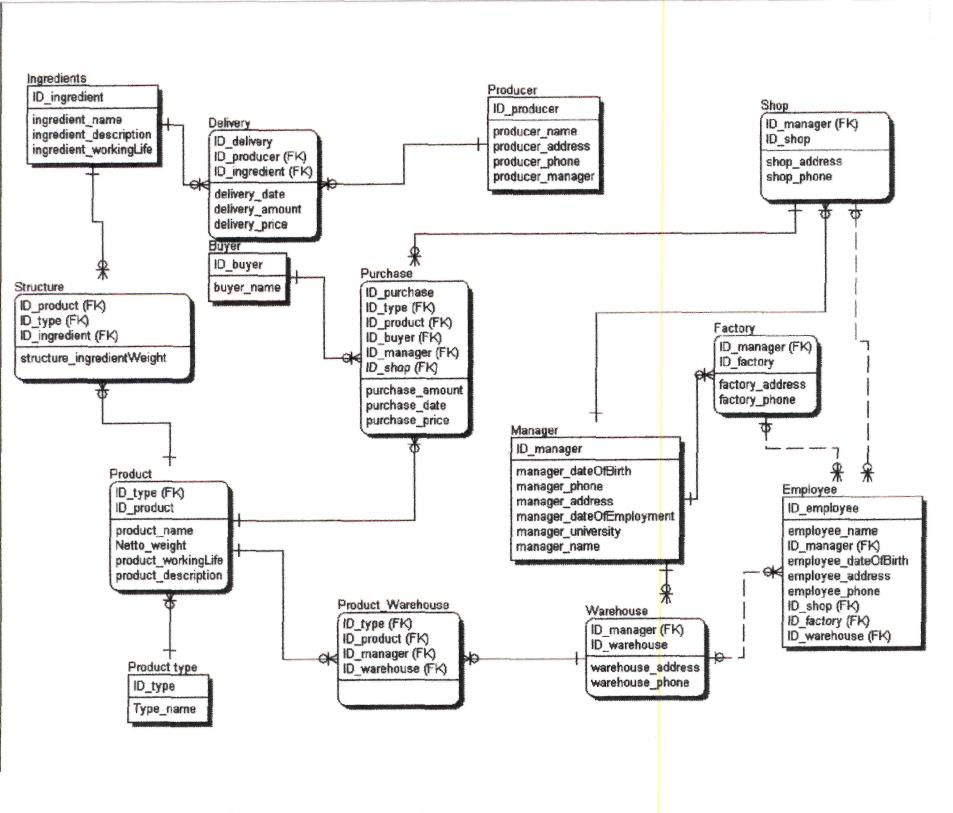
# Шаг 1: Создание базы данных

**Рекомендации**: Для работы на практике необходима база данных. Вы можете ее создать или взять готовую. Идеальный вариант, когда вы берете уже ранее созданную ВАМИ базу. В этом случае можно сразу приступать к проектированию приложения. Если же базы нет, то ее необходимо откуда-то взять.

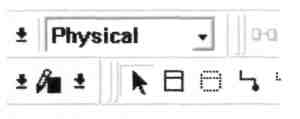
1. Выборготовой стандартной базы данных (например, в пакете с SQL Server предоставляются несколько стандартных баз), занимает наименьшее количество времени, проблема состоит в том, что потом вам придется осваивать эту базу данных, разбираться какая таблица и какое поле что значит, порой гораздо легче создать новую БД.



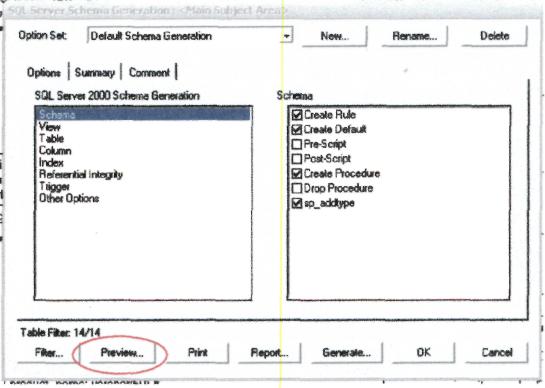
***2. Создание:*** для создания новой БД рекомендуем воспользоваться ERWin. Здесь вы просто создаете ER-модель, указываете необходимые сущности, поля, связи.



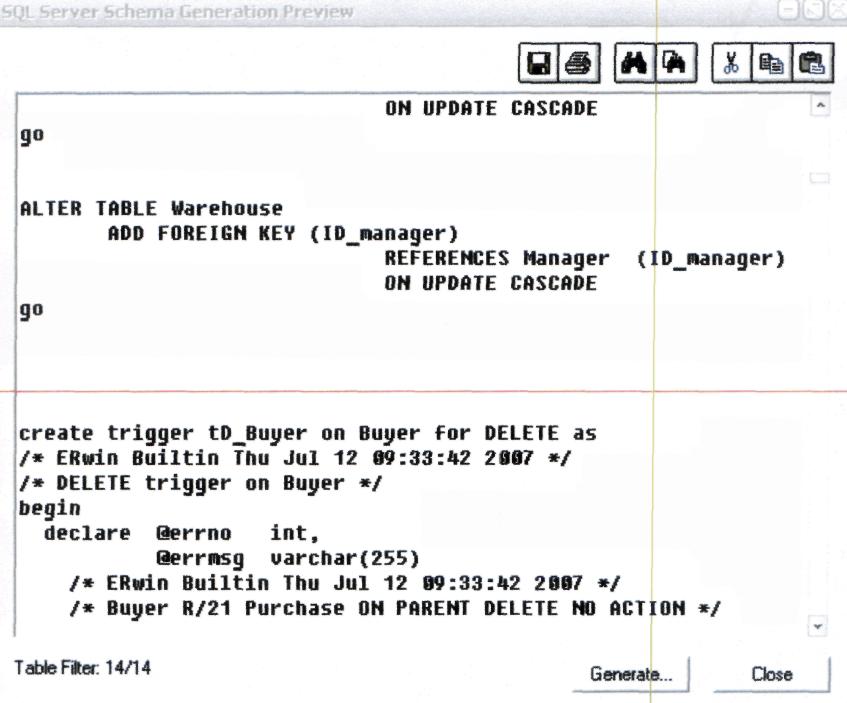
3. Затем нужно перевести это модель в SQL, для этого переходите в физическую модель:



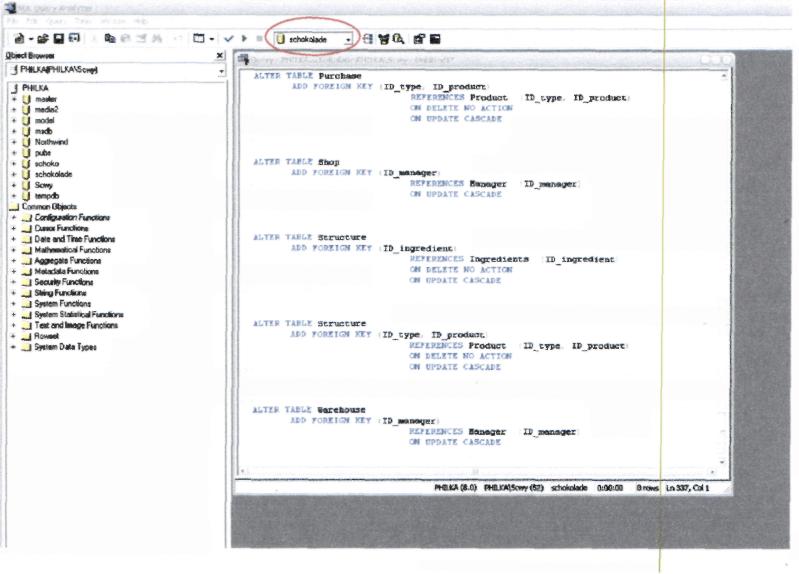
4. Для генерации модели нажмите Tools>Forward Enginireeng/Schema Generation. В открывшемся окне активизируйте Preview:



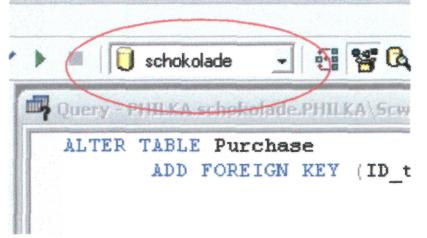
5. Скопируете полученный код из открывшегося окна до следующего текста:



6. Удалите "go", вставьте через буфер обмена в QueryAnalizer и Вы получите структуру базы дайных, с которой можно работать.



7. Не забудьте выбрать СВОЮ базу данных (пустую) которую необходимо заполнить:



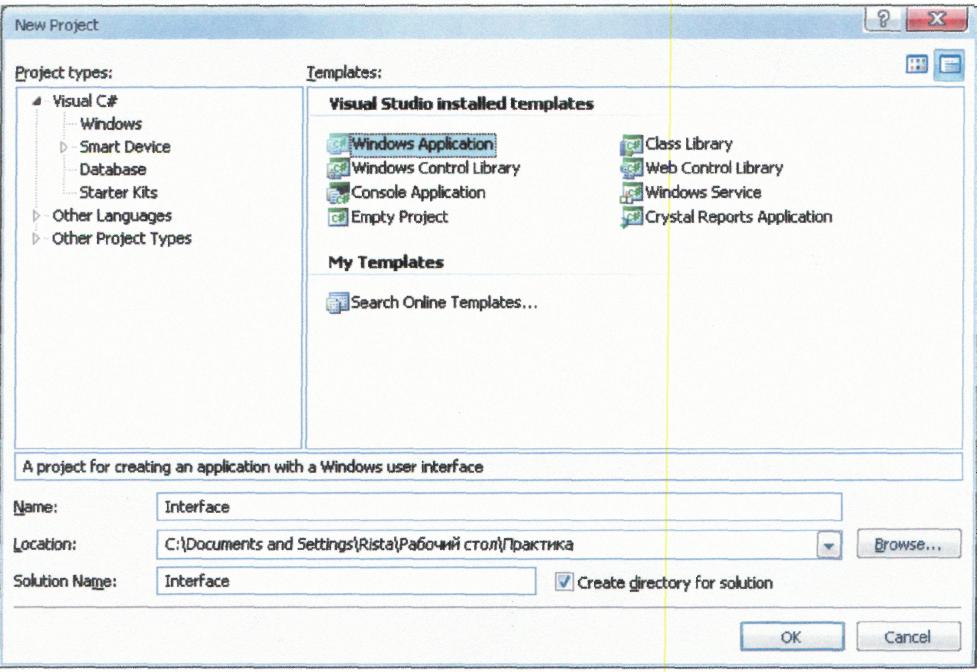
Теперь можно приступать к проектированию приложения.

# Шаг 2: Создание визуального интерфейса в Microsoft Visual Studio .NET

## **Создание файла приложения**

8. Для того, чтобы интерфейс был удобен для использования конечными пользователями, наиболее удачным вариантом будет создания Windows-приложения, состоящего из определенного количества отдельных форм, связанных с одной основной, из которой они будут **вызыватся.**

9. В первую очередь создается новый проект типа Windows Application.

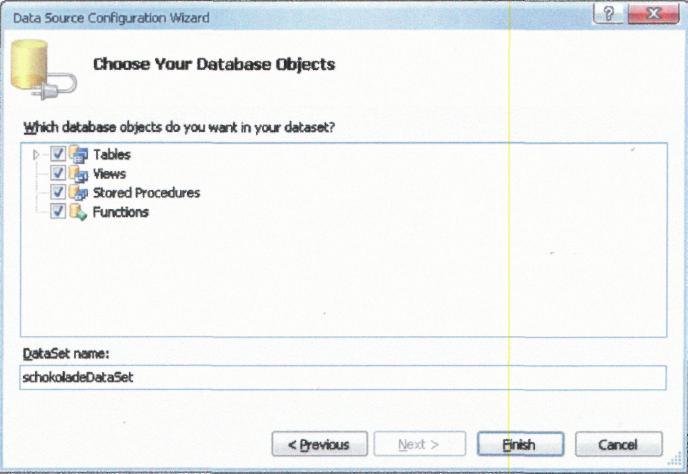


## **Подключение базы данных**

10. На следующем этапе следует создать подключение базы данных к данному приложению. Оно создается с помощь пункта меню Data > Add New Data Source. В открывшемся окошке следует выбрать вариант Database, а потом, нажав на Далее, выбрать создание нового подключения (кнопка New Connection).

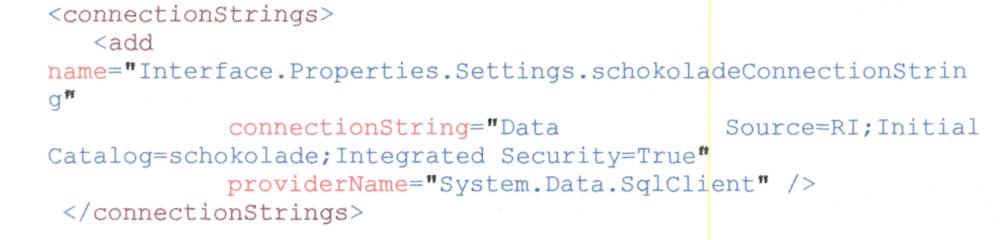


11. В ячейке Server Name вводится имя SQL сервера, на котором идет работа, а потом из списка баз данных выбирается нужная. Далее выбираются все нужные типы данных из этой базы и вводится имя Data Set.



12. Если открыть Solution Explorer (View > Solution Explorer), то обнаружится новый объект *schokoladeDataSet.xsd* - наша база данных.

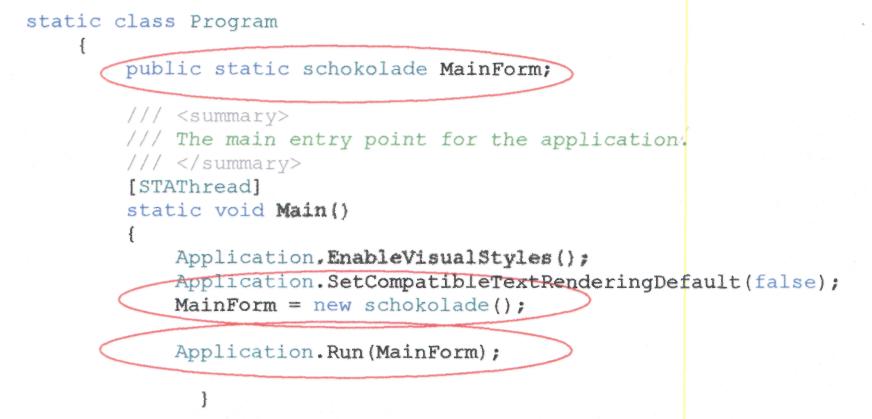
13. Стоит рассмотреть небольшой файл, находящийся все в том же Solution Explorer, под названием *app.config.* В нем прописывается Connection String, которое возникает при подключении базы. Выглядит оно так:



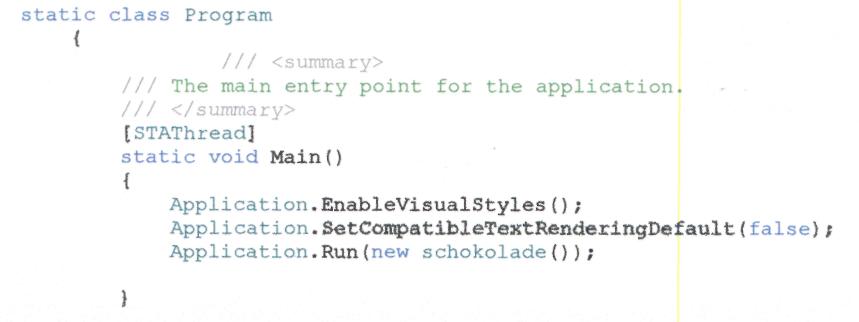
14. Для того, чтобы приложении заработало при переносе с одного компьютера на другой, следует в пункте 'Data Source-..." прописать название текущего сервера с базой данных, например localhost.

15. Кроме того, в программе потребуется вызывать соединение, поэтому лучше прописать его явно. Для удобства можно сделать так:

в файле [Prograffi.es](file:///C:\Users\User\AppData\Local\Temp\AppData\Local\Opera\Opera\temporary_downloads\Prograffi.es) прописать:

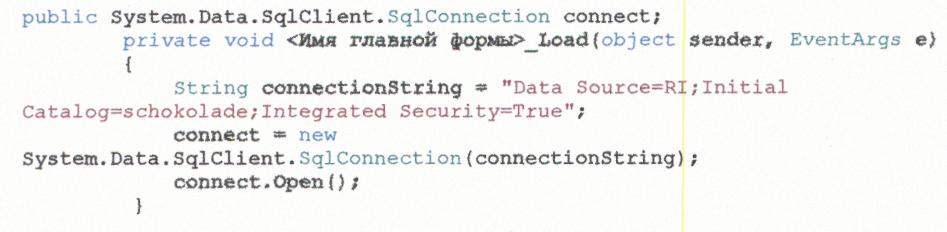


Вместо:



16. Так вы сможете видать главную форму из любой другой, и пользоваться функциями и переменными, прописанными в главной форме.

17. Далее при загрузке главной формы пропишите:



Значение всех переменных в этой записи оговаривалось ранее. Она нужна для того, чтобы соединение connect было видно из любой формы.

## **Проектирование основной формы**

18. Основную форму можно проектировать разными способами. Рекомендуем использовать Menu Strip и выпадающее меню, через которое можно получить доступ ко всем основным функциям по работе с базой. В нашем примере она выглядит так:



19.Основное меню состоит из *6* пунктов:

- вывода таблиц;

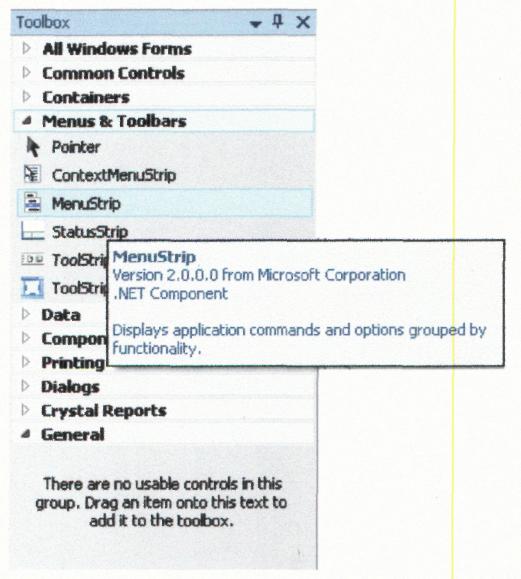
- вывода результтов запросов;

- вызова сгенерированных с помощью функции среды CrystalReports отчетов;

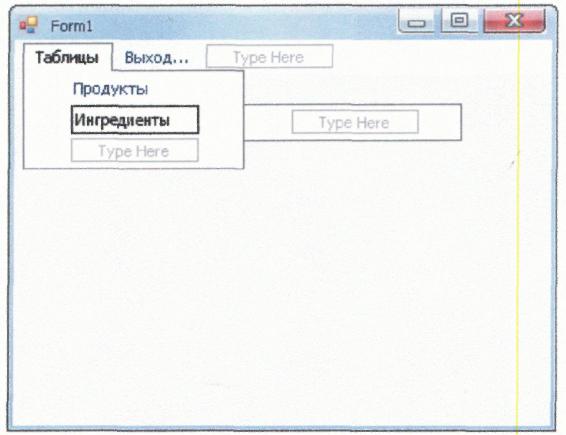
- выдачи справки и данных о разработчиках;

- пункта выхода.

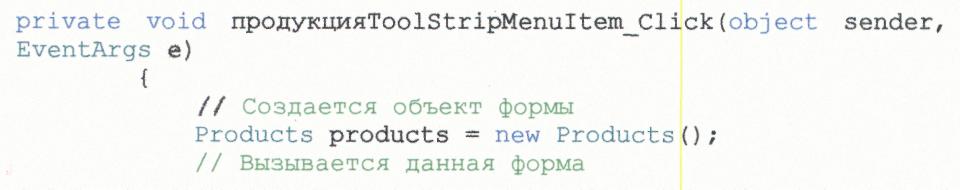
20. Для создания такого меню необходимо выполнить ряд дейтсвий. Для начала выбрать в меню View > Toolbox пункт MenuStrip и перенести его на форму.



21. Следующим действием будет заполнение пунктов меню.



22. При двойном нажатии на одном из введенных пунктов меню открывается код формы, а в нем обработчик события. В нем следует прописать связь основной формы с вызываемой. Это делается таким образом:





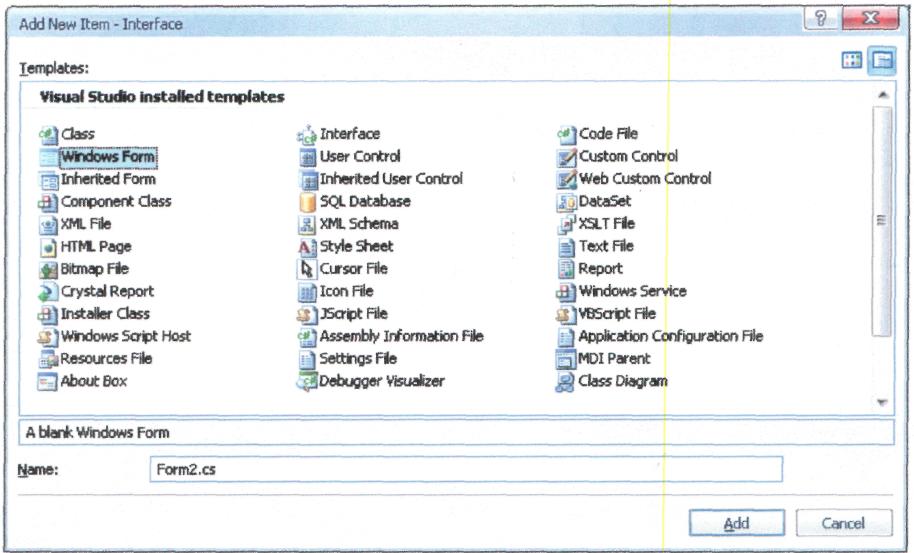
23. Для обработчика закрытия вызываемых форм используется 2 метода:

1) Close(); // Для закрытия текущей формы

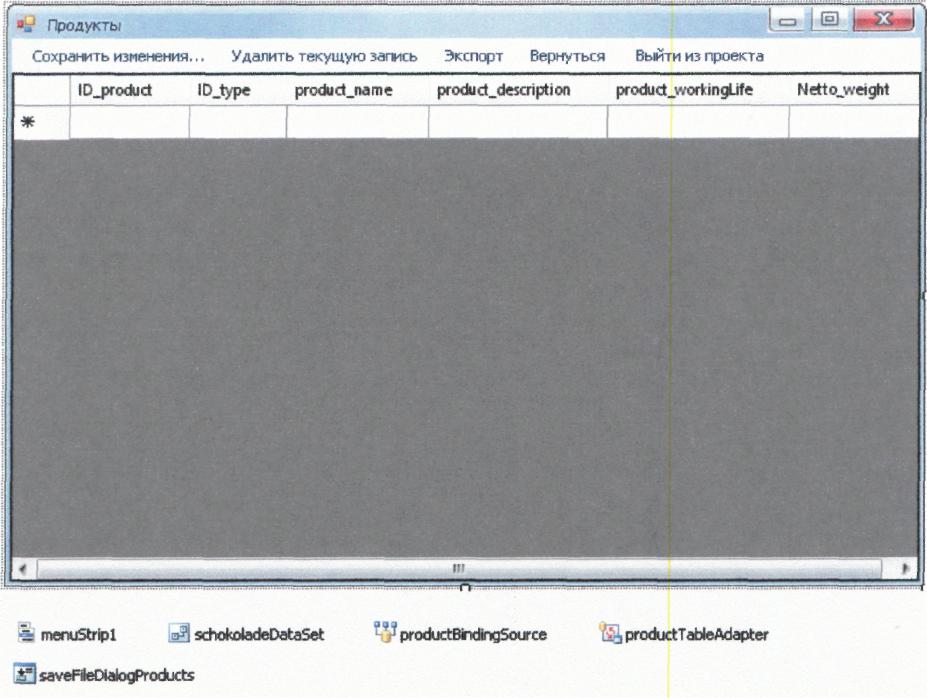
2) Application.Exit0; // Для закрытия всего приложения

## **Создание формы для работы с таблицами**

24. Создание новой формы производится нажатием Project > Add New Item > Windows Form

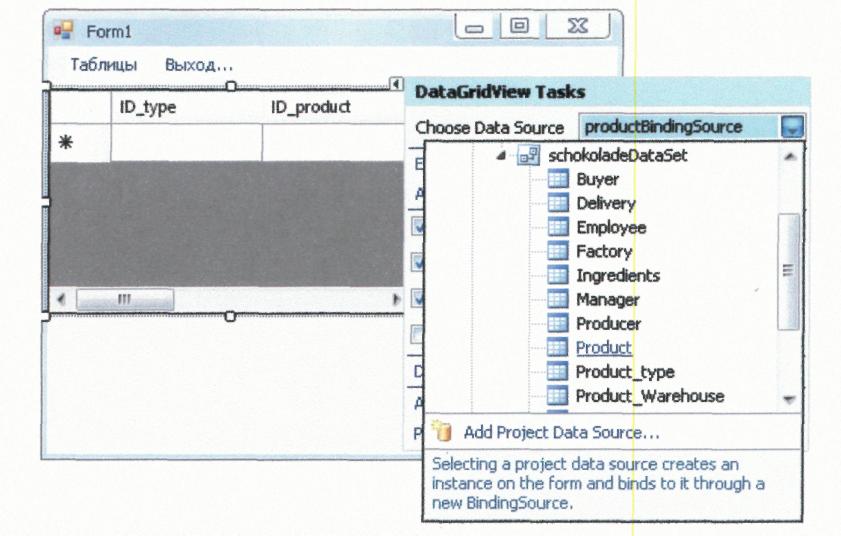


25. При создании вызываемой формы с данными по отдельным таблицам, кроме выпадающего меню, потребуется функция среды DataGridView (View > Toolbox > Data > DataGridView). Это позволит нам работать со строками таблицы. На рисунке ниже видно, как это выглядит в завершенном варианте. Под формой на сером фоне показаны те объекты, которые не видны на форме (все, кроме последнего, о котором будет написано позже, создаются автоматически).



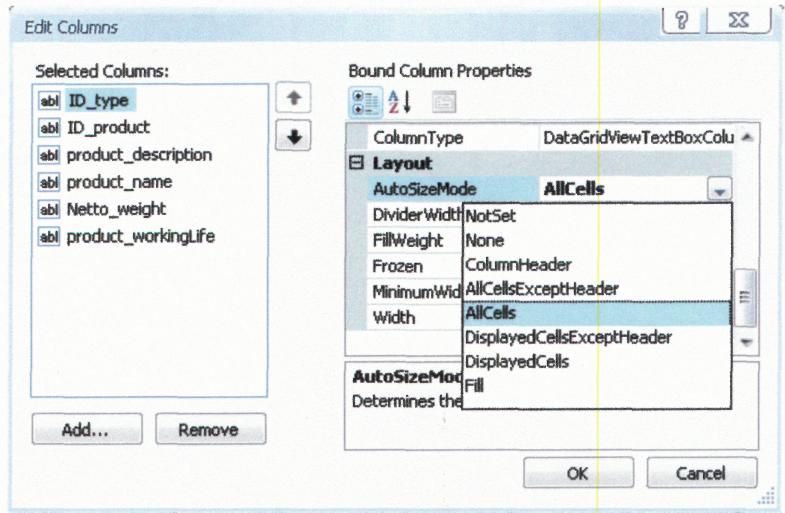
26. Для того чтобы создать DataGridView следует выбрал» его из списка a Toolbox и перенести на форму.

27. В открывшемся меню DataGridView надо выбрать подключаемую таблицу.



28. Такие функции, как Dock In Parent Container выравнивают таблицу по размеру формы.

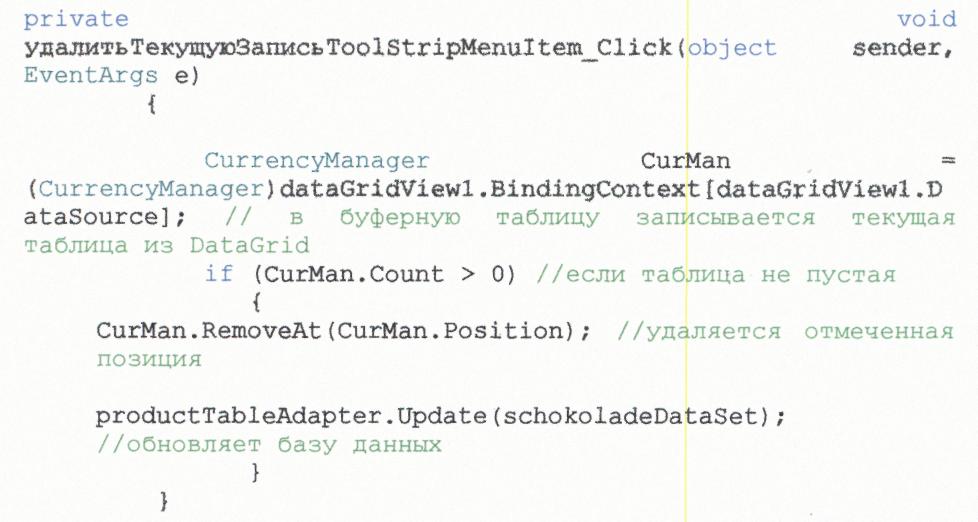
29. В подпункте меню Edit Colums можно выбрать порядок вывода столбцов или установить их автовыравнивание.



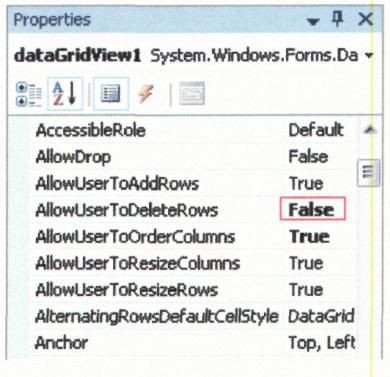
30. После подключения таблицы следует наладить ее редактирование через DataGridView (данные можно редактировать различными способами, используя textbox и методы, обрабатывающие введенные значения, но для пользователя проще всего работать непосредственно с таблицей в DataGridView).

### Обработка меню «Удалить текущую запись»

31. Для обработки пункта меню *Удалить текущую запись* (удаляет строку на которую установлен курсор) нужно прописать в его обработчике событий приблизительно следующие строки:

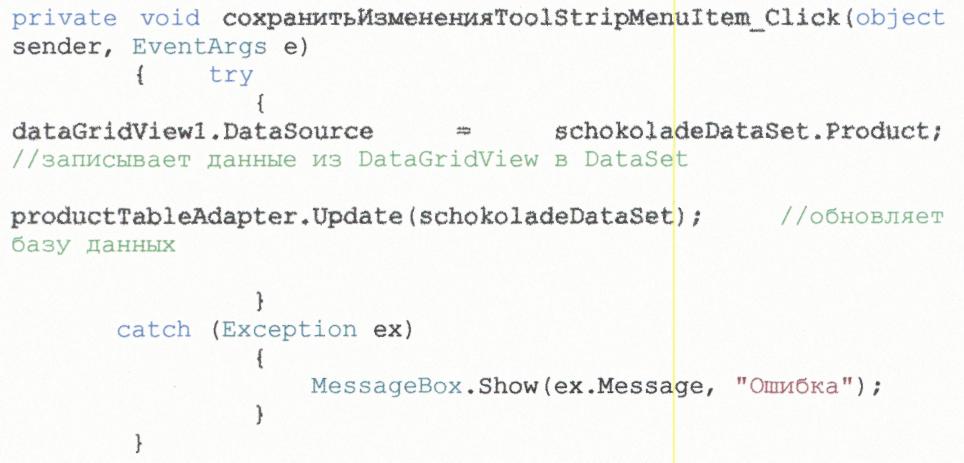


32. Параллельно с этим, установив удаление через определенную кнопку, нужно зайти в Параметры DataGridView (правая кнопка мыши на нем и пункт Properties) и установить False напротив свойства AllowUserToMeteRows. Этим действием мы уберем неоднозначности в командах.



### Обработка меню «Сохранить изменения»

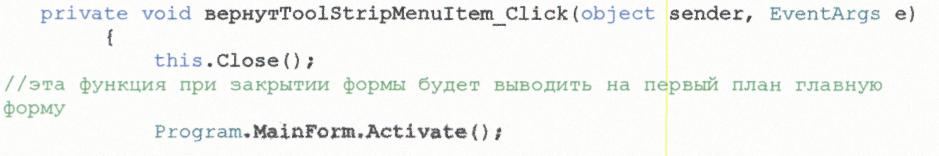
33. Для работы с пунктом меню *Сократить изменения,* надо внести следующий код:

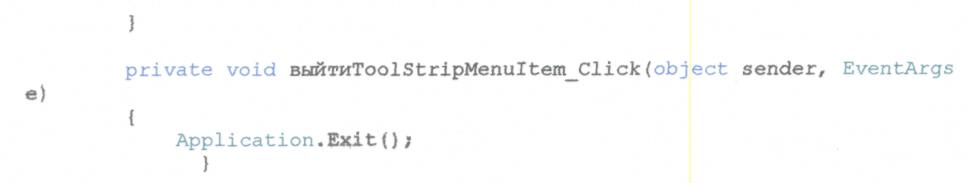


Тем самым прописывается редактирование записей в таблице. Надо отметить, что некорректность введенной информации контролируется самим SQL Server, что позволяет не прописывать многочисленные методы контроля ввода информации.

### Обработка меню «Вернуться»

34. Для пунктов меню «Вернуться» и «Выйти из проекта» прописываем следующий код:



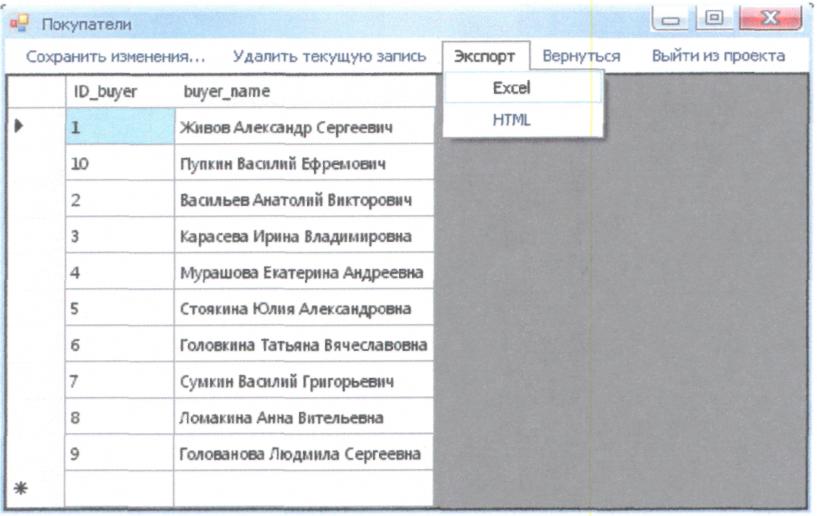


Теперь мы можем возвращаться на главную форму и закрывать проект из любой формы.

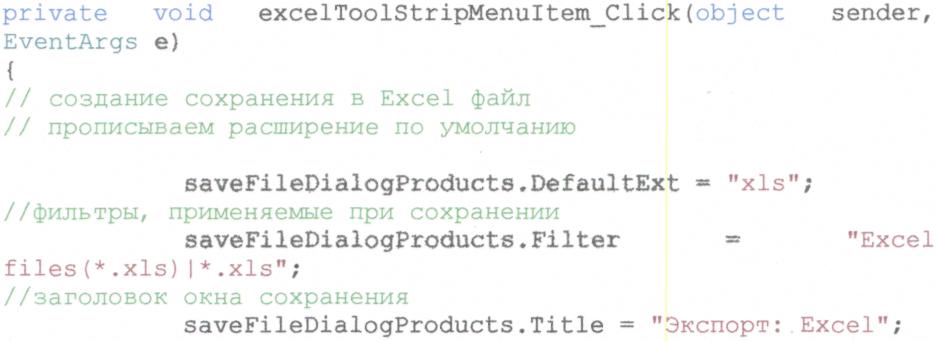
## **Меню «Экспорт»**

### Создание экспорта в Excel

35. Экспорт в Excel в данном приложении реализовуется в виде сохранения таблицы базы данных в виде Excel документа. Это удобно для конечного пользователя, так как позволяет продолжить работы с данными в Excel без затрачивания лишних усилий. В данном приложении экспорт находится в меню каждой выводимой таблицы.

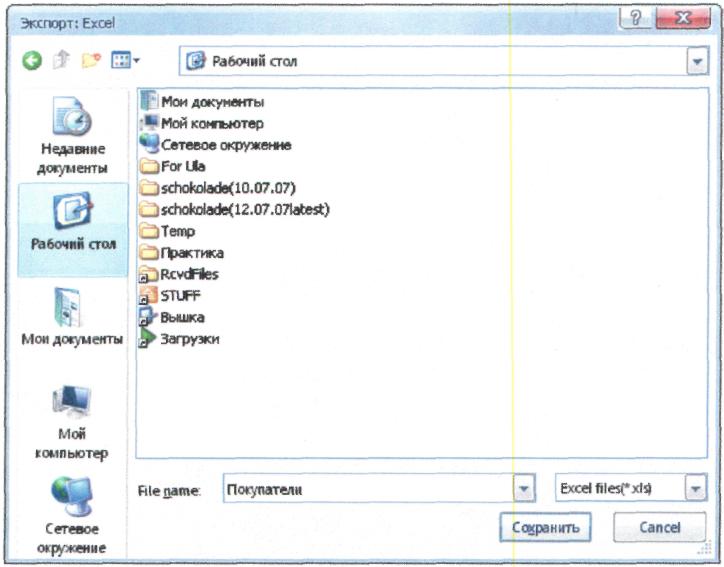


36. Прописывается он таким образом:

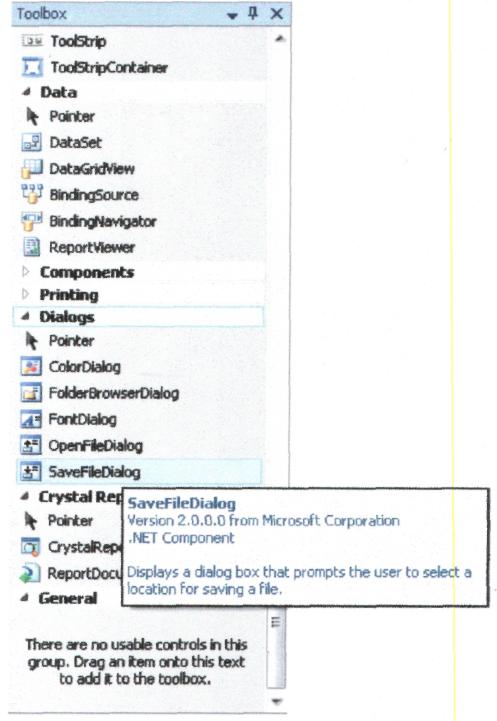




В результате, при нажатии кнопки «Экспорта в Excel» открывается окно сохранения с возможностью выбора места и введения имени (разрешение файла xls прописывается программно и изменению не подлежит).



37. Для того, чтобы это функция работала, необходимо добавить в форму с таблицей объект Toolbox SaveFileDialog (см. на следующей странице)



38. Для каждой таблицы SaveFileDialog отдельный. В нашем примере он называется *saveFileDialogProducts.* Так как на форме он не отобразится, найти его можно в меню скрытых функций формы.



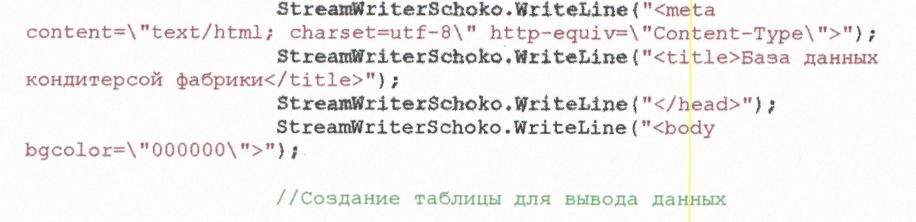
### 

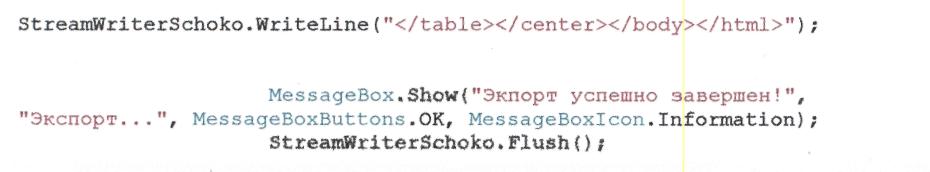
### Создание экспорта в HTML

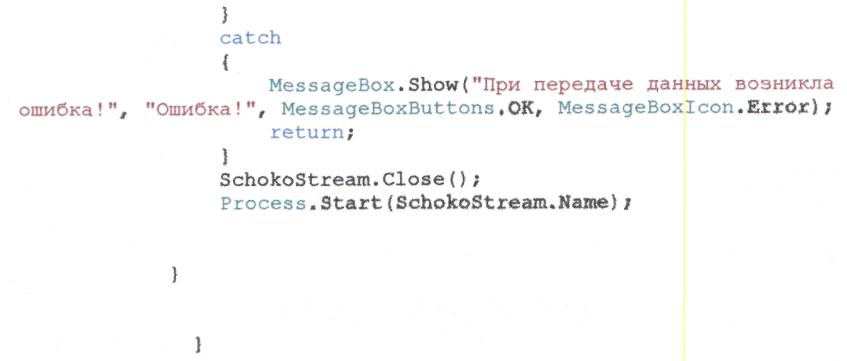
39. Для экспорта в HTML используем тот же элемент saveFileDialogProducts, что и в Excel (примечание: Products в названии это идентификатор названия таблица, у вас он будет другим).

40. Код для экспорта в HTML:







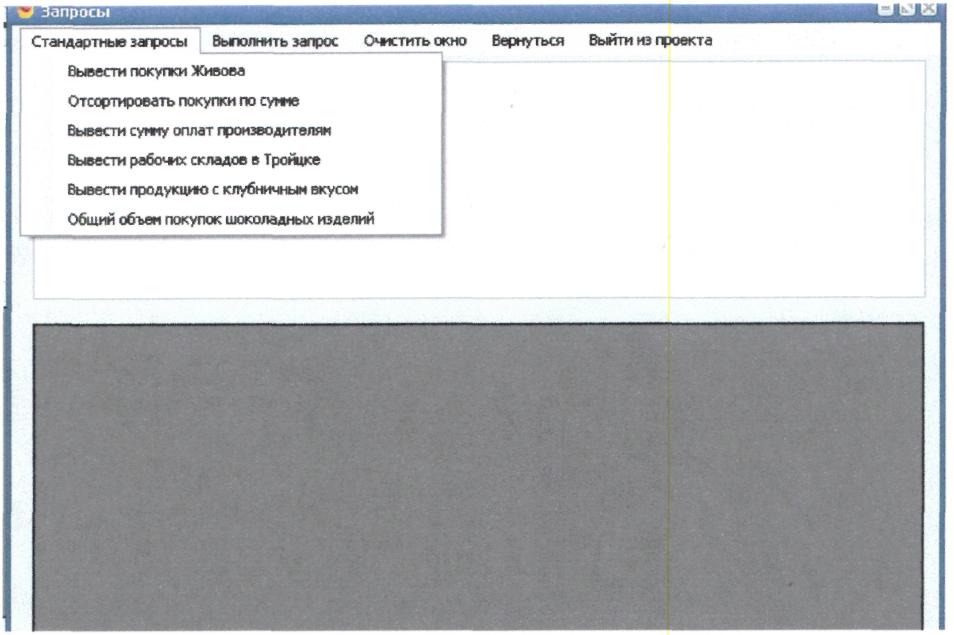


## **Обработка меню «Запросы»**

41. Сначала добавляем новую форму, в которой будем работать с запросами.

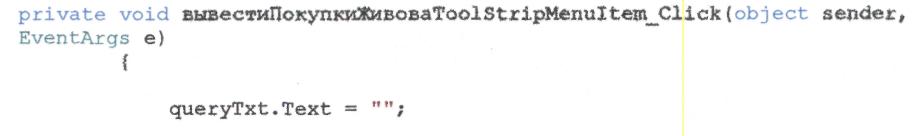
42. Удобно создать там текстовое окно, в которое будут вводиться SQL-коды запросов и таблицу типа DataGrid куда будет выводиться результат.

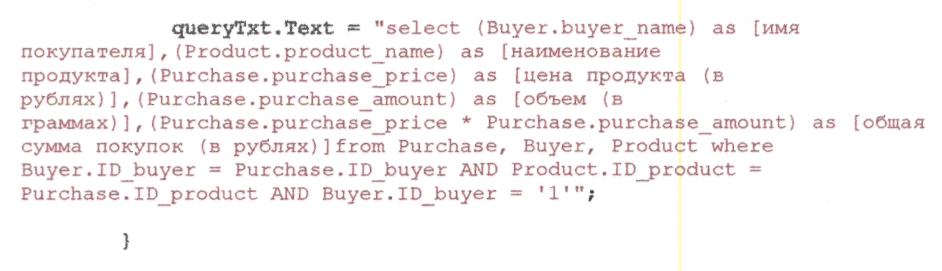
43. В форме необходимо сделать меню, в котором будут перечислены некоторые стандартные запросы. Приблизительно это должно выглядеть так:



44. Описание «Вернуться» и «Выйти из проекта» смотри выше (создание таблиц для работы с БД).

45. В каждый и стандартных запросов прописываем код:





В каждом запросе мы сначала очищаем текстовое окно, затем вводим в него текст запроса (при возможности его можно редактировать прямо в окне).

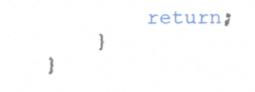
### Обработка кнопки «Выполнение запроса»

46. Для выполнения запроса из текстового окна нужно нажать кнопку «Выполнение запроса». Здесь пишем следующее:



47. Функцию Execute(); мы вынесли отдельно, котя это и не обязательно.

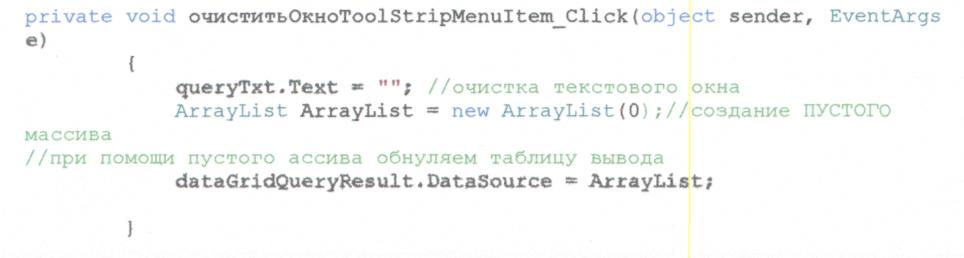




Теперь программа может выполнять запросы.

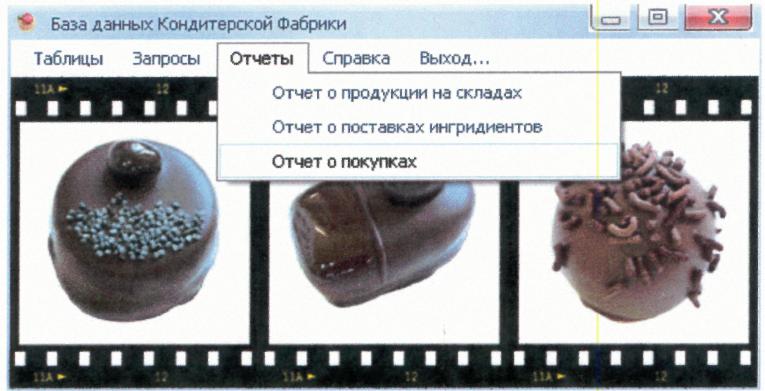
### Очистка текстового окна для произвольных запросов

48. Для того, чтобы пользователь мог ввести свой произвольный, нестандартный запрос, ему нужно чистое окно. Конечно, он может просто удались данные из тестового окна, но лучше предусмотреть для него функцию очистки. Итак, для пункта меню «Очистить окно» пишем:

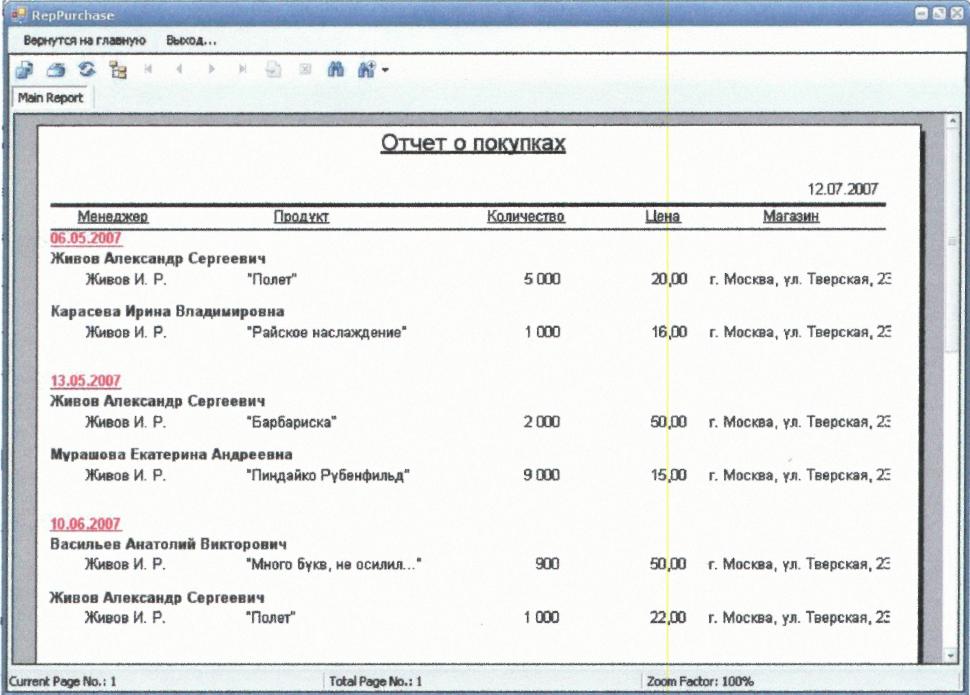


## **Создание отчетов с помощью GrystaiReports,**

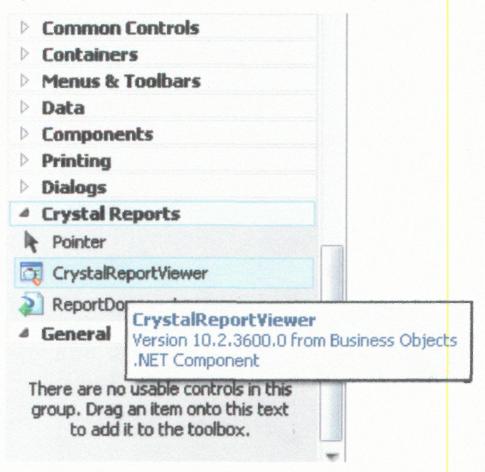
49. Для того, чтобы разобраться с созданием отчетов, создадим один из отчетов данного приложения. Он назвается «Отчет о покупках», Любой отчет выглядит, как форма, вызываемая из меню главной формы.



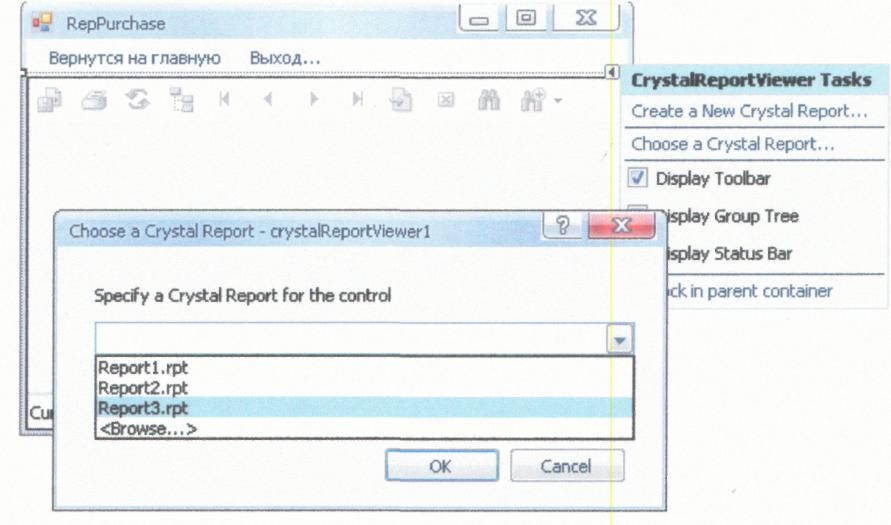
50. В конечном варианте этот отчет выглядит следующим образом:



51. Форма создается таким же образом, каким создается форма с таблицами. В MenuStrip прописаны функции возвращения на основную форму и выхода из приложения. На форму добавляется объект Toolbox CrystalRepoitViewer.

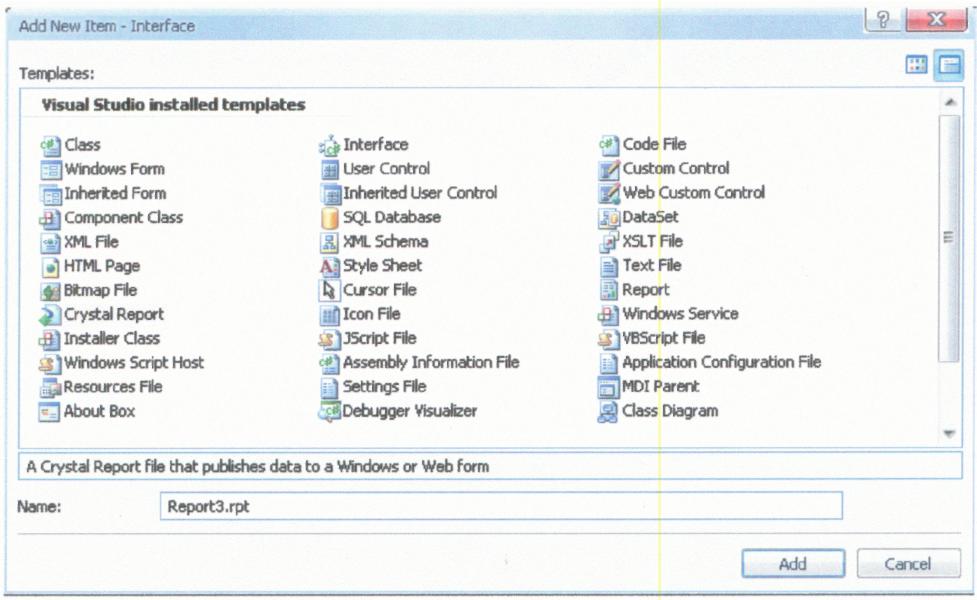


52. В меню CrystalReportViewer добавляется выводимый отчет.



53. Связь этой формы с основной прописывается аналогично связям форм с таблицами с основной формой.

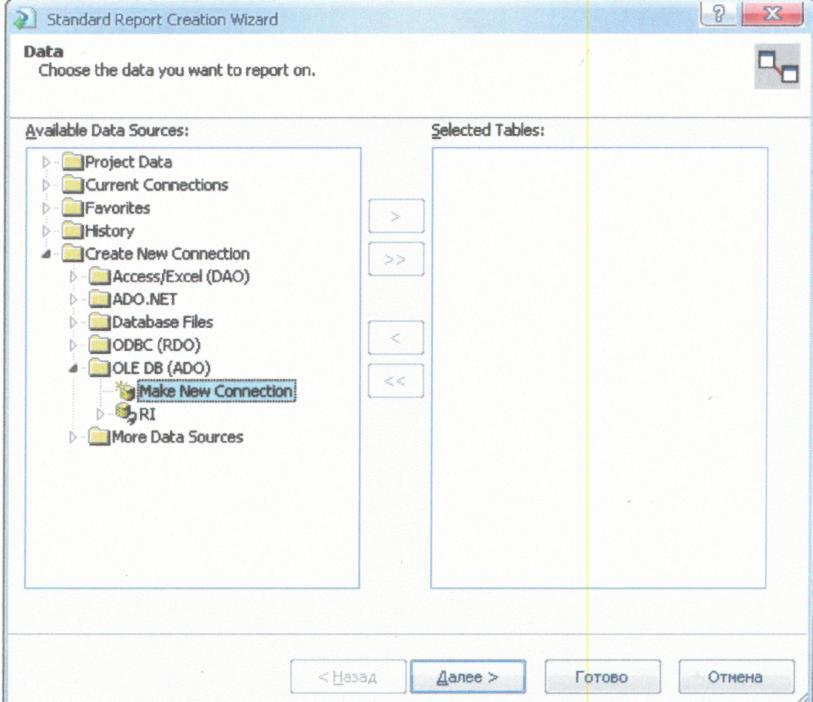
54. Для того, чтобы в выпадающем меню выбора отчетов таковые находились, необходимо их создать. Для этого нужно вызвать Project > Add New Item > CrystalRepoit



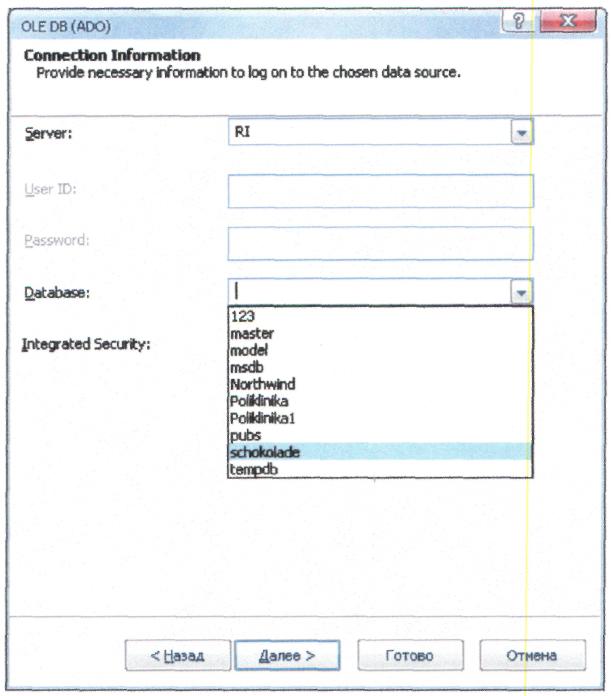
55. При добавлении нового отчета нужно выбрать создание отчета с помощью

конструктора.

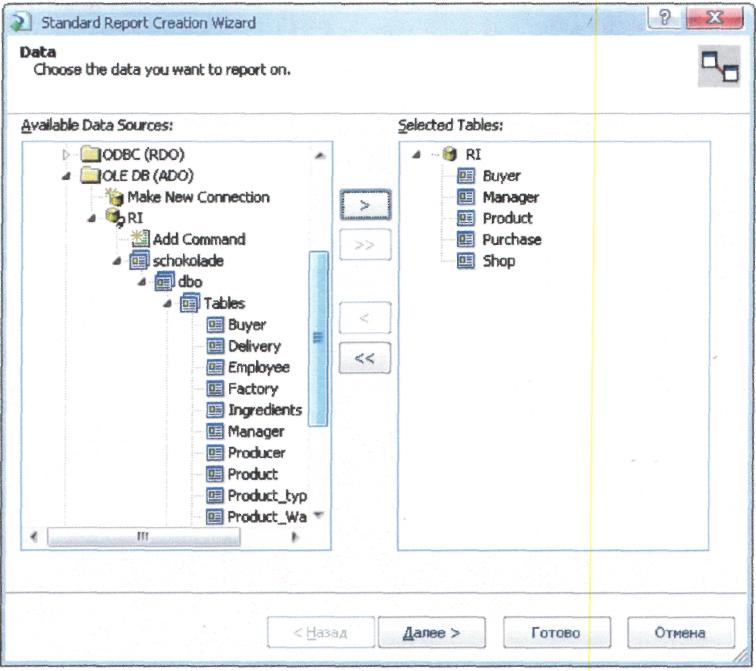
56. Далее установить соединение с базой данных. Делается это следующим образом:



57. Выбирается способ соединения *Microsoft OLE DB Provider For SQL Server* и вносится имя текущего сервера и название базы данных.

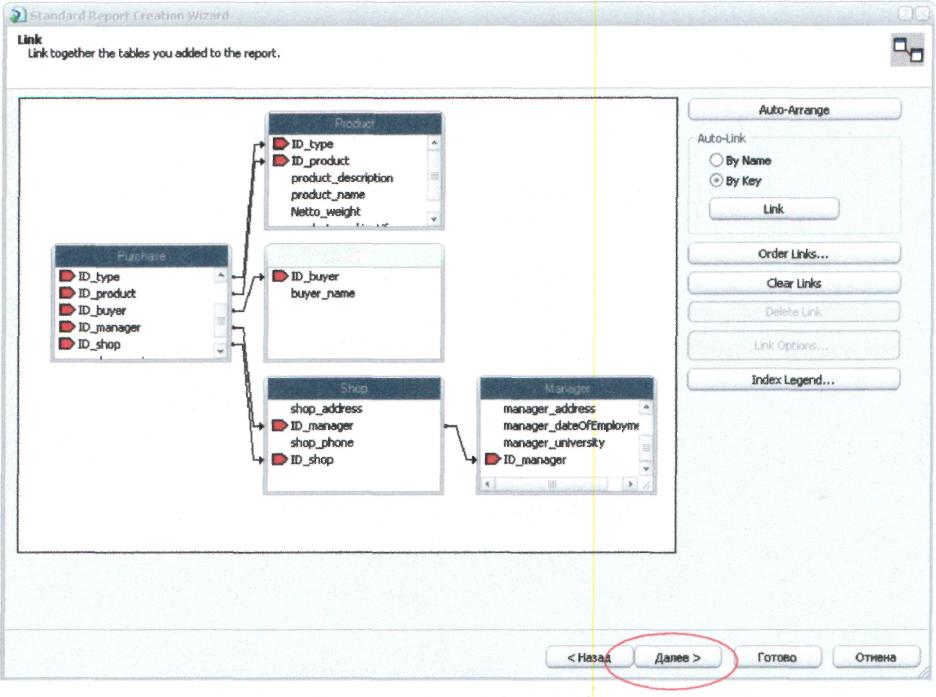


58. После установки соединения, в нем выбираются нужные таблицы.



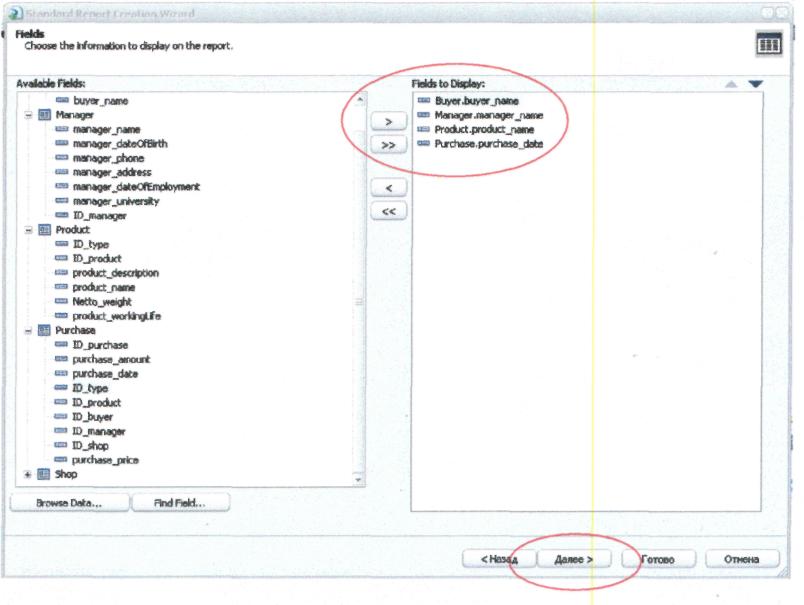
59. После нажатия на кнопку «Далее» в похожих окнах выбираются столбцы для отчета.

60. Вы увидите связи между выбраными таблицами:

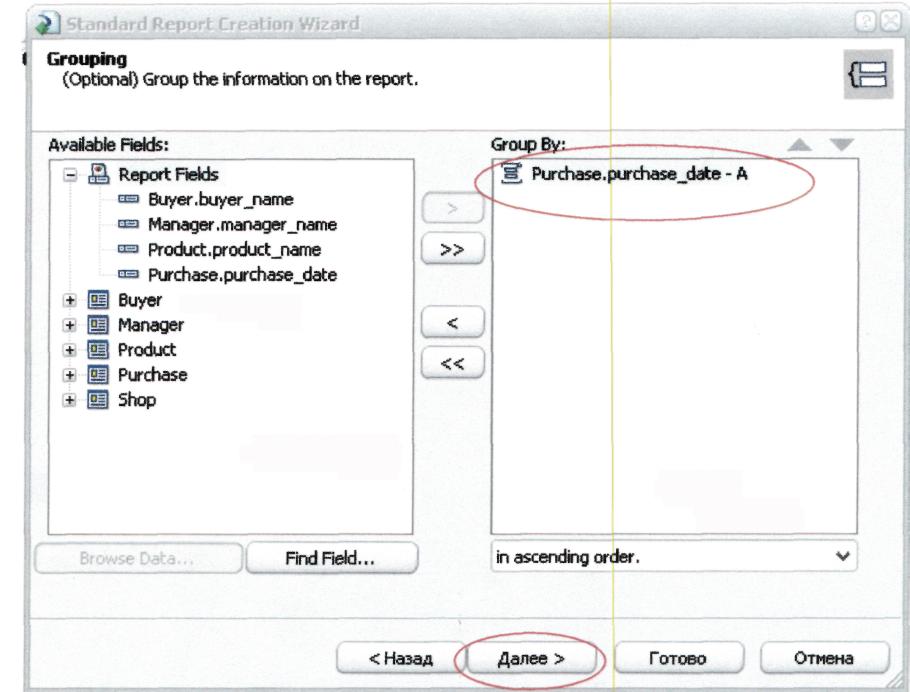


61. Жмем кнопку «Далее»

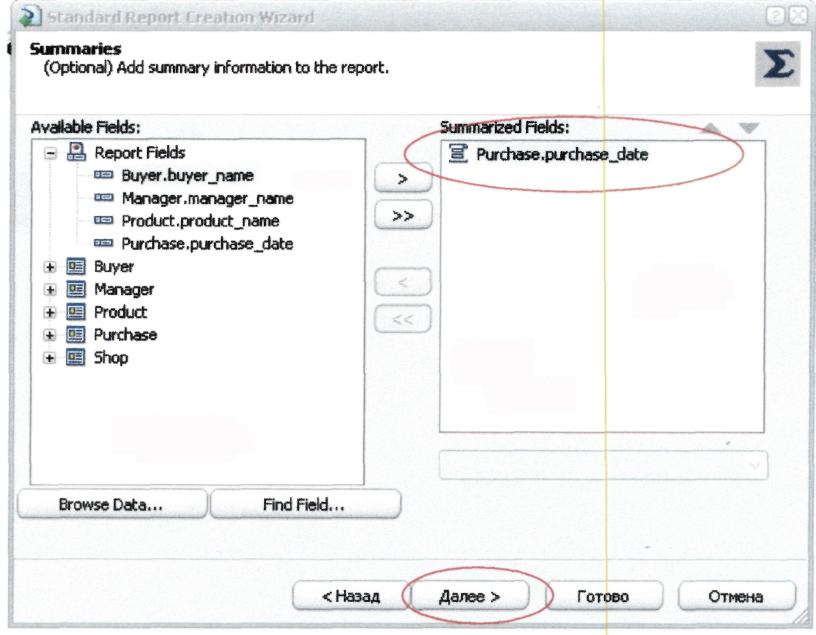
62. Выбираем поля для отображения в отчете и нажимаем кнопку «Далее»:



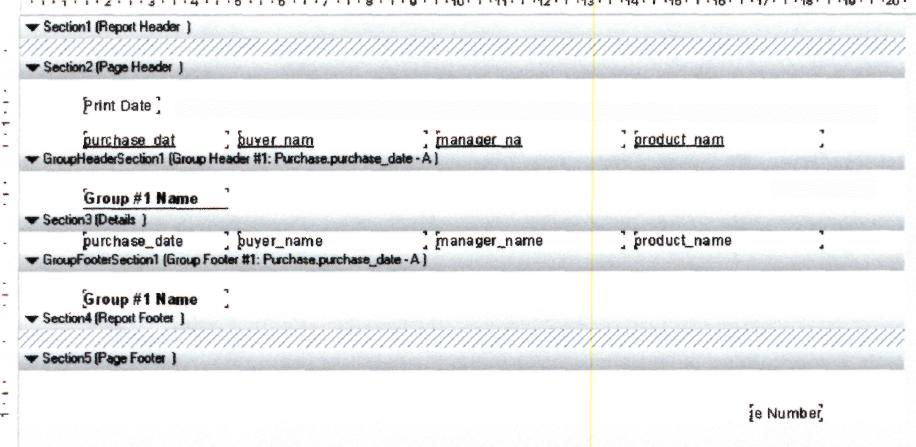
63. В следующем окне можно обозначить поля для группировок и нажать кнопку «Далее».



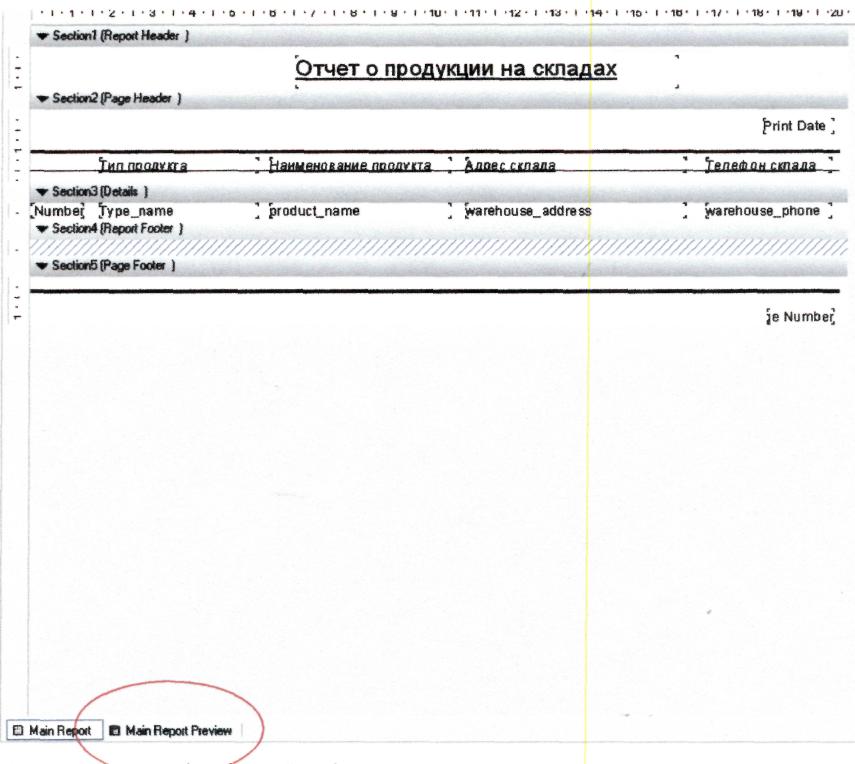
64. Если в отчете необходимы поля, по которым будет происходить суммирование, то их нужно отметить:



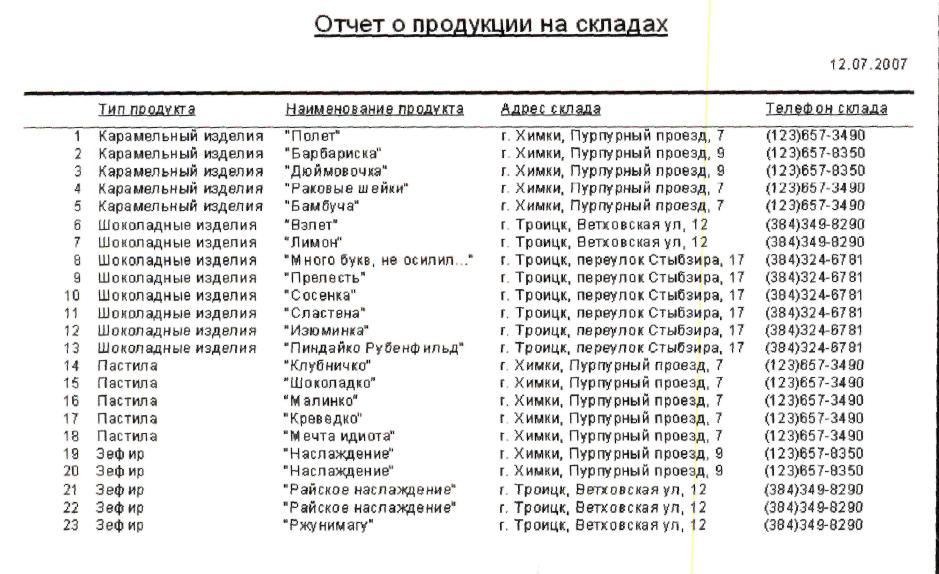
65. Дважды нажимаем кнопку «Далее», выбираем вид отчета (внешний) и затем нажимаем кнопку «Готово»:



66. Так выглядит форма для отчета. Необходимо настроить размер полей и добавить заголовок.

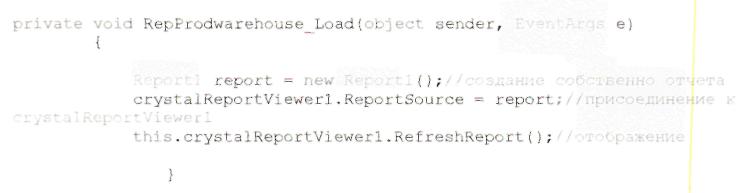


67. Переходим на main Report Preview и проверяем результат:

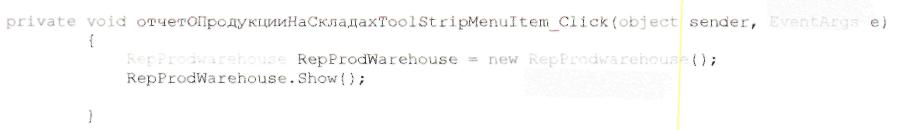


68. Отчет готов. Оталось его присоединить к меню.

69. В форме с отчетом прописываем:



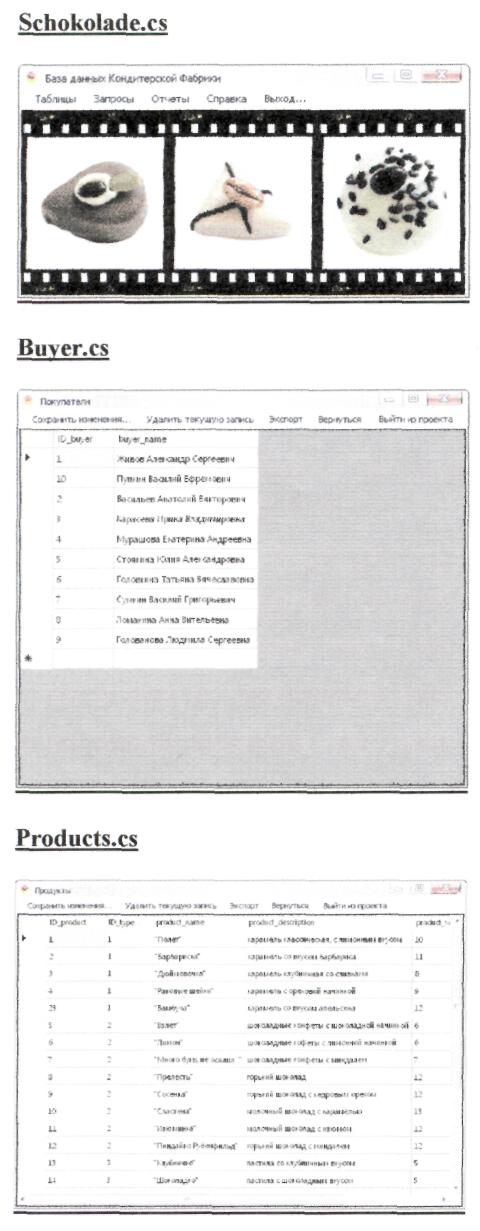
70. Теперь нужно указать отображение на вызов каждого отчета: (в главной форме)

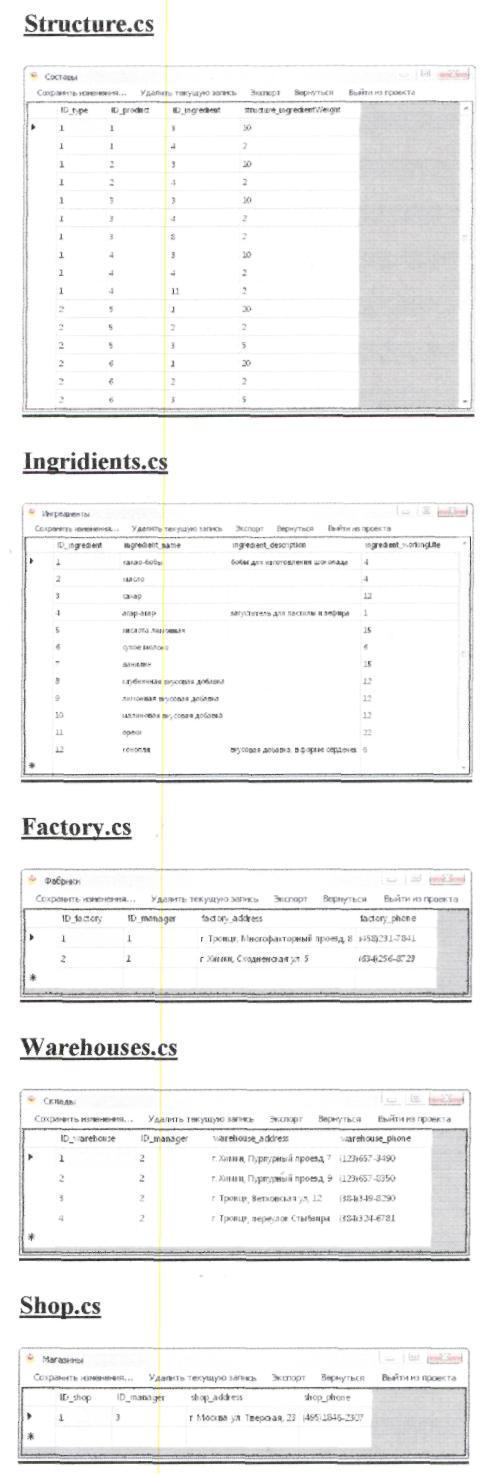


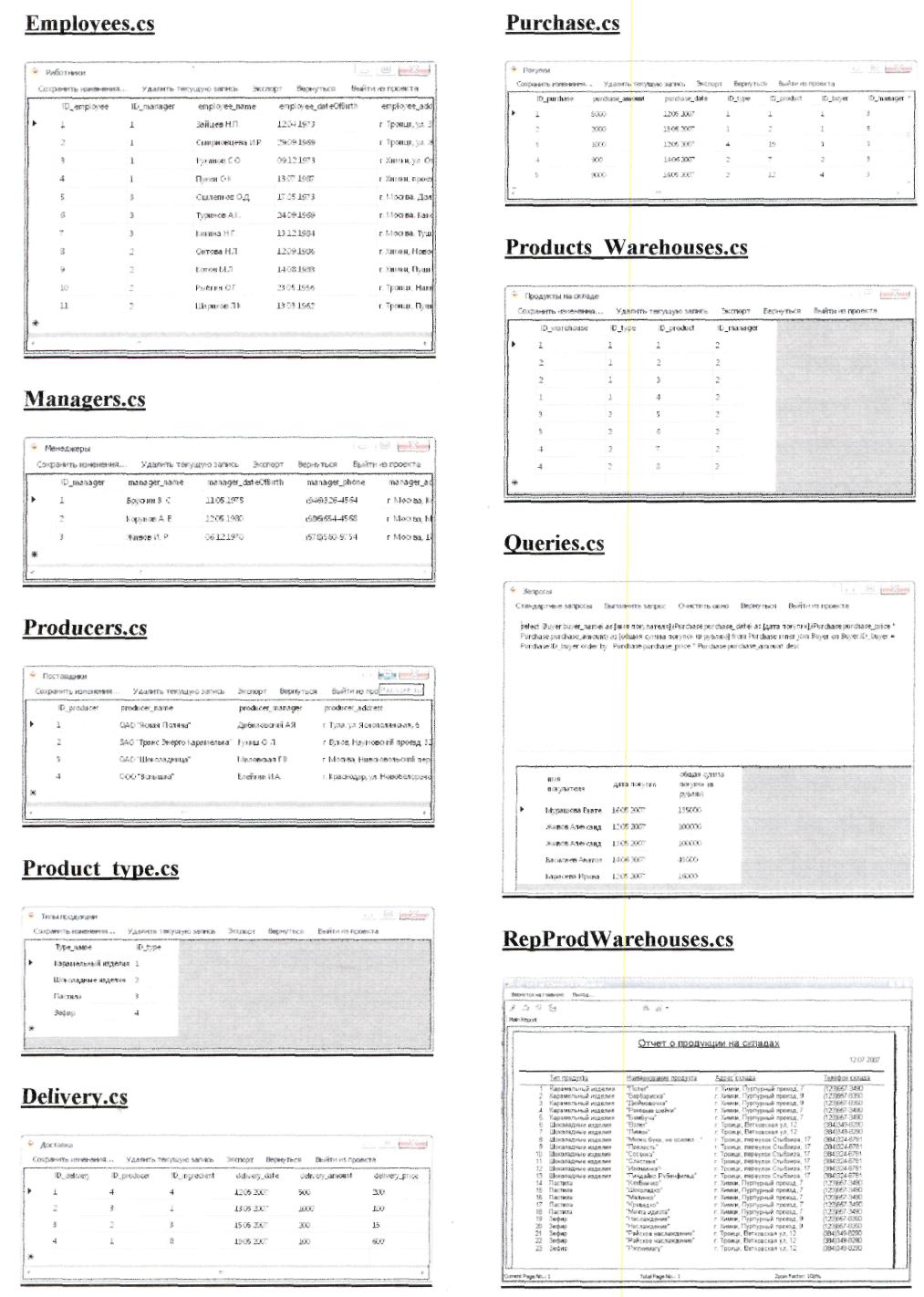
71. Не забудьте сделать кнопки/пункты меню для возрата на главную форму и выходя из приложения.

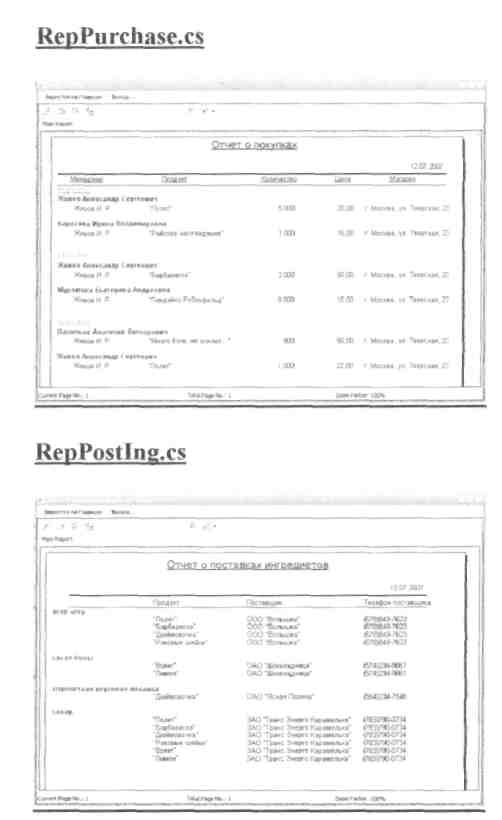
## **Приложение**

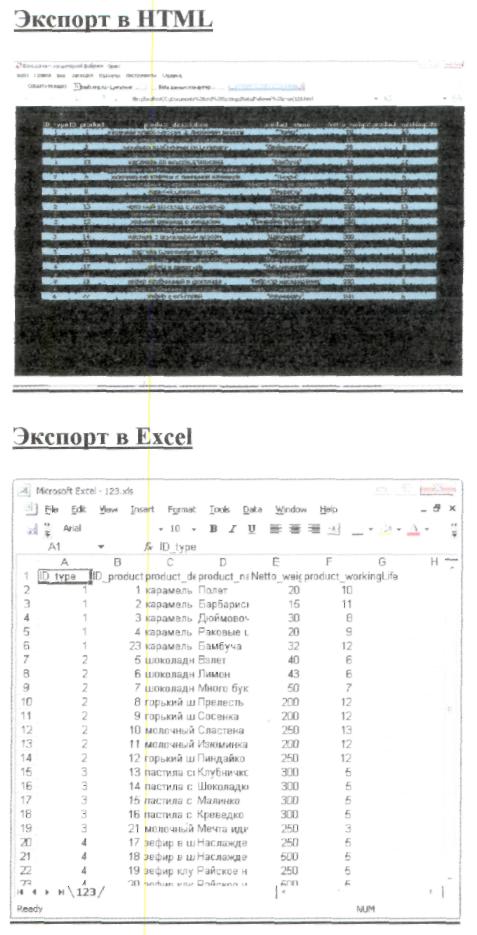
72. В качестве приложений приведем скриншоты всех форм, задействованных в рассмотренном приложении.











# 

# Заключение

В данном документе были описаны основные приемы создания интерфейса для работы с базам данных. Все функции, необходимыя для связи с SQL Server а также просто функции, необходимые для работы программы приведены и объяснены. Остальной дизайн и функциональность остаются на усмотрение разработчика.

**Приложение 3**

**Пример создания приложения к базе данных в среде C#.**



Отчет по практике

«Создание приложения для работы с базой данных»

Выполнил: (ФИО)

студент группы

2 курса факультета

«бизнес-информатика»

Проверил: (ФИО)

Г. Москва

200\_ г.

***СОДЕРЖАНИЕ:***

[***Возможности приложения*** 55](#_Toc191821515)

[***Создание главного меню*** 55](#_Toc191821516)

[***Связывание базы данных и формы Tables и отображение данных в DataGrid*** 56](#_Toc191821517)

[***Добавление новой записи в таблицу*** 61](#_Toc191821518)

[***Редактирование записей в таблицах*** 62](#_Toc191821519)

[***Удаление записей из таблиц*** 63](#_Toc191821520)

[***Составление и выполнение запросов*** 64](#_Toc191821521)

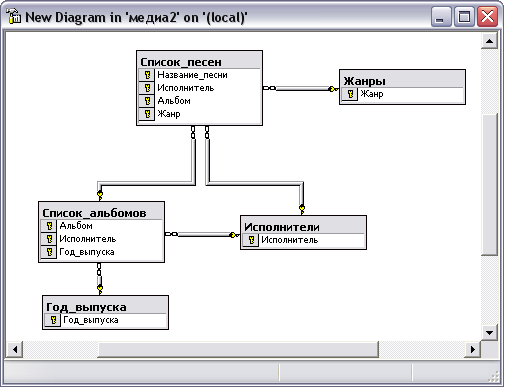
[***Создание отчетов*** 66](#_Toc191821522)

[***Экспорт данных в Excel*** 72](#_Toc191821523)

[***Экспорт данных в HTML*** 74](#_Toc191821524)

***Вступление***

Приложение «Фонотека» разрабатывалось в среде MS Visual Studio 2005 на языке Visual C# и предназначено для работы с базой данных «media2» (расположенной на сервере User1), созданной с помощью Enterprise Manager. БД содержит информацию о музыкальных композициях, исполнителях и альбомах. На рисунке ниже представлены содержащиеся в БД таблицы и связи между ними.



***Возможности приложения***

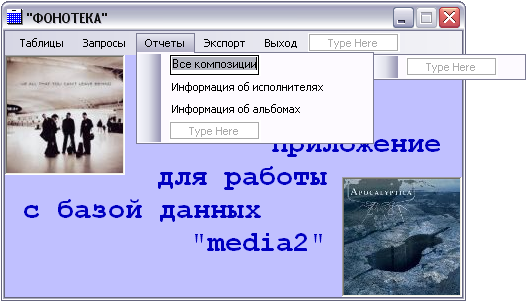
Программа «Фонотека» позволяет выполнять следующие действия:

* Просмотр таблиц и их редактирование, в том числе: добавление, удаление и изменение записей;
* Создание запросов к базе данных на языке SQL и их выполнение (также предусмотрен выбор из нескольких готовых запросов);
* Экспорт таблиц в Excel и в формат Web-страниц (HTML);
* Создание отчетов.

Подробнее функциональные возможности приложения и пути их реализации будут рассмотрены ниже.

***Создание главного меню***

Меню приложения создается довольно просто – на форму Main добавляется объект MenuStrip; пункты меню ввожу с клавиатуры:



Каждый пункт меню вызывает отдельную форму или событие:

* Таблицы – форму Tables,
* Запросы – форму Query,
* Отчеты
  + Все композиции,
  + Информация об исполнителях, формы RepSongs, RepSingers, RepAlbums
  + Информация об альбомах;
* Экспорт
  + В Excel – форму Export;
  + В HTML – форму ExportToHtml;
* Выход – выход из приложения.

Переход на другую форму (например, на Tables) осуществляется с помощью такого обработчика:

private void таблицыToolStripMenuItem\_Click(object sender, EventArgs e)

{

Tables tables = new Tables();

tables.Show();

this.Visible = false;

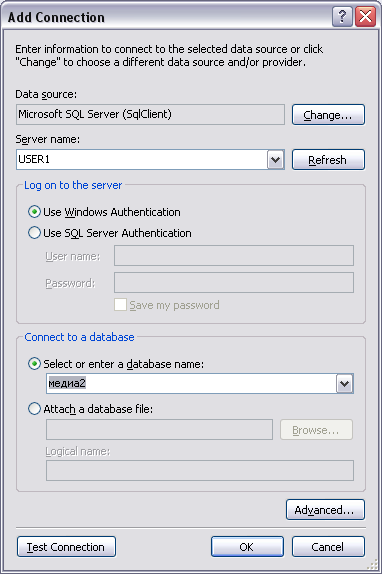
}

***Связывание базы данных и формы Tables и отображение данных в DataGrid***

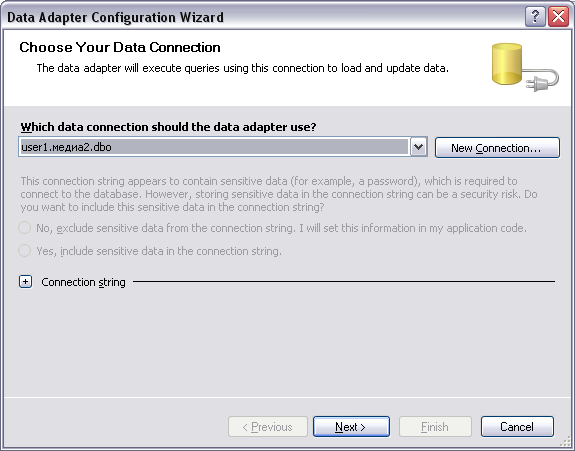
Связывание формы с БД – это ключевая операция, необходимая для корректной работы приложения. Просмотр и редактирование БД, выполнение запросов и экспорт осуществляется посредством доступа к SQL базе данных “media2”.

Рассмотрю подробнее механизм связывания.

После создания формы Tables добавляю на нее объект SqlConnection; в окне Properties в качестве значения ConnectionString выбираю New Connection и в открывшемся окне указываю необходимые параметры:



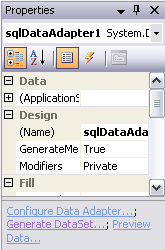
Следующий шаг – добавление SqlDataAdapter на форму. Как только объект помещается на форму, открывается окно, где необходимо указать базу данных:



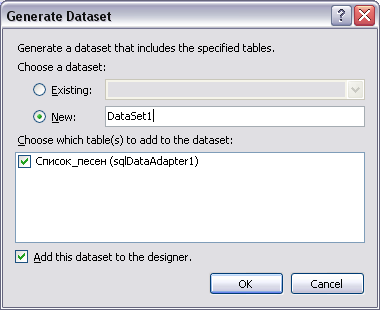
Далее мастер предлагает выбрать способ доступа к базе данных; выбираю Use SQL Statements и в следующем окне пишу нужный запрос. Например, чтобы добавить в DataSet информацию обо всех столбцах таблицы «Список\_песен», нужно написать такой запрос:



После нажатия Next мастер выполнит нужные операции или укажет на ошибку. После создания адаптера нужно сгенерировать для него набор данных. Для этого внизу окна Properties для созданного sqlDataAdapter1 нажимаю Generate DataSet.



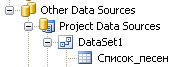
В открывшемся окне отображается список таблиц, добавленных в новый набор данных (dataSet11). Пока добавлена только одна таблица:



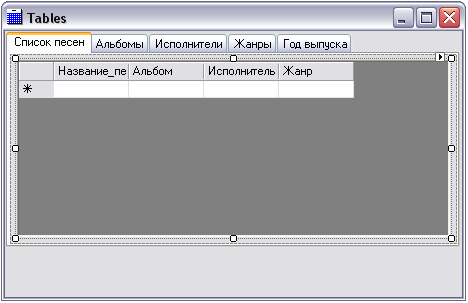
Теперь, чтобы при запуске приложения данные из БД «media2» загружались в форму, нужно в обработчик события Tables\_Load добавить следующую строку:

this.sqlDataAdapter1.Fill(dataSet11);

Таким образом, связывается БД и форма; однако данные пока не отображаются. Для их отображения добавляю на форму объект DataGrid, и в качестве его свойства DataSource выбираю Other Data Sources → Project Data Sources → DataSet1 → Список\_песен:



В результате готовая форма Tables выглядит так:



Чтобы поместить на одну форму все пять таблиц базы данных, я использую контейнер TabControl: на каждую tabPage я помещаю таблицу DataGrid и связываю ее с отдельным адаптером. Таким образом, на форме Tables содержится пять адаптеров, а в dataSet11 – все пять таблиц. После запуска приложения каждая таблица заполнится данными, если перед этим добавить в обработчик событий Tables\_Load следующие строки:

this.sqlDataAdapter1.Fill(dataSet11);

this.sqlDataAdapter2.Fill(dataSet11);

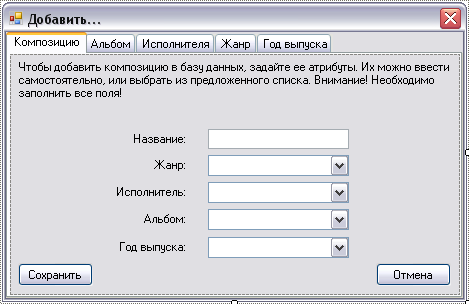
this.sqlDataAdapter3.Fill(dataSet11);

this.sqlDataAdapter4.Fill(dataSet11);

this.sqlDataAdapter5.Fill(dataSet11);

***Добавление новой записи в таблицу***

На форме Tables помещаю кнопку addBtn и в обработчике события addBtn\_Click пишу код, вызывающий форму AddRecord. Эта форма выглядит следующим образом:



Она также содержит контейнер TabControl; переключаясь между страницами, можно добавить новую запись в одну из пяти таблиц базы данных.

Рассмотрю подробнее механизм добавления новых записей на примере добавления записи в таблицу «Добавить композицию».

Прежде всего, необходимо создать SqlConnection для формы, пять SqlDataAdapter’ов (для каждой из пяти таблиц) и набор данных (DataSet1 общий для всего проекта). Только после этого можно продолжать работу. Эти объекты нужны для добавления, изменения и удаления записей.

Поля формы можно заполнить, либо введя новое значение, либо выбрав одно из предложенных в ComboBox’ах (причем эти значения обновляются в течение редактирования). Чтобы добавить в таблицу «Список\_песен» новую запись, следует в обработчик addBtn\_Click добавить такой код:

DataRow rowB = dataSet11.Tables["Список\_песен"].NewRow();

rowB["Название\_песни"] = this.textBox1.Text;

rowB["Альбом"] = this.comboBox3.Text;

rowB["Исполнитель"] = this.comboBox2.Text;

rowB["Жанр"] = this.comboBox1.Text;

dataSet11.Tables["Список\_песен"].Rows.Add(rowB);

sqlDataAdapter1.Update(dataSet11);

Однако в БД между таблицами существуют связи, и в самих таблицах существуют ограничения. Поэтому, чтобы успено добавить запись, нужно заполнить *все* поля формы, иначе программа сообщит об ошибке. Далее, если пользователь пытается ввести новое значение, которого нет в связанной таблице, то нужно сначала добавить это значение в связанную таблицу. Например, пользователь пытается в таблице «Список\_песен» в поле «Альбом» добавить значение, которого нет в связанной таблице «Список\_альбомов».

Для обработки подобных ситуаций я использую такой алгоритм:

if

ВСЕ поля НЕпусты

{

try

{

Добавить новую записть в таблицу «Жанр» из поля;

}

Catch (срабатывает, если така запись уже существует)

{

}

try

{

Добавить новую записть в таблицу «Исполнитель» из поля;

}

catch

{

}

try

{

Добавить новую записть в таблицу «Год\_выпуска» из поля;

}

catch

{

}

try

{

Добавить новую записть в таблицу «Список\_альбомов» из трех полей;

}

catch

{

}

Добавить новую запись в таблицу «Список\_песен»;

Очистить все поля;

}

else

MessageBox.Show("Заполните ВСЕ поля");

}

Подобным образом записи добавляются и в другие таблицы.

***Редактирование записей в таблицах***

В процессе просмотра данных в форме Tables в DataGrid’ах можно редактировать данные. Однако, если просто внести изменения в DataGrid, реальных изменений в самой базе данных не произойдет. Для того, чтобы сохранить изменения, добавляю на форму кнопку editBtn и в обработчике editBtn\_Click прописываю такой код:

private void editBtn\_Click(object sender, EventArgs e)

{

if (MessageBox.Show("Сохранить изменения в таблицах?", "???", MessageBoxButtons.YesNo, MessageBoxIcon.Question, MessageBoxDefaultButton.Button1) == DialogResult.Yes)

{

try

{

dataGrid1.DataSource = dataSet11.Tables[0];

sqlDataAdapter1.Update(dataSet11);

dataGrid2.DataSource = dataSet11.Tables[1];

sqlDataAdapter2.Update(dataSet11);

dataGrid3.DataSource = dataSet11.Tables[2];

sqlDataAdapter3.Update(dataSet11);

dataGrid4.DataSource = dataSet11.Tables[3];

sqlDataAdapter4.Update(dataSet11);

dataGrid5.DataSource = dataSet11.Tables[4];

sqlDataAdapter5.Update(dataSet11);

}

catch

{

MessageBox.Show("Проерьте правильность введенных данных.\r\nВозможно, сначала необходимо\r\nдобавить новые данные в базу данных");

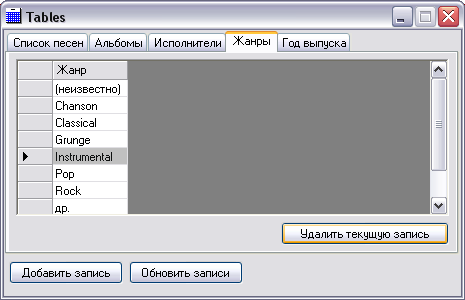
}

}

}

***Удаление записей из таблиц***

Прежде всего, стоит отметить, что, если обработчики кнопок «Добавить запись» и «Обновить записи» действуют сразу для пяти таблиц, то кнопка «Удалить запись» своя для каждой страницы TabControl.



Это вызвано тем, что при щелчке на кнопку удаляется текущая запись таблицы (а она для каждой таблицы установлена по умолчания как первая). Если пользователь пытается удалить связанную запись, программа сообщит об ошибке.

Для удаления записи использую такой код (удаление из первой таблицы):

private void button1\_Click(object sender, System.EventArgs e)

{

try

{

CurrencyManager cm = (CurrencyManager)dataGrid1.BindingContext[dataGrid1.DataSource];

if (cm.Count > 0)

if (MessageBox.Show("Удалить текущую строку?", "???", MessageBoxButtons.YesNo, MessageBoxIcon.Question, MessageBoxDefaultButton.Button1) == DialogResult.Yes)

cm.RemoveAt(cm.Position);

sqlDataAdapter1.Update(dataSet11);

}

catch

{

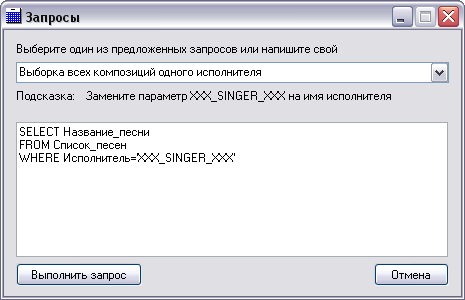
MessageBox.Show("Ой! Произошла ошибка! Возможно, Вы пытаетесь удалить связанную запись. ");

}

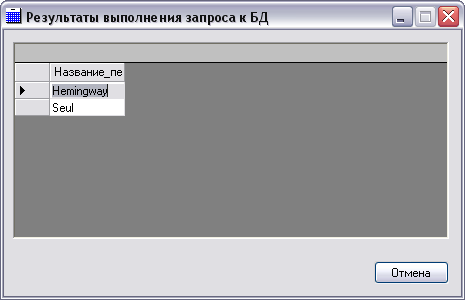
}

***Составление и выполнение запросов***

Прежде всего, помещаю на форму Query объекты SqlConnection и SqlComand. Также добавляю объекты ComboBox и TextBox. С помощью ComboBox’а пользователь может выбрать готовый запрос, который на языке SQL отобразится в TextBox’е (может понадобиться ввести необходимые параметры) или написать в TextBox’е свой запрос.



Запрос передается на обработку после клика по кнопке «Выполнить запрос» (Name=executeBtn), после чего открывается форма QueryResult c результатом запроса, или выводится сообщение о принятых изменениях.



Если в тексте запроса допущена какая-либо ошибка, приложение выдаст MessageBox c сообщением.

В обработчике executeBtn пишу такой код:

SqlDataReader ds;

if (this.textBox1.Text == "")

{

MessageBox.Show("Введите, пожалуйста, запрос!");

return;

}

SqlConnection sqlConnection1 = new SqlConnection(this.sqlConnection1.ConnectionString);

sqlConnection1.Open();

try

{

SqlCommand comm = new SqlCommand(this.textBox1.Text, sqlConnection1);

ds = comm.ExecuteReader();

}

catch

{

MessageBox.Show("В тексте запроса допущена ошибка! ");

return;

}

QueryResult result = new QueryResult();

result.Prepare(ds);

ds.Close();

result.Show();

this.Visible = false;

}

Форма ResultQuery одержит функцию Prepare():

public void Prepare(SqlDataReader ds)

{

View=ds;

DataTable dt;

int i;

list = new ArrayList(0);

foreach (DbDataRecord d in ds)

{

list.Add(d);

}

this.dataGrid1.DataSource = list;

this.dataGrid1.ReadOnly = true;

this.dataGrid1.AllowSorting = true;

if (ds.RecordsAffected != -1)

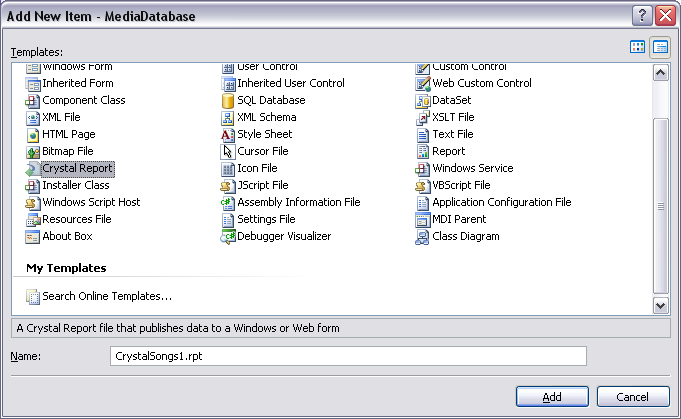
MessageBox.Show("Внесены изменения в " + ds.RecordsAffected + " записей");

}

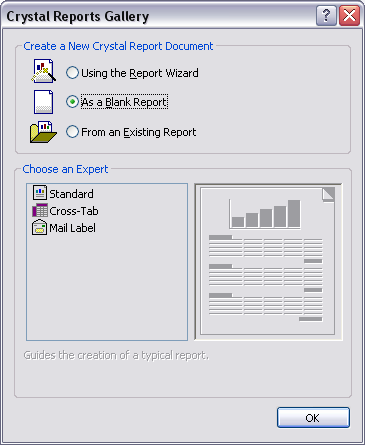
***Создание отчетов***

В моем приложении есть три отчета: «Информация обо всех композициях», «Информация обо всех исполнителях» и «Информация обо всех альбомах». Я рассмотрю создание отчетов на примере отчета по исполнителям.

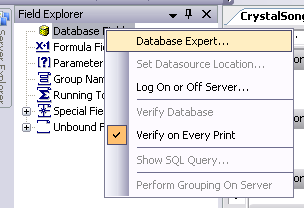
Создаю новую форму (Name=RepSongs), и помещаю на нее объект CrystalReportViewer. Кликая на название проекта в Solution Explorer правой кнопкой, выбираю Add New Item → Add и в открывшемся окне выбираю Crystal Report:



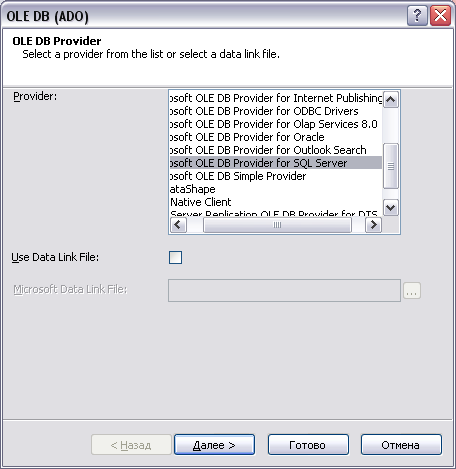
Появляется окно лицензионного соглашения, после чего предлагается выбрать способ создания нового отчета; выбираю пункт As a Blanc Report:



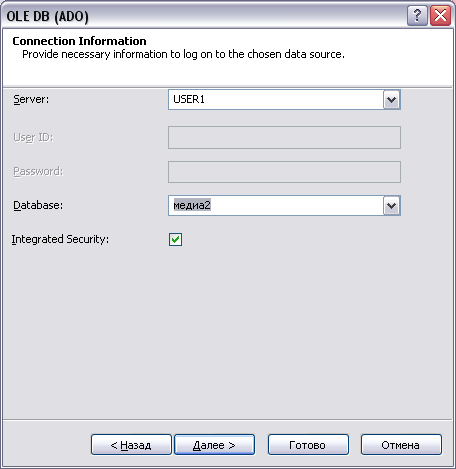
После создания нового отчета слева появляется окно Field Explore, и в нем я выбираю пункт Database Fields → Database Expert:



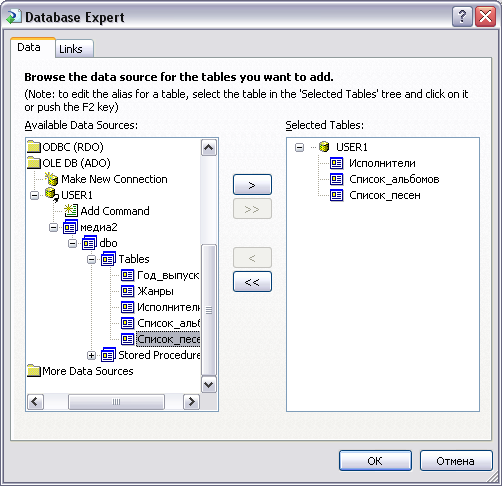
В появившемся окне выбираю Create New Connection → OLE DB (ADO) → Make New Connection, после чего из списка выбираю Microsoft OLE DB Provider for SQL Server:



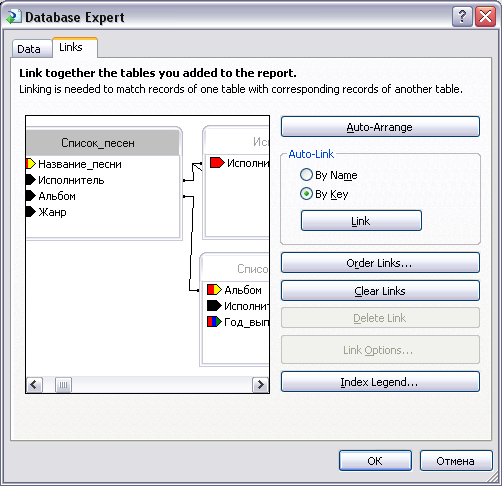
В следующем окне задаю имя своего сервера (USER1) и название базы данных и нажимаю «Готово»:



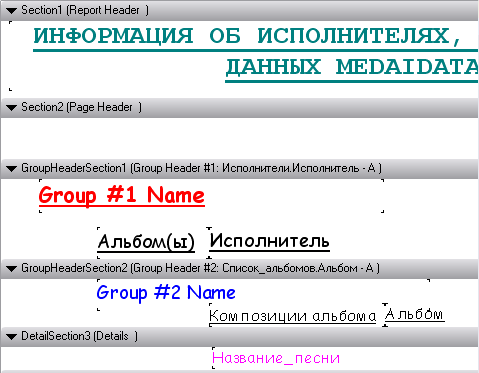
Снова открываю окно Database Expert. Из появившегося списка выбираю те таблицы, которые будут мне нужны при создании отчета:



Если выбрано более одной таблицы, то во вкладке Links следует уточнить связи между таблицами:



После этого можно уже непосредственно составлять отчет.



После составления шаблона отчета, надо подключить его к форме. Для этого добавим в обработчик события загрузки формы следующий код:

private void RepSongs\_Load(object sender, EventArgs e)

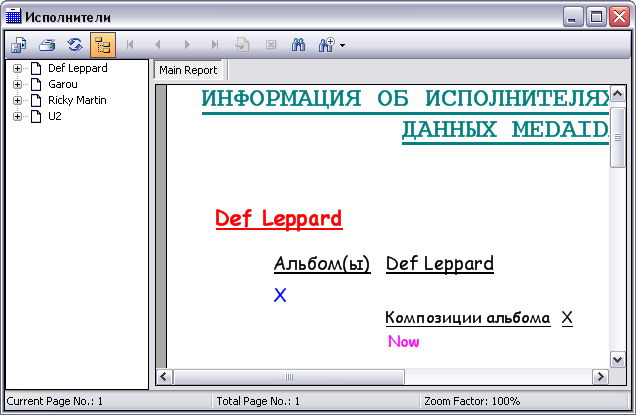
{

CrystalSingers1 songRep = new CrystalSingers1();

crystalReportViewer1.ReportSource = songRep;

}

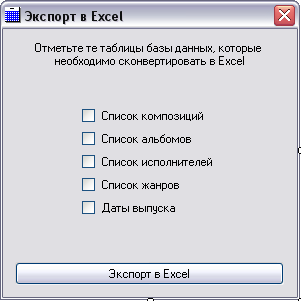
Отчет подключен, и теперь при клике на пункт меню Отчеты → Информация обо всех исполнителях открывается форма, в которой содержится готовый отчет:



Отсюда можно сохранить отчет в различных форматах (.rpt, .pdf, .xls, .doc, .rft), распечатать его или обновить.

***Экспорт данных в Excel***

При выборе пункта меню Экспорт → в Excel открывается окно Export, в котором предлагается с помощью флажков выбрать таблицы для экспорта. После нажатия кнопки «Экспорт в Excel» (Name=expExcelBtn) открываются одно или несколько новых приложений Excel с данными в зависимости от количества отмеченных таблиц.



Код обработчика expExcelBtn1\_Click выглядит так:

private void expExcelBtn1\_Click(object sender, EventArgs e)

{

if (this.checkBox1.Checked == true)

{

Excel.Application excel = new Excel.Application();

int rowIndex = 1;

int colIndex = 0;

excel.Application.Workbooks.Add(true);

DataTable table = GetData();

foreach (DataColumn col in table.Columns)

{

colIndex++;

excel.Cells[1, colIndex] = col.ColumnName;

}

foreach (DataRow row in table.Rows)

{

rowIndex++;

colIndex = 0;

foreach (DataColumn col in table.Columns)

{

colIndex++;

excel.Cells[rowIndex, colIndex] = row[col.ColumnName].ToString();

}

}

excel.Visible = true;

}

if (this.checkBox2.Checked == true)

{

Аналогичный код для таблицы «Список\_альбомов», используется GetData1();

}

....................

}

Используемая в обработчике функция GetData() существует в пяти экземплярах (GetData(), GetData1(), GetData2(), GetData3 и GetData4() для каждой из пяти таблиц). Вариант для первой таблицы («Список\_песен») выглядит так:

private System.Data.DataTable GetData()

{

try

{

sqlDataAdapter1.Fill(dataSet21, "Список\_песен");

}

catch (Exception ex)

{

MessageBox.Show(ex.ToString());

}

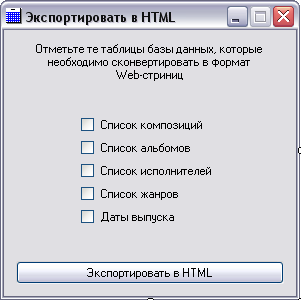
return dataSet21.Tables[0];

}

Разумеется, и в этой форме используются объекты SqlConnection, SqlDataAdapter (5 экземпляров) и DataSet.

***Экспорт данных в HTML***

Как и при экспорте в Excel, в окне ExportToHtml предлагается выбрать таблицы:



После выбора нужных таблиц и нажатия кнопки внизу формы в корневой директории приложения (в проекте это папка /bin/debug/) создается (или обновляется) файл tables.htm, содержащий выбранные таблицы. Если пользователь не выделил ни одной таблицы, программа сообщает об ошибке.

Файл tables.htm создается путем простого построчного заполнения текстового файла html-дескрипторами. Данные берутся из DataSet, также используется набор функций GetData().

Фрагмент кода представлен ниже:

int i,j;

string name="tables.htm";

FileStream file = new FileStream(name, FileMode.Create);

StreamWriter wr = new StreamWriter(file);

wr.WriteLine("<HTML>");

wr.WriteLine("<HEAD>");

wr.WriteLine("<TITLE>Generated file</TITLE>");

wr.WriteLine("</HEAD>");

wr.WriteLine("<BODY BGCOLOR=#87CEFA TEXT=#0000FF>");

wr.WriteLine("<FONT FACE=Comic Sans MS>");

if (this.checkBox1.Checked == true)

{

DataTable table = GetData();

wr.WriteLine("<CENTER><B>Таблица &laquo" + table.TableName + "&raquo</B></CENTER>");

wr.WriteLine("<BR>");

wr.WriteLine("<CENTER><TABLE cols=" + table.Columns.Count + " cellspacing=1px border=1 BGCOLOR=#B0C4DE>");

for (j = 0; j < table.Columns.Count; j++)

{

wr.WriteLine("<td>");

wr.WriteLine("<B>" + table.Columns[j].ColumnName+"</B>");

wr.WriteLine("</td>");

}

for (i = 0; i < table.Rows.Count; i++)

{

wr.WriteLine("<tr>");

for (j = 0; j < table.Columns.Count; j++)

{

wr.WriteLine("<td>");

wr.WriteLine("" + table.Rows[i][j]);

wr.WriteLine("</td>");

}

wr.WriteLine("</tr>");

}

wr.WriteLine("</TABLE></CENTER>");

}

....................

wr.WriteLine("</FONT>");

wr.WriteLine("</BODY>");

wr.WriteLine("<HTML>");

wr.Flush();

file.Close();

***Литература***

1. Кузьменко В.Г. Базы данных в Visual Basic и VBA. – Изд. «Бином», Москва, 2004, 412 с.
2. Перминов Г.И. Методические материалы к проведению практических занятий. ВШЭ, 2005
3. Скотт Ф. Баркер. Создание приложений баз данных в среде Visual Basic.Net и ADO.Net. Советы, рекомендации, примеры. – Изд. «Вильямс», Москва, 2003. 550 с.

4. Лабор В.В. Си Шарп: Создание приложений для Windows/ -Мн.: Харвест, 2003 – 384 с.

5. Фролов А.В., Фролов Г.В. Визуальное проектирование приложений С#. –М.:КУДИЦ-ОБРАЗ, 2003 -512 с.