МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Пятигорский институт (филиал) СКФУ

Методические указания

по выполнению лабораторных работ
по дисциплине
«ПРОЕКТНЫЙ ПРАКТИКУМ»
для направления подготовки **09.03.02 Информационные системы и технологии**
направленность (профиль) **Информационные системы и технологии обработки цифрового контента**

Пятигорск
2025

ВВЕДЕНИЕ

ЦЕЛЬ И ЗАДАЧИ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Целью изучения дисциплины является формирование набора общенаучных и профессиональных компетенций будущего бакалавра по направлению 09.03.02 «Информационные системы и технологии».

Задачи освоения дисциплины:

- приобретение практических навыков комплексного использования методов, инструментальных средств проектирования и сопровождения информационных систем; навыков управления ИТ- проектами;
- освоение методик проектирования обеспечивающих подсистем ИС и расчета экономической эффективности ИТ-проекта.

2.Наименование лабораторных работ

№ Темы дисциплины	Наименование тем дисциплины, их краткое содержание	Объем часов	Из них практическая подготовка, часов
8 семестр			
1.	Лабораторная работа 1. Проектирование ПО с применением UML.	1	
2.	Лабораторная работа 2. Построение моделей в BPWin. Построение IDEF0 диаграмм.	1	
3.	Лабораторная работа 3. Построение моделей в BPWin. Построение IDEF3, DFD диаграмм.	1	
4.	Лабораторная работа 4. Построение моделей в ERWin	1	
5.	Лабораторная работа 5. Создание и заполнение таблиц в Microsoft SQL Server 2012.	1	
6.	Лабораторная работа 6. Создание запросов и фильтров в Microsoft SQL Server 2012.	1	
7.	Лабораторная работа 7. Создание динамических запросов при помощи хранимых процедур в Microsoft SQL Server 2012.	1	
8.	Лабораторная работа 8. Создание функций пользователя в Microsoft SQL Server 2012.	1	
9.	Лабораторная работа 9. Обеспечение целостности данных в Microsoft SQL Server 2012. Создание диаграмм и триггеров.	1	
10.	Лабораторная работа 10. Создание простых ленточных форм для работы с базами данных в Visual Studio 2012.	1	
11.	Лабораторная работа 11. Создание сложных ленточных форм для	1	

	работы с базами данных в Visual Studio 2012		
12.	Лабораторная работа 12. Создание табличных форм для работы с базами данных в Visual Studio 2012.	1	
13.	Лабораторная работа 13. Создание отчетов в Visual Studio 2012.	1	
14.	Лабораторная работа 14. Выполнение индивидуальных заданий по проектированию информационных систем в Microsoft SQL Server 2012.	1	
15.	Лабораторная работа 15. Выполнение индивидуальных заданий по проектированию информационных систем в Visual Studio 2012. Создание ленточных и табличных форм для работы с базами данных.	1	
16.	Лабораторная работа 16. Выполнение индивидуальных заданий по проектированию информационных систем в Visual Studio 2012. Создание отчетов и диаграмм.	1	
17.	Лабораторная работа 17. Подготовка документации IT проекта.	2	
18.	Лабораторная работа 18. Расчет экономической эффективности проекта.	2	
	Итого за 8 семестр	**20**	
	Итого	**20**	

СТРУКТУРА И СОДЕРЖАНИЕ ЛАБОРАТОРНЫХ ЗАНЯТИЙ

ЛАБОРАТОРНЫЕ РАБОТЫ

Лабораторная работа № 1
«Проектирование ПО с применением UML.»

Цель работы:

- Ознакомиться с целями и задачами проектирования ПО
- Понять и усвоить методологию моделирования с применением UML
- Научиться создавать диаграммы Use Case
- Научиться создавать диаграмму вариантов использования.
- Научиться создавать диаграммы действий.

Порядок выполнения лабораторной работы.

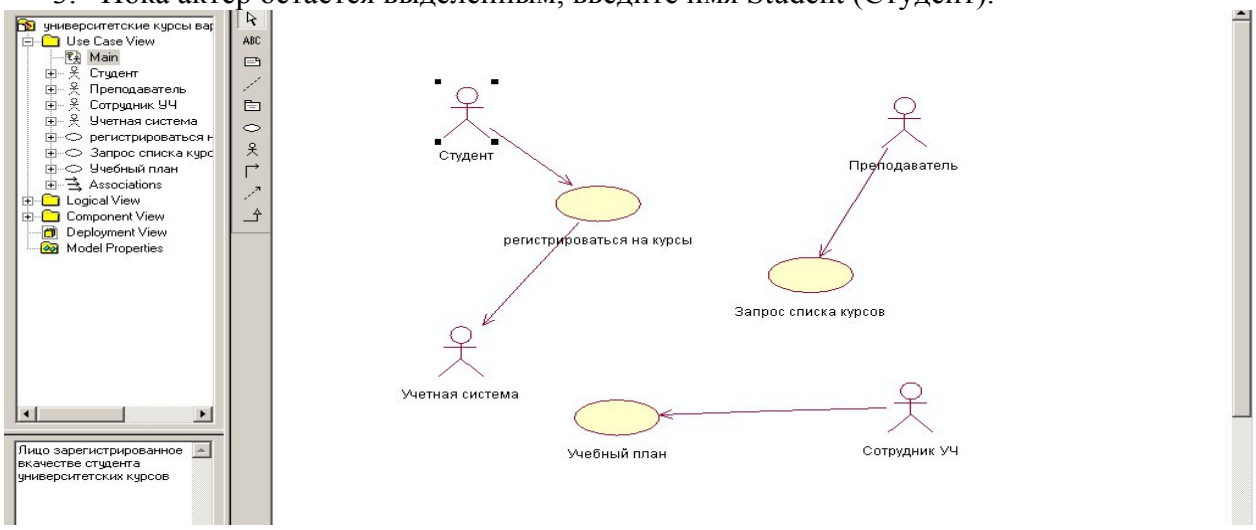
Создание диаграммы Use Case

Моделирование проблемы регистрации курсов начнем с создания диаграммы Use Case. Этот тип диаграммы представляется актерами, элементами Use Case и отношениями между ними. Откроем главную диаграмму Use Case.

1. В окне браузера щелкнем по значку + слева от пакета Use Case View.
2. Для открытия диаграммы выполним двойной щелчок по значку Main.

Первый шаг построения этой диаграммы состоит в определении актеров, фиксирующих роли внешних объектов, взаимодействующих с системой. В нашей проблемной области можно выделить *4* актера - Student (Студент), Professor (Преподаватель), Registrar (Сотрудник УЧ) и Billing System (Учетная система).

1. На панели инструментов щелкните по значку актера.
2. Для добавления актера в диаграмму щелкните в нужном месте диаграммы.
3. Пока актер остается выделенным, введите имя Student (Студент).



4. Повторите предыдущие шаги для ввода трех других актеров (Professor, Registrar и Billing System - Преподаватель, Сотрудник УЧ, Учетная система).

Далее для каждого актера нужно определить соответствующие элементы Use Case. Элемент Use Case представляет определенную часть функциональности, обеспечиваемой системой. Вы можете идентифицировать элементы Use Case путем рассмотрения каждого актера и его взаимодействия с системой. В нашей модели актер Student хочет регистрироваться на курсы (Register for Courses). Актер Billing System получает

информацию о регистрации. Актер Professor хочет запросить список курса (Request a Course Roster). Наконец, актер Registrar должен управлять учебным планом (Manage Curriculum).

1. На панели инструментов щелкните по значку элемента Use Case.
2. Для добавления элемента Use Case в диаграмму щелкните в нужном месте диаграммы.
3. Пока элемент Use Case остается выделенным, введите имя Register for Courses.
4. Повторите предыдущие шаги для ввода других элементов Use Case (Request Course Roster, Manage Curriculum).

Далее между актерами и элементами Use Case рисуются отношения. Чтобы показать направление взаимодействия (кто инициирует взаимодействие), используются однонаправленные стрелки (uni-directional arrows). В системе регистрации курсов актер Student инициирует элемент Use Case Register for Courses, который, в свою очередь, взаимодействует с актером Billing System. Актер Professor инициирует элемент Use Case Request Course Roster. Актер Registrar инициирует элемент Use Case Manage Curriculum

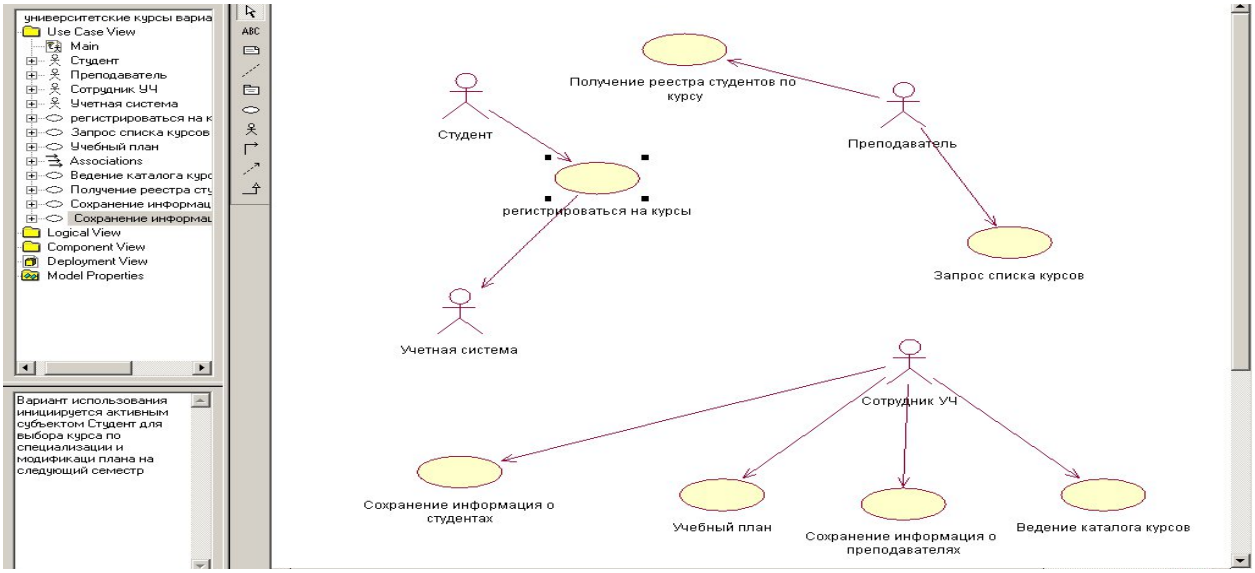
1. На панели инструментов щелкните по значку однонаправленной ассоциации (стрелке).
2. Щелкните по актеру Student и перетащите линию на элемент Use Case Register for Courses.
3. На панели инструментов щелкните по значку однонаправленной ассоциации (стрелке).
4. Щелкните по элементу Use Case Register for Courses и перетащите линию на актера Billing System.
5. Повторите предыдущие шаги для ввода других отношений (от актера Professor к элементу Use Case Request Course Roster и от актера Registrar к элементу Use Case Manage Curriculum).

В модель следует включить краткое описание каждого активного субъекта, которое должно определять роль в процессе использования системы.

Окно документирования можно открыть в меню View – Documentation.

Заполните документирование для наших актеров.

Варианты использования позволяют моделировать диалог между активным субъектом и системой и отображают функции последней, предоставляемые в распоряжение субъекта.



Написать варианты использования для остальных объектов.

Создадим потоки событий для варианта использования, которые документируются при помощи следующего шаблона, который создается в виде текстового файла:

1.0 Краткое описание
1.1 Краткое описание.
2.0 Потоки событий
2.1 Основной поток
2.2 Альтернативные потоки
2.2.1 Альтернативные поток
3.0 Специальные требования
3.1 Специальные требования
4.0 Предусловия
4.1 Предусловие
5.0 Постусловие
5.1 Постусловие
6.0 Дополнительные замечания
6.1 Дополнительные замечания

Теперь свяжем файл спецификации потока событий с вариантом использования, для этого расположим курсор мыши в окне Browser вызовем контекстное меню и выбрать элемент меню Open specification и перейти на вкладку Files диалогового окна Use Case specification, расположите курсор в пределах области окна и вновь активизируйте контекстное меню и выбрав элемент меню Insert file выберите необходимый файл после чего выберите Открыть и нажмите ОК.

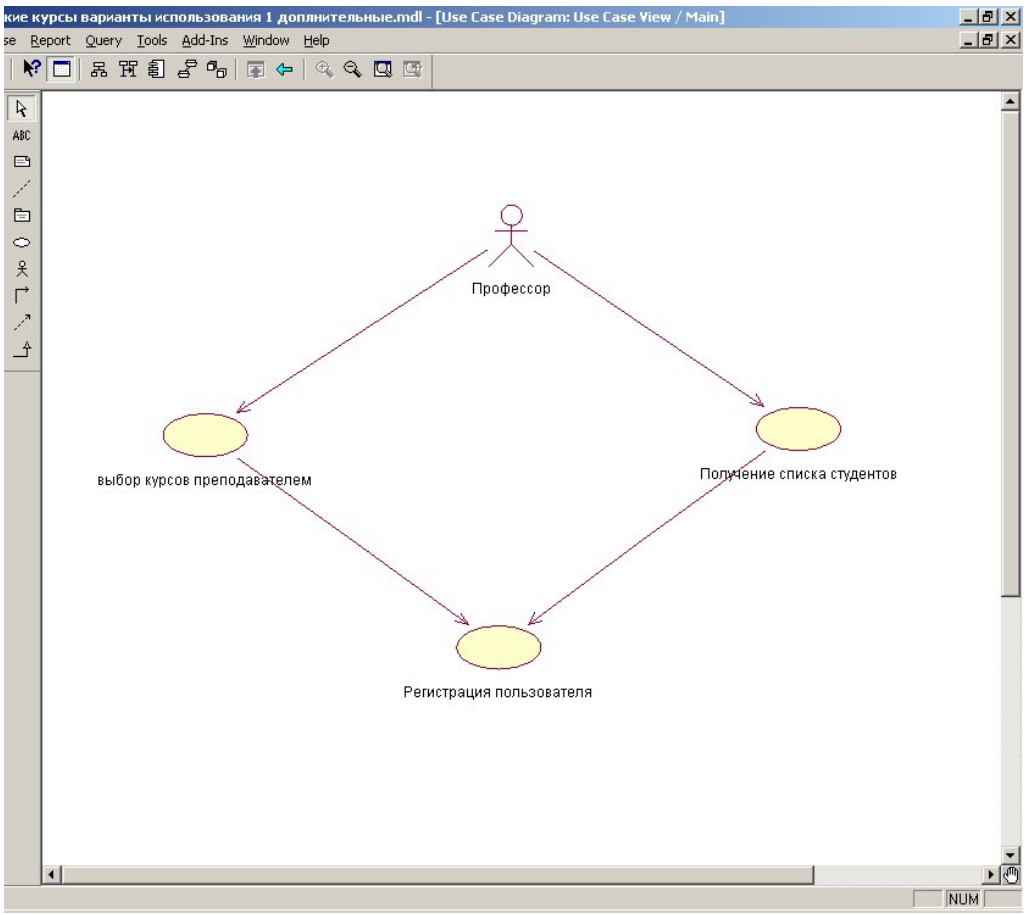
Создадим дополнительную диаграмму вариантов использования.

Расположим курсор мыши над элементом Use Case View окна Browser и щелкнуть правой кнопкой чтобы активировать контекстное меню.

Выбрать элемент New - Use Case Diagram и дерево пополнится элементом NewDiagram соответствующим новой диаграмме вариантов использования.

Изменить элемент NewDiagram

Двойным щелчком на элементе открыть окно дополнительной диаграммы вариантов использования и включить в нее требуемые активные субъекты, варианты использования и связи.



Самостоятельно.

Создать две дополнительные диаграммы для студента и Сотрудника УЧ.

Диаграммы действий воспроизводят поток функций управления, показывают, какие ветви процесса могут выполняться параллельно и определять альтернативные пути достижения целей. Диаграммы действий конструируется на начальных фазах жизненного цикла системы, представляют потоки которые охватывают несколько вариантов использования или протекают на уровне определенного варианта.

Расположить курсор мыши над элементом Use Case View и щелкнуть правой кнопкой чтобы активизировать контекстное меню.

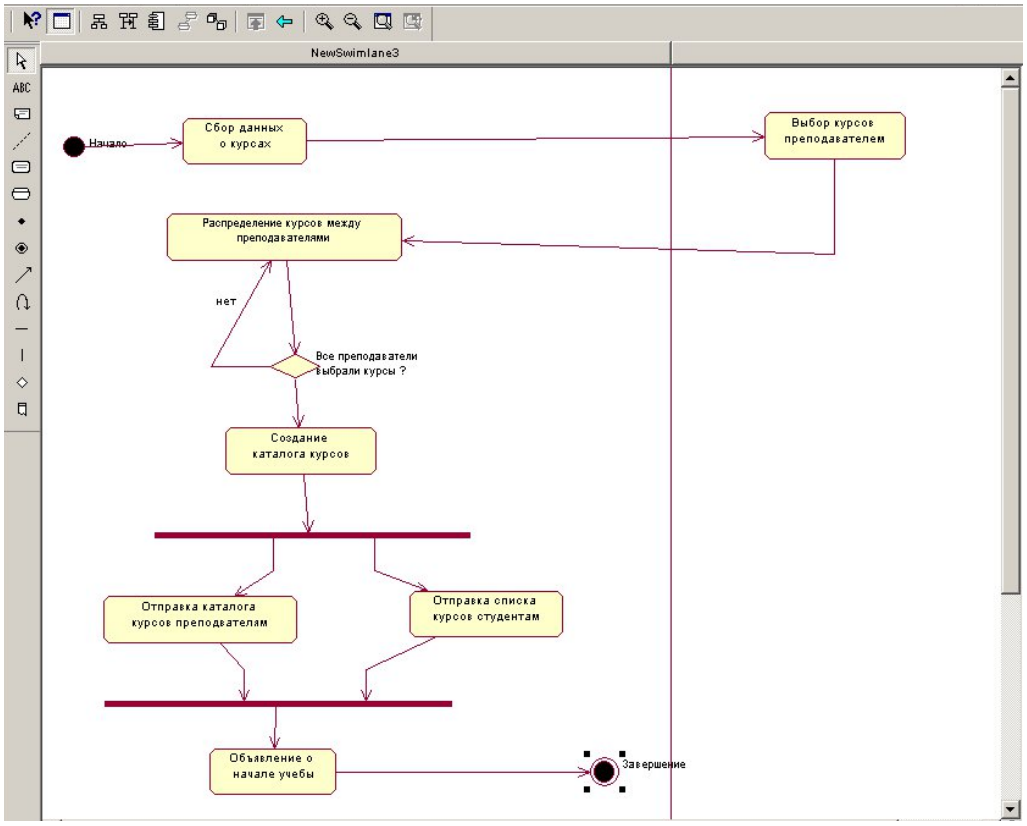
Выбрать элемент меню New – Activity Diagram, выбрать новый элемент и изменить его название и двойным щелчком открыть окно диаграммы действий.

Действия описывают некоторый фрагмент поведения системы в контексте потока функций управления.

Элемент перехода применяются в целью обозначения направления передачи управления от одного действия к другому.

Точки принятия решения - в таких точках поток обычно претерпевает ветвление в зависимости от принимаемых системой или пользователем решений.

Полосы синхронизации позволяют указать допускают единовременное выполнение или подлежат логическому выполнению.



Зоны – диаграмма может быть разделена на зоны каждая из которых связана с определенным активным субъектом.

Исходное и завершающее действия – обычно поток управления содержит одно исходное и несколько завершающих действий.

Самостоятельно написать диаграмму действий для учебной части.

Контрольные вопросы

При защите лабораторной работы студент должен ориентироваться в проделанной работе, знать:

- Цели и задачи проектирования ПО.
- Методологию функционального моделирования с применением UML.
- Как создавать диаграммы Use Case.
- Как создавать диаграмму вариантов использования.
- Как создавать диаграммы действий.

Студент должен уметь изменить модель (добавить новые или удалить элементы) по требованию преподавателя.

Лабораторная работа № 2

«Построение моделей в BPWin. Построение IDEF0 диаграмм.»

Цель работы:

- Ознакомиться с целями и задачами предпроектного обследования
- Понять и усвоить методологию функционального моделирования
- Научиться определять границы моделирования
- Научиться определять точку зрения при построении модели.
- Изучить принцип декомпозиции.

Порядок выполнения лабораторной работы.

Первый шаг при построении модели IDEF0 заключается в определении назначения модели – набора вопросов, на которые должна отвечать модель. **Границы моделирования** предназначены для обозначения ширины охвата предметной области и глубины детализации и являются логическим продолжением уже определенного назначения модели. Следующим шагом указывается предполагаемая целевая аудитория, для нужд которой создается модель. Зачастую от выбора целевой аудитории зависит уровень детализации, с которым должна создаваться модель. Перед построением модели необходимо иметь представление о том, какие сведения о предмете моделирования уже известны, какие дополнительные материалы и (или) техническая документация для понимания модели могут быть необходимы целевой аудитории, какие язык и стиль изложения являются наиболее подходящими.

Под **точкой зрения** понимается перспектива, с которой наблюдалась система при построении модели. Точка зрения выбирается таким образом, чтобы учесть уже обозначенные границы моделирования и назначение модели. Однажды выбранная точка зрения остается неизменной для всех элементов модели. При необходимости могут быть созданы другие модели, отображающие систему с других точек зрения. Вот несколько примеров точек зрения при построении моделей: клиент, поставщик, владелец, редактор.

Действие, обычно в IDEF0 называемое функцией, обрабатывает или переводит входные параметры (сырье, информацию и т.п.) в выходные. Поскольку модели IDEF0 представляют систему как множество иерархических функций, в первую очередь должна быть определена функция, описывающая систему в целом – **контекстная функция**. Функции изображаются на диаграммах как поименованные прямоугольники, или функциональные блоки. Имена функций в IDEF0 подбираются по сходным правилам с именами действий – с использованием глаголов или отглагольных существительных. Важно подбирать имена таким образом, чтобы они отражали систему так, как если бы она обозревалась с точки зрения, выбранной для моделирования. Пример функционального блока приведен на рисунке 1.

Рисунок 1.1 - Функциональный блок IDEF0

Каждый блок, не имеющей декомпозиции, помечается небольшой диагональной чертой, расположенной в левом верхнем углу блока.

Любой блок может быть декомпозирован на составляющие его блоки. Функция декомпозиции позволяет разбить сложные процессы на составляющие его операции. При этом уровень детализации процесса определяется непосредственно разработчиком модели (рисунок 2).

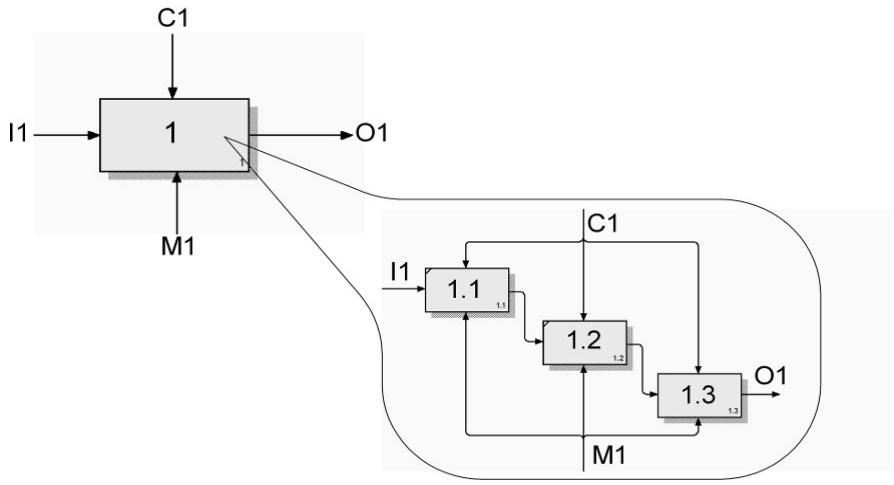


Рисунок 1.2 – Принцип декомпозиции

Декомпозиция позволяет постепенно и структурировано представлять модель системы в виде иерархической структуры отдельных диаграмм, что делает ее менее перегруженной и легко читаемой.

1.2.3 Границы и связи

Чтобы быть полезным, описание любого блока должно, как минимум, включать в себя описание объектов, которые блок создает в результате своей работы («выхода»), и объектов, которые блок потребляет или преобразует («вход»).

В IDEF0 также моделируются управление и механизмы исполнения. Под управлением понимаются объекты, воздействующие на способ, которым блок преобразует вход в выход. Механизм исполнения – объекты, которые непосредственно выполняют преобразование входа в выход, но не потребляются при этом сами по себе.

Для отображения категорий информации, присутствующих на диаграммах IDEF0, существует аббревиатура ICOM, отображающая четыре возможных типа стрелок:

I (Input) – вход – нечто, что потребляется в ходе выполнения процесса;

C (Control) – управление – ограничения и инструкции, влияющие на ход выполнения процесса;

O (Output) – выход – нечто, являющееся результатом выполнения процесса;

M (Mechanism) – исполняющий механизм – нечто, что используется для выполнения процесса, но не потребляется само по себе (рисунок 3).

Рисунок 3 показывает четыре возможных типа стрелок в IDEF0, каждый из типов соединяется со своей стороной функционального блока.

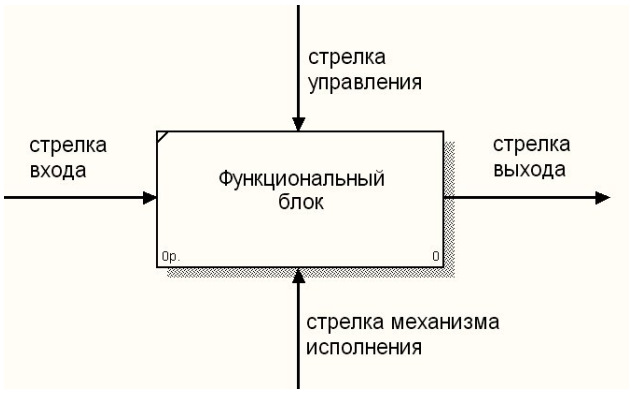


Рисунок 1.3 – Типы стрелок функционального блока

Для названия стрелок, как правило, употребляются имена существительные. Стрелки могут представлять собой людей, места, вещи, идеи или события. Как и в случае с функциональными блоками, присвоение имен всем стрелкам на диаграмме является только необходимым условием для понимания читателем сути изображенного. Отдельное описание каждой стрелки в текстовом виде может оказаться критическим фактором для построения точной и полезной модели.

Стрелки входа. Вход представляет собой сырье, или информацию, потребляемую или преобразуемую функциональным блоком для производства выхода. Стрелки входа всегда направлены в левую сторону прямоугольника, обозначающего в IDEF0 функциональный блок. Наличие входных стрелок на диаграмме не является обязательным, так как возможно, что некоторые блоки ничего не преобразуют и не изменяют. Примером блока, не имеющего входа, может служить «принятие решения руководством», где для принятия решения анализируется несколько факторов, но ни один из них непосредственно не преобразуется и не потребляется в результате принятия какого-либо решения.

Стрелки управления. Стрелки управления отвечают за регулирование того, как и когда выполняется функциональный блок, и, если он выполняется, какой выход получается

в результате его выполнения. Так как управление контролирует поведение функционального блока для обеспечения создания желаемого выхода, каждый функциональный блок должен иметь, как минимум, одну стрелку управления. Стрелки управления всегда входят в функциональный блок сверху.

Управление часто существует в виде правил, инструкций, законов, политики, набора необходимых процедур или стандартов. Влияя на работу блока, оно непосредственно не потребляется и не трансформируется в результате. Может оказаться, что целью функционального блока является как раз изменение того или иного правила, инструкции, стандарта и т.п. В этом случае стрелка, содержащая соответствующую информацию, должна рассматриваться не как управление, а как вход функционального блока.

Управление можно рассматривать как специфический вид входа. В случаях, когда неясно, относить ли стрелку к входу или к управлению, предпочтительно относить ее к управлению до момента, пока неясность не будет разрешена.

Стрелки выхода. Выход – это продукция или информация, получаемая в результате работы функционального блока. Каждый блок должен иметь, как минимум, один выход. Действие, которое не производит никакого четко определяемого выхода, не должно моделироваться вообще (по меньшей мере, должно рассматриваться в качестве одного из первых кандидатов на исключение из модели).

При моделировании непроизводственных предметных областей выходами, как правило, являются данные, в каком-либо виде обрабатываемые функциональным блоком. В этом случае важно, чтобы названия стрелок входа и выхода были достаточно различимы по своему смыслу. Например, блок «Прием пациентов» может иметь стрелку «Данные о пациенте» как на входе, так и на выходе. В такой ситуации входящую стрелку можно назвать «Предварительные данные о пациенте», а исходящую – «Подтвержденные данные о пациенте».

Стрелки механизма исполнения. Механизмы являются ресурсом, который непосредственно исполняет моделируемое действие. С помощью механизмов исполнения могут моделироваться: ключевой персонал, техника и (или) оборудование. Стрелки механизма исполнения могут отсутствовать в случае, если оказывается, что они не являются необходимыми для достижения поставленной цели моделирования.

Комбинированные стрелки. В IDEF0 существует пять основных видов комбинированных стрелок: выход – вход, выход – управление, выход – механизм исполнения, выход – обратная связь на управление и выход – обратная связь на вход.

Стрелка *выход – вход* применяется, когда один из блоков должен полностью завершить работу перед началом работы другого блока (рисунок 4).

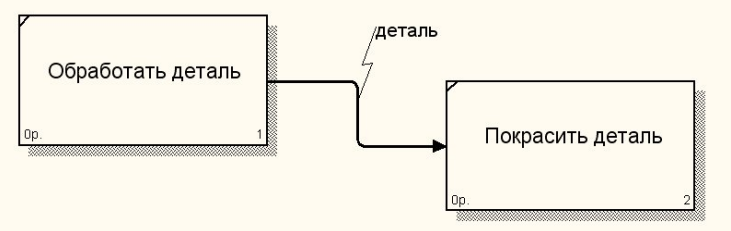


Рисунок 1.4 – Комбинированная стрелка *выход – вход*

Стрелка *выход - управление* отражает ситуацию преобладания одного блока над другим, когда один блок управляет работой другого. На рисунке 5 принципы формирования инвестиционного портфеля управляют поведением брокеров на бирже.

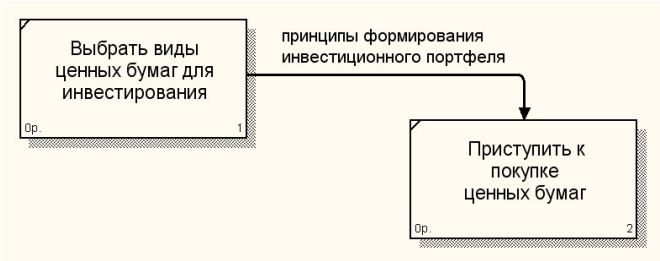


Рисунок 1.5 – Комбинированная стрелка *выход – управление*

Стрелки *выход* механизм *исполнения* встречаются реже и отражают ситуацию, когда выход одного функционального блока применяется в качестве оборудования для работы другого блока. На рисунке 6 зажим - устройство, используемое для закрепления детали во время ее сборки, должен быть собран для того, чтобы выполнить сборку детали.

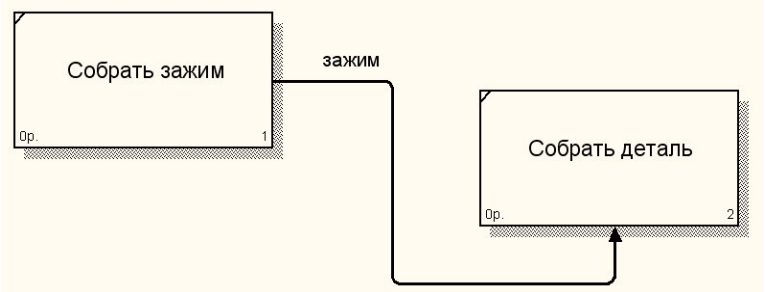


Рисунок 1.6 – Комбинированная стрелка *выход* механизм *исполнения*

Обратные связи на вход и на управление применяются в случаях, когда зависимые блоки формируют обратные связи для управляющих ими блоков. На рисунке 7 получаемая от брокеров информация о текущих биржевых курсах применяется для корректировки стратегии игры на бирже.

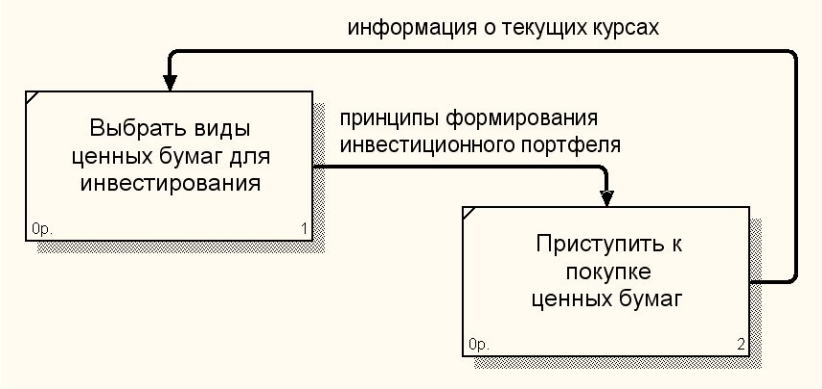


Рисунок 1.7 – Комбинированная стрелка *выход – обратная связь на управление*

Стрелка *выход - обратная связь на вход* обычно применяется для описания циклов повторной обработки чего-либо. Рисунок 8 может служить примером применения стрелки такого типа. Кроме того, связи *выход – обратная связь на вход* могут применяться в случае, если бракованная продукция может заново использоваться в качестве сырья, как это происходит, например, при производстве оконного стекла, когда разбитое в процессе производства стекло перемалывается и переплавляется заново вместе с обыкновенным сырьем.

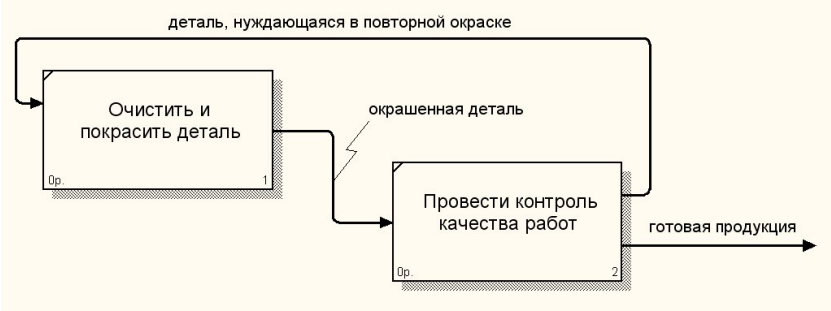


Рисунок 1.8 – Комбинированная стрелка *выход – обратная связь на вход*

Ветвление и слияние стрелок. Выход функционального блока может использоваться в нескольких других блоках. Фактически чуть ли не главная ценность IDEF0 заключается в том, что эта методология помогает выявить взаимозависимости между блоками системы. Соответственно IDEF0 предусматривает как ветвление, так и слияние стрелок на диаграмме. Разбитые на несколько частей стрелки могут иметь наименования, отличающиеся от наименования исходной стрелки. Исходная и разбитые (или объединенные) стрелки в совокупности называются связанными. Такая техника обычно применяется для того, чтобы отразить использование в процессе только части сырья или информации, обозначаемых исходной стрелкой.

Задание на лабораторную работу

1 Запустите **BPwin**. (Кнопка Start /BPwin).

2 Если появляется диалог **ModelMart Connection Manager**, нажмите на кнопку **Cancel** (Отмена).

3 Щелкните по кнопке . Появляется диалоговое окно **I would like to** (рисунок 1.9). Внесите в текстовое поле **Name** имя модели "Деятельность компании" и выберите **Type – Business Process (IDEF0)**. Нажмите кнопку **OK**.

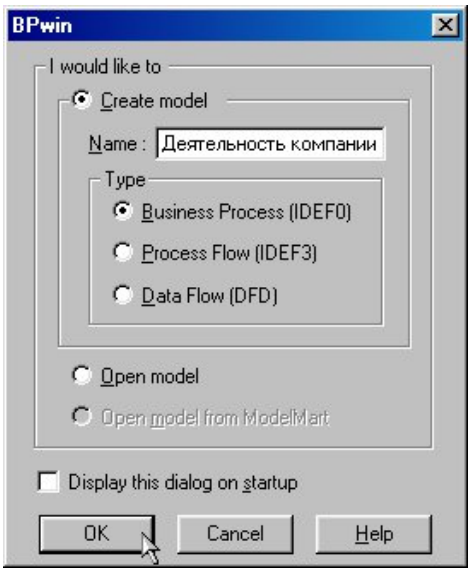


Рисунок 1.9

4 Откроется диалоговое окно **Properties for New Models (**Свойства новой модели**)** (рисунок 1.10).

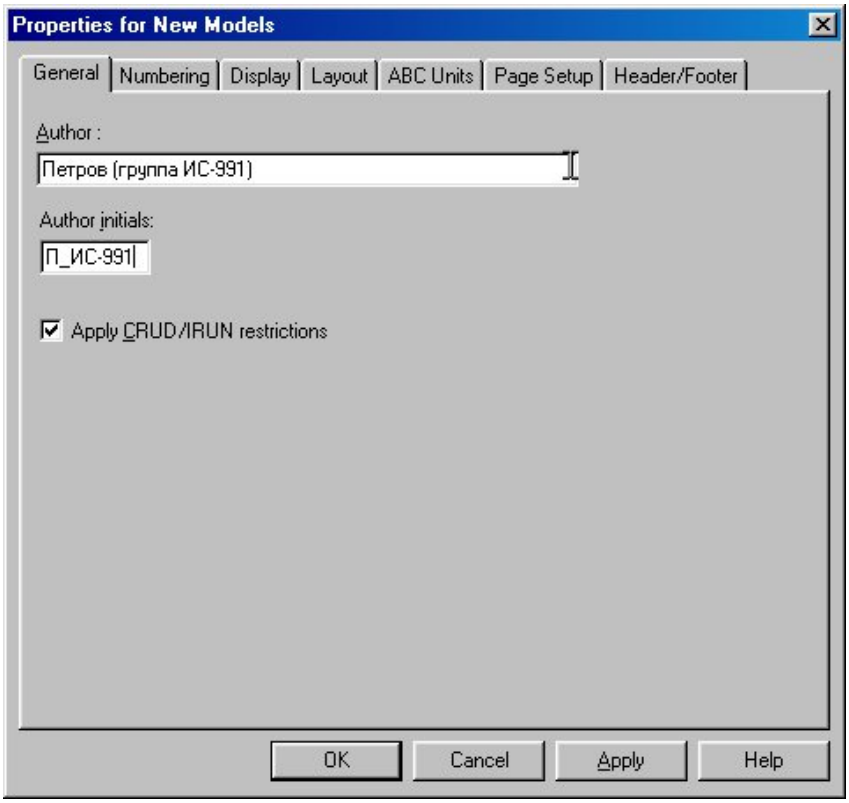


Рисунок 1.10

Введите в текстовое поле **Author** (Автор) имя автора модели и в текстовое поле **Author initials** его инициалы. Нажмите последовательно кнопки **Apply** и **OK**.

5 Автоматически создается незаполненная контекстная диаграмма (рисунок 1.11).

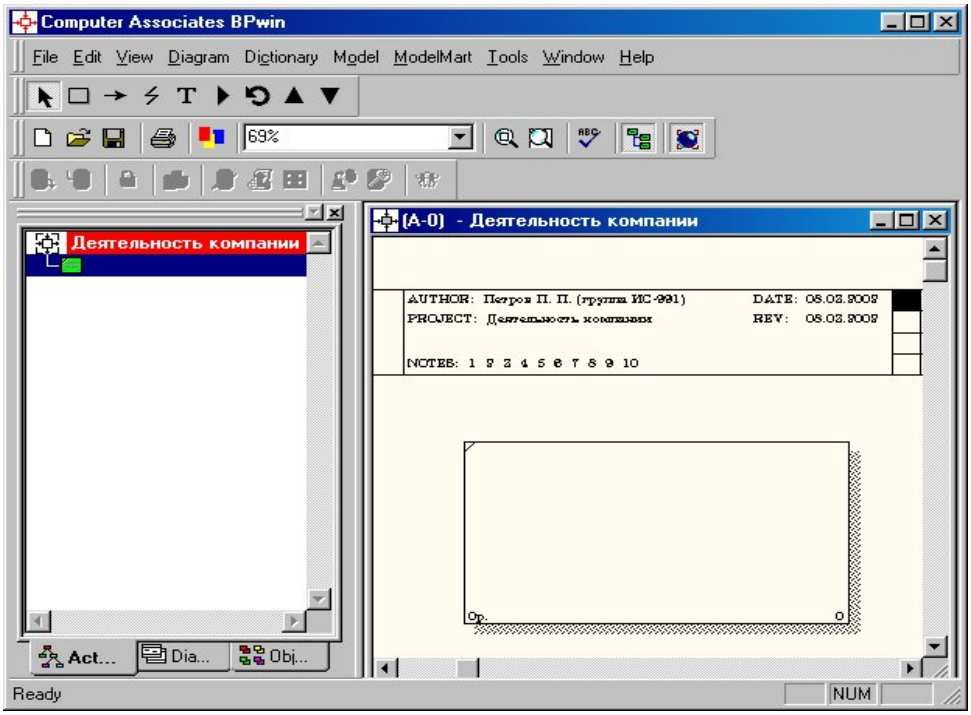


Рисунок 1.11.

6 Обратите внимание на кнопку  на панели инструментов. Эта кнопка включает и выключает инструмент просмотра и навигации – **Model Explorer** (Браузер модели). **Model Explorer** имеет три вкладки –**Activities** (

), **Diagrams** (

) и **Objects (** 

). Во вкладке **Activities** щелчок правой кнопкой по объекту в браузере модели позволяет выбрать опции редактирования его свойств (рисунок 1.12).

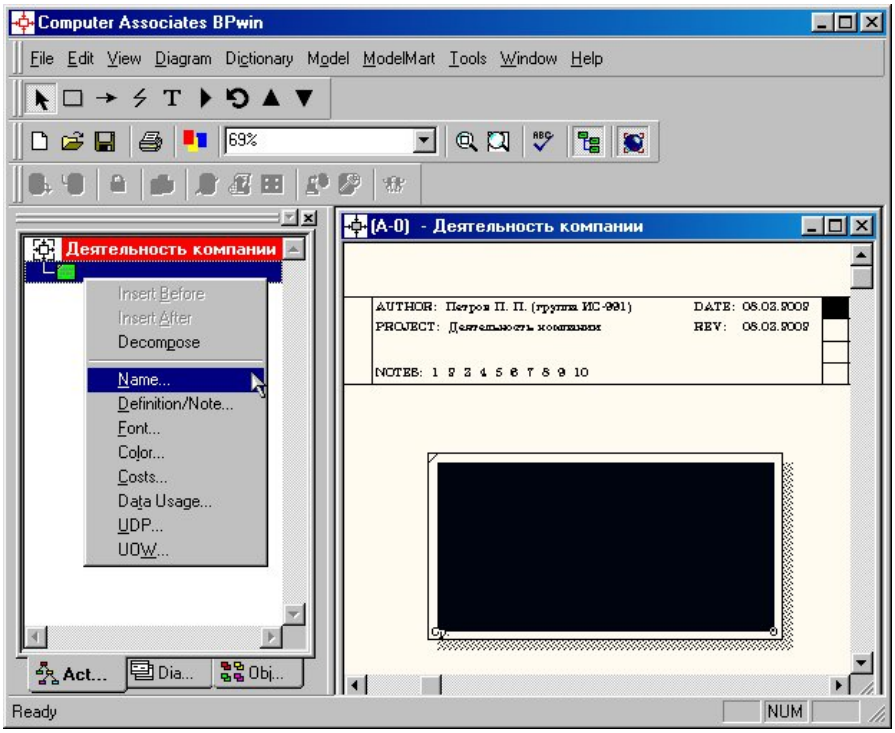


Рисунок 1.12.

7 Если вам непонятно, как выполнить то или иное действие, вы можете вызвать контекстную помощь – клавиша **F1** или воспользоваться меню **Help**.

8 Перейдите в меню **Model/Model Properties**. Во вкладке **General** диалогового окна **Model Properties** в текстовое поле **Model name** следует внести имя модели "Деятельность компании", а в текстовое поле **Project** имя проекта "Модель деятельности компании", и, наконец, в текстовое **Time Frame** (Временной охват) – **AS-IS (**Как есть**)** (рисунок 1.13).

9 Во вкладке **Purpose** диалогового окна **Model Properties** в текстовое поле **Purpose** (цель) внесите данные о цели разработки модели – " Моделировать текущие (AS-IS) бизнес-процессы компании", а в текстовое поле **Viewpoint** (точка зрения) – "Директор".

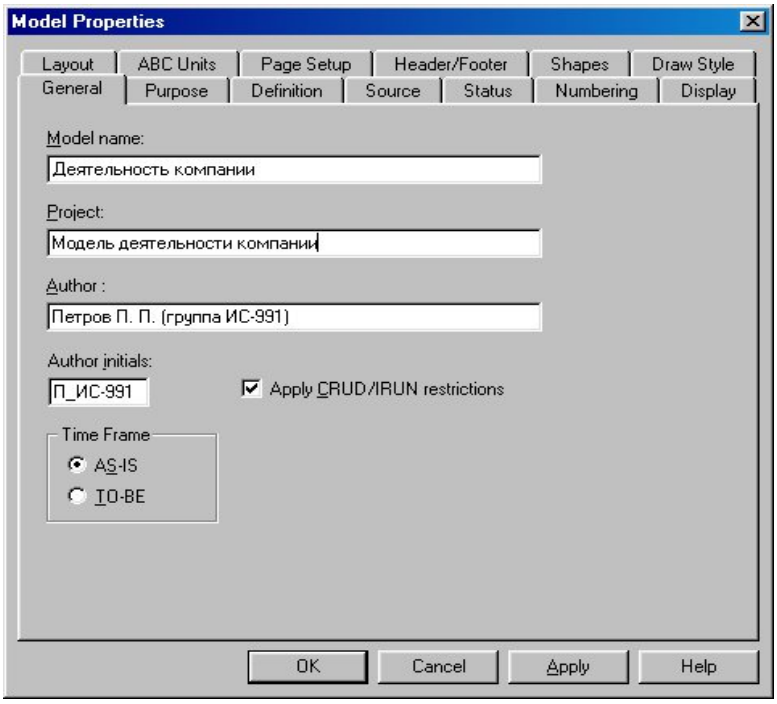


Рисунок 1.13.

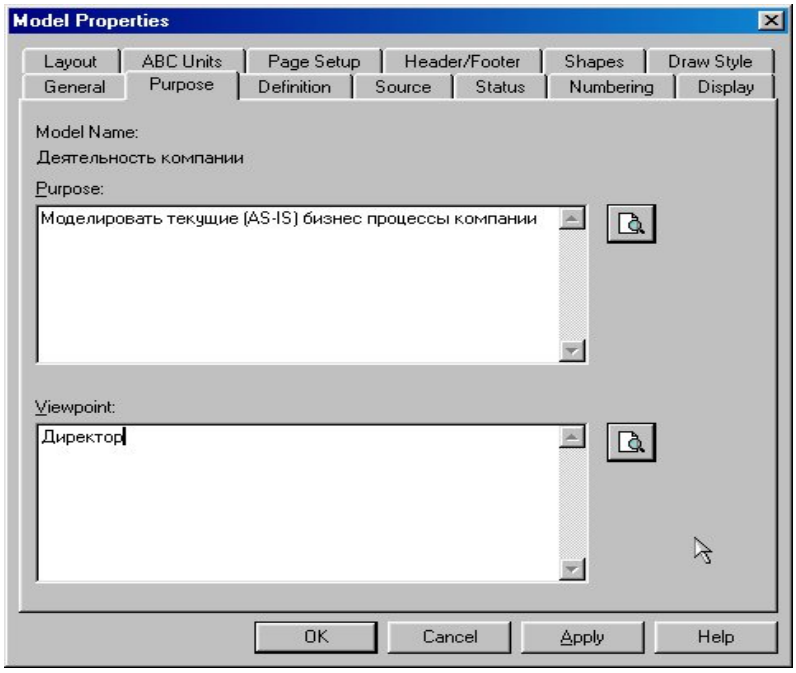


Рисунок 1.14.

10 Во вкладке **Definition** диалогового окна **Model Properties** в текстовое поле **Definition** (Определение) внесите "Это учебная модель, описывающая деятельность компании" и в текстовое поле **Scope** (охват) – "Общее управление бизнесом компании: исследование рынка, закупка компонентов, сборка, тестирование и продажа продуктов".

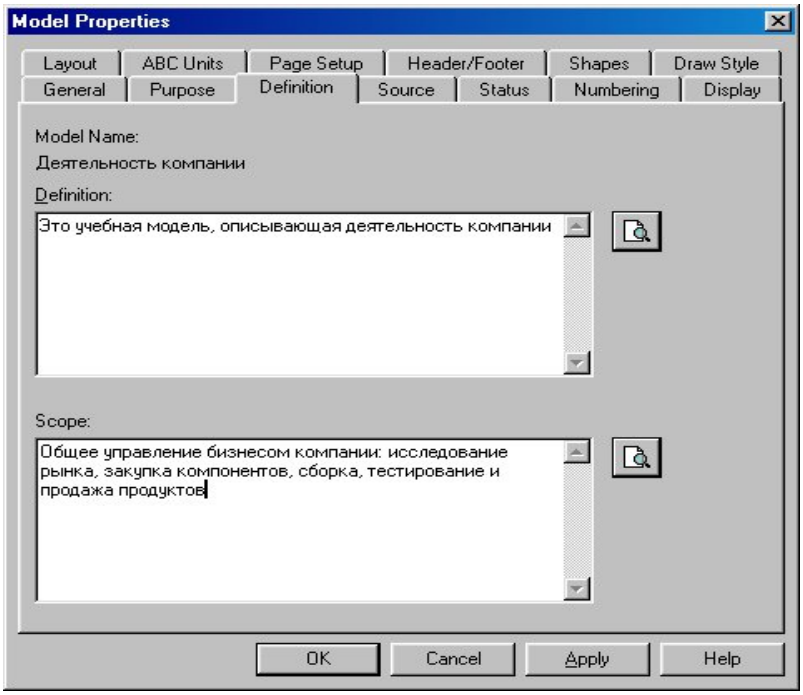


Рисунок 1.15

11 Перейдите на контекстную диаграмму и правой кнопкой мыши щелкните по прямоугольнику представляющему, в нотации **IDEF0**, условное графическое обозначение работы. В контекстном меню выберите опцию **Nam**e (рисунок 1.16). Во вкладке **Name** внесите имя "Деятельность компании" (рисунок 1.17).

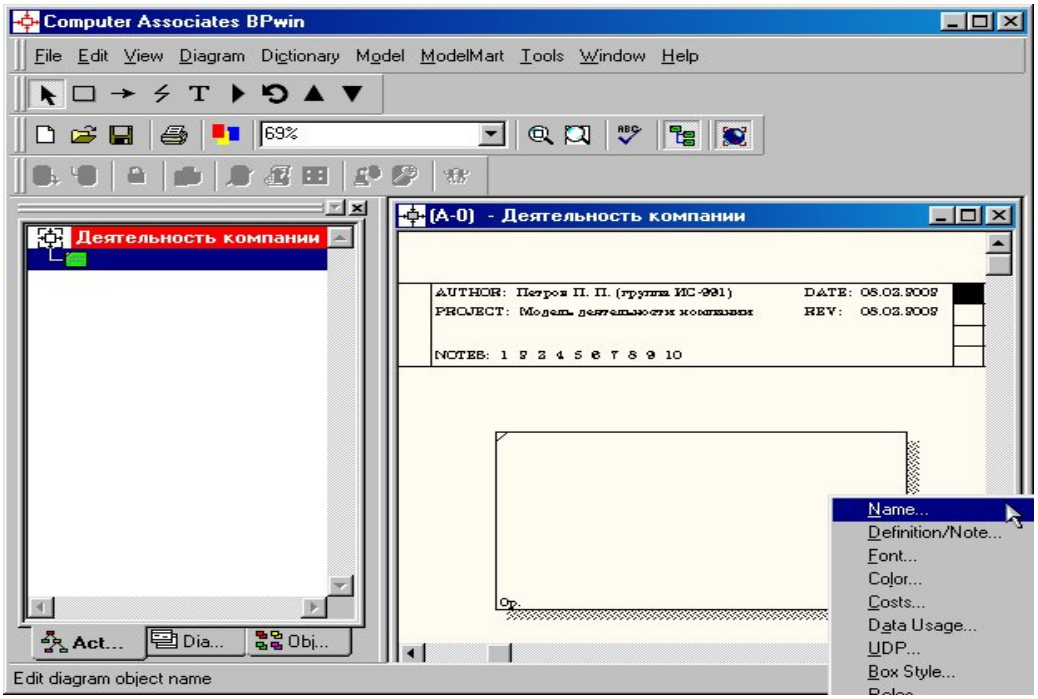


Рисунок 1.16.

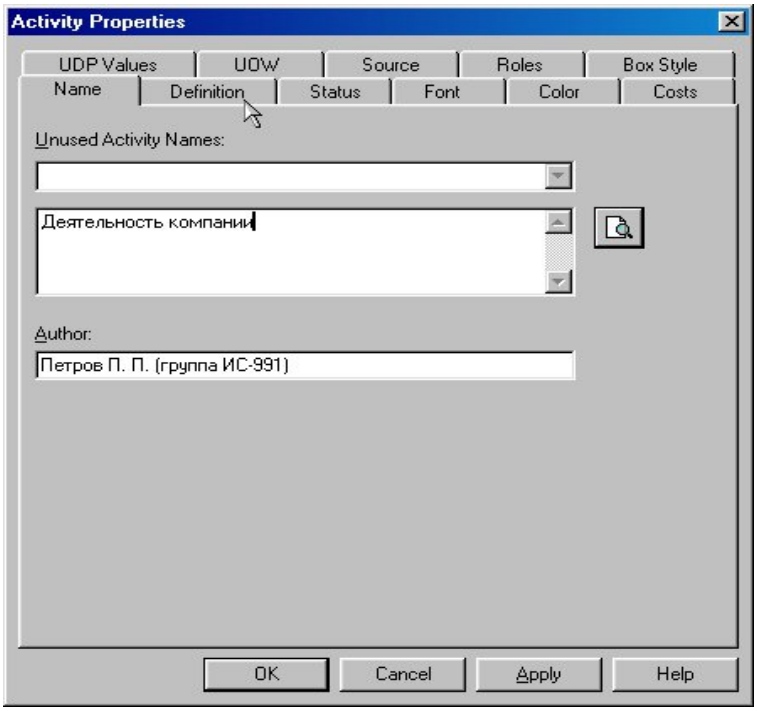


Рисунок 1.17.

Во вкладке **Definition** диалогового окна **Activity Properties** в текстовое поле **Definition** (Определение) внесите "Текущие бизнес-процессы компании" (рисунок 1.18). Текстовое поле **Note (**Примечания**)** оставьте незаполненным.

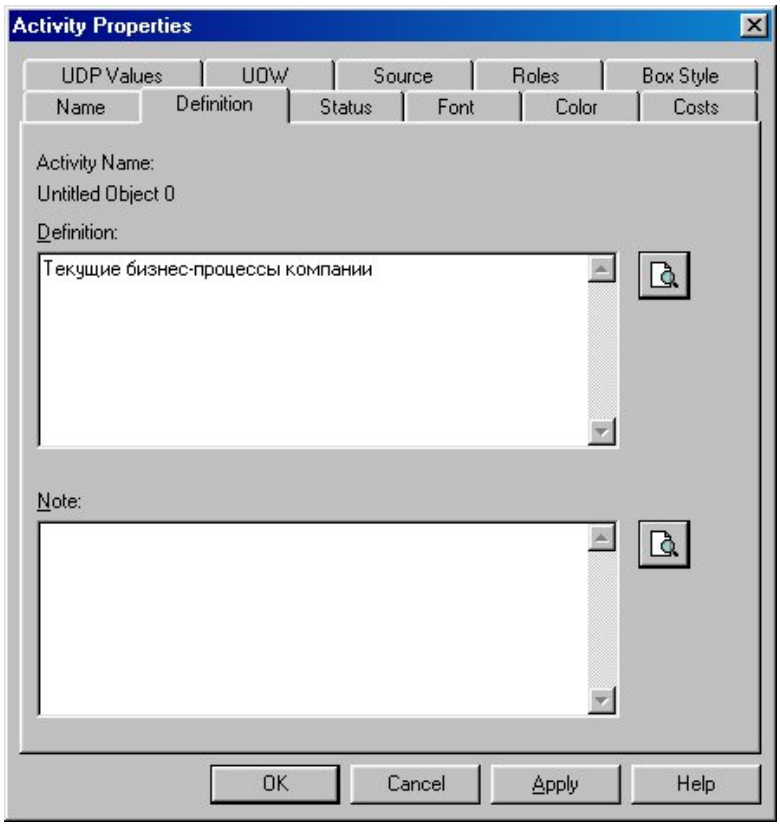


Рисунок 1.18

12 Создайте **ICOM**–стрелки на контекстной диаграмме (таблица 1).

Таблица 1. Стрелки контекстной диаграммы

Название стрелки (Arrow Name)	**Определение стрелки (Arrow Definition)**	**Тип стрелки (Arrow Type)**
Звонки клиентов	Запросы информации, заказы, техподдержка и т. д.	Input
Правила и процедуры	Правила продаж, инструкции по сборке, процедуры тестирования, критерии производительности и т. д.	Control
Проданные продукты	Настольные и портативные компьютеры	Output
Бухгалтерская система	Оформление счетов, оплата счетов, работа с заказами	Mechanism

13. С помощью кнопки  внесите текст в поле диаграммы - точку зрения и цель (рисунок 1.19). Результат показан на рисунке 1.20.

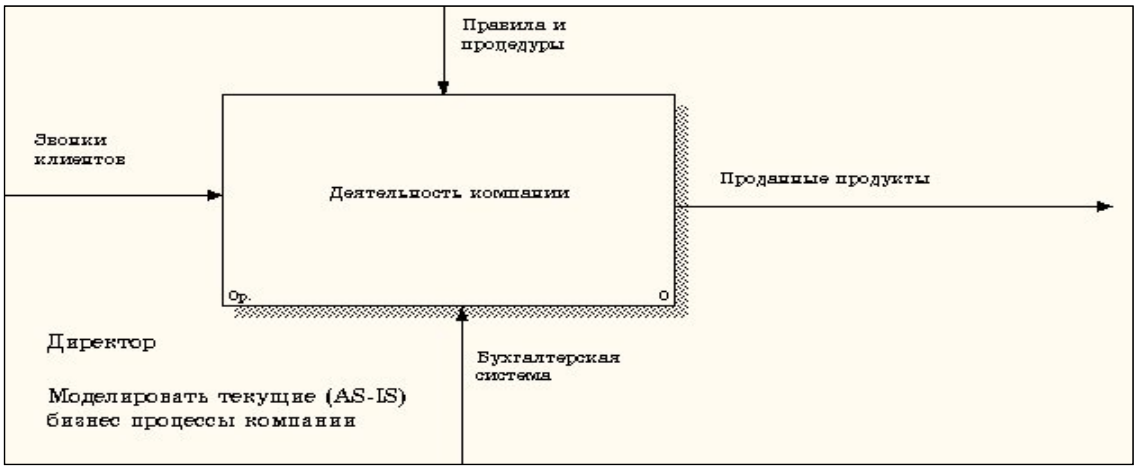


Рисунок 1.19.

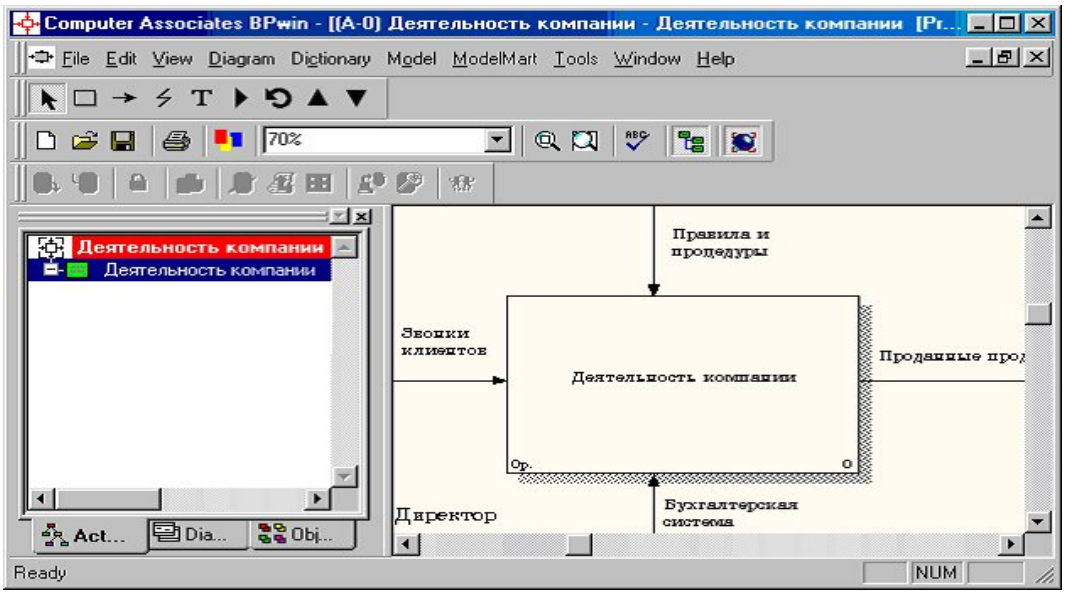


Рисунок 1.20.

Создайте отчет по модели. В меню **Tools/Reports/Model Report** (рисунок 1.21) задайте опции генерирования отчета (установите галочки) и нажмите кнопку **Preview** (Предварительный просмотр) (рисунок 1.22).

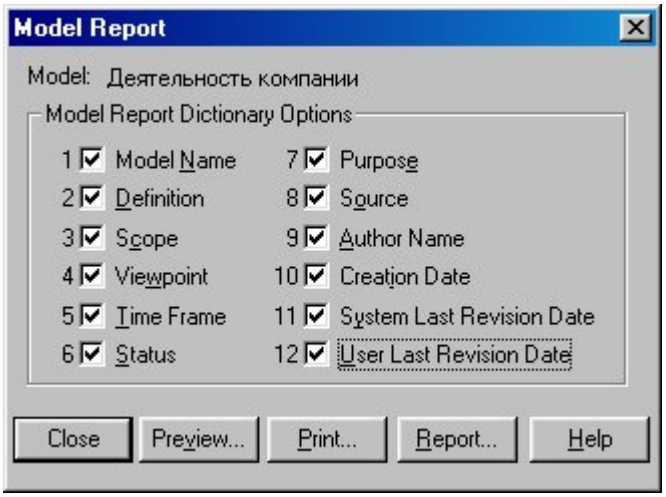


Рисунок 1.21.

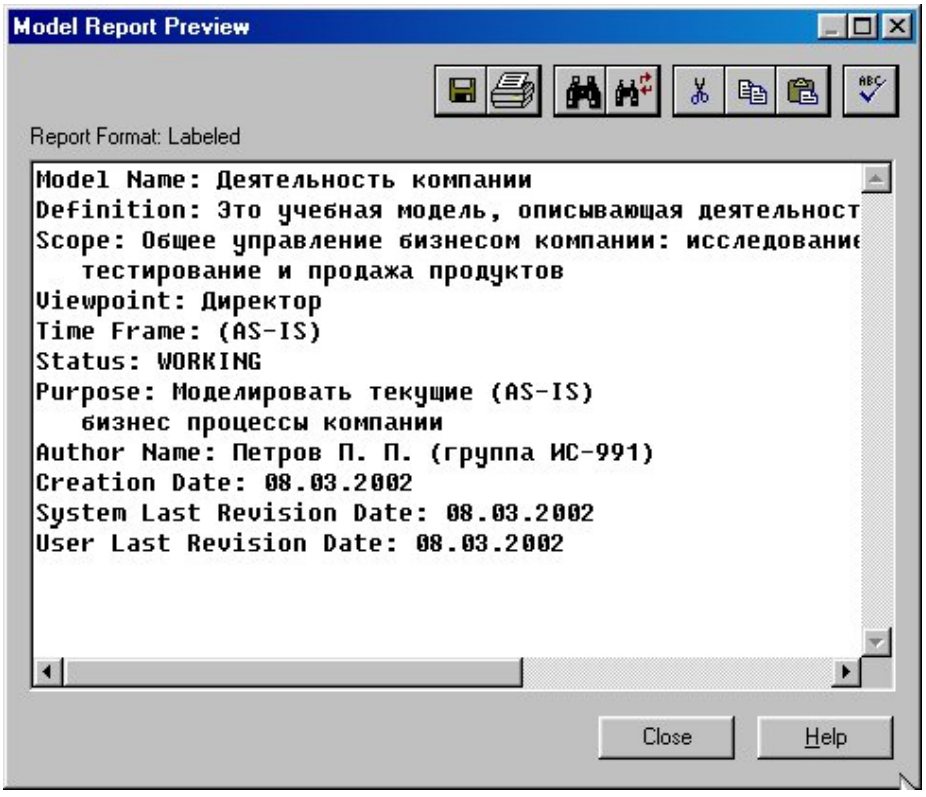


Рисунок 1.22.

Создание диаграммы декомпозиции

1 Выберите кнопку  перехода на нижний уровень в палитре инструментов и в диалоговом окне **Activity Box Count** (рисунок 1.23) установите число работ на диаграмме нижнего уровня – 3 и нажмите кнопку **OK**.

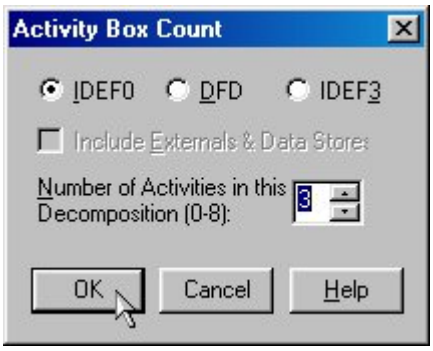


Рисунок 1.23.

2 Автоматически будет создана диаграмма декомпозиции (рисунок 1.24).

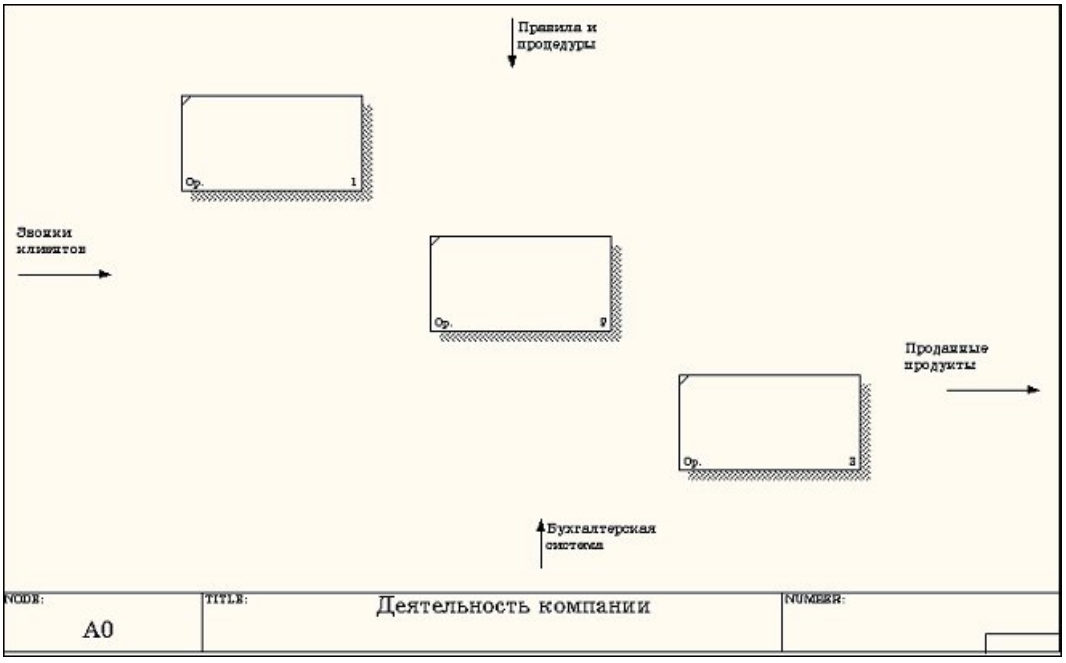


Рисунок 1.24.

Правой кнопкой мыши щелкните по работе расположенной в левом верхнем углу области редактирования модели, выберите в контекстном меню опцию **Name** и внесите имя работы. Повторите операцию для оставшихся двух работ. Затем внесите определение для каждой работы согласно данным таблицы 2.

Таблица 2

Название работы (Activity Name)	**Определение работы** (Activity Definition)
Продажи и маркетинг	Телемаркетинг и презентации, выставки

Сборка и тестирование компьютеров	Сборка и тестирование настольных и портативных компьютеров
Отгрузка и получение	Отгрузка заказов клиентам и получение компонентов от поставщиков

Диаграмма декомпозиции примет вид представленный на рисунке 1.25.

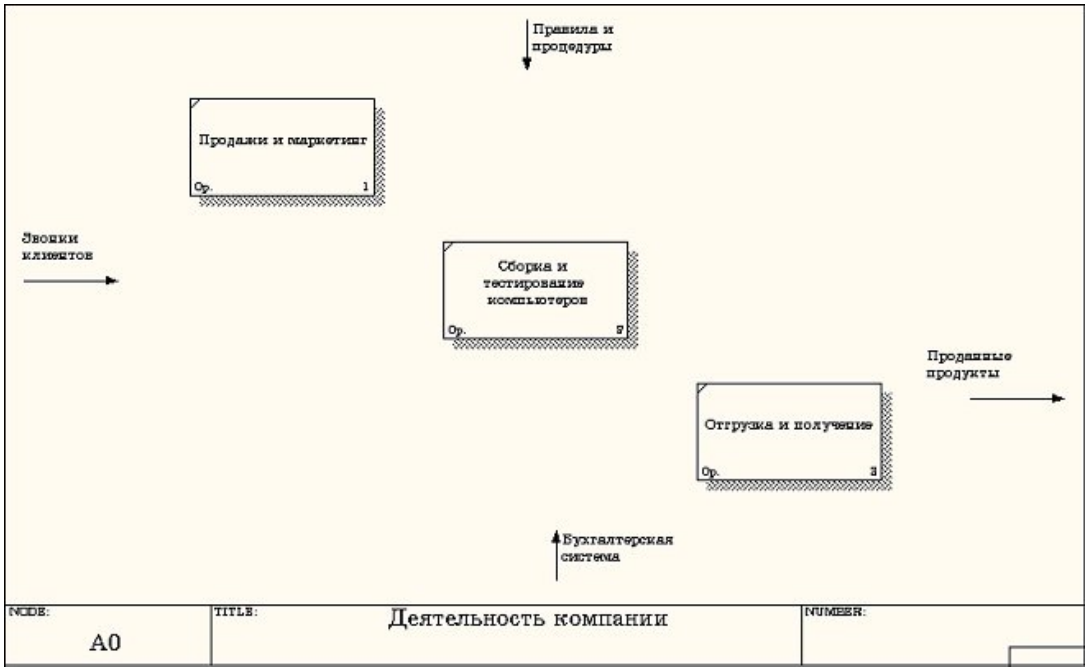


Рисунок 1.25.

3 Для изменения свойств работ после их внесения в диаграмму можно воспользоваться словарем работ (рисунок 1.26). Вызов словаря производится при помощи пункта главного меню **Dictionary /Activity**.

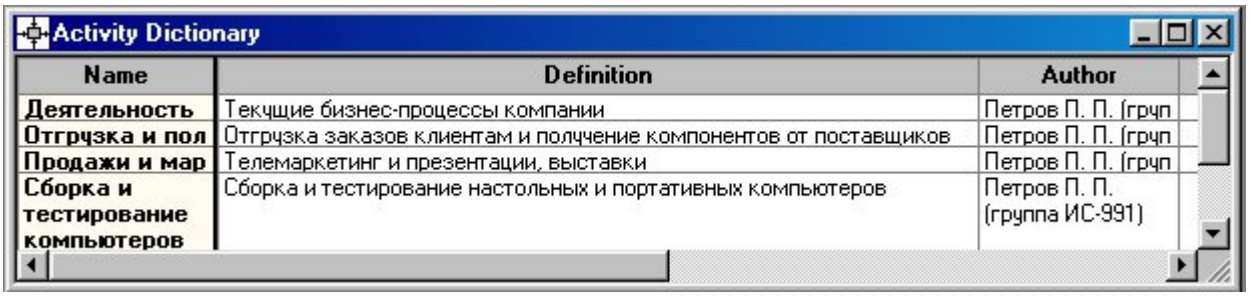


Рисунок 1.26.

Если описать имя и свойства работы в словаре, ее можно будет внести в диаграмму позже с помощью кнопки  в палитре инструментов. Невозможно удалить работу из словаря, если она используется на какой-либо диаграмме. Если работа удаляется из диаграммы, из словаря она не удаляется. Имя и описание такой работы может быть использовано в дальнейшем. Для добавления работы в словарь необходимо перейти в конец списка и щелкнуть правой кнопкой по

последней строке. Возникает новая строка, в которой нужно внести имя и свойства работы. Для удаления всех имен работ, не использующихся в модели, щелкните по кнопке  (**Purge (Чистить)**).

4 Перейдите в режим рисования стрелок и свяжите граничные стрелки, воспользовавшись кнопкой  на палитре инструментов так, как это показано на рисунке 1.27.

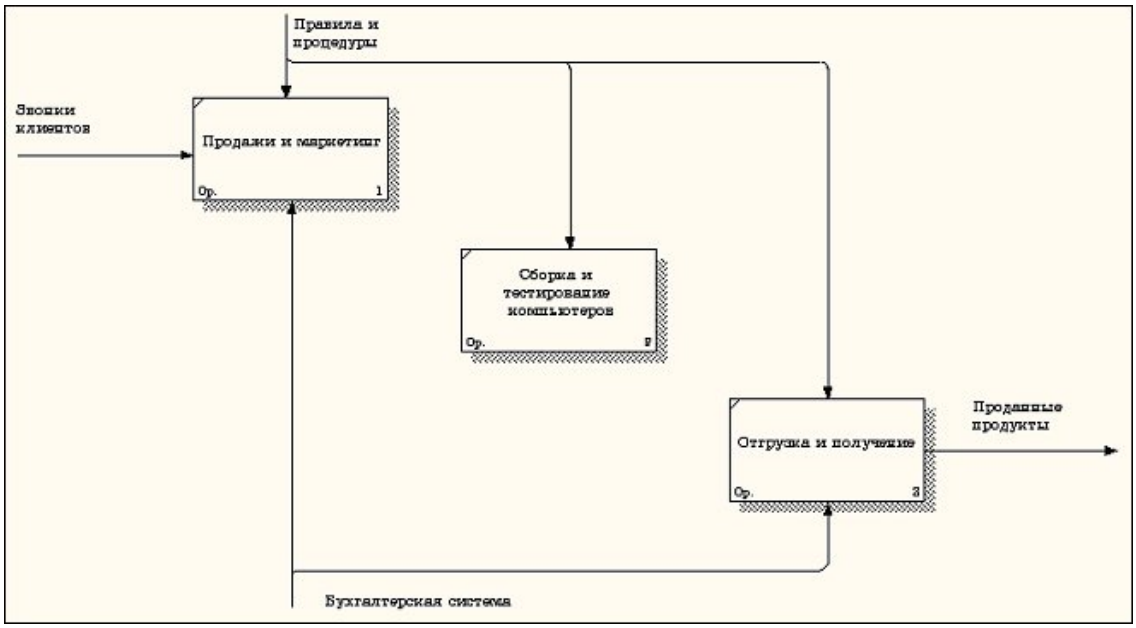


Рисунок 1.27.

5 Правой кнопкой мыши щелкните по ветви стрелки управления работы **"Сборка и тестирование компьютеров"** и переименуйте ее в **"Правила сборки и тестирования"** (рисунок 1.28).

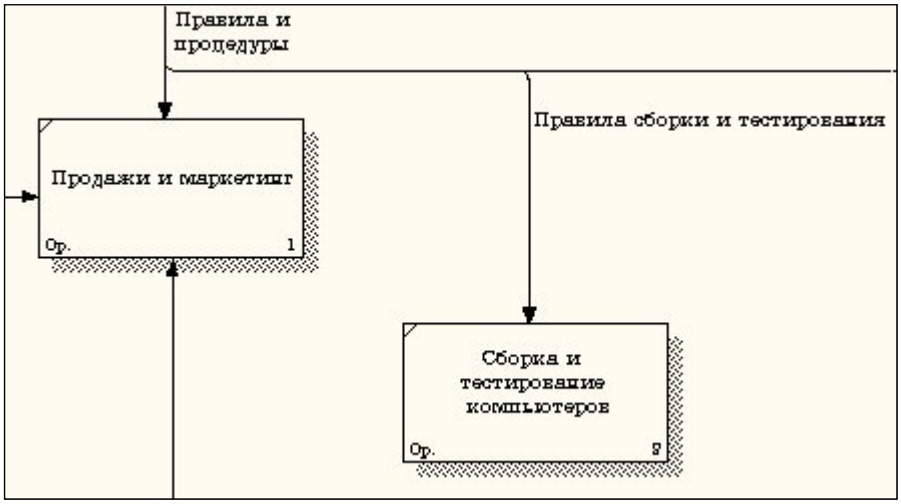


Рисунок 1.28.

Внесите определение для новой ветви: "**Инструкции по сборке, процедуры тестирования, критерии производительности и т. д.**" Правой кнопкой мыши щелкните по ветви стрелки механизма работы **"Продажи и маркетинг"** и переименуйте ее как **"Система оформления заказов" (**рисунок 1.29**).**

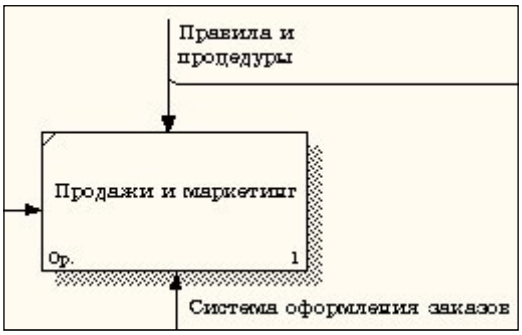


Рисунок 1.29

Альтернативный метод внесения имен и свойств стрелок – использование словаря стрелок (вызов словаря - меню **Dictionary/ Arrow**). Если внести имя и свойства стрелки в словарь (рисунок 1.30), ее можно будет внести в диаграмму позже.

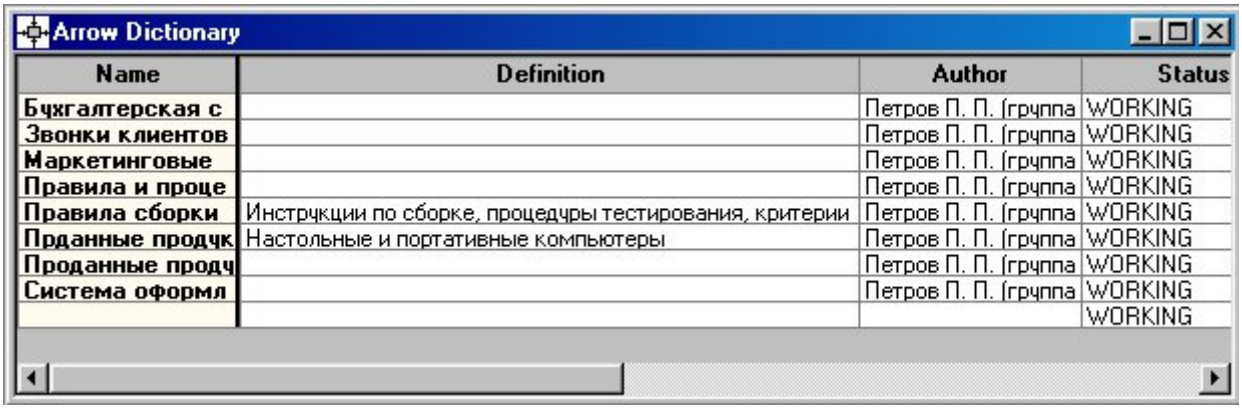


Рисунок 1.30

Стрелку нельзя удалить из словаря, если она используется на какой-либо диаграмме. Если удалить стрелку из диаграммы, из словаря она не удаляется. Имя и описание такой стрелки может быть использовано в дальнейшем. Для добавления стрелки необходимо перейти в конец списка и щелкнуть правой кнопкой по последней строке. Возникает новая строка, в которой нужно внести имя и свойства стрелки.

6 Создайте новые внутренние стрелки так, как показано на рисунке 1.31.

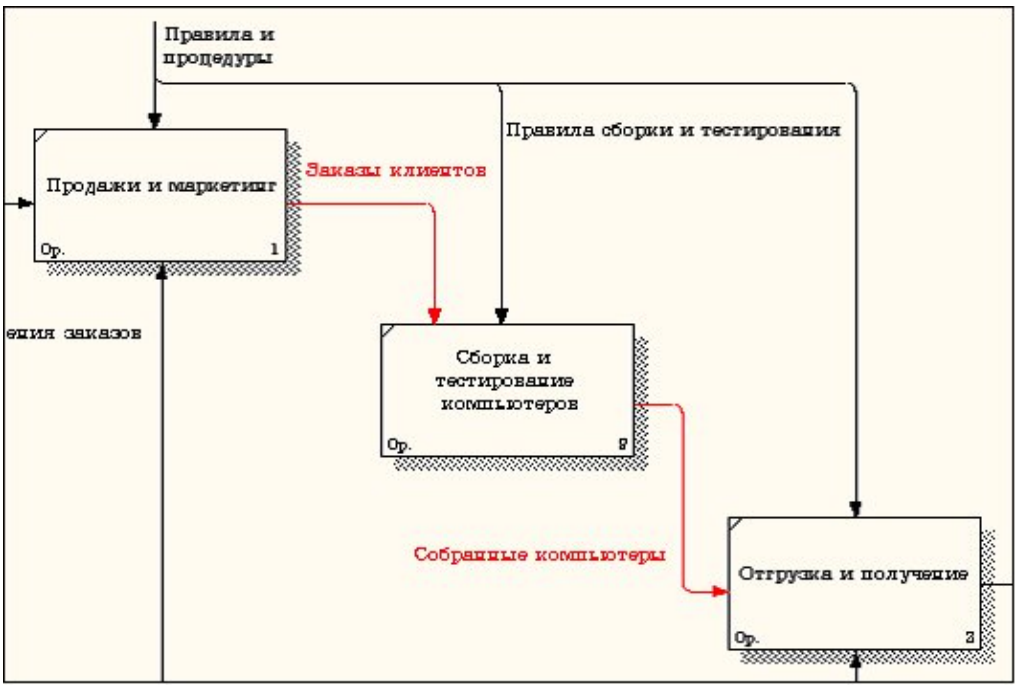


Рисунок 1.31.

Создайте стрелку обратной связи (по управлению) **"Результаты сборки и тестирования",** идущую от работы **"Сборка и тестирование компьютеров"** к работе **"Продажи и маркетинг"**. Измените, при необходимости, стиль стрелки (толщина линий) и установите опцию **Extra Arrowhead (**Дополнительный Наконечник стрелы**)** (из контекстного меню). Методом **drag&drop** перенесите имена стрелок так, чтобы их было удобнее читать. Если необходимо, установите из контекстного меню **Squiggle** (Загогулину). Результат возможных изменений показан на рисунке 1.32.

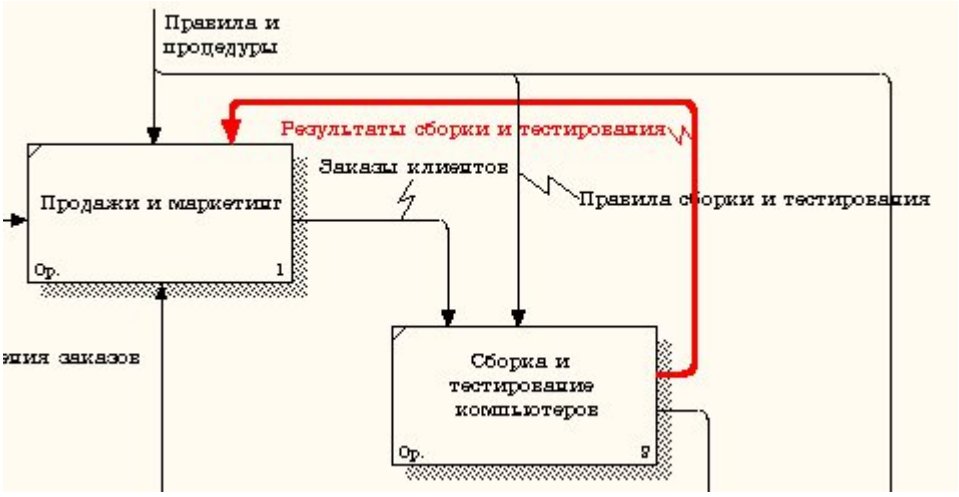


Рисунок 1.32.

Создайте новую граничную стрелку выхода **"Маркетинговые материалы"**, выходящую из работы **"Продажи и маркетинг".** Эта стрелка автоматически не попадает на диаграмму верхнего уровня и имеет квадратные скобки на наконечнике  (рисунок 1.33).

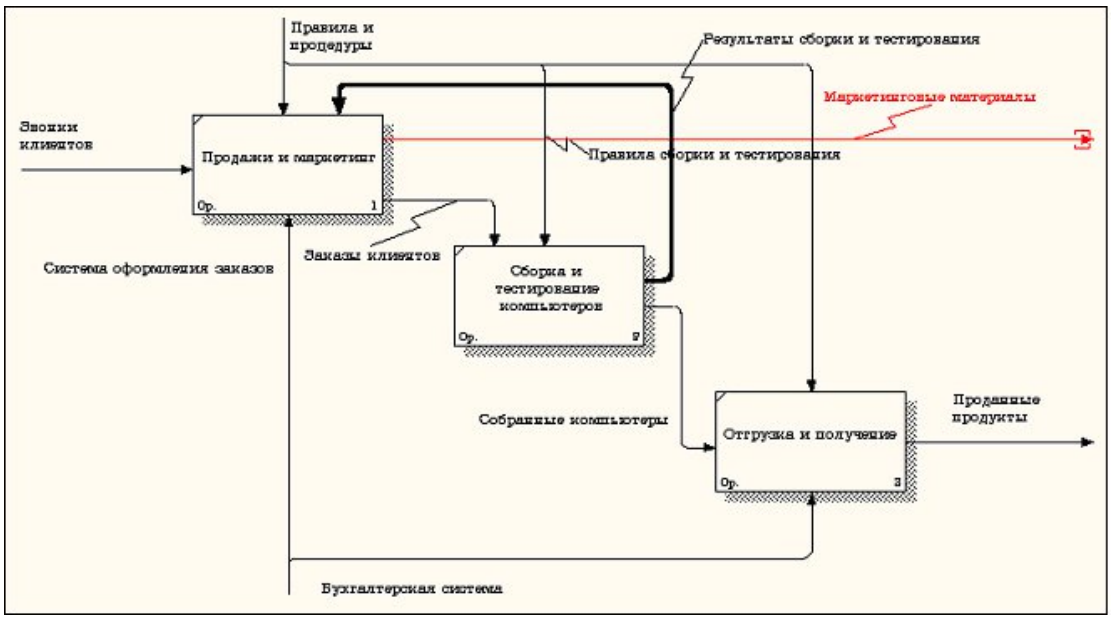


Рисунок 1.33.

Щелкните правой кнопкой мыши по квадратным скобкам и выберите пункт меню **Arrow Tunnel** (рисунок 1.34).

Рисунок 1.34.

В диалоговом окне **Border Arrow Editor (**Редактор Граничных Стрелок**)** выберите опцию **Resolve it to Border Arrow** (Разрешить как Граничную Стрелку) (рисунок 1.35).

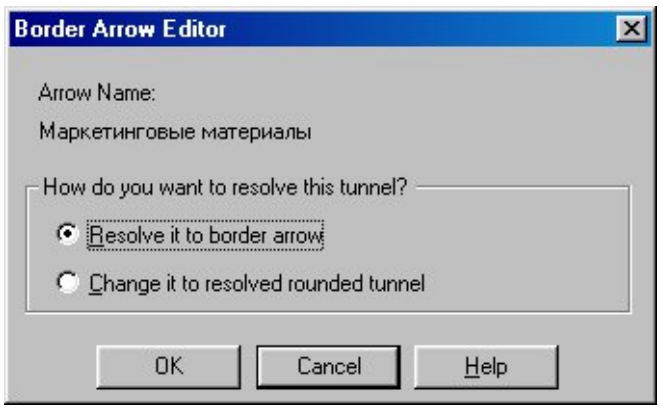


Рисунок 1.35.

Для стрелки **"Маркетинговые материалы"** выберите опцию **Trim (**Упорядочить**)** из контекстного меню. Результат показан на рис. 1.36.

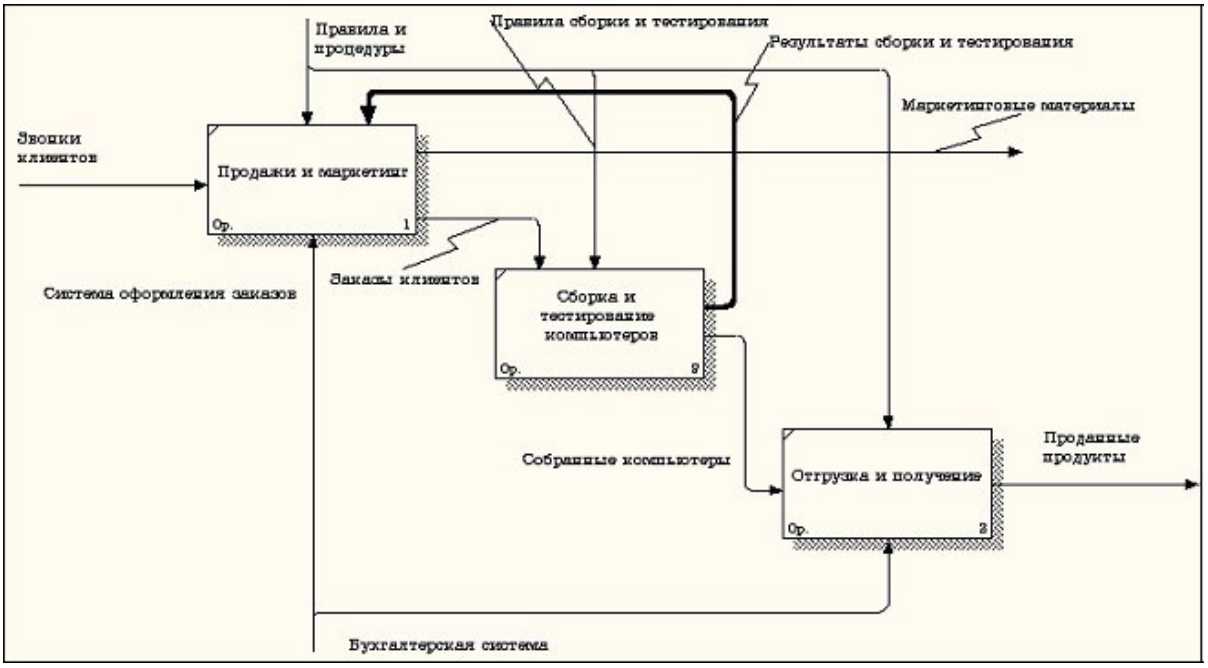


Рисунок 1.36.

Контрольные вопросы

При защите лабораторной работы студент должен ориентироваться в проделанной работе, знать:

- Цели и задачи предпроектного обследования.
- Методологию функционального моделирования.
- Как определять границы моделирования
- Как определять точку зрения при построении модели.
- Принцип декомпозиции.

Студент должен уметь изменить модель (добавить новые или удалить элементы) по требованию преподавателя.

Лабораторная работа №3
«Построение моделей в BPWin. Построение IDEF3, DFD диаграмм.»

Цель работы:

Освоение принципов построения IDEF3, DFD диаграмм. Приобретение практических навыков в создании IDEF3, DFD диаграмм.

Создание диаграммы IDEF3

1) Перейдите на диаграмму A2 и декомпозируйте работу «Сборка настольных компьютеров». В диалоге Activity Box Count (рис. 1) установите число работ 4 и нотацию IDEF3.

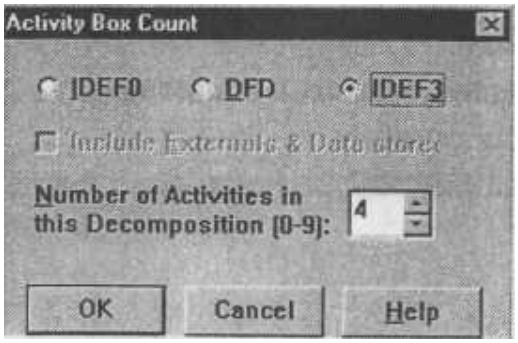


Рис. 1. Выбор нотации IDEF3 в диалоге Activity Box Count

Возникает диаграмма IDEF3, содержащая работы (UOW). Правой кнопкой мыши щелкните по работе, выберите в контекстном меню Name и внесите имя работы «Подготовка компонентов». Затем во вкладке Definition внесите определение "Подготавливаются все компоненты компьютера согласно спецификации заказа".

2) Во вкладке UOW внесите свойства работы (табл. 1).

Таблица 1. Свойства UOW

Objects	Компоненты: винчестеры, корпуса, материнские платы, видеокарты, звуковые карты, дисководы CD-ROM и флоппи, модемы, программное обеспечение
Facts	Доступные операционные системы: Windows 98, Windows NT, Windows 2000
Constrains	Установка модема требует установки дополнительного программного обеспечения

3) Внесите в диаграмму еще 3 работы (кнопка). Внесите имена работ:

- Установка материнской платы и винчестера;
- Установка модема;
- Установка дисковода CD-ROM;
- Установка флоппи- дисковода;
- Инсталляция операционной системы;
- Инсталляция дополнительного программного обеспечения.

4) С помощью кнопки – палитры инструментов создайте объект ссылки. Внесите имя объекта внешней ссылки "Компоненты". Свяжите стрелкой объект ссылки и работу "Подготовка компонент".

5) Свяжите стрелкой работы "Подготовка компонентов" (выход) и "Установка материнской платы и винчестера". Измените стиль стрелки на Object Flow.

В IDEF3 имя стрелки может отсутствовать, хотя BPwin показывает отсутствие имени как ошибку. Результат показан на рис. 2.

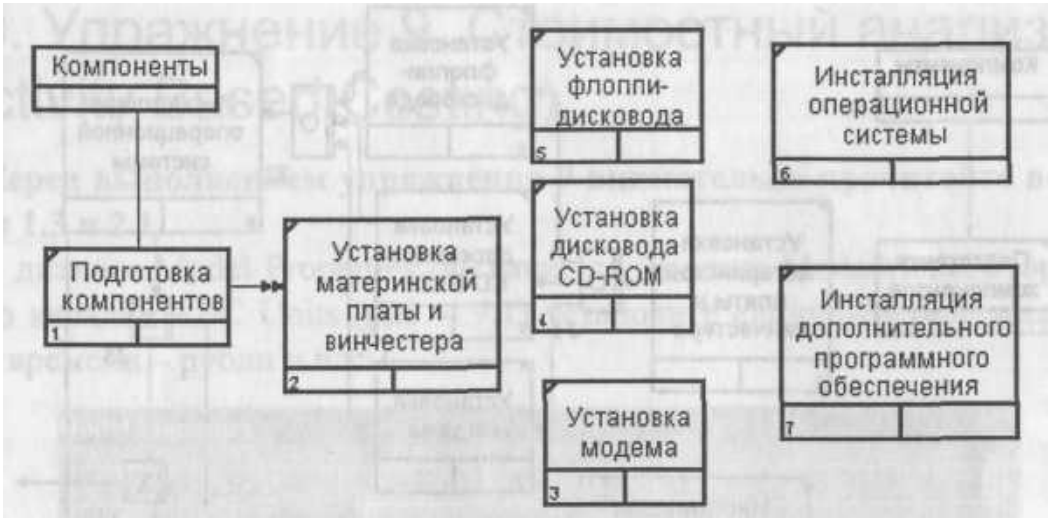


Рис. 2. Результат создания UOW и объекта ссылки

6) С помощью кнопки  на палитре инструментов внесите два перекрестка типа "асинхронное или" и свяжите работы с перекрестками, как показано на рис. 3.

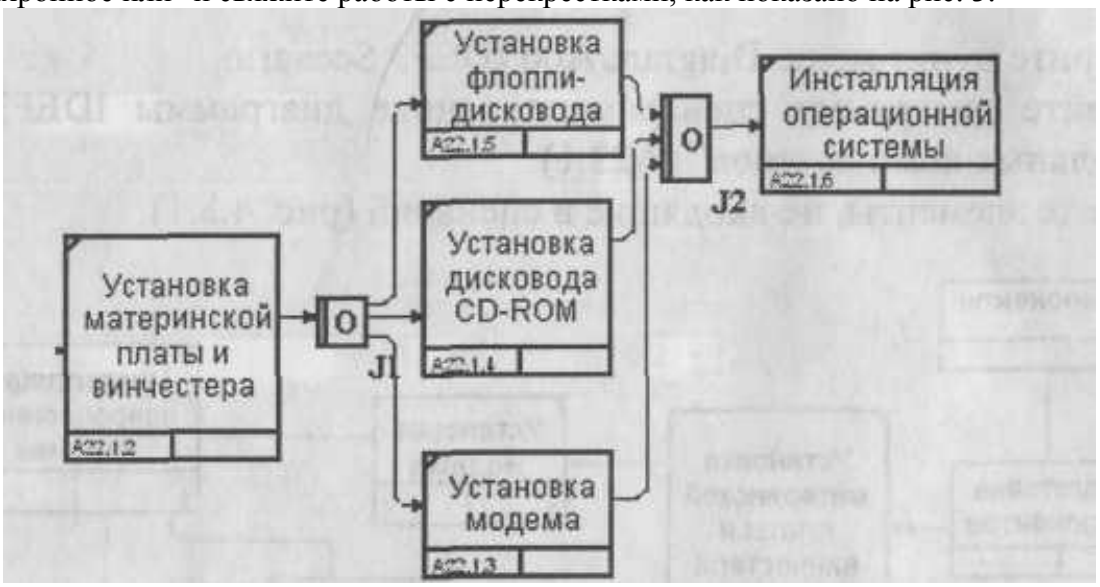


Рис. 3. Диаграмма IDEF3 после создания перекрестков

7) Правой кнопкой щелкните по перекрестку для разветвления (fan-out), выберите Name и внесите имя "Компоненты, требуемые в спецификации заказа".

Создайте два перекрестка типа исключающего "ИЛИ" и свяжите работы, как показано на рис. 4.

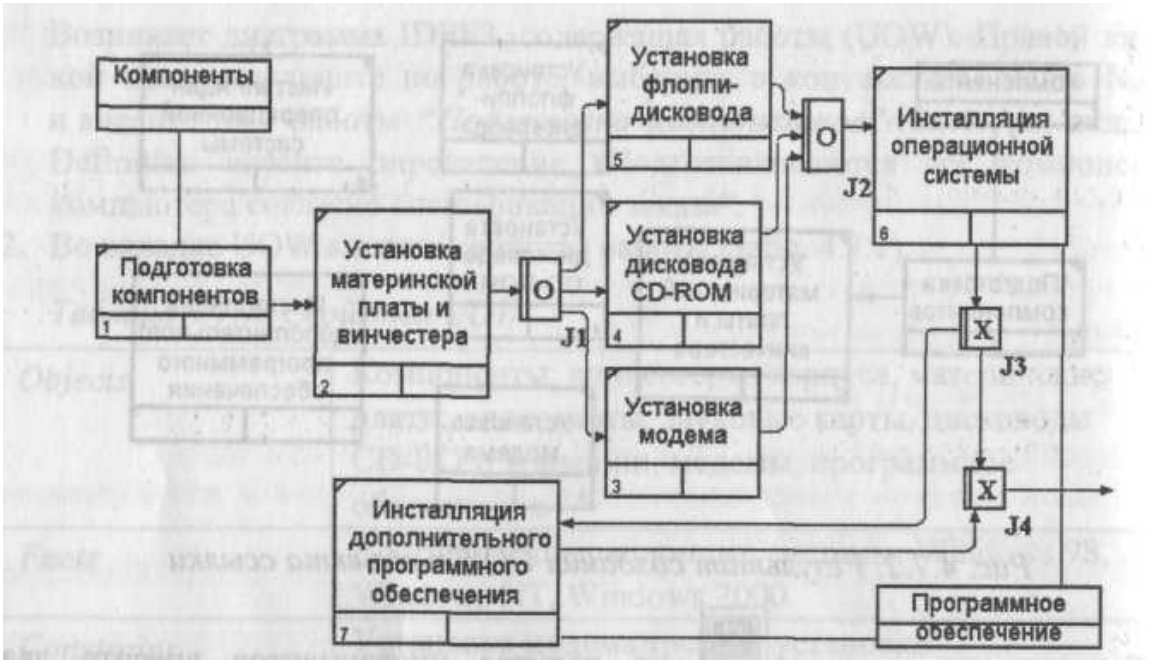


Рис. 4. Результат создания диаграммы IDEF3

Создание сценария

1) Выберите пункт меню Diagram/Add IDEF3 Scenario. Создайте диаграмму сценария на основе диаграммы IDEF3 "Сборка настольных компьютеров" (A22.1).

2) Удалите элементы, не входящие в сценарий (рис. 5).

Рис. 5. Результат создания сценария

Дополнение моделей процессов диаграммой DFD

Эти диаграммы представляют сеть связанных между собой работ. Их удобно использовать для описания документооборота и обработки информации.

DFD описывает:

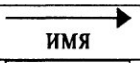
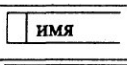
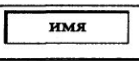
1. функции обработки информации (работы);

2. документы (стрелки, arrow), объекты, сотрудников или отделы, которые участвуют в обработке информации;

3. внешние ссылки (external reference), которые обеспечивают интерфейс с внешними объектами, находящимися за границами моделируемой системы;

4. таблицы для хранения документов (хранилиш;а данных, data store).

Для построения диаграмм DFD в BPWin используется нотация Гейна – Сарсона:

Компонент	Обозначение
Поток данных	
Процесс	
Хранилище	
Внешняя сущность	

Потоки данных являются механизмами, использующимися для моделирования передачи информации (или физических компонентов) из одной части системы в другую. Потоки изображаются на диаграмме именованными стрелками, ориентация которых указывает направление движения информации. Стрелки могут подходить к любой грани прямоугольника работы и могут быть двунаправленными для описания взаимодействия типа «команда-ответ».

Назначение процесса состоит в продуцировании выходных потоков из входных в соответствии с действием, задаваемым именем процесса. Каждый процесс должен иметь уникальный номер для ссылок на него внутри диаграммы. Этот номер молсет использоваться совместно с номером диаграммы для получения уникального индекса процесса во всей модели.

Хранилище данных позволяет на определенных участках определять данные, которые будут сохраняться в памяти между процессами. Фактически хранилище представляет «срезы» потоков данных во времени. Информация, которую оно содержит, может использоваться в любое время после ее определения, при этом данные могут выбираться в любом порядке. Имя хранилища должно идентифицировать его содержимое. В случае, когда поток данных входит в хранилище или выходит из него и его структура соответствует структуре хранилища, он должен иметь то же самое имя, которое нет необходимости отражать на диаграмме.

Внешняя сущность представляет сущность вне контекста системы, являющуюся источником или приемником данных системы. Предполагается, что объекты, представленные такими узлами, не должны участвовать ни в какой обработке. Внешние сущности изображаются в виде прямоугольника с тенью и обычно располагаются по краям диаграммы. Одна внешняя сущность может быть использована многократно на одной или нескольких диаграммах.

Для дополнения модели IDEFO диаграммой DFD нужно в процессе декомпозиции в диалоге Activity Box Count указать тип диаграммы DFD.

Контрольные вопросы

При защите лабораторной работы студент должен ориентироваться в проделанной работе, знать:

- Методологию функционального моделирования с использованием IDEF3 и DFD.
- Как создать диаграммы IDEF3
- Как создать сценарии
- Как дополнить модели процессов диаграммой DFD.
- Принцип декомпозиции.

Студент должен уметь изменить модель (добавить новые или удалить элементы) по требованию преподавателя.

Лабораторная работа № 4
«Построение моделей в ERWin.»

Цель работы:

Приобретение практических навыков построения логической модели данных выбранной предметной области в нотации IDEF1X в ERWin.

IDEF1X основан на подходе Чена и позволяет построить модель данных, эквивалентную реляционной модели в третьей нормальной форме. Нотация Чена и сам процесс построения диаграмм сущность-связь изучалась в курсе "Организация баз данных и знаний", поэтому здесь мы рассмотрим только отличия IDFE1X от нотации Чена.

Сущность (Entity) - реальный либо воображаемый объект, имеющий существенное значение для рассматриваемой предметной области. Каждая сущность должна иметь наименование, выраженное существительным в единственном числе. Каждая сущность должна обладать уникальным идентификатором. Каждый экземпляр сущности должен однозначно идентифицироваться и отличаться от всех других экземпляров данного типа сущности.

Атрибут (Attribute) - любая характеристика сущности, значимая для рассматриваемой предметной области и предназначенная для квалификации, идентификации, классификации, количественной характеристики или выражения состояния сущности. Наименование атрибута должно быть выражено существительным в единственном числе.

Связь (Relationship) - поименованная ассоциация между двумя сущностями, значимая для рассматриваемой предметной области.

В методе IDEF1X все сущности делятся на зависимые и независимые от идентификаторов. Сущность является независимой от идентификаторов или просто независимой, если каждый экземпляр сущности может быть однозначно идентифицирован без определения его отношений с другими сущностями. Сущность называется зависимой от идентификаторов или просто зависимой, если однозначная идентификация экземпляра сущности зависит от его отношения к другой сущности. Независимая сущность изображается в виде обычного прямоугольника, зависимая - в виде прямоугольника с закругленными углами.

В IDEF1X существуют следующие виды мощностей связей:

- N мощность - каждый экземпляр сущности-родителя может иметь ноль, один или более одного связанного с ним экземпляра сущности-потомка (по умолчанию);

- P мощность - каждый экземпляр сущности-родителя должен иметь не менее одного связанного с ним экземпляра сущности-потомка;
- Z мощность - каждый экземпляр сущности-родителя должен иметь не более одного связанного с ним экземпляра сущности-потомка;
- конкретное число - каждый экземпляр сущности-родителя связан с некоторым фиксированным числом экземпляров сущности-потомка.

Связь изображается линией, проводимой между сущностью-родителем и сущностью-потомком, с точкой на конце линии у сущности-потомка. По умолчанию мощность связи принимается равной N. Если экземпляр сущности-потомка однозначно определяется своей связью с сущностью-родителем, то связь называется идентифицирующей, в противном случае — неидентифицирующей. Идентифицирующая связь изображается сплошной линией, неидентифицирующая - пунктирной линией.

В ERwin'e при установлении идентифицирующей связи атрибуты первичного ключа родительской сущности автоматически переносятся в состав первичного ключа дочерней сущности. Эта операция называется миграцией атрибутов. В дочерней сущности новые атрибуты помечаются как внешний ключ (FK). При установке неидентифицирующей связи атрибуты первичного ключа родительской сущности мигрируют в состав неключевых полей дочерней сущности.

Построение логической модели данных предприятия по сборке и продаже компьютеров и ноутбуков. Построение модели данных начинается с выделения сущностей данной предметной области. В нашем случае были выделены следующие сущности:

- клиент - человек, который покупает компьютеры
- заказ - список компьютеров, которые покупает клиент
- компьютер
- комплектующие - то, из чего собирают компьютеры
- сотрудник - сотрудник предприятия, собирающий конкретный компьютер

Далее рассмотрим связи между сущностями:

- Клиент - Заказ. Один клиент может делать несколько заказов. При этом если данные о клиенте имеются в базе данных, то он сделал минимум один заказ. Поэтому мощность связи - P. Связь идентифицирующая, т.к. заказ без клиента существовать не может;
- Заказ - Компьютер. В рамках одного заказа клиент может заказать несколько компьютеров, но как минимум заказ должен состоять из одного компьютера. Поэтому мощность связи - P. Связь идентифицирующая, т.к. компьютер без заказа существовать не может;
- Компьютер - Комплектующие. В состав одного компьютера входит много различных комплектующих; один и тот же тип комплектующего может входить в состав разных компьютеров. Мощность связи - много ко многим. В IDEF1X такой тип связи отсутствует, поэтому вводим промежуточную (ассоциативную) сущность - Конфигурация. Мощность связи между сущностями Компьютер и Конфигурация - P, поскольку у любого компьютера должна быть конфигурация, мощность между сущностями Комплектующие и Конфигурация - N, поскольку какие-то комплектующие еще могут быть не установлены ни в один компьютер. Связь в обоих случаях идентифицирующая, т.к. конфигурация компьютера не может существовать без привязки к самому компьютеру и к комплектующим;

- Комплектующие - Тип комплектующих. Поскольку перечень типов комплектующих, которые могут быть установлены в компьютер, ограничен, но используется очень часто, то мы приняли решение создать еще одну сущность - Тип комплектующих. Мощность связи - P. Связь идентифицирующая;
- Компьютер - Сотрудник. Каждый компьютер собирается каким-то одним сотрудником. Какие-то сотрудники могут собирать множество компьютеров. Мощность связи - N. Тип связи - неидентифицирующая, поскольку экземпляр сущности Компьютер уже может существовать, но за ним еще может быть не закреплен ни один сотрудник. Именно из этих же соображений в свойствах этой связи мы выбрали переключатель "Nulls Allowed" (на диаграмме это отображается в виде незакрашенного ромбика со стороны сущности-родителя.

Итоговая диаграмма показана на рис. 1:

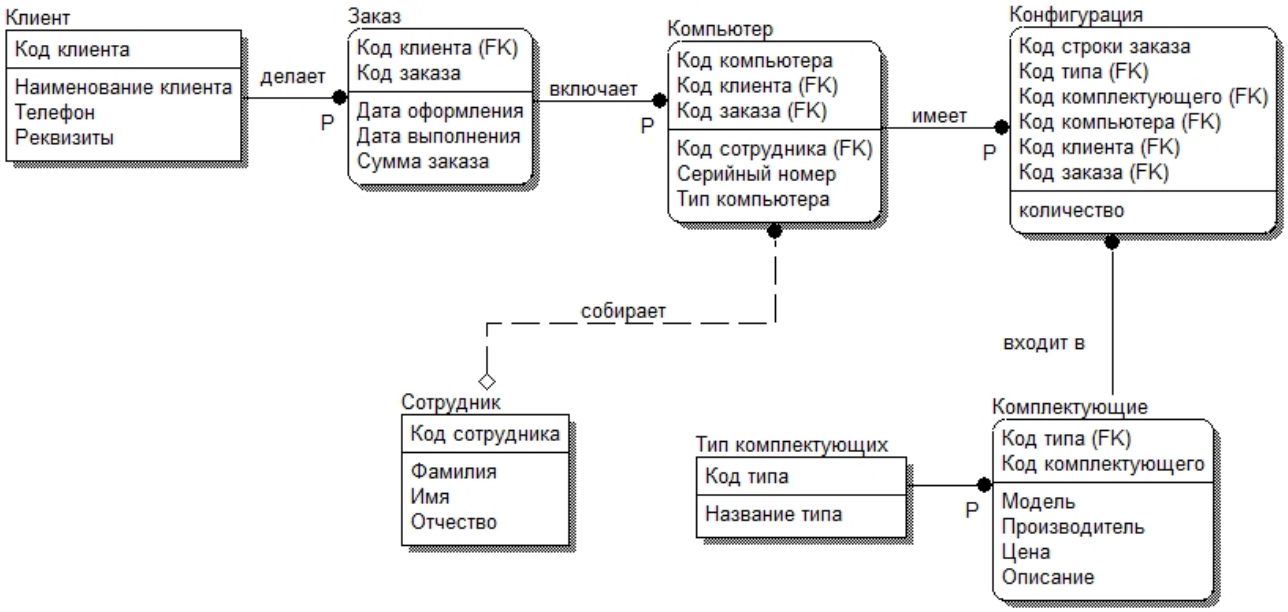


Рисунок 1. Логическая модель данных предприятия по сборке компьютеров и ноутбуков

Контрольные вопросы

1. Какие типы сущностей вы знаете?
2. Какие типы связей вы знаете?
3. Какие виды атрибутов вы знаете?
3. Какие виды мощностей связей вы знаете?

Лабораторная работа №5
«Создание и заполнение таблиц в Microsoft SQL Server 2012.»

Цель работы:

- Изучение основных конструкций структурированного языка запросов SQL.
- Изучения среды MS SQL Server Management Studio.
- Приобретение навыков проектирования структур данных.

Базы данных составляют основу для построения информационных систем любого масштаба и предназначения. В теории баз данных одними из основных являются вопросы,

связанные с анализом предметной области и моделированием структуры данных, управлением данными и их анализом.
Основой любой базы данных является реализованная в ней модель данных, представляющая собой множество структур данных, ограничений целостности и операций манипулирования данными. С помощью модели данных могут быть представлены объекты предметной области и существующие между ними связи.
Результатом лабораторной работы будет создание реляционной базы данных на основе MS SQL Server 2005.
В реляционной базе данных данные представлены в виде собрания таблиц. Таблица состоит из определенного числа столбцов (полей) и произвольного числа строк (записей).
Планируемая база данных будет представлять собой информационное хранилище данных об успеваемости студентов и состоять из следующих таблиц:

- **Speciality** (специальность)
- **Course** (курс)
- **Group** (группа)
- **Discipline** (дисциплина)
- **Account** (тип отчетности)
- **Mark** (отметка)
- **Status** (академический статус студента)
- **Position** (должность)
- **People** (люди)
- **Student** (студент)
- **Teacher** (преподаватель)
- **SemesterResults** (результаты сессии, семестра)

Структура данных таблиц приведена в Приложении.

Начало работы в Microsoft SQL Server Management Studio

Для создания баз данных используем среду Microsoft SQL Server Management Studio. На запрос соединения с сервером выбираем (рис. 1):

Тип сервера: **Компонент Database Engine**
Имя сервера: **SQL-MS.**
Под таким именем в домене fizmat.vspu.ru. доступна машина, на которой установлены серверные компоненты MS SQL Server 2005. Можно попробовать выбрать сервер из выпадающего списка серверов. Можно также обратиться к этой машине по IP-адресу 192.168.10.152 из локальной сети.
Проверка подлинности: **Проверка подлинности SQL Server.**
Такая настройка позволяет создавать пользователей данного экземпляра SQL Server независимо от компьютера, с которого производится вход.
Имя входа: **studentMBS21.**
Пароль: **student.**

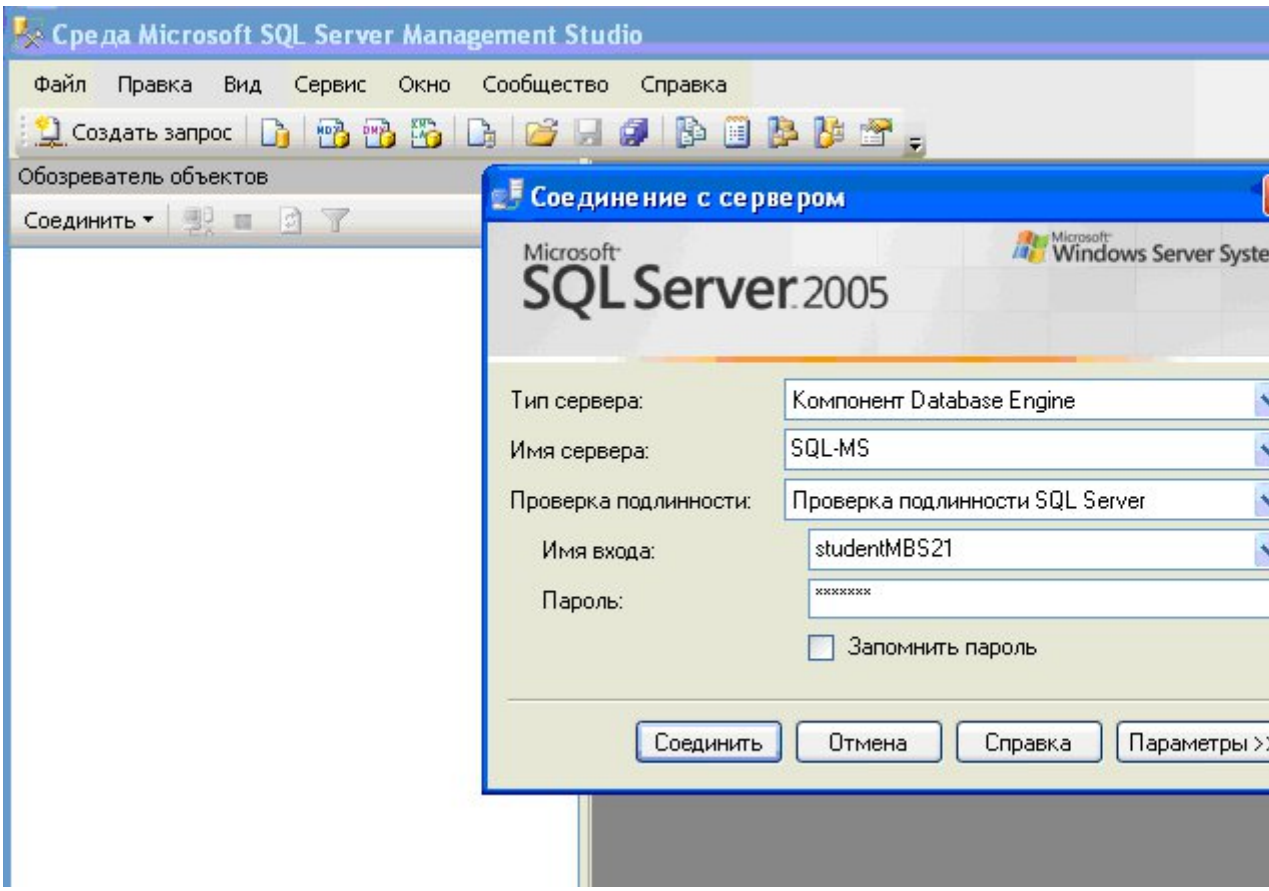


Рисунок 1. Окно входа в Microsoft SQL Server Management Studio 2005

Примечание. Пользователь studentMBS21 обладает большими полномочиями на этом сервере, поэтому пользоваться им надо очень аккуратно. Под этим пользователем мы создадим базу данных, а заполнять её и производить поиск по ней мы будем под другими пользователями. Предпочтительнее всего использовать свою учетную запись в домене fizmat.vspu.ru. В этом случае надо выбирать проверку подлинности Windows.

Теперь нажимаем кнопку «Параметры» и выбираем (рис. 2):

Соединение с базой данных → Обзор сервера… → Пользовательские базы данных → trial_base.

Сетевой протокол → TCP/IP

Нажимаем кнопку «Соединить».

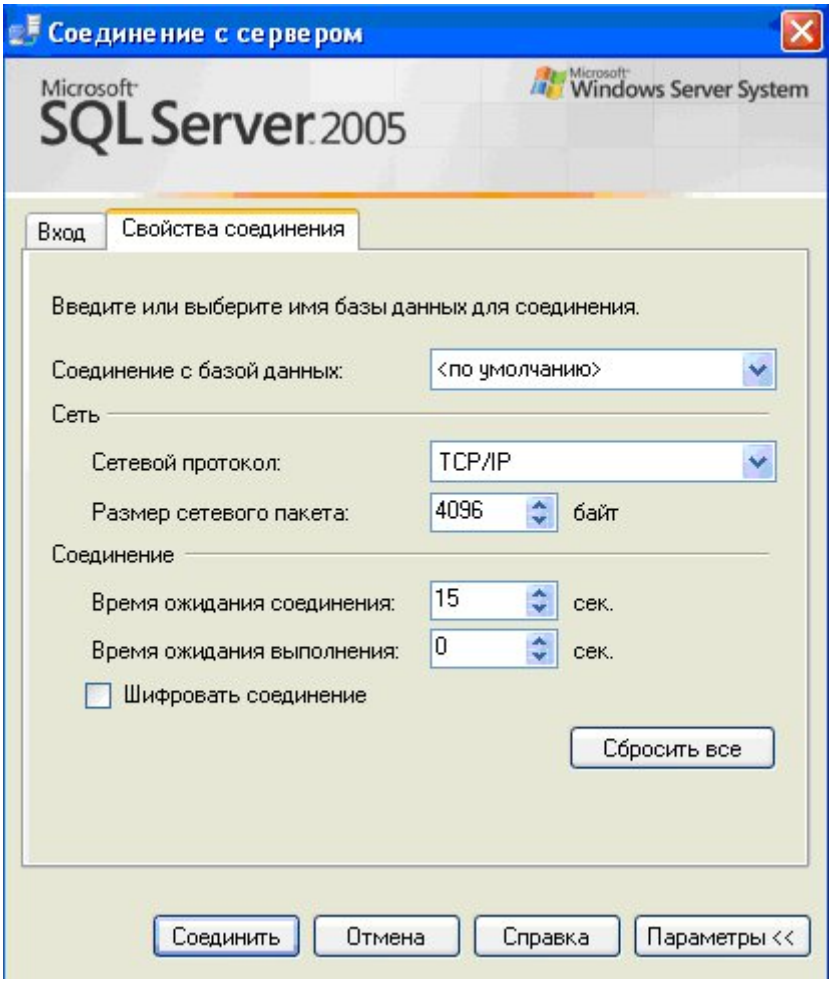


Рисунок 2. Окно входа в Microsoft SQL Server Management Studio 2005 (вкладка Параметры)

Примечание. База данных trial_base является базой данной по умолчанию для пользователя studentMBS21, она была создана при регистрации этого пользователя. В случае, когда права доступа пользователя не ограничены (как в рассматриваемом случае), вкладку Параметры можно не открывать. Если же пользователь имеет доступ только к определенным базам данных, при подключении к серверу нужно одну из этих баз указывать.

После успешного соединения с базой данных на экране видим следующую картинку (рис. 3):

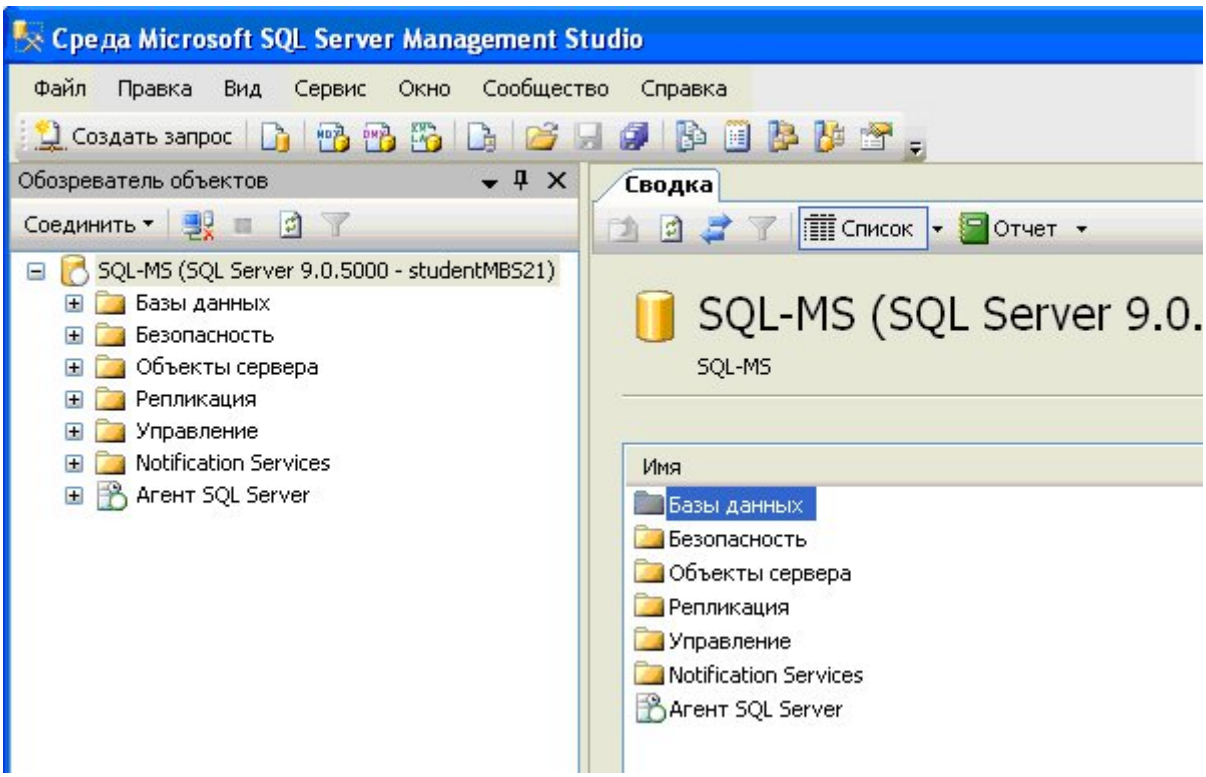


Рисунок 3. Подключение к SQL - серверу установлено

Среда MS SQL Management Studio предоставляет удобный инструментарий для создания, редактирования, заполнения баз данных. Но настоящие профессионалы в своей работе редко пользуются этой средой, а для выполнения своих задач используют SQL-запросы. Мы будем пользоваться, когда это удобно и наглядно, графическим режимом, но основной упор будем делать на освоении базы языка SQL.

Создание базы данных в среде Microsoft SQL Server Management Studio

В разделе «Базы данных» правой кнопкой выбираем «Создать базу данных…» (рис. 4). Назовем базу данных по индексу группы – mbs21. Владельцем базы данных назначим пользователя, под именем которого был произведен вход – studentMBS21. В разделе «Параметры» выбираем тип сортировки Cyrillic_General_BIN (для примера), нажимаем ОК.

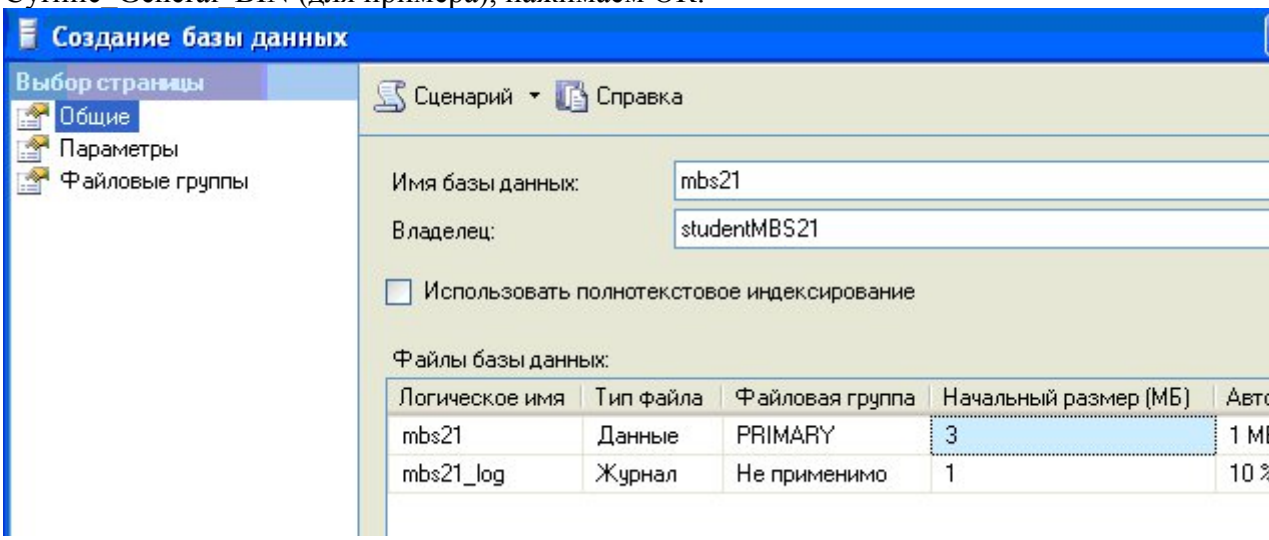
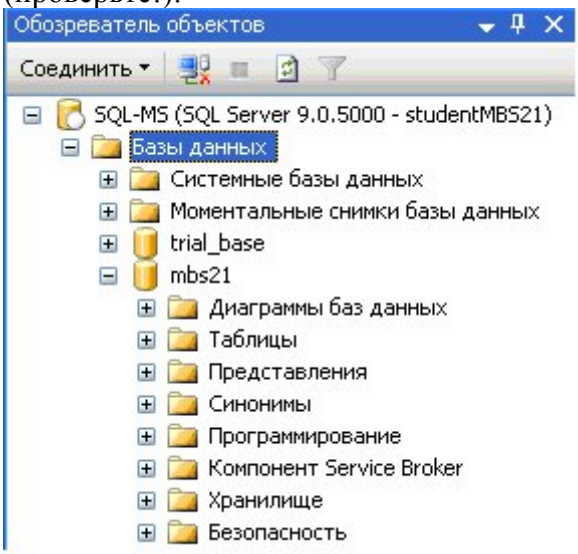


Рисунок 4. Создание базы данных

- В разделе «Базы данных» Обозревателя объектов появилась вновь созданная mbs21 (проверьте!):



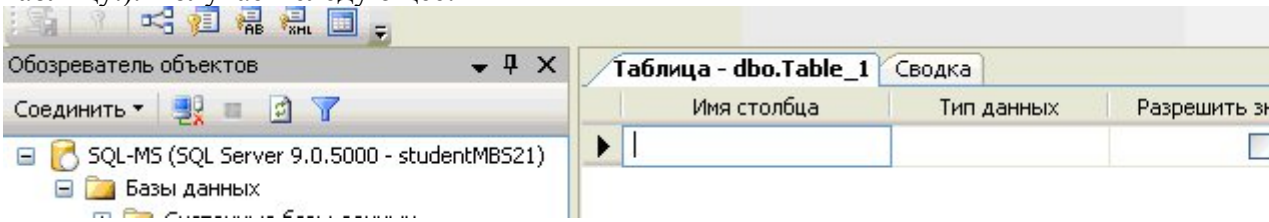
- Рисунок 5. Обозреватель объектов

Создание таблиц базы данных в среде Microsoft SQL Server Management Studio

- Начнем с создания таблицы Speciality. Структура таблицы приведена ниже:

Имя поля (столбца)	Содержание	Тип данных	Возможность содержать NULL
Num	Первичный ключ	int	нет
Name	Название специальности	varchar(60)	нет

- В реляционных базах данных первичный ключ используется как уникальный идентификатор записи. Это поле является обязательным, оно используется для связи таблиц по внешним ключам (примеры такого связывания будут рассмотрены далее). Первичный ключ должен иметь целочисленный тип (в данном случае - *int*). Во втором поле будет храниться название специальности - некоторая строка, поэтому мы выбираем для этого поля тип *varchar(60)*. Число в скобках означает максимальное число символов в строке. Детальную информацию об этих типах можно посмотреть в справке.
- Простейшим образом можно создавать таблицы средствами MS SQL Server Management Studio (правая кнопка мыши на заголовке «Таблицы» > Создать таблицу.). Получаем следующее:



- Рисунок 6. Создание таблицы
- Вводим имя первого столбца Num (первичный ключ – в том столбце хранится номер записи), выбираем из выпадающего списка тип данных int. Первичный ключ не может быть пустым, поэтому и оставляем неотмеченным поле «Разрешить

значения null». Затем аналогичным образом вводим имя второго столбца, задаем тип, запрещаем полю иметь значение null. Таблица принимает следующий вид:

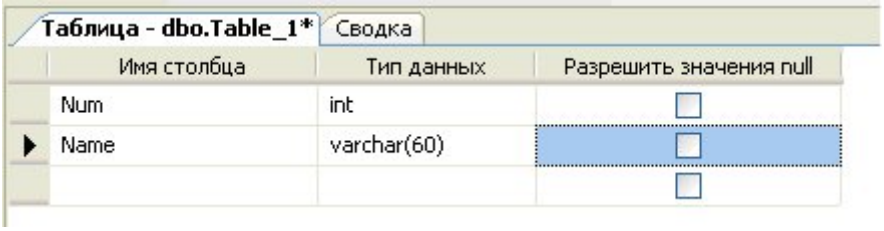


Рисунок 7.

Теперь необходимо указать, что поле *Num* будет являться первичным ключом. Правой кнопкой мыши щелкаем по этому полю и выбираем «Задать первичный ключ»:

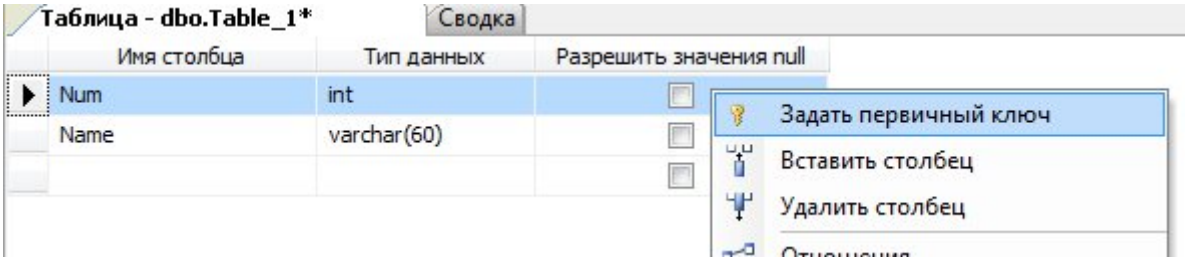


Рисунок 8.

Сохраняем таблицу под именем Speciality (после этого таблица должна появиться в обозревателе объектов). Теперь можно перейти к заполнению этой таблицы (для этого нужно в обозревателе объектов выбрать эту таблицу и в контекстном меню нажать «Открыть таблицу»):

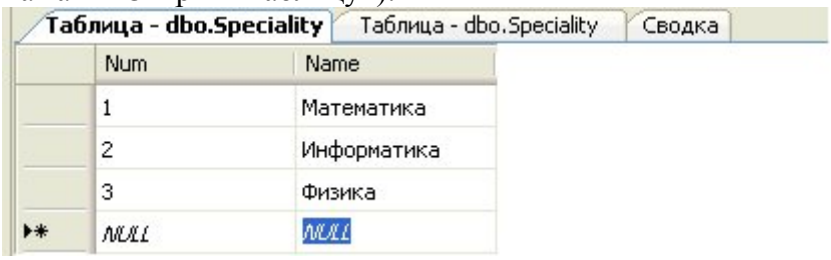


Рисунок 9.

При заполнении вы обнаружите, что каждый раз приходится вводить не только полезную информацию (название специальности), но и номер записи. Чтобы вводить номер записи автоматически, нужно задать спецификацию идентифицирующего столбца. Для этого необходимо в свойствах столбца указать, что данный столбец является идентифицирующим (рис. 10):

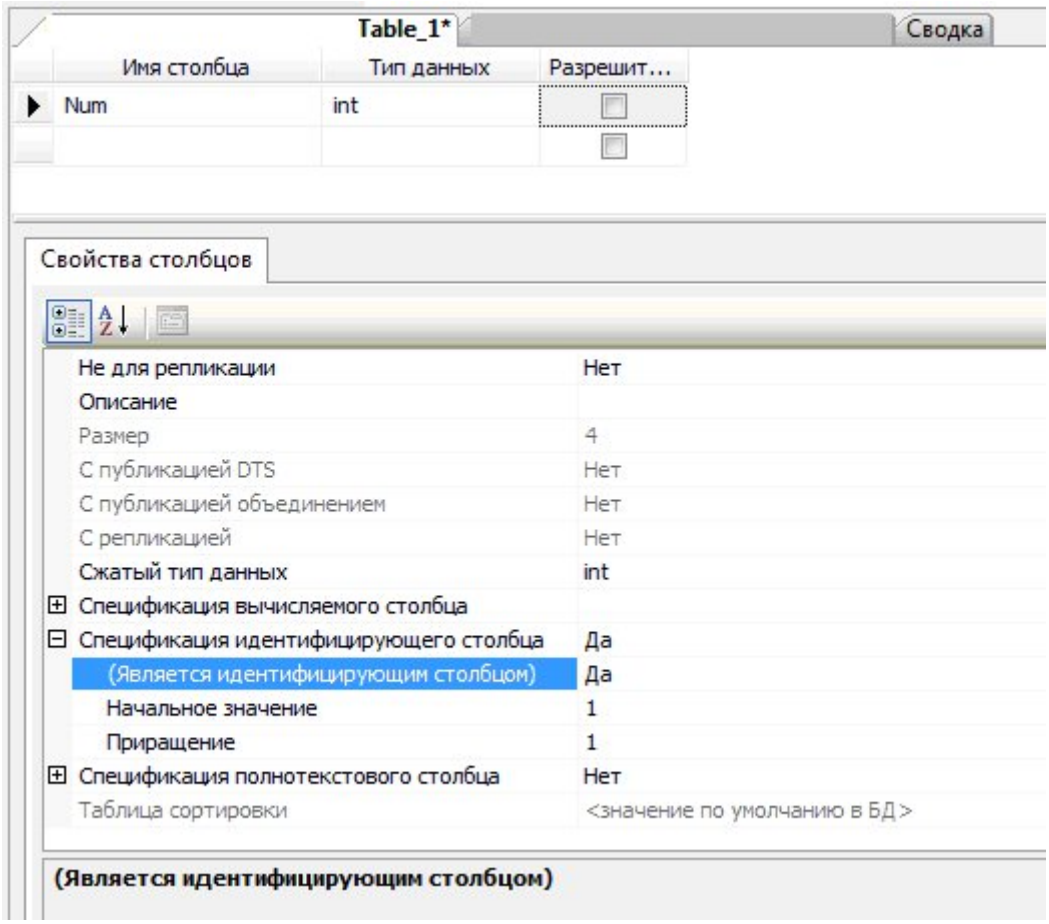


Рисунок 10. Определение свойств идентифицирующего столбца

Создание таблиц базы данных с помощью SQL-запроса

Создание таблиц в графическом режиме, безусловно, удобно, однако не универсально. При использовании других средств разработки баз данных (например, IBM DB2) придется привыкать к новым приемам работы. Использование конструкций языка SQL позволяет работать с базами данных, исходя из единого подхода, в любой среде управления базами данных.

Выберите на панели инструментов «Создать запрос»:

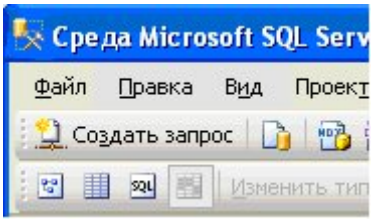


Рисунок 11.

Создадим новую базу данных запросом. Напишем

```
CREATE DATABASE mbs21_query
```

и нажмем F5. В обозревателе объектов должна появиться новая база (если сразу не появилась, то надо выделить мышью раздел «Базы данных» и в контекстном меню выбрать «Обновить»).

- Теперь создадим таблицу Speciality. Упрощенный синтаксис создания таблиц следующий:

```
CREATE TABLE <имя таблицы> (
    <имя столбца 1> <тип данных> [NOT NULL] [DEFAULT <значение по
умолчанию>],
    <имя столбца 2> <тип данных> [NOT NULL] [DEFAULT <значение по
умолчанию>],
    ...
)
```

- Введем новый запрос:

```
/* создание таблицы Специальность*/
USE mbs21_query                                    -- определяем базу
данных, в которую входит таблица
CREATE  TABLE  Speciality(
    Num  INT IDENTITY(1,1) PRIMARY KEY NOT NULL, -- первичный ключ
    NameSpec VARCHAR(60)                           -- название
специальности
)
```

- В обозревателе объектов видим, что таблица действительно создана. Файл с SQL-запросом сохраняем в своей папке (в конце работы необходимо показать запросы, которые были выполнены, преподавателю). Слово IDENTITY(1,1) добавлено, чтобы поле первичного ключа Num автоматически нумеровалось начиная с единицы (фактически, эта конструкция определяет спецификацию идентифицирующего столбца).
- Таким же образом необходимо создать остальные таблицы. Рассмотрим таблицу Course.
- Таблица **Course** (курс)

Имя поля (столбца)	Содержание	Тип данных	Возможность содержать NULL
Num	Первичный ключ	int	нет
Name	Название специальности	varchar(60)	нет
YearEntry	Год поступления	int	нет
YearFinal	Год выпуска	int	да
Speciality	Специальность (внешний ключ ссылается на первичный ключ таблицы Speciality)	int	нет

- Эта таблица содержит поле *Speciality*, которое ссылается на первичный ключ таблицы Speciality. Чтобы создать такую таблицу, необходимо выполнить запрос:

```
/* создание таблицы Курс */
USE mbs21_query                                  -- определяем базу
данных, в которую входит таблица
CREATE  TABLE  Course(
    Num  INT IDENTITY(1,1) PRIMARY KEY NOT NULL, -- первичный ключ
    YearEntry INT NOT NULL,                      -- год поступления
    YearFinal INT,                               -- год окончания
    Speciality INT FOREIGN KEY REFERENCES Speciality(Num) --
специальность,
    -- ссылка по внешнему ключу на поле Num таблицы Speciality
)
```

- **Примечание.** Ссылку можно создать только на существующую таблицу. Задать ссылку по внешнему ключу можно и после создания таблицы (подробно будет рассмотрено в следующей лабораторной работе).

- **Задание.** Создайте все остальные таблицы, указанные в Приложении, используя SQL – запросы.

Контрольные вопросы

1. Каким образом можно получить доступ к MS SQL Server?
2. С помощью каких средств можно создать таблицу для MS SQL Server?
3. Что такое первичный ключ?
4. Каким образом можно создать автоматическую нумерацию строк таблицы?
5. Что означают Not Null?

Лабораторная работа № 6
«Создание запросов и фильтров в Microsoft SQL Server 2012.»

Цель работы:

Приобрести практические навыки в создании запросов и фильтров в Microsoft SQL Server 2012.»

Цель работы:

- научиться создавать запросы и фильтры.

Перейдем к созданию статических запросов. В обозревателе объектов "Microsoft SQL Server 2008" все запросы БД находятся в папке **"Views"** (рис. 8.1).

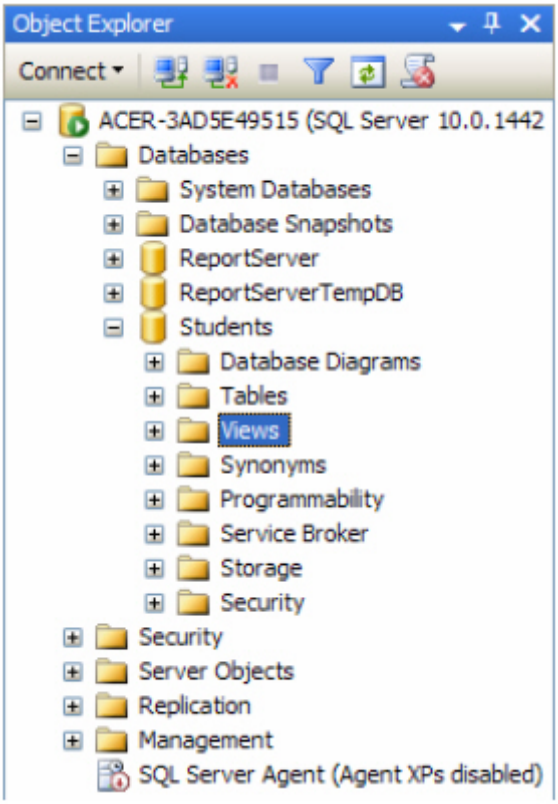


Рис. 8.1.

Создадим запрос **"Запрос Студенты+Специальности"**, связывающий таблицы **"Студенты"** и **"Специальности"** по полю связи **"Код специальности"**. Для создания нового запроса необходимо в обозревателе объектов в БД **"Students"** щелкнуть **ПКМ** по папке **"Views"**, затем в появившемся меню выбрать пункт **"New View"**. Появится окно **"Add Table"** (Добавить таблицу), предназначенное для выбора таблиц и запросов, участвующих в новом запросе (рис. 8.2).

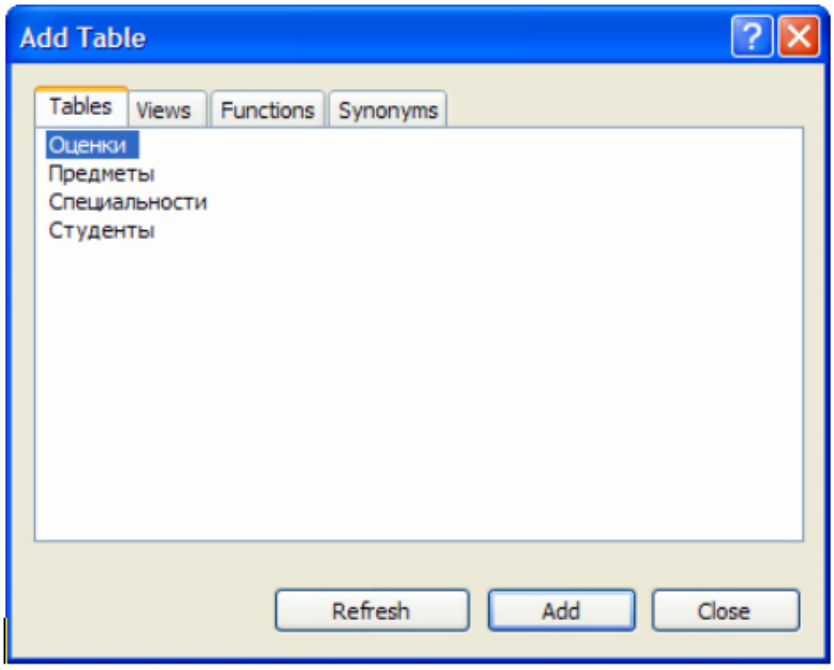


Рис. 8.2.

Добавим в новый запрос таблицы **"Студенты"** и **"Специальности"**. Для этого в окне **"Add Table"** выделите таблицу **"Студенты"** и нажмите кнопку **"Add"** (Добавить). Аналогично добавьте таблицу **"Специальности"**. После добавления таблиц участвующих в запросе закройте окно **"Add Table"** нажав кнопку **"Close"** (Закрыть). Появится окно конструктора запросов (рис. 8.3).

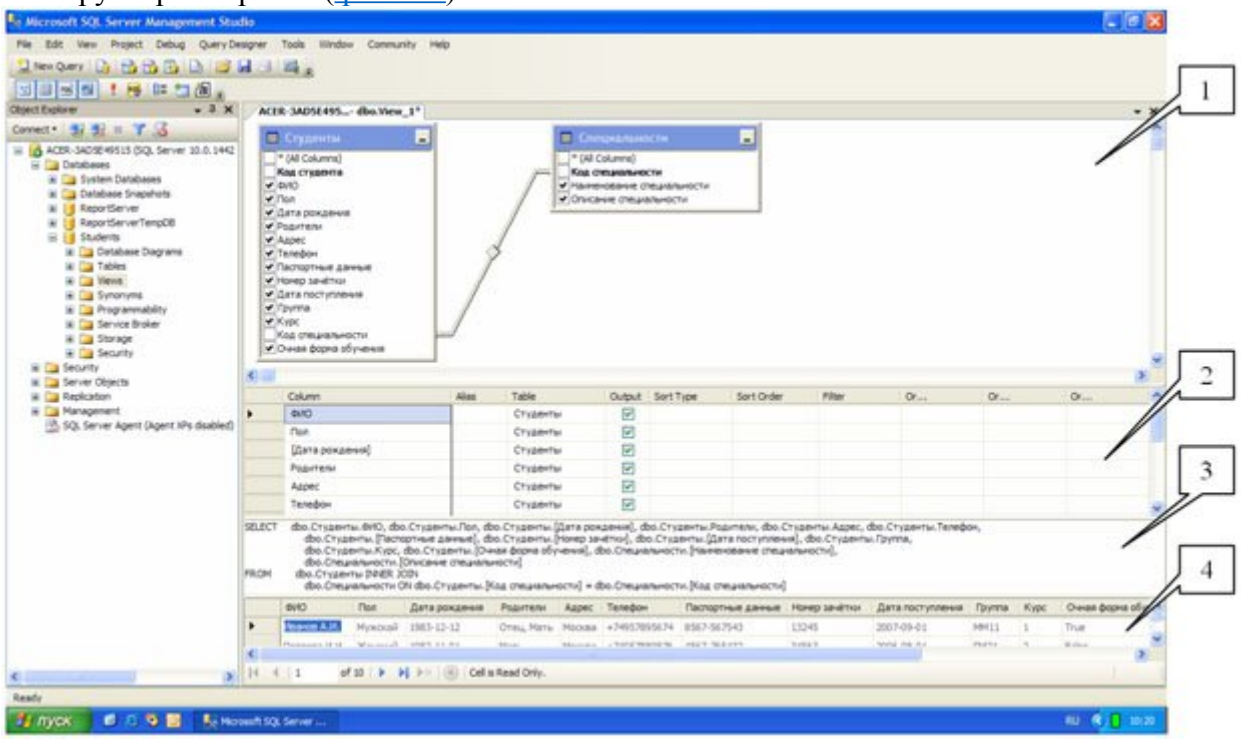


Рис. 8.3.

Замечание: Окно конструктора запросов состоит из следующих панелей:

1. **Схема данных -** отображает поля таблиц и запросов, участвующих в запросе, позволяет выбирать отображаемые поля, позволяет устанавливать связи между участниками запроса по специальным полям связи. Эта панель включается и выключается следующей кнопкой на панели инструментов

2. **Таблица отображаемых полей** - показывает отображаемые поля (столбец **"Column"**), позволяет задавать им псевдонимы (столбец **"Alias"**), позволяет устанавливать тип сортировки записей по одному или нескольким полям (столбец **"Sort Type"**), позволяет задавать порядок сортировки (столбец **"Sort Order"**), позволяет задавать условия отбора записей в фильтрах (столбцы **"Filter"** и **"Or…"**). Также эта таблица позволяет менять порядок отображения полей в запросе. Эта панель включается и выключается следующей кнопкой на панели инструментов

3. **Код SQL** - код создаваемого запроса на языке T-SQL. Эта панель включается и выключается следующей кнопкой на панели инструментов

4. **Результат** - показывает результат запроса после его выполнения. Эта панель включается и выключается следующей кнопкой на панели инструментов


Замечание: Если необходимо снова отобразить окно **"Add Table"** для добавления новых таблиц или запросов, то для этого на панели инструментов **"Microsoft SQL Server 2008"** нужно нажать кнопку

Замечание: Если необходимо удалить таблицу или запрос из схемы данных, то для этого нужно щелкнуть **ПКМ** и в появившемся меню выбрать пункт **"Remove"** (Удалить).
Теперь перейдем к связыванию таблиц **"Студенты"** и **"Специальности"** по полям связи **"Код специальности"**. Чтобы создать связь необходимо в схеме данных перетащить мышью поле **"Код специальности"** таблицы **"Специальности"** на такое же поле таблицы **"Студенты"**. Связь отобразится в виде ломаной линии соединяющей эти два поля связи (рис. 8.3).
Замечание: Если необходимо удалить связь, то для этого необходимо щелкнуть по ней **ПКМ** и в появившемся меню выбрать пункт **"Remove"**.
Замечание: После связывания таблиц (а также при любых изменениях в запросе) в области кода T-SQL будет отображаться T-SQL код редактируемого запроса.
Теперь определим поля, отображаемые при выполнении запроса. Отображаемые поля обозначаются галочкой (слева от имени поля) на схеме данных, а также отображаются в таблице отображаемых полей. Чтобы сделать поле отображаемым при выполнении запроса необходимо щелкнуть мышью по пустому квадрату (слева от имени поля) на схеме данных, в квадрате появится галочка.
Замечание: Если необходимо сделать поле невидимым при выполнении запроса, то нужно убрать галочку, расположенную слева от имени поля на схеме данных. Для этого просто щелкните мышью по галочке.
Замечание: Если необходимо отобразить все поля таблицы, то необходимо установить галочку слева от пункта **"* (All Columns)"** (Все поля), принадлежащего соответствующей таблице на схеме данных.
Определите отображаемые поля нашего запроса, как это показано на рис. 8.3 (Отображаются все поля кроме полей с кодами, то есть полей связи).
На этом настройку нового запроса можно считать законченной. Перед сохранением запроса проверим его работоспособность, выполнив его. Для запуска запроса на панели инструментов нажмите кнопку

Либо щелкните **ПКМ** в любом месте окна конструктора запросов и в появившемся меню выберите пункт **"Execute SQL"** (Выполнить SQL). Результат выполнения запроса появится в виде таблицы в области результата (рис. 8.3).

Замечание: Если после выполнения запроса результат не появился, а появилось сообщение об ошибке, то в этом случае проверьте, правильно ли создана связь. Ломаная линия связи должна соединять поля **"Код специальности"** в обеих таблицах. Если линия связи соединяет другие поля, то ее необходимо удалить и создать заново, как это описано выше.
Если запрос выполняется правильно, то необходимо сохранить. Для сохранения запроса закройте окно конструктора запросов, щелкнув мышью по кнопке закрытия

расположенной в верхнем правом углу окна конструктора (над схемой данных). Появится окно с вопросом о сохранении запроса (рис. 8.4).

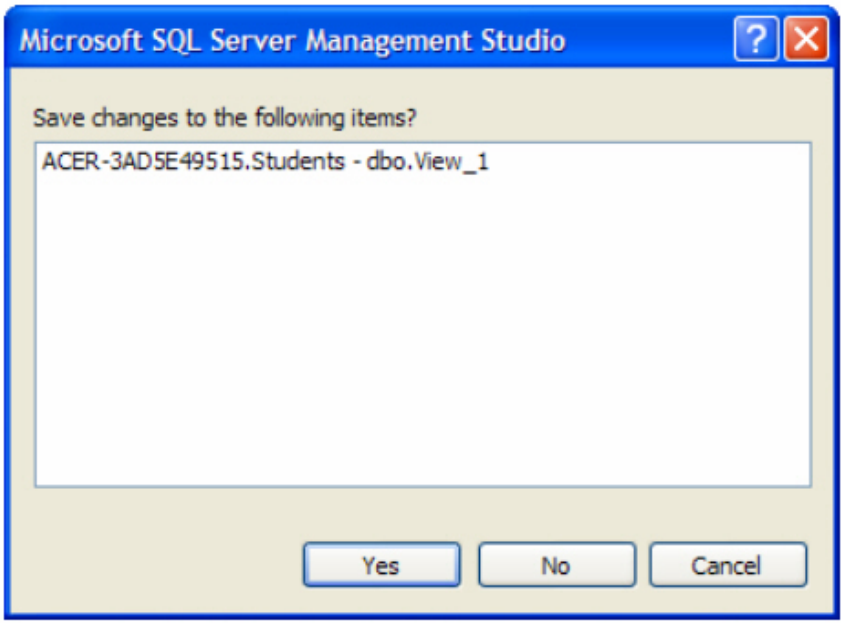


Рис. 8.4.
В данном окне необходимо нажать кнопку **"Yes"** (Да). Появится окно **"Choose Name"** (Выберите имя) (рис. 8.5).

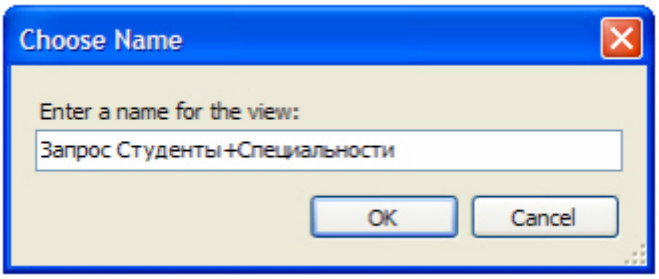


Рис. 8.5.
В данном окне зададим имя нового запроса **"Запрос Студенты+Специальности"** и нажмем кнопку **"Ok"**. Запрос появится в папке **"Views"** БД **"Students"** в обозревателе объектов (рис. 8.6).

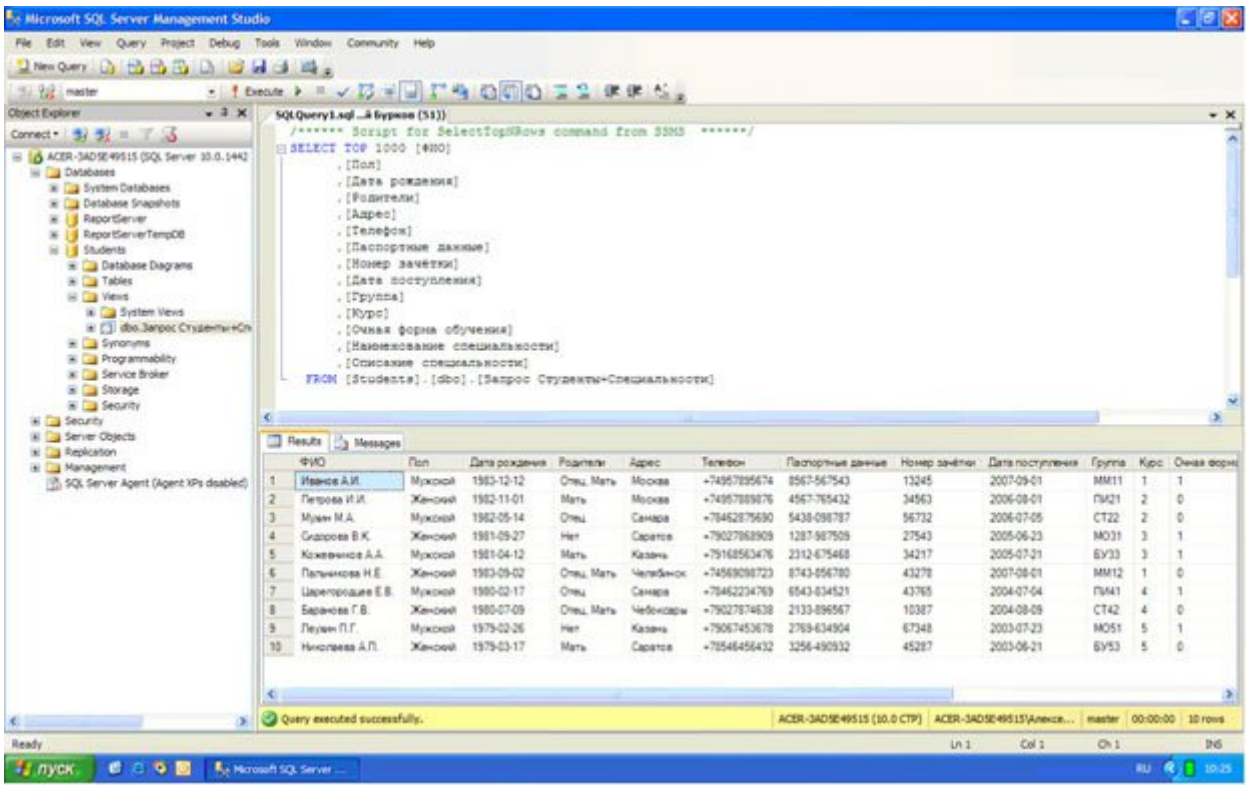


Рис. 8.6.

Проверим работоспособность созданного запроса вне конструктора запросов. Запустим вновь созданный запрос **"Запрос Студенты+Специальности"** без использования конструктора запросов. Для выполнения уже сохраненного запроса необходимо щелкнуть **ПКМ** по запросу и в появившемся меню выбрать пункт **"Select top 1000 rows"** (Отобразить первые 1000 записей). Выполните эту операцию для запроса **"Запрос Студенты+Специальности"**. Результат представлен на рис. 8.6.

Перейдем к созданию запроса **"Запрос Студенты+Оценки"**. В обозревателе объектов в БД **"Students"** щелкните **ПКМ** по папке **"Views"**, затем в появившемся меню выберите пункт **"New View"**. Появиться окно **"Add Table"** (рис. 8.2).

В запросе **"Запрос Студенты+Оценки"** мы связываем таблицы **"Студенты"** и **"Оценки"** по полям связи **"Код студента"**. Следовательно, в окне **"Add Table"** в новый запрос добавляем таблицы **"Студенты"** и **"Оценки"**. Более того, в данном запросе таблица **"Оценки"** связывается с таблицей **"Предметы"** не по одному полю, а по трем полям. То есть поля **"Код предмета 1"**, **"Код предмета 2"** и **"Код предмета 3"** таблицы **"Оценки"** связаны с полем **"Код предмета"** таблицы **"Предметы"**. Поэтому добавим в запрос три экземпляра таблицы **"Предметы"** (по одному экземпляру для каждого поля связи таблицы оценки). В итоге в запросе должны участвовать таблицы **"Студенты"**, **"Оценки"** и три экземпляра таблицы **"Предметы"** (в запросе они будут называться **"Предметы"**, **"Предметы_1"** и **"Предметы_2"**). После добавления таблиц закройте окно **"Add Table"**, появится окно конструктора запросов.

В окне конструктора запросов установите связи между таблицами и определите отображаемые поля, как показано на рис. 8.7.

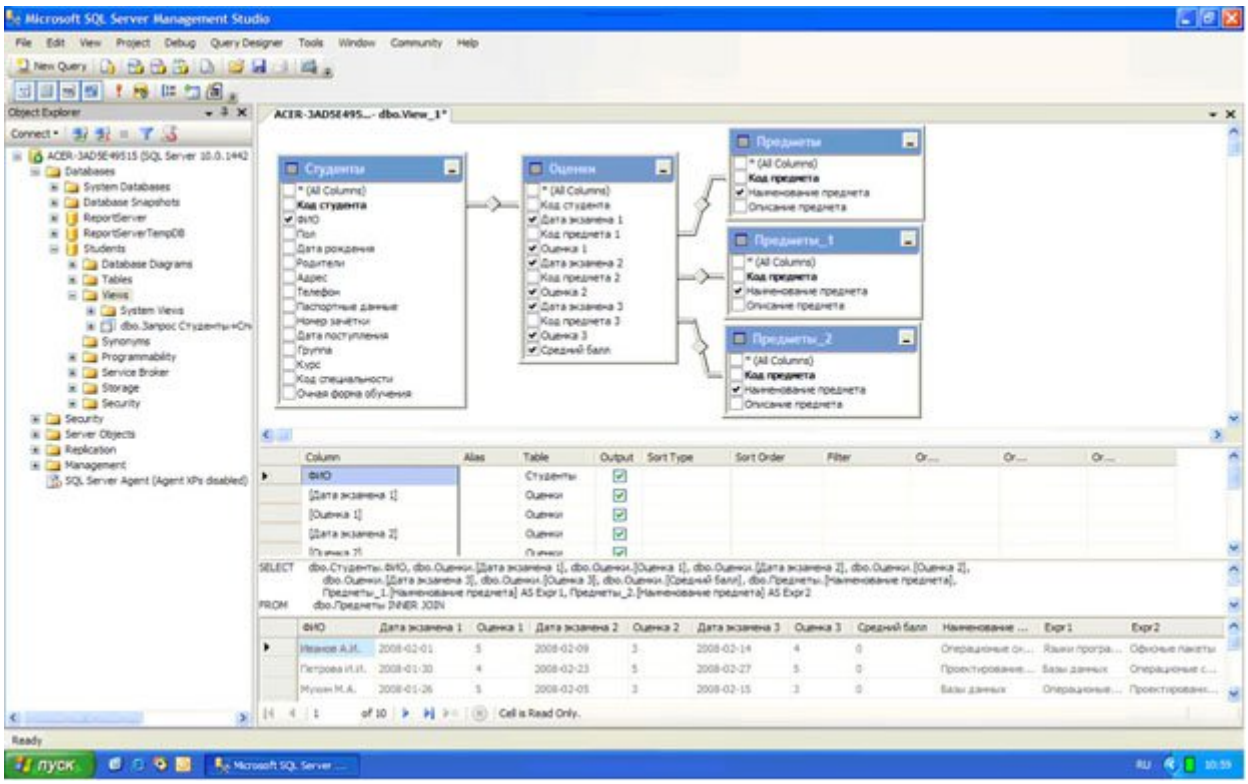
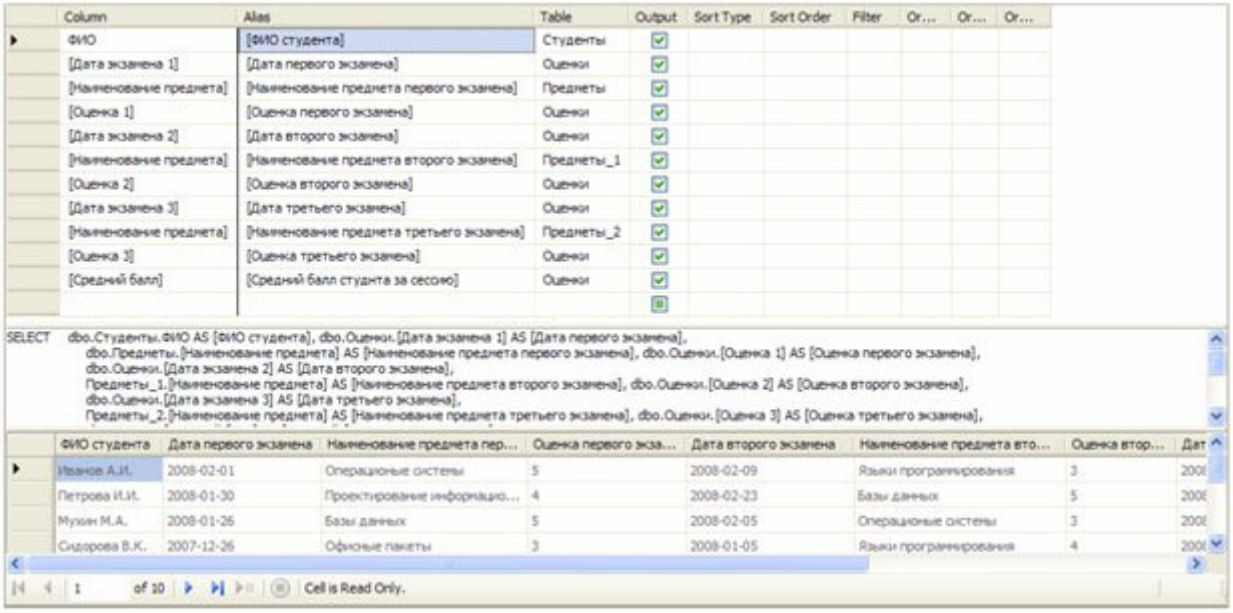


Рис. 8.7.

Теперь поменяем порядок отображаемых полей в запросе, для этого в таблице отображаемых полей необходимо перетащить поля мышью вверх или вниз за заголовок строки таблицы (столбец перед столбцом **"Column"**). Расположите отображаемые поля в таблице отображаемых полей как показано на рис. 8.8.



увеличить изображение

Рис. 8.8.

Задайте псевдонимы для каждого из полей, просто записав псевдонимы в столбце **"Alias"** таблицы отображаемых полей, как на рис. 8.8.

Проверьте работоспособность нового запроса, выполнив его. Обратите внимание на то, что реальные названия полей были заменены их псевдонимами. Закройте окно конструктора запросов. В появившемся окне **"Choose Name"** задайте имя нового запроса **"Запрос Студенты+Оценки"** (рис. 8.9).

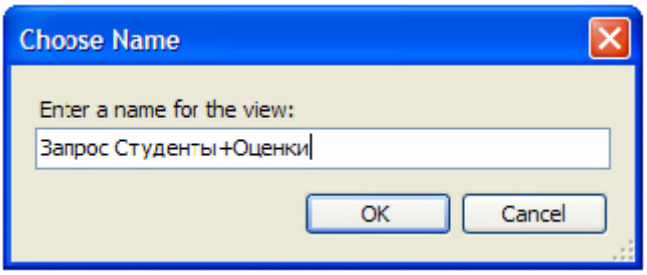
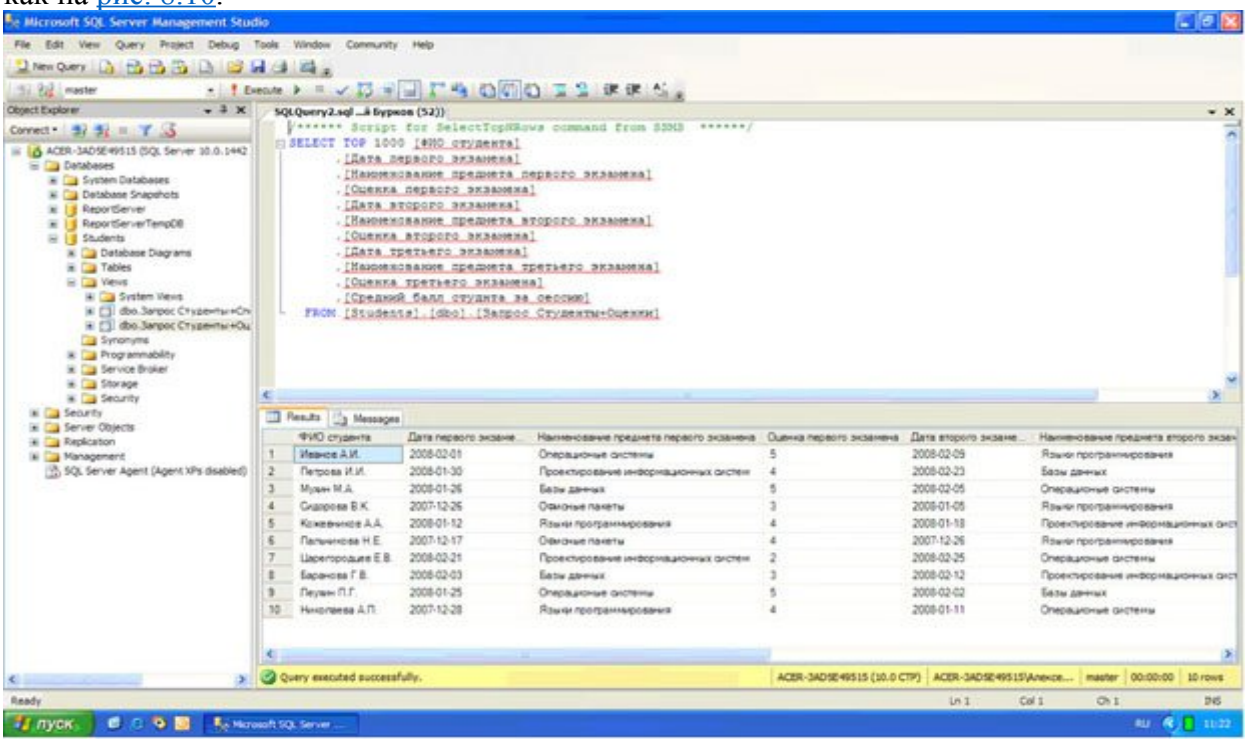


Рис. 8.9.

Проверьте работоспособность нового запроса вне конструктора. Для этого запустите запрос. Результат выполнения запроса **"Запрос Студенты+Оценки"** должен выглядеть как на рис. 8.10.



увеличить изображение

Рис. 8.10.

На этом мы заканчиваем рассмотрение обычных запросов и переходим к созданию фильтров.

На основе запроса **"Запрос Студенты+Специальности"** создадим фильтры, отображающие студентов отдельных специальностей. Создайте новый запрос. Так как он будет основан на запросе **"Запрос Студенты+Специальности"**, то в окне **"Add Table"** перейдите на вкладку **"Views"** и добавьте в новый запрос **"Запрос Студенты+Специальности"** (рис. 8.11). Затем закройте окно **"Add Table"**.

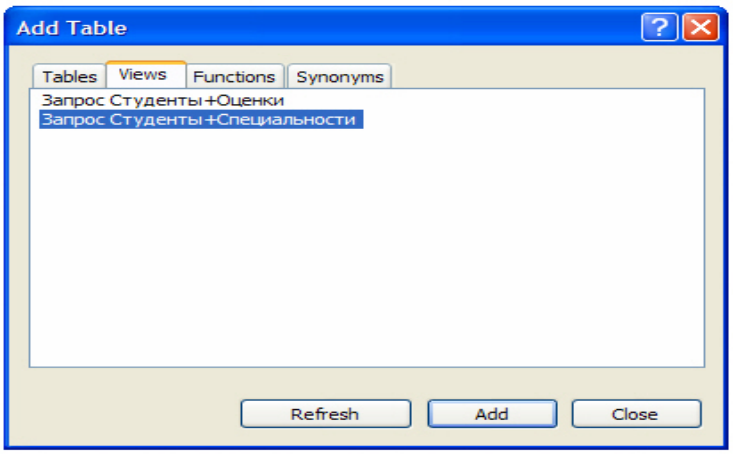
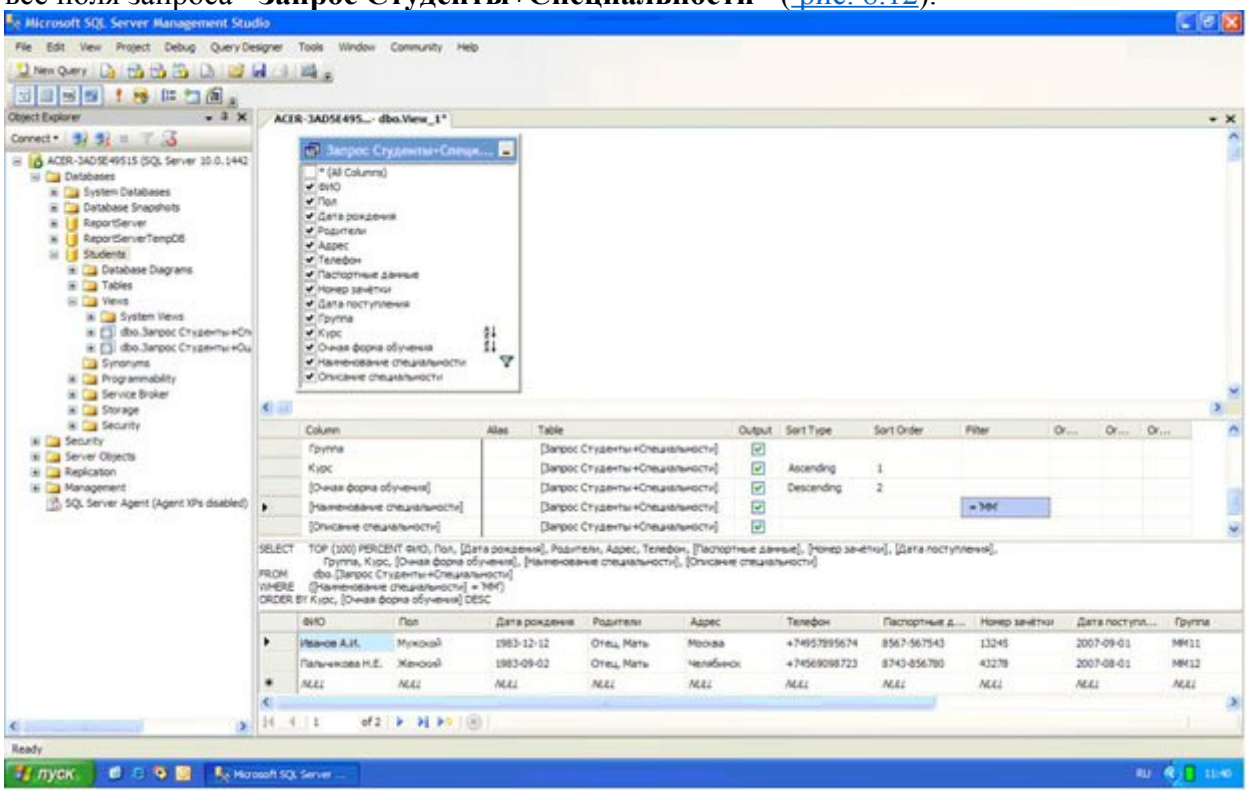


Рис. 8.11.

В появившемся окне конструктора запросов определите в качестве отображаемых полей все поля запроса **"Запрос Студенты+Специальности"** (рис. 8.12).



увеличить изображение

Рис. 8.12.

Замечание: Для отображения всех полей запроса, в данном случае, мы не можем использовать пункт **"* (All Columns)"** (Все поля). Так как в этом случае мы не можем устанавливать критерий отбора записей в фильтре, а также невозможно установить сортировку записей.

Теперь установим критерий отбора записей в фильтре. Пусть наш фильтр отображает только студентов имеющих специальность "ММ". Для определения условия отбора записей в таблице отображаемых полей в строке, соответствующей полю, на которое накладывается условие, в столбце **"Filter"**, необходимо задать условие. В нашем случае условие накладывается на поле **"Наименование специальности"**. Следовательно, в строке **"Наименование специальности"**, в столбце **"Filter"** нужно задать следующее условие отбора **"='ММ'"** (рис. 8.12).

В заключение настроим сортировку записей в фильтре. Пусть при выполнении фильтра сначала происходит сортировка записей по возрастанию по полю **"Очная форма**

обучения", а затем по убыванию по полю **"Курс"**. Для установки сортировки записей по возрастанию, в таблице определяемых полей, в строке для поля **"Очная форма обучения"**, в столбце **"Sort Type"** (Тип сортировки), задайте **"Ascending"** (По возрастанию), а в строке для поля **"Курс"** - задайте **"Descending"** (По убыванию). Для определения порядка сортировки для поля **"Очная форма обучения"** в столбце **"Sort Order"** (Порядок сортировки) поставьте 1, а для поля **"Курс"** поставьте 2 (рис. 8.12). То есть, при выполнении запроса записи сначала сортируются по полю **"Очная форма обучения"**, а затем по полю **"Курс"**.
Замечание: После установки условий отбора и сортировки записей на схеме данных напротив соответствующих полей появятся специальные значки. Значки

и

обозначают сортировку по возрастанию и убыванию, а значок

показывает наличие условия отбора.
После установки сортировки записей в фильтре проверим его работоспособность, выполнив его. Результат выполнения фильтра должен выглядеть как на рис. 8.12. Закройте окно конструктора запросов. В качестве имени нового фильтра в окне **"Choose Name"** задайте **"Фильтр ММ"** (рис. 8.13) и нажмите кнопку **"Ok"**.

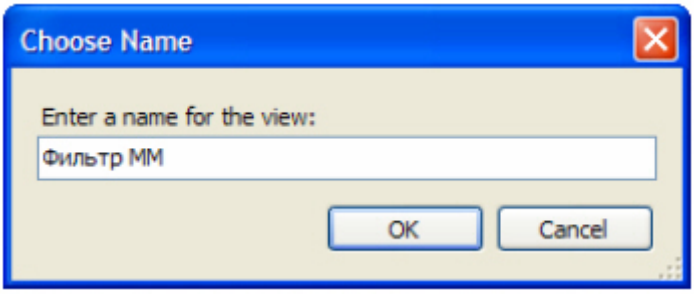
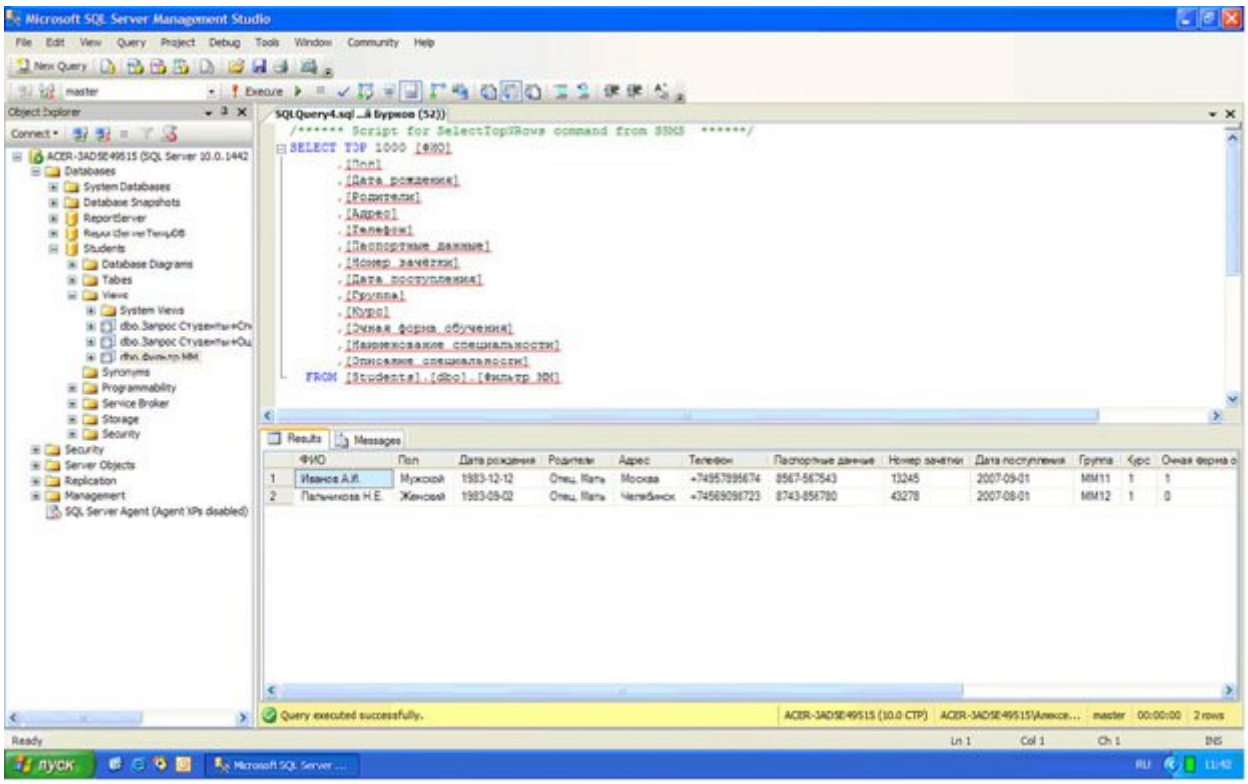


Рис. 8.13.
Фильтр **"Фильтр ММ"** появится в обозревателе объектов. Выполните созданный фильтр вне окна конструктора запросов. Результат должен быть таким же как на рис. 8.14.

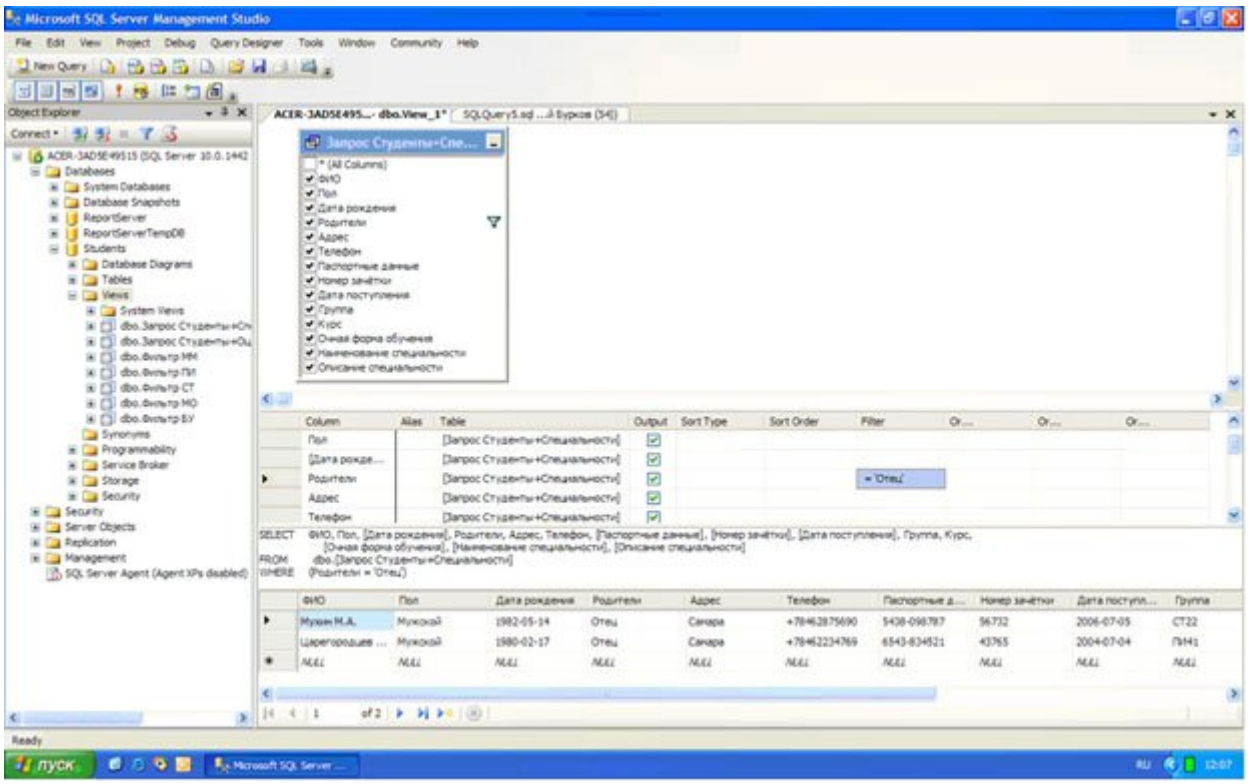


увеличить изображение

Рис. 8.14.

Самостоятельно создайте фильтры для отображения других специальностей. Данные фильтры создаются аналогично фильтру **"Фильтр ММ"** (смотри выше). Единственным отличием является условие отбора, накладываемое на поле **"Наименование специальности"**, оно должно быть не **"='ММ'"**, а **"='ПИ'"**, **"='СТ'"**, **"='МО'"** или **"='БУ'"**. При сохранении фильтров задаем их имена соответственно их условиям отбора, то есть **"Фильтр ПИ"**, **"Фильтр СТ"**, **"Фильтр МО"** или **"Фильтр БУ"**. Проверьте созданные фильтры на работоспособность.

Теперь на основе запроса **"Запрос Студенты+Специальности"** создадим фильтры, отображающие студентов имеющих отдельных родителей. Для начала создадим фильтр для студентов, из родителей только **"Отец"**. Создайте новый запрос и добавьте в него запрос **"Запрос Студенты+Специальности"** (рис. 8.11). После закрытия окна **"Add Table"** сделайте отображаемыми все поля запроса (рис. 8.15).



увеличить изображение

Рис. 8.15.

В таблице отображаемых полей в строке для поля **"Родители"**, в столбце **"Filter"**, задайте условие отбора равное **"='Отец'"**. Проверьте работу фильтра, выполнив его. В результате выполнения фильтра окно конструктора запросов должно выглядеть как на рис. 8.15.

Закройте окно конструктора запросов. В окне **"Choose Name"** задайте имя нового фильтра как **"Фильтр Отец"** (рис. 8.16).

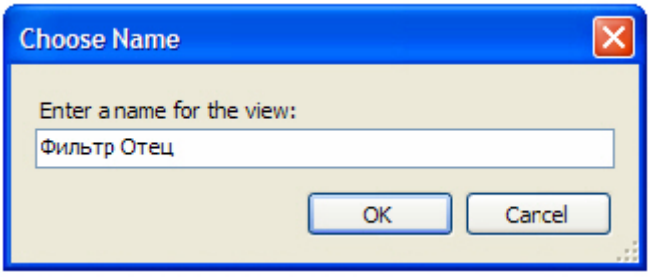
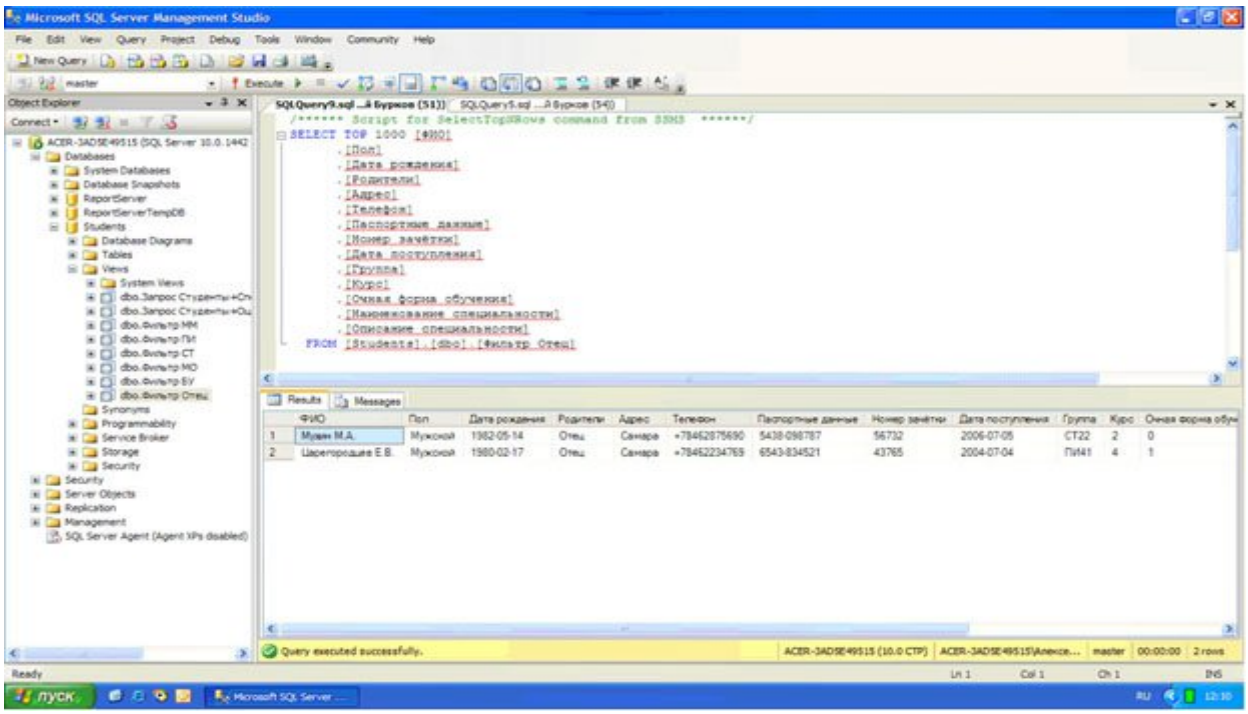


Рис. 8.16.

Выполните фильтр **"Фильтр Отец"** вне конструктора запросов. Результат должен быть аналогичен рис. 8.17.

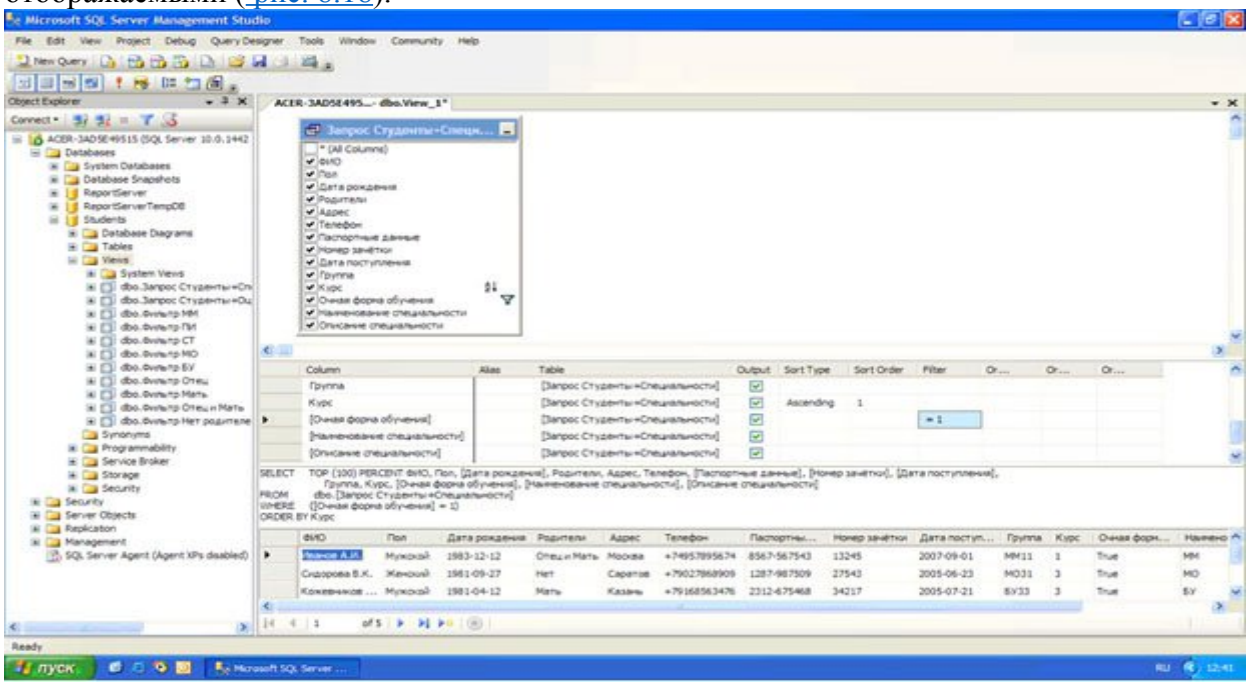


увеличить изображение

Рис. 8.17.

Создайте фильтры для отображения студентов с другими вариантами родителей. Данные фильтры создаются аналогично фильтру **"Фильтр Отец"** (смотри выше). Единственным отличием является условие отбора, накладываемое на поле **"Родители"**, оно должно быть не **"='Отец'"**, а **"='Мать'"**, **"='Отец, Мать'"** или **"='Нет'"**. При сохранении фильтров задаем их имена соответственно их условиям отбора, то есть **"Фильтр Мать"**, **"Фильтр Отец и Мать"** или **"Фильтр Нет родителей"**. Проверьте созданные фильтры на работоспособность.

Наконец создадим фильтры для отображения студентов очной и заочной формы обучения. Начнем с очной формы обучения. Создайте новый запрос и добавьте в него запрос **"Запрос Студенты+Специальности"**. Как и ранее сделайте все поля запроса отображаемыми (рис. 8.18).



увеличить изображение

Рис. 8.18.

В таблице отображаемых полей в столбце **"Filter"**, в строке для поля **"Очная форма обучения"** установите условие отбора равное **"=1"**
Замечание: Поле **"Очная форма обучения"** является логическим полем, оно может принимать значения либо **"True"** (Истина), либо **"False"** (Ложь). В качестве синонимов этих значений в "Microsoft SQL Server 2008" можно использовать 1 и 0 соответственно.
Установите сортировку по возрастанию, по полю курс, задав в строке для этого поля, в столбце **"Sort Type"**, значение **"Ascending"**.
Проверьте работу фильтра, выполнив его. После выполнения фильтра окно конструктора запросов должно выглядеть точно также как на рис. 8.18.
Закройте окно конструктора запросов. Сохраните фильтр под именем **"Фильтр очная форма обучения"** (рис. 8.19).

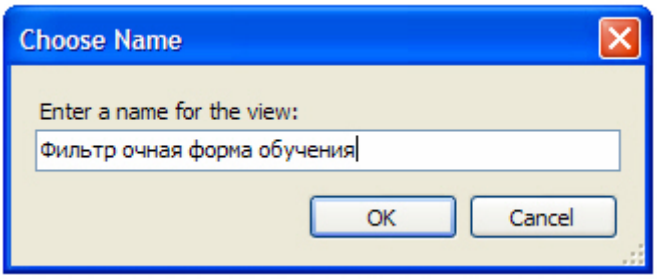
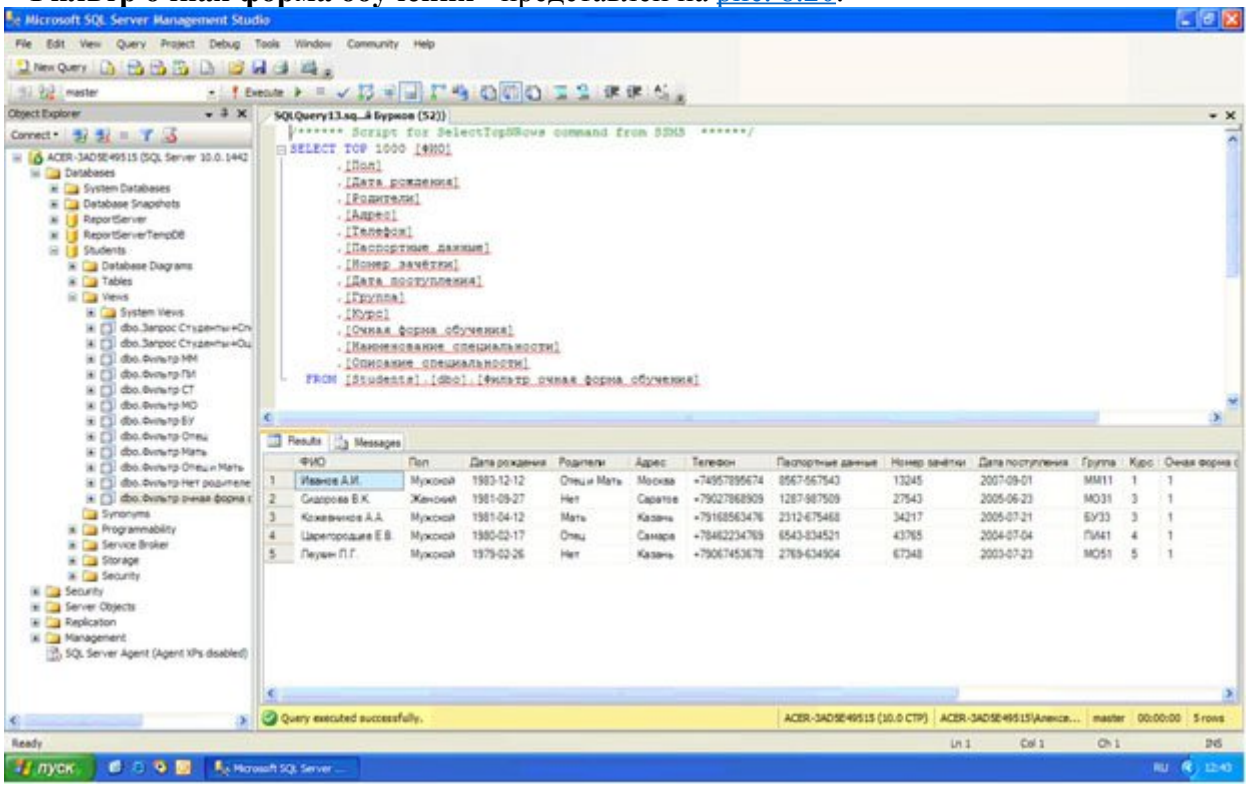


Рис. 8.19.
После появления фильтра **"Фильтр очная форма обучения"** в обозревателе объектов выполните фильтр вне окна конструктора запросов. Результат выполнения фильтра **"Фильтр очная форма обучения"** представлен на рис. 8.20.



увеличить изображение
Рис. 8.20.
Самостоятельно создайте фильтр для отображения студентов заочной формы обучения. Данный фильтр создается точно также как и фильтр **"Фильтр очная форма обучения"**. Единственным отличием является условие отбора, накладываемое на поле **"Очная форма обучения"**, оно должно быть не "=1", а "=0". При сохранении фильтра задайте его имя как "Фильтр заочная форма обучения". Проверьте созданный фильтр на работоспособность.

В итоге, после создания всех запросов и фильтров окно обозревателя объектов должно выглядеть следующим образом (рис. 8.21):

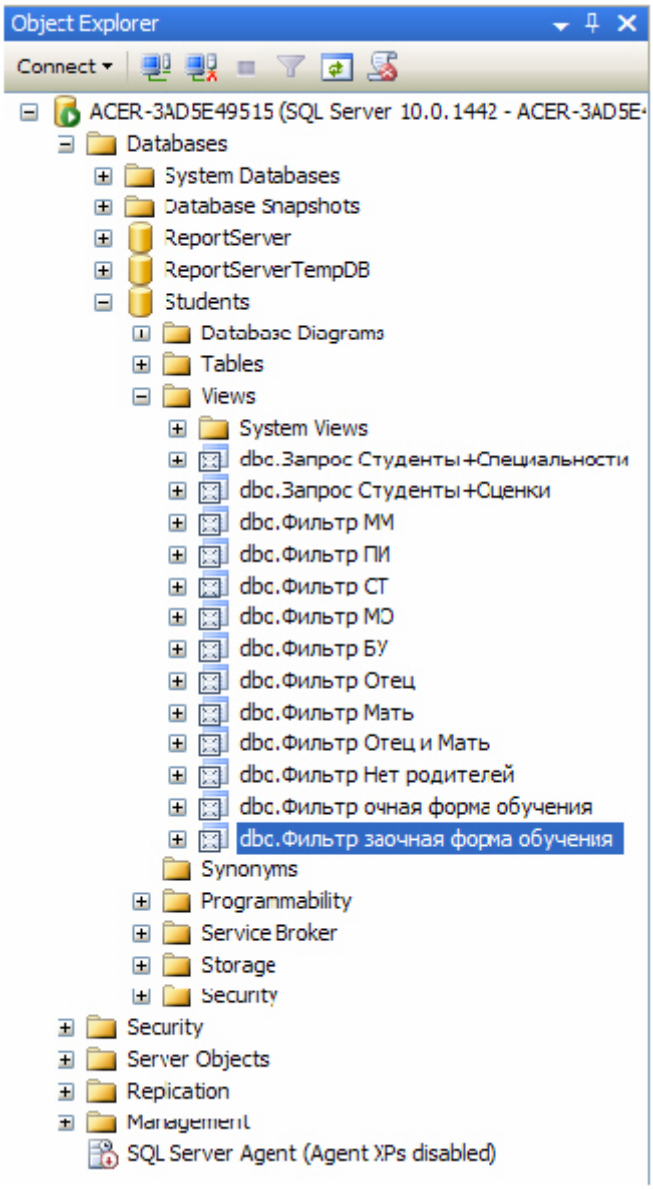


Рис. 8.21.

Контрольные вопросы

1. Как создать новый запрос?
2. Как создать новый фильтр?
3. Как использовать фильтр?

Лабораторная работа № 7

«Создание динамических запросов при помощи хранимых процедур в Microsoft SQL Server 2012»

Цель работы:

- научиться работать с хранимыми процедурами.

Перейдем к созданию хранимых процедур. Для работы с хранимыми процедурами в обозревателе объектов необходимо выделить папку **"Programmability/Stored Procedures"** базы данных **"Students"** (рис. 10.1).

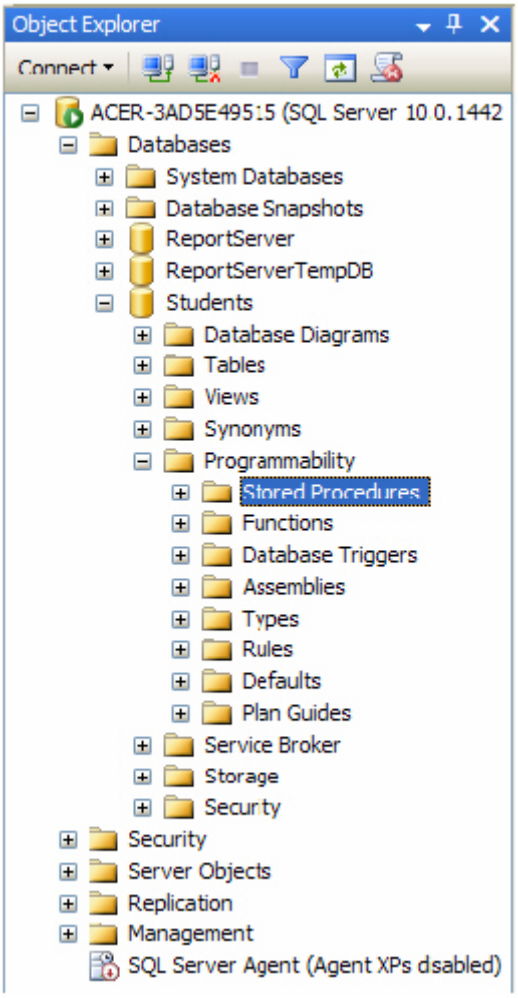
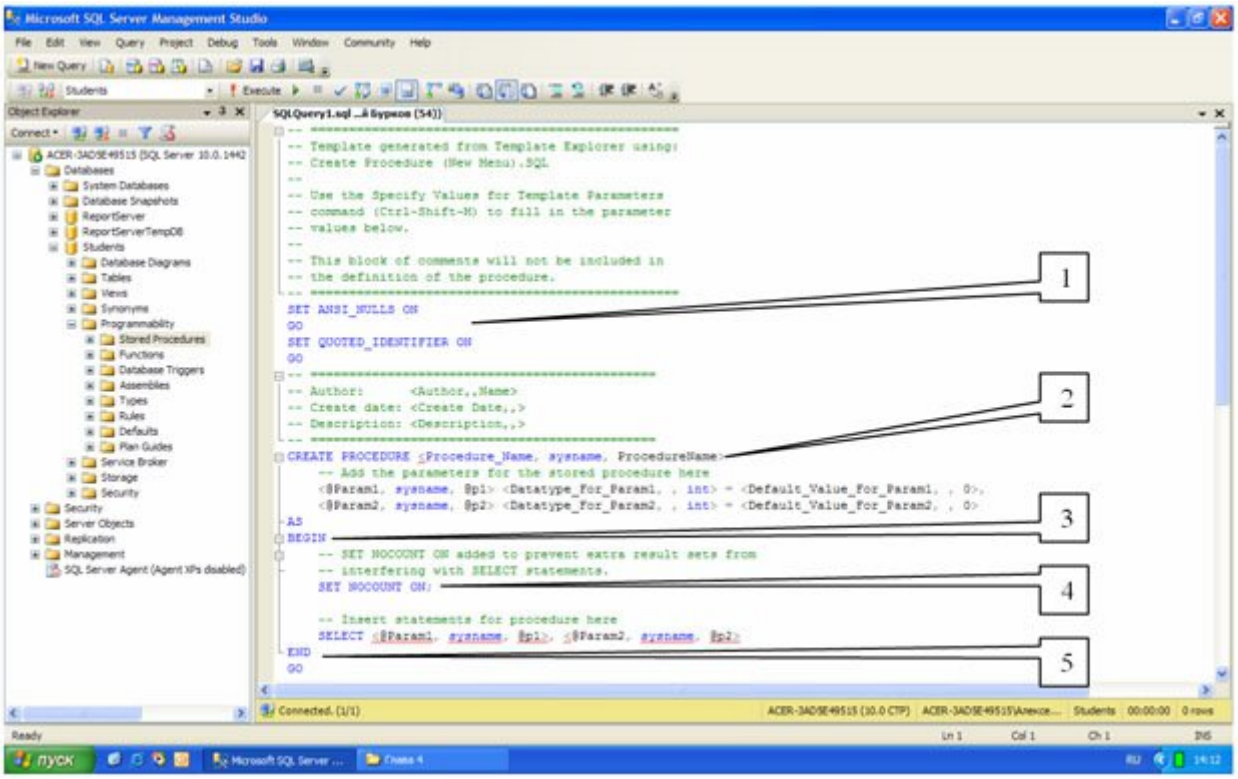


Рис. 10.1.

Создадим процедуру, вычисляющую среднее трех чисел. Для создания новой хранимой процедуры щелкните **ПКМ** по папке **"Stored Procedures"** (рис. 10.1) и в появившемся меню выберите пункт **"New Stored Procedure"**. Появится окно кода новой хранимой процедуры (рис. 10.2).



увеличить изображение

Рис. 10.2.

Хранимая процедура имеет следующую структуру (рис. 10.2):

1. Область настройки параметров синтаксиса процедуры. Позволяет настраивать некоторые синтаксические правила, используемые при наборе кода процедуры. В нашем случае это:
   - SET ANSI_NULLS ON - включает использование значений NULL (Пусто) в кодировке ANSI,
   - SET QUOTED_IDENTIFIER ON - включает возможность использования двойных кавычек для определения идентификаторов;
2. Область определения имени процедуры (**Procedure_Name**) и параметров передаваемых в процедуру (**@Param1, @Param2**). Определение параметров имеет следующий синтаксис:
   @<Имя параметра> <Тип данных> = <Значение по умолчанию>
   Параметры разделяются между собой запятыми;
3. Начало тела процедуры, обозначается служебным словом "BEGIN" ;
4. Тело процедуры, содержит команды языка программирования запросов T-SQL;
5. Конец тела процедуры, обозначается служебным словом "END".

Замечание: В коде зеленым цветом выделяются комментарии. Они не обрабатываются сервером и выполняют функцию пояснений к коду. Строки комментариев начинаются с подстроки **"--"**. Далее в коде, мы не будем отображать комментарии, они будут свернуты. Слева от раздела с комментариями будет стоять знак **"+"**, щелкнув по которому можно развернуть комментарий.

Наберем код процедуры вычисляющей среднее трех чисел, как это показано на рис. 10.3.

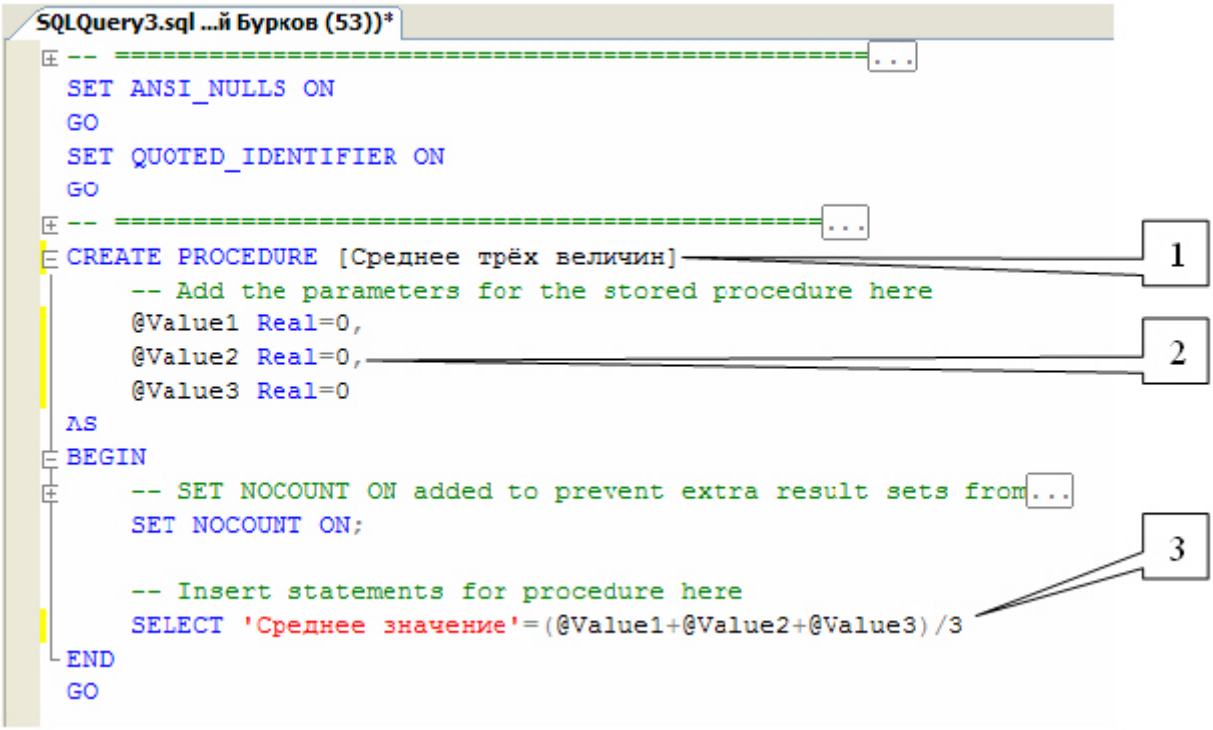


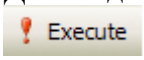
Рис. 10.3.

Рассмотрим код данной процедуры более подробно (рис. 10.3):

1. CREATE PROCEDURE [Среднее трех величин] - определяет имя создаваемой процедуры как "Среднее трех величин";
2. @Value1 Real = 0, @Value2 Real = 0, @Value3 Real = 0 - определяют три параметра процедуры Value1, Value2 и Value3. Данным параметрам можно присвоить дробные числа (Тип данных Real), значения по умолчанию равны 0;
3. SELECT 'Среднее значение'=(@Value1+@Value2+@Value3)/3 - вычисляет среднее и выводит результат с подписью "Среднее значение".

Остальные фрагменты кода рассмотрены выше (рис. 10.2).

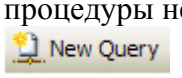
Для создания процедуры, выполним вышеописанный код, нажав кнопку



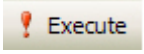
(Выполнить) на панели инструментов. В нижней части окна с кодом появиться сообщение **"Command(s) completed successfully."**. Закройте окно с кодом, щелкнув мышью по кнопке закрытия

расположенной в верхнем правом углу окна с кодом процедуры.

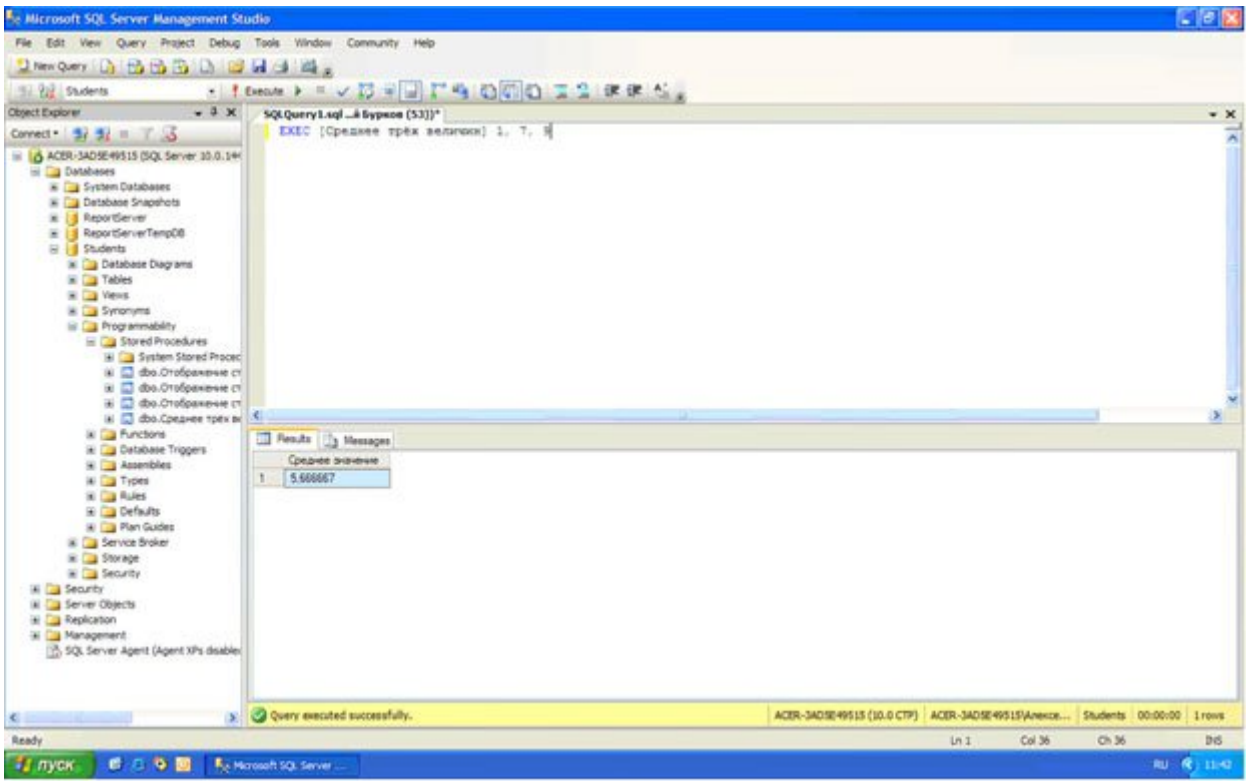
Проверим работоспособность созданной хранимой процедуры. Для запуска хранимой процедуры необходимо создать новый пустой запрос, нажав на кнопку



(Новый запрос) на панели инструментов. В появившемся окне с пустым запросом наберите команду EXEC [Среднее трех величин] 1, 7, 9 и нажмите кнопку



на панели инструментов (рис. 10.4).

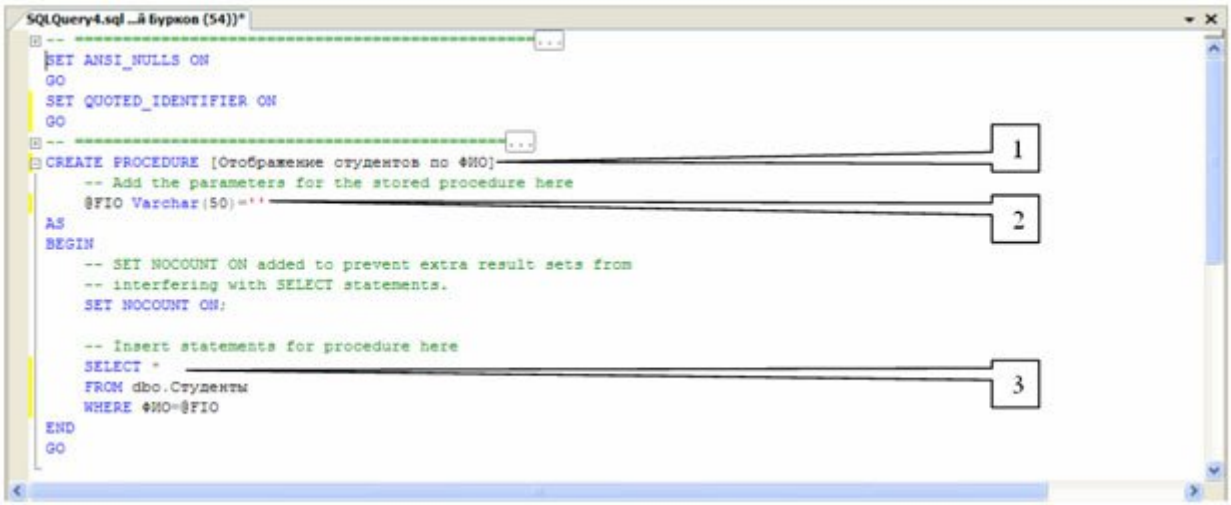


увеличить изображение

Рис. 10.4.

В нижней части окна с кодом появится результат выполнения новой хранимой процедуры: **Среднее значение 5,66667** (рис. 10.4).

Теперь создадим хранимую процедуру для отбора студентов из таблицы студенты по их "ФИО". Для этого создайте новую хранимую процедуру, как это описано выше, и наберите код новой процедуры как на рис. 10.5.



увеличить изображение

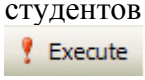
Рис. 10.5.

Рассмотрим код процедуры **"Отображение студентов по ФИО"** более подробно (рис. 10.5):

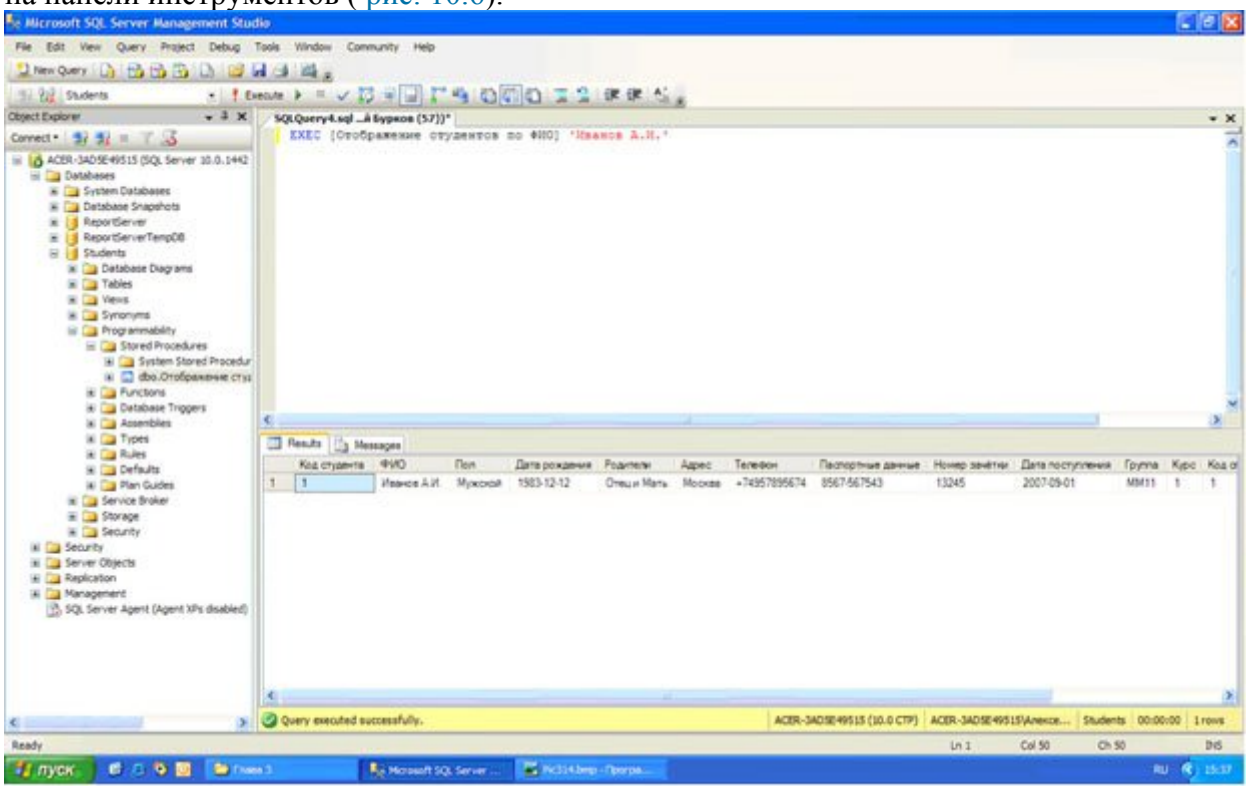
1. CREATE PROCEDURE [Отображение студентов по ФИО] - определяет имя создаваемой процедуры как "Отображение студентов по ФИО";
2. @FIO Varchar(50)='' - определяют единственный параметр процедуры **FIO**. Параметру можно присвоить текстовые строки переменной длины, длинной до 50 символов (Тип данных Varchar(50)), значения по умолчанию равны пустой строке;

3. SELECT * FROM dbo.Студенты WHERE ФИО=@FIO - отобразить все поля (*) из таблицы студенты (dbo.Студенты), где значение поля ФИО равно значению параметра **FIO (ФИО=@FIO)**.

Выполним вышеописанный код и закроем окно с кодом, как описано выше.
Проверим работоспособность созданной хранимой процедуры. Создайте новый пустой запрос. В появившемся окне с пустым запросом наберите команду EXEC [Отображение студентов по ФИО] 'Иванов А.И.' и нажмите кнопку



на панели инструментов (рис. 10.6).

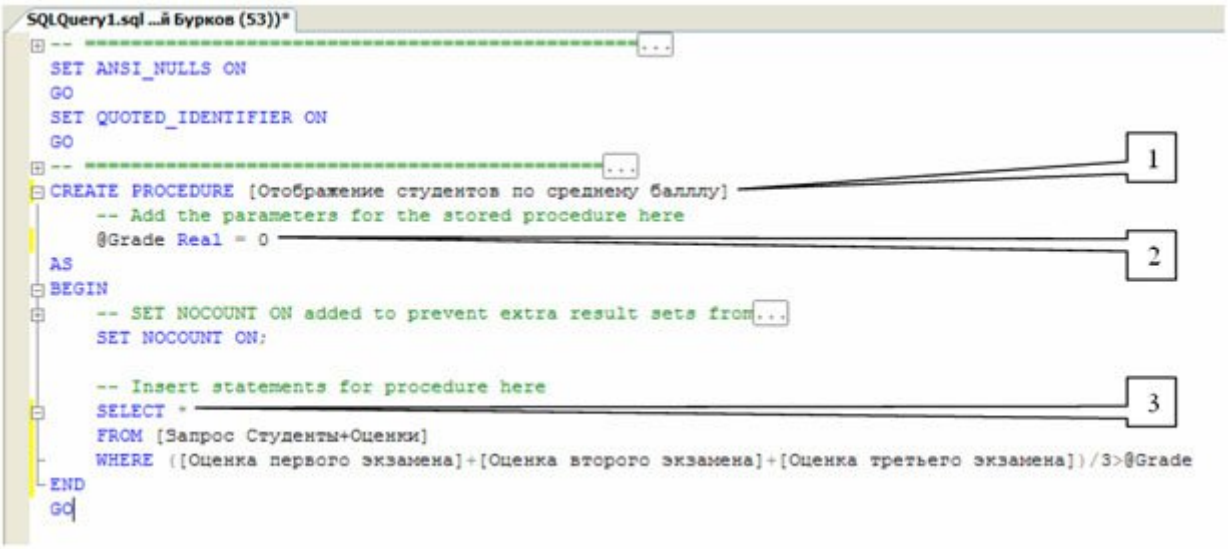


увеличить изображение

Рис. 10.6.

В нижней части окна с кодом появиться результат выполнения хранимой процедуры **"Отображение студентов по ФИО"** (рис. 10.6).

Теперь перейдем к более сложной задаче - отобразить студентов, у которых средний балл выше заданного. Создайте новую хранимую процедуру и наберите код новой процедуры как на рис. 10.7.



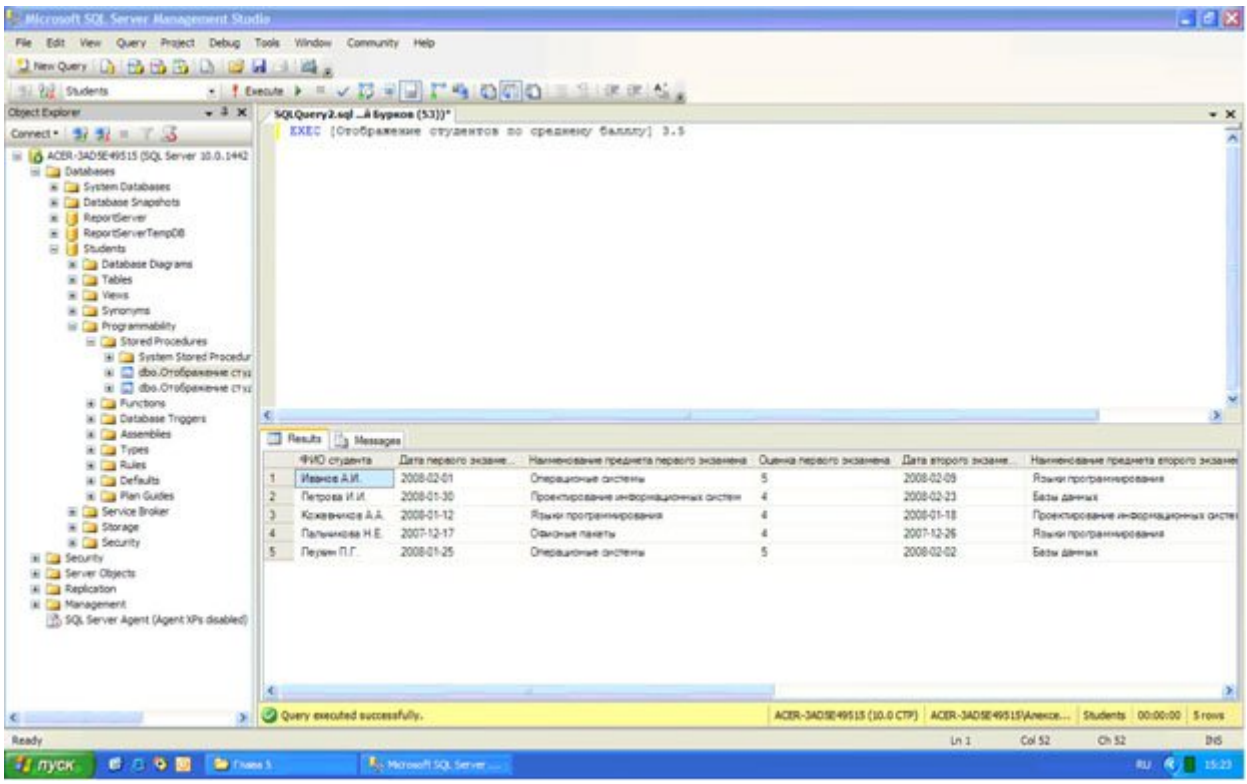
увеличить изображение

Рис. 10.7.

Рассмотрим код процедуры **"Отображение студентов по среднему баллу"** более подробно (рис. 10.7):

1. CREATE PROCEDURE [Отображение студентов по среднему баллу] - определяет имя создаваемой процедуры как "Отображение студентов по среднему баллу";
2. @Grade Real=0 - определяют параметр процедуры **Grade**. Параметру можно присвоить дробные числа (Тип данных Real), значения по умолчанию равны 0;
3. SELECT * FROM [Запрос Студенты+Оценки] WHERE ([Оценка первого экзамена]+[Оценка второго экзамена]+[Оценка третьего экзамена])/3>@Grade - отобразить все поля (*) из запроса **"Запрос Студенты+Оценки"** (Запрос Студенты+Оценки), где средний балл больше чем значение параметра **Grade (([Оценка первого экзамена]+[Оценка второго экзамена]+[Оценка третьего экзамена])/3>@Grade)**.

Выполним вышеописанный код и закроем окно с кодом, как описано выше. Проверим, как работает запрос, описанный выше. Для этого, создайте новый запрос и в нем наберите команду EXEC [Отображение студентов по среднему баллу] 3.5 и выполните ее (Смотри выше) (рис. 10.8).



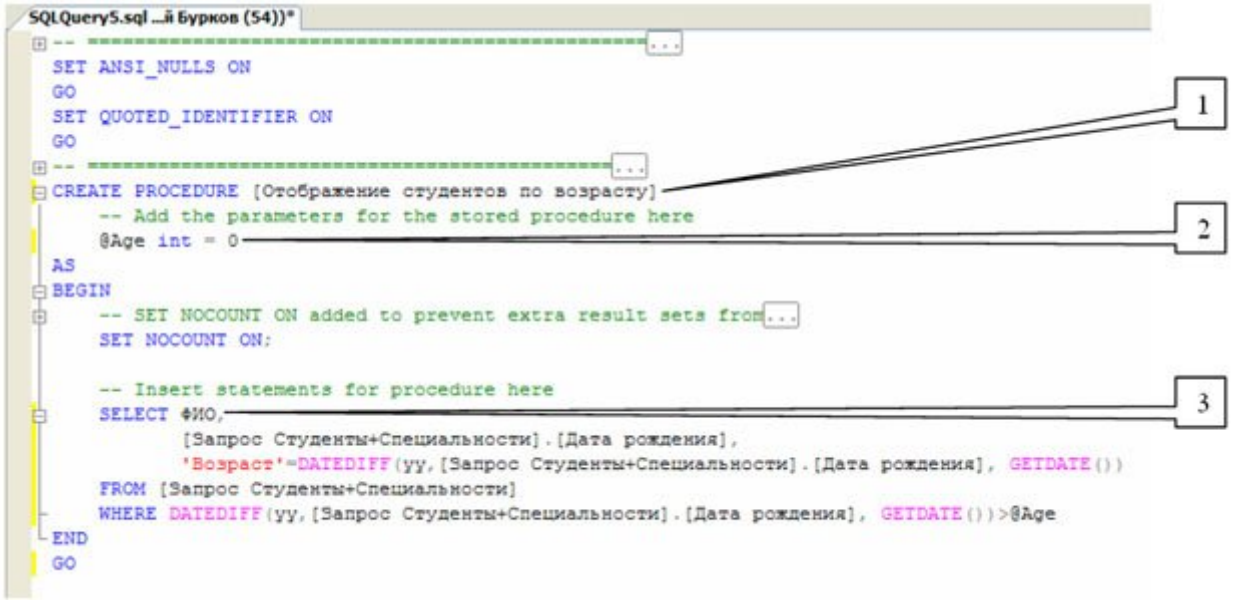
увеличить изображение

Рис. 10.8.

В нижней части окна с кодом появится результат выполнения хранимой процедуры **"Отображение студентов по среднему баллу"** (рис. 10.8).

В заключение решим более сложную задачу - отображение студентов старше заданного возраста. При чем возраст будет автоматически вычисляться в зависимости от даты рождения.

Создадим новую хранимую процедуру и наберем код новой процедуры как представлено на рис. 10.9.



увеличить изображение

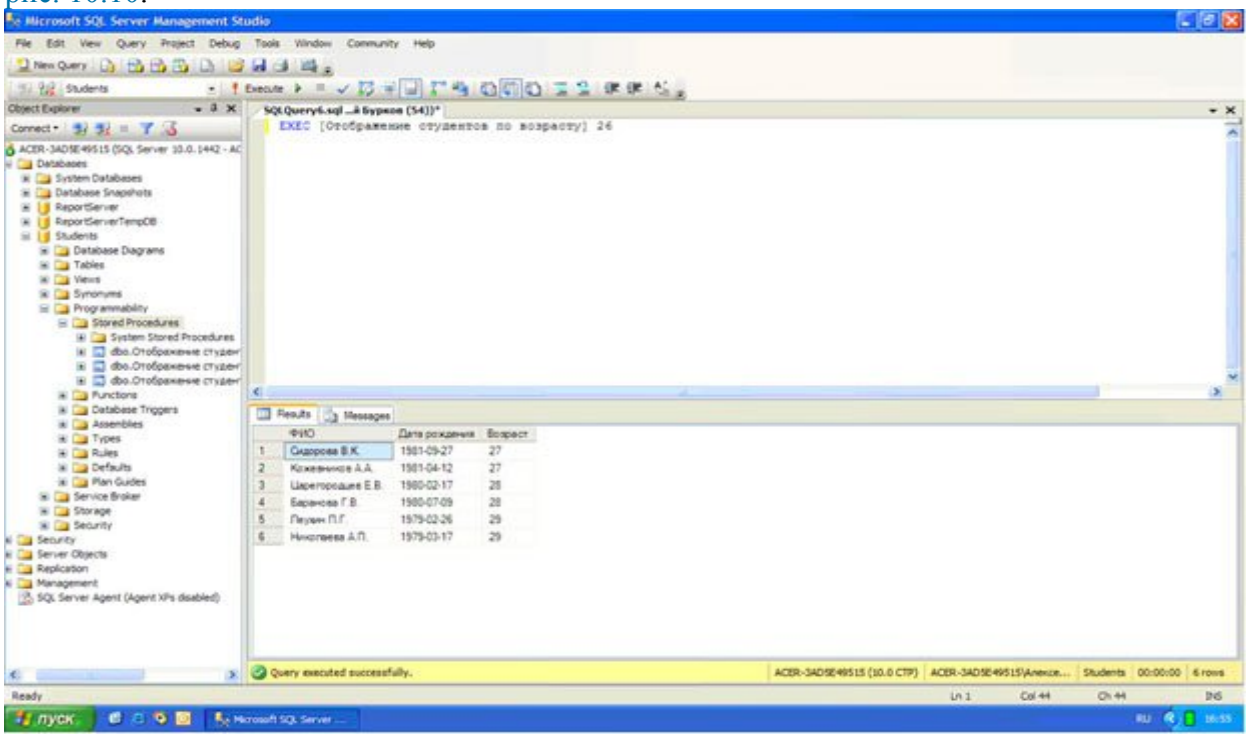
Рис. 10.9.

Рассмотрим код создаваемой процедуры **"Отображение студентов по возрасту"** более подробно (рис. 10.9):

1. CREATE PROCEDURE [Отображение студентов по возрасту] - определяет имя создаваемой процедуры как "Отображение студентов по возрасту";
2. @Age int=0 - определяют параметр процедуры **Age**. Параметру можно присвоить целые числа (Тип данных int), значения по умолчанию равны 0;
3. ФИО, [Запрос Студенты+Специальности].[Дата рождения], 'Возраст'=DATEDIFF(yy,[Запрос Студенты+Специальности].[Дата рождения], GETDATE()) - отображает из запроса **"Запроса Студенты+Специальности"** (FROM [Запрос Студенты+Специальности]) поля **"ФИО"** (ФИО) и **"Дата рождения"** ([Запрос Студенты+Специальности].[Дата рождения]), а также отображает возраст студента ('Возраст') в годах (yy), вычисленный исходя из его даты рождения и текущей даты (DATEDIFF(yy,[Запрос Студенты+Специальности].[Дата рождения], GETDATE())). Более того, выводятся студенты возраст которых больше определенного в параметре **"Age"** (DATEDIFF(yy,[Запрос Студенты+Специальности].[Дата рождения], GETDATE())>@Age).

Замечание: Встроенная функция **DATEDIFF** вычисляющая количество периодов между двумя датами, имеет следующий синтаксис: DATEDIFF(<период>,<начальная дата>, <конечная дата>)

Выполним код запроса **"Отображение студентов по возрасту"**, а затем закроем окно с кодом, как описано выше. Проверим, как работает запрос. Для этого, создадим новый запрос и в нем наберем команду EXEC [Отображение студентов по возрасту] 26 и выполните ее. Должен появиться результат аналогичный результату, представленному на рис. 10.10.



увеличить изображение

Рис. 10.10.

На этом мы заканчиваем описание хранимых процедур и переходим к рассмотрению пользовательских функций. В итоге, обозреватель объектов должен иметь вид как на рис. 10.11.

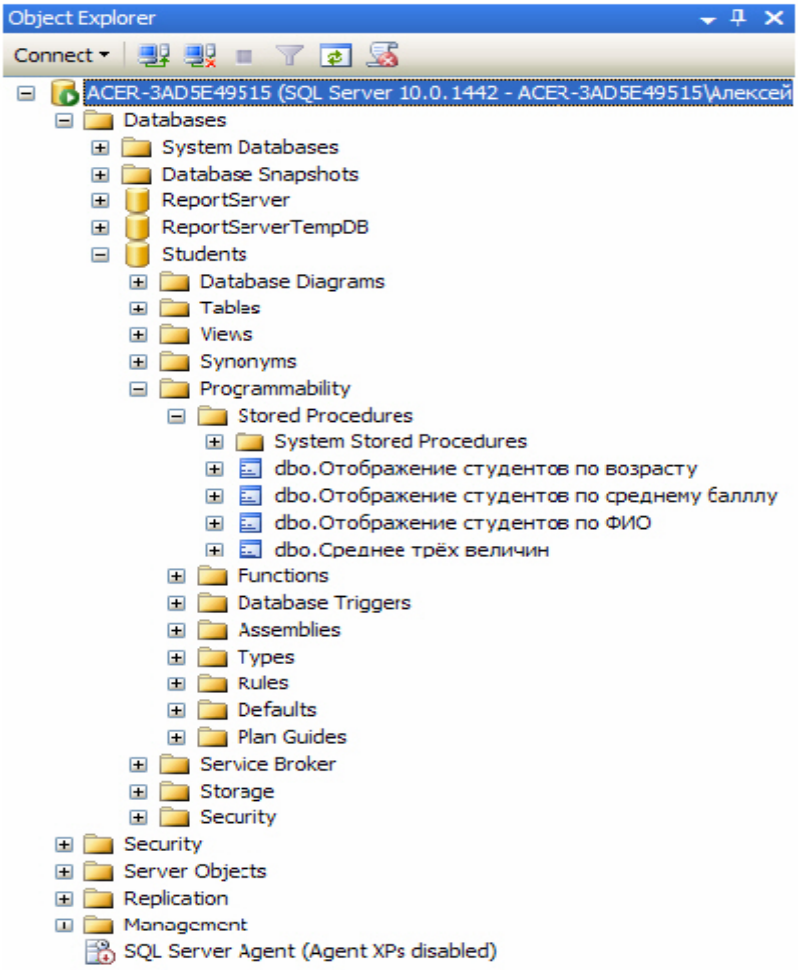


Рис. 10.11.

Контрольные вопросы

1. Что такое хранимая процедура?
2. Какова структура хранимой процедуры?

Лабораторная работа № 8
«Создание функций пользователя в Microsoft SQL Server 2012.»

Цель работы:

Научиться работать с пользовательскими функциями.

Теперь рассмотрим создание и применение пользовательских функций. В БД "Microsoft SQL Server 2008" все пользовательские функции находятся в папке **"Functions"** расположенной в папке **"Programmability"** в обозревателе объектов (рис. 12.1).

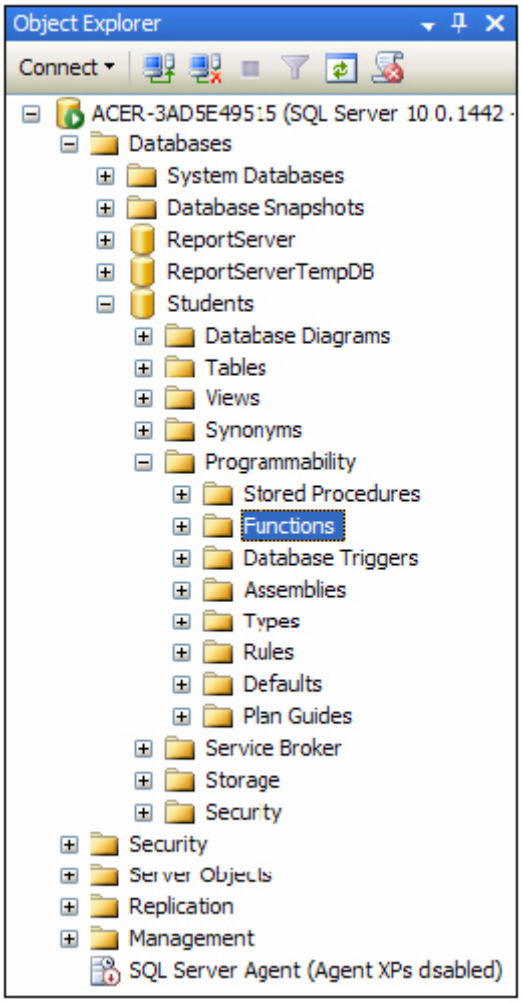
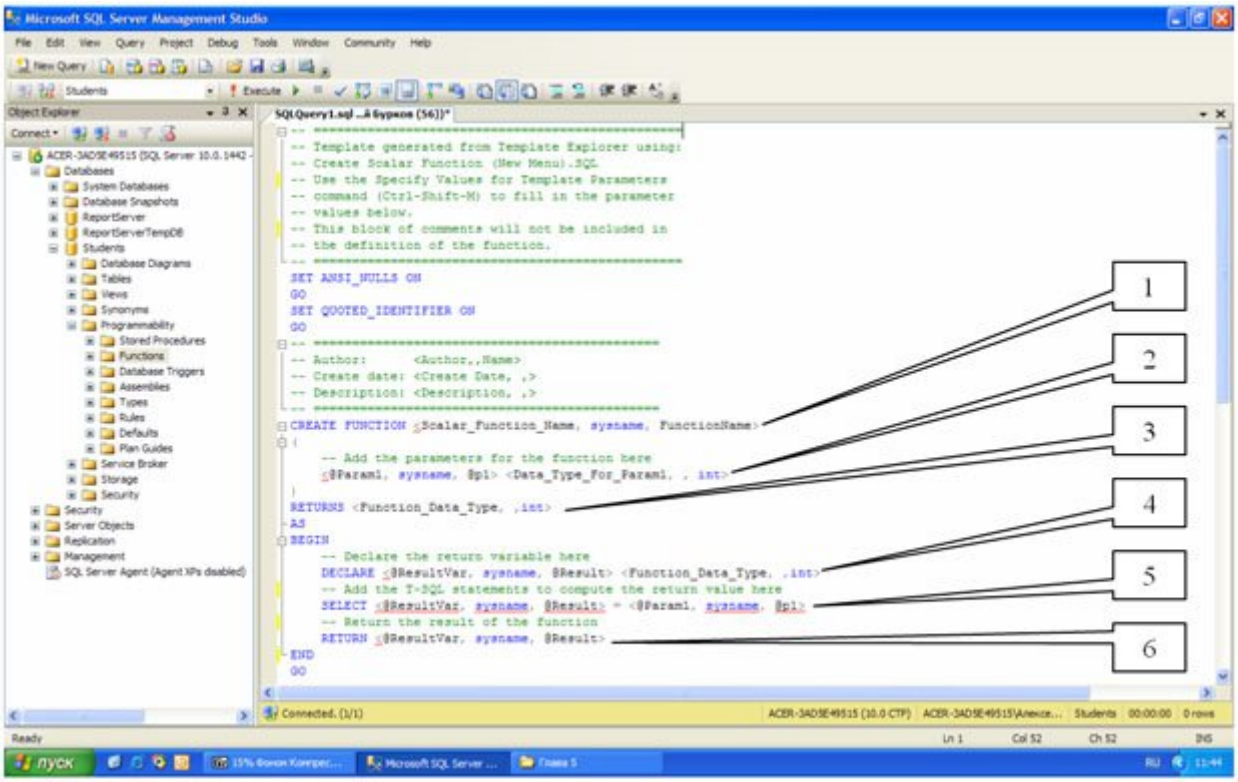


Рис. 12.1.

Начнем с создания скалярных пользовательских функций. Для создания новой скалярной пользовательской функции в обозревателе объектов, в БД **"Students"**, в папке **"Programmability"**, щелкните **ПКМ** по папке **"Functions"** и в появившемся меню выберите пункт **"New/Scalar-valued Function"**. Появится окно новой скалярной пользовательской функции (рис. 12.2)



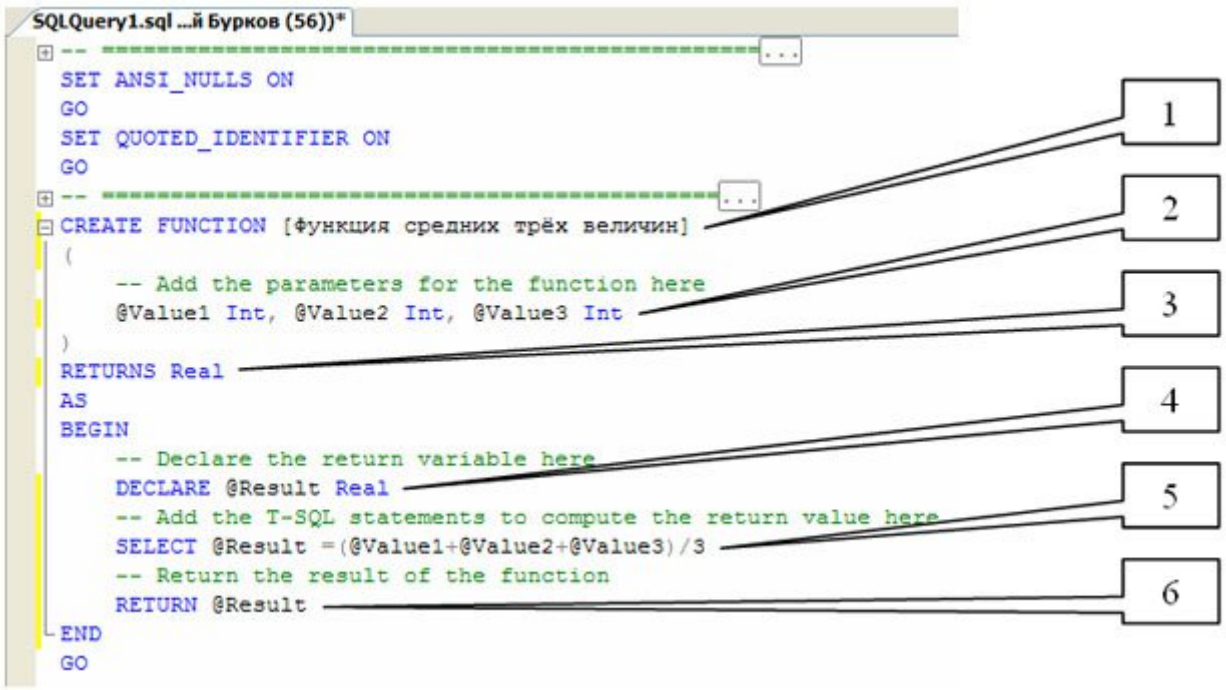
увеличить изображение

Рис. 12.2.

Синтаксис скалярной пользовательской функции похож на синтаксис хранимой процедуры (см. "Интерфейс информационных систем. Создание интерфейса пользователя"). Однако имеется ряд существенных отличий (рис. 12.2):

1. Область определения имени функции (Inline_Function_Name);
2. Параметры, передаваемые в процедуру **(@Param1)**. Определение параметров аналогично определению параметров в хранимой процедуре (см. "Таблицы. Типы данных и свойства полей. Создание и заполнение таблиц");
3. Тип данных значения возвращаемого процедурой;
4. Область объявления переменных, используемых внутри функции. Объявление переменных имеет следующий синтаксис:
   DECLARE @<Имя переменной> <Тип данных>
5. Тело самой пользовательской функции, содержит команды языка программирования запросов T-SQL;
6. Команда RETURN возвращающая результат выполнения функции. Имеет следующий синтаксис:
   RETURN @<Имя переменной с результатом>
   Переменная должна быть того же типа данных, который был указан в пункте 3.

Создадим скалярную пользовательскую функцию, вычисляющую среднее трех величин. В окне новой пользовательской функции наберите код представленный на рис. 12.3.



увеличить изображение

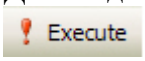
Рис. 12.3.

Рассмотрим более подробно код данной скалярной пользовательской функции (рис. 12.3):

1. CREATE FUNCTION [Функция средних трех величин] - определяет имя создаваемой функции как "Функция средних трех величин";
2. @Value1 Real, @Value2, @Value3 - определяют три параметра процедуры **Value1**, **Value2** и **Value3**. Данным параметрам можно присвоить целые числа (Тип данных Int);
3. RETURNS Real - показывает, что функция возвращает дробные числа (Тип данных Real);
4. DECLARE @Result Real - объявляется переменная @Result для хранения результата работы функции, то есть дробного числа (Тип данных Real);
5. SELECT @Result=(@Value1+@Value2+@Value3)/3 - вычисляет среднее и помещает результат в переменную @Result ;
6. RETURN @Result - возвращает значение переменной @Result.

Остальные фрагменты кода рассмотрены выше (рис. 12.2).

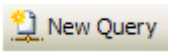
Для создания функции, выполним вышеописанный код, нажав кнопку



(Выполнить) на панели инструментов. В нижней части окна с кодом появиться сообщение **"Command(s) completed successfully."**. Закройте окно с кодом, щелкнув мышью по кнопке закрытия

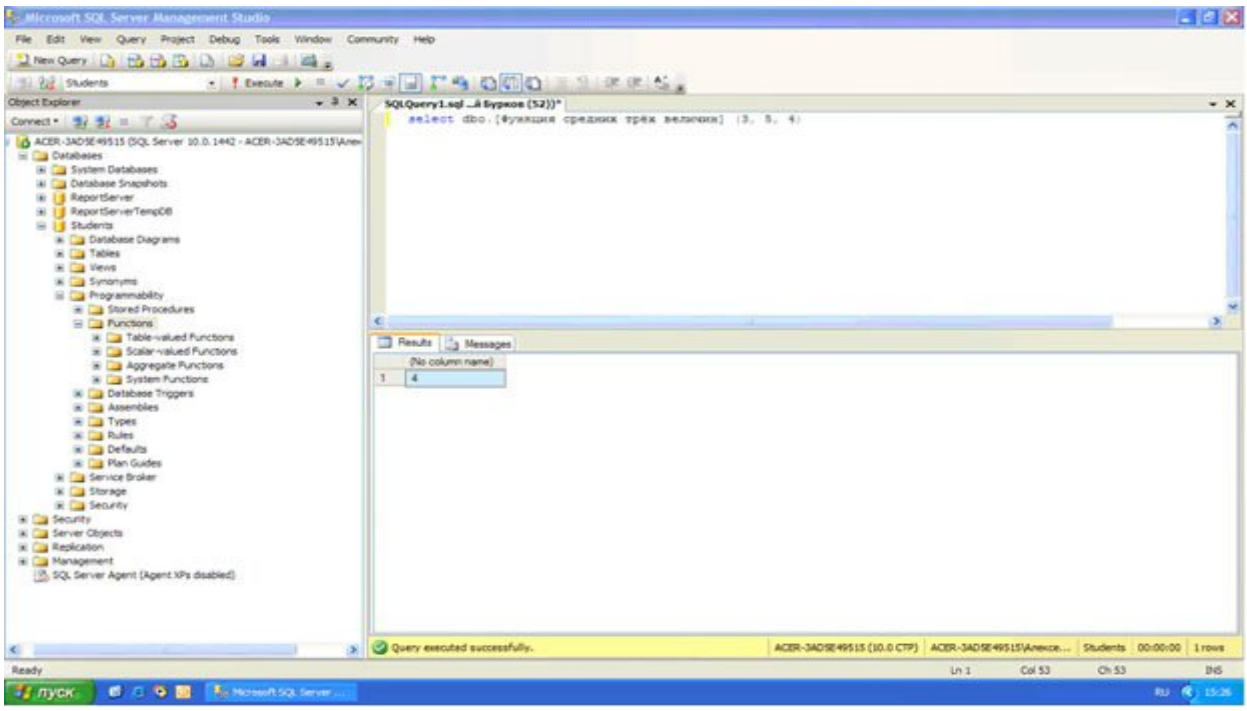
расположенной в верхнем правом углу окна с кодом функции.

Проверим работу созданной скалярной пользовательской функции. Для запуска пользовательской функции необходимо создать новый пустой запрос, нажав на кнопку



(Новый запрос) на панели инструментов. В появившемся окне с пустым запросом наберите команду SELECT dbo.[Функция средних трех величин] (3, 5, 4) и нажмите кнопку

на панели инструментов (рис. 12.4).



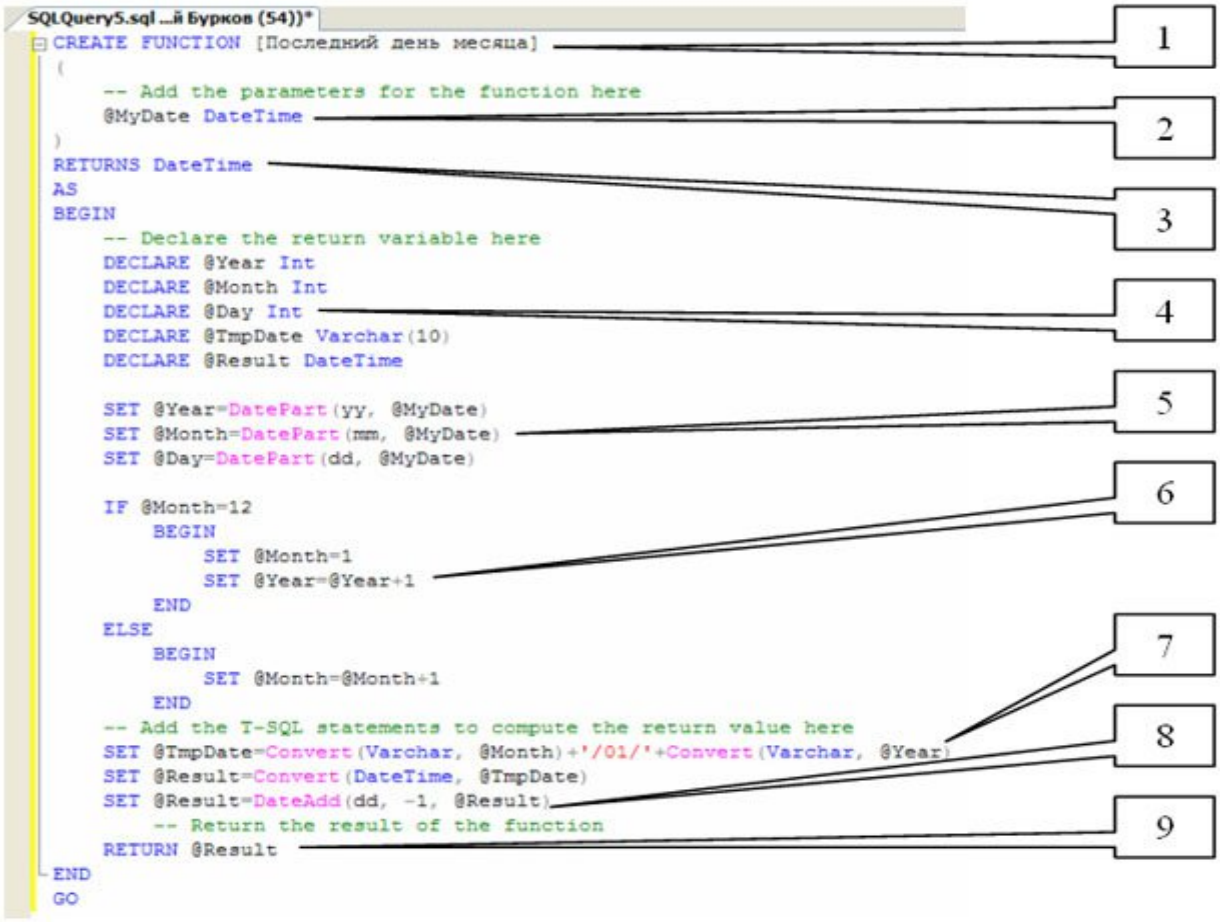
увеличить изображение

Рис. 12.4.

В нижней части окна с кодом появится результат выполнения новой скалярной пользовательской функции: **4** (рис. 12.4).

Теперь создадим более сложную скалярную пользовательскую функцию, предназначенную для определения последнего дня месяца введенной даты.

Создайте новую скалярную пользовательскую функцию, так как об этом сказано выше. В окне новой пользовательской функции наберите следующий код (рис. 12.5):



увеличить изображение

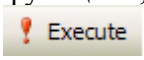
Рис. 12.5.

Перейдем к рассмотрению вышеприведенного кода (рис. 12.5). Код состоит из следующих групп команд:

1. CREATE FUNCTION [Последний день месяца] - определяет имя создаваемой функции как "Последний день месяца";
2. @MyDate - определяют параметр процедуры **MyDate**. Параметру можно присвоить значения дат или времени (Тип данных DateTime);
3. RETURNS DateTime - показывает, что функция возвращает дату или время (Тип данных DateTime);
4. DECLARE @Year Int, DECLARE @Month Int, DECLARE @Day Int - объявляются переменные @Year, @Month и @Day для хранения целочисленных значений года, месяца и дня введенной даты (Тип данных Int).
   DECLARE @TmpDate VarChar(10) объявляет переменную "TmpDate" для хранения промежуточного значения даты в строке длинной до 10 символов (Тип данных VarChar(10)).
   DECLARE @Result DateTime объявляет переменную "Result" для хранения результата - даты последнего дня месяца (Тип данных DateTime).
5. SET @Year=DatePart(yy, @MyDate), SET @Month=DatePart(mm, @MyDate), SET @Day=DatePart(dd, @MyDate) - определяются части введенной даты и помещаются в переменныне @Year, @Month и @Day. Для определения частей даты используется функция **DatePart**, имеющая следующий синтаксис: DatePart(<часть даты>, <дата>). Здесь **"часть даты"** - это закодированная специальными символами определяемая часть даты (yy - год, mm - месяц, dd - день), **"дата"** - это дата, части которой определяем.
6. IF @Month=12
7. BEGIN
8. SET @Month=1
9. SET @Year=@Year+1
10. END
11. ELSE
12. BEGIN
13. SET @Month=@Month+1
14. END
    Вышепреведенный фрагмент кода выполняет следующие действия: Если номер месяца равен 12 то установить номер месяца (@Month) равным 1 и увеличить год (@Year) на 1, иначе увеличить месяц на 1.
15. SET @TmpDate=Convert(Varchar, @Month)+'/01/'+Convert(Varchar, @Year), SET @Result=Convert(DateTime, @TmpDate) - переводит числовые значения даты в дату в строковом формате и записывает ее в переменную @TmpDate, затем переводит дату в строковом формате в тип данных даты и времени и помещает ее в переменную @Result. Для конвертации используется функция **Convert**, имеющая следующий синтаксис:
    Convert(<тип данных>, <значение>), здесь "тип данных" это тип данных в который переводится "значение".
16. SET @Result=DateAdd(dd, -1, @Result) - из даты, хранимой в перменной @Result вычитается 1 день, для этого используется функция **DateAdd**, имеющая следующий синтаксис:
    DateAdd(<часть даты>, <количество периодов>, <дата>) - здесь "часть даты" - это закодированная специальными символами определяемая часть даты (см. функцию **DatePart**), "количество периодов" - это количество частей даты прибавляемой к введенной дате (параметр **"дата"**).

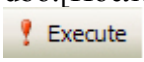
17. RETURN @Result - возвращает значение, хранимое в переменной @Result.

Для создания функции, выполним вышеописанный код, как и в случае с предыдущей функцией, нажав кнопку

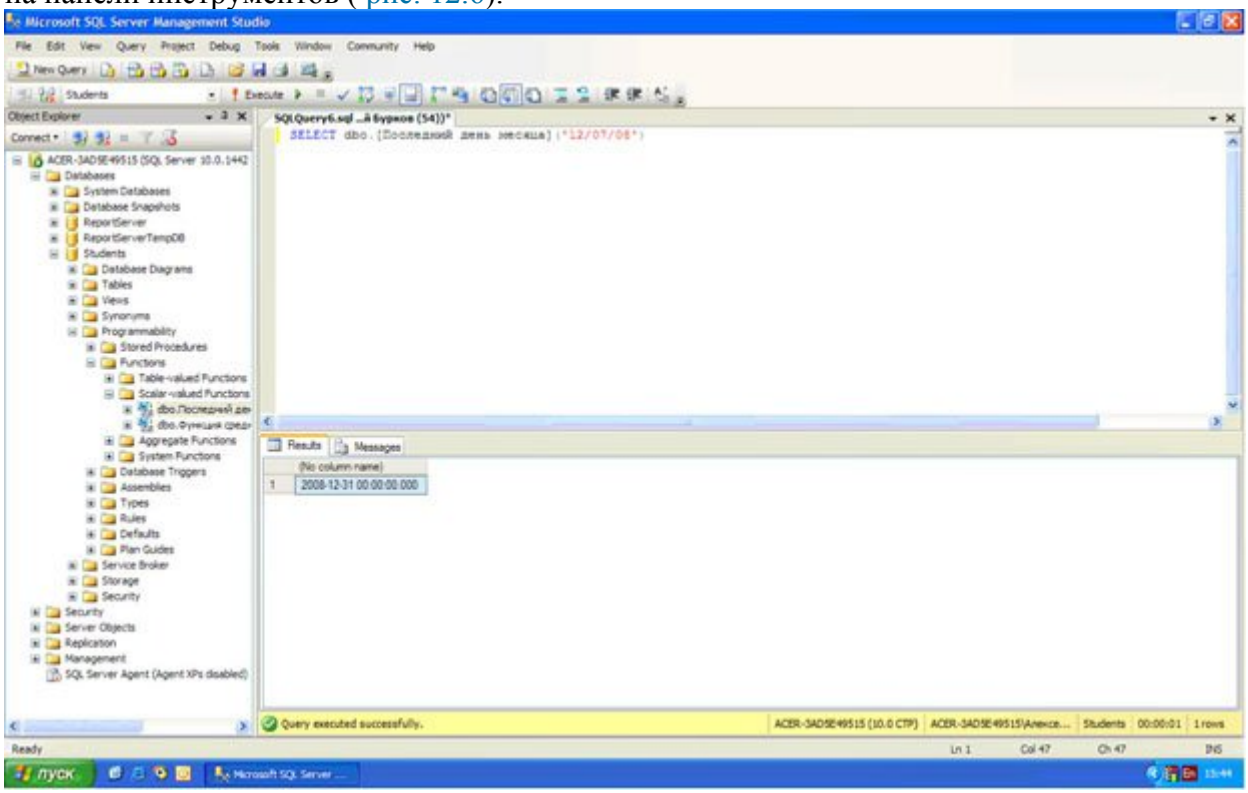


После появления сообщения **"Command(s) completed successfully."** закройте окно с кодом.

Проверим работу функции **"Последний день месяца"** выполнив ее. Создайте новый пустой запрос, затем в окне с пустым запросом наберите команду SELECT dbo.[Последний день месяца] ('12/07/08') и нажмите кнопку



на панели инструментов (рис. 12.6).

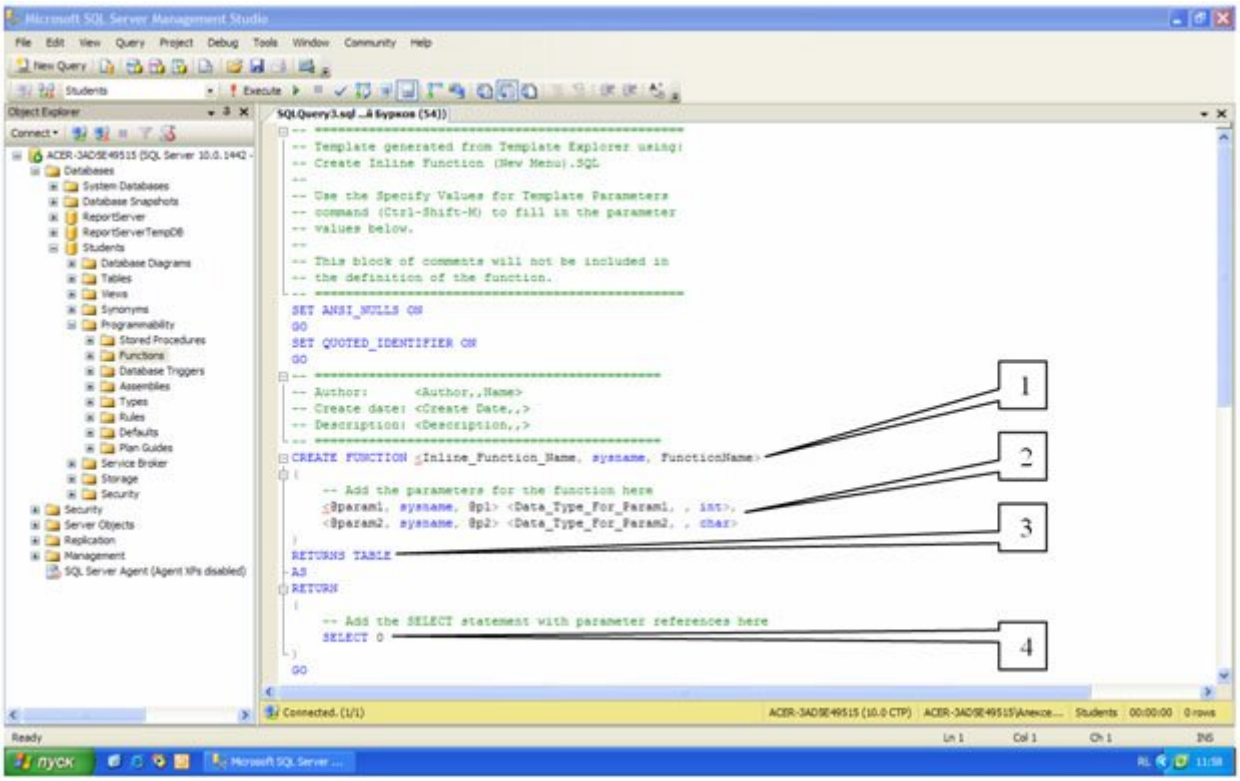


увеличить изображение

Рис. 12.6.

Появится результат выполнения новой скалярной пользовательской функции: **2008-12-31** (рис. 12.6).

Теперь перейдем к созданию табличных пользовательских функций. Для создания табличной пользовательской функции в обозревателе объектов, в БД **"Students"**, в папке **"Programmability"**, щелкните **ПКМ** по папке **"Functions"** и в появившемся меню выберите пункт **"New/Table-valued Function"**. Появится окно новой табличной пользовательской функции (рис. 12.7)



увеличить изображение

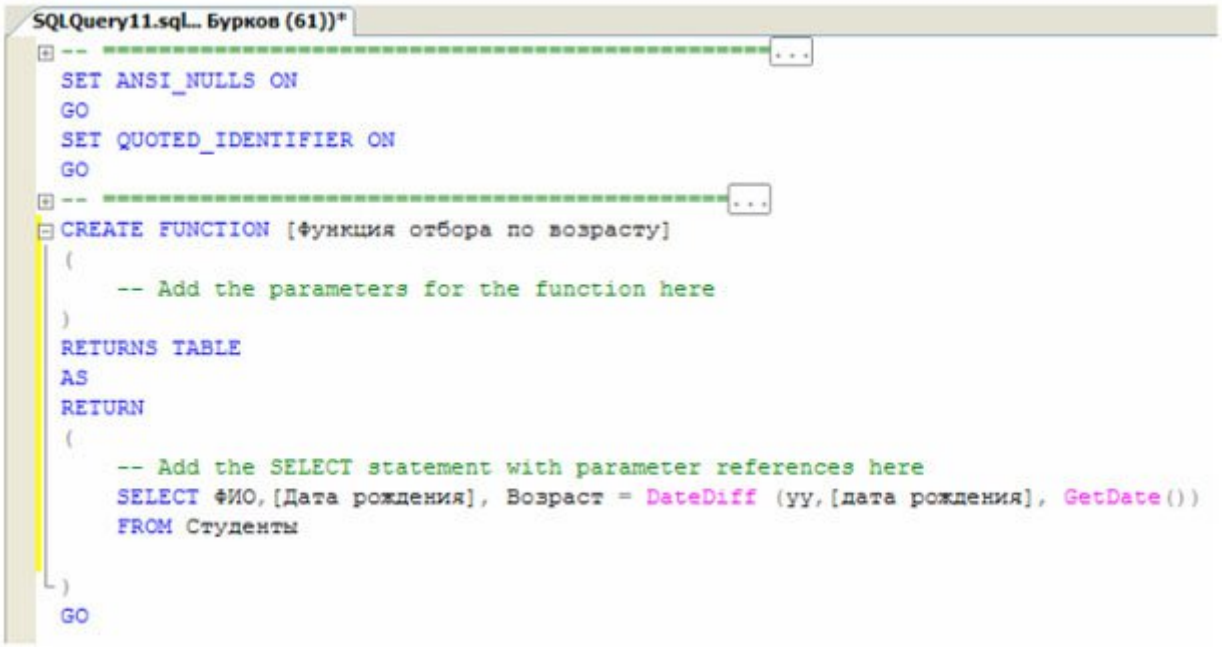
Рис. 12.7.

Рассмотрим структуру кода табличной пользовательской функции. Табличная пользовательская функция состоит из следующих разделов:

1. Область определения имени функции (Inline_Function_Name);
2. Параметры, передаваемые в процедуру (**@Param1**, **@Param2**);
3. RETURNS TABLE показывает что функция является табличной, то есть возвращает таблицу;
4. Тело самой пользовательской функции, состоит из команды SELECT языка программирования запросов T-SQL.

Остальные разделы табличной пользовательской функции аналогичны таким же разделам хранимых процедур и скалярных пользовательских функций.

В заключение рассмотрим создание табличной пользовательской функции **"Функция отбора по возрасту"**, вычисляющих текущий возраст студентов в зависимости от их даты рождения. В окне новой пользовательской функции (рис. 12.7) наберите следующий код (рис. 12.8):



увеличить изображение

Рис. 12.8.

Из кода представленного на рис. 12.8 видно, что данная табличная функция не имеет параметров и реализуется командой

SELECT ФИО, [Дата рождения], Возраст = DateDiff(yy, [Дата рождения], GetDate())
FROM Студенты

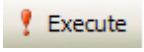
Из вышепредставленной команды видно, что из таблицы **"Студенты"** отображаются поля **"ФИО"** и **"Дата рождения"**, а также вычислимое поле **"Возраст"**. Поле **"Возраст"** вычисляется при помощи встроенной функции **DateDiff** вычисляющей различие между датами в определенных единицах измерения (частях даты) и имеющей следующий синтаксис:

DateDiff(<часть даты>, <начальная дата>, <конечная дата>).

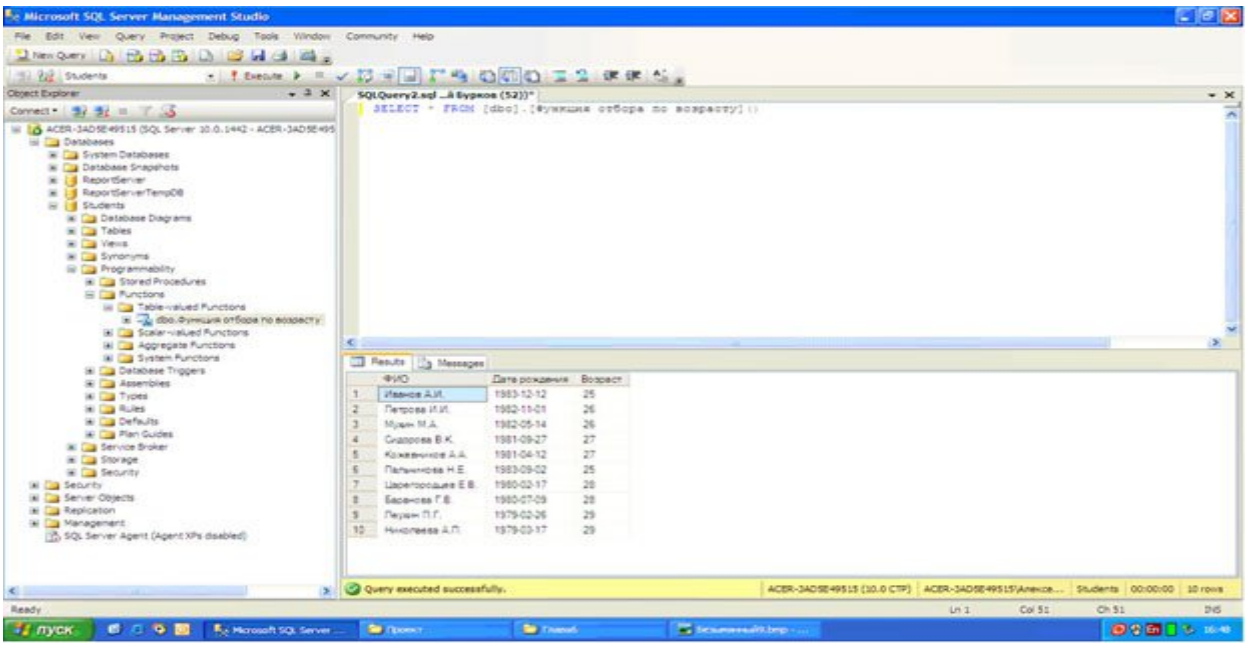
Здесь "часть даты" - это закодированные специальными символами единицы измерения (часть даты) (yy - год, mm - месяц, dd - день), "начальная дата" - дата начала периода и "конечная дата" - дата конца периода. В нашем случае в качестве начальной даты берем дату рождения студента, а в качестве конечной даты берем текущую дату (функция **GetDate()**).

Для создания функции, выполним вышеописанный код, как и в случае с предыдущей функцией. После появления сообщения **"Command(s) completed successfully."** закройте окно с кодом.

Проверим работоспособность новой табличной пользовательской функции. Создайте новый пустой запрос, затем в окне с пустым запросом наберите команду SELECT * FROM dbo.[Функция отбора по возрасту]() и нажмите кнопку



на панели инструментов (рис. 12.9).



увеличить изображение

Рис. 12.9.

В нижней части окна появиться таблица с фамилиями, датами рождения и возрастом студентов на данный момент времени (рис. 12.9).

Замечание: Обратите внимание на тот факт, что мы работаем с табличной функцией как с обыкновенной таблицей.

На этом мы заканчиваем рассмотрение пользовательских функций и переходим к рассмотрению целостности данных, диаграмм и триггеров. По окончании выполнения главы 6 обозреватель объектов будет иметь следующий вид (рис. 12.10):

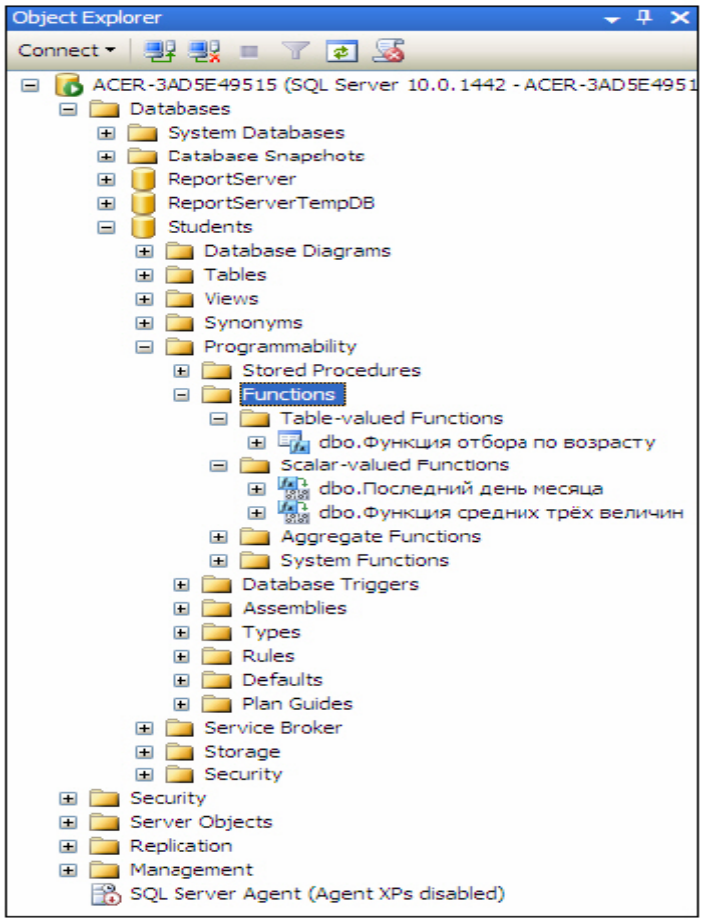


Рис. 12.10.

Контрольные вопросы

1 Назначение пользовательских функций.

2. Синтаксис скалярной пользовательской функции.

3. Табличные пользовательские функции.

Лабораторная работа № 9

«Обеспечение целостности данных в Microsoft SQL Server 2012. Создание диаграмм и триггеров.»

Цель работы:

Научиться создавать диаграммы и триггеры.

Перейдем теперь к созданию диаграмм. В БД "Microsoft SQL Server 2008" все диаграммы находятся в папке **"Database Diagrams"** обозревателя объектов (рис. 14.1).

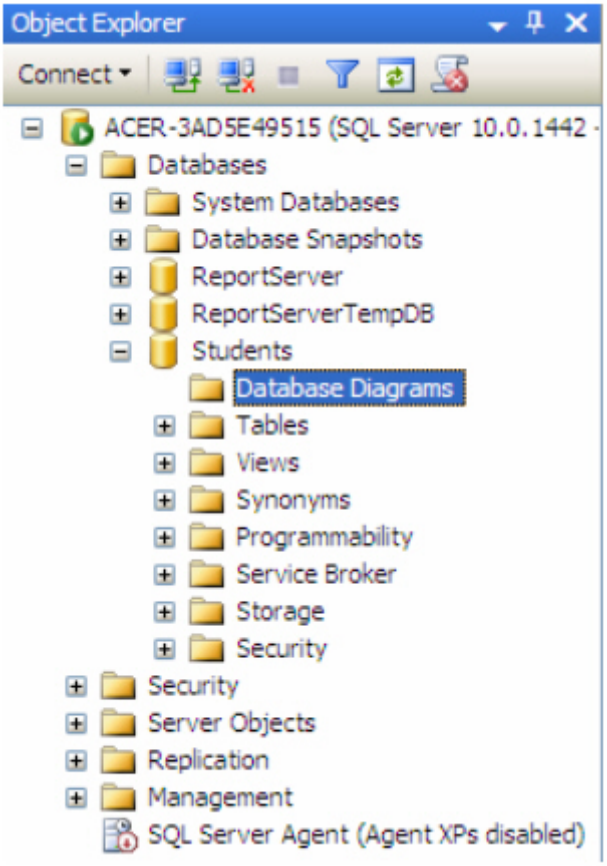


Рис. 14.1.

Создадим диаграмму, обеспечивающую целостность данных нашей БД **"Students"**. Для создания новой диаграммы в БД **"Students"** щелкните **ПКМ** по папке **"Database Diagrams"** и в появившемся меню выберем пункт **"New Database Diagram"**. Сначала появится окно с вопросом о добавлении нового объекта **"Диаграмма"**. В этом окне нужно нажать кнопку **"Yes"**. Затем появится окно **"Add Table"** предназначенное для добавления таблиц в новую диаграмму (рис. 14.2).

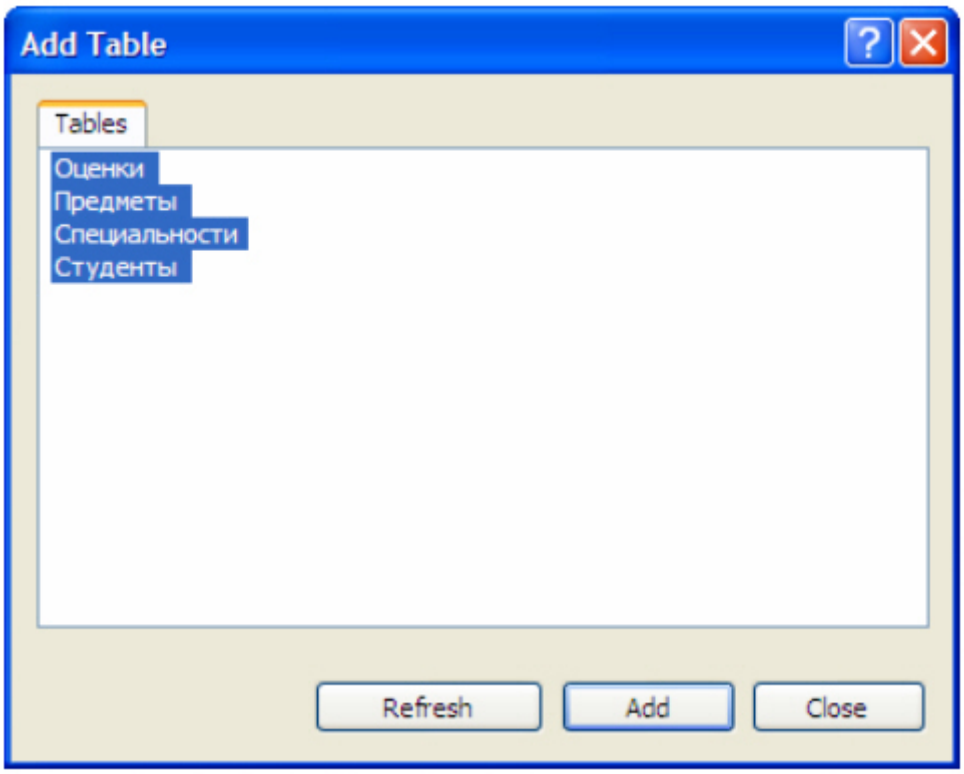


Рис. 14.2.

В окне добавления таблиц выделите все таблицы нашей БД и нажмите кнопку **"Add"** (рис. 14.2). Закройте окно **"Add Table"** нажатием на кнопку **"Close"**.

Появится окно диаграммы, где будут отображены отобранные таблицы. Теперь необходимо определить связи между таблицами. Перетащите поле **"Код специальности"** из таблицы **"Специальности"** на такое же поле в таблице **"Студенты"**. Появится окно создания связи между таблицами **"Tables and Columns"** (рис. 14.3).

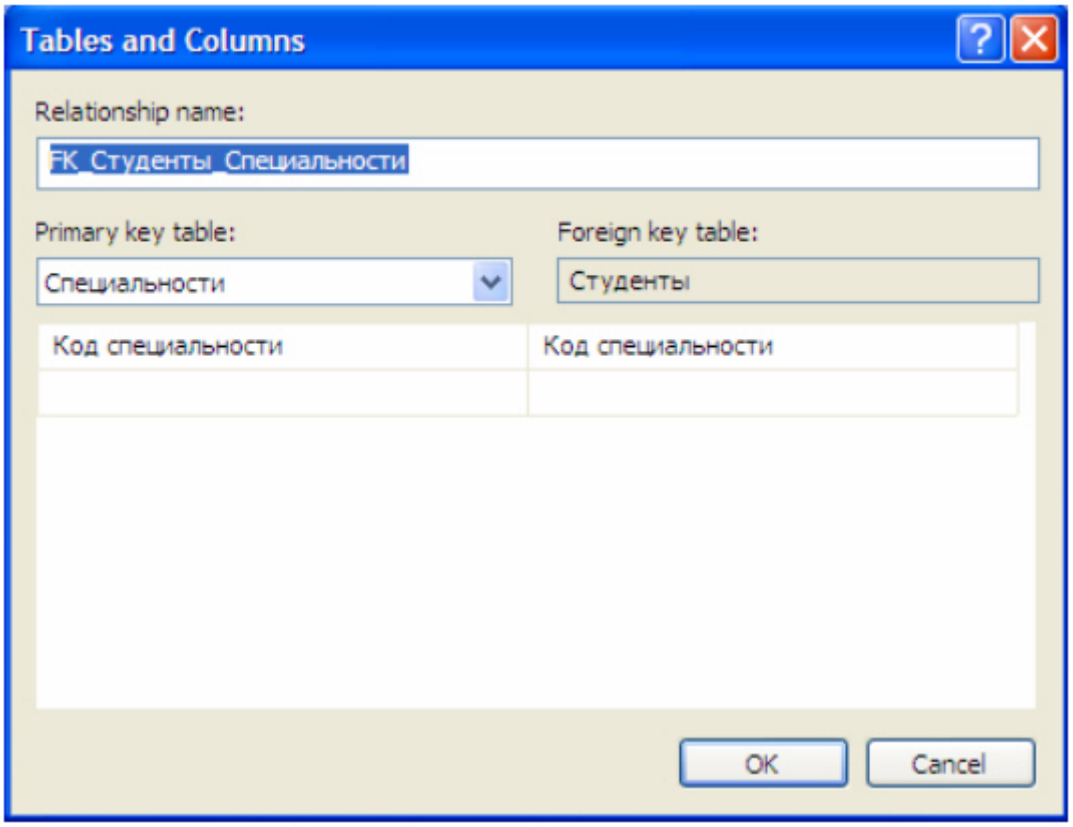
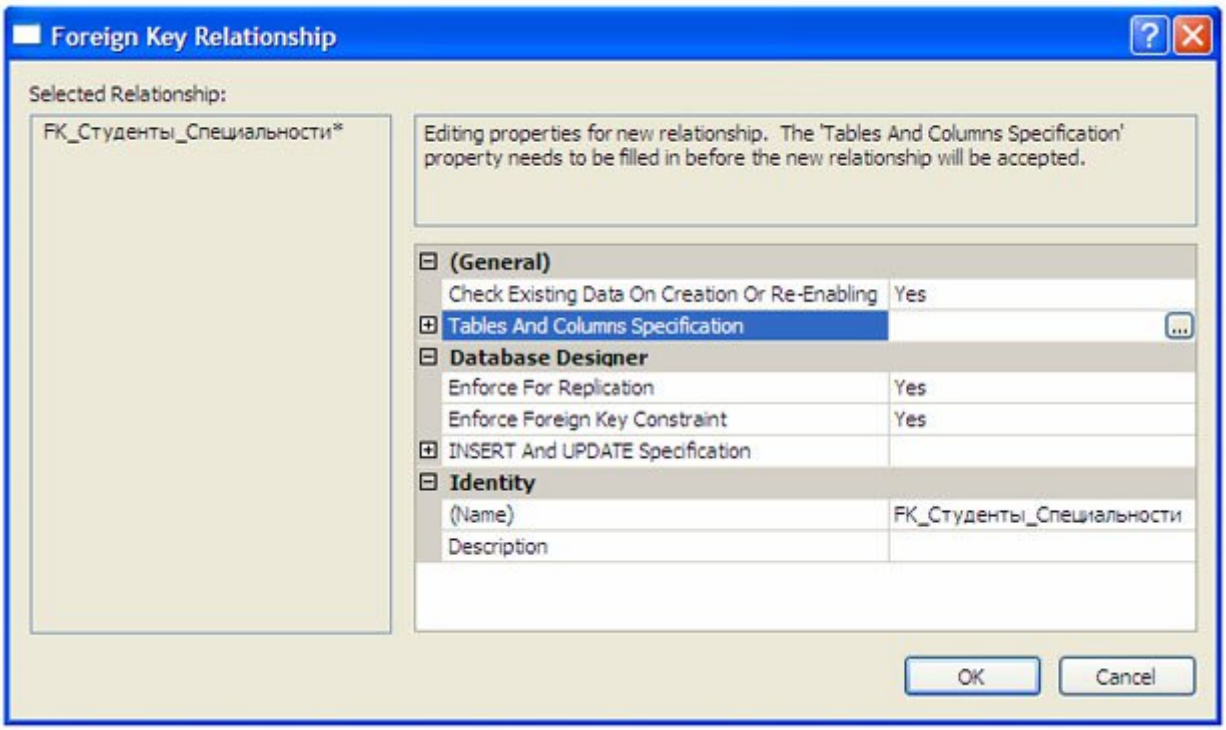


Рис. 14.3.

В окне создания связи нажмите кнопку **"Ok"**. Появится окно настройки свойств связи **"Foreign Key Relationship"** (рис. 14.4).

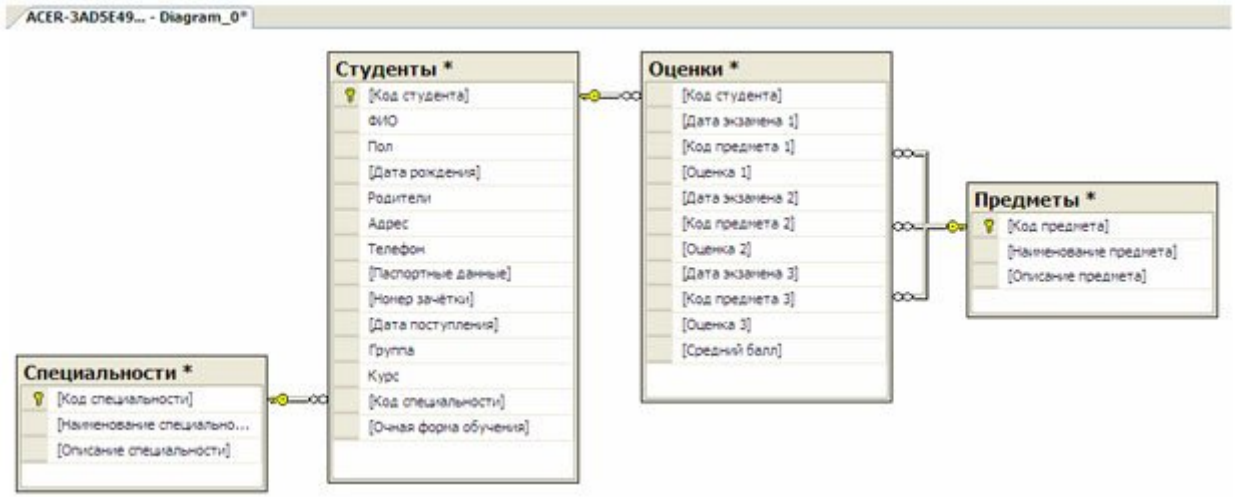


увеличить изображение

Рис. 14.4.

Оставьте свойства связи без изменений и в окне свойств связи нажмите кнопку **"Ok"**.В диаграмме между таблицами **"Студенты"** и **"Специальности"** появится связь в виде ломаной линии (рис. 14.5).

Аналогичным образом создайте связь таблицы **"Студенты"** с таблицей **"Оценки"**, перетащив поле **"Код студента"** из таблицы **"Студенты"** на одноименное поле в таблице **"Оценки"**. Затем, свяжите таблицы **"Предметы"** и **"Оценки"**, перетащив поле **"Код предмета"** из таблицы **"Предметы"** на поля **"Код предмета 1"**, **"Код предмета 2"** и **"Код предмета 3"** таблицы **"Оценки"**. После выполнения вышеперечисленных действий диаграмма примет следующий вид (рис. 14.5).



увеличить изображение

Рис. 14.5.

Закройте окно с диаграммой, щелкнув мышью по кнопке закрытия

расположенной в верхнем правом углу окна с диаграммой. Появится окно с вопросом о сохранении новой диаграммы, где необходимо нажать кнопку **"Yes"** (рис. 14.6).

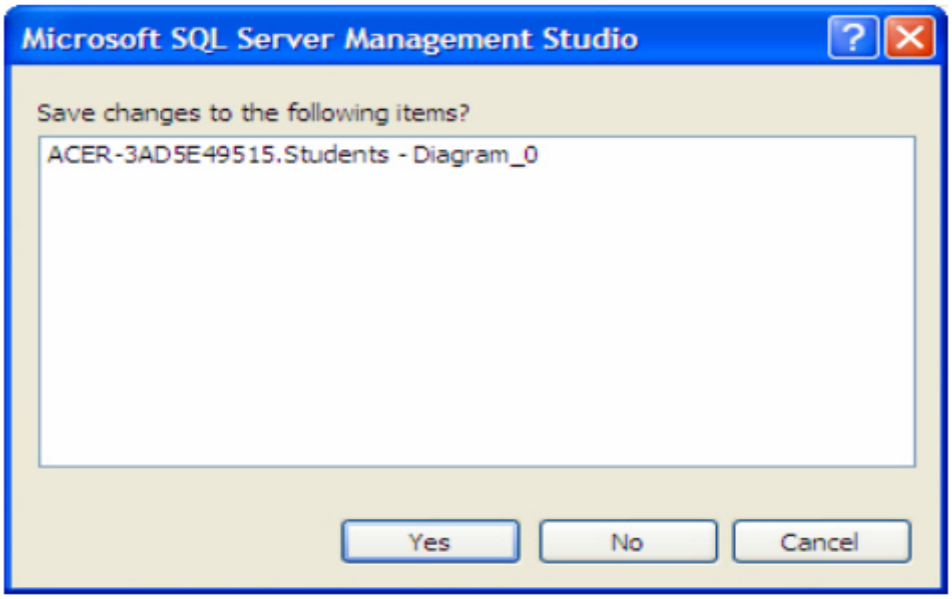


Рис. 14.6.

Появится окно определения имени новой диаграммы **"Choose Name"**. В окне определения имени, задайте имя диаграммы как "Диаграмма БД Студенты" и нажмите кнопку **"Ok"** (рис. 14.7).

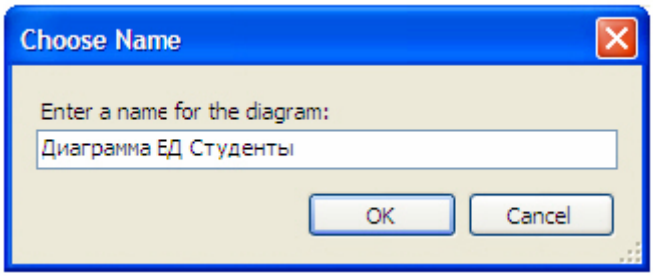


Рис. 14.7.

Появится окно **"Save"** с запросом сохранения таблиц, входящих в диаграмму. В данном окне необходимо нажать кнопку **"Yes"** (рис. 14.8).

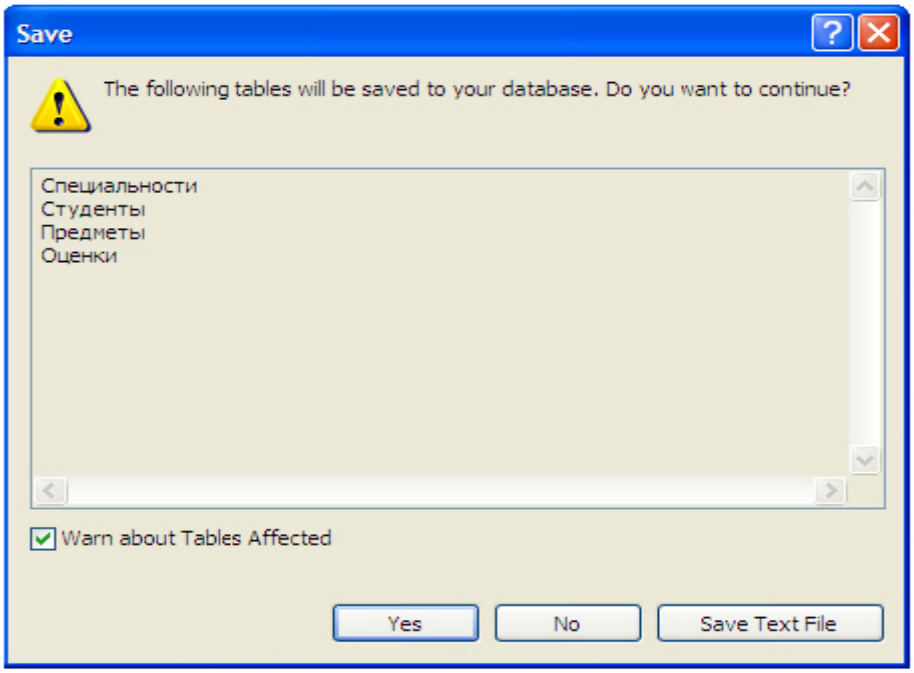


Рис. 14.8.

Перейдем к созданию триггеров. Создадим триггеры для таблицы **"Студенты"**. Триггеры создаются отдельно для каждой таблицы и располагаются в обозревателе объектов в папке **"Triggers"**. В нашем случае, папка **"Triggers"** входит в состав таблицы **"Студенты"** (рис. 14.9).

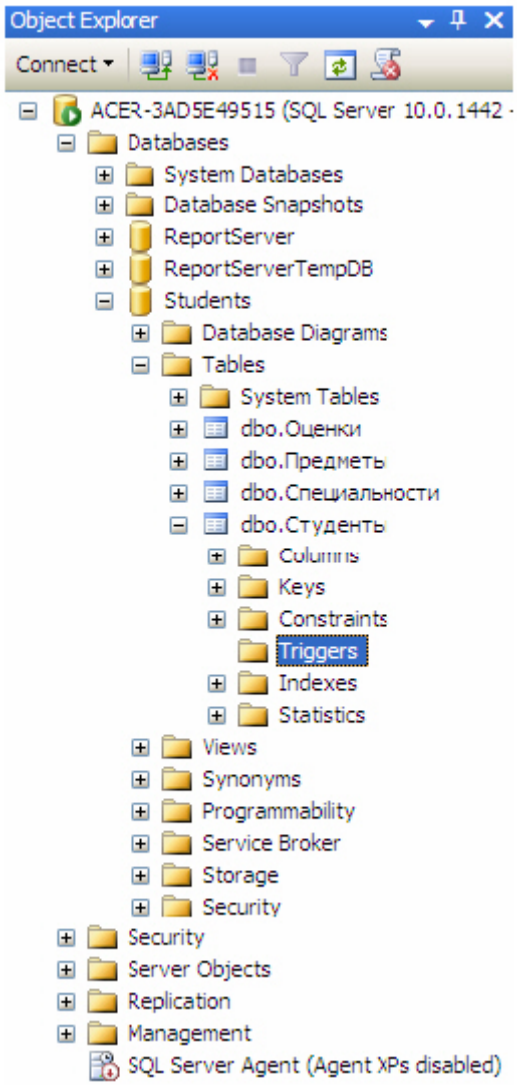
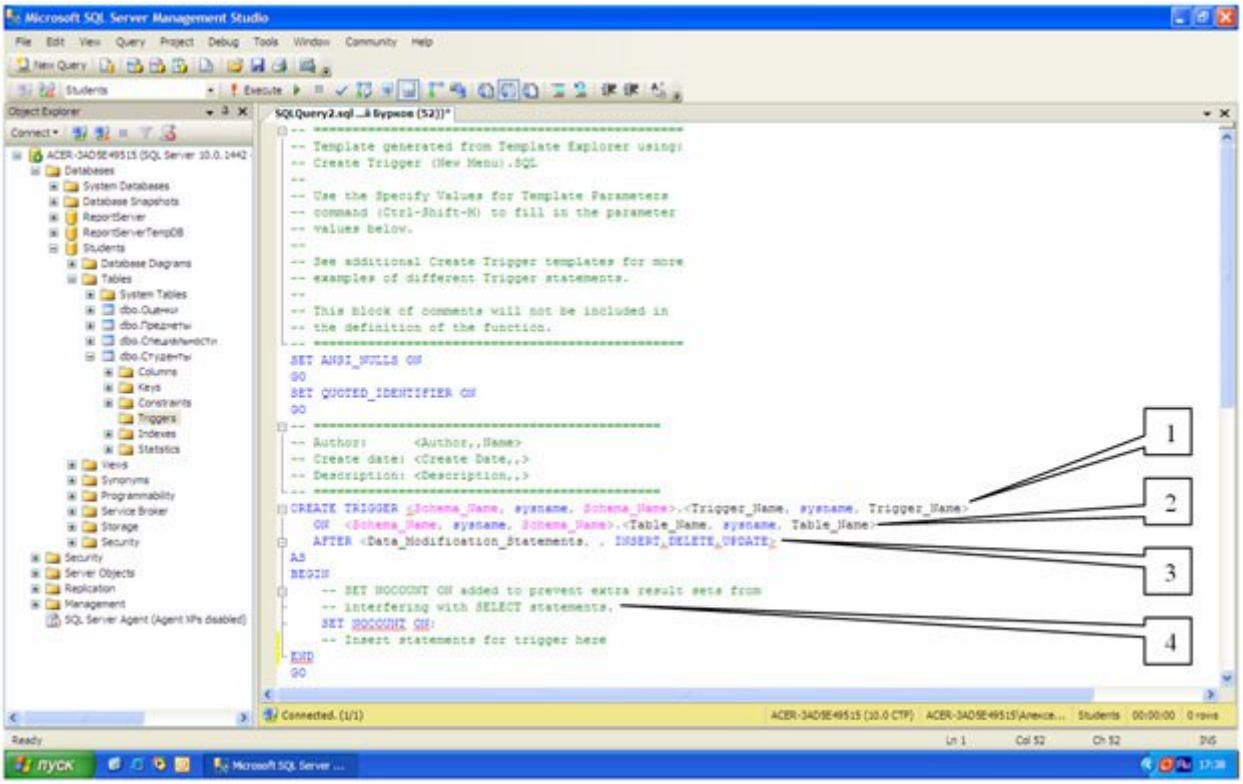


Рис. 14.9.

Для начала создадим триггер, выводящий сообщение **"Запись добавлена"** при добавлении записи в таблицу **"Студенты"**. Создадим новый триггер, щелкнув **ПКМ** по папке **"Triggers"** в таблице **"Студенты"** и в появившемся меню выбрав пункт **"New Trigger"**. Появится следующее окно с новым триггером (рис. 14.10):



увеличить изображение

Рис. 14.10.

Рассмотрим структуру триггеров:

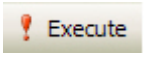
1. Область определения имени функции (**Trigger_Name**);
2. Область, показывающая для какой таблицы создается триггер (**Table_Name**);
3. Область, показывающая когда выполнять триггер (**INSERT** - при создании записи в таблице, **DELETE** - при удалении и **UPDATE** - при изменении) и как его выполнять (**AFTER** - после выполнения операции, **INSTEAD OF** - вместо выполнения операции);
4. Тело триггера, содержит команды языка программирования запросов T-SQL.

В окне нового триггера наберите код как показано на рис. 14.11.

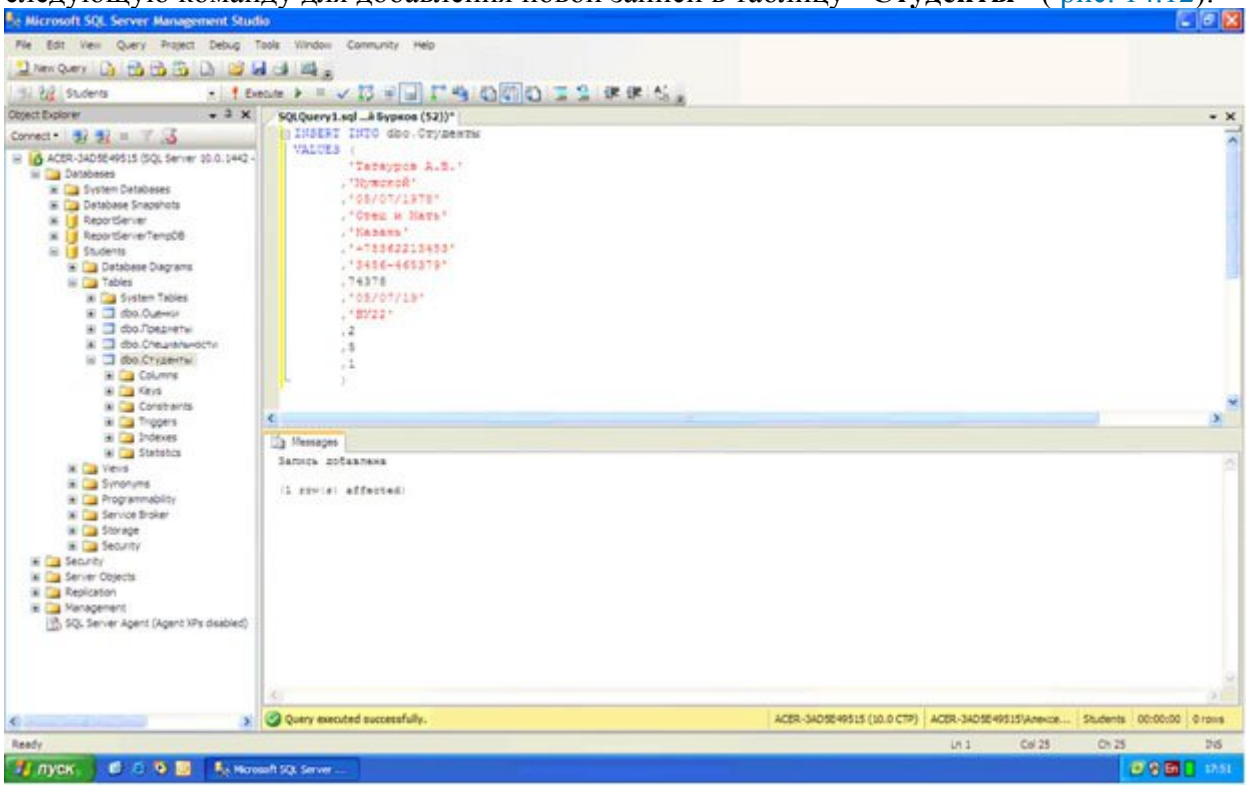
```
SQLQuery6.sql ...й Бурков (52))*
-- ============================================...
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- ============================================...
CREATE TRIGGER [Индикатор добавления]
   ON  dbo.Студенты
   AFTER INSERT
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from...
    SET NOCOUNT ON;
    -- Insert statements for trigger here
    PRINT 'Запись добавлена'
END
GO
```

Рис. 14.11.

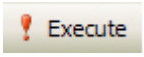
Из рис. 14.11 видно, что создаваемый триггер **"Индикатор добавления"** выполняется после добавления записи (**AFTER INSERT**) в таблицу "Студенты" (**ON dbo.Студенты**). После добавления записи триггер выведет на экран сообщение "Запись добавлена" (**PRINT 'Запись добавлена'**). Выполните набранный код, нажав кнопку



на панели инструментов. В нижней части окна с кодом появится сообщение **"Command(s) completed successfully."**.
Проверим, как работает новый триггер. Создайте новый пустой запрос и в нем наберите следующую команду для добавления новой записи в таблицу **"Студенты"** (рис. 14.12):



увеличить изображение
Рис. 14.12.
Выполните набранную команду, нажав кнопку



на панели инструментов. В таблицу будет добавлена новая запись, и триггер выведет сообщение **"Запись добавлена"** (рис. 14.12).
Теперь создадим триггер отображающий сообщение **"Запись изменена"**. Создайте новый триггер, как в предыдущем случае. В окне нового триггера наберите следующий код (рис. 14.13):

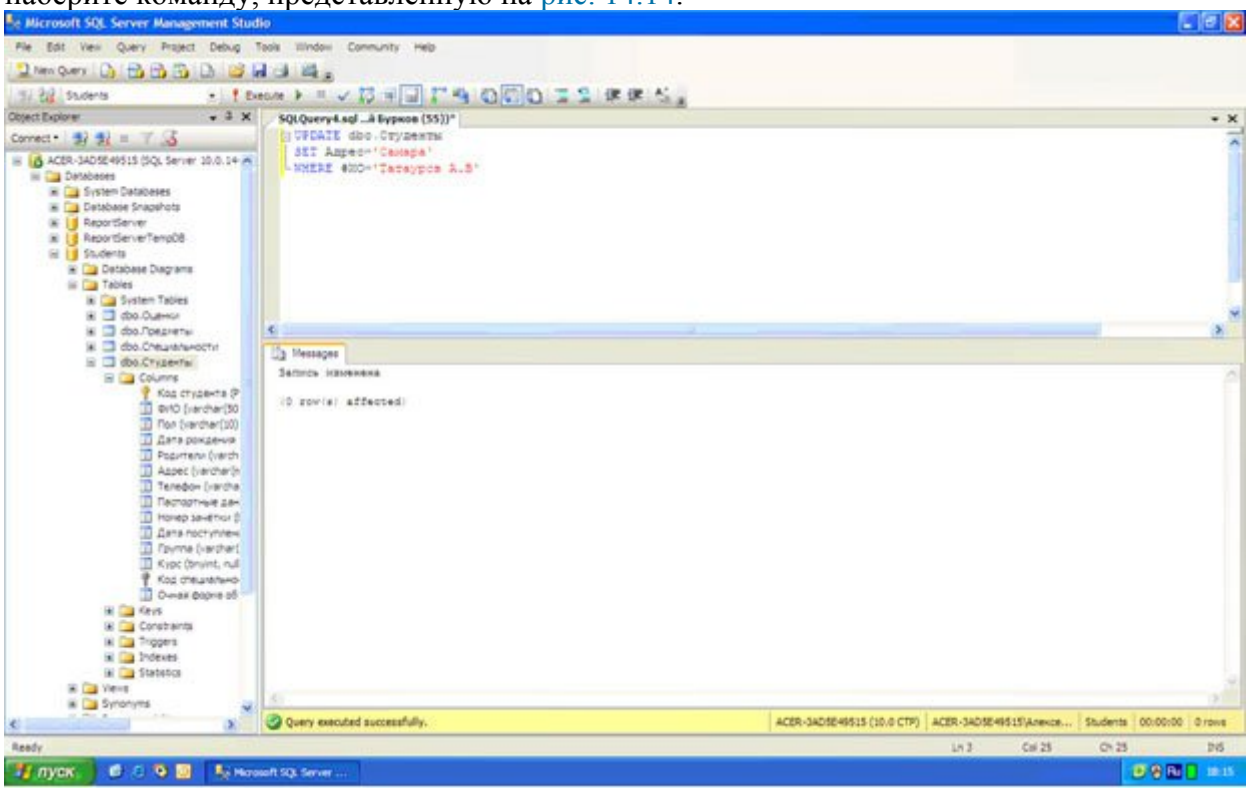
```
SQLQuery8.sql ...й Бурков (51))*
-- ============================================...
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- ============================================...
CREATE TRIGGER [Индикатор изменения]
   ON  dbo.Студенты
   AFTER UPDATE
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from...
    SET NOCOUNT ON;
    -- Insert statements for trigger here
    PRINT 'Запись изменена'
END
GO
```

увеличить изображение

Рис. 14.13.

Из рис. 14.13 видно, что новый триггер **"Индикатор изменения"** выполняется после изменения записи (**AFTER UPDATE**) в таблице "Студенты" (**ON dbo.Студенты**). После изменения записи триггер выведет на экран сообщение "Запись изменена" (**PRINT 'Запись изменена'**). Выполните набранный код. В нижней части окна с кодом появится сообщение **"Command(s) completed successfully."**.

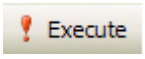
Проверим работоспособность созданного триггера. Создайте новый запрос и в нем наберите команду, представленную на рис. 14.14.



увеличить изображение

Рис. 14.14.

Выполните набранную команду, нажав кнопку



на панели инструментов. В таблицу будет добавлена новая запись, и триггер выведет сообщение **"Запись изменена"** (рис. 14.14).

Для полноты картины создадим триггер, выводящий сообщение при удалении записи из таблицы **"Студенты"**. Создайте новый триггер и в нем наберите код, показанный на рис. 14.15.

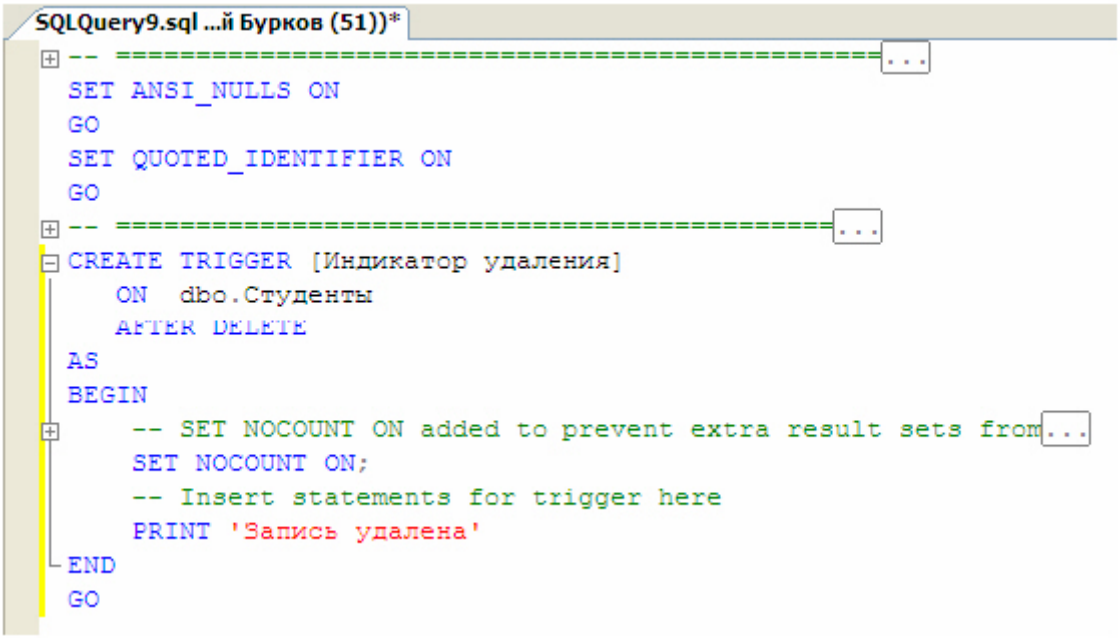
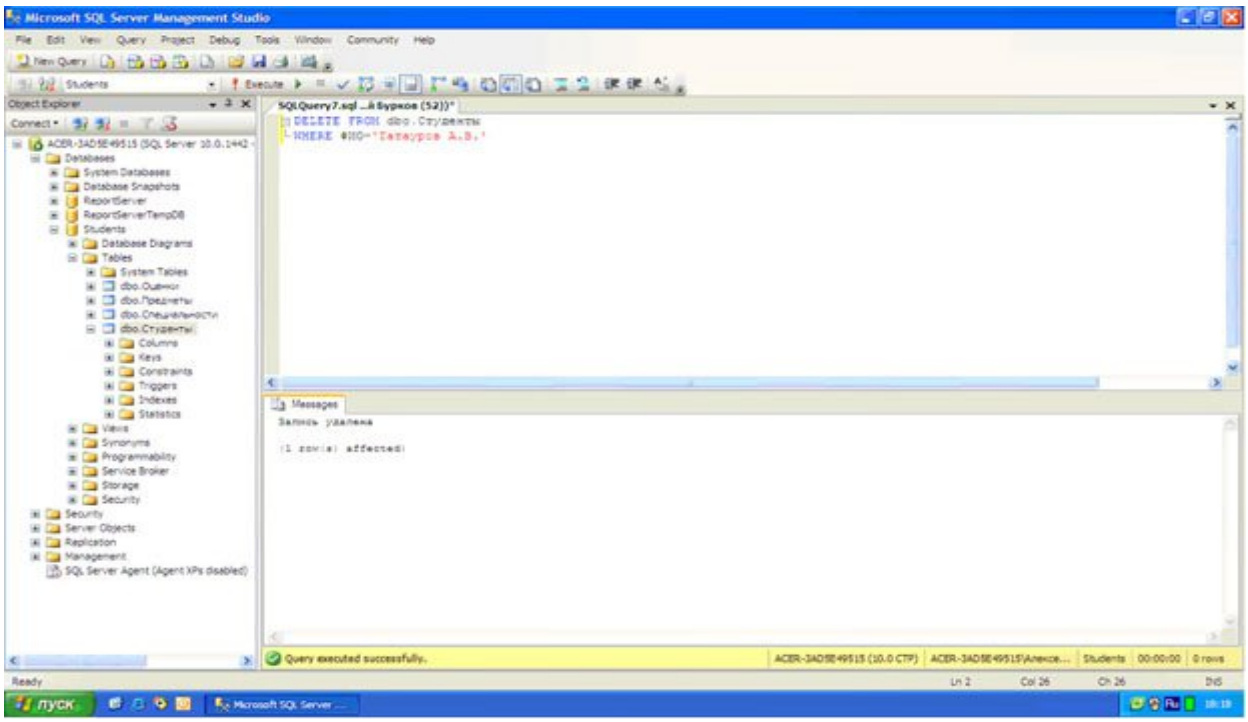


Рис. 14.15.

Создаваемый триггер **"Индикатор удаления"** выполняется после удаления записи (**AFTER DELETE**) из таблицы студенты (**ON dbo.Студенты**). После удаления записи триггер выводит сообщение "Запись удалена" (**PRINT 'Запись удалена'**).

Выполните код, представленный рис. 14.15. В нижней части окна с кодом появится сообщение **"Command(s) completed successfully."**.

Проверим работу триггера **"Индикатор удаления"** удалив созданную ранее запись из таблицы **"Студенты"**. Для этого создайте новый запрос и в нем наберите следующую команду (рис. 14.16):



увеличить изображение

Рис. 14.16.

Выполните вышеприведенную команду. После удаления записи триггер **"Индикатор удаления"** отобразит сообщение **"Запись удалена"** (рис. 14.16).

В заключение рассмотрим пример применения триггеров для обеспечения целостности данных. Создадим триггер **"Удаление студента"**, который при удалении записи из таблицы Студенты сначала удаляет все связанные с ней записи из таблицы **"Оценки"**, а затем удаляет саму запись из таблицы **"Студенты"**, тем самым обеспечивается целостность данных.

Создайте новый триггер и в нем наберите следующий код (рис. 14.17):

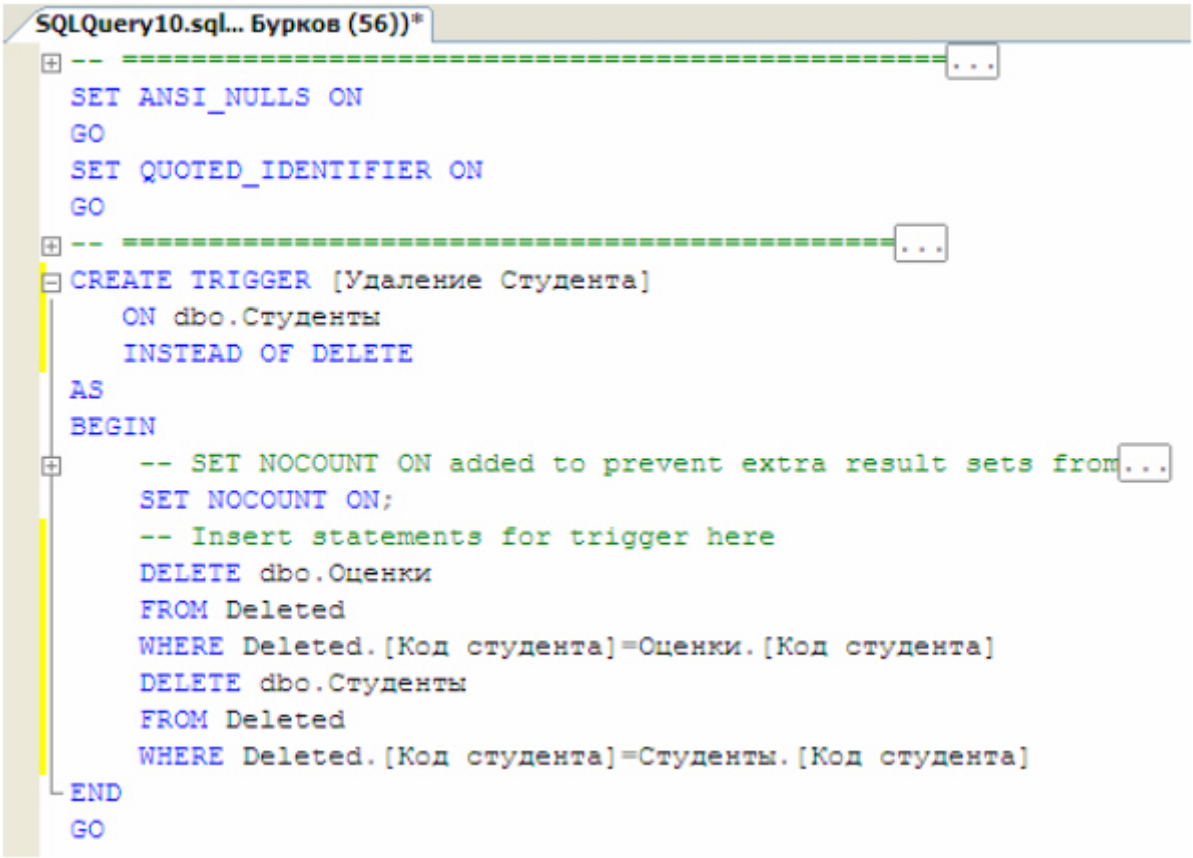


Рис. 14.17.

Создаваемый триггер **"Удаление студента"** выполняется вместо удаления записи (**INSTEAD OF DELETE**) из таблицы "Студенты" (**ON dbo.Студенты**).

Замечание: При срабатывании триггера вместо удаления записи создается временная константа **Deleted**, содержащая имя таблицы из которой должно было быть произведено удаление.

После срабатывания триггера из таблицы **"Оценки"** удаляется запись, у которой значение поля **"Код студента"** равно значению такого же поля у удаляемой записи из таблицы **"Студенты"**. Эту операцию выполняют следующие команды:

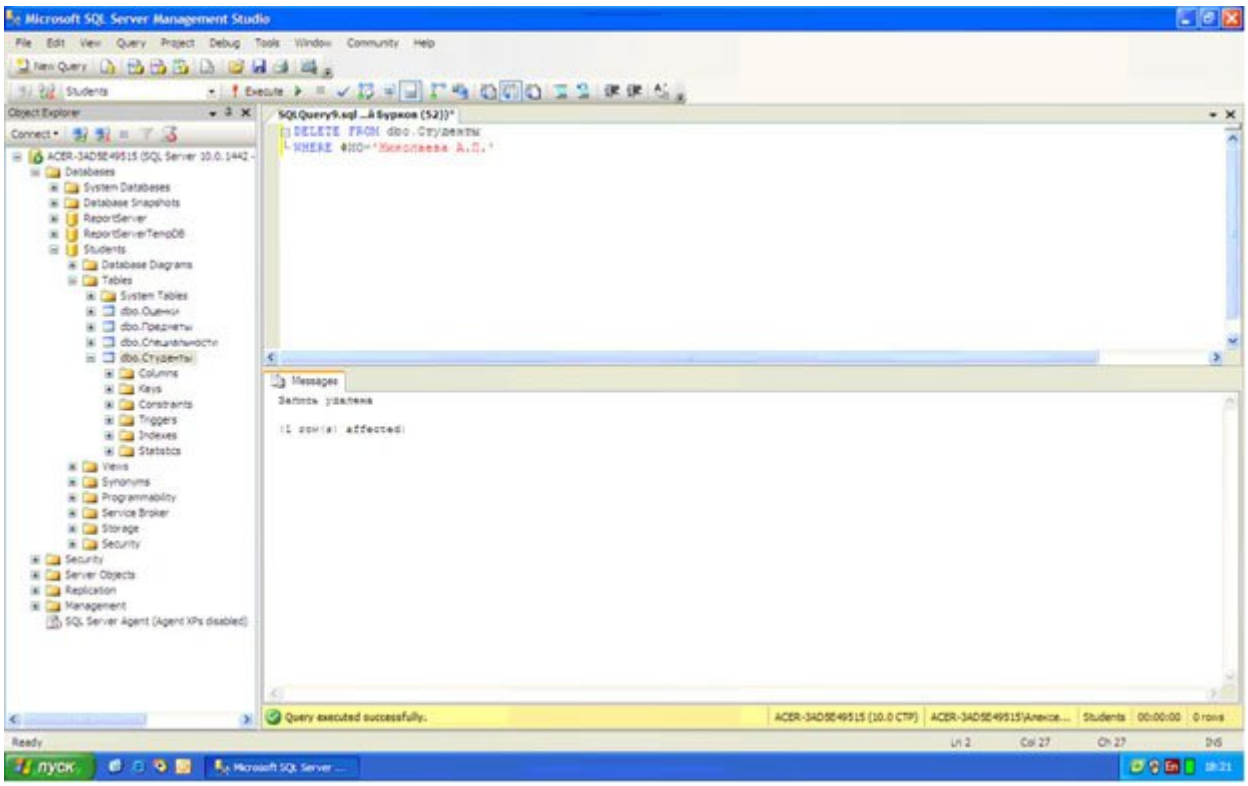
DELETE dbo.Оценки FROM Deleted
WHERE Deleted.[Код студента] = Оценки.[Код студента]

Затем удаляется запись из таблицы **"Студенты"**, которую удаляли до срабатывания триггера. Удаление выполняется следующими командами:

DELETE dbo.Студенты
FROM Deleted
WHERE Deleted.[Код студента] = Студенты.[Код студента]

Выполните код, представленный на рис. 14.17. В нижней части окна с кодом появиться сообщение **"Command(s) completed successfully."**.

Проверим, как работает триггер **"Удаление студента"**. Для этого создайте новый запрос и в нем наберите следующий код (рис. 14.18):



увеличить изображение

Рис. 14.18.

При срабатывании триггера сначала из таблицы **"Оценки"** удалятся все связанные с удаляемой записью записи, а затем удаляется сама удаляемая запись из таблицы **"Студенты"**, при этом сохраняется целостность данных.

Замечание: Хотелось бы заметить, что без использования триггера **"Удаление студента"** нам бы не удалось удалить запись из таблицы **"Студенты"**. Команда удаления была бы заблокирована диаграммой **"Диаграмма БД Студенты"** во избежание нарушения целостности данных.

На этом мы завершаем работу с диаграммами и триггерами. После выполнения всех вышеописанных действий обозреватель объектов будет иметь следующий вид (рис. 14.19):

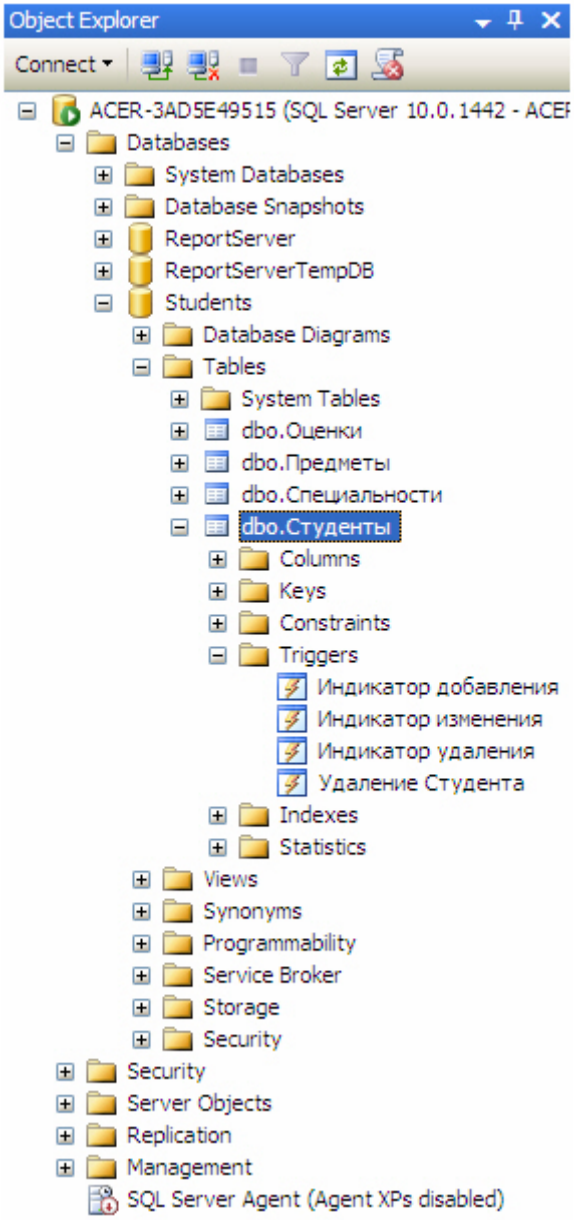


Рис. 14.19.

Контрольные вопросы

1. Структура триггеров.
2. Для чего используется диаграмма.
3. Связи между таблицами.

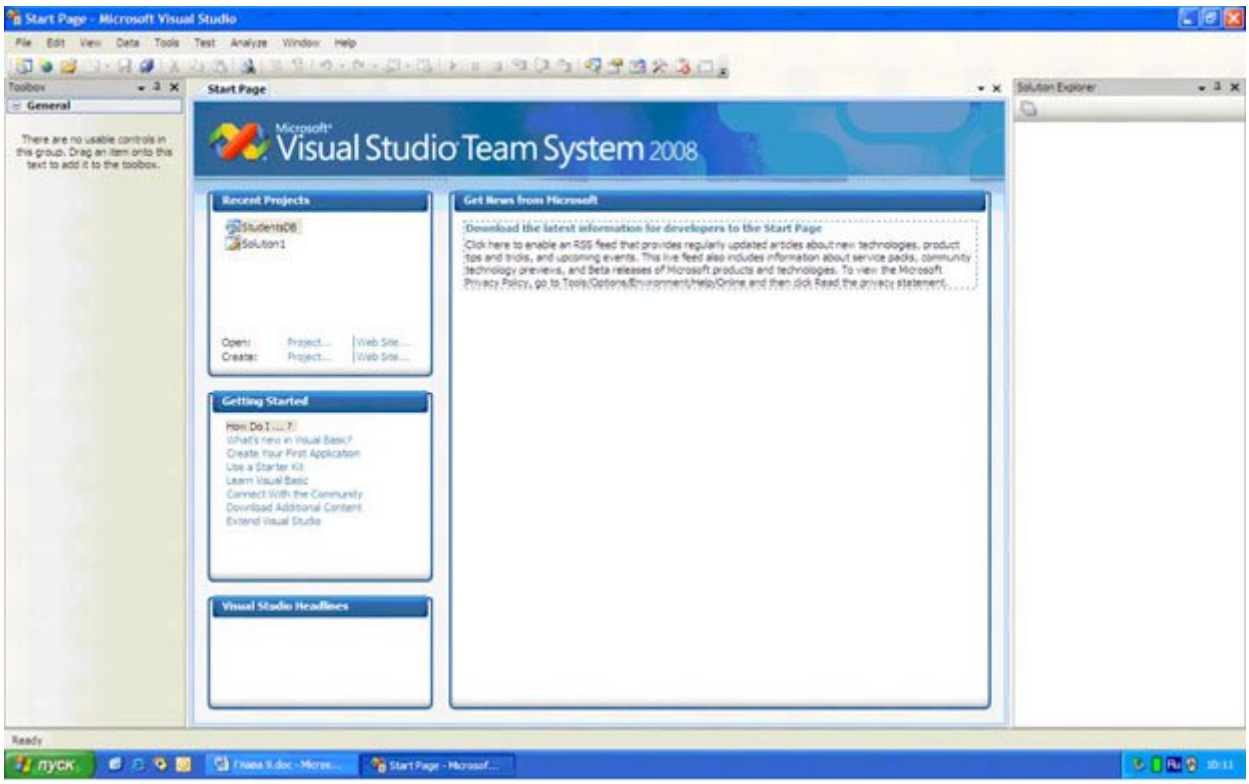
Лабораторная работа № 10

«Создание простых ленточных форм для работы с базами данных в Visual Studio 2012.»

Цель работы:

Научиться создавать пользовательский интерфейс (главная кнопочная форма, простые ленточные формы для работы с данными).

Перейдем теперь к созданию пользовательского интерфейса. Его создание начнем с создания главной кнопочной формы. Запустите "Microsoft Visual Studio 2008" и откройте созданный ранее проект "StudentsDB", щелкнув по его значку в области "Recent Projects" стартовой страницы "Start Page" (рис. 18.1).



увеличить изображение

Рис. 18.1.

После появления стандартного окна среды разработки в рабочей области на форму поместите надпись (**Label**) и четыре кнопки (**Button**) как показано на рис. 18.2.

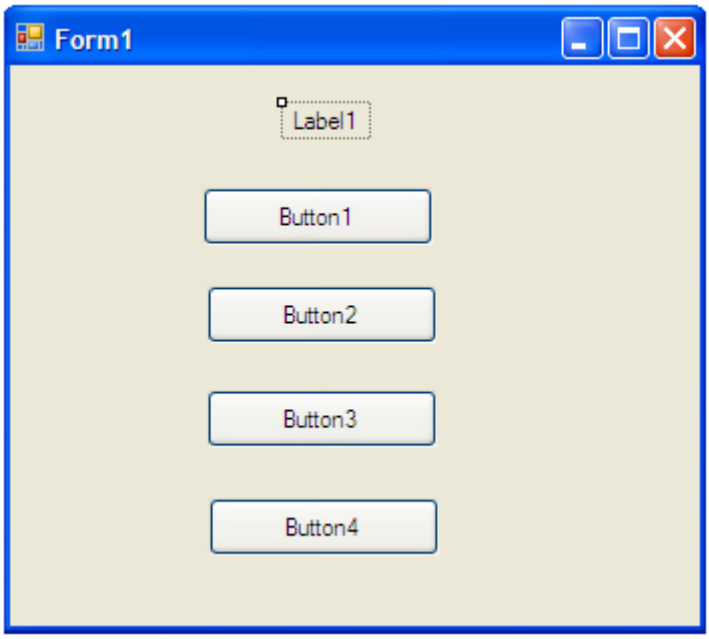
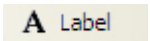
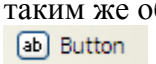


Рис. 18.2.

Замечание: Для создания надписи на панели объектов необходимо нажать кнопку



а затем нарисовать прямоугольник мышью на форме, удерживая **ЛКМ**. Кнопки создаются таким же образом, только на панели объектов нажмите кнопку



После создания объектов перейдем к настройке их свойств. Начнем с настройки свойств формы. Выделите форму, щелкнув **ЛКМ** в пустом месте формы. На панели свойств задайте свойства формы как представлено ниже:

- **FormBorderStyle** (Стиль границы формы): Fixed3D;
- **MaximizeBox** (Кнопка развертывания формы во весь экран): False;
- **MinimizeBox** (Кнопка свертывания формы на панель задач): False;
- **Text** (Текст надписи в заголовке формы): База данных "Студент".

На форме выделите надпись, щелкнув по ней **ЛКМ** и на панели свойств, задайте свойства надписи следующим образом:

- **AutoSize** (Авторазмер): False;
- **Font** (Шрифт): Microsoft Sans Serif, размер 14;
- **ForeColor** (Цвет текста): Темно синий;
- **Text** (Текст надписи): База данных "Студент";
- **TextAlign** (Выравнивание текста): MiddleCenter.

У кнопок задайте надписи (свойство **"Text"**) как показано на рис. 18.3.

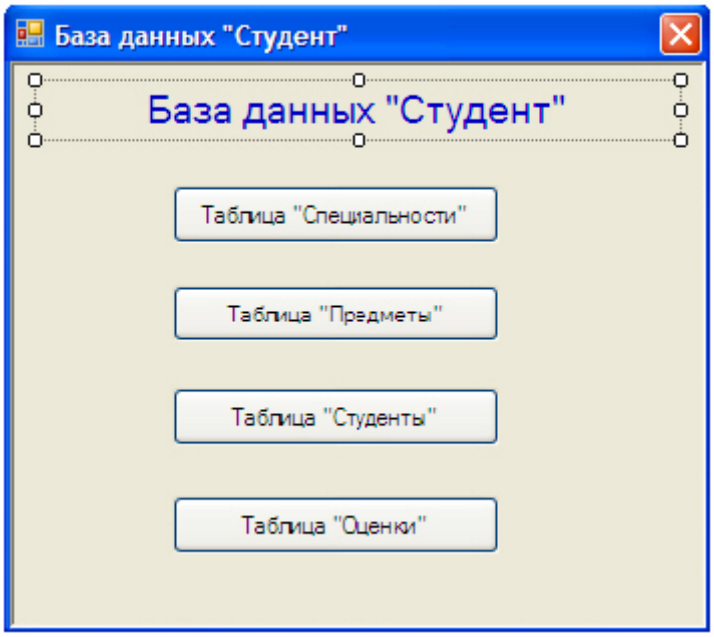
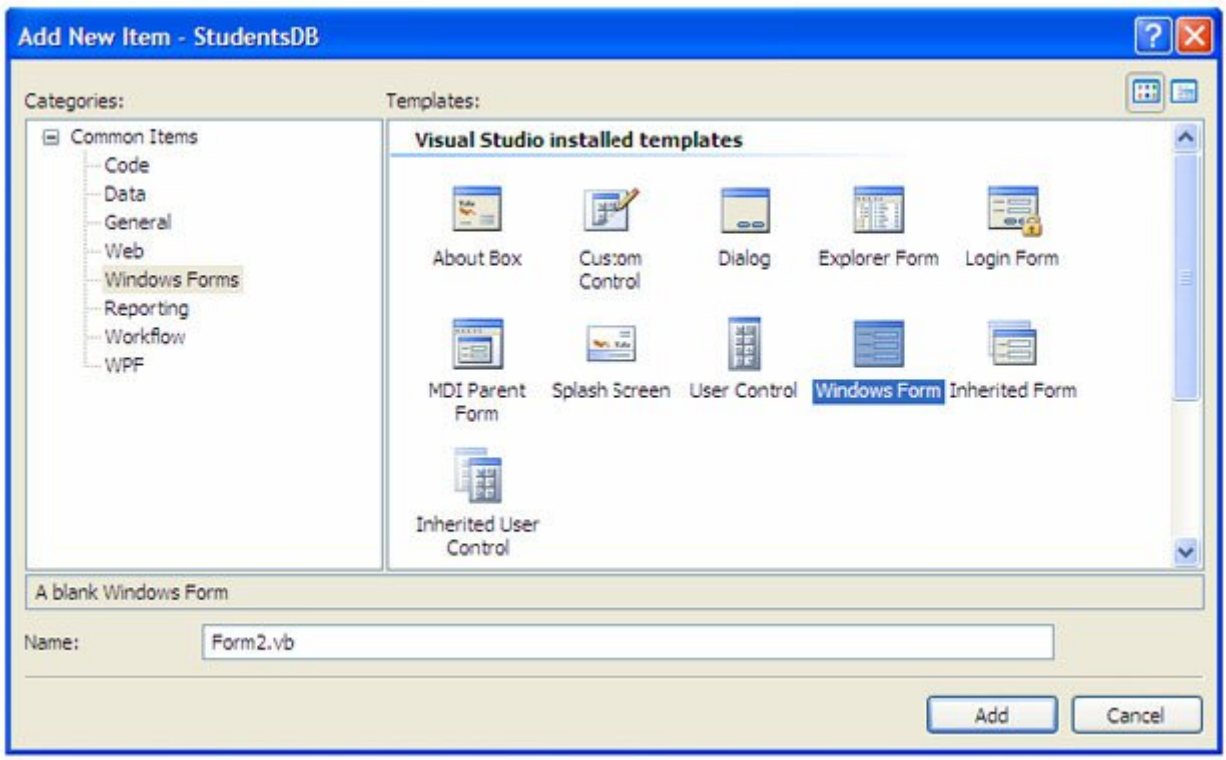


Рис. 18.3.

После настройки свойств вышеперечисленных объектов форма примет вид представленный на рис. 18.3.

Теперь перейдем к созданию простых ленточных форм для работы с данными. Для начала создадим ленточную форму, отображающую таблицу **"Специальности"**. Добавим в проект новую пустую форму. Для этого в оконном меню выберите пункт **"Project/Add Windows Form"**. Появится окно **"Add New Item - StudentsDB"** (Добавить новый компонент) (рис. 18.4).



увеличить изображение

Рис. 18.4.

В данном окне в разделе **"Categories:"** (Категории) выберите **"Windows Forms"** (Формы Windows), затем в разделе **"Templates:"** (Шаблоны) выберите **"Windows Form"** (Форма Windows) и нажмите кнопку **"Add"** (Добавить). Новая пустая форма появится в рабочей области среды разработки.

В верхней части новой формы создайте надпись (**Label**), как это показано на рис. 18.5.

Рис. 18.5.

Перейдем к настройке свойств формы и надписи. Выделите форму, щелкнув **ЛКМ** в пустом месте формы. На панели свойств задайте свойства формы следующим образом:

- **FormBorderStyle** (Стиль границы формы): Fixed3D;

- **MaximizeBox** (Кнопка развертывания формы во весь экран): False;
- **MinimizeBox** (Кнопка свертывания формы на панель задач): False;
- **Text** (Текст надписи в заголовке формы): Таблица "Специальности".

На форме выделите надпись, щелкнув по ней **ЛКМ** и на панели свойств, задайте свойства надписи как показано ниже:

- **AutoSize** (Авторазмер): False;
- **Font** (Шрифт): Microsoft Sans Serif, размер 14;
- **ForeColor** (Цвет текста): Темно синий;
- **Text** (Текст надписи): Таблица "Специальности";
- **TextAlign** (Выравнивание текста): MiddleCenter.

После настройки всех вышеперечисленных свойств форма будет выглядеть следующим образом (рис. 18.6):

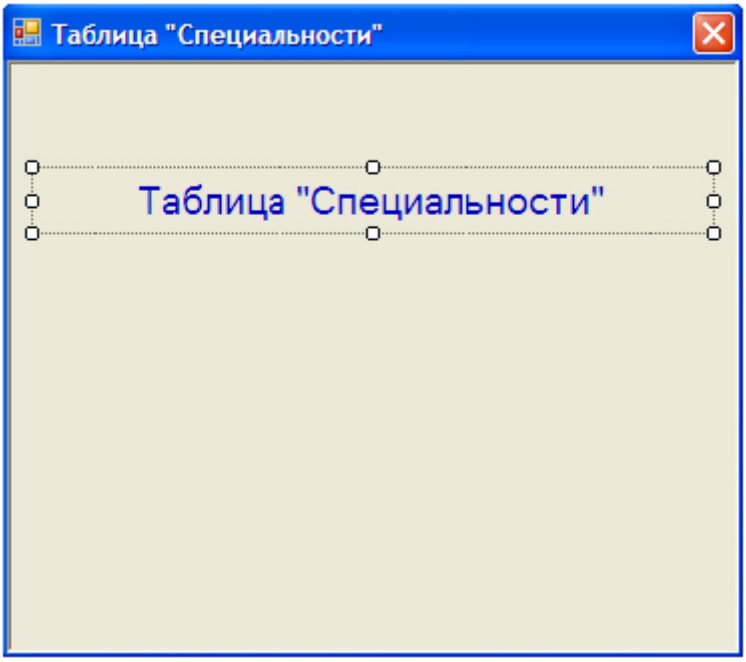


Рис. 18.6.

Теперь поместим на форму поля таблицы **"Специальности"**. Сначала откройте панель **"Источники данных"** (Data Sources), щелкнув по ее вкладке в правой части окна среды разработки (смотри рис. 18.6). На панели **"Источники данных"** отобразите поля таблицы **"Специальности"**, щелкнув по значку **"+"**, расположенному слева от имени таблицы (рис. 18.7).

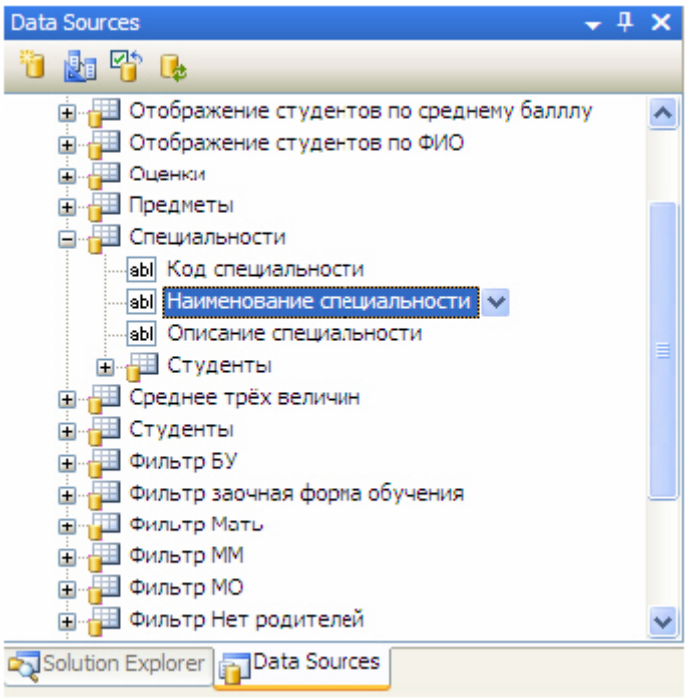


Рис. 18.7.

Панель **"Источники данных"** примет вид, представленный на рис. 18.7.

Замечание: Под полями таблицы специальности в виде подтаблицы располагается таблица **"Студенты"** (рис. 18.7). Подтаблица показывает, что таблица **"Студенты"** является вторичной по отношению к таблице специальности.

Замечание: При выделении, какого либо поля таблицы, оно будет отображаться в виде выпадающего списка (рис. 18.7), позволяющего выбирать объект, отображающий содержимое выделенного поля (рис. 18.8).

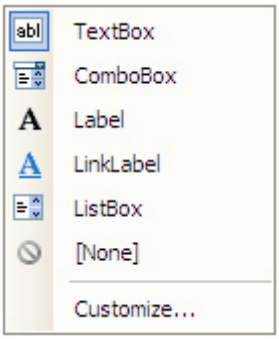
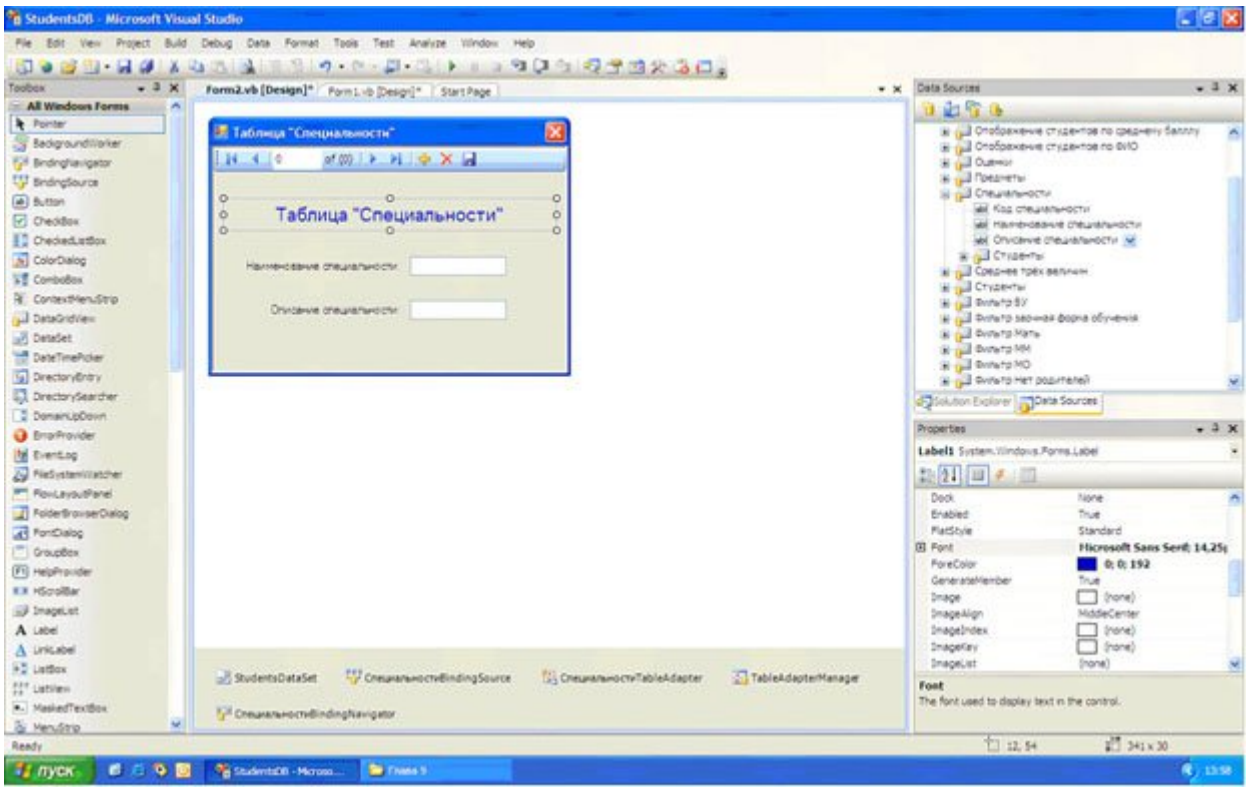


Рис. 18.8.

Для того чтобы поместить на новую форму поля таблицы их необходимо перетащить из панели **"Источники данных"** на форму. Из таблицы **"Специальности"** перетащите мышью на форму поля **"Наименование специальности"** и **"Описание специальности"**. Форма примет вид, представленный на рис. 18.9



увеличить изображение

Рис. 18.9.

Замечание: Мы не помещаем поле **"Код специальности"** на нашу форму, так как данное поле является первичным полем связи и заполняется автоматически. Конечный пользователь не должен видеть такие поля.

Замечание: Обратите внимание, что после перетаскивания полей с панели **"Источники данных"** на форму в верхней части формы появилась навигационная панель, а в нижней части рабочей области среды разработки появились пять невидимых объектов. Эти объекты предназначены для связи нашей формы с таблицей **"Специальности"**, расположенной на сервере. Рассмотрим функции этих объектов:

- **StudentDataSet** (Набор данных Student) - обеспечивает подключение формы к конкретной БД на сервере (в нашем случае это БД Students);
- **СпециальностиBindingSource** (Источник связи для таблицы **"Специальности"**) - обеспечивает подключение к конкретной таблице (в нашем случае к таблице специальности), а также позволяет управлять таблицей;
- **СпециальностиTableAdapter** (Адаптер таблиц для таблицы **"Специальности"**) - обеспечивает передачу данных с формы в таблицу и наоборот.
- **TableAdapterManager** (Менеджер адаптера таблиц) - управляет работой объекта **СпециальностиTableAdapter** ;
- **СпециальностиBindingNavigator** (Панель управления таблицей **"Специальности"**) - голубая панель с кнопками управления таблицей, расположенная в верхней части формы

Теперь нам необходимо проверить работоспособность новой формы. Для отображения формы **"Специальности"** ее необходимо подключить к главной кнопочной форме, а затем запустить проект и открыть форму **"Специальности"** при помощи кнопки на главной кнопочной форме.

Отобразите главную кнопочную форму в рабочей области среды разработки, щелкнув по вкладке **"Form1.vb [Design]"** в верхней части рабочей области. Для подключения новой формы **"Специальности"** к главной кнопочной форме дважды щелкните **ЛКМ** по кнопке **"Таблица "Специальности""**, расположенной на главной кнопочной форме (рис. 18.3). В появившемся окне кода формы в процедуре **"Button1_Click"** наберите команду

"Form2.Show()", предназначенную для открытия формы **"Таблица "Специальности""** (Form2), как это показано на рис. 18.10.

```
Public Class Form1

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        Form2.Show()
    End Sub
End Class
```

увеличить изображение

Рис. 18.10.

Теперь запустим проект, нажав на панели инструментов кнопку

На экране появится главная кнопочная форма. Для открытия формы, отображающей таблицу **"Специальности"** на главной кнопочной форме нажмите кнопку **"Таблица "Специальности""**. Появится форма с соответствующей таблицей (рис. 18.11).

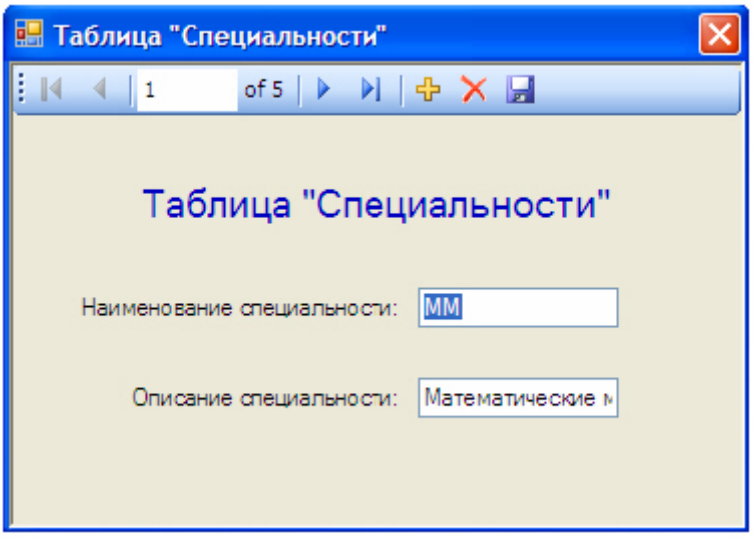


Рис. 18.11.

Проверьте работу панели навигации, расположенной в верхней части формы, понажимав на ней различные кнопки. Вернитесь в среду разработки, просто закрыв форму с таблицей **"Специальности"** и главную кнопочную форму.

Теперь создадим форму для просмотра таблицы предметы. Добавьте в проект новую форму. На форму добавьте надпись. Настройте свойства формы и надписи, как это было сделано для формы таблицы **"Специальности"**. Затем из таблицы **"Предметы"** на новую форму поместите поля **"Наименование предмета"** и **"Описание предмета"**. После выполнения всех вышеописанных действий форма для таблицы предметы будет выглядеть следующим образом (рис. 18.12):

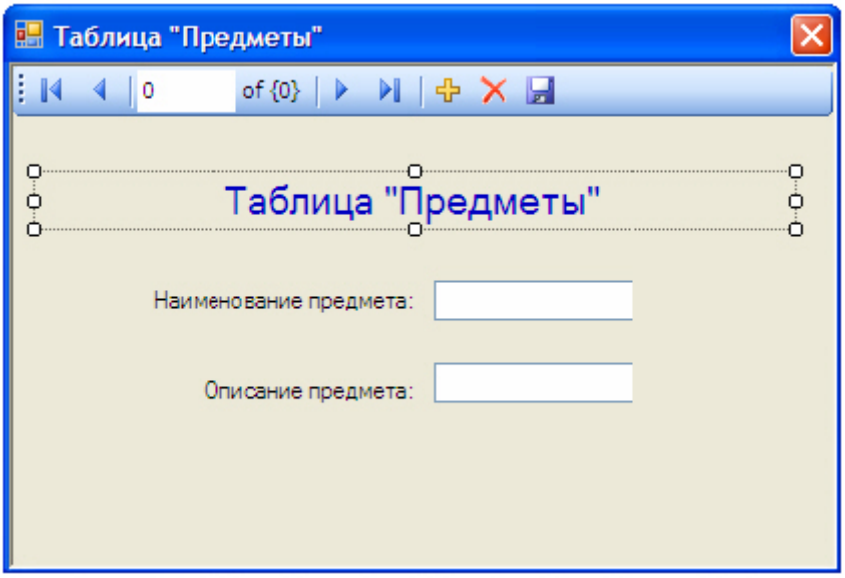


Рис. 18.12.

На главной кнопочной форме дважды щелкните **ЛКМ** по кнопке **"Таблица "Предметы""** и в появившемся окне кода в процедуре **"Button2_Click"** наберите **"Form3.Show()"** (рис. 18.13).

```
    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
        Form3.Show()
    End Sub
End Class
```

увеличить изображение

Рис. 18.13.

Проверим работу новой формы, отображающей таблицу **"Предметы"**. Запустите проект и на главной кнопочной форме нажмите кнопку **"Таблица "Предметы""**. Отобразится таблица предметы имеющая следующий вид (рис. 18.14):

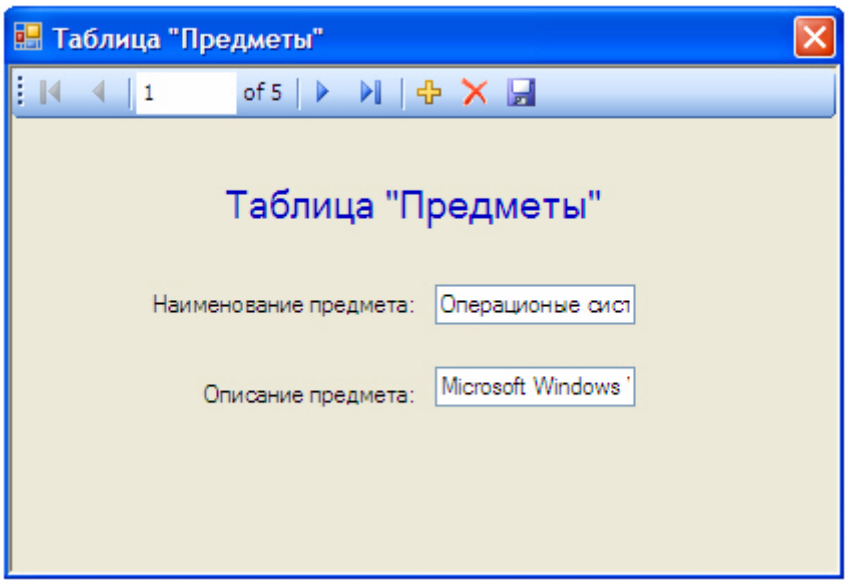


Рис. 18.14.

Проверьте работу панели навигации, нажатием на кнопки на данной панели в верхней части формы. Для возвращения в среду разработки закройте форму таблицы **"Предметы"** и главную кнопочную форму.

Теперь создадим простую ленточную форму для отображения таблицы **"Студенты"**. Для начала отобразите поля таблицы **"Студенты"** на панели **"Источники данных"**, щелкнув **ЛКМ** по знаку **"+"**, расположенному слева от названия таблицы. Отобразятся все поля таблицы **"Студенты"** (рис. 18.15).

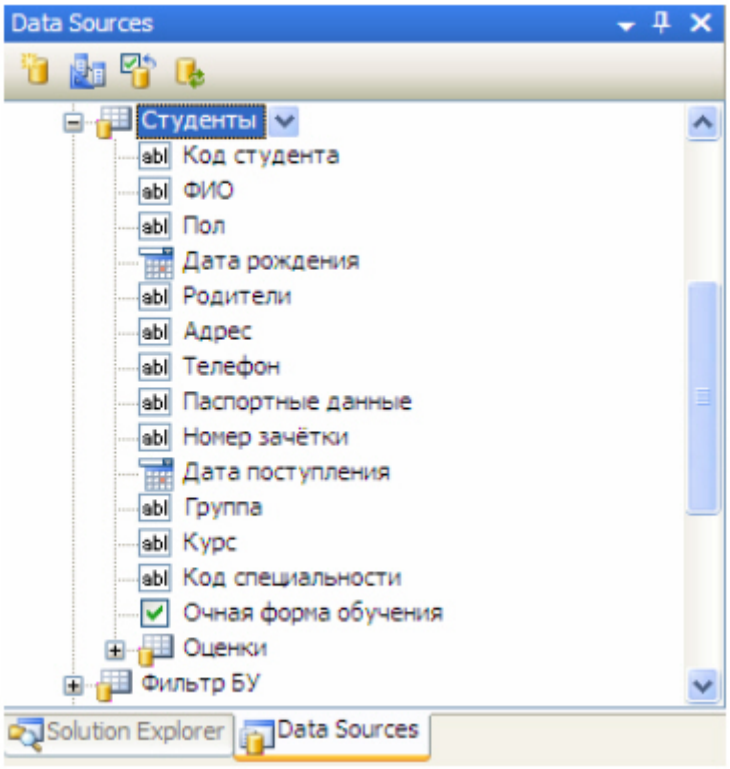


Рис. 18.15.

Замечание: Обратите внимание на тот факт, что поля **"Дата рождения"** и **"Дата поступления"** отображаются объектом **"Выбор даты" (DataPicker)**, так как данные поля содержат значения дат. Поле **"Очная форма обучения"** является логическим, следовательно, для его отображения используется объект **"Переключатель" (CheckBox)**. Остальные поля отображаются при помощи текстовых полей ввода (**TextBox**) (рис. 18.15).

Создайте новую форму и поместите в ее верхнюю часть надпись. Задайте заголовок формы как **"Таблица "Студенты""**. В верхнюю часть формы поместите надпись. В качестве текста надписи задайте тот же самый текст, что был задан в качестве заголовка формы. Настройте свойства формы и надписи, аналогично формам созданным ранее. На форму с панели **"Источники данных"** переместите все поля кроме поля **"Код студента"**. Так как данное поле является первичным полем связи. Новая форма примет вид (рис. 18.16):

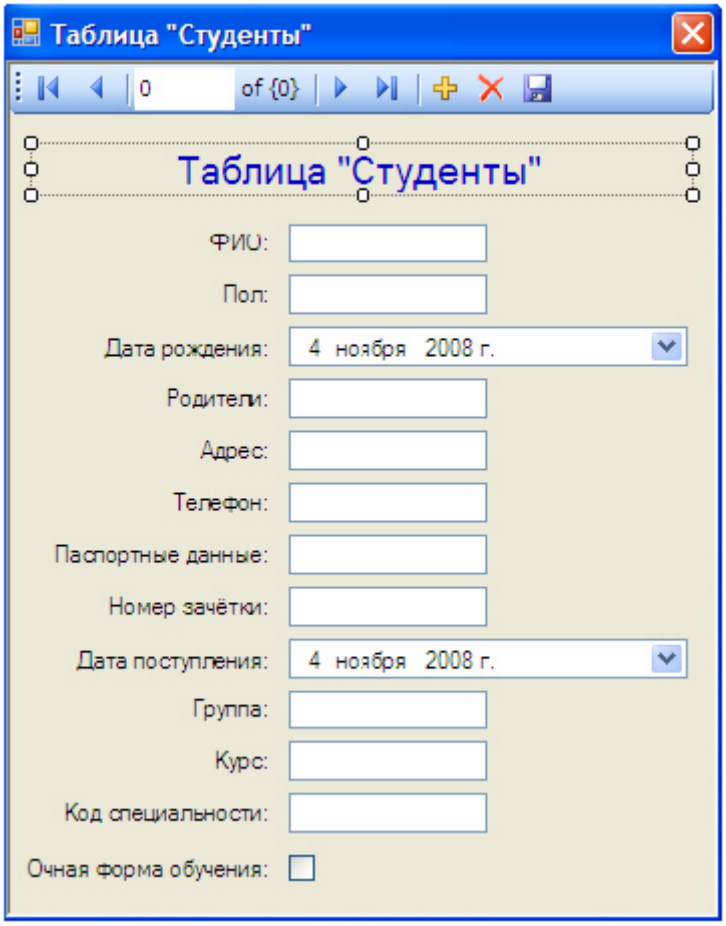


Рис. 18.16.

Обратите внимание на объекты, отображающие поля **"Дата рождения"**, **"Дата поступления"** и **"Очная форма обучения"**.

Подключим форму, отображающую таблицу **"Студенты"** к главной кнопочной форме. Отобразите главную кнопочную форму и на ней дважды щелкните **ЛКМ** по кнопке **"Таблица "Студенты""**. В появившемся окне кода, в процедуре **"Button3_Click"** наберите следующую команду для открытия формы таблицы **"Студенты"** - **"Form4.Show"** (рис. 18.17).

```
Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button3.Click
    Form4.Show()
End Sub
```

увеличить изображение

Рис. 18.17.

Теперь запустим проект. На экране появится главная кнопочная форма. Для открытия формы, отображающей таблицу **"Студенты"** на главной кнопочной форме нажмите кнопку **"Таблица "Студенты""**. Появится форма с соответствующей таблицей (рис. 18.18).

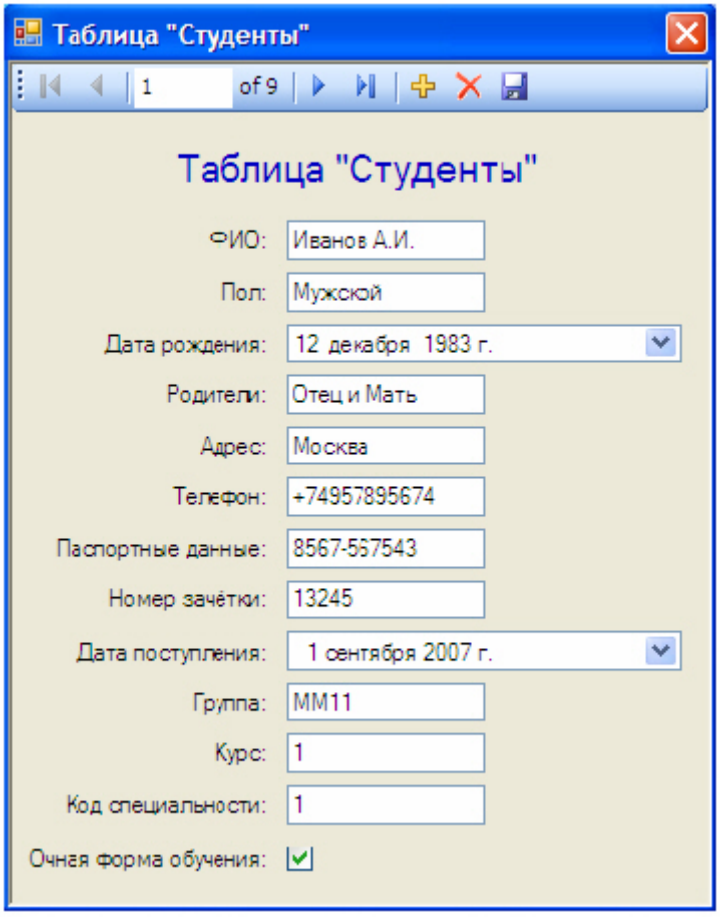


Рис. 18.18.

Проверьте работу формы нажатием кнопок на панели навигации, расположенной в верхней части формы. Закройте форму, отображающую таблицу **"Студенты"** и главную кнопочную форму.

Аналогичным образом создайте форму для отображения таблицы **"Оценки"**. Добавьте на новую форму надпись, добавьте на форму все поля из таблицы **"Оценки"** и настройте их свойства, как описано выше. В итоге, форма для отображения таблицы **"Оценки"** будет выглядеть следующим образом (рис. 18.19):

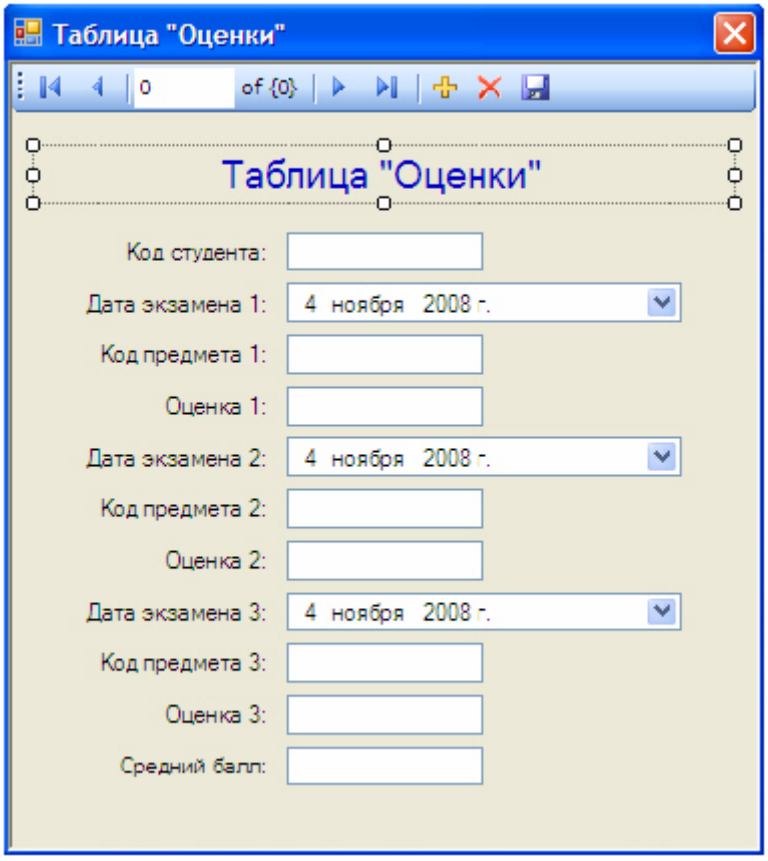


Рис. 18.19.

Подключите вновь созданную форму таблицы **"Оценки"** к главной кнопочной форме. Для этого отобразите главную кнопочную форму и на ней дважды щелкните **ЛКМ** по кнопке **"Таблица "Оценки""**. В появившемся окне с кодом, в процедуре **"Button4_Click"** наберите команду **"Form5.Show"** (рис. 18.20).

```
Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button4.Click
    Form5.Show()
End Sub
```

увеличить изображение

Рис. 18.20.

Проверьте работу формы таблицы **"Оценки"**, запустив проект, и на главной кнопочной форме нажмите кнопку **"Таблица "Оценки""**. Появится вновь созданная форма (рис. 18.21).

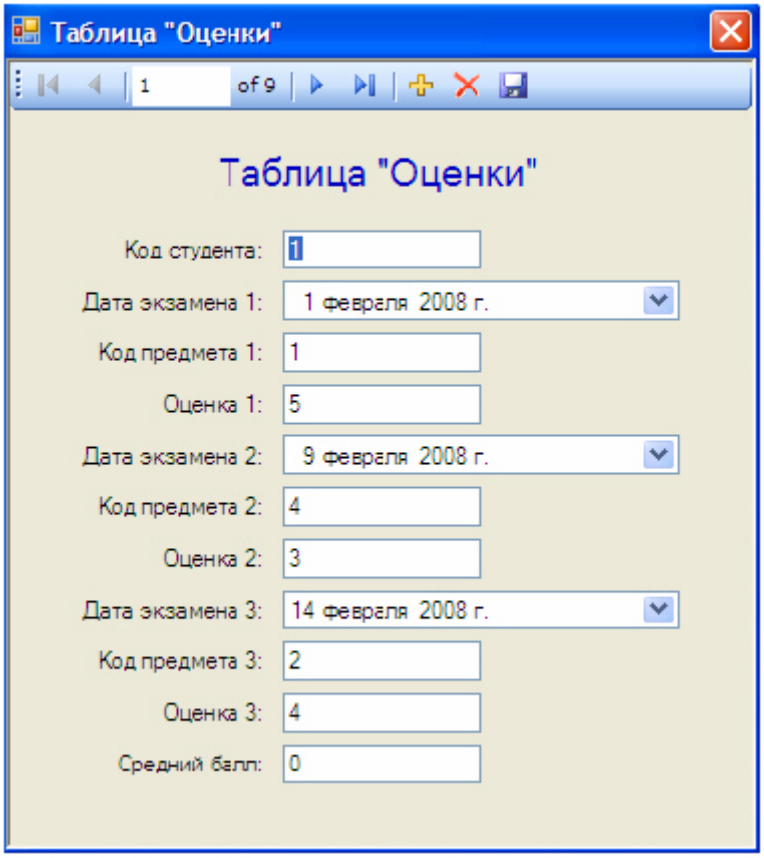


Рис. 18.21.

В заключение, откройте обозреватель проекта (**Solution Explorer**) щелкнув по его вкладке в правой части окна среды разработки. На данной панели должны отобразиться все выше созданные формы (рис. 18.22).

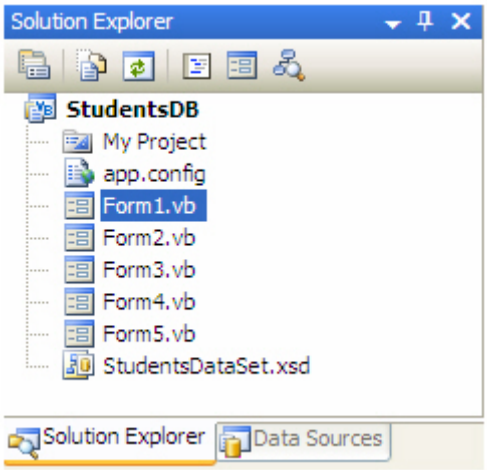


Рис. 18.22.

Контрольные вопросы

1. Настройки свойств формы.
2 Главная кнопочная форма.

Лабораторная работа № 11

«Создание сложных ленточных форм для работы с базами данных в Visual Studio 2012.»

Цель работы:

Научиться создавать сложные ленточные формы для работы с данными.

Модернизируем форму для таблицы **"Студенты"**. Сначала программно продублируем кнопки панели навигации, расположенной в верхней части формы. Откройте проект **"StudentsDB"** и отобразите форму таблицы студенты (**Form4**). В нижней части формы расположите семь кнопок, как это показано на рис. 20.1.

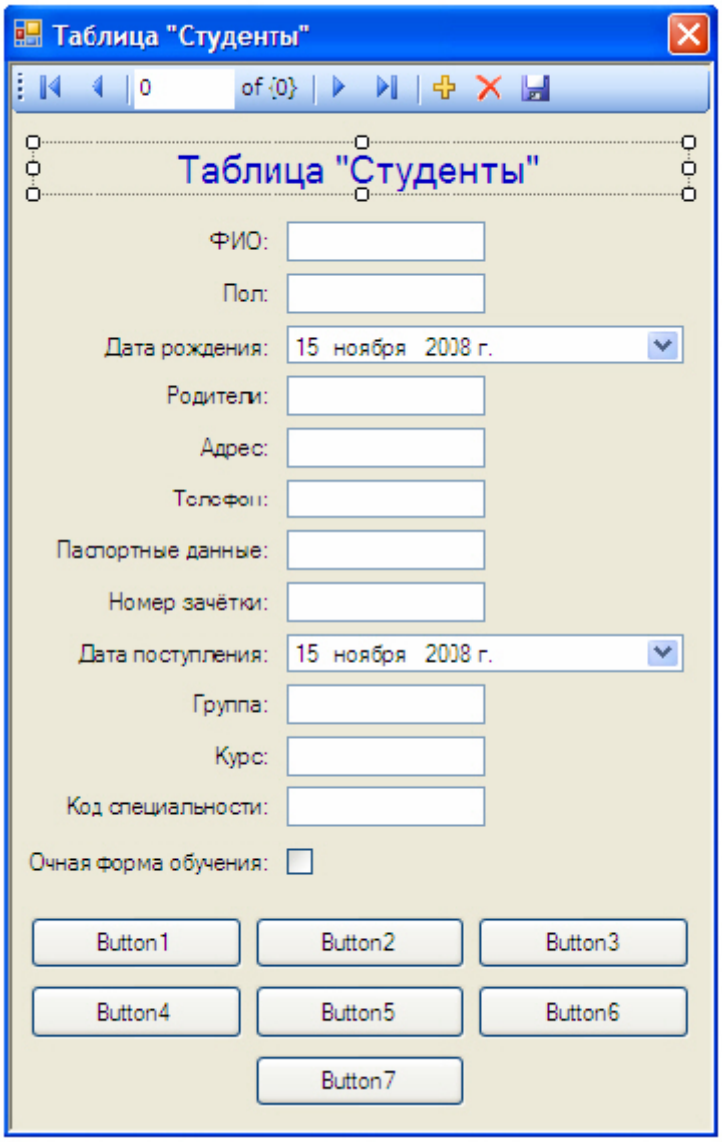


Рис. 20.1.

В качестве надписей на созданных кнопках (Свойство **"Caption"**) задайте как: **"Первая"**, **"Предыдущая"**, **"Добавить"**, **"Последняя"**, **"Следующая"**, **"Удалить"** и **"Сохранить"** (рис. 20.2).

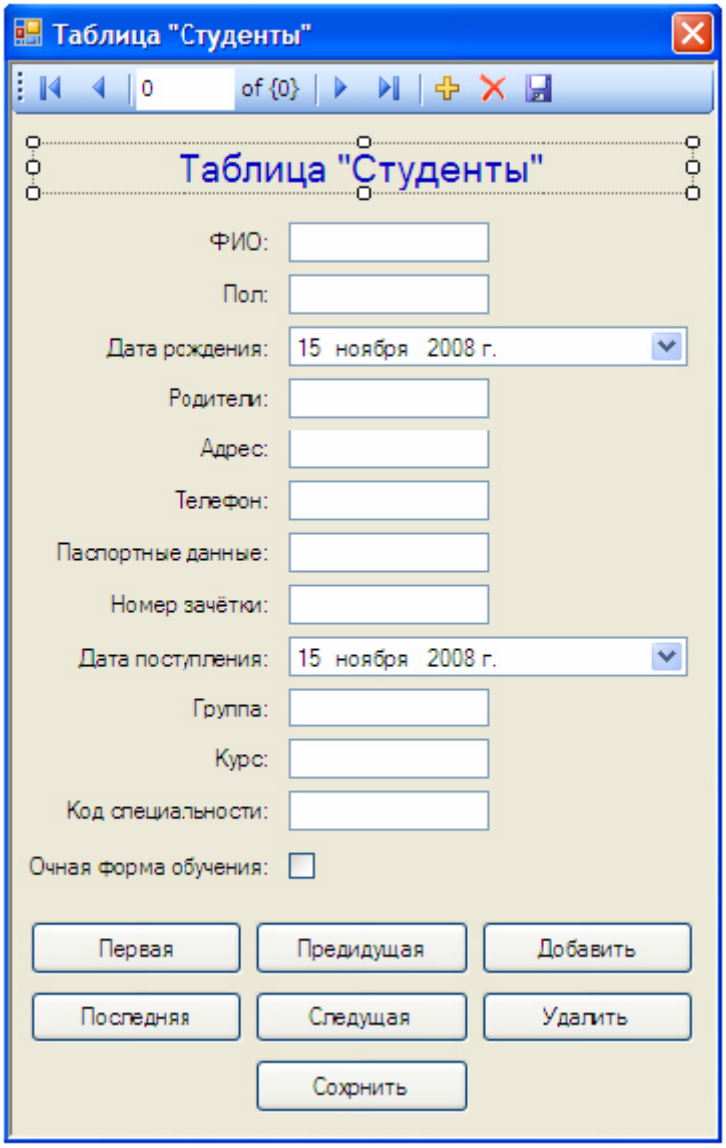


Рис. 20.2.

Дважды щелкните **ЛКМ** по кнопке **"Первая"** и в появившемся окне кода формы **"Form4"** в процедуре **"Button1_Click"** наберите команду для перехода к первой записи **"СтудентыBindingSource.MoveFirst()"** (рис. 20.3).

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    СтудентыBindingSource.MoveFirst()
End Sub
```

увеличить изображение

Рис. 20.3.

Дважды щелкните **ЛКМ** по кнопке **"Предыдущая"** и в появившемся окне кода формы **"Form4"** в процедуре **"Button2_Click"** наберите команду для перехода к предыдущей записи **"СтудентыBindingSource.MovePrevious()"** (рис. 20.4).

```
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
    СтудентыBindingSource.MovePrevious()
End Sub
```

увеличить изображение

Рис. 20.4.

Дважды щелкните **ЛКМ** по кнопке **"Добавить"** и в появившемся окне кода формы **"Form4"** в процедуре **"Button3_Click"** наберите команду для добавления новой записи **"СтудентыBindingSource.AddNew()"** (рис. 20.5).

```
Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button3.Click
    СтудентыBindingSource.AddNew()
End Sub
```

увеличить изображение

Рис. 20.5.

Дважды щелкните **ЛКМ** по кнопке **"Последняя"** и в появившемся окне кода формы **"Form4"** в процедуре **"Button4_Click"** наберите команду для перехода к последней записи **"СтудентыBindingSource.MoveLast()"** (рис. 20.6).

```
Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button4.Click
    СтудентыBindingSource.MoveLast()
End Sub
```

увеличить изображение

Рис. 20.6.

Дважды щелкните **ЛКМ** по кнопке **"Следующая"** и в появившемся окне кода формы **"Form4"** в процедуре **"Button5_Click"** наберите команду для перехода к следующей записи **"СтудентыBindingSource.MoveNext()"** (рис. 20.7).

```
Private Sub Button5_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button5.Click
    СтудентыBindingSource.MoveNext()
End Sub
```

увеличить изображение

Рис. 20.7.

Дважды щелкните **ЛКМ** по кнопке **"Удалить"** и в появившемся окне кода формы **"Form4"** в процедуре **"Button6_Click"** наберите команду для удаления текущей записи **"СтудентыBindingSource.RemoveCurrent()"** (рис. 20.8).

```
Private Sub Button6_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button6.Click
    СтудентыBindingSource.RemoveCurrent()
End Sub
```

увеличить изображение

Рис. 20.8.

Дважды щелкните **ЛКМ** по кнопке **"Сохранить"** и в появившемся окне кода формы **"Form4"** в процедуре **"Button7_Click"** наберите команду для сохранения изменений, отображенную на рис. 20.9.

```
Private Sub Button7_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button7.Click
    Me.Validate()
    Me.СтудентыBindingSource.EndEdit()
    Me.TableAdapterManager.UpdateAll(Me.StudentsDataSet)
End Sub
```

увеличить изображение

Рис. 20.9.

Рассмотрим последнюю процедуру более подробно. Она содержит следующие команды:

- Me.Validate() - проверяет введенные в поля данные на соответствие типам данных полей;
- Me.СтудентыBindingSource.EndEdit() - закрывает подключение с сервером;
- Me.TableAdapterManager.UpdateAll(Me.StudentsDataSet) - обновляет данные на сервере.

Для проверки работы созданных кнопок запустите проект откройте форму **"Таблица "Студенты""** и нажмите каждую из кнопок.

Теперь изменим объекты, отображающие поля для более удобного ввода информации. Для начала удалите текстовые поля ввода (**TextBox**), отображающие следующие поля таблицы **"Студенты"**: **"Пол"**, **"Родители"**, **"Телефон"**, **"Паспортные данные"**,

"Номер зачетки", **"Курс"** и **"Код специальности"**. После удаления, перечисленных полей форма, отображающая таблицу **"Студенты"** примет следующий вид (рис. 20.10):

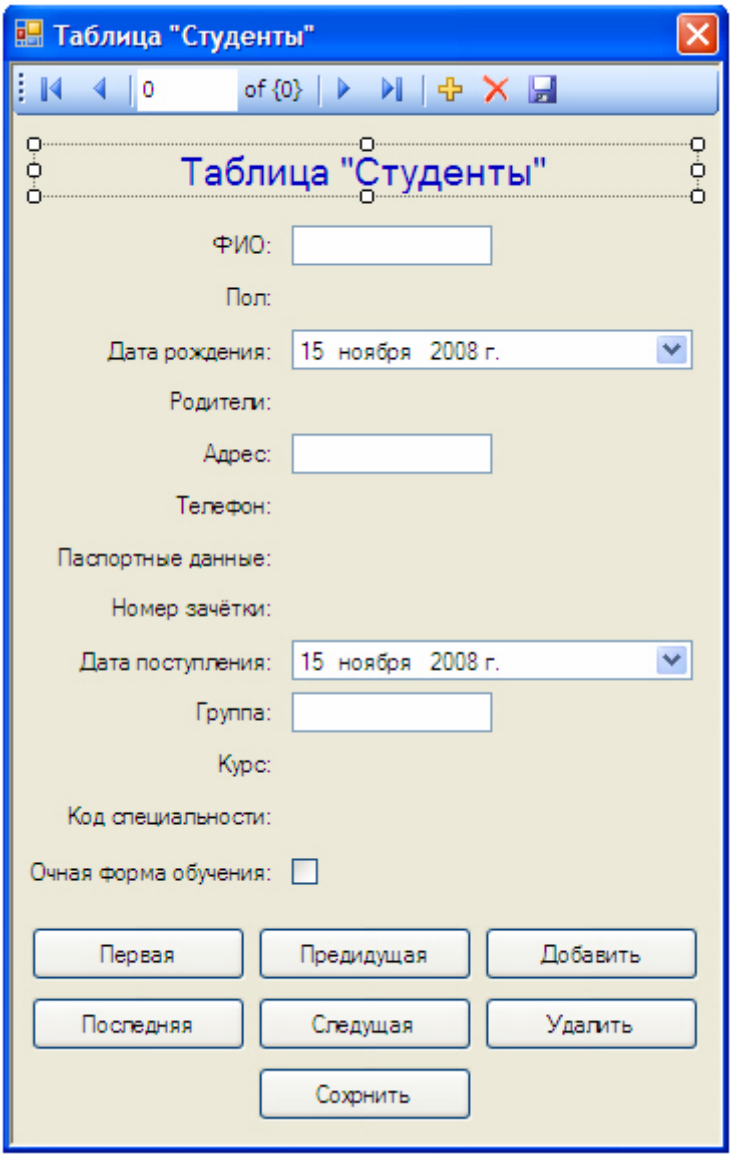
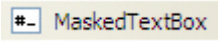


Рис. 20.10.

Для отображения полей **"Телефон"**, **"Паспортные данные"** и **"Номер зачетки"** будем использовать текстовые поля ввода по маске (**MaskedTextBox**). Объект текстовое поле ввода по маске отсутствует в выпадающем списке объектов для отображения полей в окне **"Источники данных"**, поэтому будем создавать данные объекты при помощи панели объектов (**Toolbox**), а затем подключать их к соответствующим полям вручную. Для создания текстовых полей ввода по маске на панели объектов используется кнопка



Создайте текстовые поля ввода по маске справа от надписей **"Телефон"**, **"Паспортные данные"** и **"Номер зачетки"**, как это показано на рис. 20.11.

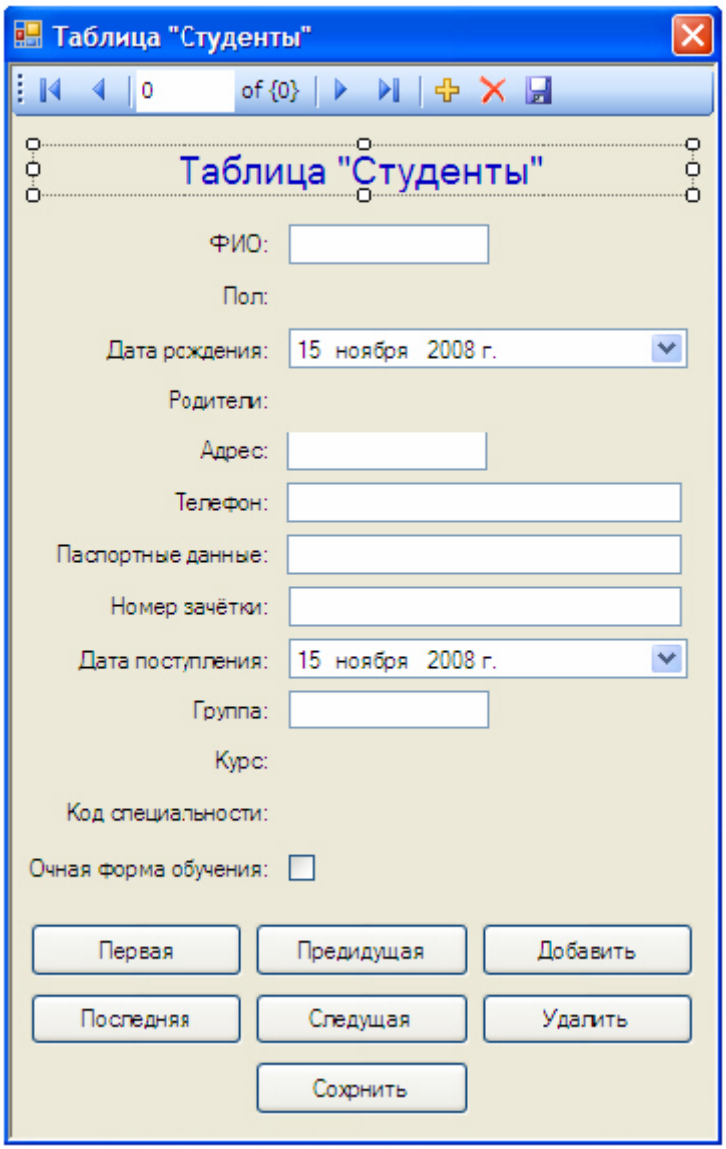


Рис. 20.11.

Теперь у созданных объектов настроим маски ввода. Начнем с объекта, отображающего номер зачетки. На форме выделите соответствующее полю **"Номер зачетки"** текстовое поле ввода по маске. Для задания маски в меню действий с объектом выберите пункт **"Set Mask…"** (Установить маску…) (рис. 20.12).

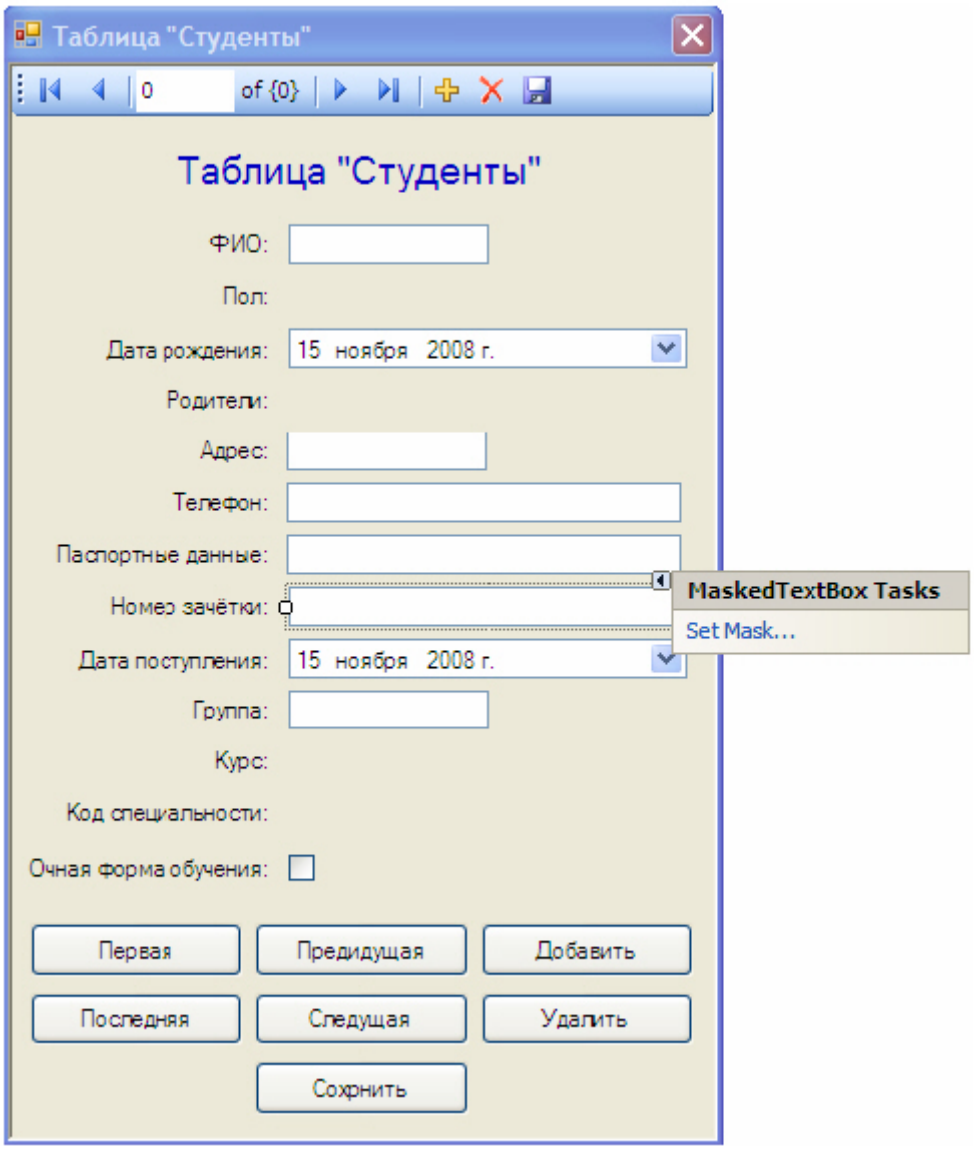


Рис. 20.12.

Замечание: Для отображения меню действий в верхнем правом углу объекта необходимо нажать кнопку

(рис. 20.12).

После выбора пункта **"Set Mask…"** на экране появится окно задания маски **"Input Mask"** (Введите маску) (рис. 20.13).

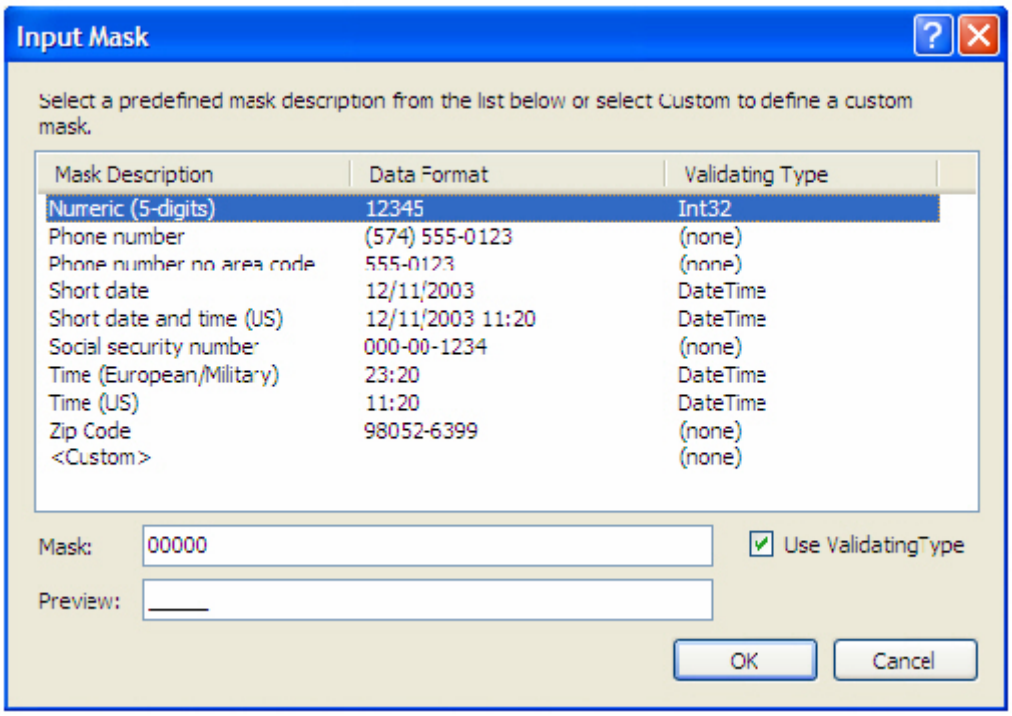


Рис. 20.13.

В окне **"Input Mask"** выберите маску **"Numeric (5-digits)"** (Числовое (5-цифр)) и нажмите кнопку **"Ok"** (рис. 20.13).

Для текстового поля ввода по маске для поля **"Паспортные данные"** задайте маску как показано на рис. 20.14.

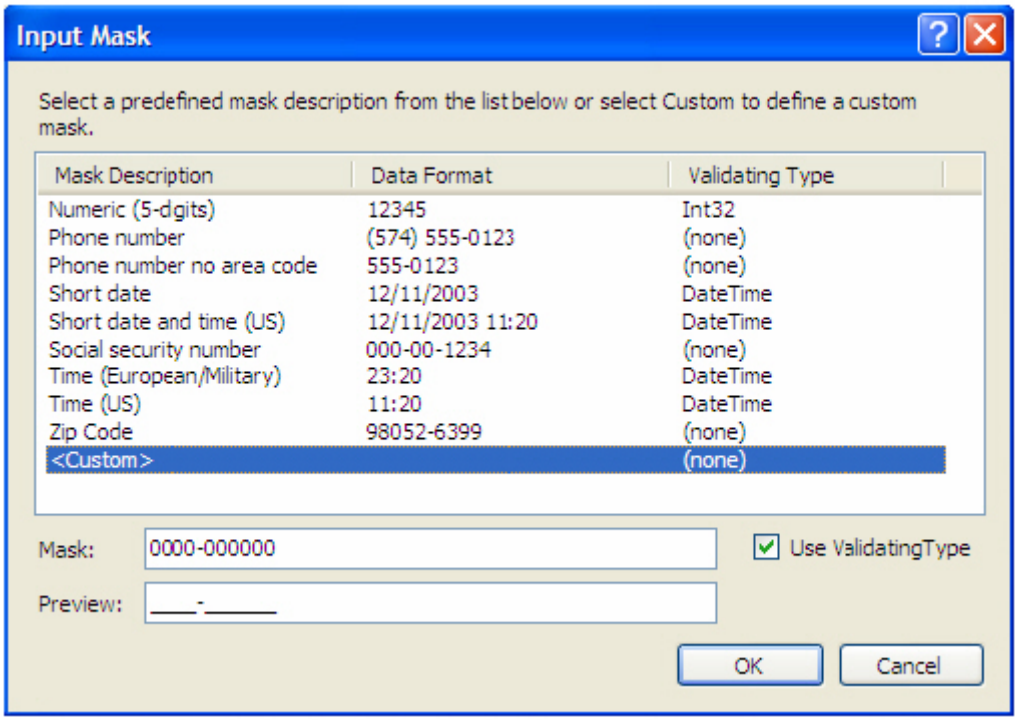


Рис. 20.14.

Замечание: Обратите внимание, что паспортные данные отображаются как четыре числа, тире, шесть чисел. То есть в поле **"Mask"** (Маска) нужно задать **"0000-000000"**. Знак **"0"** обозначает цифру. В поле **"Preview"** (Предварительный просмотр) отображается вид текстового поля ввода по маске на форме.

После определения маски для поля **"Паспортные данные"** в окне **"Input Mask"** нажмите кнопку **"Ok"**.
Теперь зададим маску для текстового поля ввода по маске отображающего поле **"Телефон"**. Задайте маску как показано на рис. 20.15.

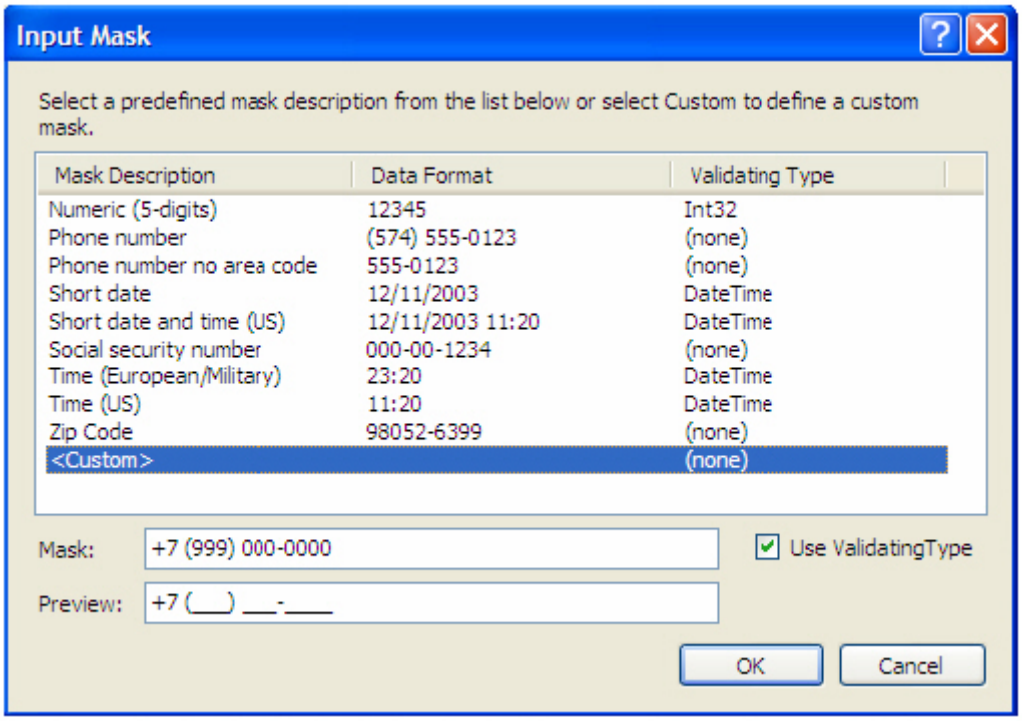


Рис. 20.15.
Теперь нам необходимо подключить созданные текстовые поля ввода по маске к соответствующим полям. Для этого с панели **"Источники данных" (DataSources)** перетащите поле **"Номер зачетки"** на текстовое поле ввода по маске, расположенное справа от надписи **"Номер зачетки"**. Проделайте такую же операцию с полями **"Паспортные данные"** и **"Телефон"**, перетащив их на соответствующие им текстовые поля ввода по маске.
На этом мы заканчиваем работу с текстовыми полями ввода по маске и переходим к отображению поля **"Курс"** при помощи числового счетчика (объект **NumericUpDown**). Для этого, на панели **"Источники данных"** нажмите кнопку, расположенную справа от поля **"Курс"** и в выпадающем списке выберите объект для отображения данного поля как **"NumericUpDown"** (рис. 20.16).

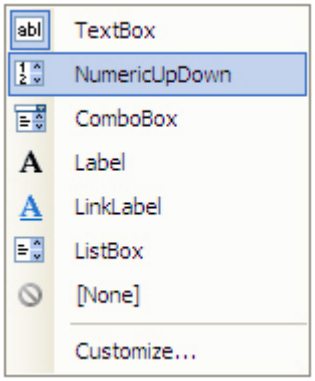


Рис. 20.16.
Затем перетащите поле на форму мышью, расположив, его справа от надписи **"Курс"**.

Замечание: После перетаскивания поля **"Курс"** на форму слева от него появится еще одна надпись **"Курс"**. Удалите ее, щелкнув по ней **ЛКМ**, а затем нажав кнопку **"Delete"** на клавиатуре.
Отобразим поля **"Пол"** и **"Родители"** в виде выпадающих списков (Объект **ComboBox**). Для этого, на панели **"Источники данных"** нажмите кнопку, расположенную справа от поля **"Пол"** и в выпадающем списке выберите объект для отображения данного поля как **"ComboBox"** (рис. 20.17).

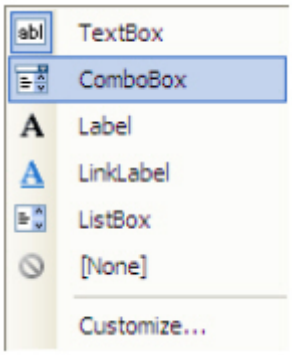


Рис. 20.17.
Такую же операцию проделайте с полем **"Родители"**. Затем перетащите мышью поля на форму, расположив их напротив соответствующих надписей. Удалите лишние надписи.
Теперь заполним выпадающие списки. Выделите выпадающий список, отображающий поле **"Пол"**. На панели свойств (**Properties**) и нажмите кнопку в свойстве **"Items"** (Элементы списка). Появится окно **"String Collection Editor"** (Редактор строковых коллекций) (рис. 20.18).

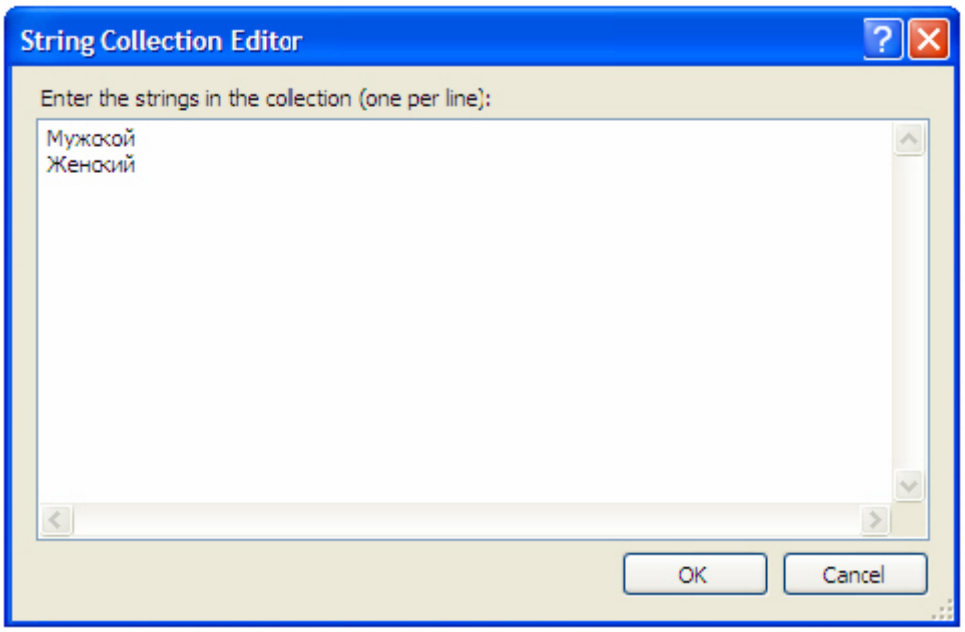


Рис. 20.18.
В появившемся окне в отдельных строках наберите элементы выпадающего списка: **"Мужской"** и **"Женский"** (рис. 20.18). Затем нажмите кнопку **"Ok"**.
Для выпадающего списка, отображающего поле **"Родители"**, проделайте аналогичную операцию, только в качестве пунктов списка задайте: **"Отец и Мать"**, **"Мать"**, **"Отец"** и **"Нет"** (рис. 20.19).

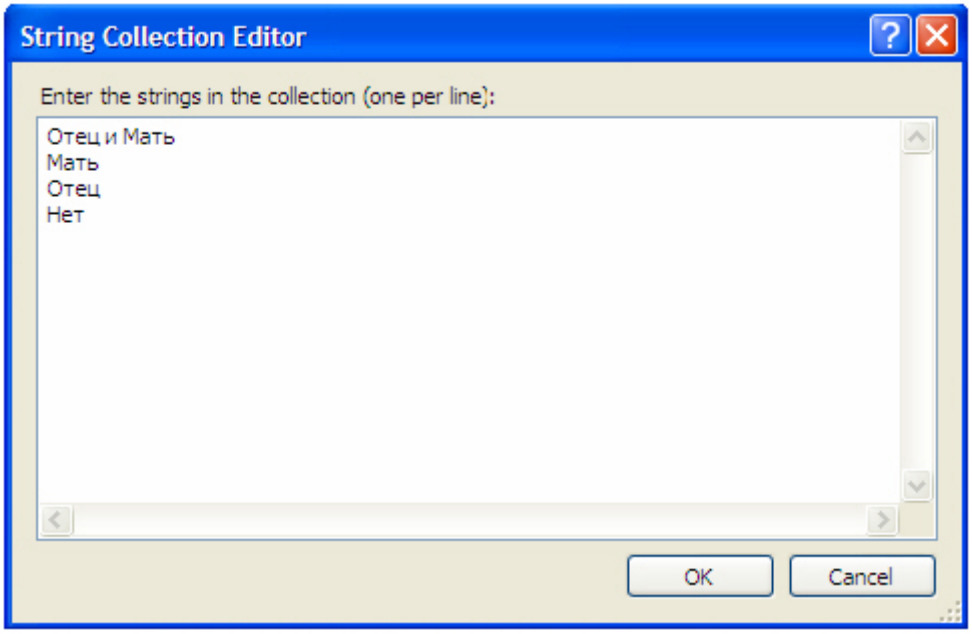
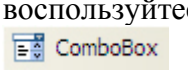


Рис. 20.19.

В заключение отобразим вместо поля **"Код специальности"** специальность соответствующую заданному коду, при помощи выпадающего списка. При этом сам выпадающий список будет заполнен специальностями из таблицы **"Специальности"** и при выборе специальности ее код будет автоматически подставляться в поле **"Код специальности"** таблицы **"Студенты"**.

Поместите справа от надписи **"Код специальности"**, неподключенный ни к каким полям выпадающий список. Для создания выпадающего списка на панели объектов воспользуйтесь кнопкой



После создание выпадающего списка подключим его к полю **"Код специальности"** из таблицы **"Студенты"** и настроим заполнение списка значениями поля **"Наименование специальности"** из таблицы студенты. Для этого выделите вновь созданный выпадающий список, отобразите меню действий и в меню действий включите опцию **"Use data bound items"** (Использовать связанные с данными элементы списка) (рис. 20.20).

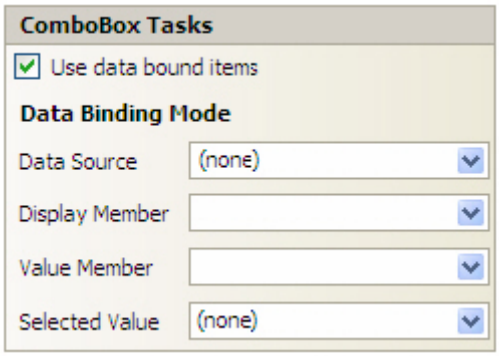


Рис. 20.20.

В панели действий под опцией **"Use data bound items"** расположены следующие параметры:

- **Data Source (Источник данных)** - определяет таблицу или запрос из которого заполняется список;

- **Display Member (Член отображения)** - определяет поле значениями которого заполняется список;
- **Value Member (Член значений)** - определяет значения какого поля подставляются в связанное с выпадающим списком поле;
- **Selected Value (Выбранное значение)** - определяет связанное с выпадающим списком поле.

Для изменения параметров необходимо нажать кнопку

внутрии поля параметра. Появится древовидная структура выбора источника данных (рис. 20.21).

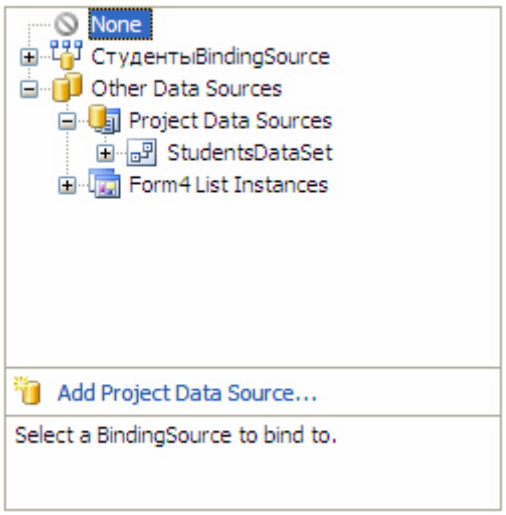


Рис. 20.21.

В нашем случае зададим вышеперечисленные параметры следующим образом:

- Параметр **"DataSource"** задайте как **"Other Data Sources\Project Data Sources\StudentsDataSet\Специальности"** ;
- Параметр **"DataMember"** задайте как **"Наименование специальности"** ;
- Параметр **"Value Member"** задайте как **"Код специальности"** ;
- Параметр **"Selected Value"** задайте как **"СтудентыBindingSource\Код специальности"**.

После задания всех вышеперечисленных параметров панель действий выпадающего списка примет вид (рис. 20.22):

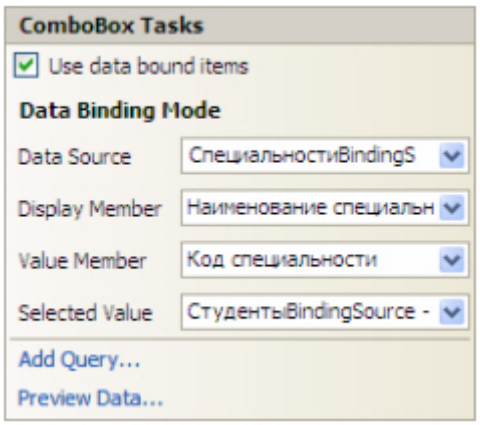
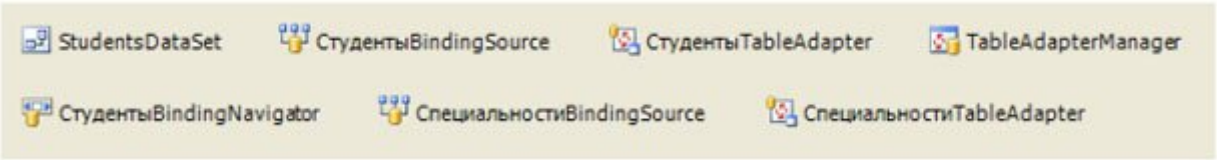


Рис. 20.22.

Обратите внимание на то, что на панели невидимых объектов, расположенной в нижней части рабочей области среды разработки, появилось два новых объекта: **"СпециальностиBindingSource"** и **"СпециальностиTableAdapter"** (рис. 20.23).



увеличить изображение

Рис. 20.23.

Данные объекты предназначены для заполнения выпадающего списка значениями поля **"Наименование специальности"** таблицы **"Специальности"**.

После всех вышеперечисленных действий форма, отображающая таблицу **"Студенты"** примет вид, представленный на рис. 20.24.

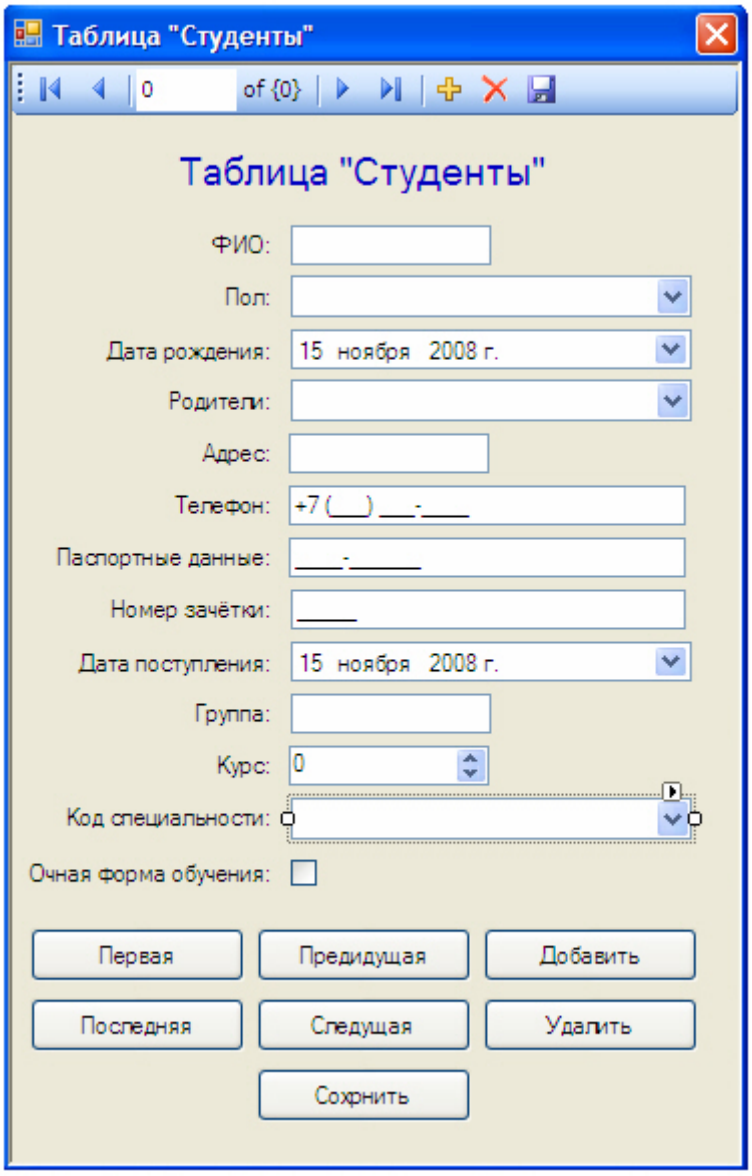


Рис. 20.24.

Проверим работу формы, отображающей таблицу **"Студенты"**. Запустите проект и на главной кнопочной форме нажмите кнопку **"Таблица "Студенты""**. Появится форма, имеющая следующий вид (рис. 20.25):

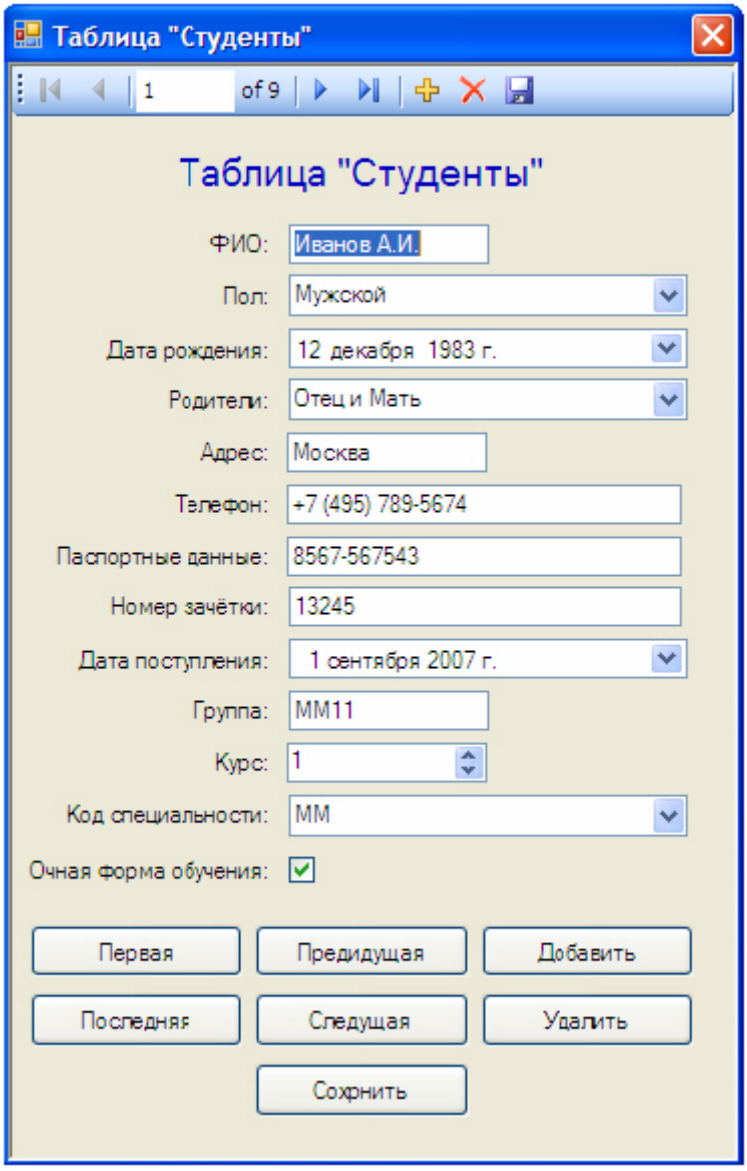


Рис. 20.25.

На этом мы заканчиваем работу с формой, отображающей таблицу **"Студенты"** и переходим к реализации вычислимых полей. Для этого рассмотрим форму, отображающую таблицу **"Оценки" (Form5)**. Рассмотрим вычисление поля **"Средний балл"** на основе среднего трех полей:

Отобразите форму для таблицы **"Оценки"**, щелкнув **ЛКМ** по ее вкладке в верхней части рабочей области среды разработки. На форму, справа от поля **"Средний балл"** поместите кнопку (рис. 20.26).

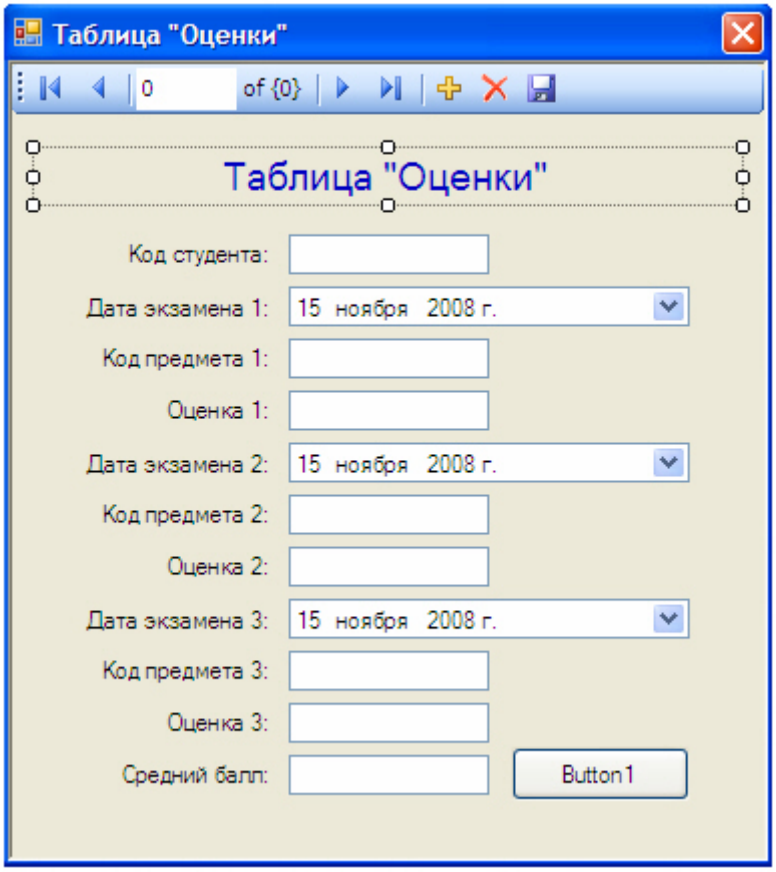


Рис. 20.26.

Задайте свойство **"Text"** у вновь созданной кнопки как **"Вычислить"** (рис. 20.27).

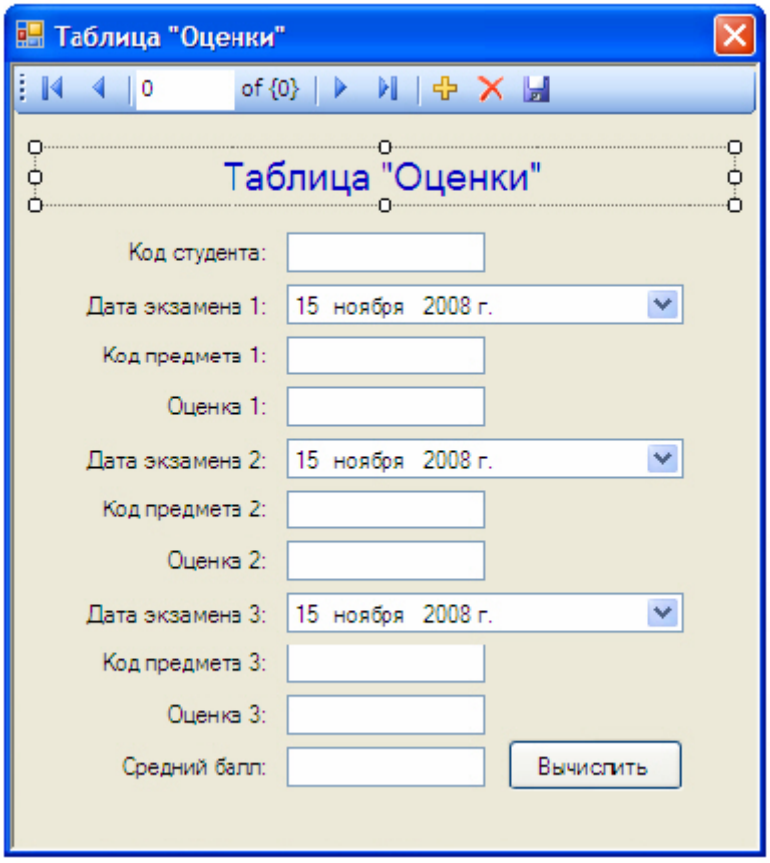


Рис. 20.27.
Теперь дважды щелкните **ЛКМ** по кнопке **"Вычислить"** и в появившемся коде процедуры **"Button1_Click"** наберите код, представленный на рис. 20.28, вычисляющий среднее оценок.

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    Средний_баллTextBox.Text = (Val(Оценка_1TextBox.Text) + Val(Оценка_2TextBox.Text) + Val(Оценка_3TextBox.Text)) / 3
End Sub
```

увеличить изображение
Рис. 20.28.
Теперь проверим, как работает наша вновь созданная кнопка для вычисления поля **"Средний балл"**. Запустите проект и на главной кнопочной форме нажмите кнопку **"Таблица "Оценки""**. Появится форма, отображающая таблицу **"Оценки"**, на форме нажмите кнопку **"Вычислить"**. Будет вычислен средний балл по оценкам. Если нажать кнопку сохранения на панели инструментов формы

то средний балл будет сохранен в таблицу **"Оценки"** (рис. 20.29).

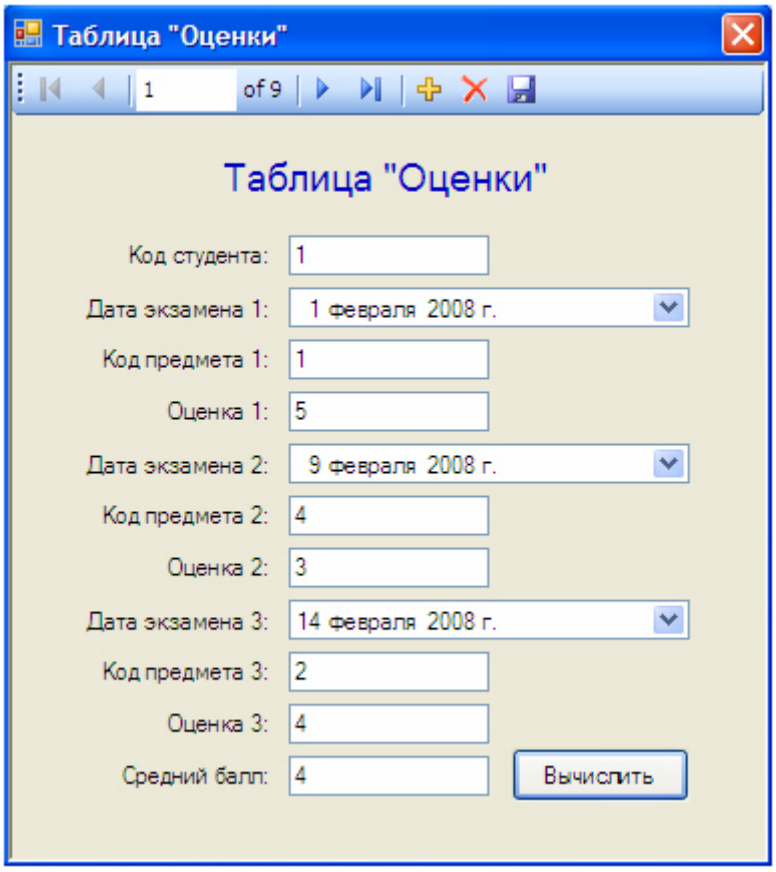


Рис. 20.29.
На этом мы заканчиваем рассмотрение ленточных форм и переходим к рассмотрению табличных форм.
Контрольные вопросы
1. Как настроить маски ввода.
2. Как создать выпадающий список.

Лабораторная работа № 12

«Создание табличных форм для работы с базами данных в Visual Studio 2012.»

Цель работы:

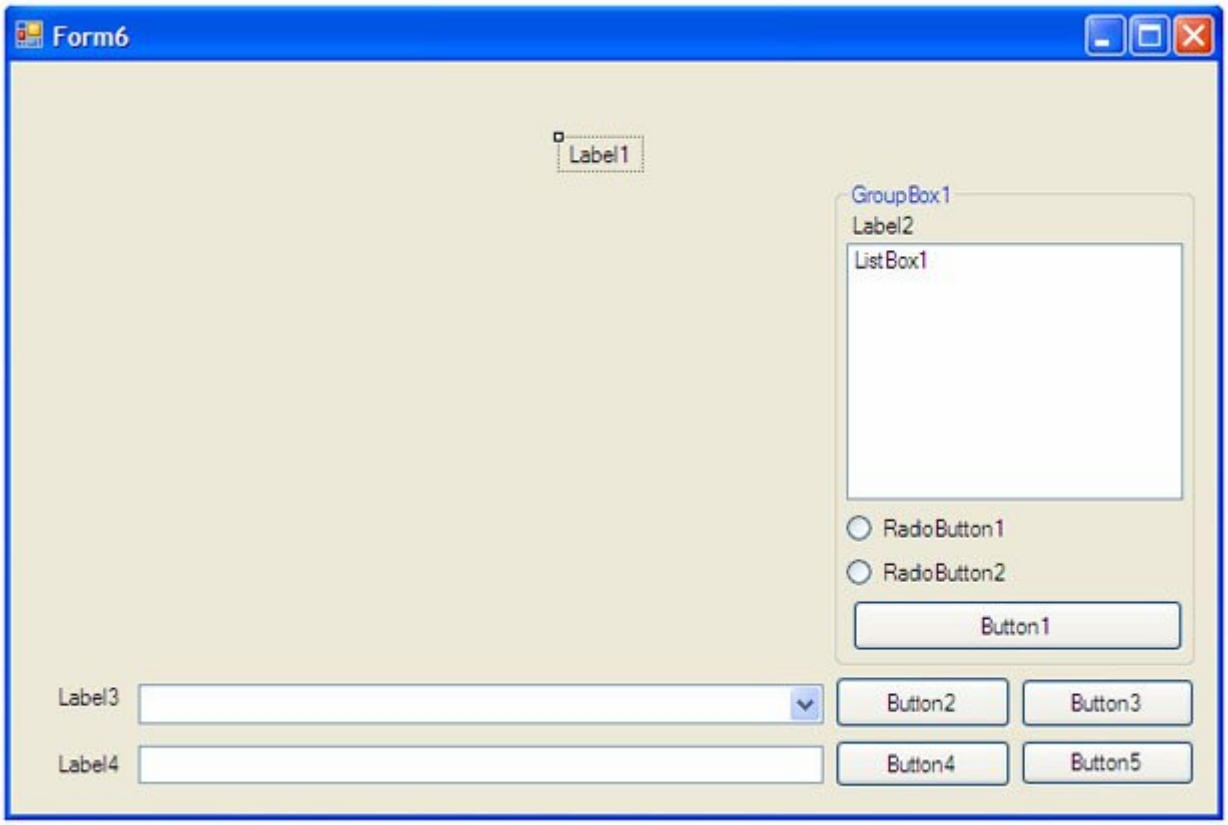
Научиться создавать табличные формы.

Перейдем теперь к созданию табличных форм для отображения данных. В данной главе также затрагиваются вопросы фильтрации и сортировки данных, а также реализуется поиск информации в таблице.

Рассмотрим создание табличной формы на примере формы, отображающей таблицу **"Студенты"**. Добавьте в проект новую форму и на нее поместите следующие объекты:

- четыре надписи (**Label**),
- пять кнопок (**Button**),
- выпадающий список (**ComboBox**),
- текстовое поле ввода (**TextBox**),
- группирующую рамку (**GroupBox**),
- список (**ListBox**),
- два переключателя (**RadioButton**).

Расположите объекты как показано на рис. 22.1.



увеличить изображение

Рис. 22.1.

Замечание: Для создания объекта группирующая рамка используется кнопка ### на панели объектов (**Toolbox**), а для создания переключателя - кнопка ###.

Добавим на форму таблицу для отображения данных (**DataGridView**) из таблицы **"Студенты"**. Для этого на панели **"Источники данных" (Data Sources)**, нажмите кнопку

расположенную справа от таблицы **"Студенты"**. В появившемся списке объектов для отображения всей таблицы выберите **"DataGridView"** (рис. 22.2).

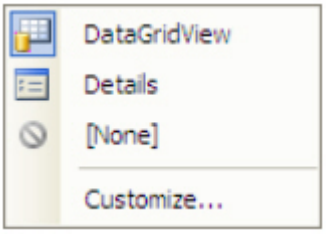
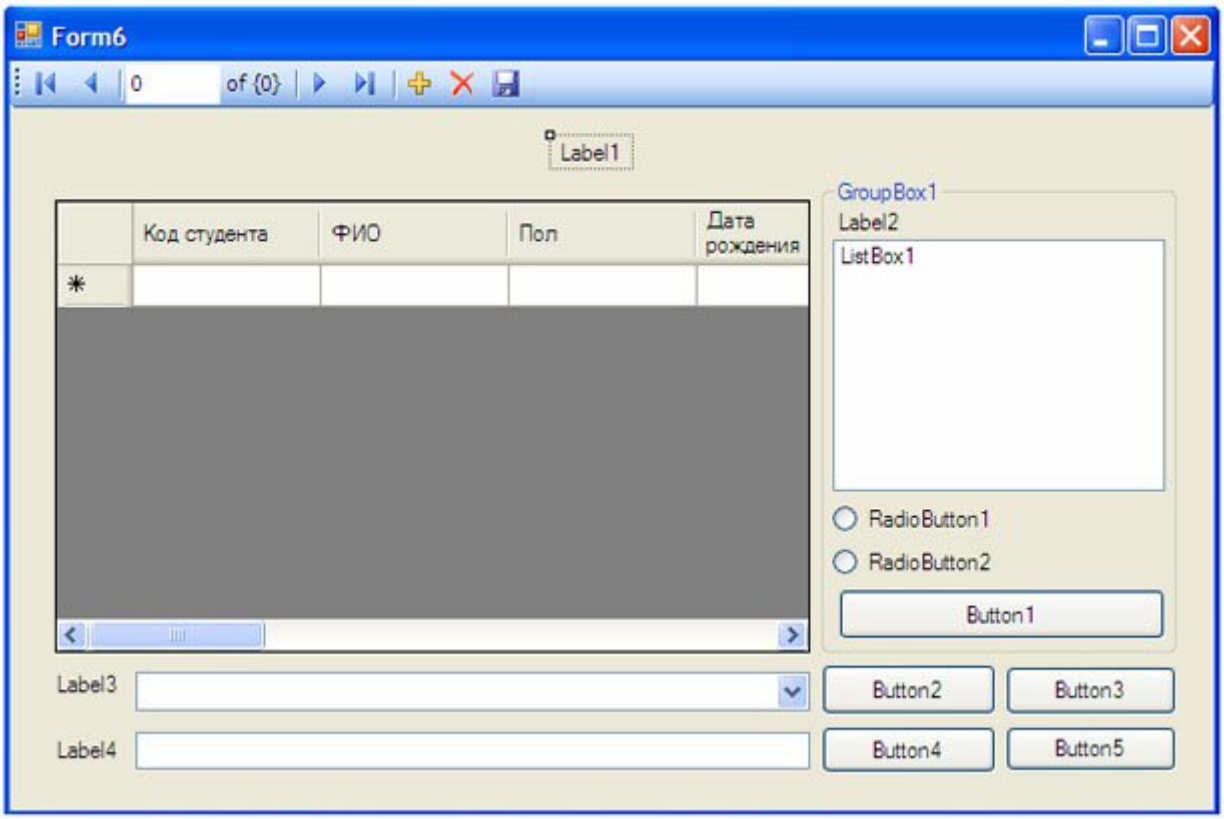


Рис. 22.2.

Перетащите таблицу **"Студенты"** из панели **"Источники данных"** на форму. Форма примет следующий вид (рис. 22.3):



увеличить изображение

Рис. 22.3.

Обратите внимание на то, что на форме появилась таблица для отображения данных, подключенная к таблице **"Студенты"**. Также появились объекты связи и панель навигации (рис. 22.4).

увеличить изображение

Рис. 22.4.

Теперь перейдем к настройке свойств объектов. Начнем с настройки свойств формы. Задайте свойства формы следующим образом:

- **FormBorderStyle (Стиль границы формы)**: Fixed3D;
- **MaximizeBox (Кнопка развертывания формы во весь экран)**: False;
- **MinimizeBox (Кнопка свертывания формы на панель задач)**: False;
- **Text (Текст надписи в заголовке формы)**: Таблица "Студенты" (Табличный вид).

Задайте свойства надписей (**Label1**, **Label2**, **Label3** и **Label4**) как:

- **AutoSize (Авторазмер)**: False;

- **Text (Текст надписи)**: "Таблица "Студенты" (Табличный вид)", "Поле для сортировки", "ФИО:" и "Критерий" (Соответственно для Label1, Label2, Label3 и Label4).

Для надписи **Label1** задайте:

- **Font (Шрифт)**: Microsoft Sans Serif, размер 14;
- **ForeColor (Цвет текста)**: Темно синий;
- **TextAlign (Выравнивание текста)**: MiddleCenter.

Задайте надписи на кнопках как: **"Сортировать"**, **"Фильтровать"**, **"Показать все"**, **"Найти"** и **"Закрыть"** (Соответственно для кнопок **Button1**, **Button2**, **Button3**, **Button4** и **Button5**). Для того чтобы нельзя было произвести сортировку не выбрав поля изначально заблокируем кнопку **"Сортировать" (Button1)**.

У группирующей рамки задайте заголовок (Свойство **Text**) равным **"Сортировка"**. У переключателей (Объекты **RadioButton1** и **RadioButton2**) задайте надписи как **"Сортировка по возрастанию"** и **"Сортировка по убыванию"**, а у переключателя **"Сортировка по возрастанию" (RadioButton1)** задайте свойство **Checked (Включен)** равное **True (Истина)**.

Заполните список (**ListBox1**) значениями, представленными на рис. 22.5, а затем нажмите кнопку **"Ok"**.

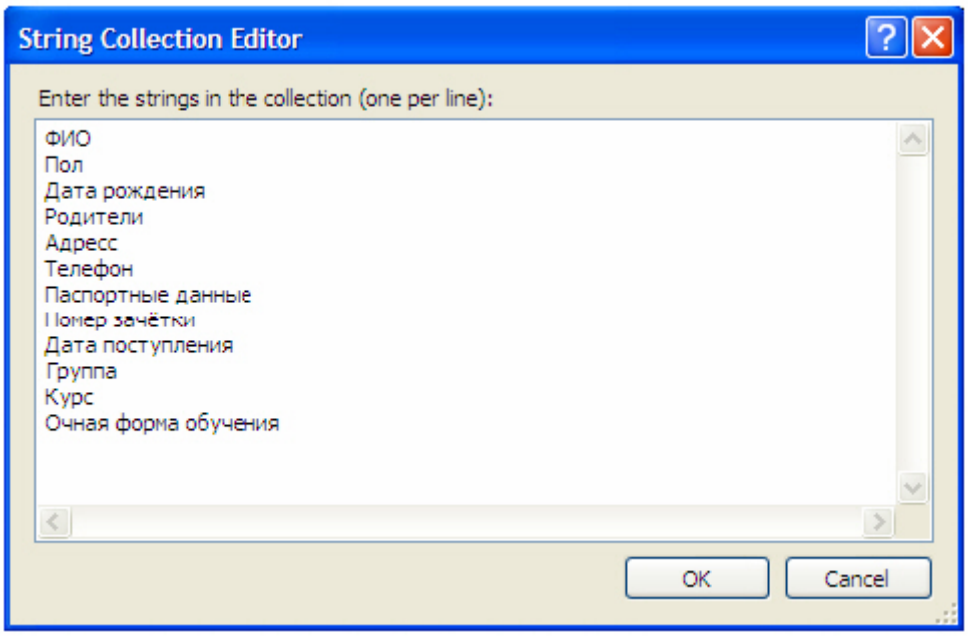


Рис. 22.5.

Настроим таблицу для отображения данных, удалив из нее поля с кодами. Выделите таблицу на форме и отобразите ее меню действий, щелкнув **ЛКМ** по кнопке

расположенной в верхнем правом углу таблицы. В меню действий выберите пункт **"Edit columns…"** (рис. 22.6).

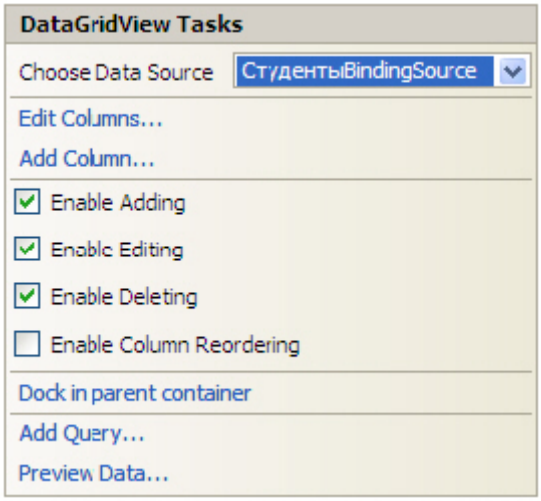


Рис. 22.6.

Появится окно настройки свойств полей таблицы **"Edit Columns"** (рис. 22.7).

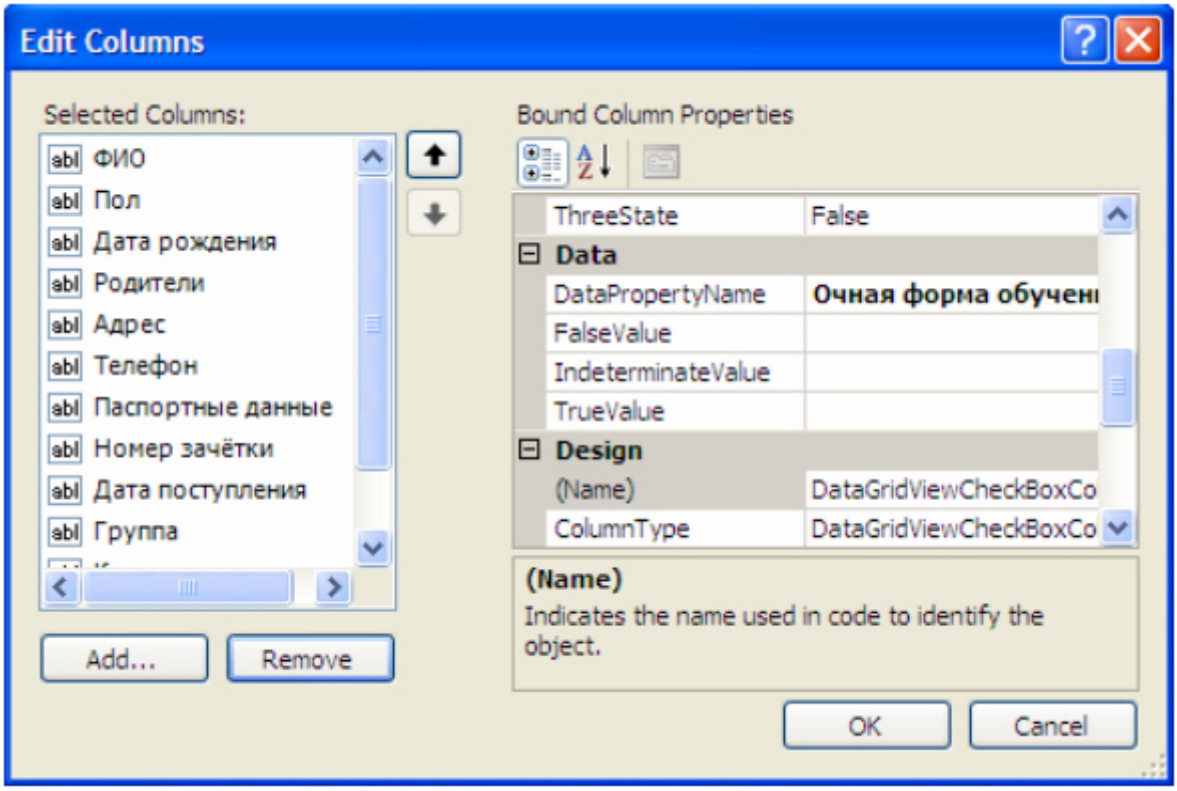


Рис. 22.7.

В окне **"Edit Columns"** из списка полей удалите поля **"Код студента"** и **"Код специальности"**, выделив их и нажав кнопку **"Remove"** (Удалить). Список полей примет вид показанный на рис. 22.7. Для закрытия окна редактирования полей, и сохранения изменений нажмите кнопку **"Ok"**.

Настроим заполнение выпадающего списка именами студентов из таблицы студенты. Отобразите меню действий выпадающего списка. Включите опцию **"Use Data Bound Items"**. Установите параметр **"Data Source"** равным **"Other Data Sources\Project Data Sources\StudentsDataSet\Студенты"**, а параметр **"Display Member"** равным **"ФИО"**. Остальные параметры оставьте без изменений (рис. 22.8).

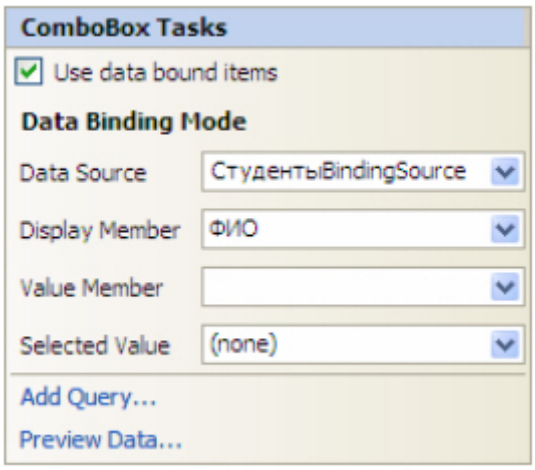


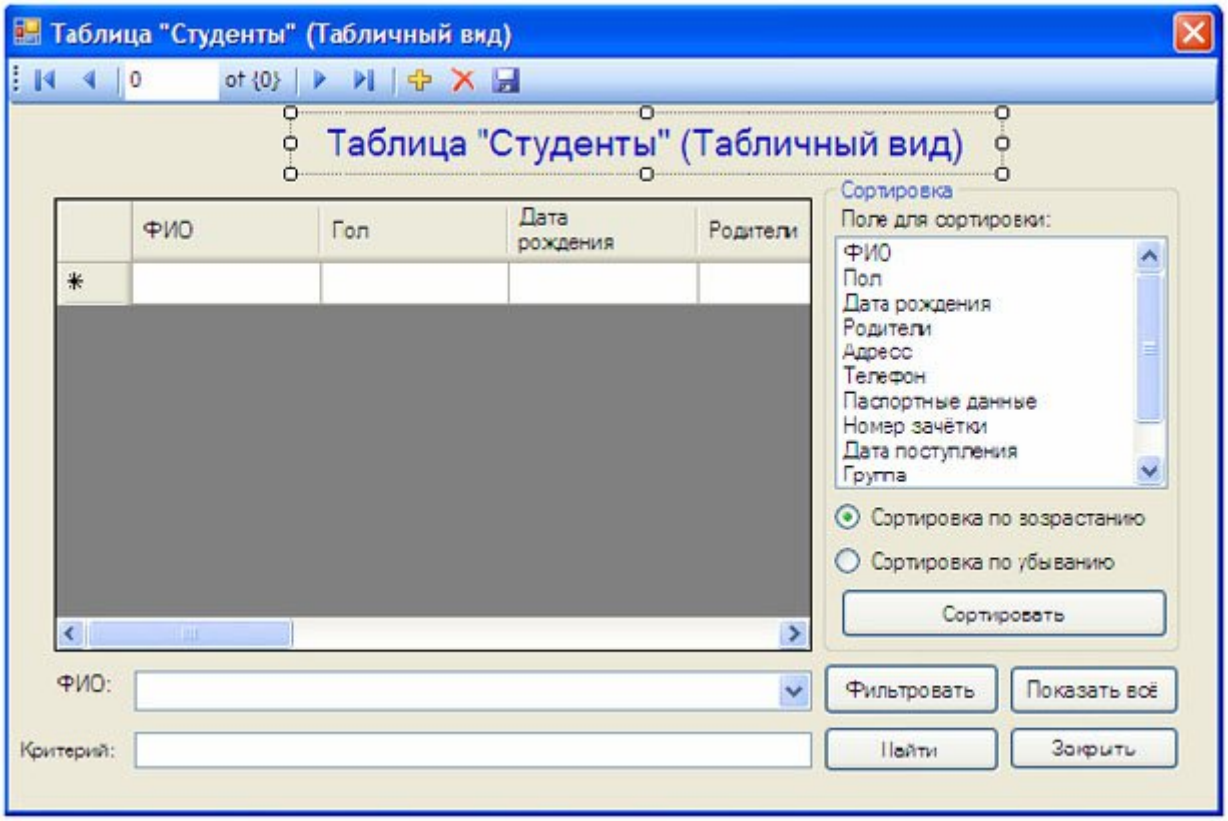
Рис. 22.8.

Закройте окно действий выпадающего списка. На панели невидимых объектов появится дополнительный объект связи **"СтудентыBindingSource1"**, предназначенный для заполнения выпадающего списка (рис. 22.9).

увеличить изображение

Рис. 22.9.

После настройки всех вышеперечисленных свойств объектов новая форма примет вид (рис. 22.10):



увеличить изображение

Рис. 22.10.

На этом мы заканчиваем настройку свойств объектов и переходим к написанию кода обработчиков событий объектов.
Работу с кодом начнем с написания кода для разблокирования кнопки **"Сортировать"**, при выборе пункта списка (**ListBox1**). Для создания процедуры события дважды щелкните **ЛКМ** по списку. Появится процедура обработки события, происходящего при выборе пункта списка (**ListBox1_SelectedIndexChanged**). В процедуре наберите команду разблокировки кнопки **"Сортировать" (Button1)**: Button1.Enabled = True (рис. 22.11).

```
Private Sub ListBox1_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Button1.Enabled = True

End Sub
```

увеличить изображение
Рис. 22.11.
Теперь перейдем к созданию кода сортирующего нашу таблицу в зависимости от выбранного поля и порядка сортировки при нажатии кнопки **"Сортировать"**. Дважды щелкните **ЛКМ** по кнопке **"Сортировать"**. Появится процедура **"Button1_Click"**, выполняемая при щелчке **ЛКМ** по кнопке. В процедуре наберите код, представленный на рис. 22.12.

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    Dim Col As System.Windows.Forms.DataGridViewColumn
    Select Case ListBox1.SelectedIndex
        Case 0
            Col = DataGridViewTextBoxColumn2
        Case 1
            Col = DataGridViewTextBoxColumn3
        Case 2
            Col = DataGridViewTextBoxColumn4
        Case 3
            Col = DataGridViewTextBoxColumn5
        Case 4
            Col = DataGridViewTextBoxColumn6
        Case 5
            Col = DataGridViewTextBoxColumn7
        Case 6
            Col = DataGridViewTextBoxColumn8
        Case 7
            Col = DataGridViewTextBoxColumn9
        Case 8
            Col = DataGridViewTextBoxColumn10
        Case 9
            Col = DataGridViewTextBoxColumn11
        Case 10
            Col = DataGridViewTextBoxColumn12
    End Select
    If RadioButton1.Checked Then
        СтудентыDataGridView.Sort(Col, System.ComponentModel.ListSortDirection.Ascending)
    Else
        СтудентыDataGridView.Sort(Col, System.ComponentModel.ListSortDirection.Descending)
    End If

End Sub
```

увеличить изображение
Рис. 22.12.
Рассмотрим код более подробно:

- Команда Dim Col As System.Windows.Forms.DataGridViewColumn создает переменную Col для хранения имени выбранного столбца таблицы;
- Затем следует блок Select Case…End Select, присваивающий в переменную Col имя выбранного столбца таблицы в зависимости от номера выбранного пункта списка (**ListBox1.SelectedIndex**). Если выбран первый пункт списка, то в переменную Col записывается столбец **DataGridViewTextBoxColumn2**, если второй, то - **DataGridViewTextBoxColumn3** и так далее. Хотелось бы отметить тот факт, что нумерация пунктов списка начинается с нуля, а нумерация столбцов с единицы. Первый

столбец **"ФИО"** носит имя **DataGridViewTextBoxColumn2**, так как имя **DataGridViewTextBoxColumn1** имеет столбец заголовков строк;

- Блок If…End If выполняет следующую операцию: если включен переключатель **"Сортировка по возрастанию" (RadioButton1)**, то отсортировать таблицу по полю заданному в переменной Col по возрастанию (**СтудентыDataGridView.Sort (Col, System.ComponentModel.ListSortDirection. Ascending)**), иначе по убыванию (**СтудентыDataGridView.Sort (Col, System. ComponentModel.ListSortDirection. Descending)**).

Рассмотрим код обработчика события нажатия кнопки **"Фильтровать" (Button2)**. Дважды щелкните по кнопке **"Фильтровать"** и в процедуре обработки события **"Button2_Click"** наберите код: СтудентыBindingSource.Filter = "ФИО='" & ComboBox1.Text & "'" (рис. 22.13).

```
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
    СтудентыBindingSource.Filter = "ФИО='" & ComboBox1.Text & "'"

End Sub
```

увеличить изображение

Рис. 22.13.

Замечание: У объекта **СтудентыBindingSource** имеется текстовое свойство **Filter** (рис. 22.13), которое определяет условие фильтрации. Условие фильтрации имеет синтаксис: "<Имя поля><Оператор>'<Значение>'".В нашем случае значение поля **"ФИО"** приравнивается к значению, выбранному в выпадающем списке (**ComboBox1.Text**) (рис. 22.13).

Теперь перейдем к кнопке **"Показать все"**, отменяющей фильтрацию записей. Дважды щелкните по вышеперечисленной кнопке. Появится процедура **Button3_Click**. В появившейся процедуре наберите команду СтудентыBindingSource.Filter = "" (рис. 22.14).

```
Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button3.Click
    СтудентыBindingSource.Filter = ""

End Sub
```

увеличить изображение

Рис. 22.14.

Заметим, что если присвоить свойству **"Filter"** значение пустой строки (**""**), то его действие будет отменено (рис. 22.14).

Далее рассмотрим реализацию поиска информации в таблице. Дважды щелкните по кнопке **"Найти"**. В появившейся процедуре обработки нажатия кнопки **"Button4_Click"** наберите следующий код (рис. 22.15).

```
Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button4.Click
    For i = 0 To СтудентыDataGridView.ColumnCount - 1
        For j = 0 To СтудентыDataGridView.RowCount - 1
            СтудентыDataGridView.Item(i, j).Style.BackColor = Color.White
            СтудентыDataGridView.Item(i, j).Style.ForeColor = Color.Black
        Next j
    Next i
    For i = 0 To СтудентыDataGridView.ColumnCount - 1
        For j = 0 To СтудентыDataGridView.RowCount - 1
            If InStr(СтудентыDataGridView.Item(i, j).Value, TextBox1.Text) Then
                СтудентыDataGridView.Item(i, j).Style.BackColor = Color.AliceBlue
                СтудентыDataGridView.Item(i, j).Style.ForeColor = Color.Blue
            End If
        Next j
    Next i

End Sub
```

увеличить изображение

Рис. 22.15.

Рассмотрим более подробно код вышеприведенной процедуры. Данная процедура состоит из двух частей:

- Первый блок **For i=0…..Next i**. перебирает все ячейки таблицы и устанавливает в них белый цвет фона и черный цвет текста. То есть, отменяет результаты предыдущего поиска;
- Второй блок **For i=0…..Next i**. перебирает все ячейки таблицы и если они содержат текст, введенный в поле ввода (**TextBox1**), то устанавливает в них голубой цвет фона и синий цвет текста, чем выделяет искомые ячейки.

Наконец рассмотрим код для кнопки **"Закрыть"**. Дважды щелкните **ЛКМ** по этой кнопке и в появившейся процедуре **"Button5_Click"** наберите команду "Me.Close()", закрывающую выше рассматриваемую форму (рис. 22.16).

```
Private Sub Button5_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button5.Click
    Me.Close()

End Sub
```

увеличить изображение

Рис. 22.16.

В заключение создадим кнопку на ленточной форме, отображающей таблицу **"Студенты"**, для отображения соответствующей табличной формы. Откройте ленточную форму для таблицы **"Студенты" (Form4)** и поместите на нее новую кнопку, как это показано на рис. 22.17.

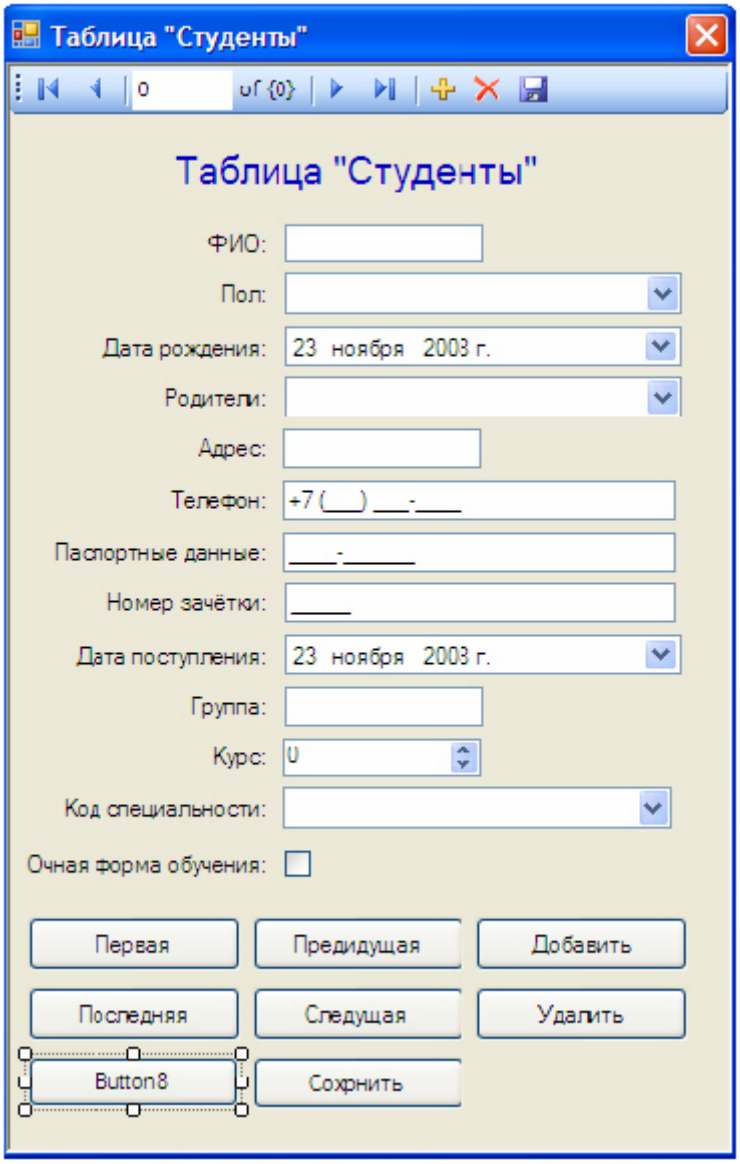


Рис. 22.17.

Задайте надпись у новой кнопки (свойство **Text**), как **"Таблица"**. Форма примет следующий вид (рис. 22.18):

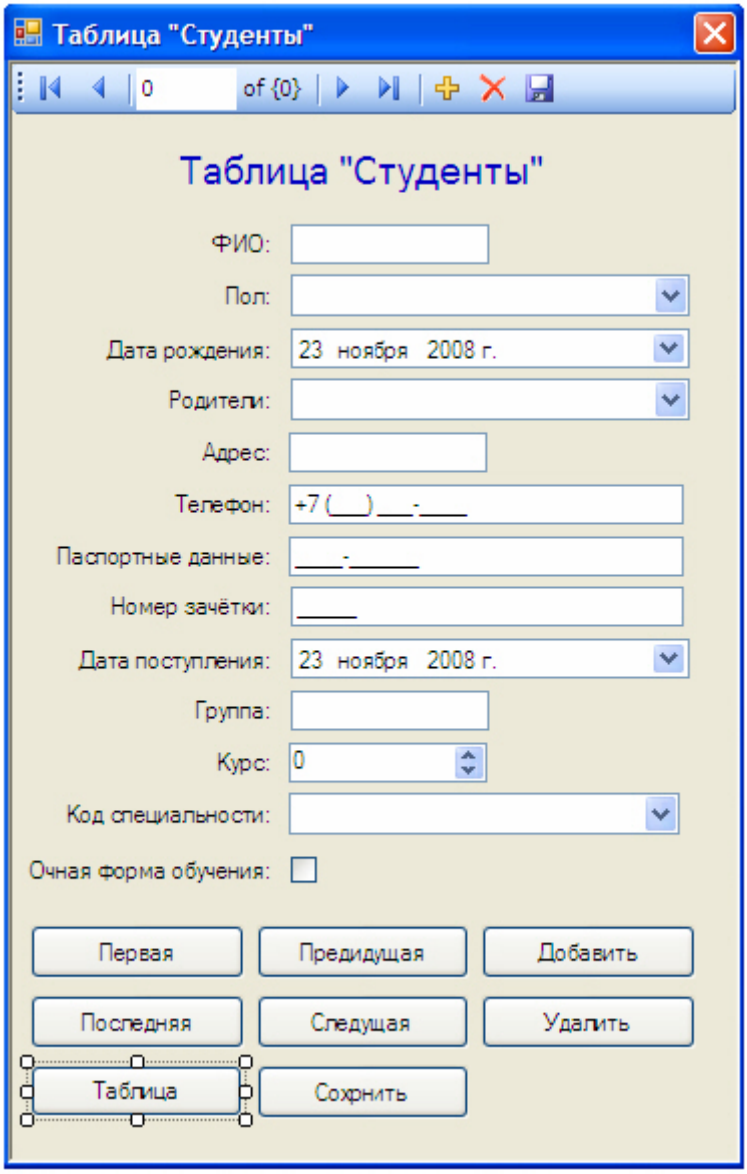


Рис. 22.18.

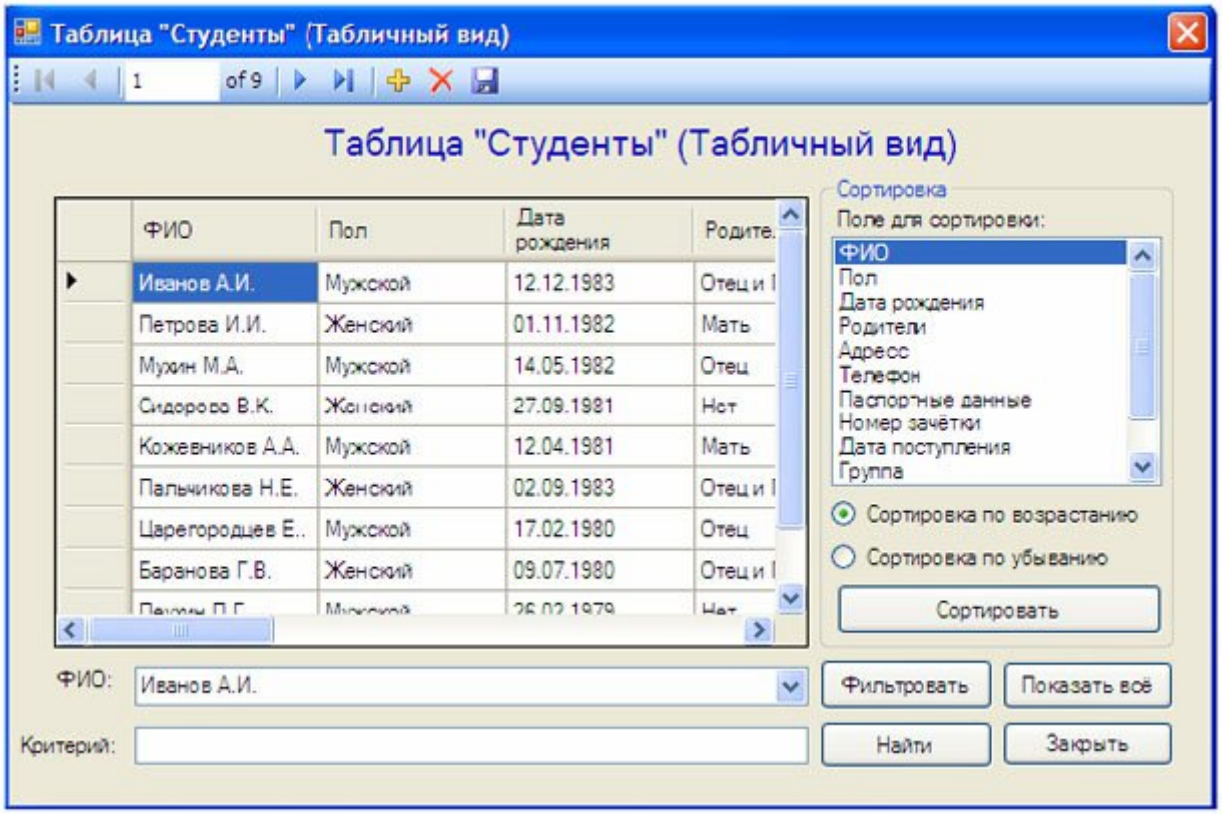
Подключим к кнопке **"Таблица"** созданную ранее табличную форму (**Form6**). Для этого дважды щелкните **ЛКМ** по кнопке **"Таблица"** и в появившейся процедуре **"Button8_Click"** наберите команду "Form6.Show" (рис. 22.19).

```
Private Sub Button8_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Form6.Show()
End Sub
```

увеличить изображение

Рис. 22.19.

Теперь проверим работоспособность созданной табличной формы. Запустите проект и на главной кнопочной форме нажмите кнопку **"Таблица "Студенты""**. На появившейся ленточной форме, отображающей таблицу **"Студенты"** нажмите кнопку **"Таблица"**. Появится новая табличная форма (рис. 22.20).



увеличить изображение

Рис. 22.20.

Проверьте, как работает поиск, фильтрация и сортировка записей в таблице, нажимая на соответствующие кнопки. После проверки работы формы для возвращения в среду разработки просто закройте все формы.

Хотелось бы отметить тот факт, что после проведения всех вышеописанных действий панель обозревателя проекта (**Solution Explorer**) примет вид (рис. 22.21):

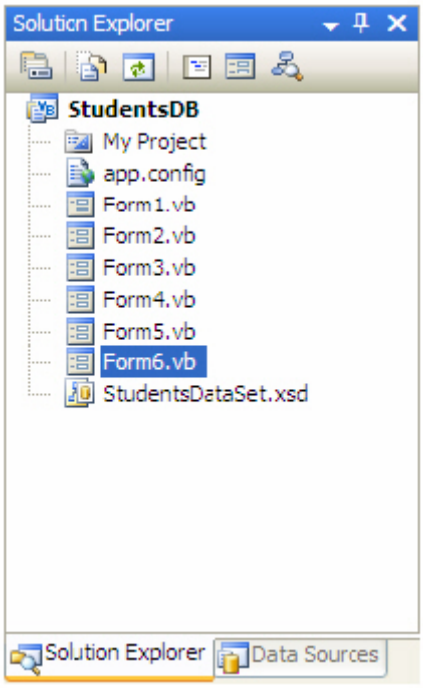


Рис. 22.21.

На этом мы заканчиваем работу с формами для работы с данными и переходим к отчетам.

Контрольные вопросы

1. Как создать новую табличную форму.
2. Как обеспечить фильтрацию и сортировку данных.
3. Как реализуется поиск информации в таблице.

Лабораторная работа № 13

«Создание отчетов в Visual Studio 2012.»

Цель работы:

Научиться создавать отчеты.

Начнем рассмотрение отчетов с создания ленточного отчета, отображающего таблицу **"Студенты"**. Для начала добавим в проект новый пустой отчет. Для этого в оконном меню выберите пункт **"Project\Add New Item…"** (рис. 24.1).

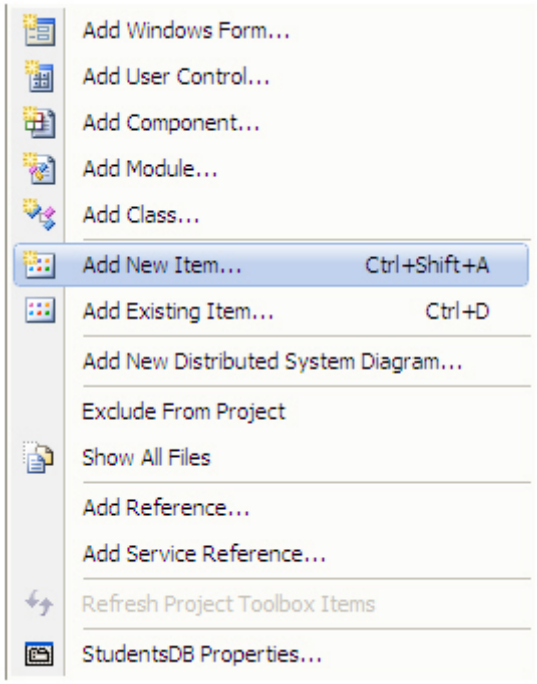
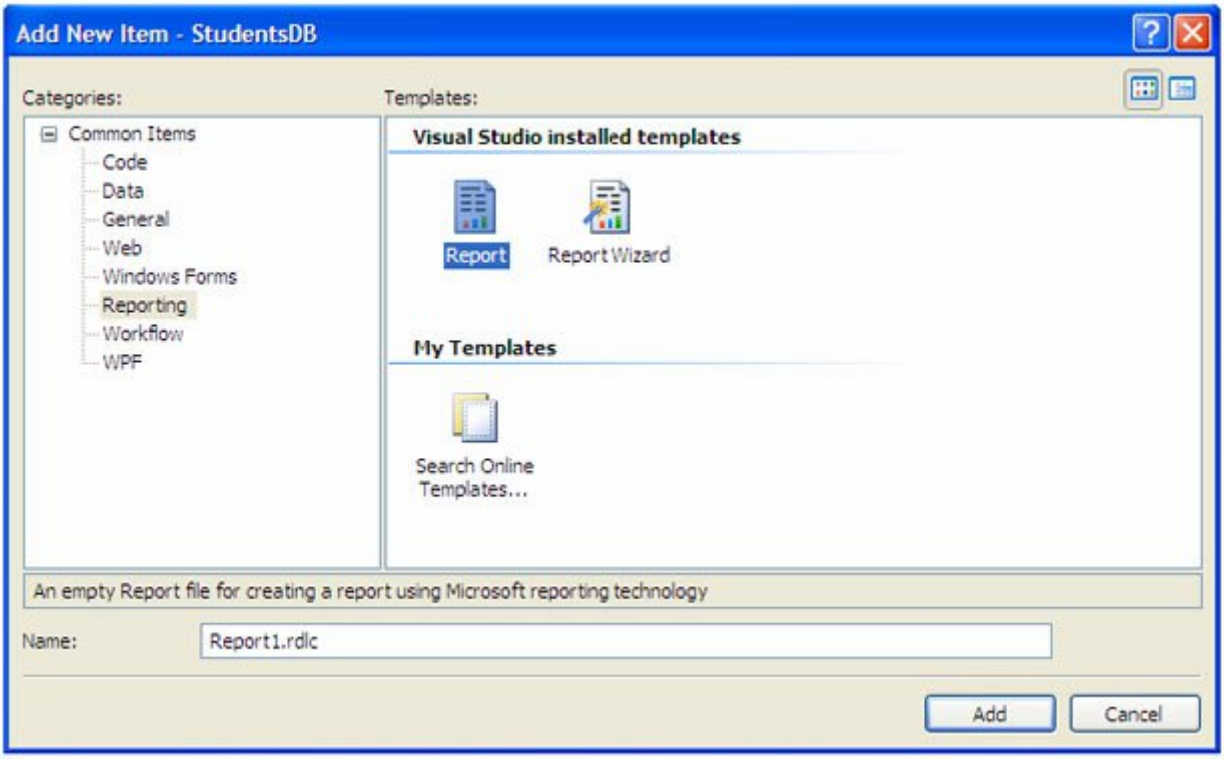


Рис. 24.1.

Появится окно **"Add New Item-StudentsDB"** (Добавить новый элемент - StudentsDB). В данном окне в списке **"Categories"** (Категории) выберите пункт **"Reporting"** (Отчеты), затем в области **"Templates"** (Шаблоны) выберите шаблон **"Report"** (Отчет) и нажмите кнопку **"Add"** (Добавить) (рис. 24.2).



увеличить изображение

Рис. 24.2.

В рабочей области среды разработки появится пустой отчет. Новый отчет также отобразится и на панели обозревателя проекта (**Solution Explorer**) (рис. 24.3).

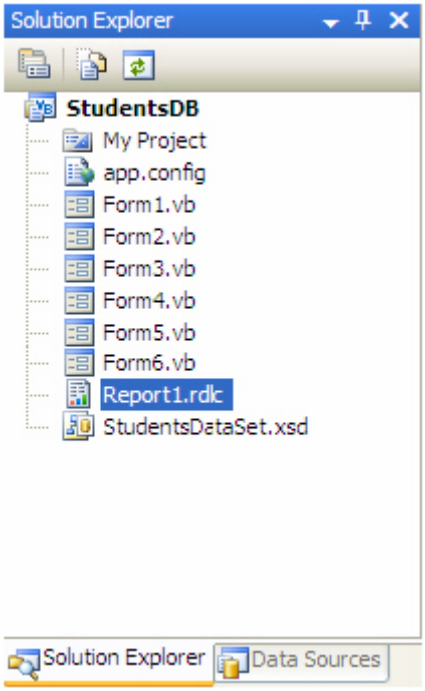


Рис. 24.3.

Для того чтобы в отчет поместить поля таблицы **"Студенты"** в него необходимо добавить объект **"Table"** (Таблица). Для этого на панели объектов (**Toolbox**) нажмите кнопку

а затем в отчете нарисуйте прямоугольник. Отчет примет вид, представленный на рис. 24.4.

увеличить изображение

Рис. 24.4.

Замечание: Объект таблица имеет три строки:

- **Header** (заголовок) - верхняя часть первой страницы отчета, содержит заголовок отчета;
- **Detail** (область данных) - средняя часть каждой страницы отчета, содержит поля отображаемой таблицы;
- **Footer** (примечание) - нижняя часть последней страницы отчета, содержит итоговую информацию по отчету.

Добавим в таблицу в область данных дополнительные строки для отображения полей таблицы **"Студенты"**. Выделите область данных, как это показано на рис. 24.5, щелкнув **ЛКМ** по заголовку строки области данных

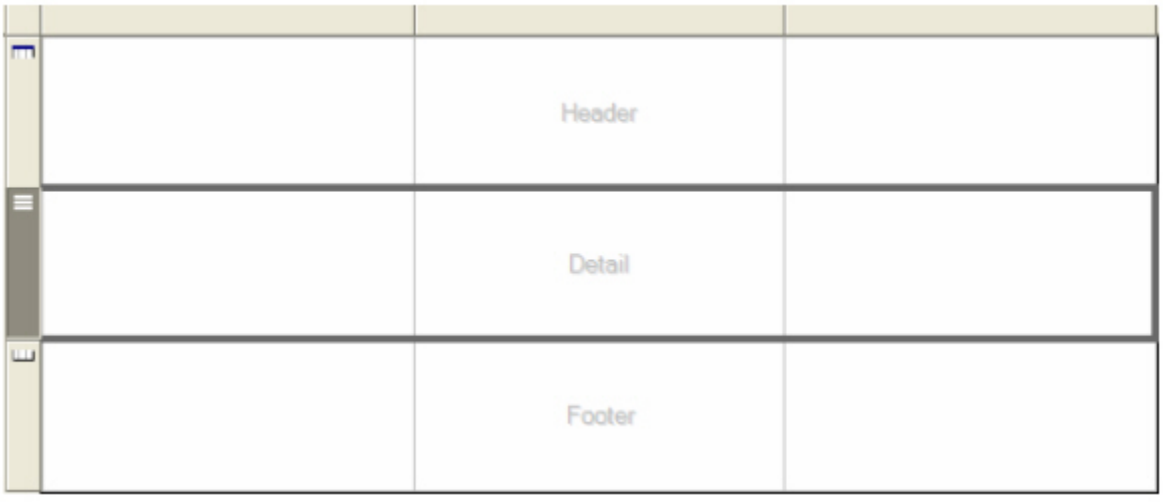


Рис. 24.5.

Для вставки новой строки щелкните **ПКМ** по заголовку выделенной строки

и в появившемся меню выберите пункт **"Insert Row Below"** (Вставить строку ниже) (рис. 24.6).

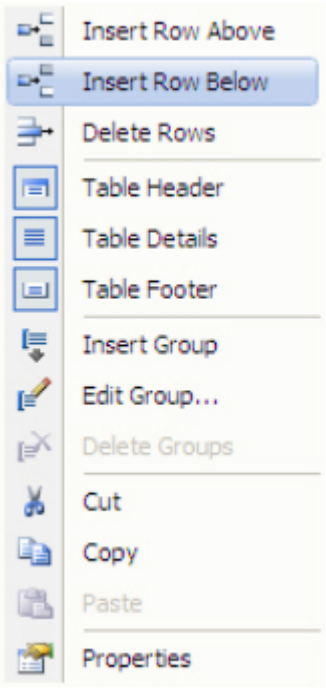


Рис. 24.6.

Проделайте эту операцию одиннадцать раз. Таблица примет вид (рис. 24.7)

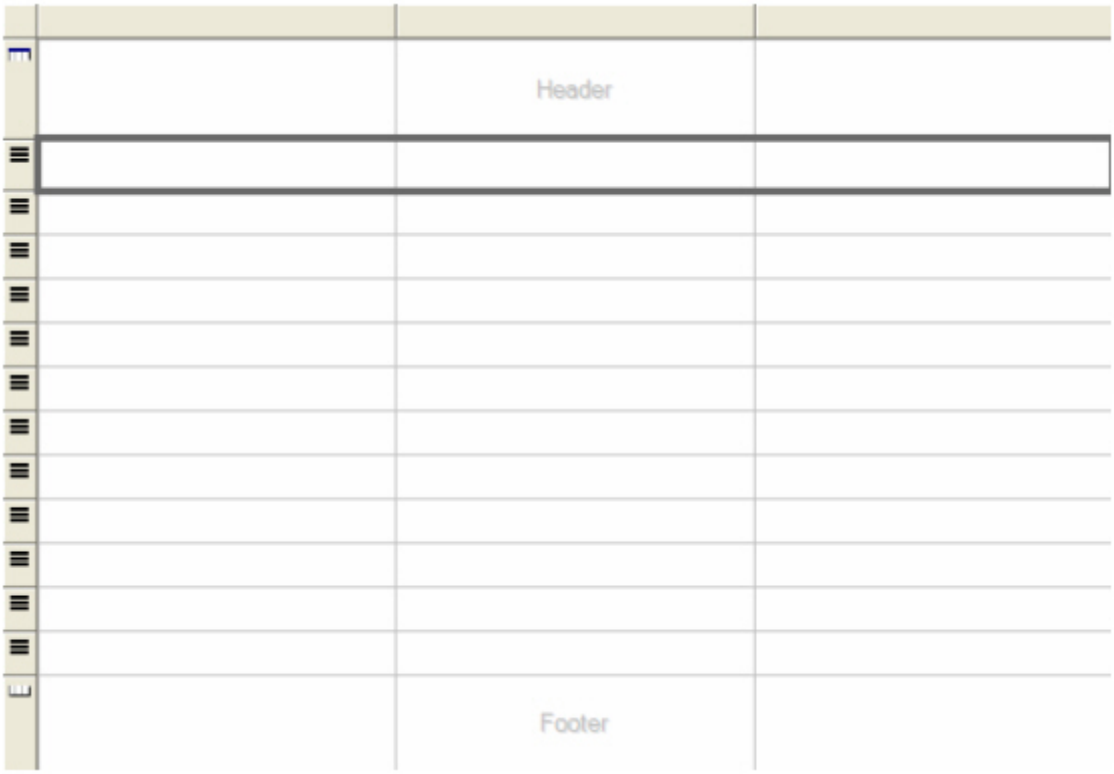


Рис. 24.7.

Наша таблица имеет три столбца. Один столбец является лишним, так как нам необходимо всего два столбца: один для отображения полей таблицы **"Студенты"**, а другой для подписей к ним. Поэтому удалим первый столбец таблицы. Щелкните **ПКМ** по заголовку первого столбца таблицы и в появившемся меню выберите пункт **"Delete Columns"** (Удалить строки) (рис. 24.8).

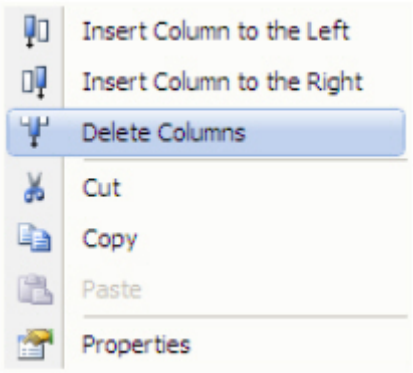


Рис. 24.8.

Теперь объединим ячейки в заголовке и примечании таблицы. Выделите ячейки заголовка таблицы, обведя их курсором мыши удерживая **ЛКМ**. Затем щелкните по выделению **ПКМ** и в появившемся меню выберите пункт **"Merge Cells"** (Объединить ячейки) (рис. 24.9).

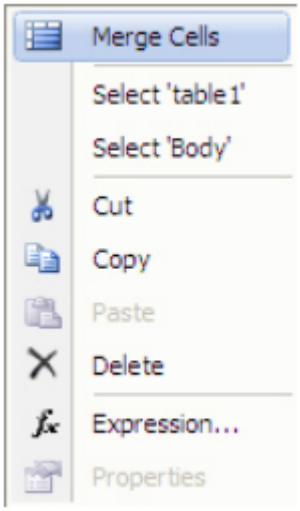


Рис. 24.9.

Проделайте аналогичную операцию с ячейками примечания. После выполнения вышеперечисленных операций таблица примет вид (рис. 24.10):

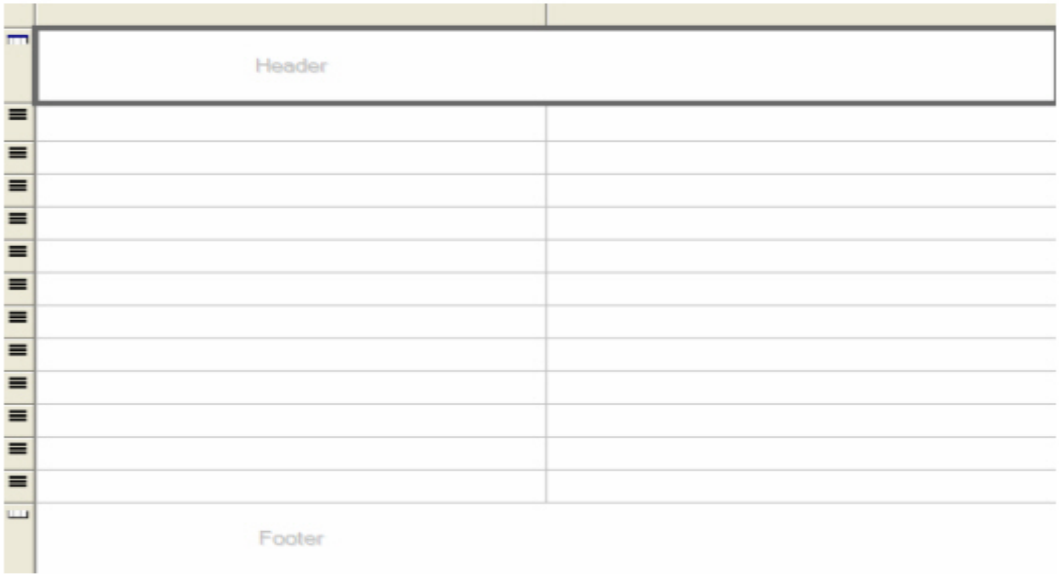


Рис. 24.10.

Поместим в таблицу поля таблицы **"Студенты"**. Для этого перетащите поля таблицы **"Студенты"** с панели **"Источники данных" (Data Sources)** в ячейки правого столбца таблицы, расположенные под заголовком, как показано на рис. 24.11. В одну ячейку перетаскивается одно поле Поля связи, имеющие в своем имени слово **"Код"** перетаскивать не нужно.

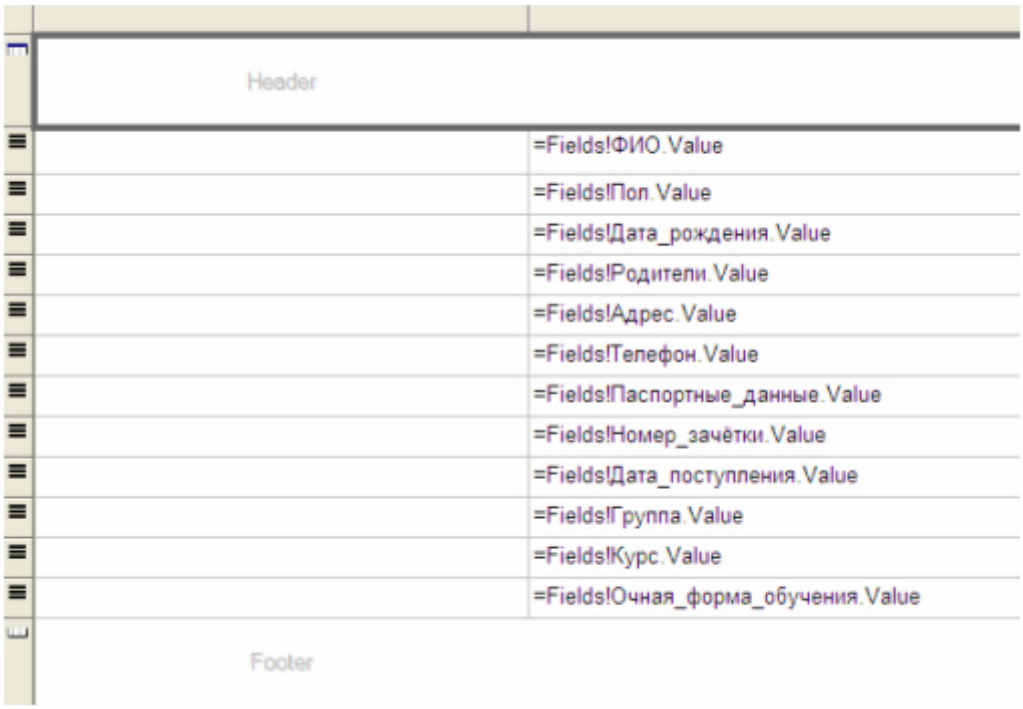


Рис. 24.11.

В левом столбце таблицы наберите имена полей и установите их выравнивание по правому краю (Свойство **TextAlign**). В заголовке наберите заголовок отчета **"Отчет таблицы "Студенты""** и сделайте выравнивание текста в нем по центру (рис. 24.12).

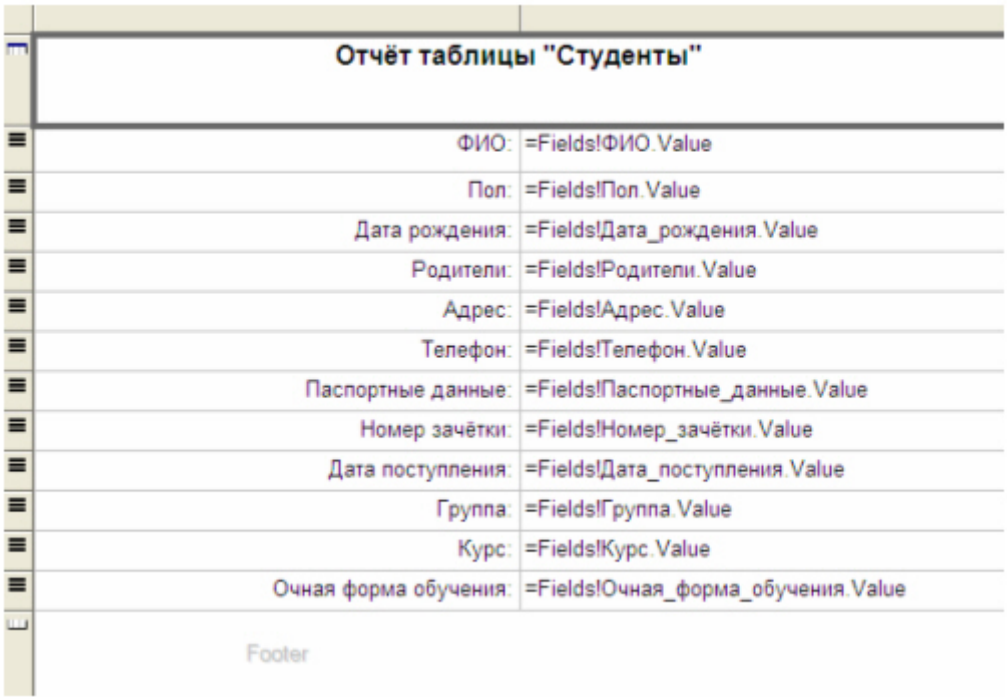


Рис. 24.12.
Теперь выделим ячейки, отображающие поле **"ФИО"** серым цветом для логического отделения одного студента от другого. Выделите вторую строку таблицы и на панели свойств (**Properties**) в свойстве **"BackColor"** (Цвет фона) выберите серый цвет. Таблица примет следующий вид (рис. 24.13).

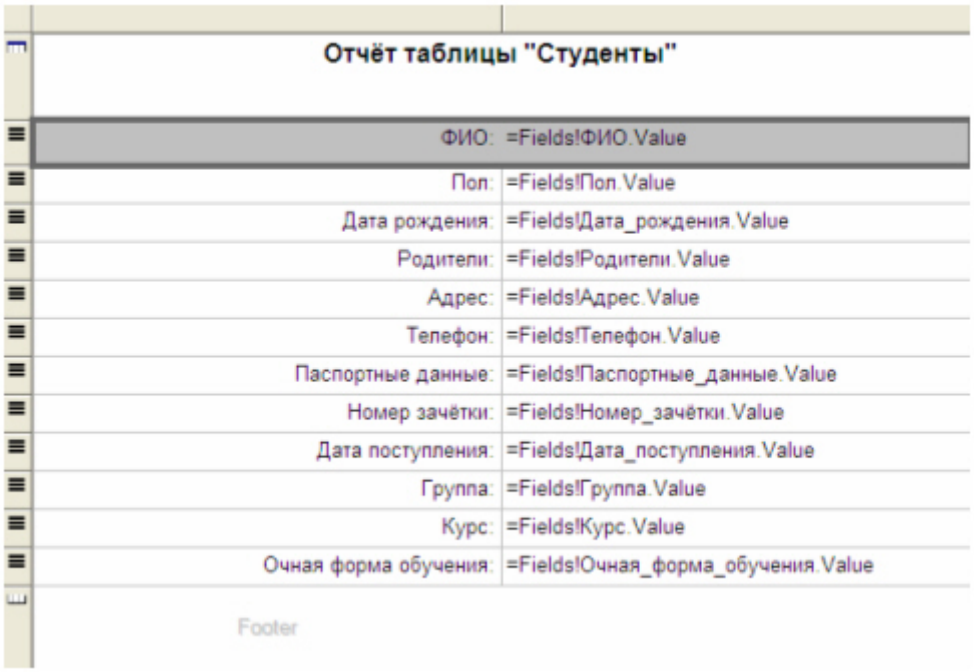


Рис. 24.13.
Заключительным шагом в настройке таблицы будет включение отображения границ ячеек. Выделите все ячейки с полями и подписями к ним. Затем на панели инструментов при помощи кнопки

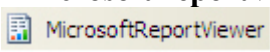
включите границы выделенных ячеек таблицы (рис. 24.14).
Замечание: Если кнопка

отсутствует на панели инструментов, то необходимо включить панель редактирования границ отчетов (**Report borders**). Для этого щелкните **ПКМ** по панели инструментов и в появившемся меню выберите пункт **"Report borders"**.

Рис. 24.14.

Теперь создадим форму отображающую созданный отчет. Добавьте в проект новую форму (**Form7**). Определите заголовок формы (Свойство **Text**) как **"Отчет таблицы "Студенты""**.

Поместите на форму специальный объект, отображающий отчеты **"MicrosoftReportViewer"**, используя кнопку



расположенную на панели объектов (**Toolbox**). К объекту, отображающему отчеты подключите, созданный ранее отчет. Для этого в меню действий в выпадающем списке **"Choose report"** (Выберите отчет) выберите отчет **"StudentsDB.Report1.rdlc"**.

Разверните объект, отображающий отчеты во всю форму. Для этого в меню действий объекта выберите пункт **"Dock in Parent Container"** (Развернуть в родительский контейнер). Меню действий примет вид (рис. 24.15):

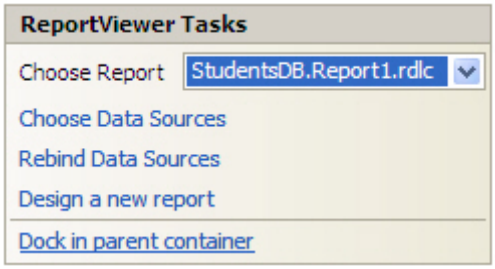


Рис. 24.15.

Замечание: Обратите внимание на тот факт, что после подключения отчета к объекту, отображающему отчеты, на панели невидимых объектов появились объекты связи, подключающие отчет к таблице **"Студенты"** (рис. 24.16).

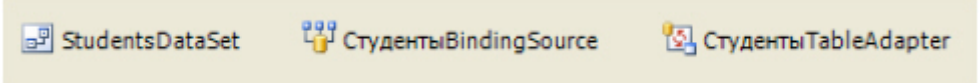
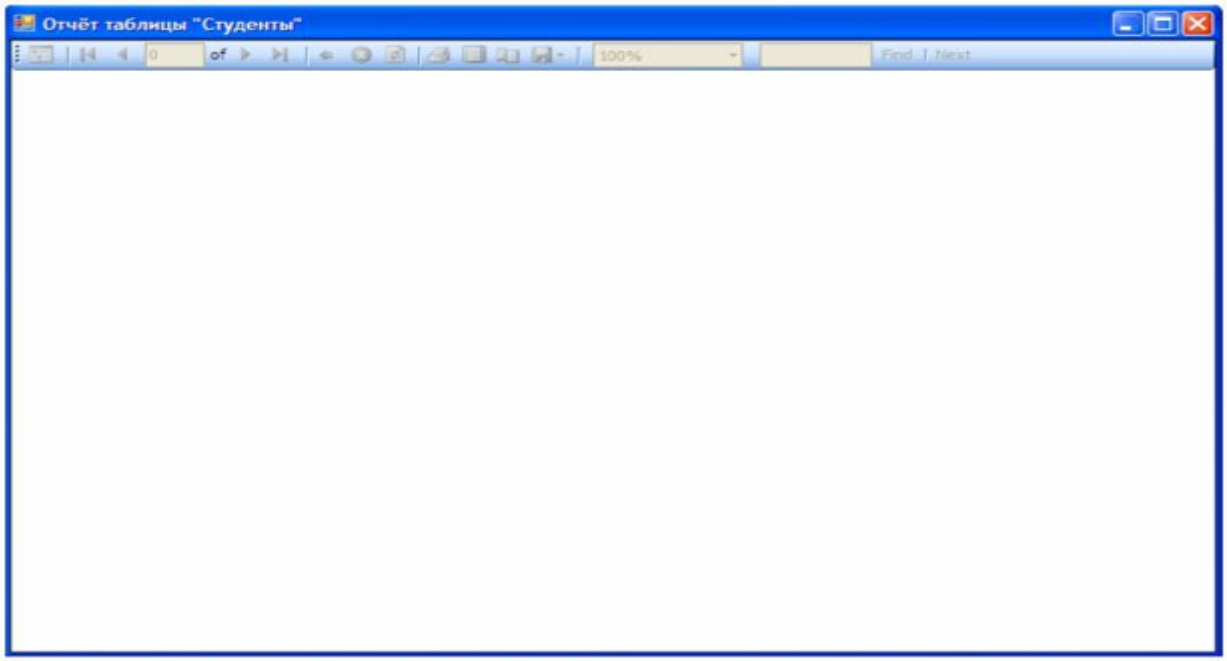


Рис. 24.16.

После выполнения всех вышеперечисленных действий форма, отображающая отчет примет вид, представленный на рис. 24.17.



увеличить изображение

Рис. 24.17.

Проверим работоспособность нового отчета, подключив форму для его отображения к кнопке на форме **"Таблица "Студенты""**. На форме, отображающей таблицу **"Студенты"** создайте кнопку (**Button9**) (рис. 24.18).

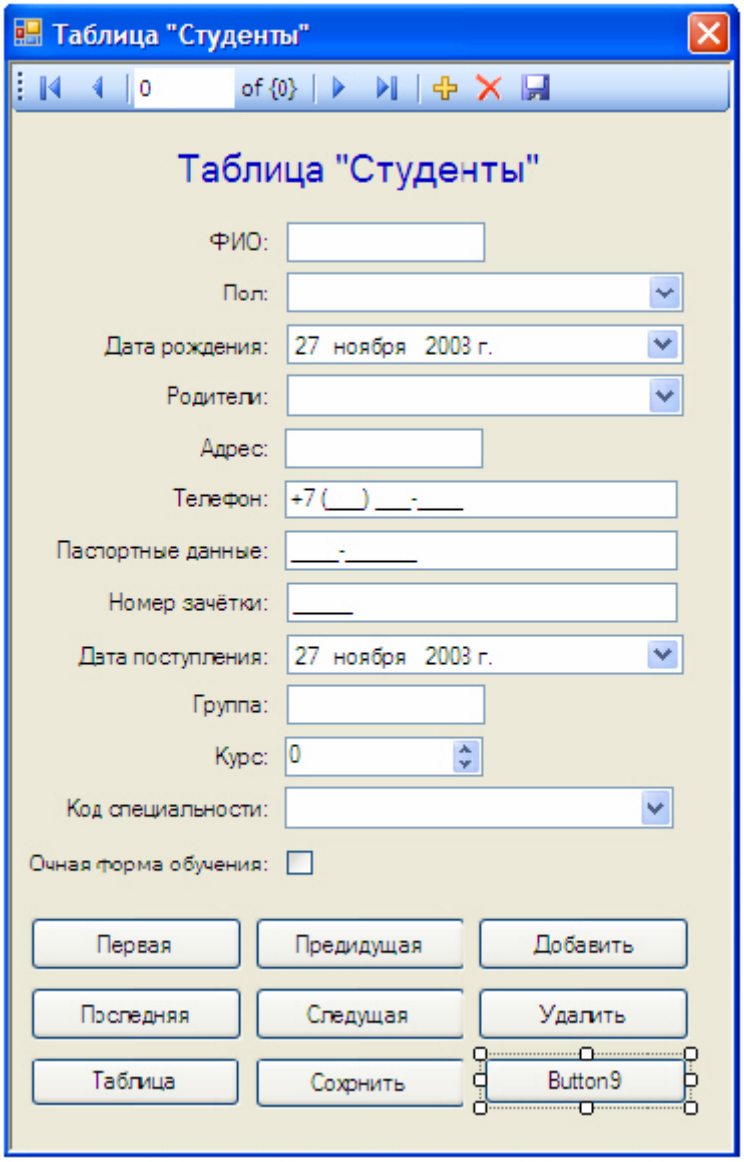


Рис. 24.18.

Задайте надпись на кнопке (Свойство **Text**) равную **"Отчет"** (рис. 24.19).

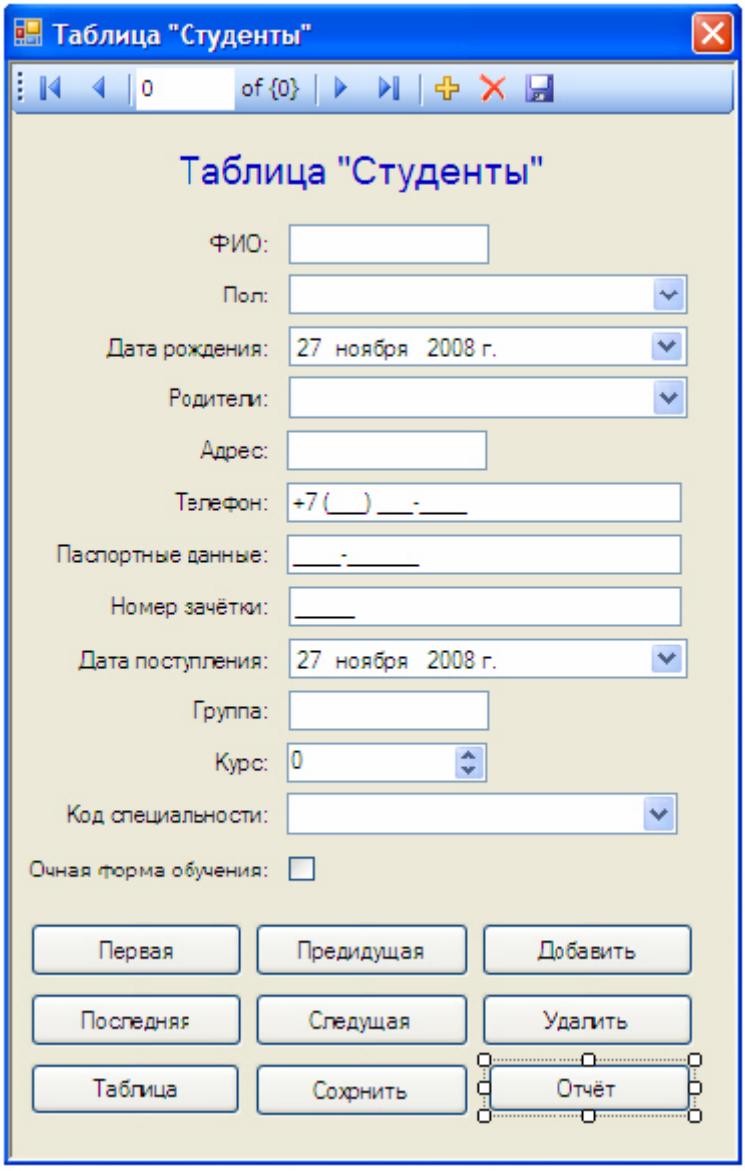


Рис. 24.19.

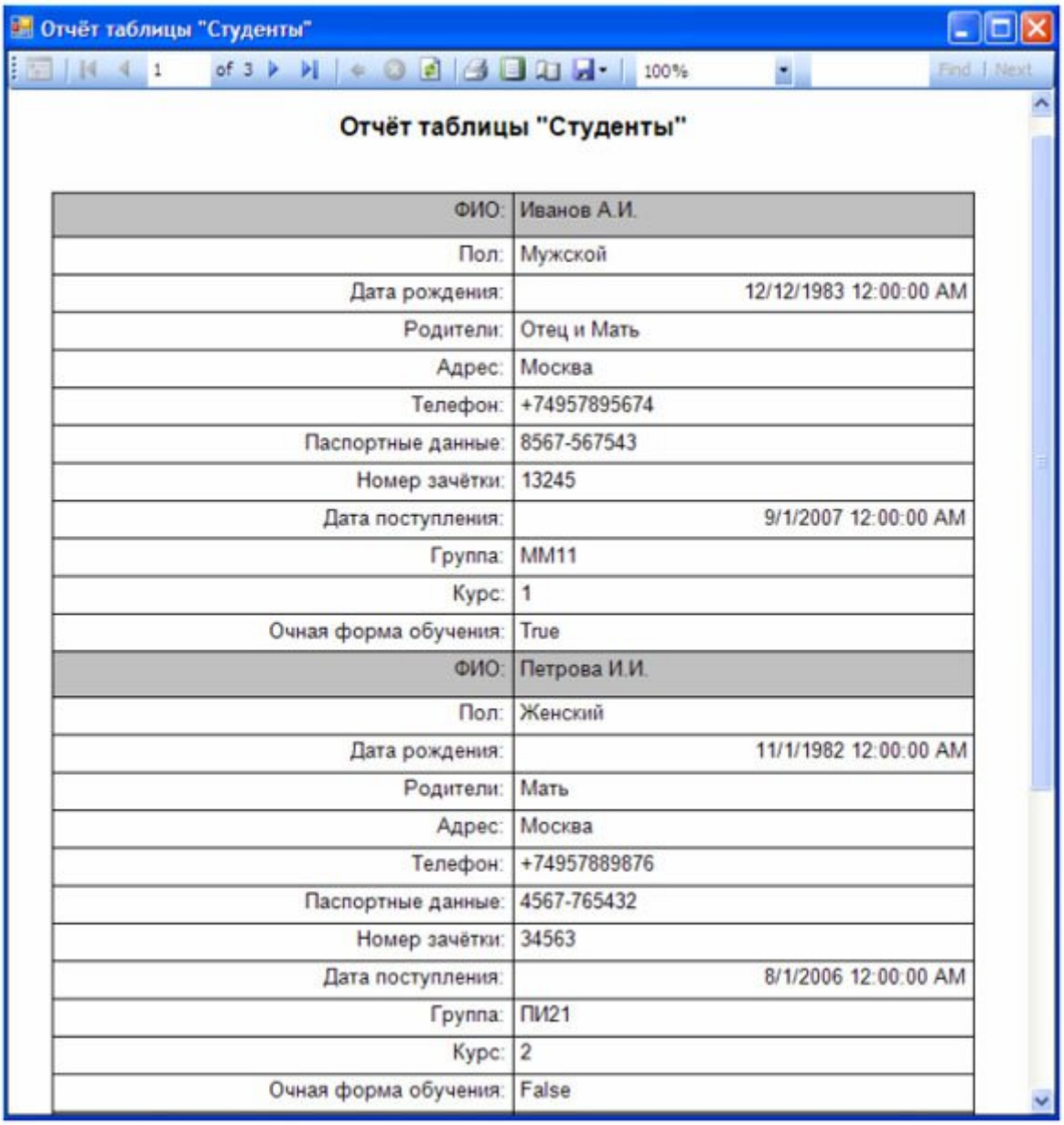
Теперь определим код обработчика события нажатия кнопки. Дважды щелкните **ЛКМ** по кнопке **"Отчет"** и в появившейся процедуре **"Button9_Click"** наберите команду "Form7.Show()" (рис. 24.20).

```
Private Sub Button9_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button9.Click
    Form7.Show()
End Sub
```

увеличить изображение

Рис. 24.20.

Запустите проект и на главной кнопочной форме нажмите кнопку **"Таблица "Студенты""**. На появившейся ленточной форме, отображающей таблицу **"Студенты"** нажмите кнопку **"Отчет"**. Появится новая форма с отчетом, построенным по таблице **"Студенты"** (рис. 24.21).



увеличить изображение

Рис. 24.21.

Проверьте работу отчета. Для завершения работы проекта просто закройте все открытые формы.

На этом мы завершаем разработку нашей БД **"Студент"**.

Контрольные вопросы

1. Как создать ленточный отчет.
2. Как создать форму отображающую созданный отчет.

Лабораторная работа № 14

«Выполнение индивидуальных заданий по проектированию информационных систем в Microsoft SQL Server 2012.»

Цель работы:

Научиться создавать в Microsoft SQL Server такие объекты, как таблицы, запросы, фильтры, хранимые процедуры, пользовательские функции, диаграммы и триггеры.

Задание. В программе Microsoft SQL Server создать базу данных для учета успеваемости студентов, содержащую следующие объекты:

Таблицы:	1. Специальности (Код специальности, Наименование специальности, Описание специальности)[5 записей]. 2. Предметы (Код предмета, Наименование предмета, Описание предмета)[5 записей]. 3. Студенты (Код студента, ФИО, Пол, Дата рождения, Родители, Телефон, Дата поступления, Паспортные данные, Группа, Курс, Код специальности, Очная форма обучения, Номер зачетки)[10 записей]. 4. Оценки (Код студента, Дата экзамена 1, Код предмета 1, Оценка 1, Дата экзамена 2, Код предмета 2, Оценка 2 Дата экзамена 3, Код предмета 3, Оценка 3, Средний балл)[10 записей].
Запросы:	1. Студенты+Специальности (Связывает таблицы "Студенты" и "Специальности" по полю "Код специальности"). 2. Студенты+Оценки (Связывает таблицы "Студенты" и "Оценки" по полю "Код студента", а также таблицу "Предметы" по полю "Код предмета" с таблицей "Оценки" по полям и "Код предмета 1" и "Код предмета 2" и "Код предмета 3").
Фильтры:	1. Фильтры для отображения студентов отдельных специальностей (На основе запроса "Студенты+Специальности"). 2. Фильтры для отображения студентов, не имеющих родителей или имеющих только одного родителя (На основе запроса "Студенты+Специальности"). 3. Фильтры для отображения студентов заданной формы обучения (На основе запроса "Студенты+Специальности").
Хранимые процедуры	1. Хранимая процедура для вычисления среднего арифметического трех величин. 2. Хранимая процедура для отбора студентов из таблицы "Студенты" по их "ФИО" 3. Хранимая процедура для отбора студентов, у которых средний балл выше заданного. 4. Хранимая процедура для отображения студентов старше заданного возраста
Пользовательские функции	1. Скалярная пользовательская функция, вычисляющая среднее трех величин. 2. Табличная пользовательская функция, вычисляющих текущий возраст студентов в зависимости от их даты рождения.
Диаграммы	1. "Диаграмма БД Студенты", отображающая связи между таблицами.
Триггеры	1. Триггер, выводящий сообщение "Запись добавлена" при добавлении записи в таблицу "Студенты". 2. Триггер, отображающий сообщение "Запись изменена" при обновлении записи в таблице "Студенты". 3. Триггер "Удаление студента" для обеспечения целостности данных, который при удалении записи из таблицы Студенты

	сначала удаляет все связанные с ней записи из таблицы "Оценки", а затем удаляет саму запись из таблицы "Студенты".

Контрольные вопросы

1. Связывание таблиц. Создание запросов.
2. Создание фильтров. Установка критериев отбора записей в фильтре. Сортировка записей в фильтре.
3. Создание хранимых процедур.
4. Создание пользовательских функций.
5. Создание диаграмм.
6. Создание триггеров.

Лабораторная работа № 15

«Выполнение индивидуальных заданий по проектированию информационных систем в Visual Studio 2012. Создание ленточных и табличных форм для работы с базами данных.»

Цель работы:

Научиться подключать файлы данных SQL Server к проекту Visual Studio, создавать пользовательский интерфейс (главная кнопочная форма, простые и сложные ленточные формы для работы с данными, табличные формы).

В программе Microsoft Visual Studio создать приложение для работы с базой данных по учету успеваемости студентов, созданной в лабораторной работе № 14.
Создать главную кнопочную форму, простые и сложные ленточные формы для работы с данными, табличные формы и отчет «Студенты».

Контрольные вопросы

1. Подключение базы данных SQL Server к проекту Visual Studio. Окно «Источники данных».
2. Создание простых ленточных форм для работы с данными.
3. Создание сложных ленточных форм для работы с данными. Основные команды для работы с записями.
4. Задание маски ввода в ленточной форме.
5. Отображение полей при помощи числового счетчика.
6. Заполнение полей формы при помощи выпадающего списка.
7. Создание табличных форм.

Лабораторная работа № 16

«Выполнение индивидуальных заданий по проектированию информационных систем в Visual Studio 2012. Создание отчетов и диаграмм.»

Цель работы:

Научиться создавать создавать отчеты и диаграммы.

В программе Microsoft Visual Studio для работы с базой данных по учету успеваемости студентов, созданной в лабораторной работе № 14.
Создать отчет «Студенты».

Контрольные вопросы

1. Задание сортировки данных в табличной форме.
2. Задание фильтрации данных в табличной форме.
3. Создание отчетов.

Лабораторная работа № 17

«Подготовка документации IT проекта.»

Цель работы:

Научиться готовить необходимую организационно-техническую и эксплуатационную документацию IT проекта.

Общие требования к документированию

Документы должны быть представлены на бумажном виде (оригинал) и на магнитном носителе (копия). Исходные тексты программ - только на магнитном носителе (оригинал). Возможно предоставление комплекта документации и текстов программ на компакт-дисках.

Все документы должны быть оформлены на русском языке. Состав документов на общее программное обеспечение, поставляемое в составе АИС, должен соответствовать комплекту поставки компании - изготовителя.

Перечень подлежащих разработке документов

В ходе создания Подсистемы должен быть подготовлен и передан Заказчику комплект документации в составе:

- проектная документация и материалы техно-рабочего проекта на разработку Подсистемы;
- конструкторская, программная и эксплуатационная документация на Подсистему;
- сопроводительная документация на поставляемые программно-аппаратные средства в комплектности поставки заводом-изготовителем;
- предложения по организации системно-технической поддержки функционирования Подсистемы.

Состав и содержание комплекта документации на Подсистему может быть уточнен на стадии проектирования.

Подготовленные документы должны удовлетворять требованиям государственных стандартов и рекомендаций по оформлению, содержанию, форматированию, использованию терминов, определений и надписей, обозначений программ и программных документов.

Порядок выполнения лабораторной работы

По заданному преподавателем описанию предметной области разработать организационно-техническую и эксплуатационную документацию (использовать результаты предыдущих работ).

Контрольные вопросы

1. Каковы общие требования к документированию.
2. Каков перечень подлежащих разработке документов.

Лабораторная работа № 18

«Расчет экономической эффективности проекта.»

Цель работы:

Научиться создавать сложные ленточные формы для работы с данными.

Как было указано ранее, обычно сначала строится функциональная модель существующей организации работы — AS-IS (как есть). После построения модели AS-IS проводится анализ бизнес-процессов, потоки данных и объектов перенаправляются и улучшаются, в результате строится модель TO-BE. Как правило, строится несколько моделей TO-BE, из которых по какому-либо критерию выбирается наилучшая. Проблема состоит в том, что таких критериев много и непросто определить важнейший. Для того

чтобы определить качество созданной модели с точки зрения эффективности бизнес-процессов, необходима система метрики, т. е. качество следует оценивать количественно.

BPwin предоставляет аналитику два инструмента для оценки модели — *стоимостный анализ*, основанный на работах (Activity Based Costing, ABC), и *свойства, определяемые пользователем* (User Defined Properties, UDP). Функциональное оценивание – ABC – это технология выявления и исследования стоимости выполнения той или иной функции (действия). Исходными данными для функционального оценивания являются затраты на ресурсы (материалы, персонал и т.д.). В сравнении с традиционными способами оценки затрат, при применении которых часто недооценивается продукция, производимая в незначительном объеме, и переоценивается массовый выпуск, ABC обеспечивает более точный метод расчета стоимости производства продукции, основанный на стоимости выполнения всех технологических операций, выполняемых при ее выпуске.

Стоимостный анализ *представляет собой соглашение об учете, используемое для сбора затрат, связанных с работами, с целью определить общую стоимость* процесса. Стоимостный анализ основан на модели работ, потому что количественная оценка невозможна без детального понимания функциональности предприятия. Обычно ABC применяется для того, чтобы понять происхождение выходных затрат и облегчить выбор нужной модели работ при реорганизации деятельности предприятия (Business Process Reengineering, BPR). С помощью стоимостного анализа можно решить такие задачи, как определение действительной стоимости производства продукта, определение действительной стоимости поддержки клиента, идентификация наиболее дорогостоящих работ (тех, которые должны быть улучшены в первую очередь), обеспечение менеджеров финансовой мерой предлагаемых изменений и т.д.

ABC-анализ может проводиться только тогда, когда модель работы последовательная (следует синтаксическим правилам IDEF0), корректная (отражает бизнес), полная (охватывает всю рассматриваемую область) и стабильная (проходит цикл экспертизы без изменений), другими словами, когда создание модели работы закончено.

ABC включает следующие основные понятия:

- **объект затрат —** *причина, по которой работа выполняется, обычно основной выход работы.* Стоимость работ есть суммарная стоимость объектов затрат (**"Сборка и тестирование компьютеров"**, рис. 1);
- **двигатель затрат** — *характеристики входов и управлений работы* ("Заказы клиентов", "Правила сборки и тестирования", "Персонал производственного отдела", рис. 1), которые влияют на то, как выполняется и как долго длится работа;
- **центры затрат***, которые можно трактовать как статьи расхода*.

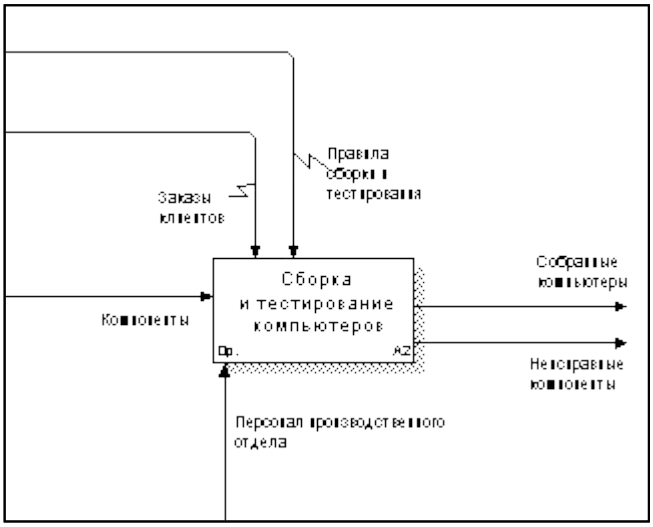


Рисунок 1 - Иллюстрация терминов ABC

При проведении стоимостного анализа в BPwin сначала задаются единицы измерения времени и денег. Для задания единиц измерения следует вызвать диалог Model Properties (меню Model), закладка ABC Units (рис. 2).

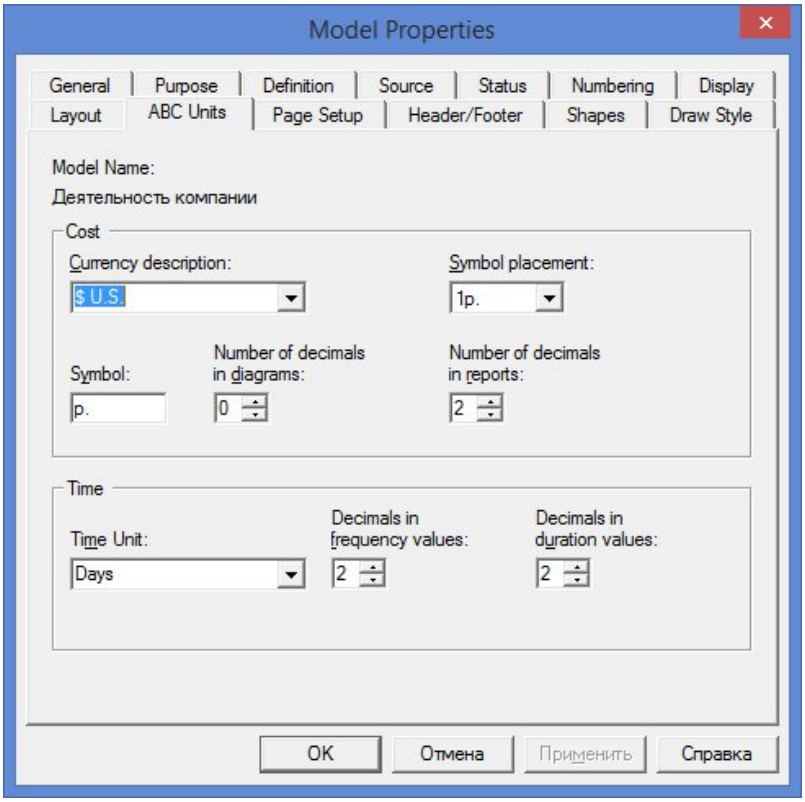


Рисунок 2 - Настройка единиц измерения валюты и времени

Если в списке выбора отсутствует необходимая валюта (например, рубль), ее можно добавить. Диапазон измерения времени в списке Time Unit достаточен для большинства случаев — от секунд до лет.

Затем описываются центры затрат (cost centers). Для внесения центров затрат необходимо вызвать диалог Cost Center Editor из меню Model (рис. 3).

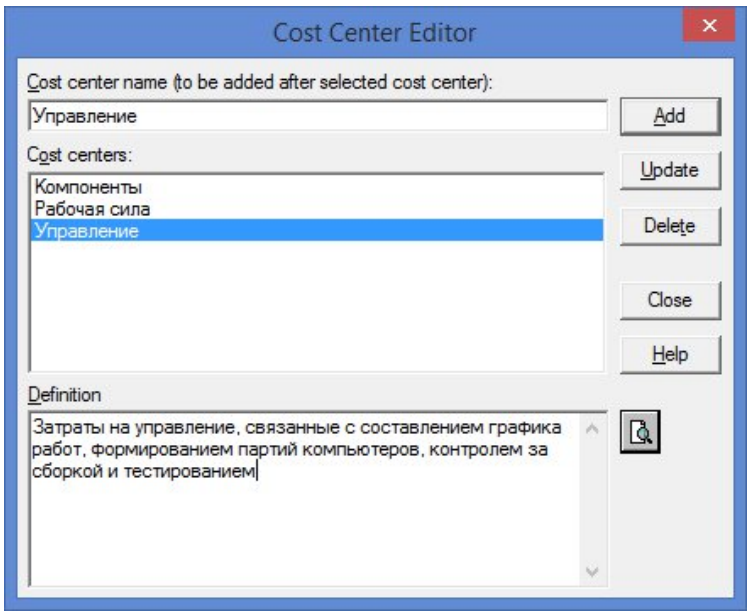


Рисунок 3 - Диалог Cost Center Editor

Каждому центру затрат следует дать подробное описание в окне Definition. Для задания стоимости работы (для каждой работы на диаграмме декомпозиции) следует щелкнуть правой кнопкой мыши по работе и на всплывающем меню выбрать Cost (рис. 4). В диалоге Activity Cost указывается частота проведения данной работы в рамках общего процесса (окно Frequency) и продолжительность (Duration). Затем следует выбрать в списке один из центров затрат и в окне Cost задать его стоимость. Аналогично назначаются суммы по каждому центру затрат, т. е. задается стоимость каждой работы по каждой статье расхода. Если в процессе назначения стоимости возникает необходимость внесения дополнительных центров затрат, диалог Cost Center Editor вызывается прямо из диалога Activity Properties/Cost соответствующей кнопкой.

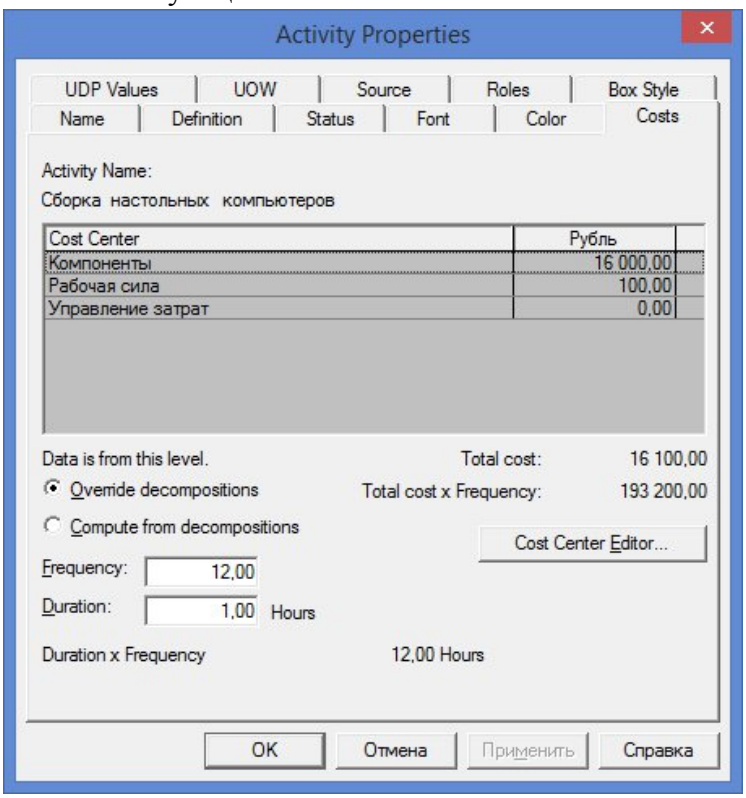


Рисунок 4 - Задание стоимости работ в диалоге Activity Properties/Cost

Общие затраты по работе рассчитываются как сумма по всем центрам затрат. При вычислении затрат вышестоящей (родительской) работы сначала вычисляется произведение затрат дочерней работы на частоту работы (число раз, которое работа выполняется в рамках проведения родительской работы), затем результаты складываются. Если во всех работах модели включен режим Compute from Decompositions (рис. 4), подобные вычисления автоматически проводятся по всей иерархии работ снизу вверх (рис. 5).

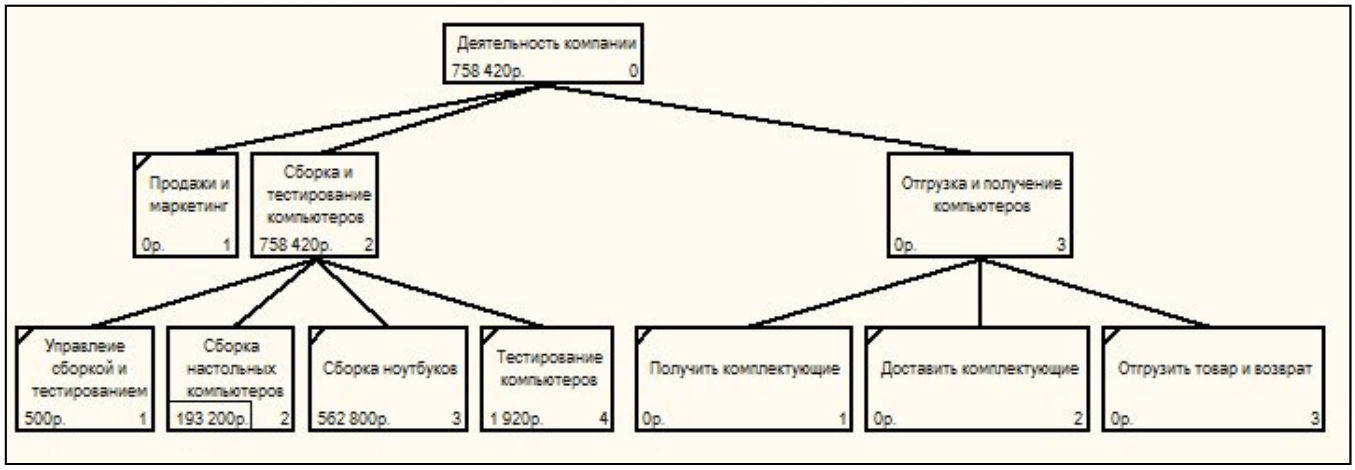


Рисунок 5 - Вычисление затрат родительской работы

Этот достаточно упрощенный принцип подсчета справедлив, если работы выполняются последовательно. Встроенные возможности BPwin позволяют разрабатывать упрощенные модели стоимости, которые, тем не менее, оказываются чрезвычайно полезными при предварительной оценке затрат. Если схема выполнения более сложная (например, работы производятся альтернативно), можно отказаться от подсчета и задать итоговые суммы для каждой работы вручную (Override Decompositions). В этом случае результаты расчетов с нижних уровней декомпозиции будут игнорироваться, и при расчетах на верхних уровнях будет учитываться сумма, заданная вручную. На любом уровне результаты расчетов сохраняются независимо от выбранного режима, поэтому при выключении опции Override Decompositions расчет снизу вверх производится обычным образом.

Результаты стоимостного анализа могут существенно повлиять на очередность выполнения работ. Предположим, что для оценки качества изделия необходимо провести три работы:

- внешний осмотр — стоимость 50 руб.;
- пробное включение — стоимость 150 руб.;
- испытание на стенде — стоимость 300 руб.

Предположим также, что с точки зрения технологии очередность проведения работ несущественна, а вероятность выявления брака одинакова (50%). Пусть необходимо проверить восемь изделий. Если проводить работы в убывающем по стоимости порядке, то затраты на получение готового изделия составят:

300 руб. (испытание на стенде)*8 +150 руб. (пробное включение) *4 + 50 руб. (внешний осмотр) *2 = 3100 руб.

Если проводить работы в возрастающем по стоимости порядке, то на получение готового изделия будет затрачено:

50 руб. (внешний осмотр) *8 +150 руб. (пробное включение) *4 + 300 руб. (испытание на стенде) *2 = 1600 руб.

Следовательно, с целью минимизации затрат первой должна быть выполнена наиболее дешевая работа, затем — средняя по стоимости и в конце — наиболее дорогая.

Результаты стоимостного анализа наглядно представляются на специальном отчете BPwin, настройка которого производится в диалоговом окне Activity Cost Report (меню Tools/Reports/Activity Cost Report) (рис. 6). Отчет позволяет документировать имя, номер, определение и стоимость работ, как суммарную, так и раздельно по центрам затрат.

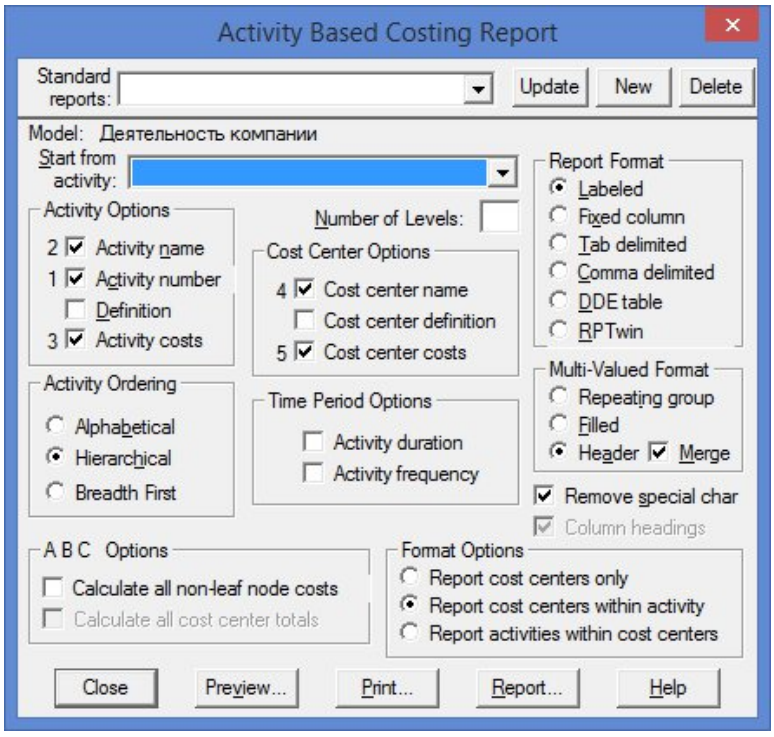


Рисунок 6 - Диалог настройки отчета по стоимости работ

Результаты отображаются и непосредственно на диаграммах. В левом нижнем углу прямоугольника работы может показываться либо стоимость (по умолчанию), либо продолжительность, либо частота проведения работы. Настройка отображения осуществляется в диалоге Model Properties (меню Model/Model Properties), закладка Display (ABC Data, ABC Units).

Контрольные вопросы

1. Каковы цели стоимостного анализа.
2. АВС-анализ.

УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

Рекомендуемая литература

Основная литература:

1. Белов В.В. Проектирование информационных систем: учебник для студ. учреждений ВПО / В.В. Белов, В.И. Чистякова; под ред. В.В. Белова - М.: Издательский центр «Академия», 2013. – 352 с.

Дополнительная литература:

1. Илюшечкин В.М. Основы использования и проектирования баз данных: учеб. пособие. - М.: ЮРАЙТ, 2010.
2. Гвоздева Т.В. Проектирование информационных систем: учеб. пособие / Т.В. Гвоздева, Б.А. Баллод. – Ростов н/Д: Феникс, 2009. – 508 с.
3. Куперштейн, В.И. Microsoft Project 2010 в управлении проектами: В. И. Куперштейн ; ред. А. В. Цветков - СПб.: БХВ-Петербург, 2011.
4. Култыгин О.П. Администрирование баз данных. СУБД MS SQL Server: учеб. пособие / О.П. Култыгин. – М.: Московская финансово-промышленная академия, 2012. – 232 с.
5. Битюцкая Н.И. Курс лекций по дисциплине «Проектный практикум», 2014.
6. Стасышин В.М. Проектирование информационных систем и баз данных. Учебное пособие. НГТУ, 2012, - 100 с. (ЭБС «Университетская библиотека онлайн»).
7. Золотов С.Ю. Проектирование информационных систем и баз данных. Учебное пособие. Эль Контент, 2013, - 88 с. (ЭБС «Университетская библиотека онлайн»).

Методическая литература:

1. методические указания к лабораторным работам;
2. методические указания к самостоятельной работе.

Интернет-ресурсы:

1. http://www.intuit.ru – сайт дистанционного образования в области информационных технологий
2. http://e.lanbook.com – ЭБС издательства «Лань».
3. http://www.biblioclub.ru – университетская библиотека онлайн.
4. http://window.edu.ru – образовательные ресурсы ведущих вузов
5. http://ncfu.ru – сайт СКФУ

Программное обеспечение:

1. Rational Rose,
2. AllFusion Process Modeler (BPWin, ERWin),
3. Microsoft SQL Server 2012,
4. Visual Studio 2012.

Материально-техническое обеспечение

1. Лабораторные и практические занятия проводятся в компьютерных классах, в которых установлено вышеперечисленное программное обеспечение.
2. Лекционный курс проводится в аудиториях, оснащенных проектором.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Пятигорский институт (филиал) СКФУ

Методические указания

для обучающихся по организации и проведению самостоятельной работы
по дисциплине «**ПРОЕКТНЫЙ ПРАКТИКУМ**»
для студентов направления подготовки **09.03.02 Информационные системы и технологии**
направленность (профиль) **Информационные системы и технологии обработки цифрового контента**

Пятигорск, 2025

СОДЕРЖАНИЕ

1. Общие положения 3
2.Цель и задачи самостоятельной работы 4
3.Технологическая карта самостоятельной работы студента 4
4.Порядок выполнения самостоятельной работы студентом 5
4.1. Методические рекомендации по работе с учебной литературой 5
4.2. Методические рекомендации по подготовке к практическим занятиям 6
4.3. Методические рекомендации по самопроверке знаний 7
4.4. Методические рекомендации по написанию научных текстов (докладов, рефератов, эссе, научных статей и т.д.) 8
4.5. Методические рекомендации по подготовке к зачетам 10
Список литературы для выполнения СРС 10

1. Общие положения

Самостоятельная работа – планируемая учебная, учебно-исследовательская, научно-исследовательская работа студентов, выполняемая во внеаудиторное (аудиторное) время по заданию и при методическом руководстве преподавателя, но без его непосредственного участия (при частичном непосредственном участии преподавателя, оставляющем ведущую роль за работой студентов).

Самостоятельная работа студентов (СРС) в ВУЗе является важным видом учебной и научной деятельности студента. Самостоятельная работа студентов играет значительную роль в рейтинговой технологии обучения.

К основным видам самостоятельной работы студентов относятся:

- формирование и усвоение содержания конспекта лекций на базе рекомендованной лектором учебной литературы, включая информационные образовательные ресурсы (электронные учебники, электронные библиотеки и др.);
- написание докладов;
- подготовка к семинарам, практическим и лабораторным работам, их оформление;
- составление аннотированного списка статей из соответствующих журналов по отраслям знаний (педагогических, психологических, методических и др.);
- выполнение учебно-исследовательских работ, проектная деятельность;
- подготовка практических разработок и рекомендаций по решению проблемной ситуации;
- выполнение домашних заданий в виде решения отдельных задач, проведения типовых расчетов, расчетно-компьютерных и индивидуальных работ по отдельным разделам содержания дисциплин и т.д.;
- компьютерный текущий самоконтроль и контроль успеваемости на базе электронных обучающих и аттестующих тестов;
- выполнение курсовых работ (проектов) в рамках дисциплин;
- выполнение выпускной квалификационной работы и др.

Методика организации самостоятельной работы студентов зависит от структуры, характера и особенностей изучаемой дисциплины, объема часов на ее изучение, вида заданий для самостоятельной работы студентов, индивидуальных качеств студентов и условий учебной деятельности.

Процесс организации самостоятельной работы студентов включает в себя следующие этапы:

- подготовительный (определение целей, составление программы, подготовка методического обеспечения, подготовка оборудования);
- основной (реализация программы, использование приемов поиска информации, усвоения, переработки, применения, передачи знаний, фиксирование результатов, самоорганизация процесса работы);
- заключительный (оценка значимости и анализ результатов, их систематизация, оценка эффективности программы и приемов работы, выводы о направлениях оптимизации труда).

2.Цель и задачи самостоятельной работы

Ведущая цель организации и осуществления СРС совпадает с целью обучения студента – формирование универсальных компетенций.

При организации СРС важным и необходимым условием становятся формирование умения самостоятельной работы для приобретения знаний, навыков и возможности организации учебной и научной деятельности. Целью самостоятельной работы студентов является овладение фундаментальными знаниями, профессиональными умениями и навыками деятельности по профилю, опытом творческой, исследовательской деятельности. Самостоятельная работа студентов способствует развитию самостоятельности, ответственности и организованности, творческого подхода к решению проблем учебного и профессионального уровня.

Задачами СРС являются:

- систематизация и закрепление полученных теоретических знаний и практических умений студентов;
- углубление и расширение теоретических знаний;
- формирование умений использовать нормативную, правовую, справочную документацию и специальную литературу;
- развитие познавательных способностей и активности студентов: творческой инициативы, самостоятельности, ответственности и организованности;
- формирование самостоятельности мышления, способностей к саморазвитию, самосовершенствованию и самореализации;
- развитие исследовательских умений;
- использование материала, собранного и полученного в ходе самостоятельной работы и лабораторных занятий.

3.Технологическая карта самостоятельной работы студента

Коды реализуемых компетенций, индикатора(ов)	Вид деятельности студентов	Средства и технологии оценки	Объем часов, в том числе		
			СРС	Контактная работа с преподавателем	Всего
8 семестр					
ИД-1ПК-5, ИД-2 ПК5 ИД-1ПК-7, ИД-2 ПК7 ИД-1ПК-8, ИД-2 ПК8	Самостоятельное изучение литературы	Собеседование	42,84	4,76	47,6
ИД-1ПК-5, ИД-2 ПК5 ИД-1ПК-7, ИД-2 ПК7 ИД-1ПК-8, ИД-2 ПК8	Подготовка к лабораторным работам	Собеседование	18,36	2,04	20,4
ИД-1ПК-5, ИД-2 ПК5 ИД-1ПК-7, ИД-2 ПК7 ИД-1ПК-8, ИД-2 ПК8	Подготовка доклада	Доклад	9	1	10
Итого за 8 семестр			70,2	7,8	78
Итого			70,2	7,8	78

4.Порядок выполнения самостоятельной работы студентом

4.1. Методические рекомендации по работе с учебной литературой

При работе с книгой необходимо подобрать литературу, научиться правильно ее читать, вести записи. Для подбора литературы в библиотеке используются алфавитный и систематический каталоги.

Важно помнить, что рациональные навыки работы с книгой - это всегда большая экономия времени и сил.

Правильный подбор учебников рекомендуется преподавателем, читающим лекционный курс. Необходимая литература может быть также указана в методических разработках по данному курсу.

Изучая материал по учебнику, следует переходить к следующему вопросу только после правильного уяснения предыдущего, описывая на бумаге все выкладки и вычисления (в том числе те, которые в учебнике опущены или на лекции даны для самостоятельного вывода).

При изучении любой дисциплины большую и важную роль играет самостоятельная индивидуальная работа.

Особое внимание следует обратить на определение основных понятий курса. Студент должен подробно разбирать примеры, которые поясняют такие определения, и уметь строить аналогичные примеры самостоятельно. Нужно добиваться точного представления о том, что изучаешь. Полезно составлять опорные конспекты. При изучении материала по учебнику полезно в тетради (на специально отведенных полях) дополнять конспект лекций. Там же следует отмечать вопросы, выделенные студентом для консультации с преподавателем.

Выводы, полученные в результате изучения, рекомендуется в конспекте выделять, чтобы они при перечитывании записей лучше запоминались.

Опыт показывает, что многим студентам помогает составление листа опорных сигналов, содержащего важнейшие и наиболее часто употребляемые формулы и понятия. Такой лист помогает запомнить формулы, основные положения лекции, а также может служить постоянным справочником для студента.

Чтение научного текста является частью познавательной деятельности. Ее цель – извлечение из текста необходимой информации. От того на сколько осознанна читающим собственная внутренняя установка при обращении к печатному слову (найти нужные сведения, усвоить информацию полностью или частично, критически проанализировать материал и т.п.) во многом зависит эффективность осуществляемого действия.

Выделяют ***четыре основные установки в чтении научного текста***:

информационно-поисковый (задача – найти, выделить искомую информацию)

усваивающая (усилия читателя направлены на то, чтобы как можно полнее осознать и запомнить как сами сведения излагаемые автором, так и всю логику его рассуждений)

аналитико-критическая (читатель стремится критически осмыслить материал, проанализировав его, определив свое отношение к нему)

творческая (создает у читателя готовность в том или ином виде – как отправной пункт для своих рассуждений, как образ для действия по аналогии и т.п. – использовать суждения автора, ход его мыслей, результат наблюдения, разработанную методику, дополнить их, подвергнуть новой проверке).

Основные виды систематизированной записи прочитанного:

Аннотирование – предельно краткое связное описание просмотренной или прочитанной книги (статьи), ее содержания, источников, характера и назначения;

Планирование – краткая логическая организация текста, раскрывающая содержание и структуру изучаемого материала;

Тезирование – лаконичное воспроизведение основных утверждений автора без привлечения фактического материала;

Цитирование – дословное выписывание из текста выдержек, извлечений, наиболее существенно отражающих ту или иную мысль автора;

Конспектирование – краткое и последовательное изложение содержания прочитанного.

Конспект – сложный способ изложения содержания книги или статьи в логической последовательности. Конспект аккумулирует в себе предыдущие виды записи, позволяет всесторонне охватить содержание книги, статьи. Поэтому умение составлять план, тезисы, делать выписки и другие записи определяет и технологию составления конспекта.

Методические рекомендации по составлению конспекта:

1. Внимательно прочитайте текст. Уточните в справочной литературе непонятные слова. При записи не забудьте вынести справочные данные на поля конспекта.
2. Выделите главное, составьте план.
3. Кратко сформулируйте основные положения текста, отметьте аргументацию автора.
4. Законспектируйте материал, четко следуя пунктам плана. При конспектировании старайтесь выразить мысль своими словами. Записи следует вести четко, ясно.
5. Грамотно записывайте цитаты. Цитируя, учитывайте лаконичность, значимость мысли.

В тексте конспекта желательно приводить не только тезисные положения, но и их доказательства. При оформлении конспекта необходимо стремиться к емкости каждого предложения. Мысли автора книги следует излагать кратко, заботясь о стиле и выразительности написанного. Число дополнительных элементов конспекта должно быть логически обоснованным, записи должны распределяться в определенной последовательности, отвечающей логической структуре произведения. Для уточнения и дополнения необходимо оставлять поля.

Овладение навыками конспектирования требует от студента целеустремленности, повседневной самостоятельной работы.

4.2. Методические рекомендации по подготовке к практическим занятиям

Для того чтобы практические занятия приносили максимальную пользу, необходимо помнить, что упражнение и решение задач проводятся по вычитанному на лекциях материалу и связаны, как правило, с детальным разбором отдельных вопросов лекционного курса. Следует подчеркнуть, что только после усвоения лекционного материала с определенной точки зрения (а именно с той, с которой он излагается на лекциях) он будет закрепляться на лабораторных занятиях как в результате обсуждения и анализа лекционного материала, так и с помощью решения проблемных ситуаций, задач. При этих условиях студент не только хорошо усвоит материал, но и научится применять его на практике, а также получит дополнительный стимул (и это очень важно) для активной проработки лекции.

При самостоятельном решении задач нужно обосновывать каждый этап решения, исходя из теоретических положений курса. Если студент видит несколько путей решения проблемы (задачи), то нужно сравнить их и выбрать самый рациональный. Полезно до начала вычислений составить краткий план решения проблемы (задачи). Решение проблемных задач или примеров следует излагать подробно, вычисления располагать в строгом порядке, отделяя вспомогательные вычисления от основных. Решения при необходимости нужно сопровождать комментариями, схемами, чертежами и рисунками.

Следует помнить, что решение каждой учебной задачи должно доводиться до окончательного логического ответа, которого требует условие, и по возможности с выводом. Полученный ответ следует проверить способами, вытекающими из существа данной задачи. Полезно также (если возможно) решать несколькими способами и сравнить

полученные результаты. Решение задач данного типа нужно продолжать до приобретения твердых навыков в их решении.

4.3. Методические рекомендации по самопроверке знаний

После изучения определенной темы по записям в конспекте и учебнику, а также решения достаточного количества соответствующих задач на практических занятиях и самостоятельно студенту рекомендуется провести самопроверку усвоенных знаний, ответив на контрольные вопросы по изученной теме.

В случае необходимости нужно еще раз внимательно разобраться в материале.

Иногда недостаточность усвоения того или иного вопроса выясняется только при изучении дальнейшего материала. В этом случае надо вернуться назад и повторить плохо усвоенный материал. Важный критерий усвоения теоретического материала – умение отвечать на вопросы для собеседования.

Вопросы для собеседования

Базовый уровень

Тема 1. Методологии моделирования предметной области. Унифицированный язык моделирования UML.	1. Основные фазы ИТ-проекта. 2. Структурный, функциональный и объектно-ориентированный подходы к анализу и проектированию, сущность и отличия. 3. Функциональное моделирование (IDEF0). 4. Описание бизнес-процессов (IDEF3). 5. Диаграммы потоков данных (DFD). .
Тема 2. Моделирование бизнес-процессов средствами BPwin. Инструментальное средство ERwin.	1. Инструментальная среда BPwin. Построение модели IDEF0. 2. Диаграммы дерева узлов и FEO. 3. Каркас диаграммы. Слияние и расщепление моделей. 4. Создание отчетов в BPwin. 5. Стоимостной анализ в BPwin. 6. Диаграммы потоков данных DFD в BPwin. Метод описания процессов IDEF3.
Тема 3. Проектирование информационных систем в Microsoft SQL Server 2012.	1. Проектирование информационных систем в Microsoft SQL Server 2012 и Visual Studio 2012. 2. Создание и заполнение таблиц. 3. Создание запросов и фильтров. 4. Вычисление при помощи оператора SELECT.
Тема 4. Проектирование информационных систем в Visual Studio 2012.	1. Создание проекта и интерфейса пользователя в Visual Studio 2012.
Тема 5. Управление ИТ-проектом информационной системы.	1. Управление ИТ-проектом информационной системы. Команда ИТ-проекта, структура работ, ресурсы ИТ-проекта. 2. Анализ и управление стоимостью, качеством, временем и рисками ИТ-проекта. 3. Управление ходом выполнения работ ИТ-проекта. 4. Документация ИТ- проекта.
Тема 6. Оценка экономической эффективности ИТ-	1. Оценка экономической эффективности ИТ- проекта. 2. Оценка полных затрат ИТ- проекта, методика Total Cost Ownership (TCO).

проекта.	

Повышенный уровень

Тема 1. Методологии моделирования предметной области. Унифицированный язык моделирования UML.	1. Рациональный процесс управления ИТ-проектами Rational Unified Process (RUP). Нотации языка UML. 2. Основные типы UML-диаграмм: диаграммы прецедентов, диаграммы классов, диаграммы взаимодействия, диаграммы состояний, диаграммы видов деятельности, диаграммы компонентов, диаграммы базы данных, диаграммы развертывания. 3. Взаимосвязи между диаграммами. 4. Этапы проектирования ПО с применением UML.
Тема 2. Моделирование бизнес-процессов средствами BPwin. Инструментальное средство ERwin.	1. Логический и физический уровни представления модели. Основные компоненты диаграммы ERwin. 2. Правила валидации и значения по умолчанию в ERwin. Индексы. Триггеры и хранимые процедуры. Проектирование хранилищ данных. 3. Прямое и обратное проектирование в ERwin. 4. Генерация кода клиентской части с помощью ERwin.
Тема 3. Проектирование информационных систем в Microsoft SQL Server 2012.	1. Встроенные функции. 2. Создание динамических запросов при помощи хранимых процедур в Microsoft SQL Server 2012. 3. Целостность данных. Диаграммы и триггеры.
Тема 4. Проектирование информационных систем в Visual Studio 2012.	1. Создание табличных форм и отчетов.
Тема 5. Управление ИТ-проектом информационной системы.	1. Методология сервис - менеджмента (ITSM). 2. ИТ - сервисы управления изменениями, эксплуатацией, поддержкой и оптимизацией решений ИТ -проекта.
Тема 6. Оценка экономической эффективности ИТ-проекта.	1. Оценка эффективности инвестиций в ИТ-проект, методика Rapid Economic Justification (REJ).

4.4. Методические рекомендации по написанию научных текстов (докладов, рефератов, эссе, научных статей и т.д.)

Перед тем, как приступить к написанию научного текста, важно разобраться, какова истинная цель вашего научного текста - это поможет вам разумно распределить свои силы и время.

Во-первых, сначала нужно определиться с идеей научного текста, а для этого необходимо научиться либо относиться к разным явлениям и фактам несколько критически (своя идея – как иная точка зрения), либо научиться увлекаться какими-то известными идеями, которые нуждаются в доработке (идея – как оптимистическая позиция и направленность на дальнейшее совершенствование уже известного). Во-вторых, научиться организовывать свое время.

Писать следует ясно и понятно, стараясь основные положения формулировать четко и недвусмысленно (чтобы и самому понятно было), а также стремясь структурировать свой текст.

Систематизация и анализ изученной литературы по проблеме исследования позволяют студенту написать работу.

Рабочий вариант текста доклада предоставляется руководителю на проверку. На основе рабочего варианта текста руководитель вместе со студентом обсуждает возможности доработки текста, его оформление.

Структура доклада:

- Введение (не более 3-4 страниц). Во введении необходимо обосновать выбор темы, ее актуальность, очертить область исследования, объект исследования, основные цели и задачи исследования.
- Основная часть состоит из 2-3 разделов. В них раскрывается суть исследуемой проблемы, проводится обзор мировой литературы и источников Интернет по предмету исследования, в котором дается характеристика степени разработанности проблемы и авторская аналитическая оценка основных теоретических подходов к ее решению. Изложение материала не должно ограничиваться лишь описательным подходом к раскрытию выбранной темы. Оно также должно содержать собственное видение рассматриваемой проблемы и изложение собственной точки зрения на возможные пути ее решения.
- Заключение (1-2 страницы). В заключении кратко излагаются достигнутые при изучении проблемы цели, перспективы развития исследуемого вопроса
- Список использованной литературы (не меньше 10 источников), в алфавитном порядке, оформленный в соответствии с принятыми правилами. В список использованной литературы рекомендуется включать работы отечественных и зарубежных авторов, в том числе статьи, опубликованные в научных журналах в течение последних 3-х лет и ссылки на ресурсы сети Интернет.
- Приложение (при необходимости).

Требования к оформлению:

- текст с одной стороны листа;
- шрифт Times New Roman;
- кегль шрифта 14;
- межстрочное расстояние 1,5;
- поля: сверху 2,5 см, снизу – 2,5 см, слева - 3 см, справа 1,5 см;
- реферат должен быть представлен в сброшюрованном виде.

Порядок защиты доклада:

На защиту доклада отводится 5-7 минут времени, в ходе которого студент должен показать свободное владение материалом по заявленной теме. При защите доклада приветствуется использование мультимедиа-презентации.

Доклад оценивается по следующим критериям: соблюдение требований к его оформлению; необходимость и достаточность для раскрытия темы приведенной в тексте доклада информации; умение студента свободно излагать основные идеи, отраженные в докладе; способность студента понять суть задаваемых преподавателем и сокурсниками вопросов и сформулировать точные ответы на них.

Критерии оценки:

Оценка «отлично» выставляется студенту, если в докладе студент исчерпывающе, последовательно, четко и логически стройно излагает материал; свободно справляется с задачами, вопросами и другими видами применения знаний; использует для написания доклада современные научные материалы; анализирует полученную информацию; проявляет самостоятельность при написании доклада.

Оценка «хорошо» выставляется студенту, если качество выполнения доклада достаточно высокое. Студент твердо знает материал, грамотно и по существу излагает его, не допуская существенных неточностей в ответе на вопросы по теме доклада.

Оценка «удовлетворительно» выставляется студенту, если материал доклада излагается частично, но пробелы не носят существенного характера, студент допускает неточности и ошибки при защите доклада, дает недостаточно правильные формулировки, наблюдаются нарушения логической последовательности в изложении материала.

Оценка «неудовлетворительно» выставляется студенту, если он не подготовил доклад или допустил существенные ошибки. Студент неуверенно излагает материал доклада, не отвечает на вопросы преподавателя.

Описание шкалы оценивания

Максимально возможный балл за весь текущий контроль устанавливается равным 55. Текущее контрольное мероприятие считается сданным, если студент получил за него не менее 60% от установленного для этого контроля максимального балла. Рейтинговый балл, выставляемый студенту за текущее контрольное мероприятие, сданное студентом в установленные графиком контрольных мероприятий сроки, определяется следующим образом:

Уровень выполнения контрольного задания	Рейтинговый балл (в % от максимального балла за контрольное задание)
Отличный	**100**
Хороший	**80**
Удовлетворительный	**60**
Неудовлетворительный	**0**

Темы эссе (рефератов, докладов, сообщений)

Базовый уровень

Тема 1. Введение в проектный менеджмент

1. Оптимизация и реинжиниринг инженерного проекта
2. Инновационные проекты инженерных решений.
3. Тестирование инженерного проекта
4. Стандартизация качества инженерного проекта

Тема 2. Методы и средства проектного менеджмента

5. Формализация требований к инженерному проекту
6. Понятие конфигурационного управления проектом
7. Управление версиями инженерного проекта
8. Управление сборками при разработке инженерного проекта
9. Средства версионного контроля инженерного проекта

Тема 3. Менеджмент этапов жизненного цикла инженерного проекта

10. Диаграммные техники в работе со знаниями
11. Диаграммы использования в работе со знаниями
12. Карты памяти для проекта инженерного решения
13. Основные принципы MSF

Тема 4. Технологии проектного менеджмента в решении инженерных задач

14. Инновационные инженерные проекты.
15. Тестирование информационной системы для инженерного проекта
16. Стандартизация качества информационных систем для инженерного проекта
17. Методы обеспечения качества информационных систем для инженерного проекта

Тема 5. Методология проектного менеджмента инженерных задач

18. Понятие тестирования информационной системы для инженерного проекта
19. Масштабирование команды MSF. Модель процесса инженерного проекта
20. Разработка инженерного проекта. Понятие CMMI.

Тема 6. Методы управления качеством инженерного проекта

21. Уровни зрелости процессов по CMMI
22. Области усовершенствования в методологии CMMI.
23. Общее описание "гибких" методов разработки инженерного проекта

Тема 7. Обзор современных технологий менеджмента инженерных задач

24. Верификация, валидация и аудит информационных систем для инженерного проекта
25. Метрики качества программного обеспечения для инженерного проекта
26. Стандартный метод оценки значений показателей качества проекта

Тема 8. Гибкая методология управления проектами: Agile, Scrum, Kanban, XP, APF

27. Управление качеством инженерного проекта
28. Extreme Programming: общее описание, основные принципы решения
29. Разработка информационных систем для инженерного проекта. Scrum
30. Обзор технологии Microsoft Visual Studio Team System

Повышенный уровень

Тема 1. Введение в проектный менеджмент

1. Верификация и валидация программных продуктов
2. Понятие тестирования программных средств для инженерного проекта
3. Методы верификации объектно-ориентированных программ
4. Качество и надежность программного обеспечения для инженерного проекта

Тема 2. Методы и средства проектного менеджмента

5. Профили открытых информационных систем
6. Функциональные и технологические стандарты инженерного проекта
7. Многопользовательская информационная система для инженерного проекта

Тема 3. Менеджмент этапов жизненного цикла инженерного проекта

8. Рабочий проект информационной системы. Дисциплина обязательств.
9. Технический проект информационной системы для инженерного проекта
10. Управление проектом информационной системы

Тема 4. Технологии проектного менеджмента в решении инженерных задач

11. Принципы организации проектирования и программных комплексов
12. Задачи обеспечения качества программных компонентов для инженерного проекта
13. Методы исследования качества программных компонентов
14. Задачи обеспечения надежности программных компонентов

Тема 5. Методология проектного менеджмента инженерных задач

15. Методы исследования надежности программных компонентов
16. Экономико-правовые основы разработки инженерного проекта
17. Автоматическое тестирование инженерного проекта.

Тема 6. Методы управления качеством инженерного проекта

18. Открытая архитектура информационных систем для инженерного проекта
19. Системная инженерия: точка зрения и характеристики точек зрения
20. Управление качеством инженерного проекта
21. Оптимизация и реинжиниринг инженерного проекта

Тема 7. Обзор современных технологий менеджмента инженерных задач

22. Архитектура программных комплексов для инженерного проекта
23. Стандарты проектирования программного обеспечения
24. Стандарты разработки программного обеспечения для инженерного проекта
25. Методы разработки программных комплексов для инженерного проект

Тема 8. Гибкая методология управления проектами: Agile, Scrum, Kanban, XP, APF

26. Методы оценки сложности алгоритмов и программ
27. Применение инструментов разработки информационных систем
28. Управление требованиями к информационной системе
29. Виды требований к информационной системе для инженерного проект
30. Моделирование структуры информационных систем, виды моделей
31. Объектно-ориентированное моделирование инженерного решения

32. Принципы верификации и тестирования инженерного проекта
33. Верификация и валидация программных продуктов
34. Понятие тестирования программных средств
35. Методы верификации объектно-ориентированных программ
36. Качество и надежность программного обеспечения для инженерного проект
37. Метрики качества инженерного проекта
38. Стандартный метод оценки значений показателей качества инженерного проекта
39. Управление качеством инженерного проекта
40. Оптимизация и реинжиниринг инженерного проекта

4.5. Методические рекомендации по подготовке к зачетам

Процедура зачета как отдельное контрольное мероприятие не проводится, оценивание знаний обучающегося происходит по результатам текущего контроля.

Зачет выставляется по результатам работы в семестре, при сдаче всех контрольных точек, предусмотренных текущим контролем успеваемости. Если по итогам семестра обучающийся имеет от 33 до 60 баллов, ему ставится отметка «зачтено». Обучающемуся, имеющему по итогам семестра менее 33 баллов, ставится отметка «не зачтено».

Количество баллов за зачет (Sзач) при различных рейтинговых баллах по дисциплине по результатам работы в семестре

Рейтинговый балл по дисциплине по результатам работы в семестре (*Rсем*)	**Количество баллов за зачет (*S*зач)**
$50 \leq Rсем \leq 60$	**40**
$39 \leq Rсем < 50$	**35**
$33 \leq Rсем < 39$	**27**
$Rсем < 33$	**0**

Контроль самостоятельной работы студентов

Контроль самостоятельной работы проводится преподавателем в аудитории.

Предусмотрены следующие виды контроля: собеседование, оценка выполнения доклада и его презентации.

Подробные критерии оценивания компетенций приведены в Фонде оценочных средств для проведения текущей и промежуточной аттестации.

Список литературы для выполнения СРС

Основная литература:

2. Белов В.В. Проектирование информационных систем: учебник для студ. учреждений ВПО / В.В. Белов, В.И. Чистякова; под ред. В.В. Белова - М.: Издательский центр «Академия», 2013. – 352 с.

Дополнительная литература:

8. Илюшечкин В.М. Основы использования и проектирования баз данных: учеб. пособие. - М.: ЮРАЙТ, 2010.

9. Гвоздева Т.В. Проектирование информационных систем: учеб. пособие / Т.В. Гвоздева, Б.А. Баллод. – Ростов н/Д: Феникс, 2009. – 508 с.
10. Куперштейн, В.И. Microsoft Project 2010 в управлении проектами: В. И. Куперштейн ; ред. А. В. Цветков - СПб.: БХВ-Петербург, 2011.
11. Култыгин О.П. Администрирование баз данных. СУБД MS SQL Server: учеб. пособие / О.П. Култыгин. – М.: Московская финансово-промышленная академия, 2012. – 232 с.
12. Битюцкая Н.И. Курс лекций по дисциплине «Проектный практикум», 2014.
13. Стасышин В.М. Проектирование информационных систем и баз данных. Учебное пособие. НГТУ, 2012, - 100 с. (ЭБС «Университетская библиотека онлайн»).
14. Золотов С.Ю. Проектирование информационных систем и баз данных. Учебное пособие. Эль Контент, 2013, - 88 с. (ЭБС «Университетская библиотека онлайн»).

Методическая литература:

3. методические указания к лабораторным работам;
4. методические указания к самостоятельной работе.

Интернет-ресурсы:

6. http://www.intuit.ru – сайт дистанционного образования в области информационных технологий
7. http://e.lanbook.com – ЭБС издательства «Лань».
8. http://www.biblioclub.ru – университетская библиотека онлайн.
9. http://window.edu.ru – образовательные ресурсы ведущих вузов
10. http://ncfu.ru – сайт СКФУ

Программное обеспечение:

1	Альт Рабочая станция 10
2	Альт Рабочая станция K
3	Альт «Сервер»
4	Пакет офисных программ - Р7-Офис