МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Институт сервиса, туризма и дизайна (филиал) СКФУ в г. Пятигорске

УТВЕРЖДАЮ
Зав. кафедрой СУиИТ
_____ И.М. Першин
«__» _____ 2020_ г.

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ ПО ДИСЦИПЛИНЕ

ПРОЕКТНЫЙ ПРАКТИКУМ

Направление подготовки	09.03.02 Информационные системы и технологии
Профиль подготовки	Информационные системы и технологии
Квалификация выпускника	Бакалавр
Форма обучения	очная

Пятигорск, 2020

СОДЕРЖАНИЕ

1. Цель и задачи освоения дисциплины

Целью изучения дисциплины является формирование набора общепрофессиональных и профессиональных компетенций 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. Место дисциплины в структуре образовательной программы

Дисциплина «Проектный практикум» относится к дисциплинам по выбору вариативной части блока Б1 учебного плана подготовки бакалавров направления 09.03.02 Информационные системы и технологии. Ее освоение происходит в 8 семестре.

3. Связь с предшествующими дисциплинами

Содержание данной учебной дисциплины опирается на знание дисциплин: Интеллектуальные системы и технологии, Методы и средства проектирования информационных систем и технологий, Безопасность информационных систем, Основы Web-технологий, Инструментальные средства информационных систем, Базы данных в распределенных системах обработки информации, Основы численного моделирования, Численные методы в научных расчетах.

4. Связь с последующими дисциплинами

Знания, полученные при изучении данной дисциплины, необходимы дисциплин: Преддипломная практика, Защита выпускной квалификационной работы, включая подготовку к процедуре защиты и процедуру защиты.

5. Перечень планируемых результатов обучения по дисциплине (модулю), соотнесённых с планируемыми результатами освоения образовательной программы

5.1. Наименование компетенций

Код	Формулировка:
ОПК-6	способностью выбирать и оценивать способ реализации информационных систем и устройств (программно-, аппаратно- или программно-аппаратно-) для решения поставленной задачи
ПК-6	способностью оценивать надежность и качество функционирования объекта проектирования
ПК-7	способностью осуществлять сертификацию проекта по стандартам качества
ПК-14	способностью использовать знание основных закономерностей

	функционирования биосферы и принципов рационального природопользования для решения задач профессиональной деятельности
ПК-21	способностью осуществлять организацию контроля качества входной информации
ПК-30	способностью поддерживать работоспособность информационных систем и технологий в заданных функциональных характеристиках и соответствии критериям качества
ПК-37	способностью выбирать и оценивать способ реализации информационных систем и устройств (программно-, аппаратно- или программно-аппаратно-) для решения поставленной задачи

6. НАИМЕНОВАНИЕ ЛАБОРАТОРНЫХ РАБОТ

№ Темы	Наименование тем дисциплины, их краткое содержание	Объем часов	Интерактивная форма проведения
3 семестр			
Тема 6 **Проектирование информационных систем в Microsoft SQL Server 2012. Часть2.**			
6	Лабораторная работа №1. **«Выполнение индивидуальных заданий по проектированию информационных систем в Microsoft SQL Server 2012.»** *Содержание: Научиться создавать в Microsoft SQL Server такие объекты, как таблицы, запросы, фильтры, хранимые процедуры, пользовательские функции, диаграммы и триггеры.*	3	Компьютерные симуляции
Тема 7. **Проектирование информационных систем в Visual Studio 2012.**			
7	Лабораторная работа №2. **«Выполнение индивидуальных заданий по проектированию информационных систем в Visual Studio 2012. Создание ленточных и табличных форм для работы с базами данных.»** *Содержание: 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).*	3	Компьютерные симуляции
7	Лабораторная работа №3. **«Выполнение индивидуальных заданий по проектированию информационных систем в Visual Studio 2012. Создание отчетов и диаграмм.»** *Содержание: Научиться создавать создавать отчеты и диаграммы.*	3	Компьютерные симуляции
Тема 8. **Управление ИТ-проектом информационной системы.**			
8	Лабораторная работа №4. **«Подготовка документации IT проекта.»** *Содержание: Научиться разрабатывать организационно-техническую и эксплуатационную документацию.*	3	
Тема 9. **Оценка экономической эффективности ИТ- проекта.**			
10	Лабораторная работа №5. **«Расчет экономической эффективности проекта.»** *Содержание: Научиться проводить стоимостный анализ.*	1,5	
	Итого за 8 семестр	13,5	**9**
	Итого	**13,5**	**9**

7. СОДЕРЖАНИЕ ЛАБОРАТОРНЫХ РАБОТ

Лабораторная работа № 1. «Выполнение индивидуальных заданий по проектированию информационных систем в Microsoft SQL Server 2012.»

Форма проведения: лабораторная работа (3 часа).

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.

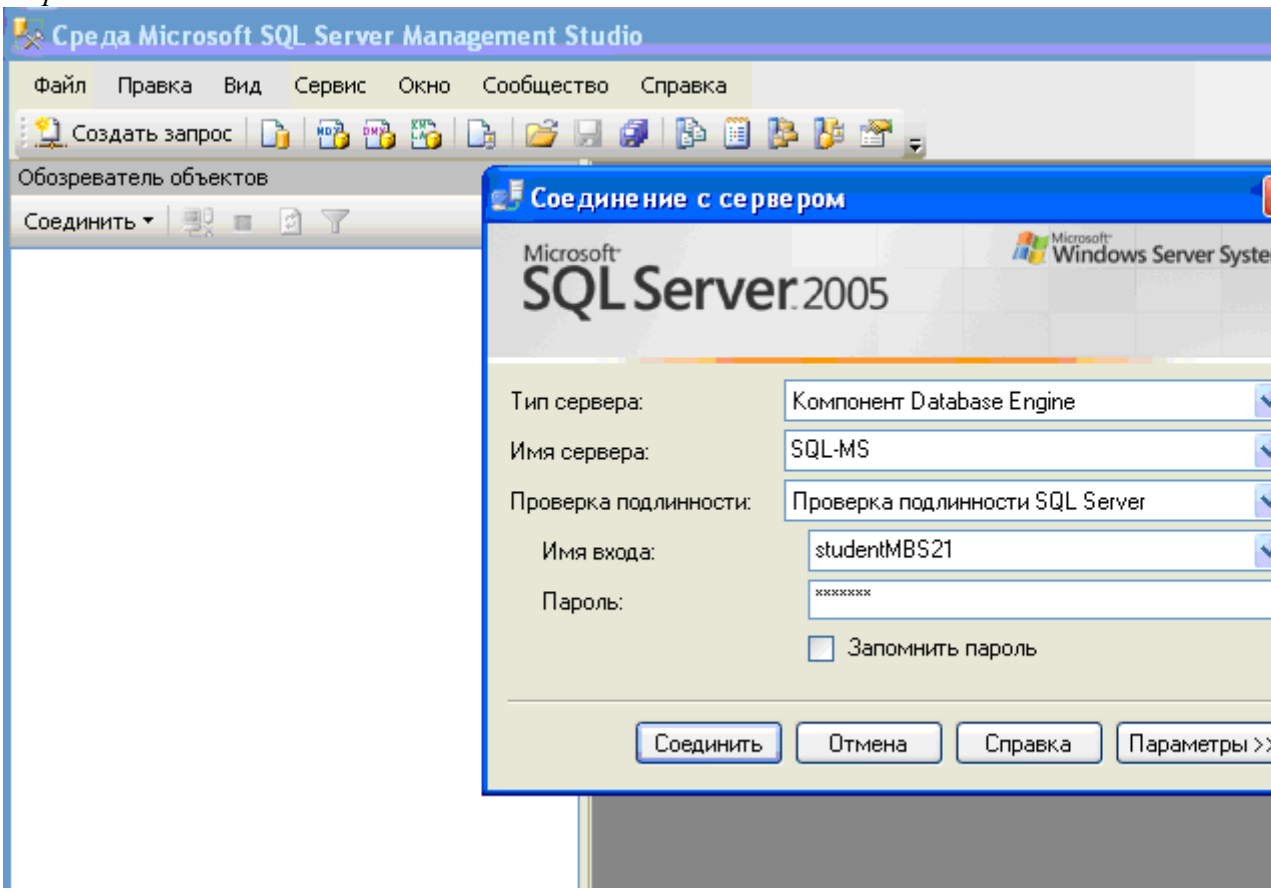
Начало работы в Microsoft SQL Server Management Studio

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

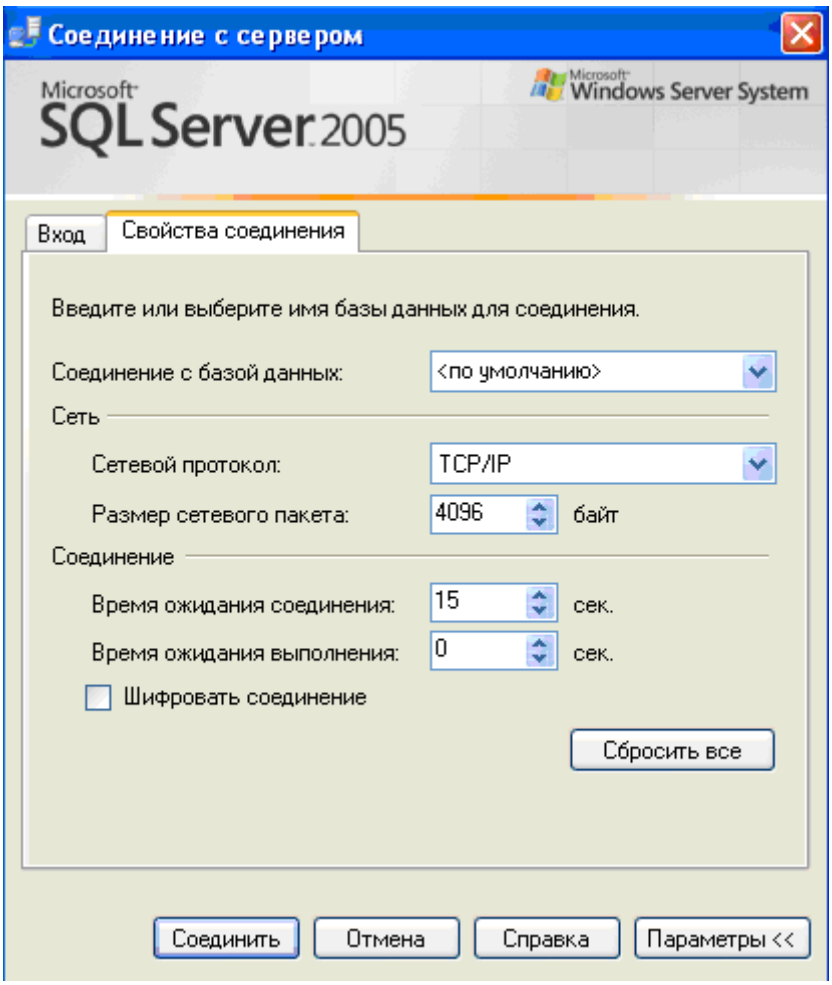
- *Тип сервера:* **Компонент Database Engine**
- *Имя сервера:* **SQL-MS.**
- Под таким именем в домене fizmat.vspu.ru. доступна машина, на которой установлены серверные компоненты MS SQL Server 2005. Можно попробовать

выбрать сервер из выпадающего списка серверов. Можно также обратиться к этой машине по IP-адресу 192.168.10.152 из локальной сети.

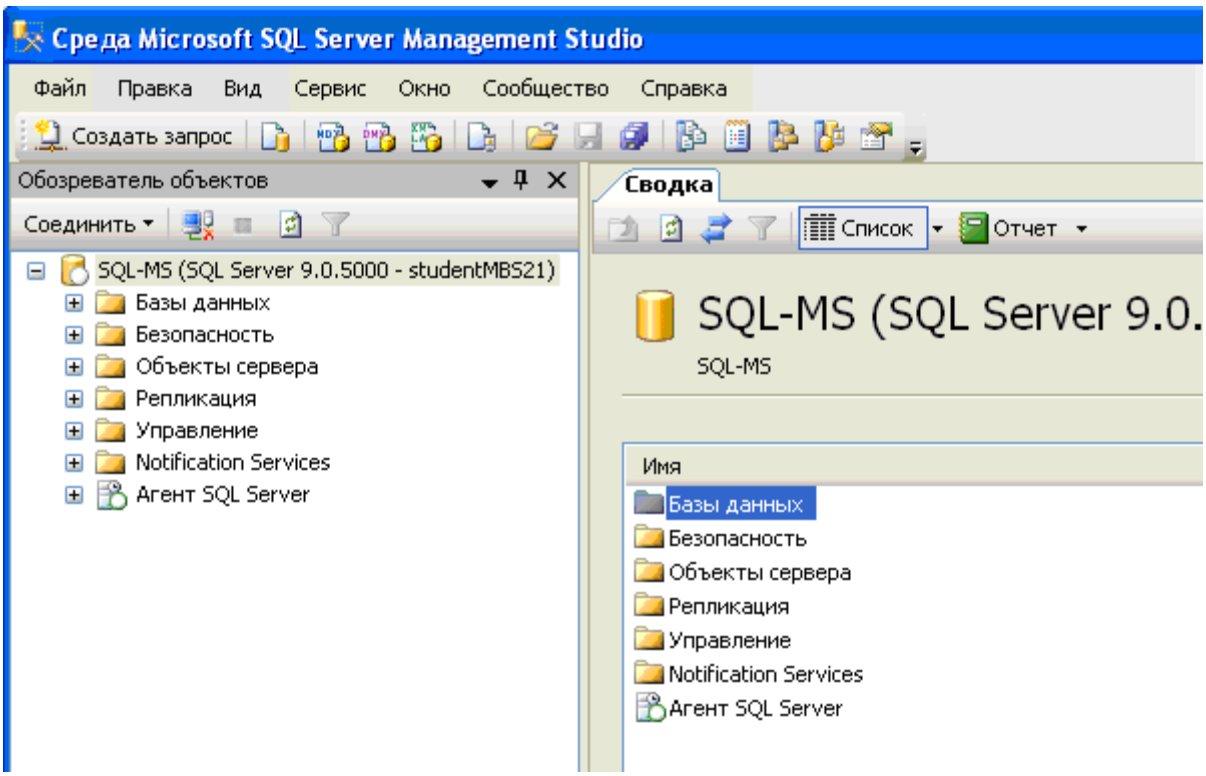
- *Проверка подлинности:* **Проверка подлинности SQL Server.**
- Такая настройка позволяет создавать пользователей данного экземпляра SQL Server независимо от компьютера, с которого производится вход.
- *Имя входа:* **studentMBS21.**
- *Пароль:* **student.**



- 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):

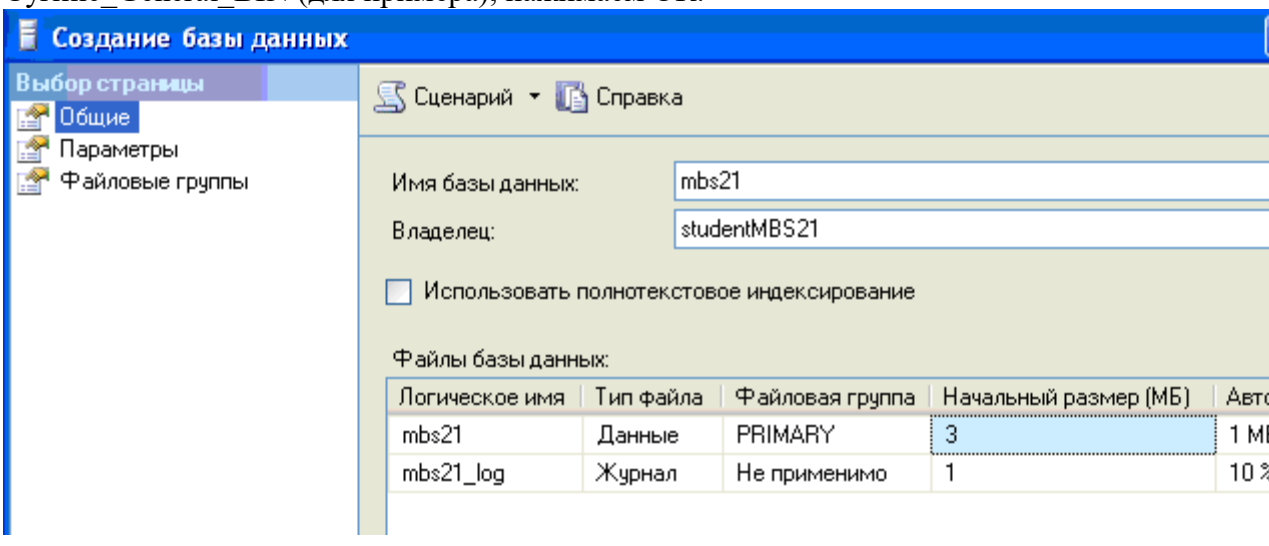


– Рисунок 3. Подключение к SQL - серверу установлено

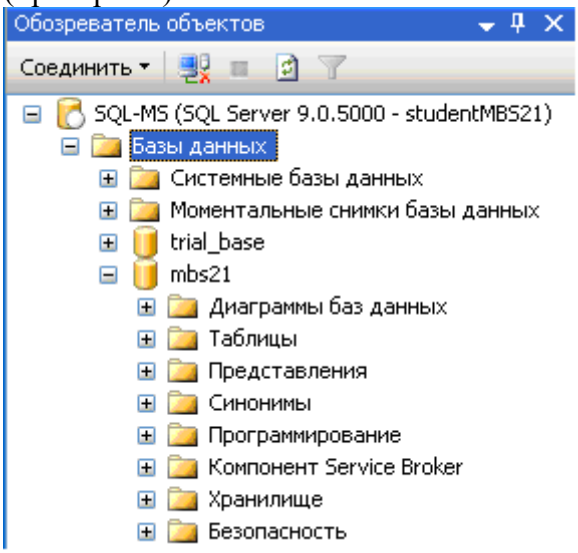
– Среда MS SQL Management Studio предоставляет удобный инструментарий для создания, редактирования, заполнения баз данных. Но настоящие профессионалы в своей работе редко пользуются этой средой, а для выполнения своих задач используют SQL-запросы. Мы будем пользоваться, когда это удобно и наглядно, графическим 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!):



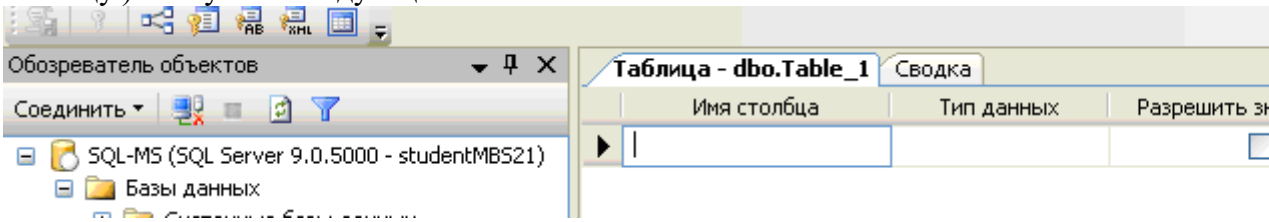
- Рисунок 5. Обозреватель объектов

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:

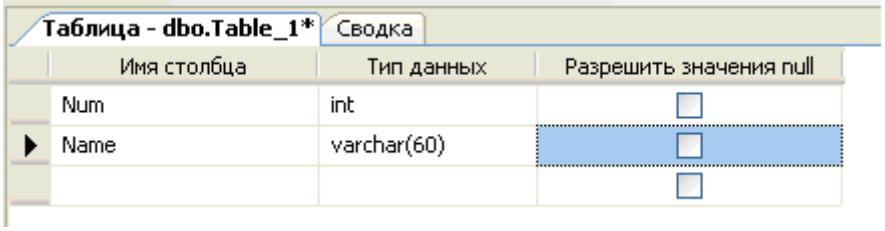


Рисунок 7.

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

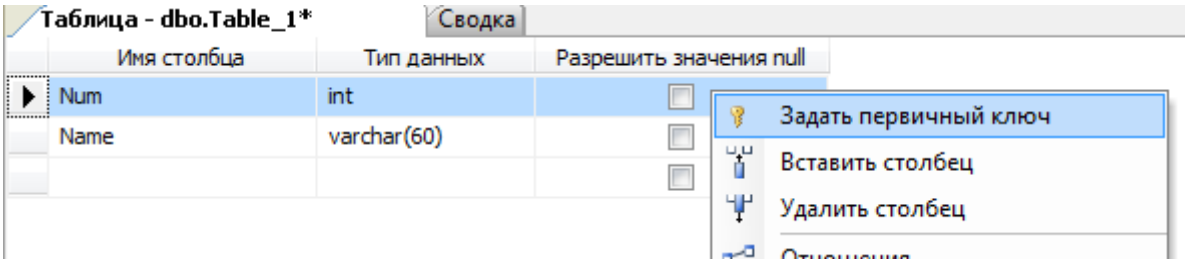


Рисунок 8.

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»):

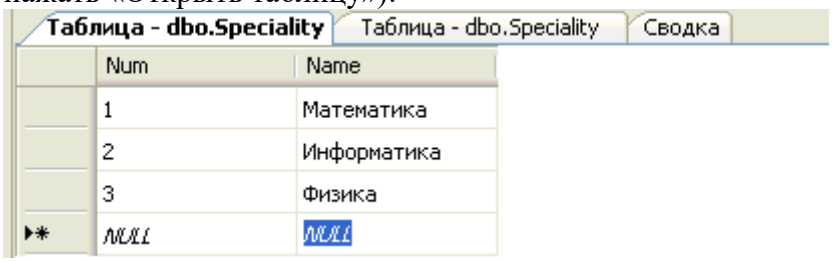


Рисунок 9.

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):

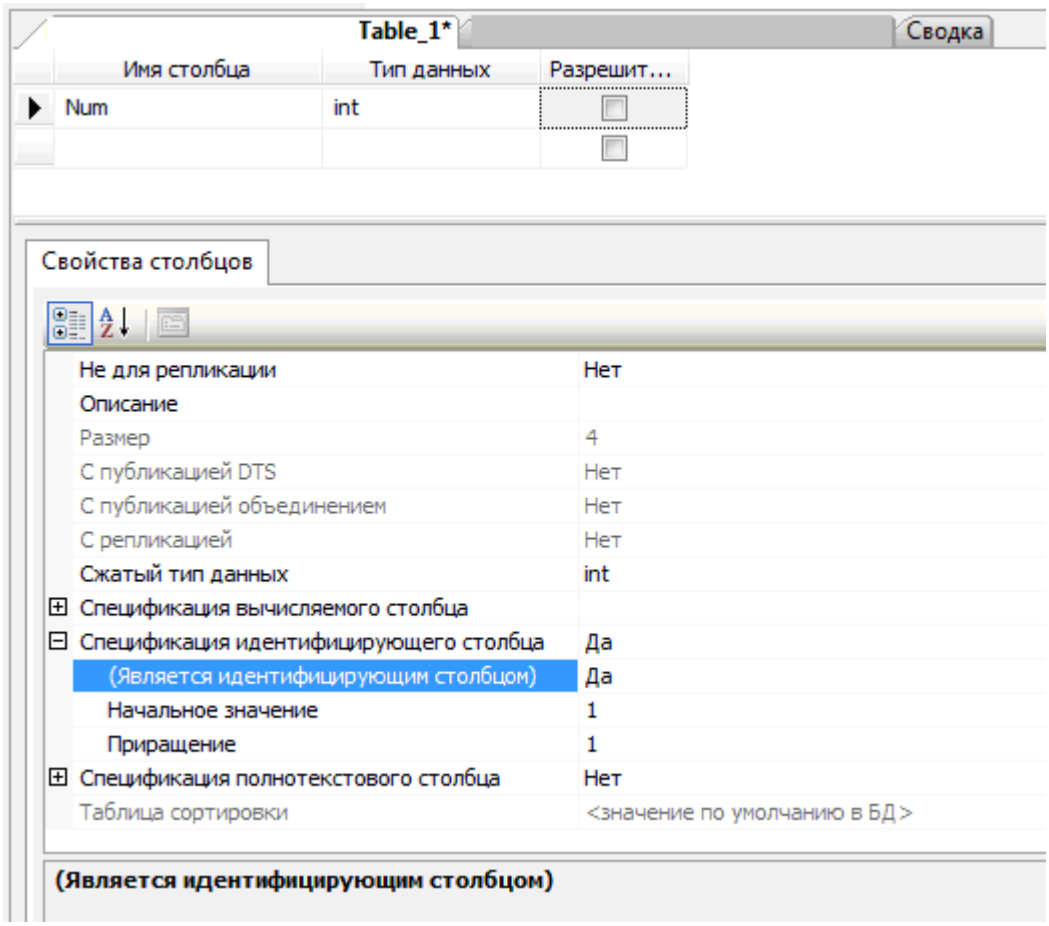


Рисунок 10. Определение свойств идентифицирующего столбца

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»:

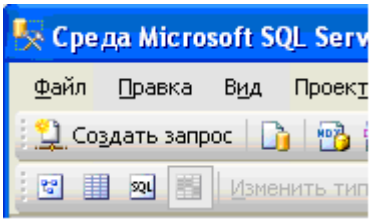


Рисунок 11.

- 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.

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

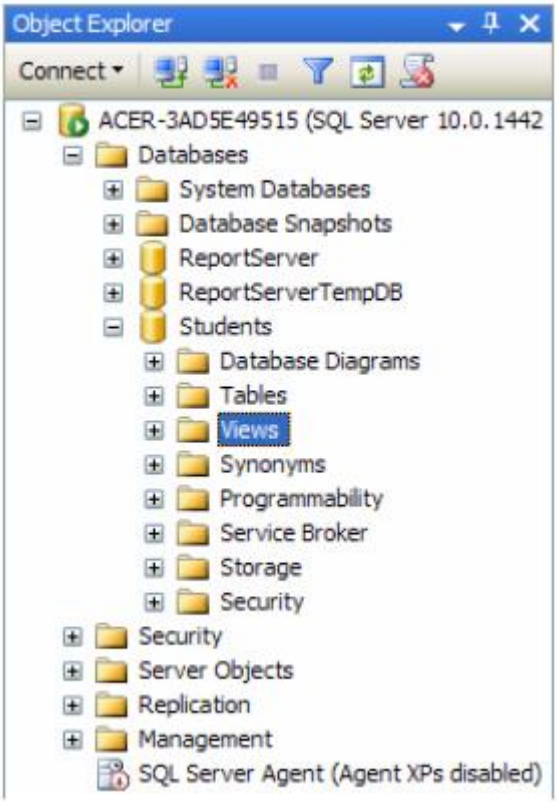


Рис. 8.1.

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).

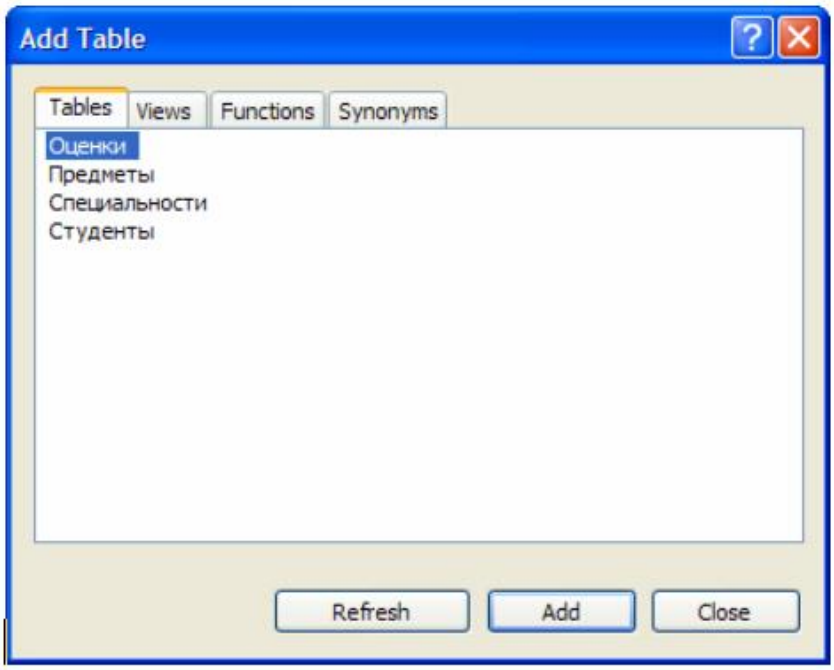


Рис. 8.2.

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).

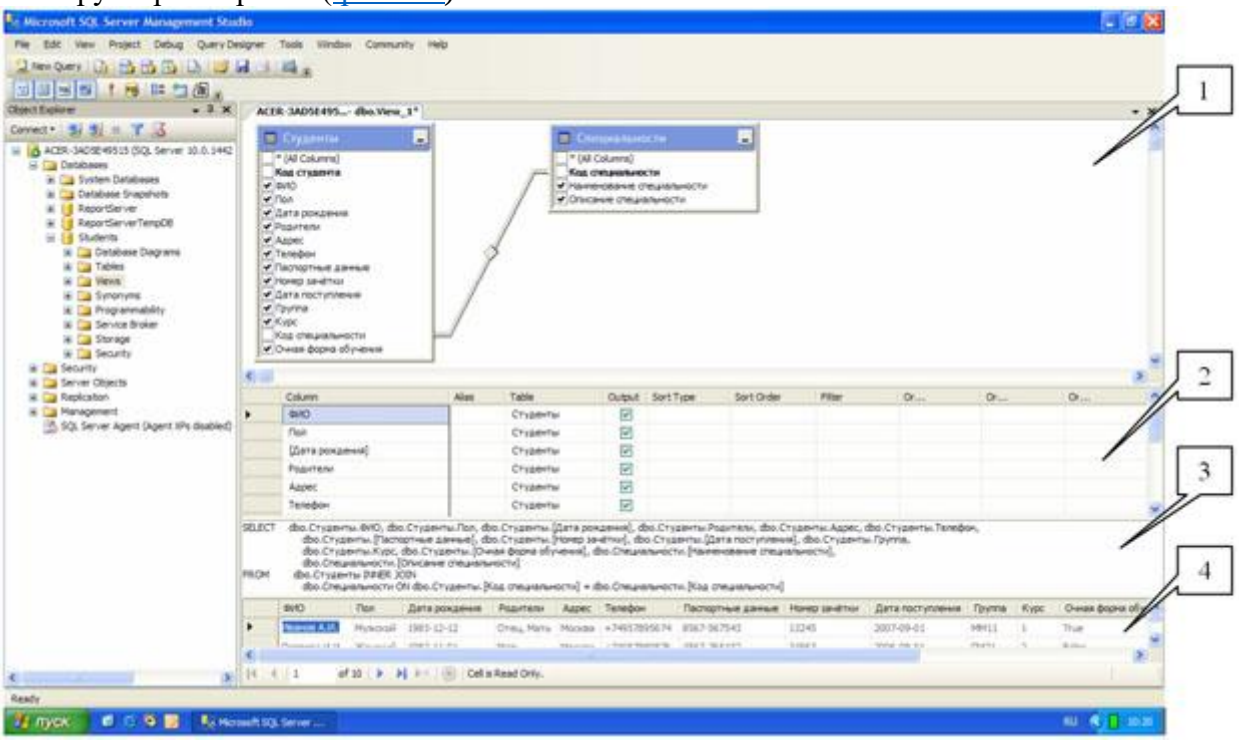


Рис. 8.3.

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).

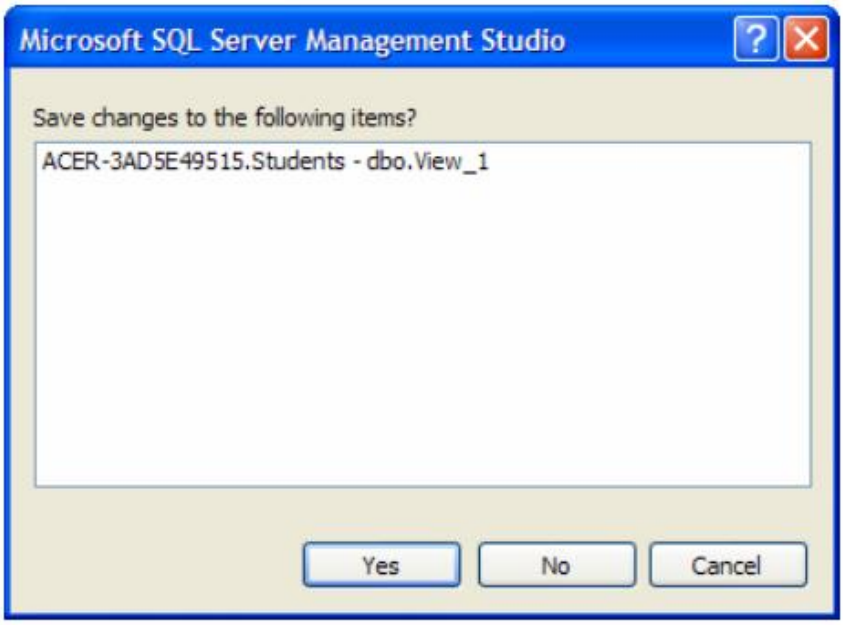


Рис. 8.4.

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

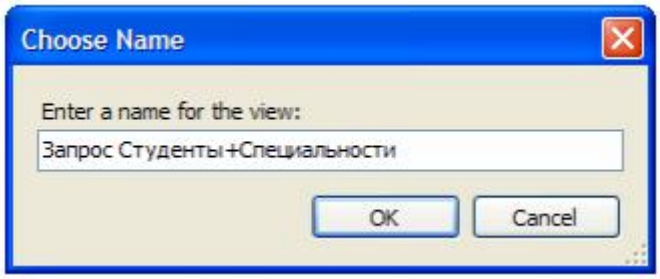


Рис. 8.5.

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).

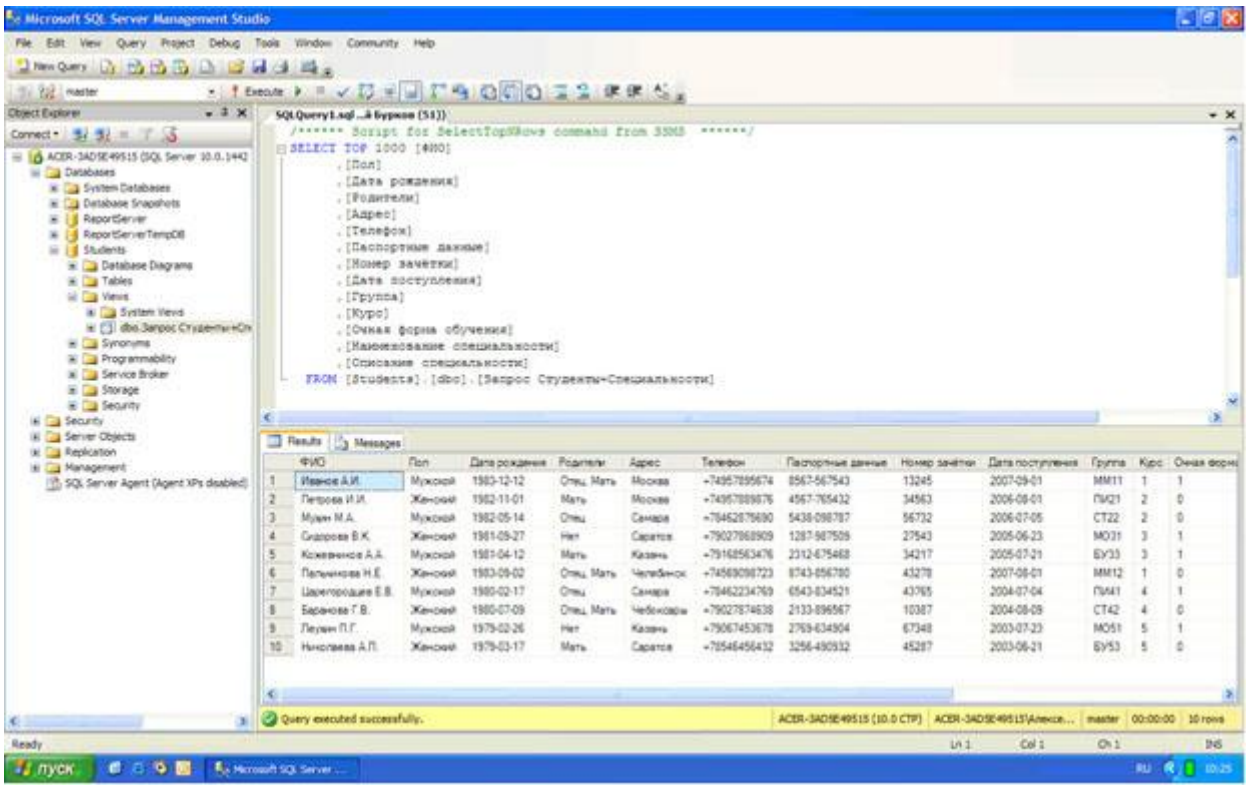


Рис. 8.6.

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.

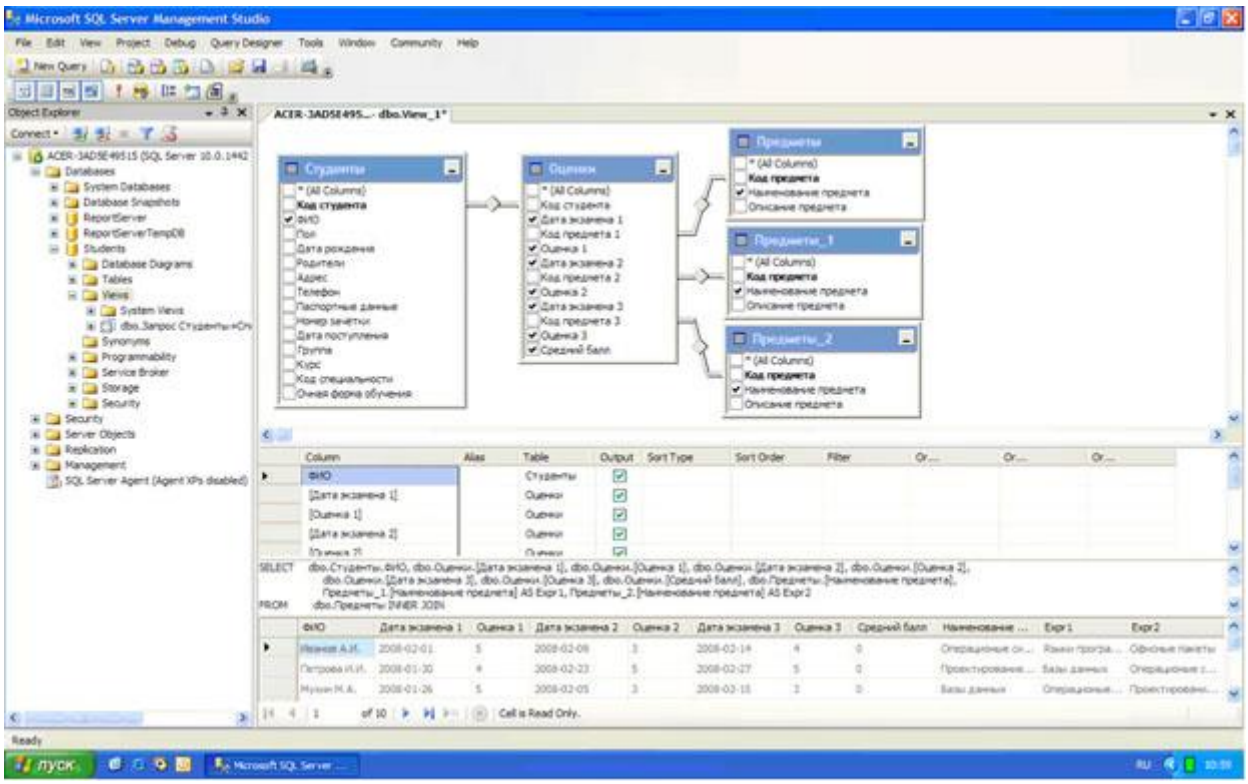
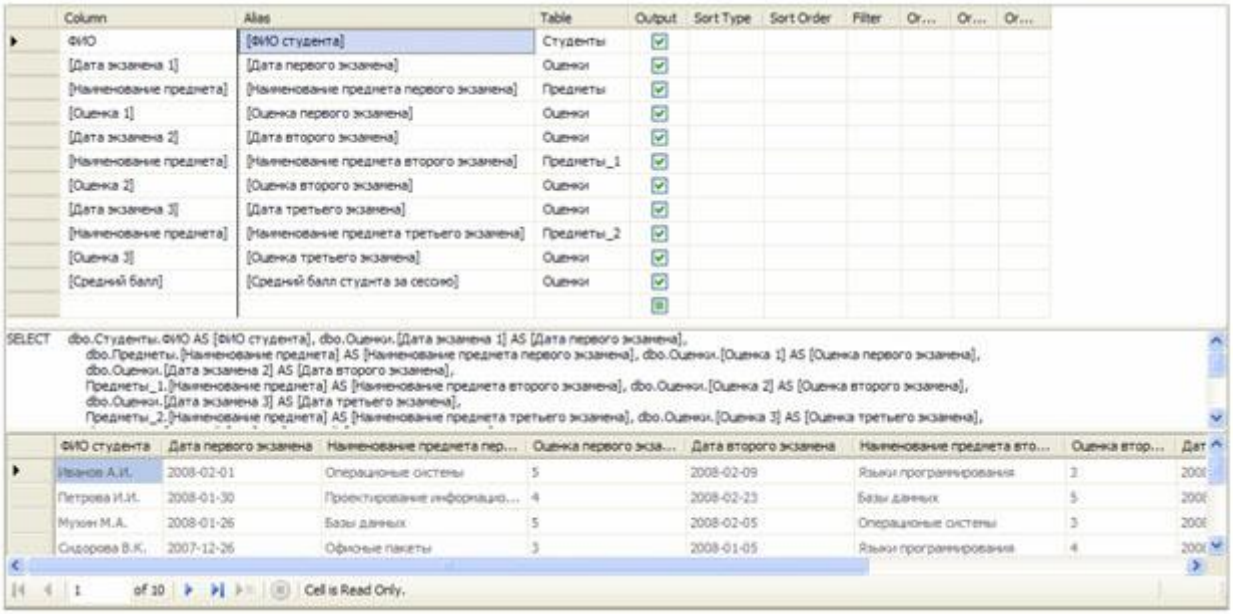


Рис. 8.7.

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).

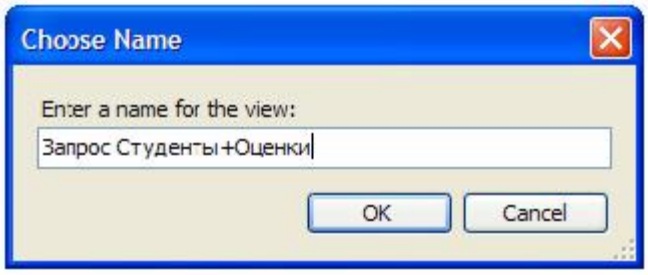
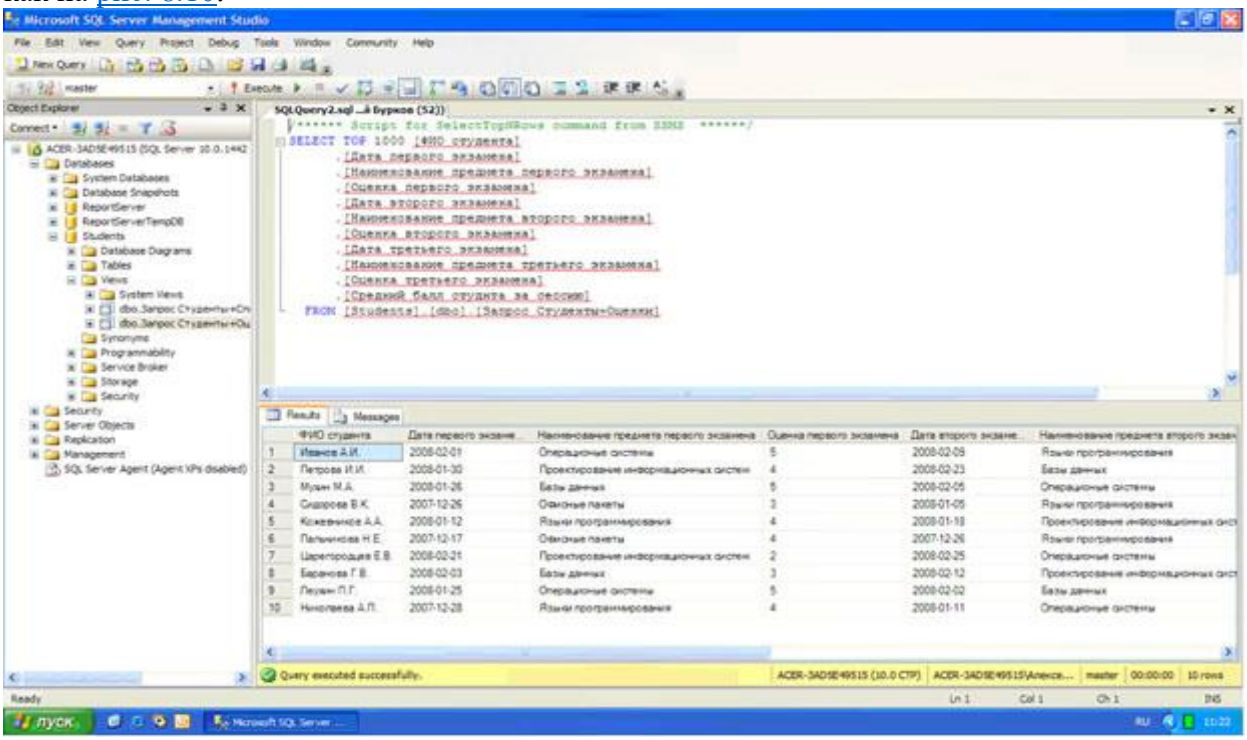


Рис. 8.9.

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"**.

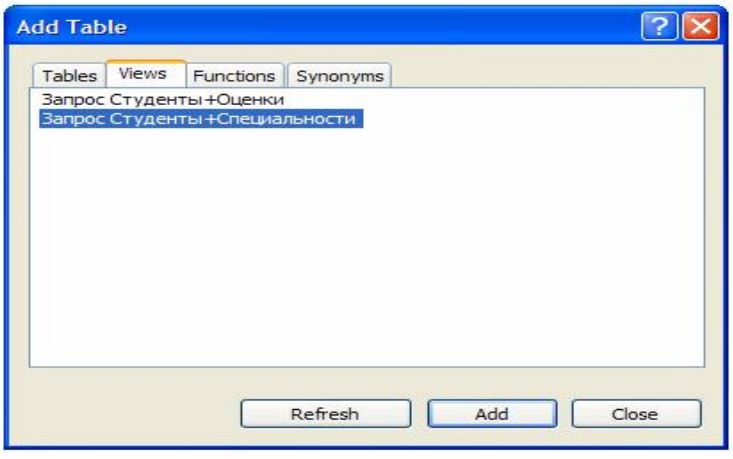
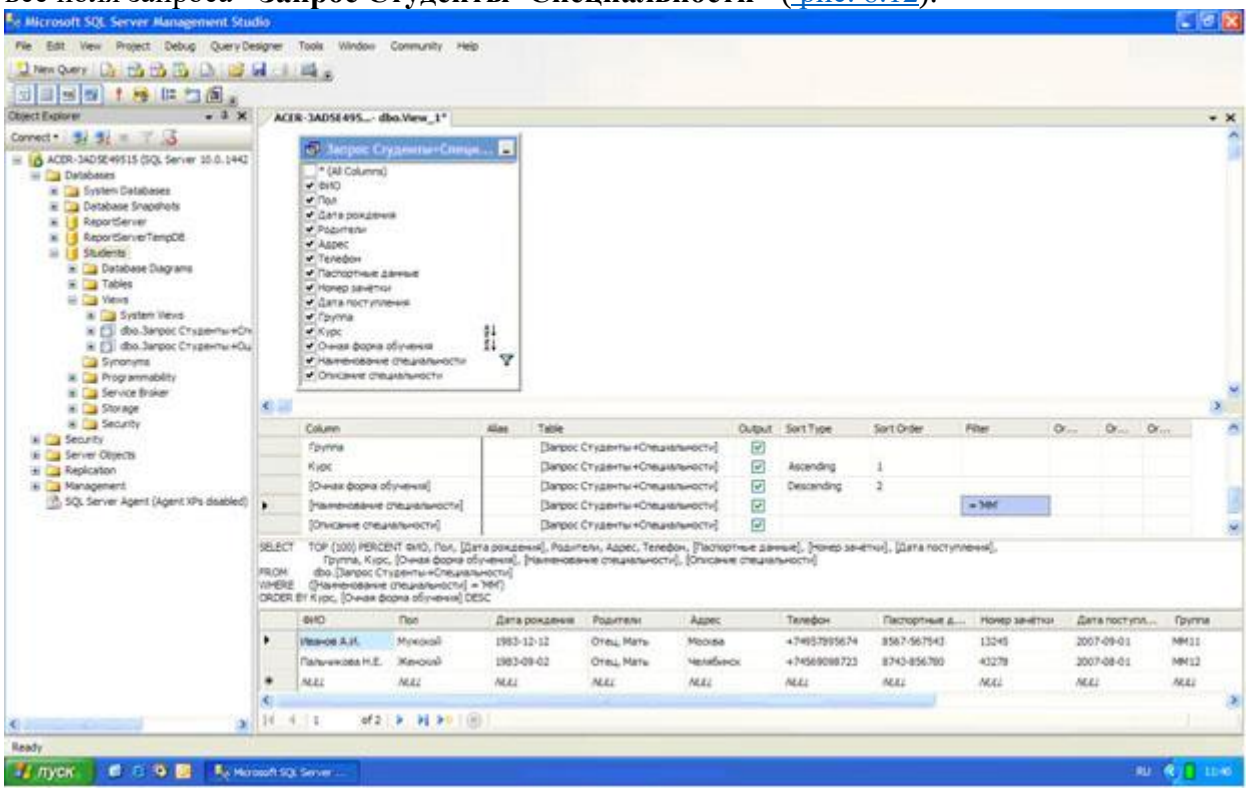


Рис. 8.11.

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"**.

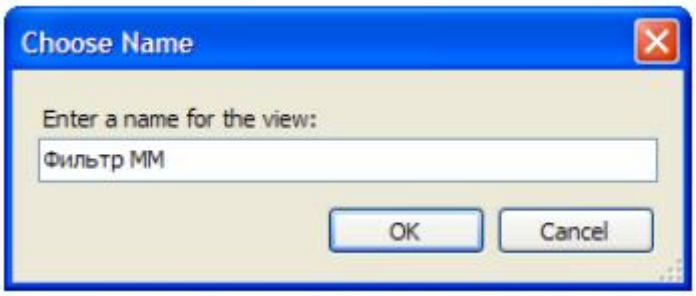
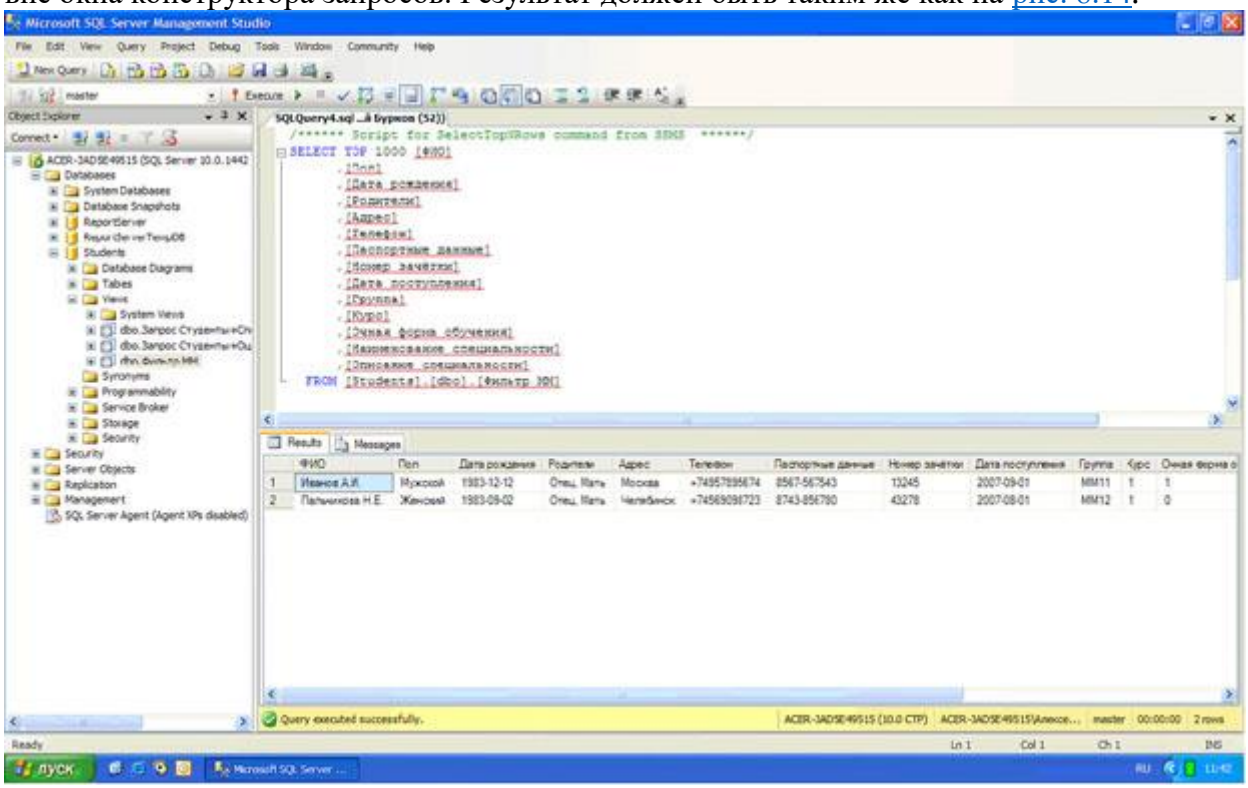


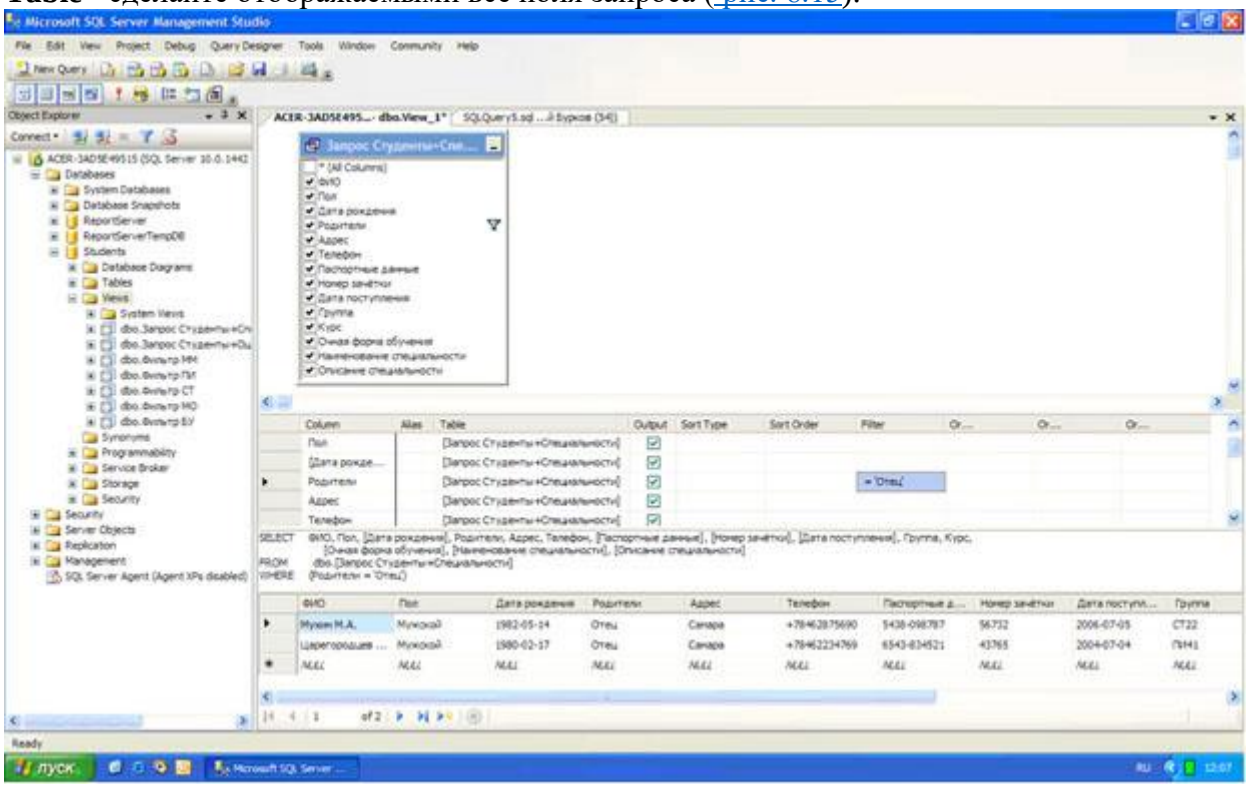
Рис. 8.13.
Фильтр **"Фильтр ММ"** появится в обозревателе объектов. Выполните созданный фильтр вне окна конструктора запросов. Результат должен быть таким же как на рис. 8.14.



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).

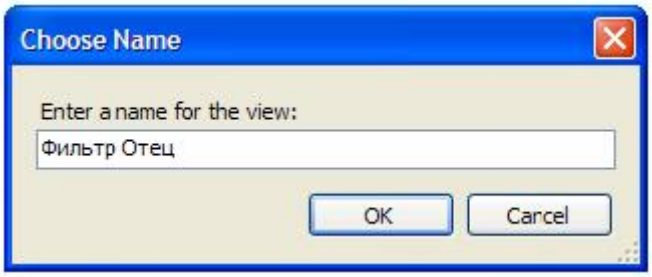
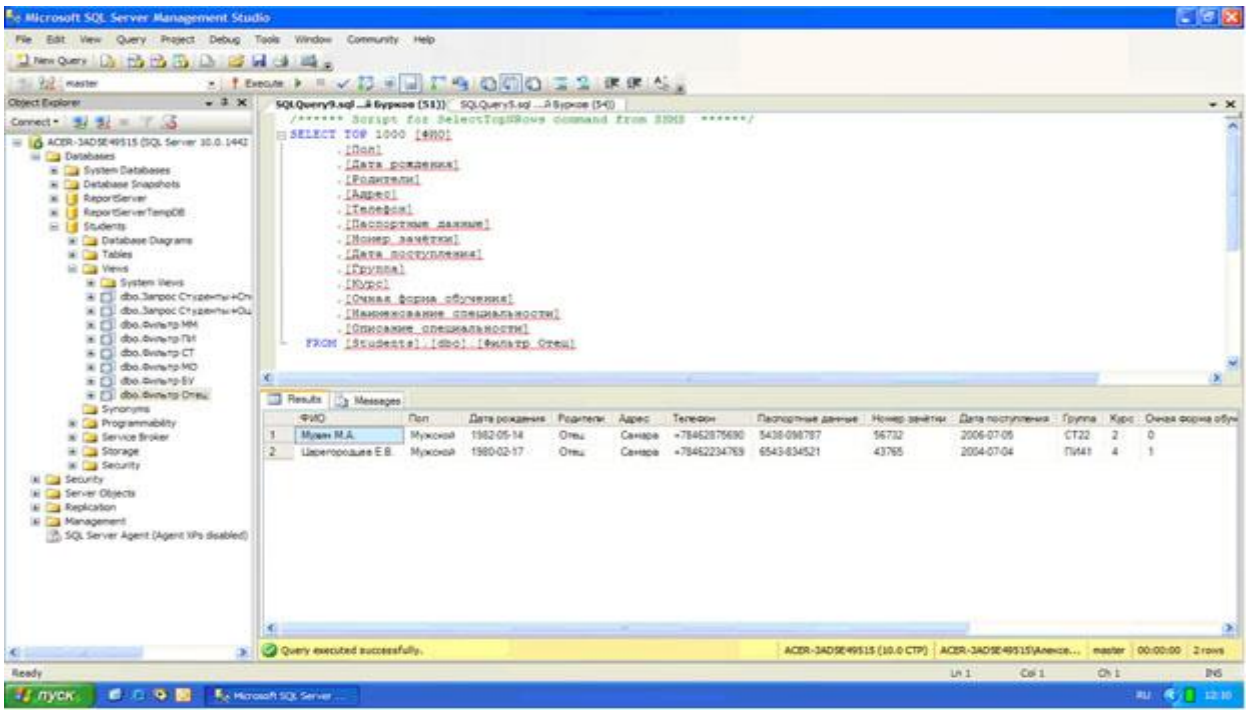


Рис. 8.16.

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

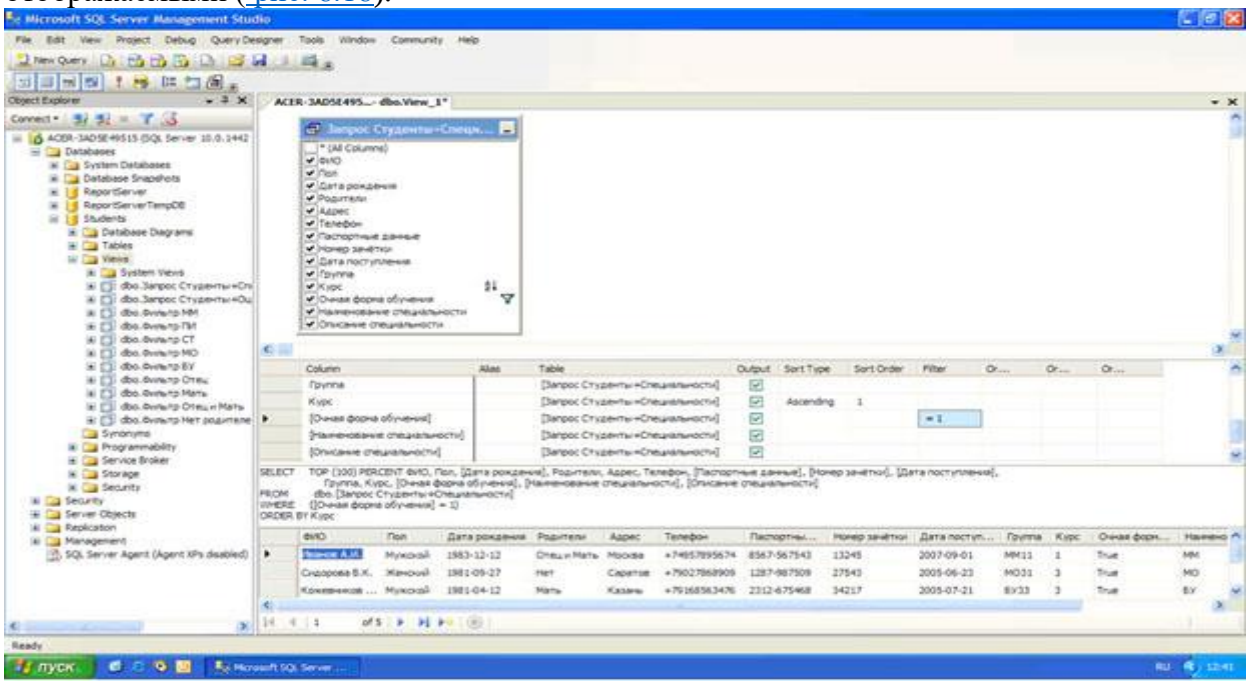


увеличить изображение

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).

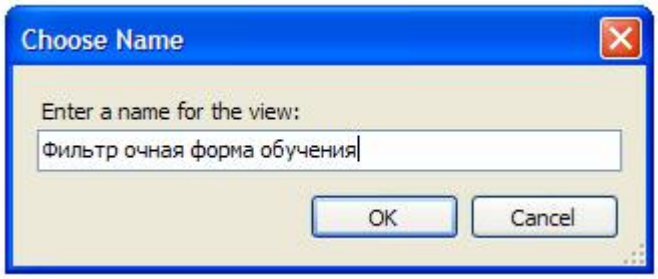
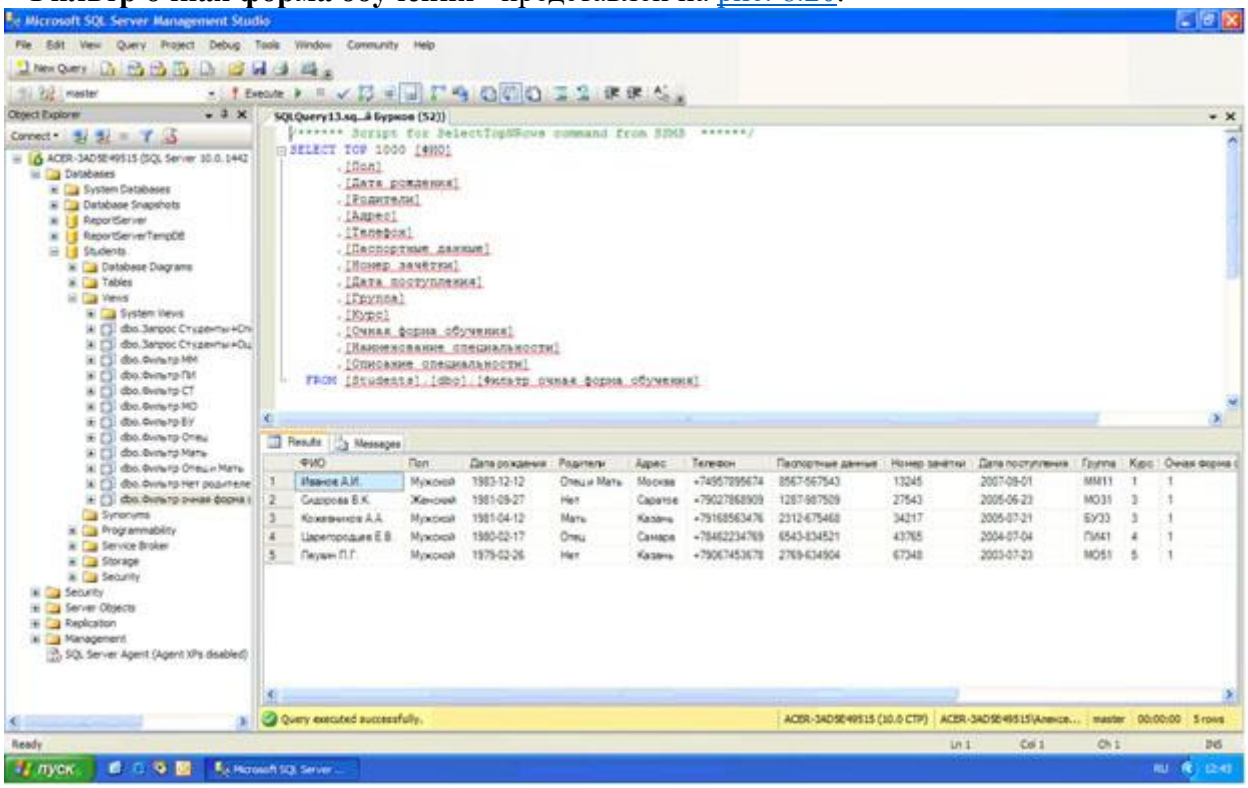


Рис. 8.19.
После появления фильтра **"Фильтр очная форма обучения"** в обозревателе объектов выполните фильтр вне окна конструктора запросов. Результат выполнения фильтра **"Фильтр очная форма обучения"** представлен на рис. 8.20.



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):

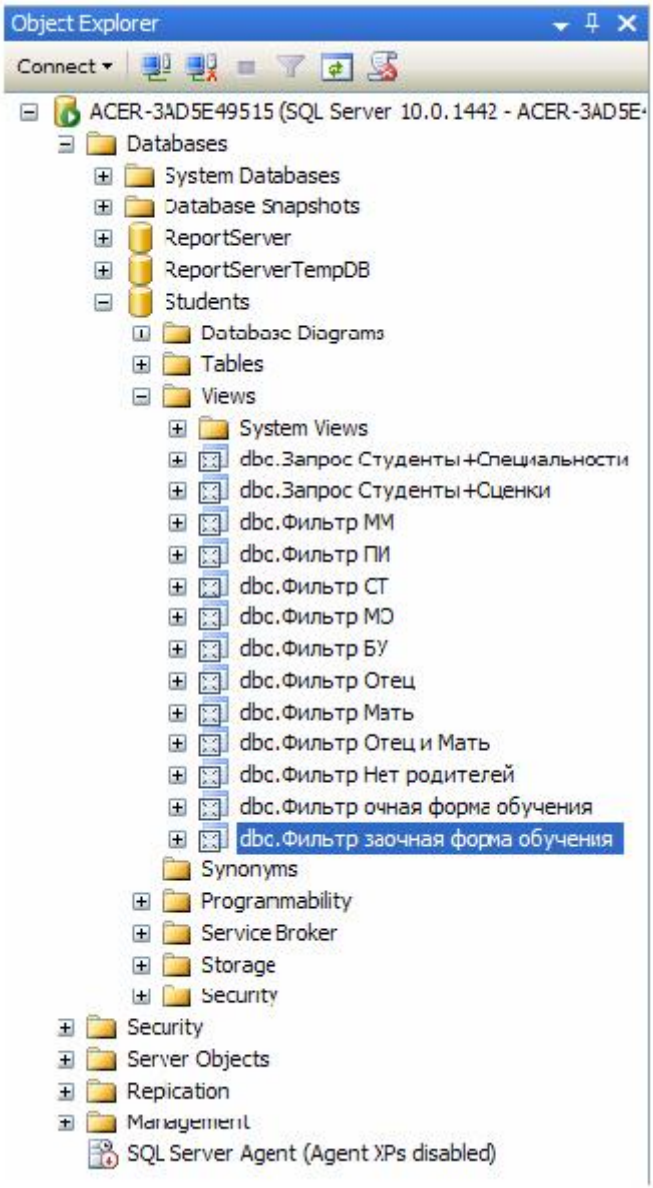


Рис. 8.21.

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).

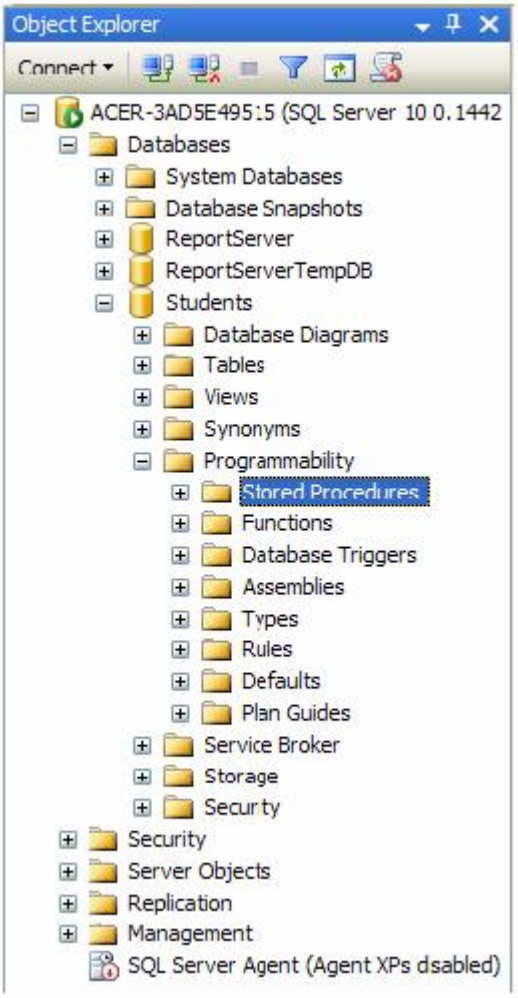
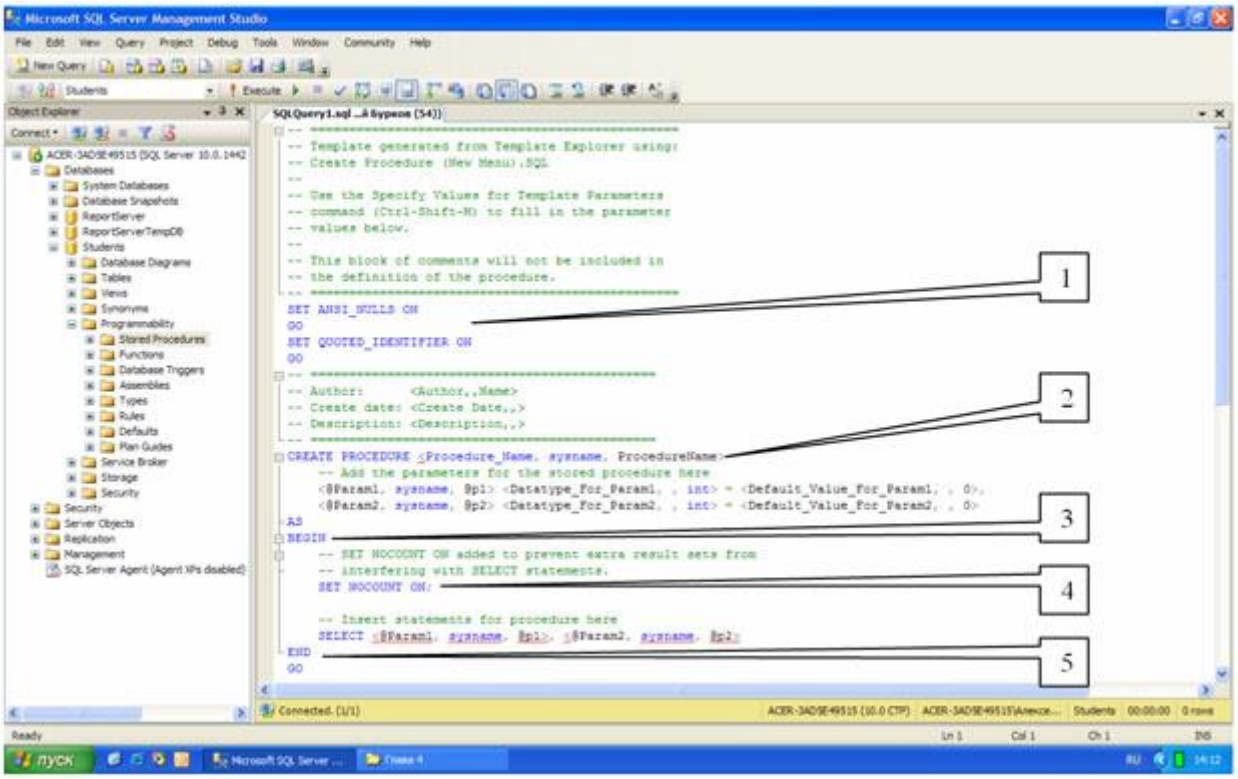


Рис. 10.1.

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.

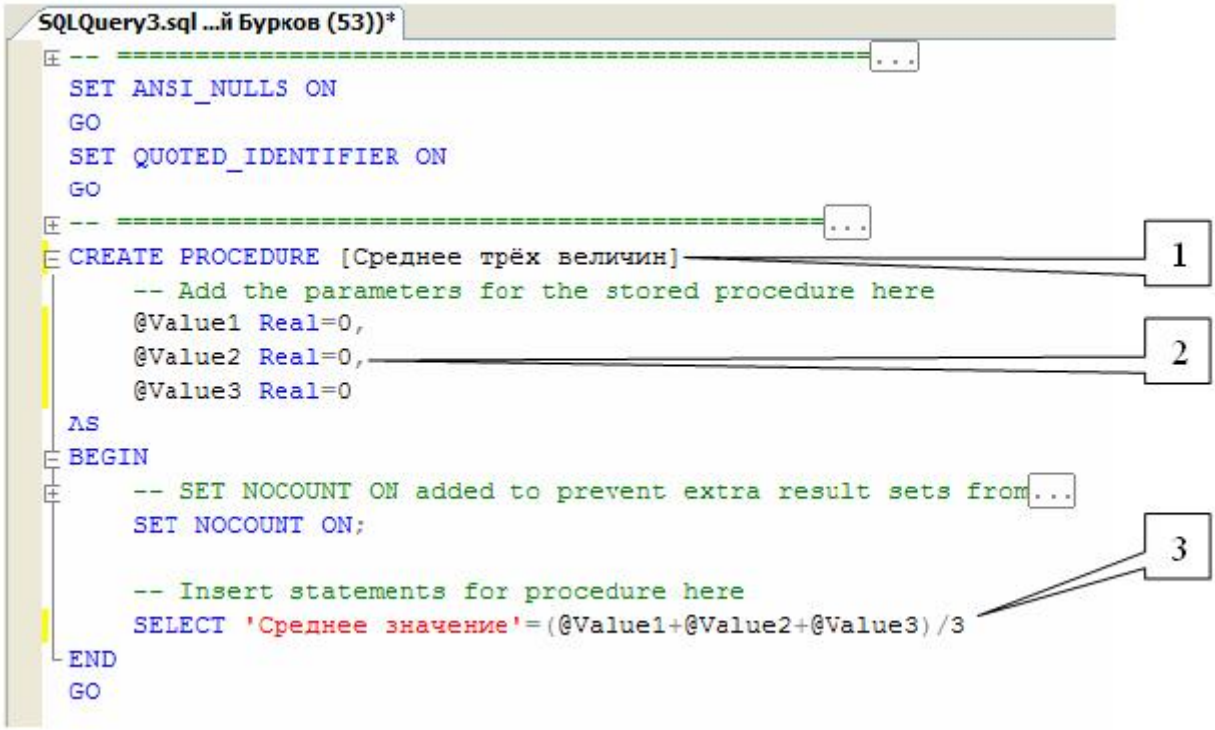


Рис. 10.3.

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).

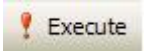
Для создания процедуры, выполним вышеописанный код, нажав кнопку

(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

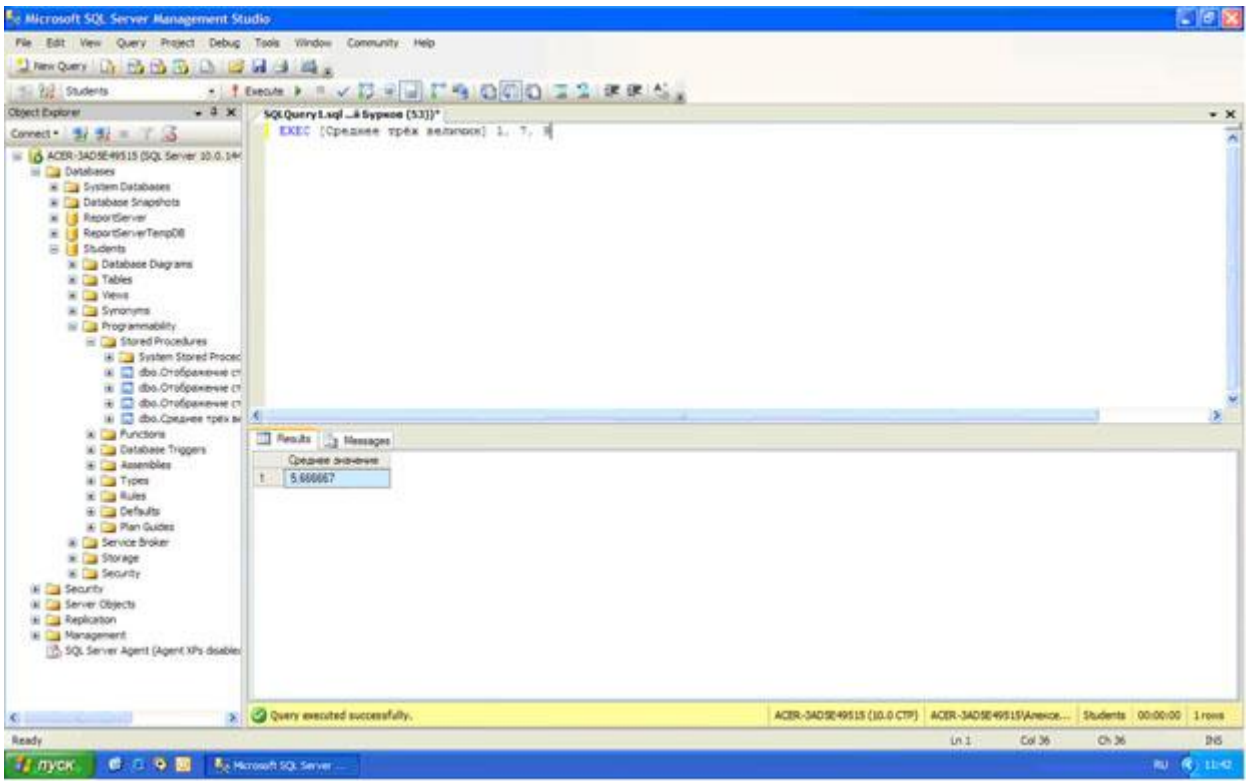
расположенной в верхнем правом углу окна с кодом процедуры.

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).

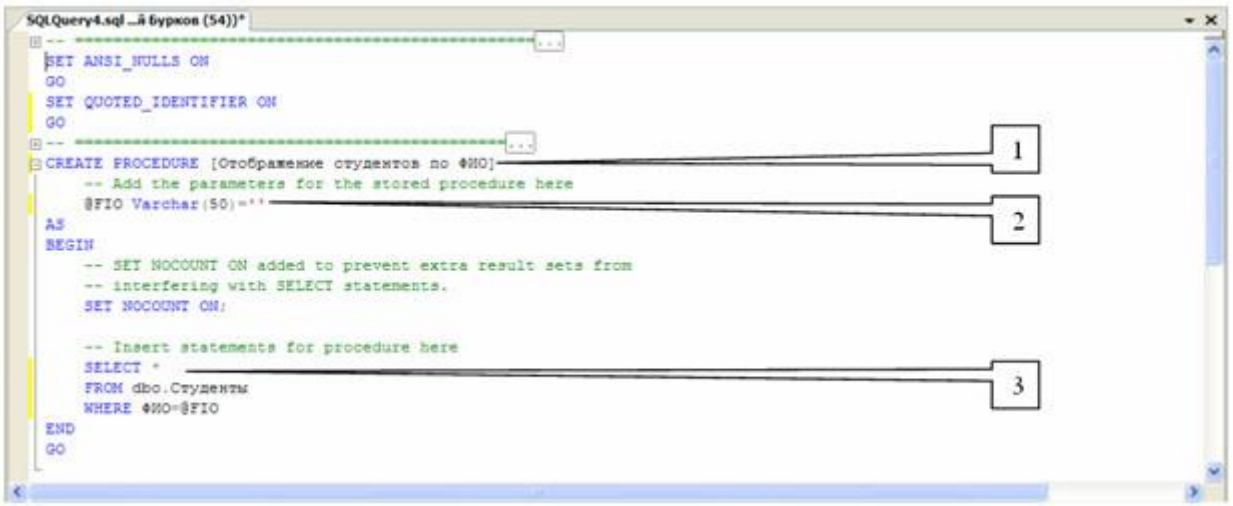


увеличить изображение

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

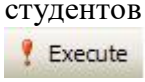
Рис. 10.5.

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

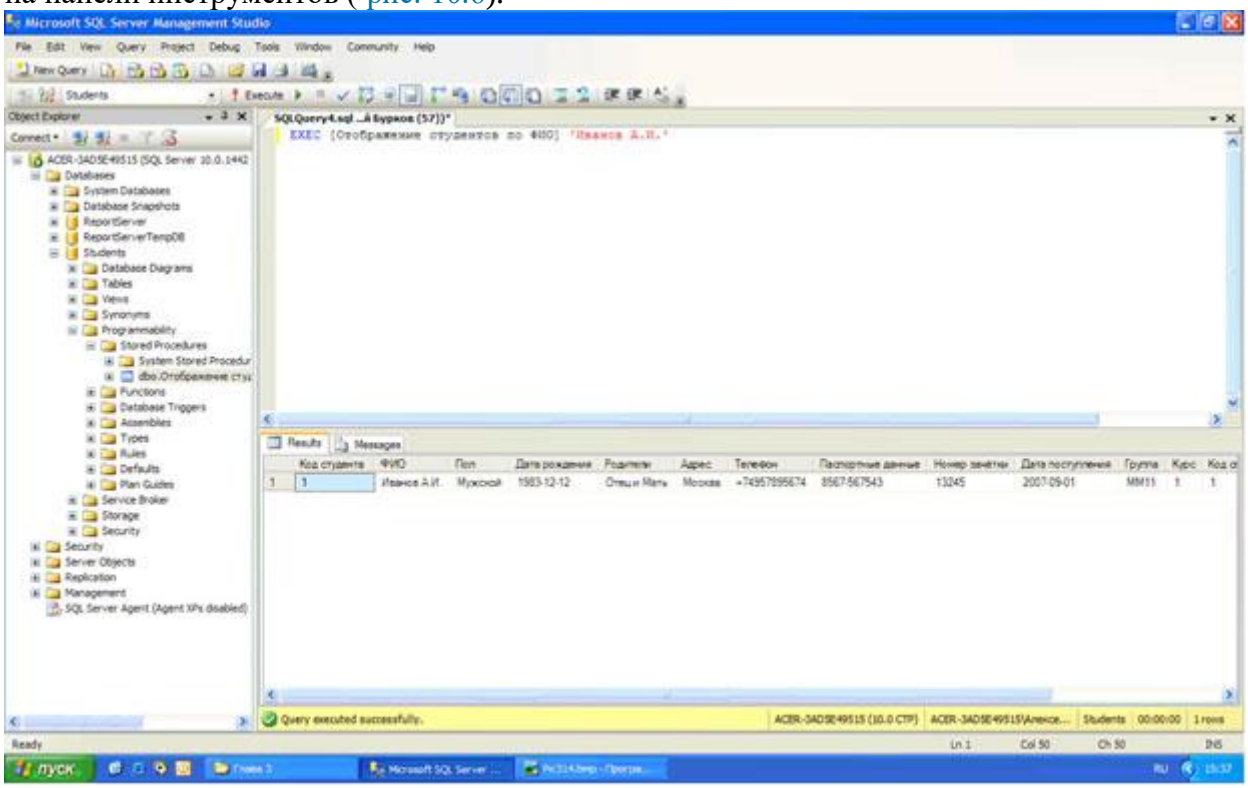
1. CREATE PROCEDURE [Отображение студентов по ФИО] - определяет имя создаваемой процедуры как "Отображение студентов по ФИО";
2. @FIO Varchar(50)='' - определяют единственный параметр процедуры **FIO**. Параметру можно присвоить текстовые строки переменной длины, длинной до 50 символов (Тип данных Varchar(50)),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

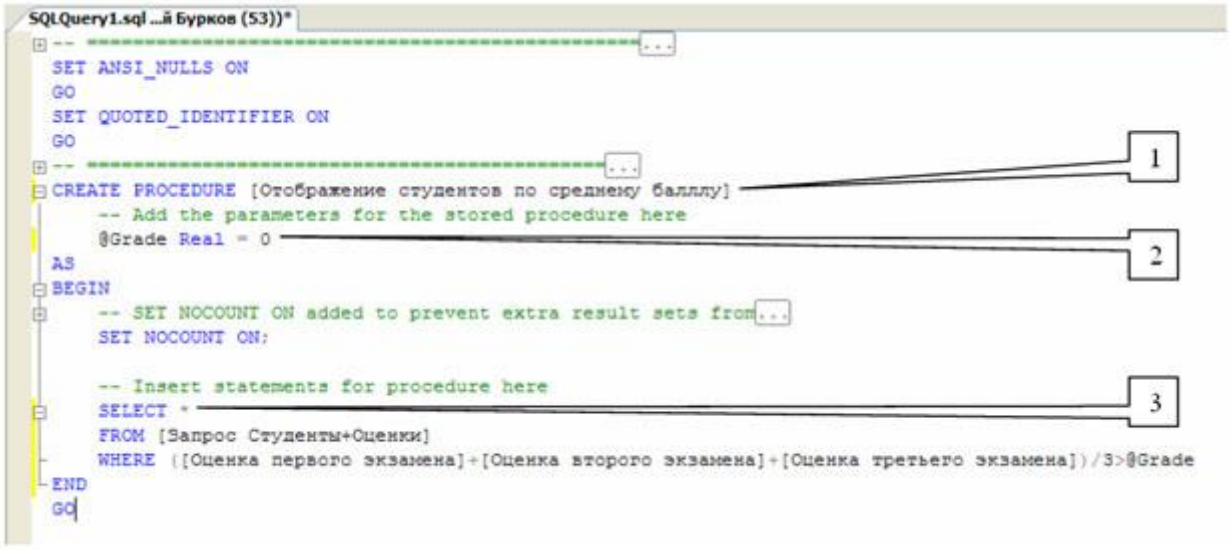


увеличить изображение

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.



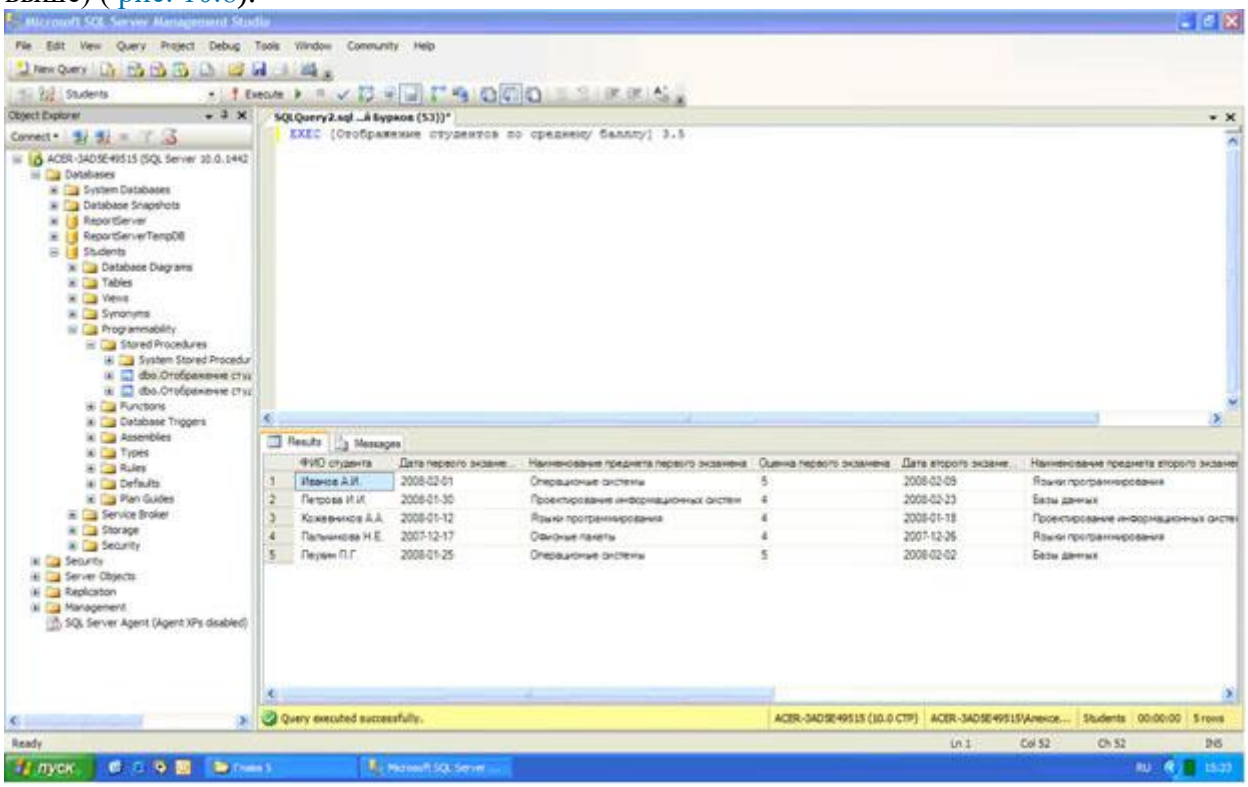
увеличить изображение

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).



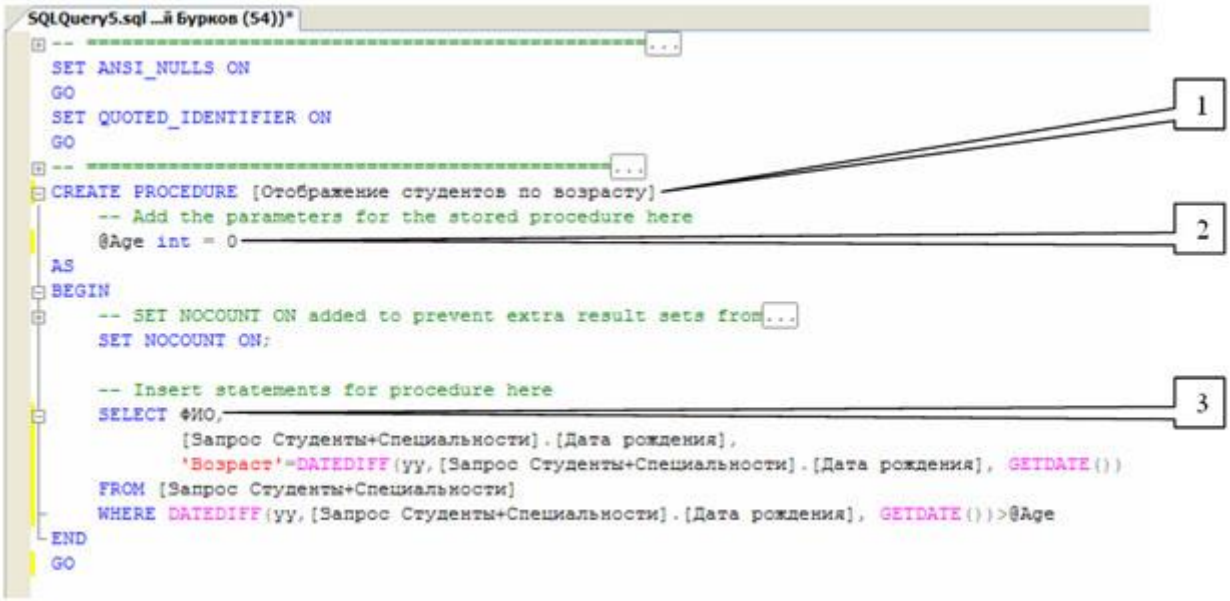
увеличить изображение

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

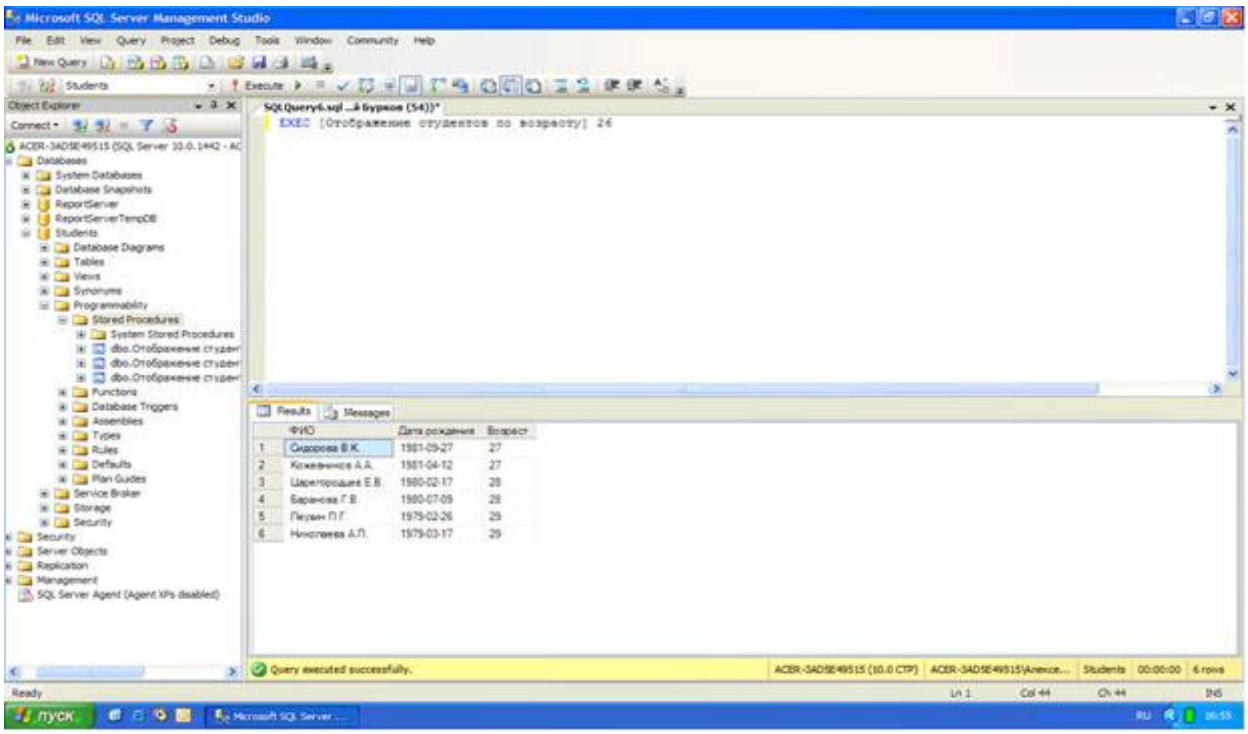
Рис. 10.9.

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.

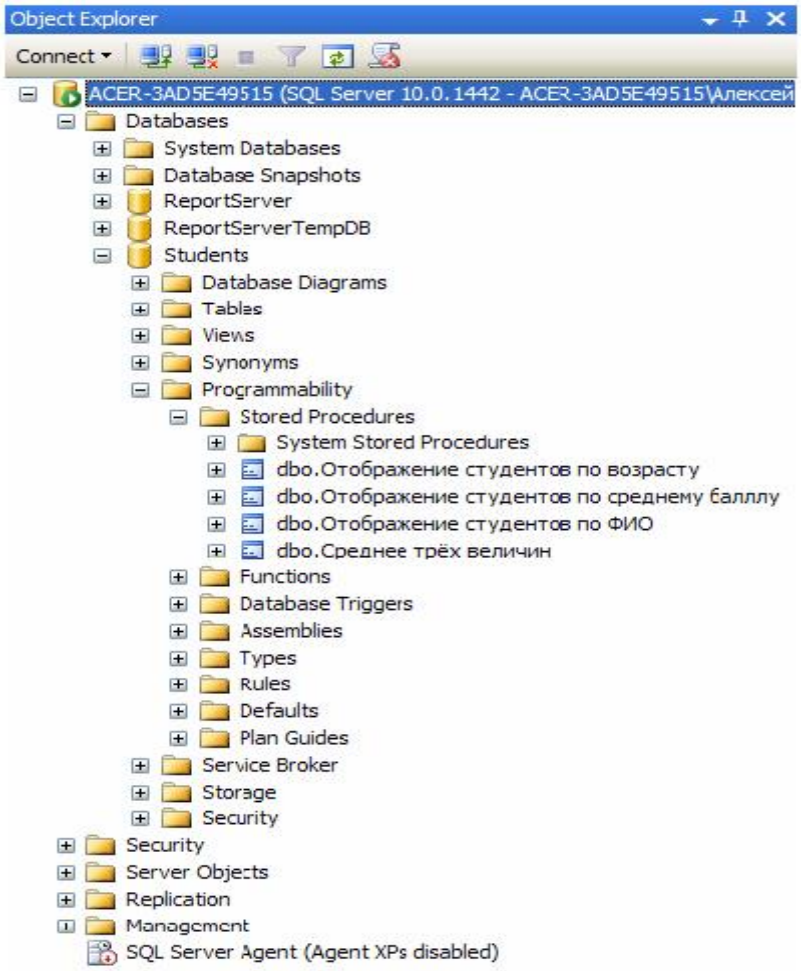


Рис. 10.11.

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).

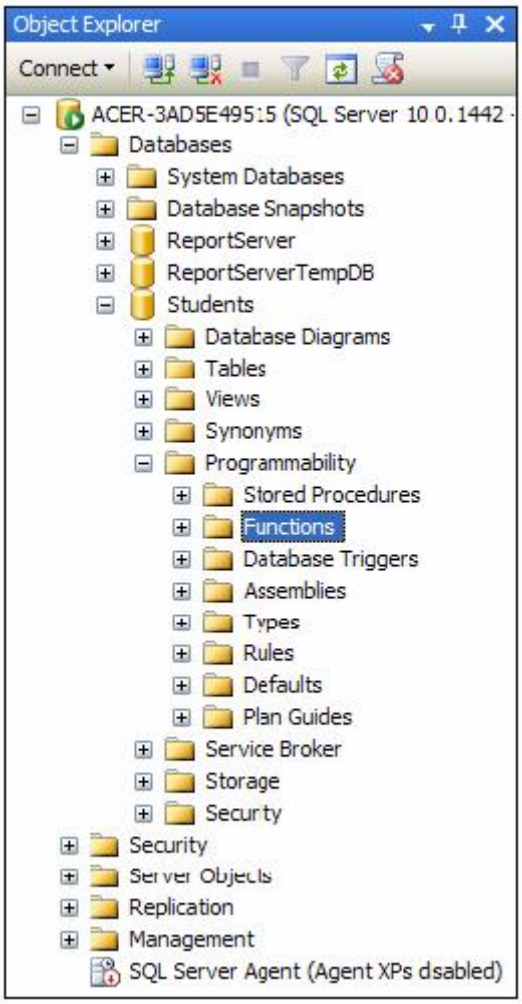
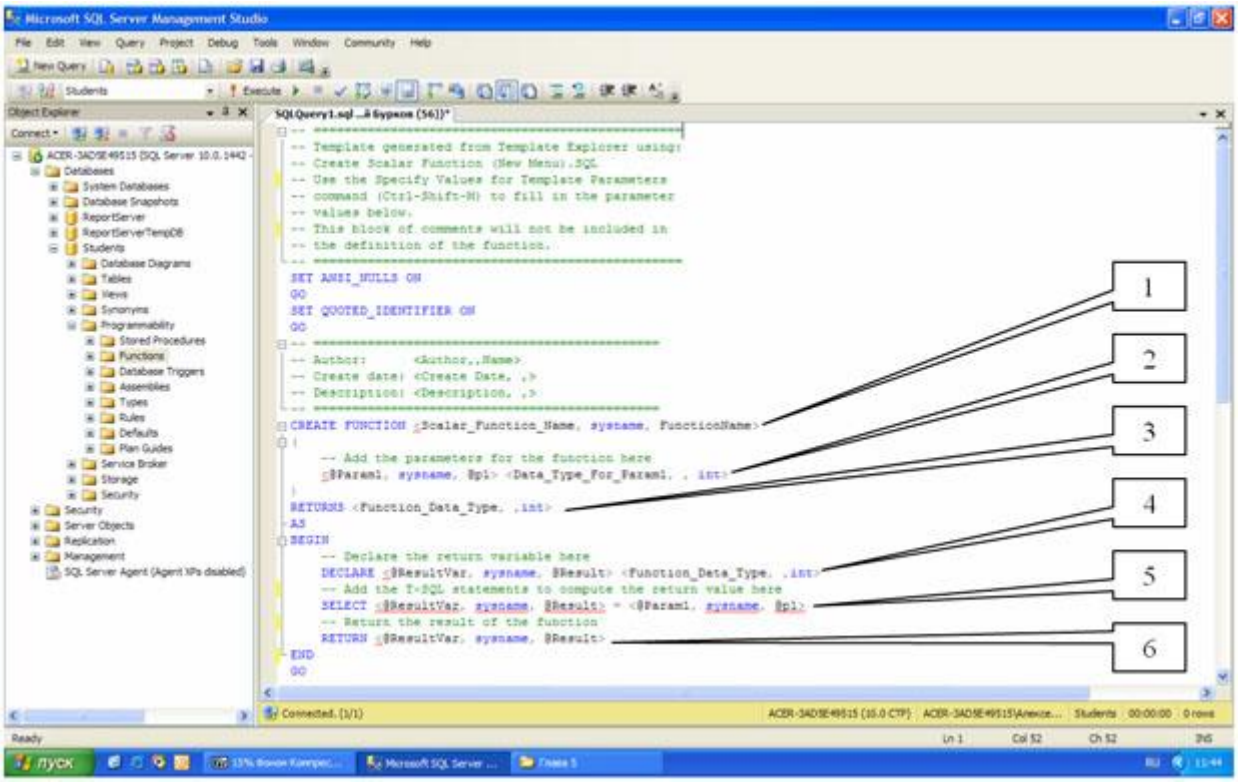


Рис. 12.1.

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)



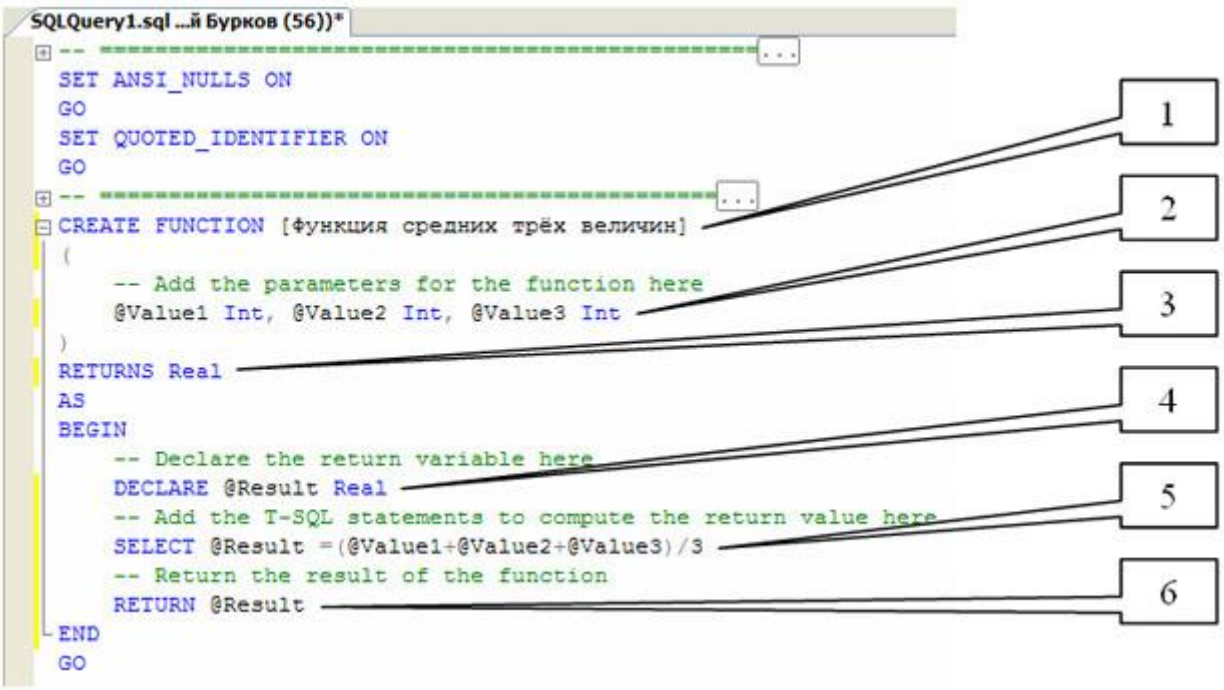
увеличить изображение

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.

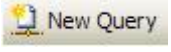
Остальные фрагменты кода рассмотрены выше (рис. 12.2).

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

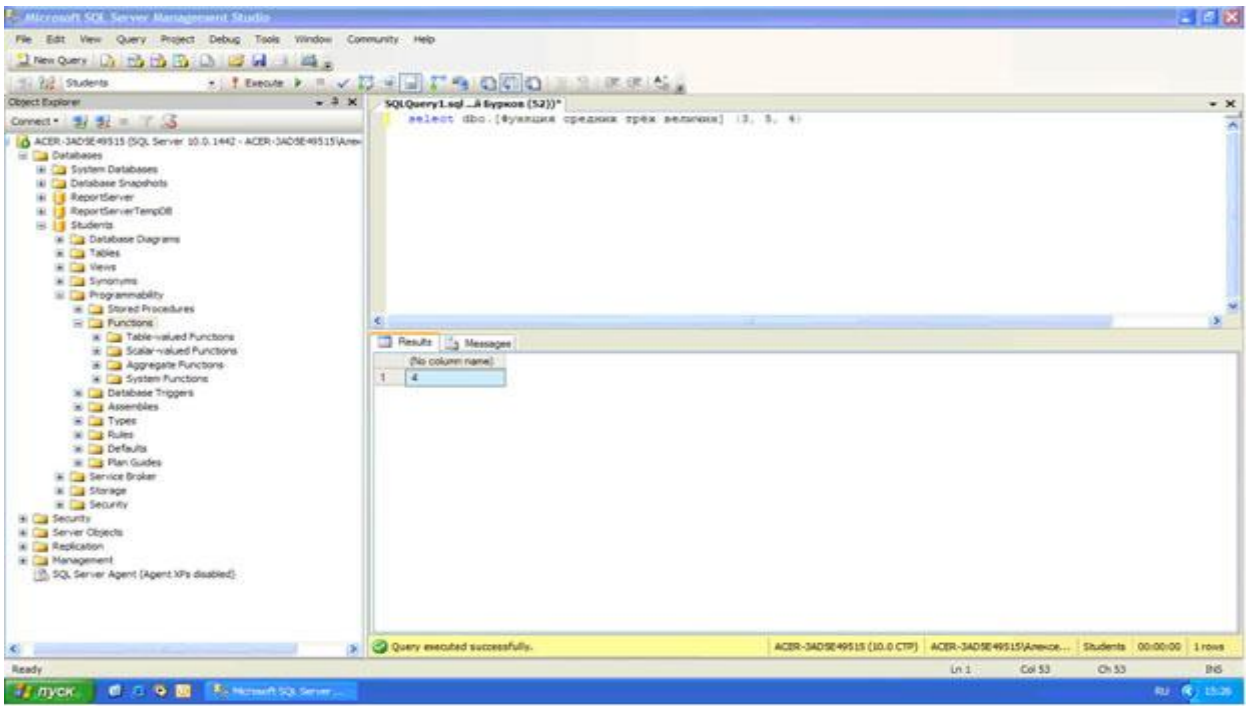
расположенной в верхнем правом углу окна с кодом функции.

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).



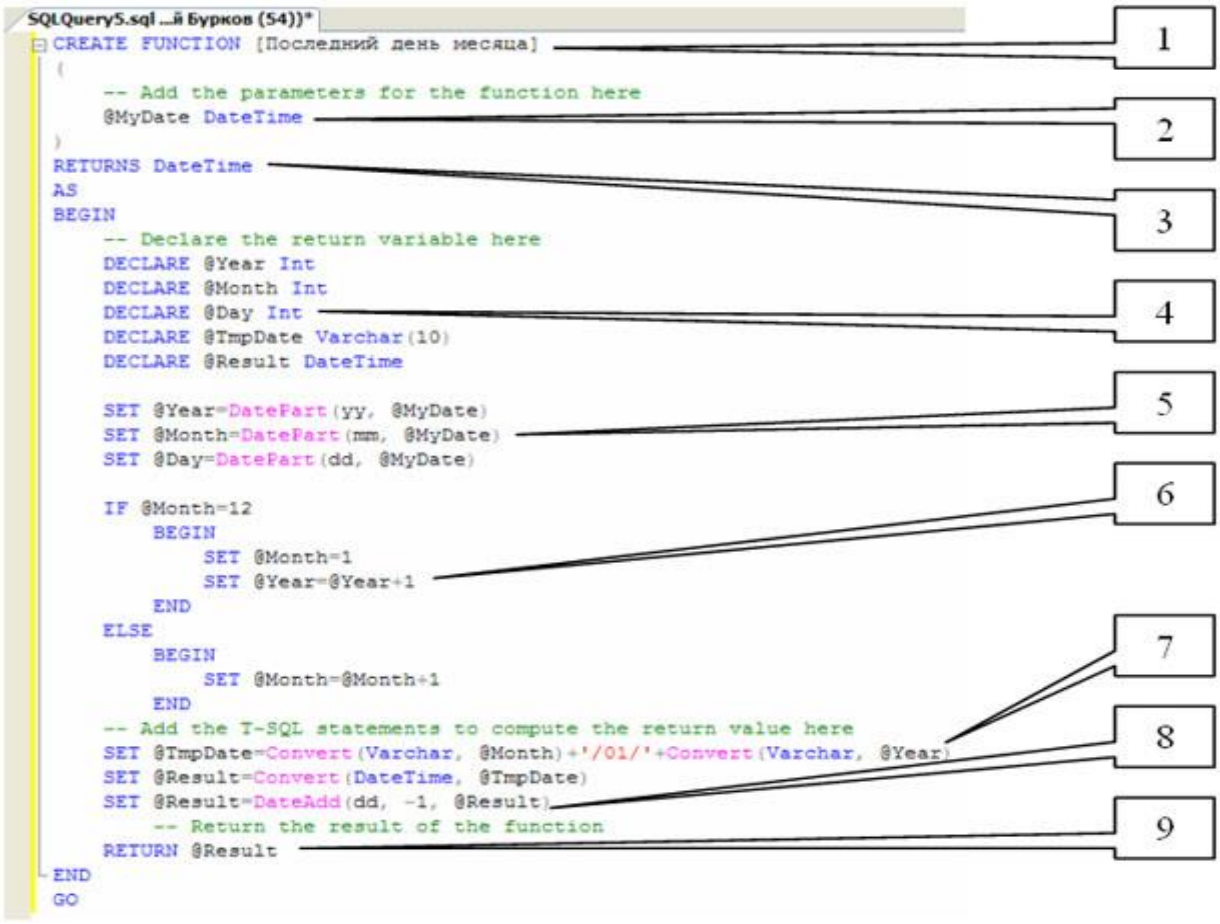
увеличить изображение

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.

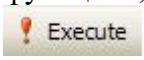
Перейдем к рассмотрению вышеприведенного кода (рис. 12.5). Код состоит из следующих групп команд:

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"**).

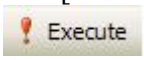
17. RETURN @Result - возвращает значение, хранимое в переменной @Result.

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

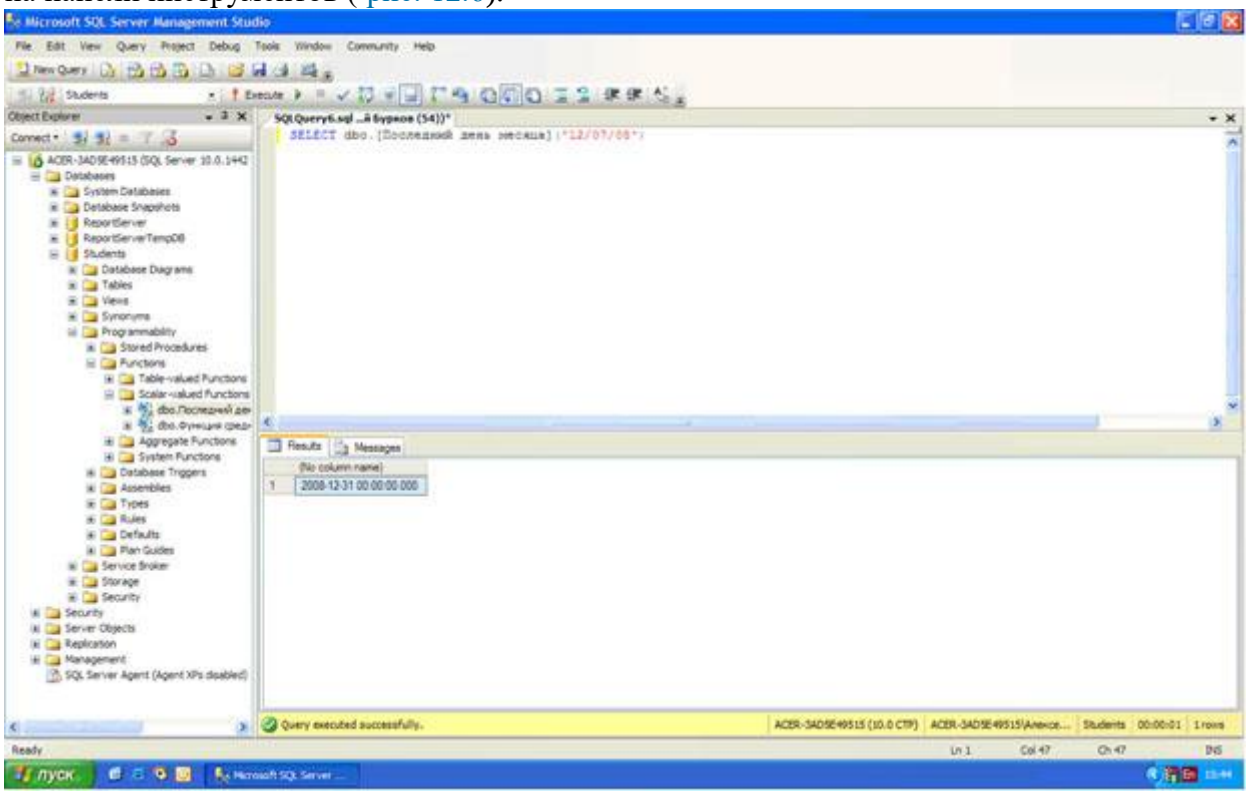


После появления сообщения **"Command(s) completed successfully."** закройте окно с кодом.

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).

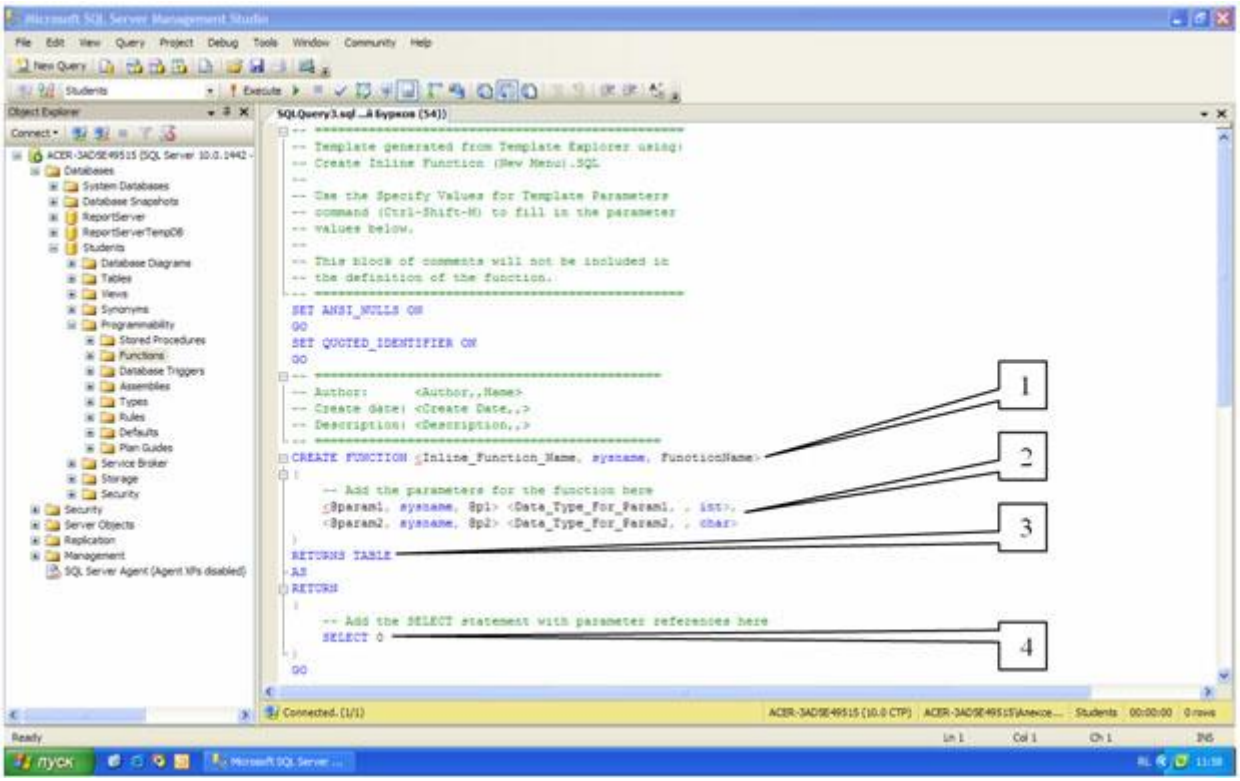


увеличить изображение

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

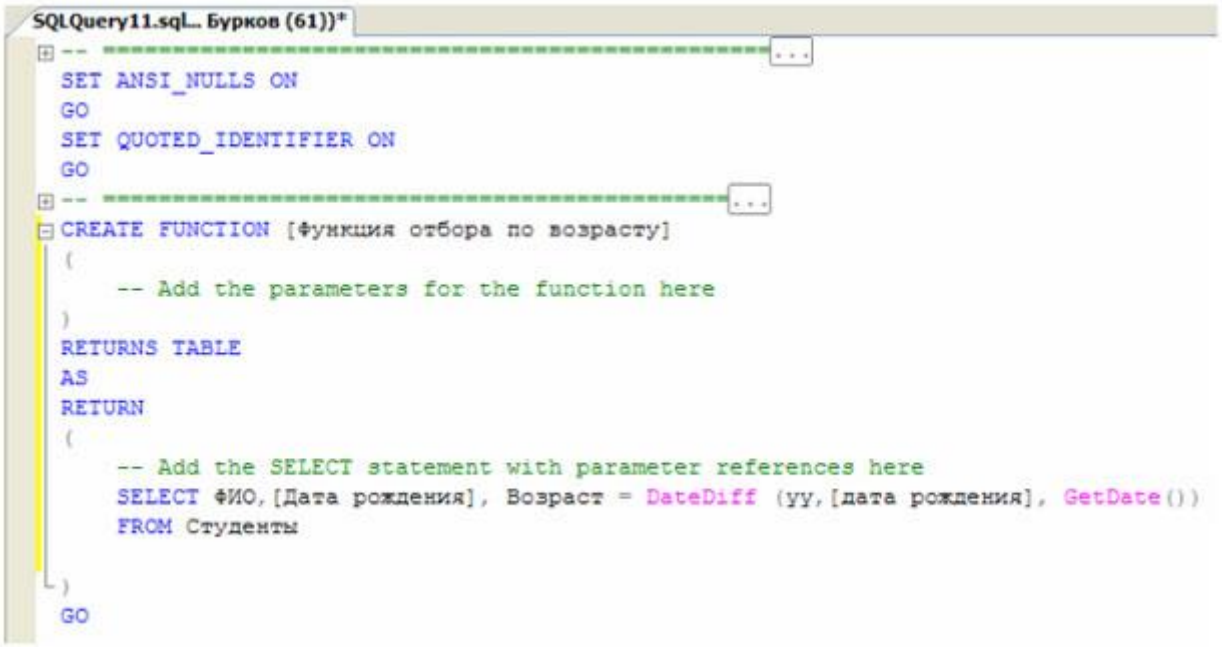
Рис. 12.7.

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

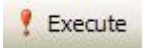
Из вышепредставленной команды видно, что из таблицы **"Студенты"** отображаются поля **"ФИО"** и **"Дата рождения"**, а также вычислимое поле **"Возраст"**. Поле **"Возраст"** вычисляется при помощи встроенной функции **DateDiff** вычисляющей различие между датами в определенных единицах измерения (частях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

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

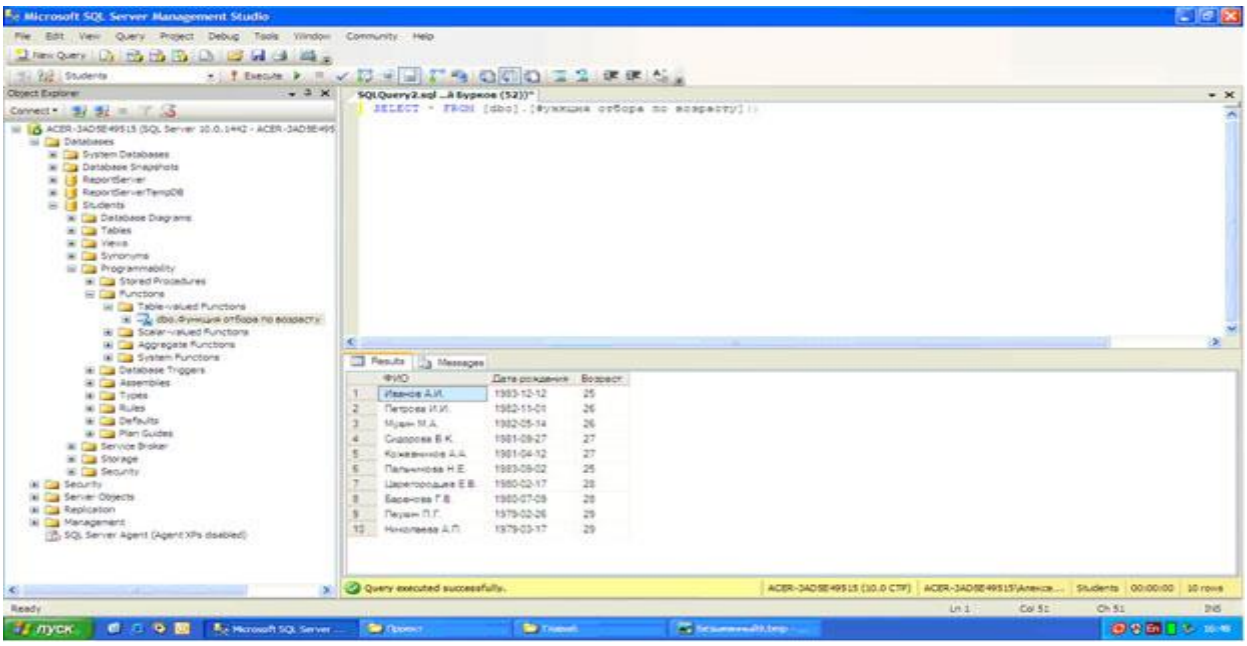
Здесь "часть даты" - это закодированные специальными символами единицы измерения (часть даты) (yy - год, mm - месяц, dd - день), "начальная дата" - дата начала периода и "конечная дата" - дата конца периода. В нашем случае в качестве начальной даты берем дату рождения студента, а в качестве конечной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

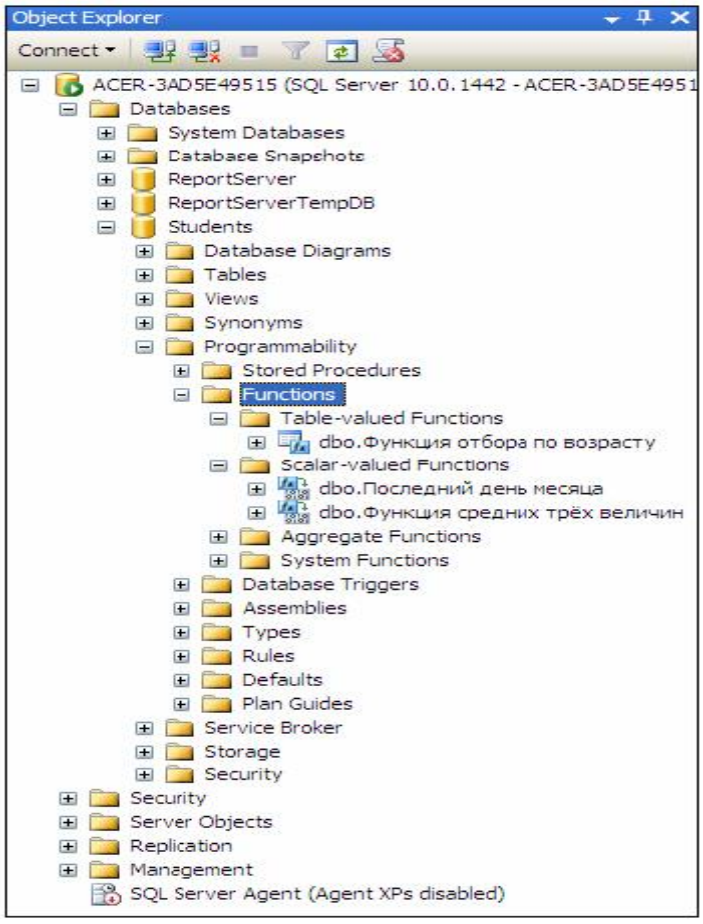


Рис. 12.10.

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

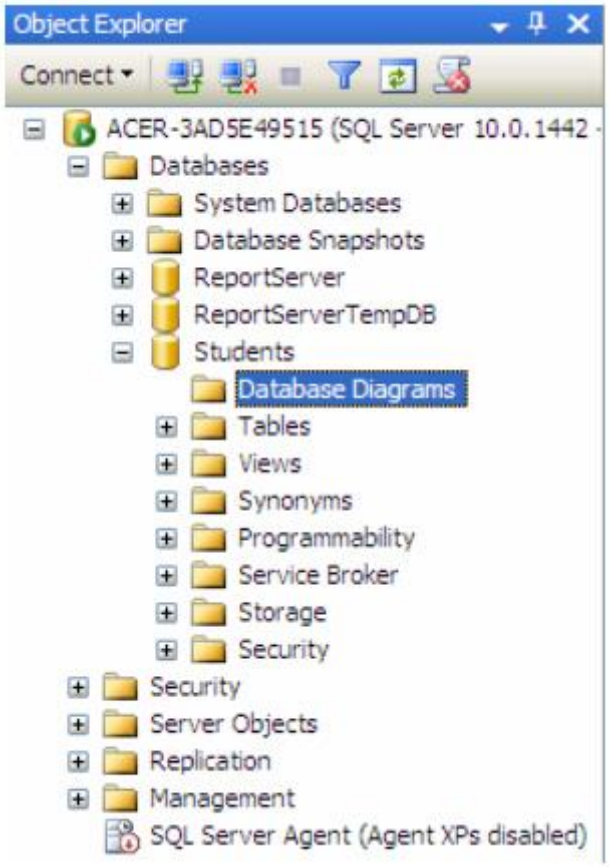


Рис. 14.1.

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).

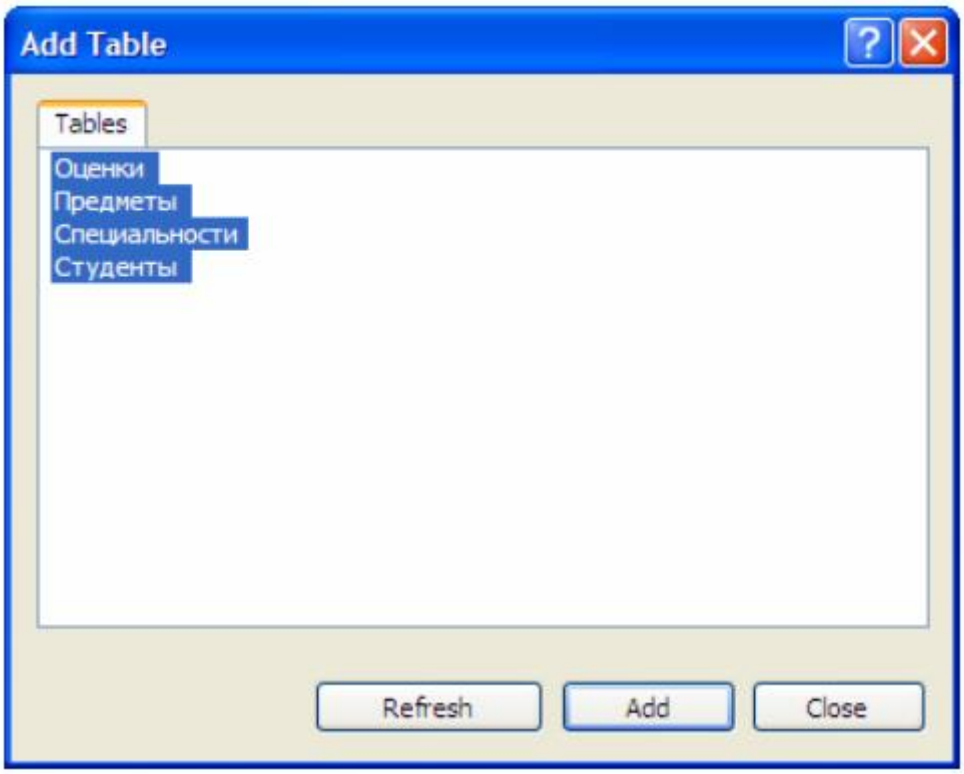


Рис. 14.2.

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).

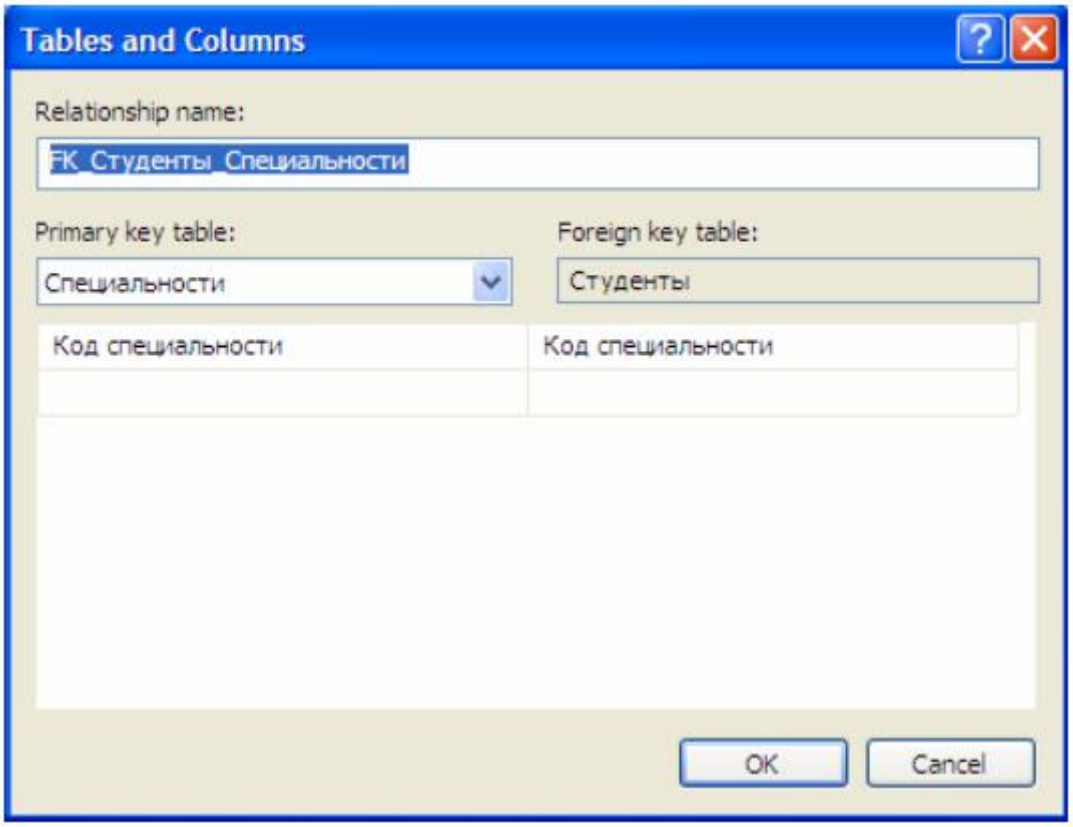
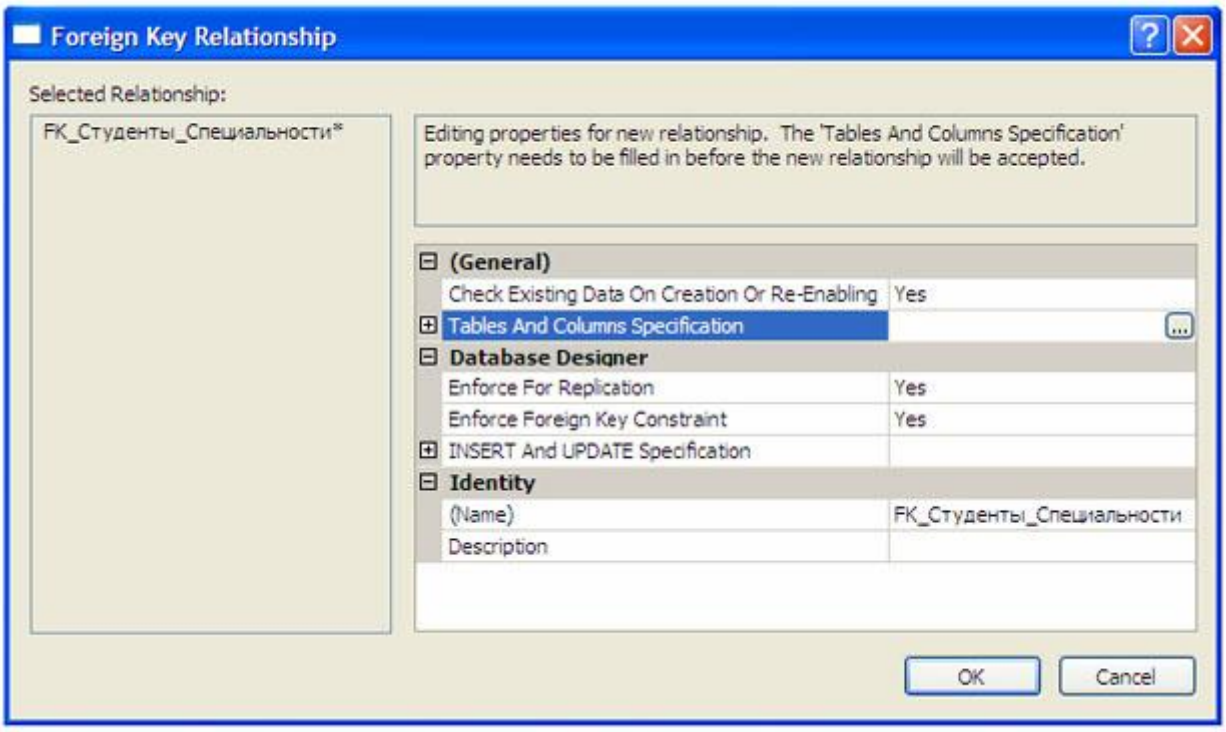


Рис. 14.3.

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

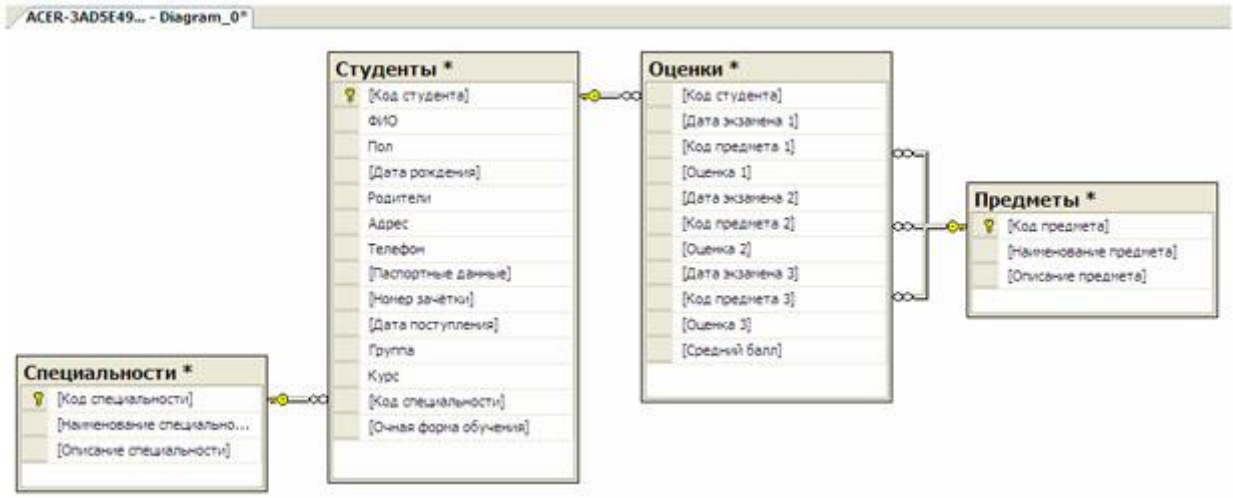


увеличить изображение

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).

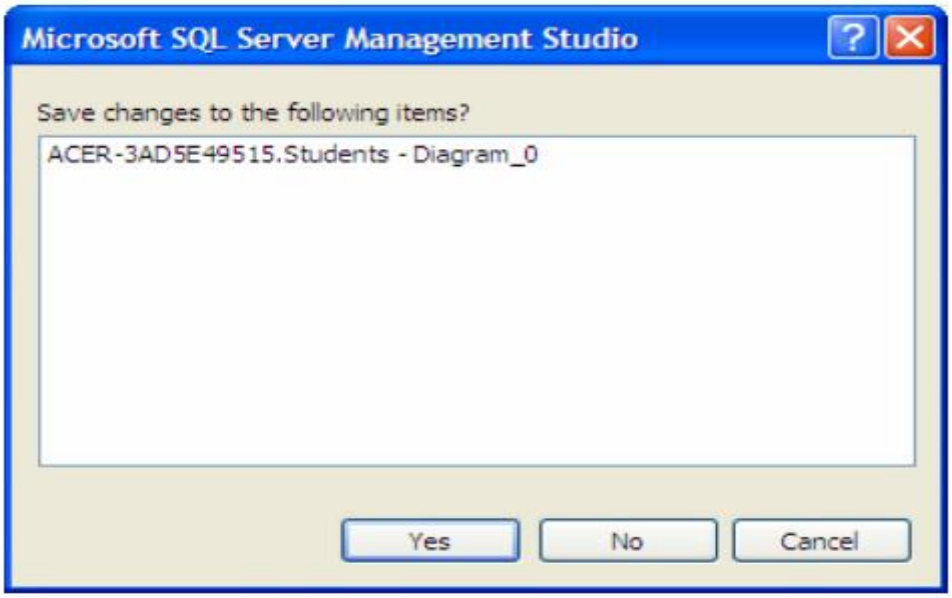


Рис. 14.6.

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).

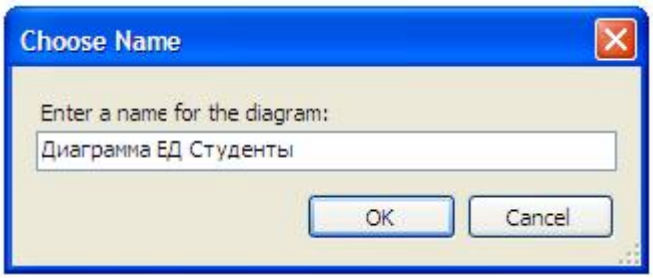


Рис. 14.7.

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

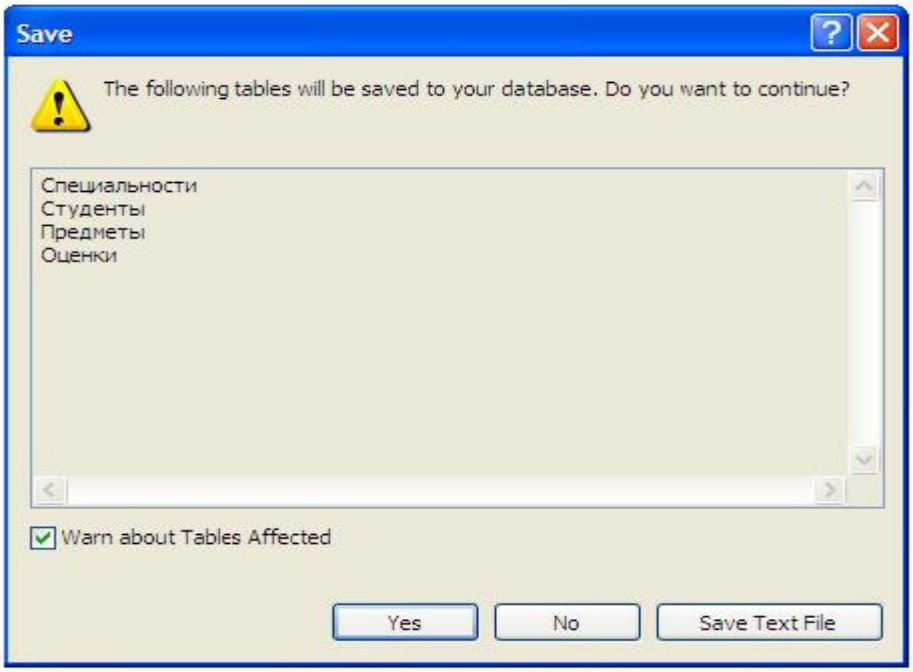


Рис. 14.8.

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).

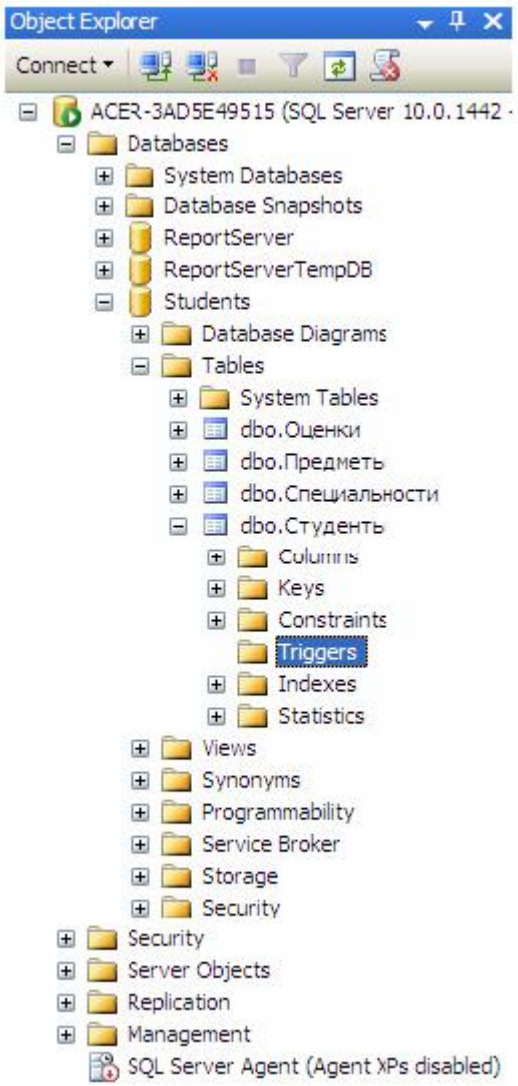
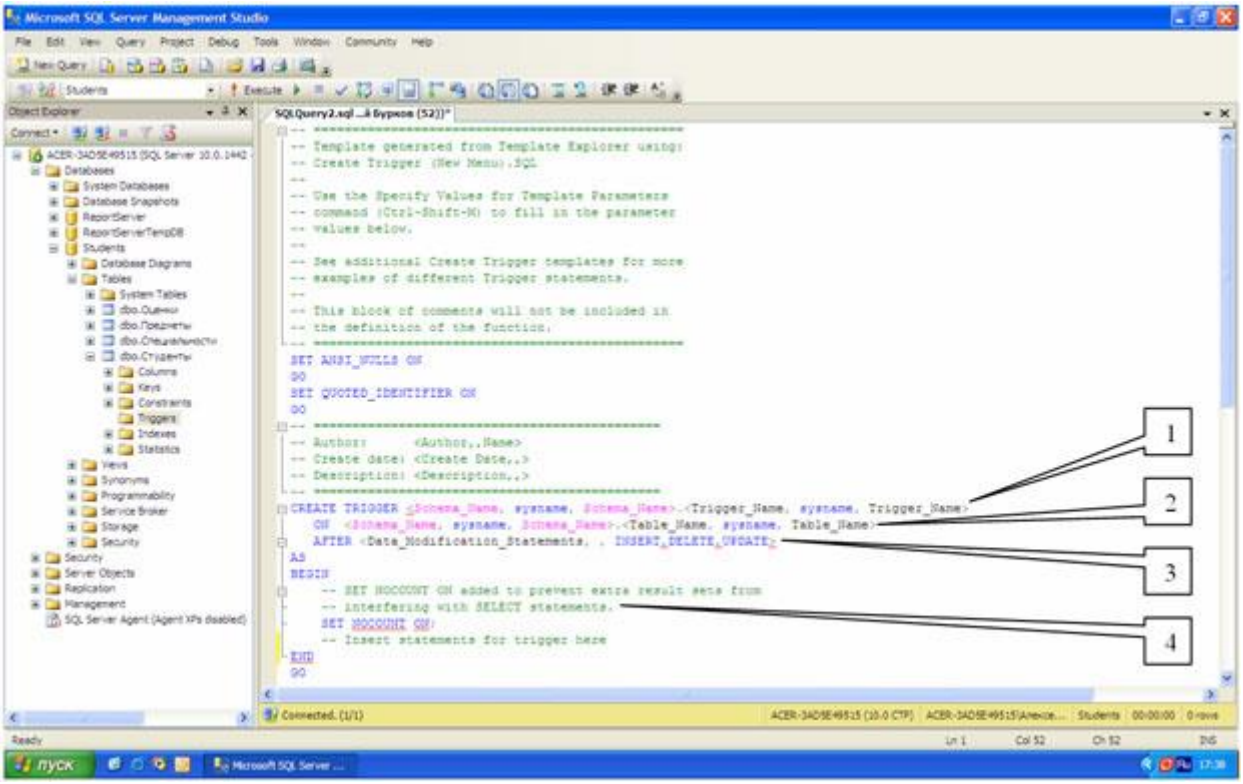


Рис. 14.9.

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.

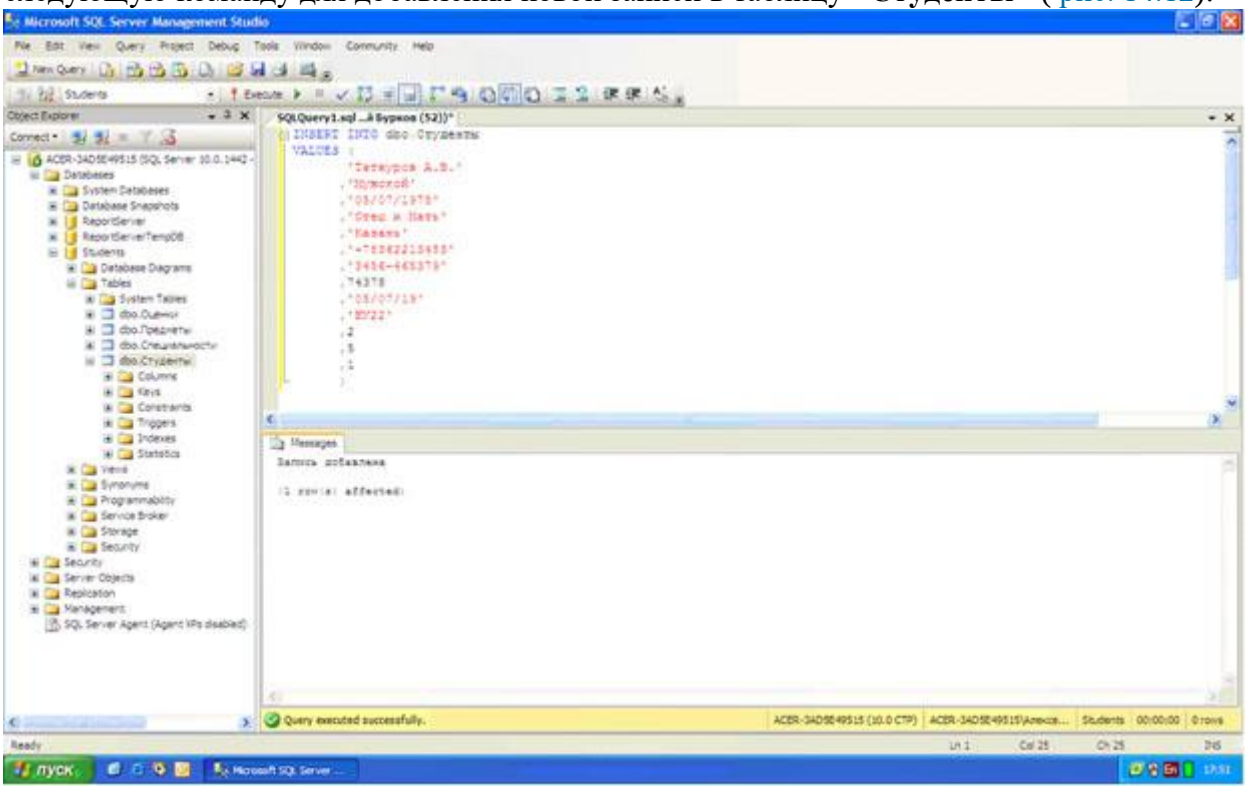
```
SQLQuery6.sql ...й Бурков (52))*
-- ============================================...
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- ============================================...
CREATE TRIGGER [Индикатор добавления]
   ON  dbo.Студенты
   AFTER INSERT
AS
BEGIN
    -- SET NOCOUNT ON added to 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):

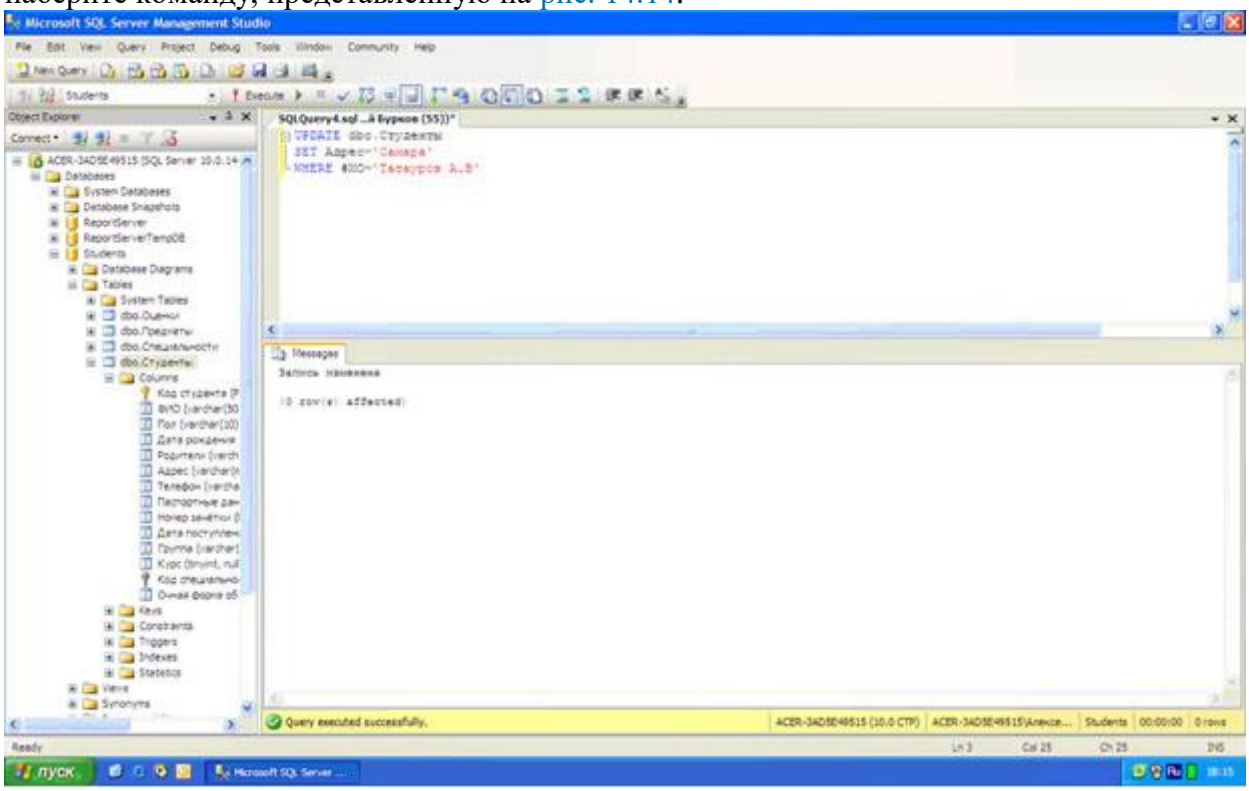
```
SQLQuery8.sql ...й Бурков (51))*
-- ============================================...
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- ============================================...
CREATE TRIGGER [Индикатор изменения]
   ON  dbo.Студенты
   AFTER UPDATE
AS
BEGIN
    -- SET NOCOUNT ON added to 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.

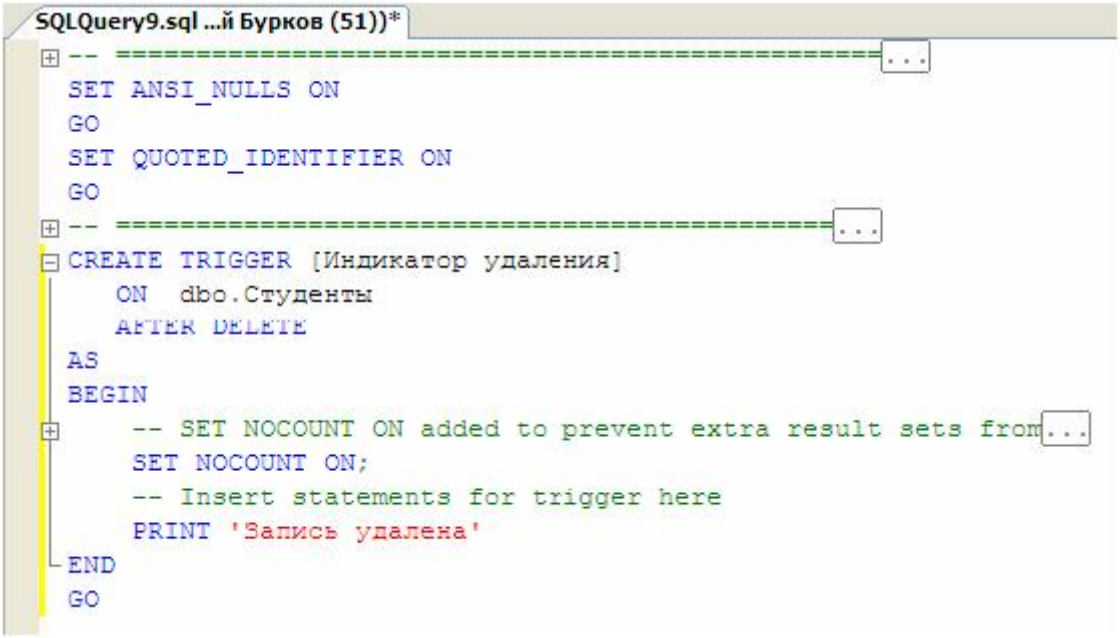
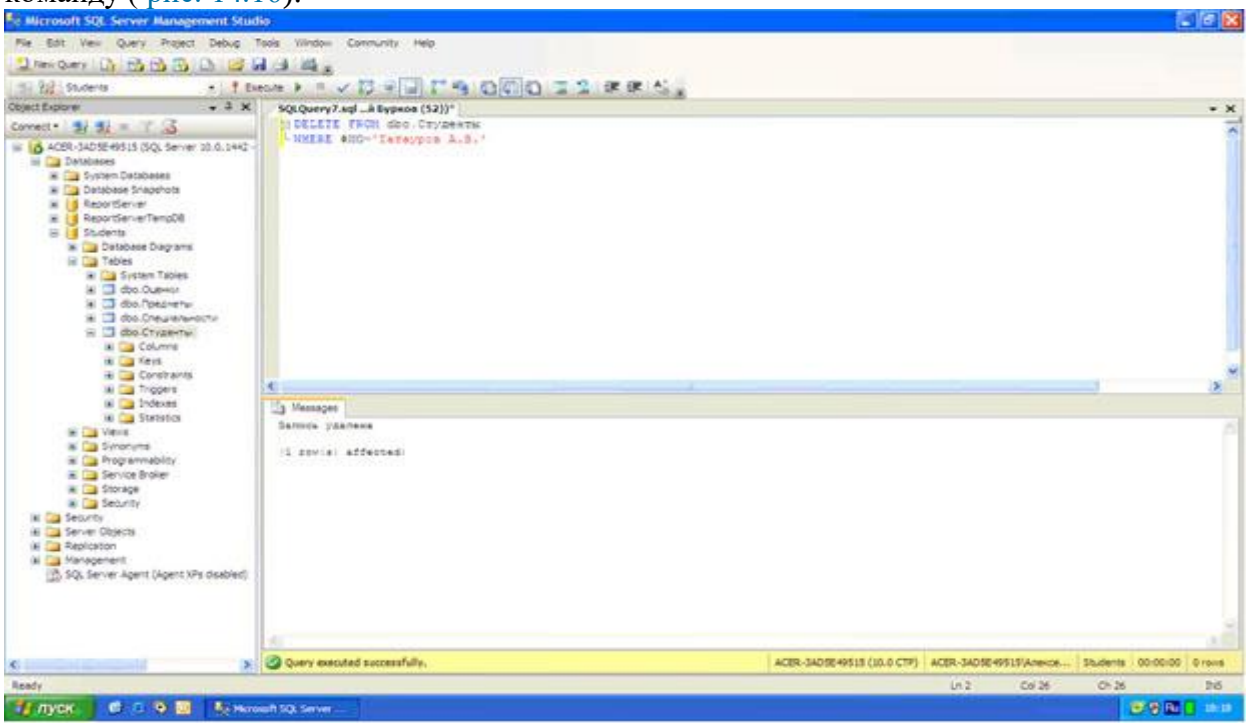


Рис. 14.15.

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):

SQLQuery10.sql... Бурков (56))*

```
-- ============================================...
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
-- ============================================...
CREATE TRIGGER [Удаление Студента]
   ON dbo.Студенты
   INSTEAD OF DELETE
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from...
    SET NOCOUNT ON;
    -- Insert statements for trigger here
    DELETE dbo.Оценки
    FROM Deleted
    WHERE Deleted.[Код студента]=Оценки.[Код студента]
    DELETE dbo.Студенты
    FROM Deleted
    WHERE Deleted.[Код студента]=Студенты.[Код студента]
END
GO
```

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:

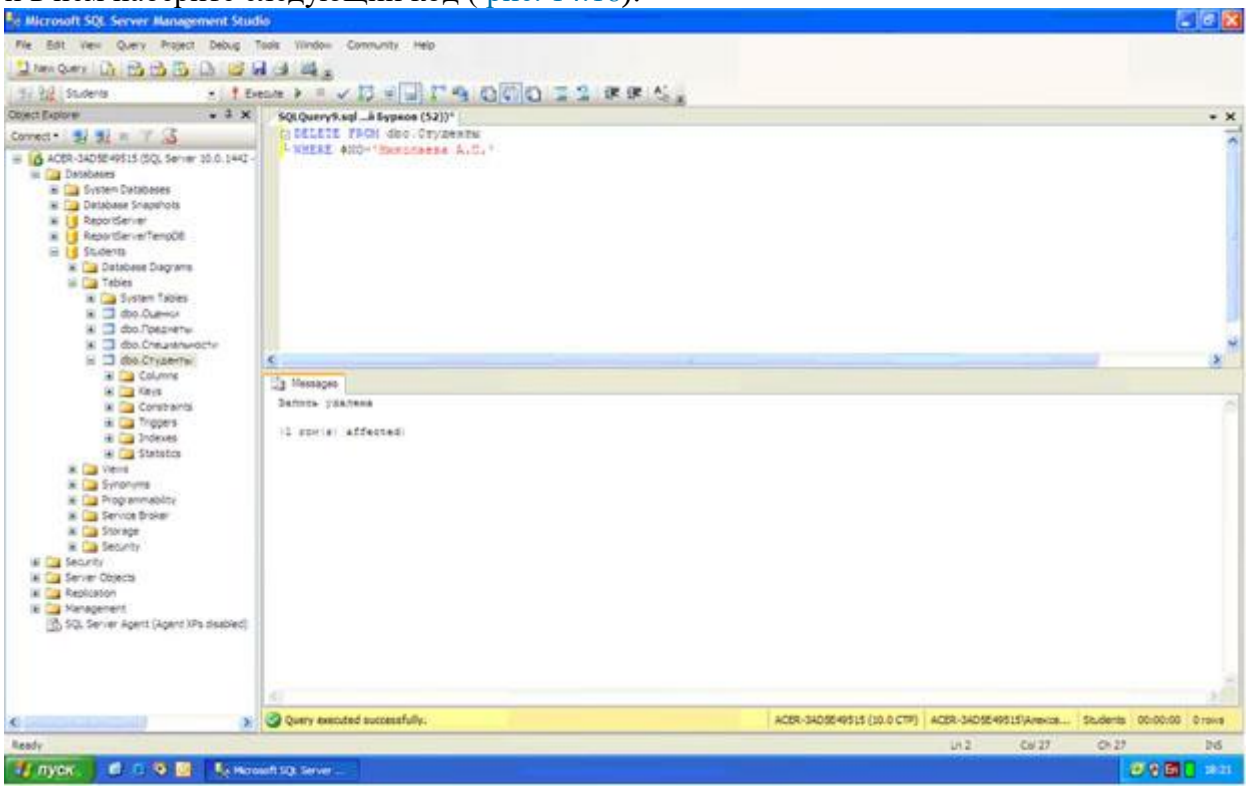
DELETE dbo.Оценки FROM Deleted
WHERE Deleted.[Код студента] = Оценки.[Код студента]

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):

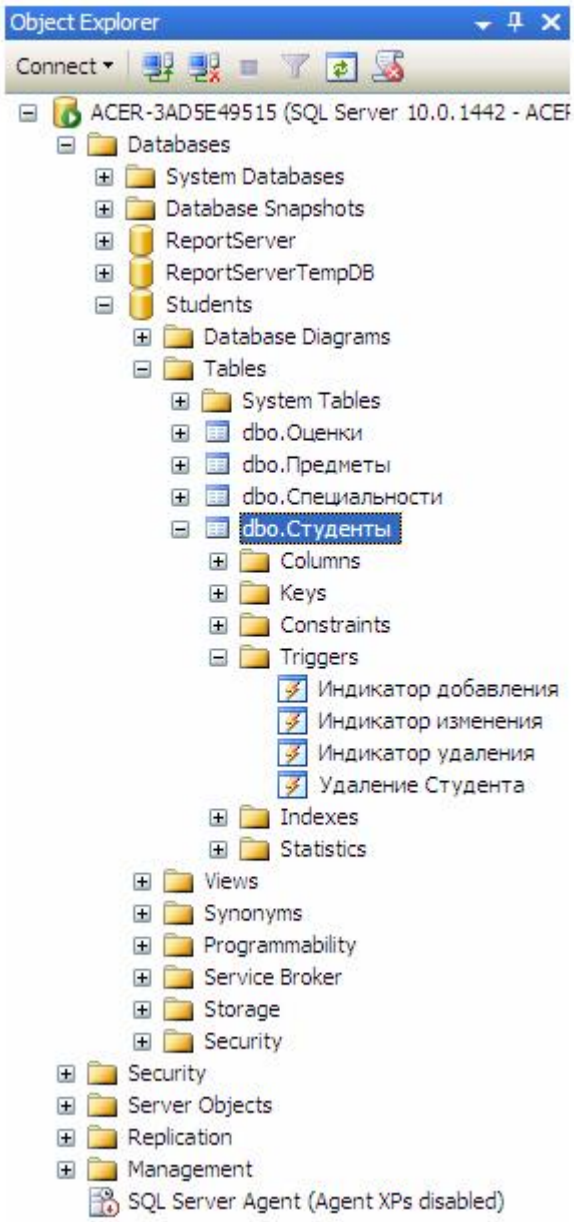


Рис. 14.19.

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. 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?
6. Как создать новый запрос?
7. Как создать новый фильтр?
8. Как использовать фильтр?
9. Что такое хранимая процедура?
10. Какова структура хранимой процедуры?
11. Назначение пользовательских функций.
12. Синтаксис скалярной пользовательской функции.
13. Табличные пользовательские функции.
14. 1. Структура триггеров.
15. 2. Для чего используется диаграмма.
16. 3. Связи между таблицами.

Работа с литературой:

Рекомендуемые источники информации (№ источника)			
Основная	Дополнительная	Методическая	Интернет-ресурсы
1-2	1-2	1-2	1-3

Оценочные средства: устный отчет к лабораторной работе (См.: Фонд оценочных средств)

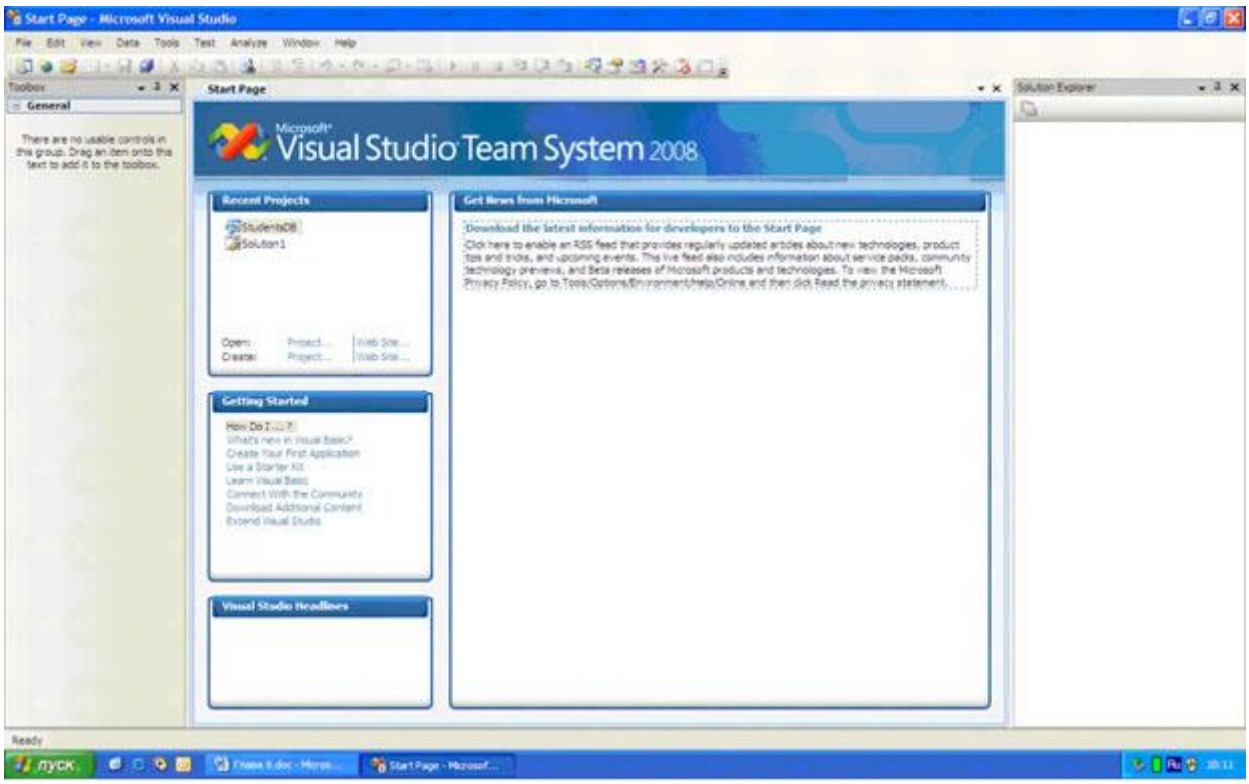
Лабораторная работа № 2. «Выполнение индивидуальных заданий по проектированию информационных систем в Visual Studio 2012. Создание ленточных и табличных форм для работы с базами данных.»

Форма проведения лабораторная работа (3 часа)

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.

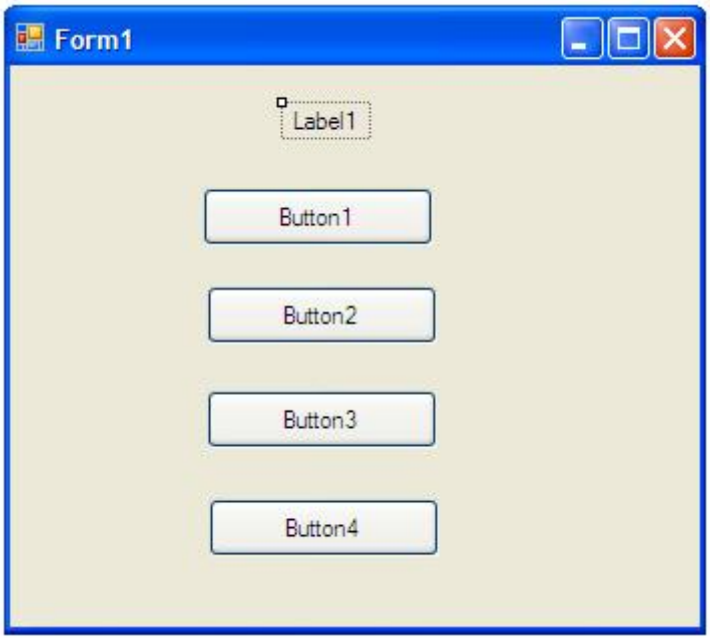
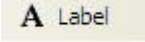


Рис. 18.2.

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.

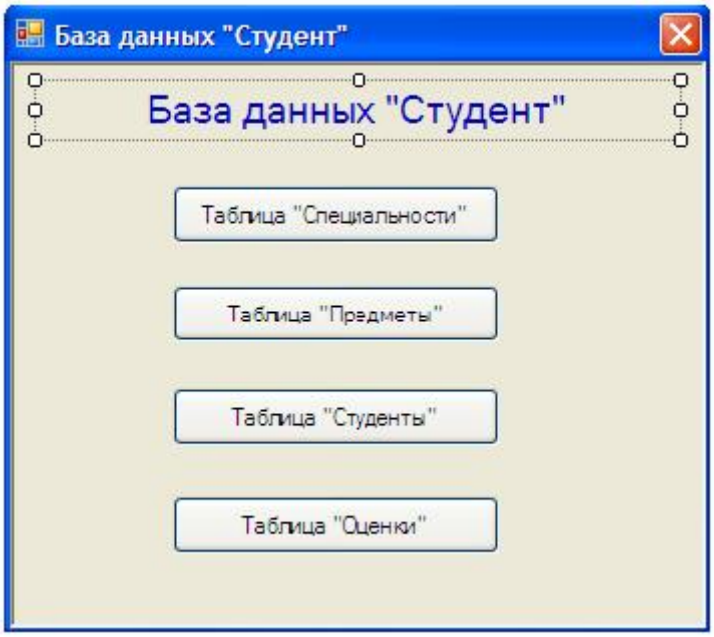
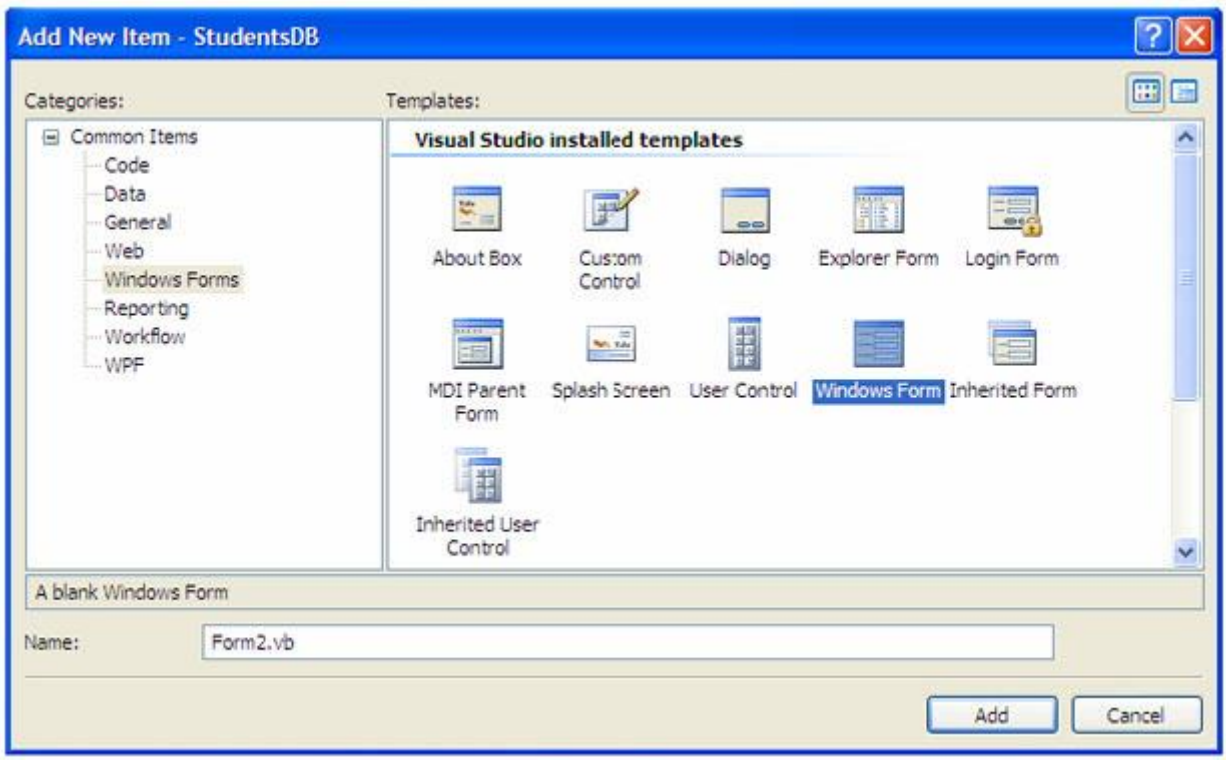


Рис. 18.3.

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):

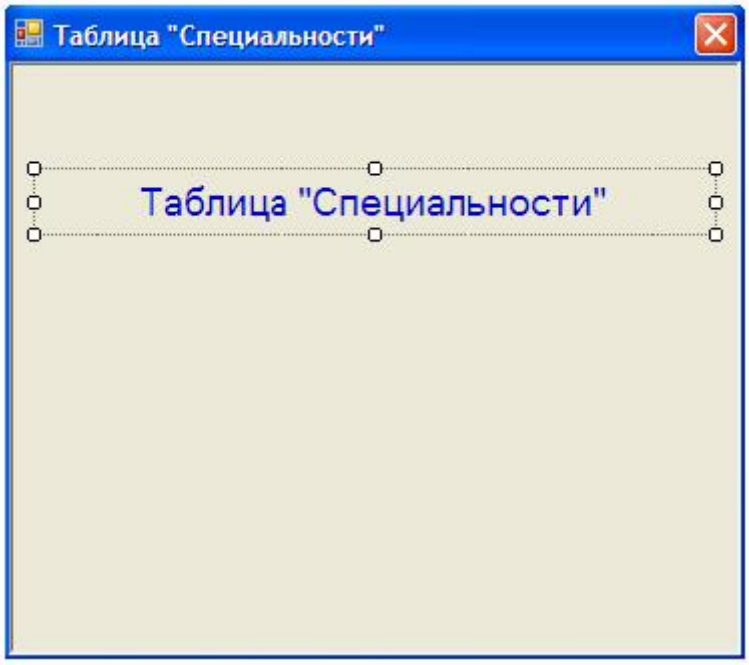


Рис. 18.6.

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).

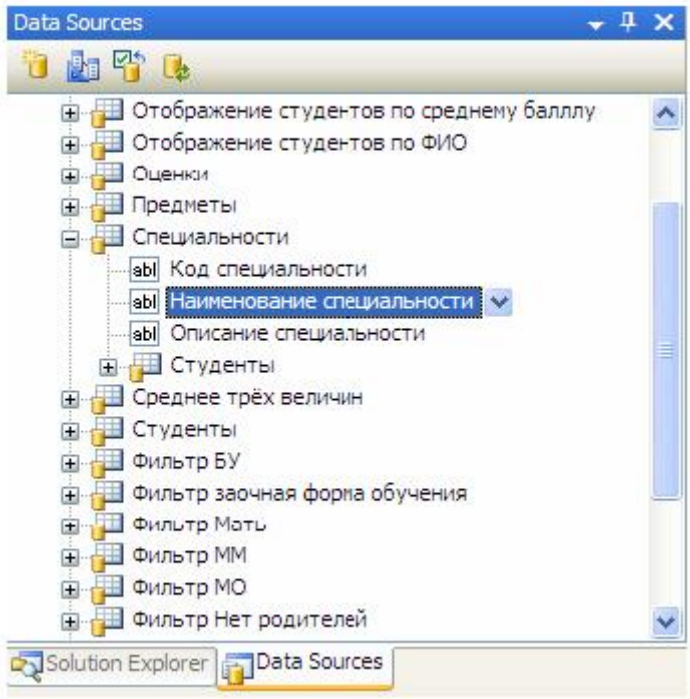


Рис. 18.7.

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).

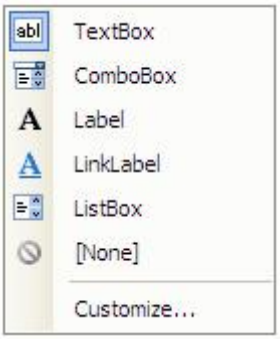
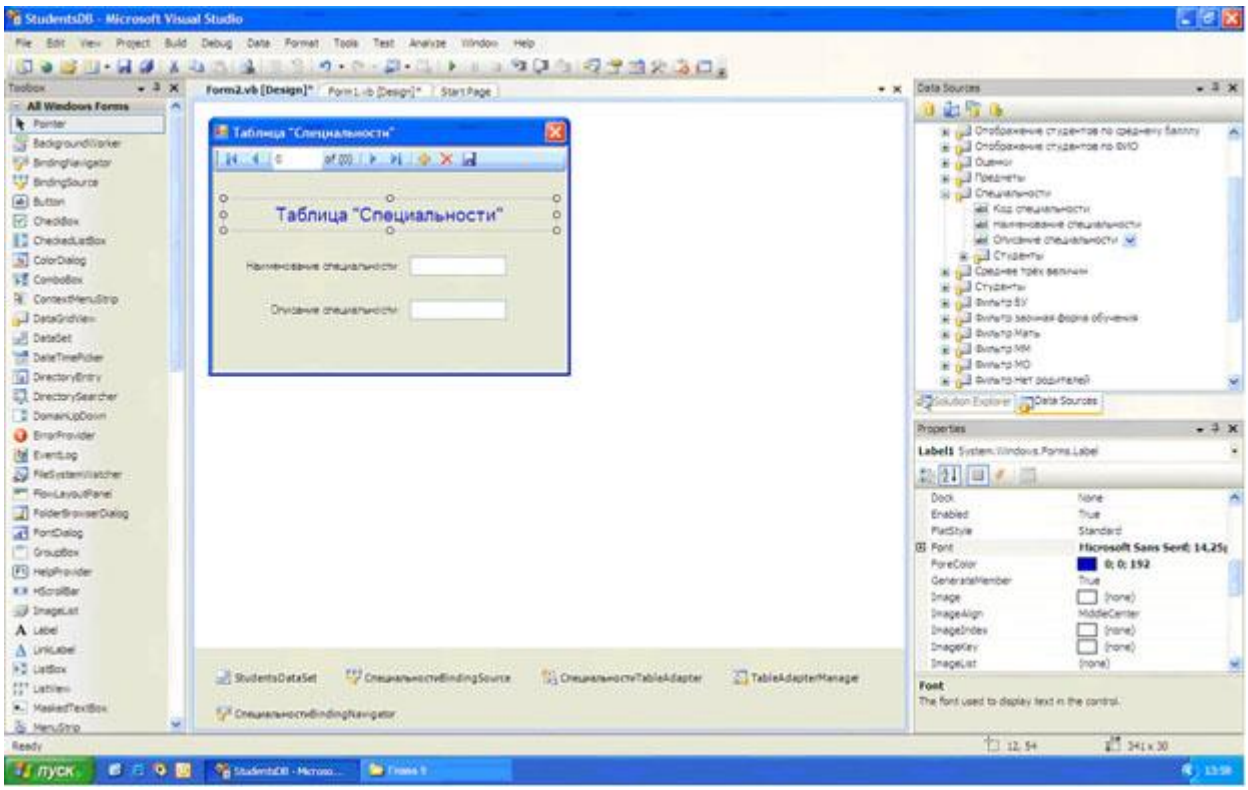


Рис. 18.8.

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).

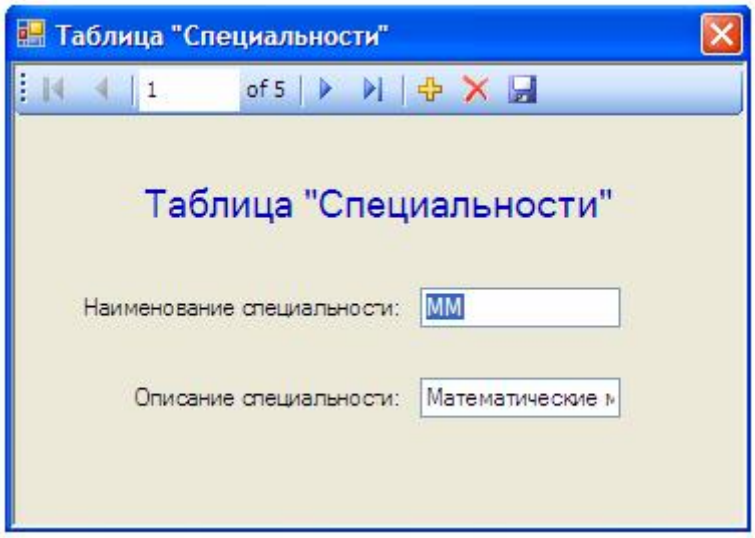


Рис. 18.11.

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):

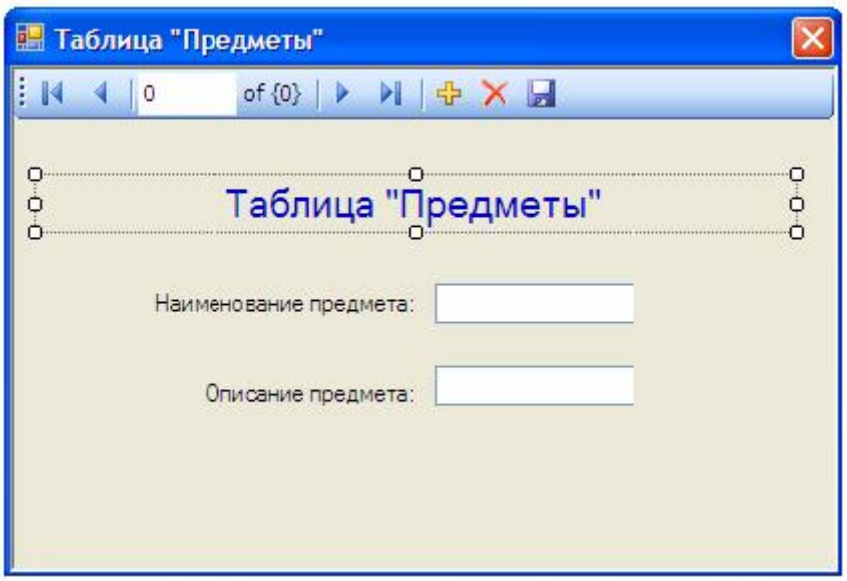


Рис. 18.12.

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):

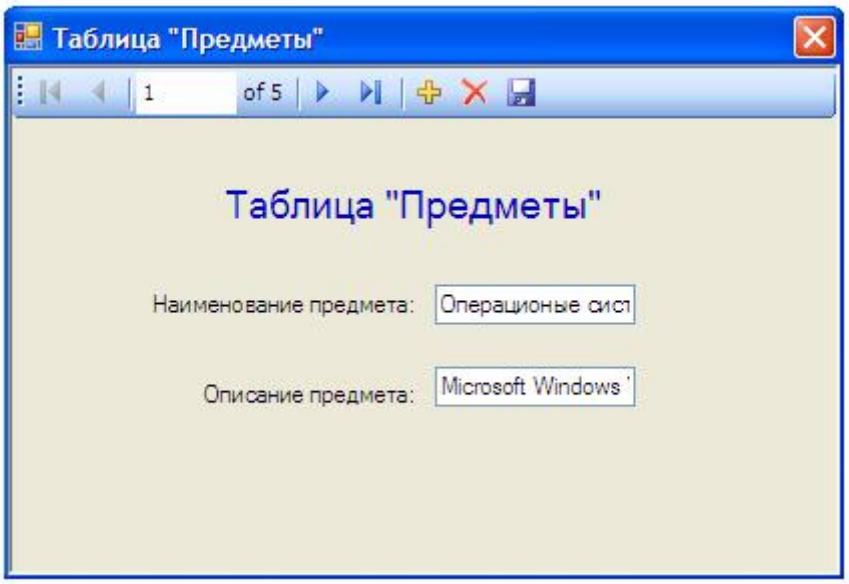


Рис. 18.14.

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).

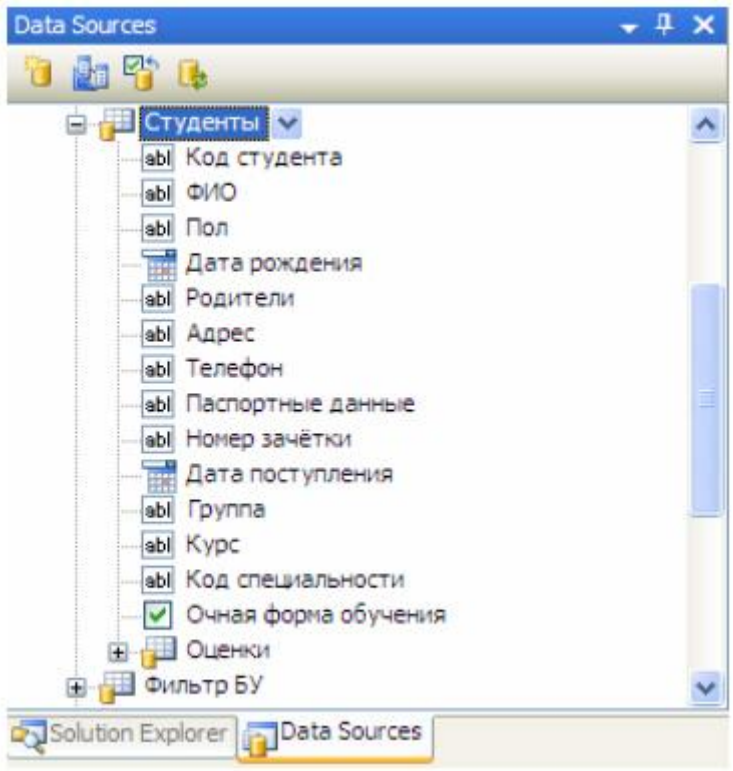


Рис. 18.15.

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):

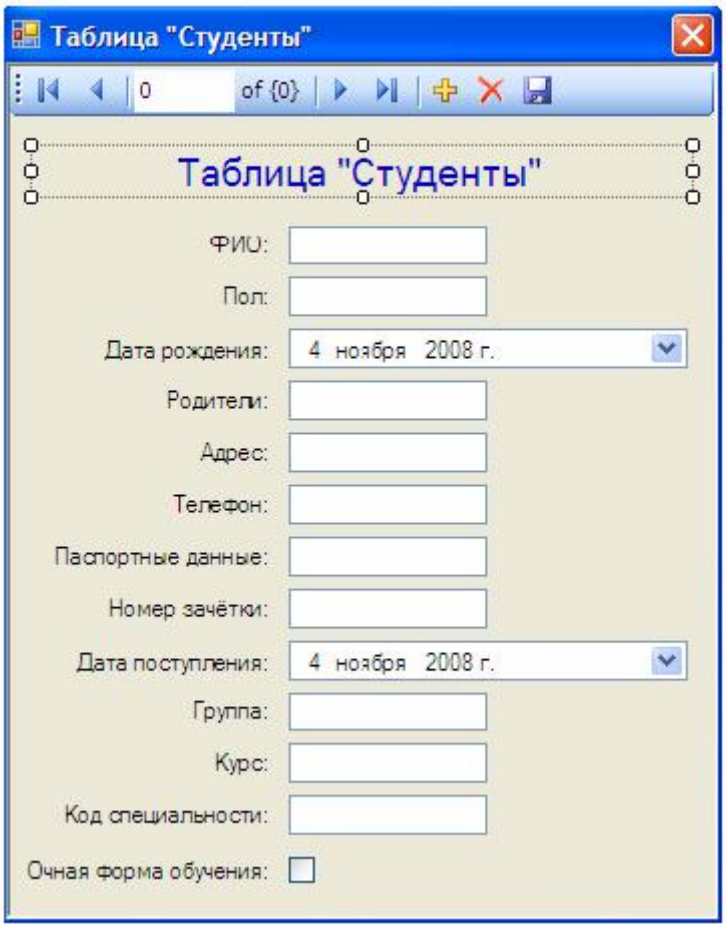


Рис. 18.16.

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).

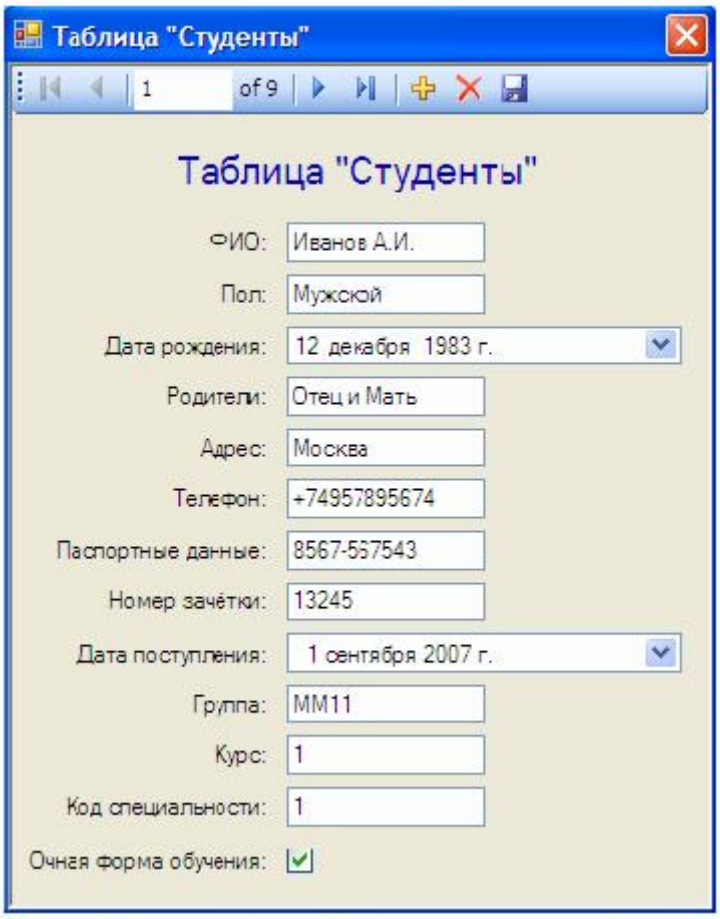


Рис. 18.18.

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):

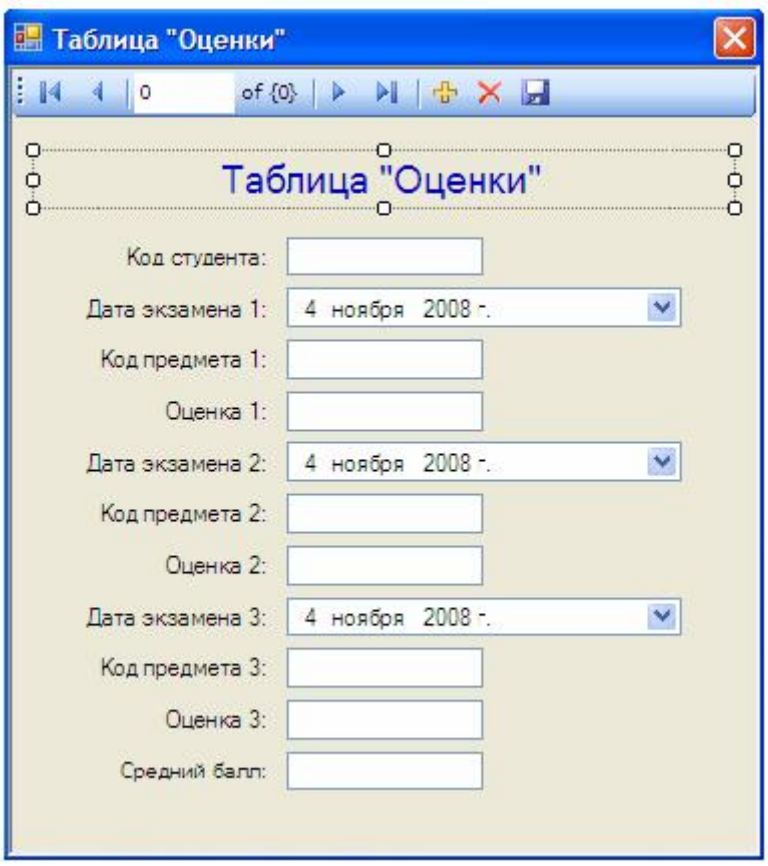


Рис. 18.19.

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).

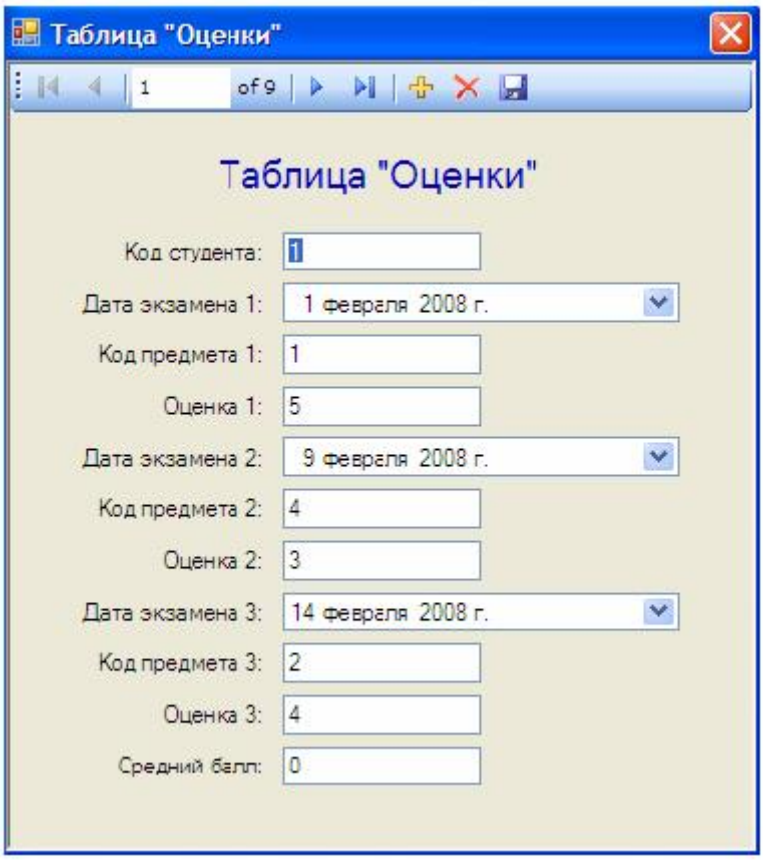


Рис. 18.21.

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).

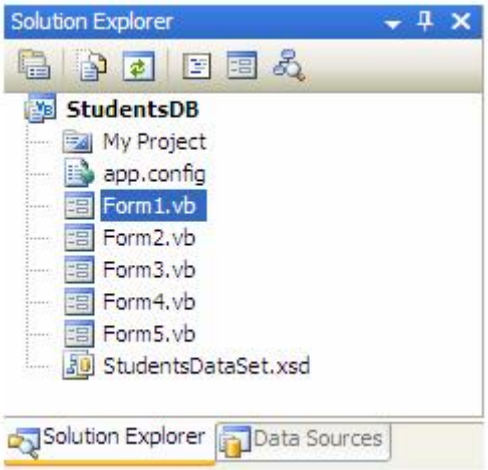


Рис. 18.22.

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.

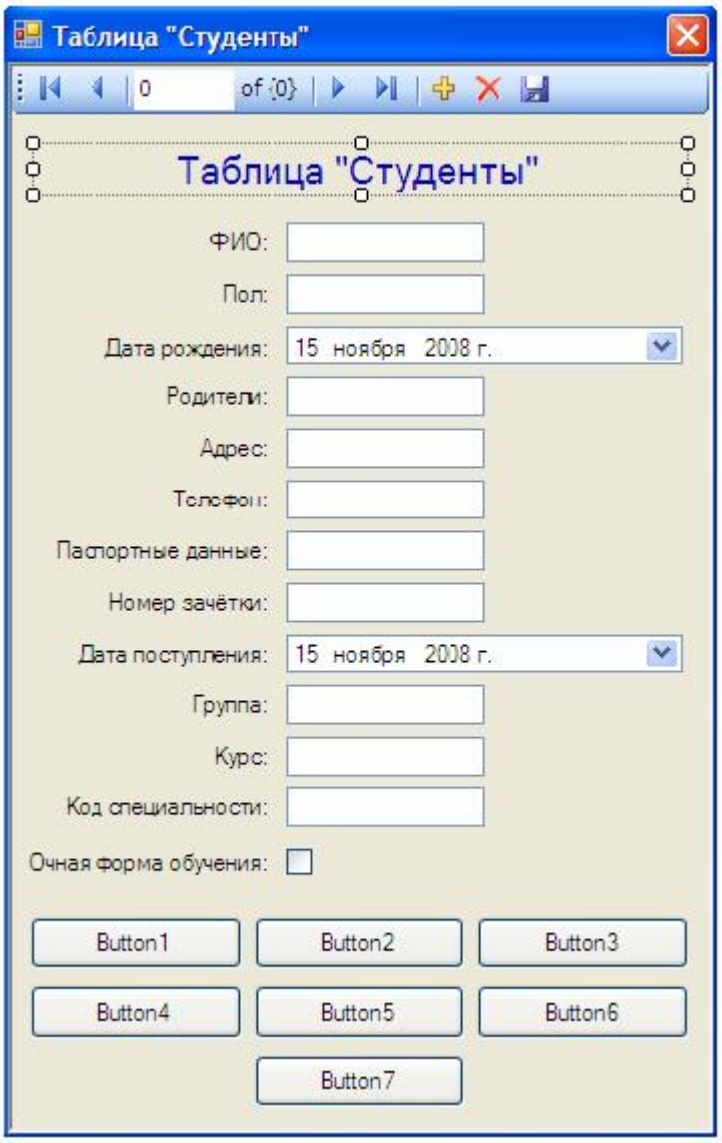


Рис. 20.1.

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

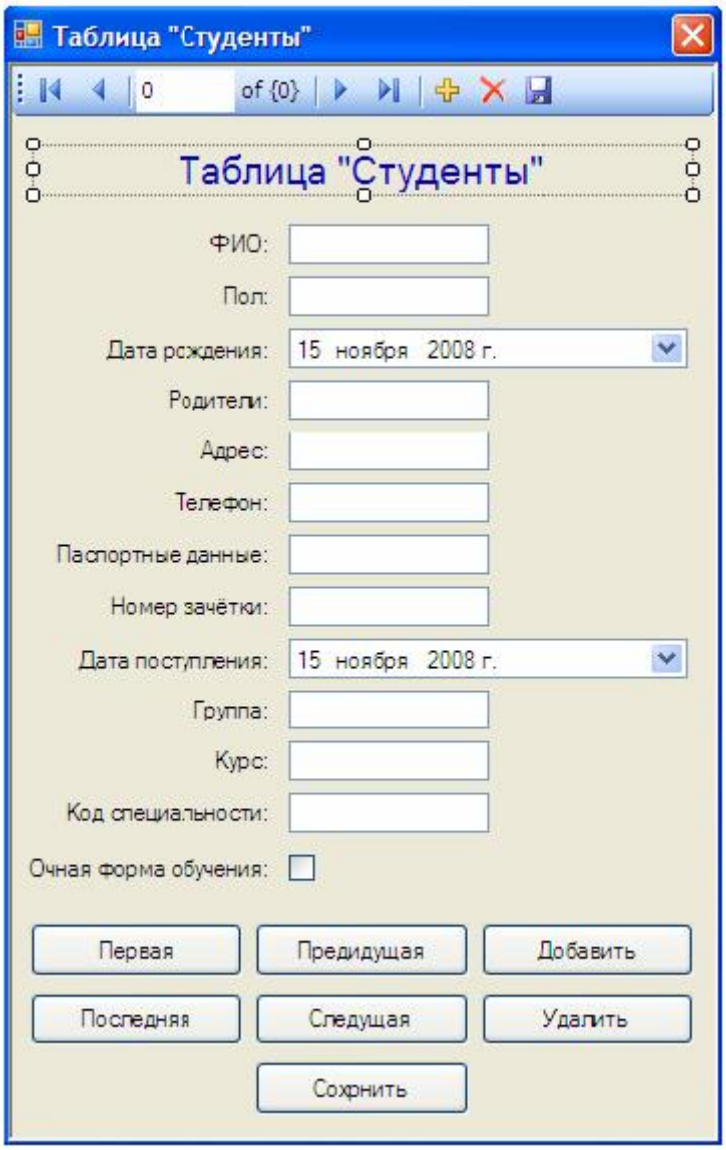


Рис. 20.2.

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):

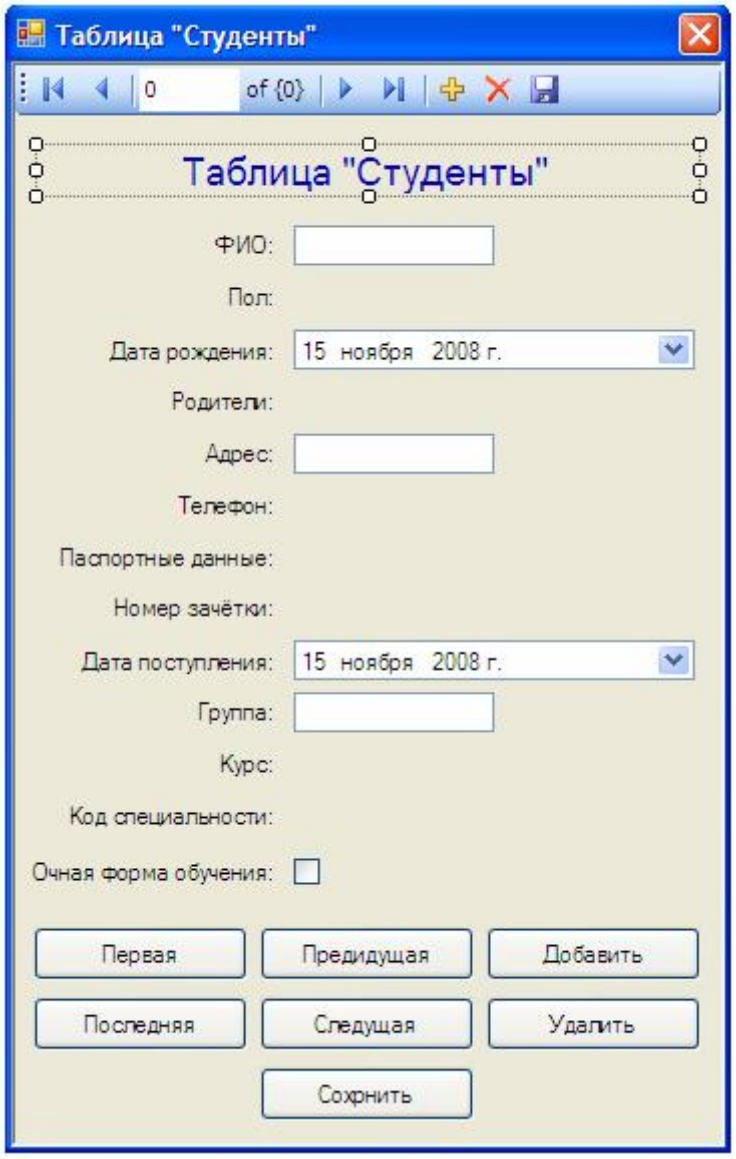
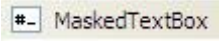


Рис. 20.10.

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.

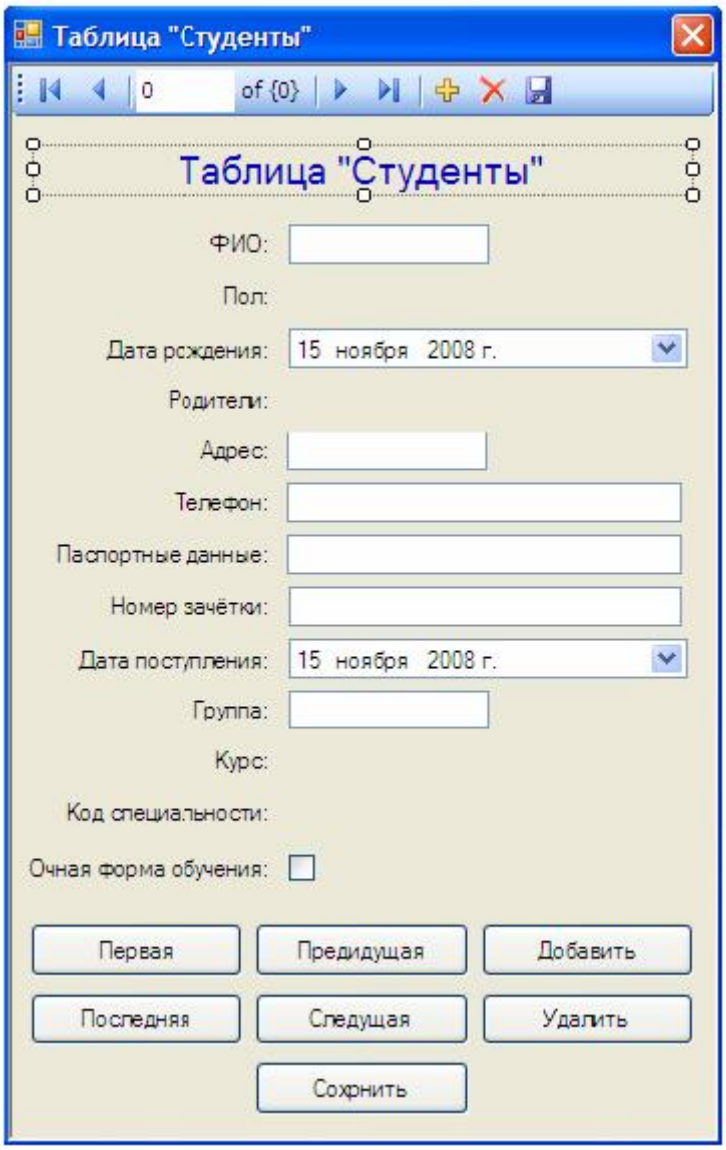


Рис. 20.11.

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).

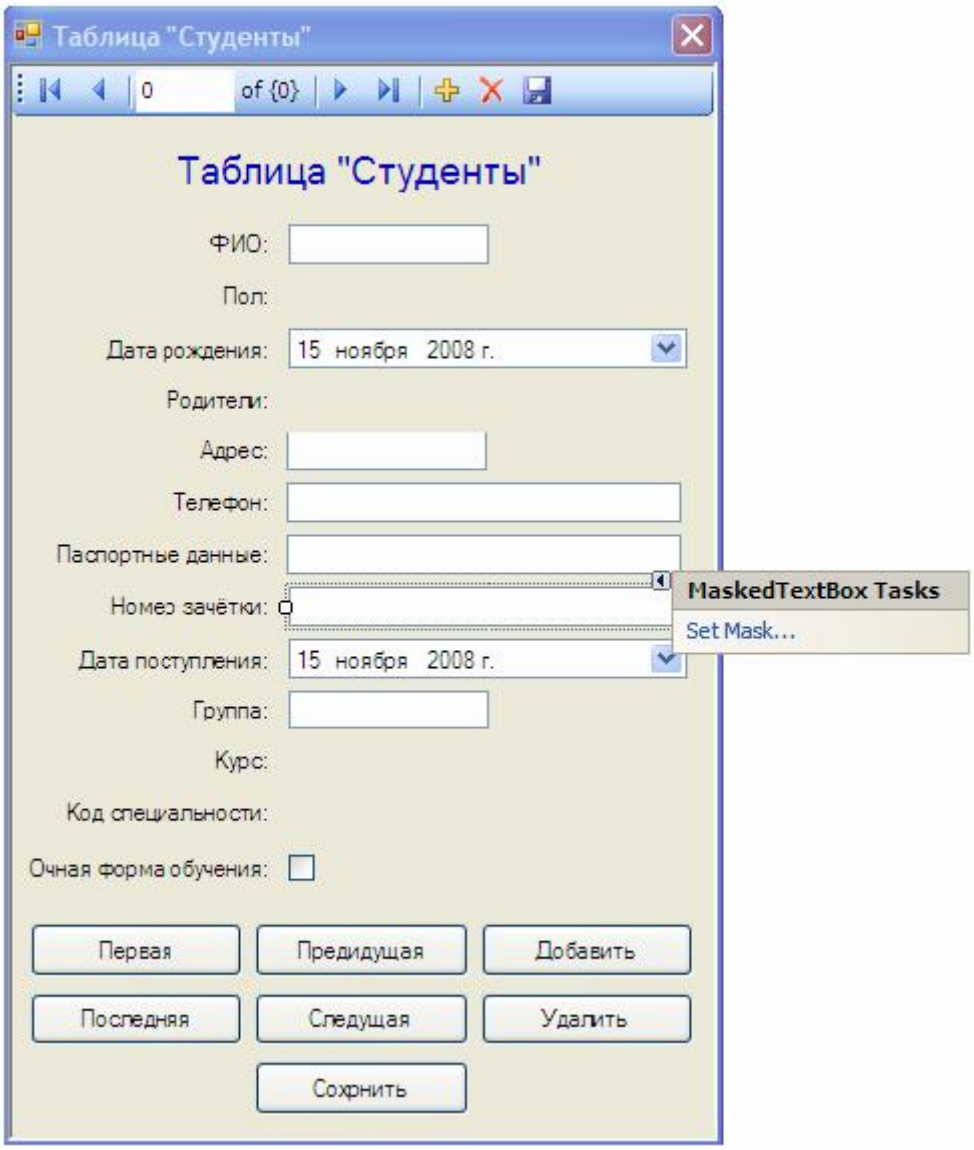


Рис. 20.12.

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).

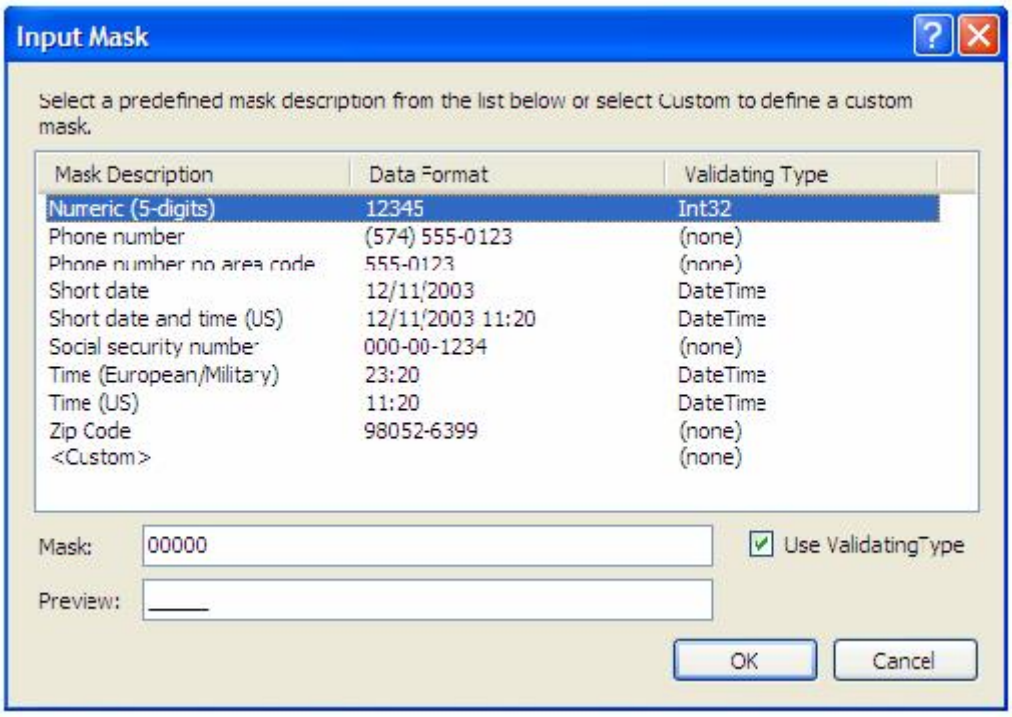


Рис. 20.13.

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.

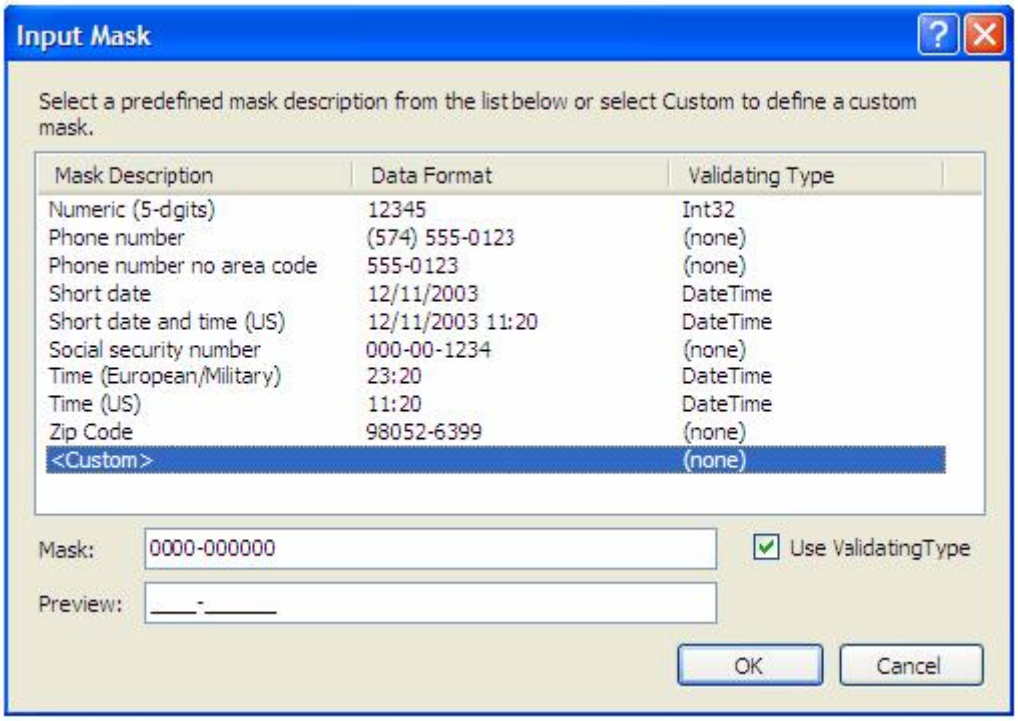


Рис. 20.14.

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.

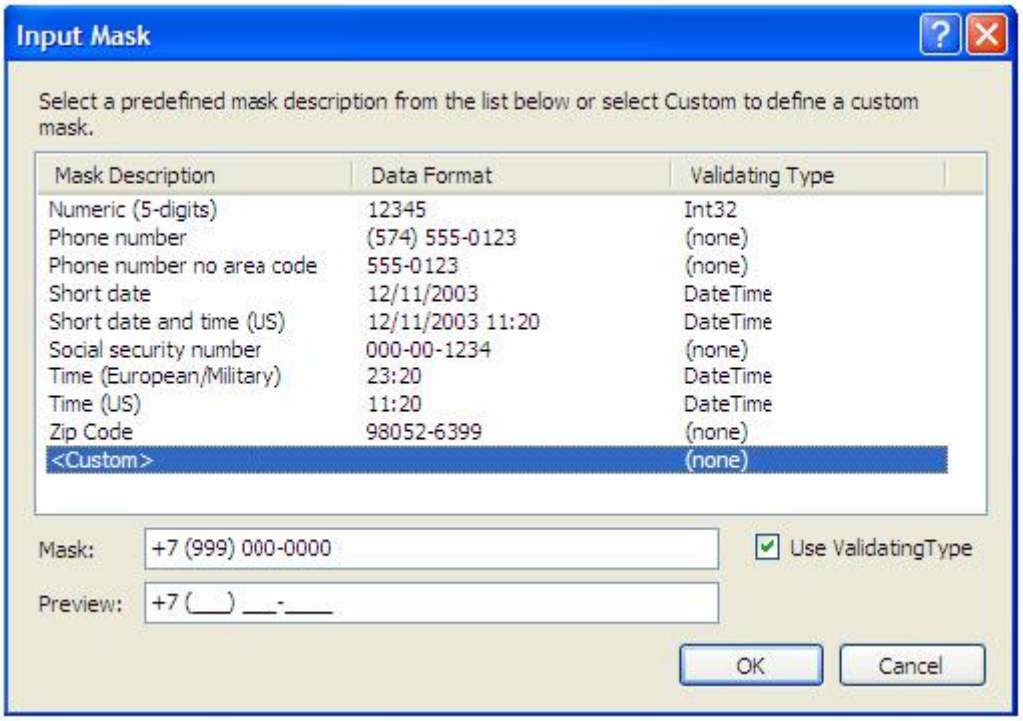


Рис. 20.15.
Теперь нам необходимо подключить созданные текстовые поля ввода по маске к соответствующим полям. Для этого с панели **"Источники данных" (DataSources)** перетащите поле **"Номер зачетки"** на текстовое поле ввода по маске, расположенное справа от надписи **"Номер зачетки"**. Проделайте такую 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).

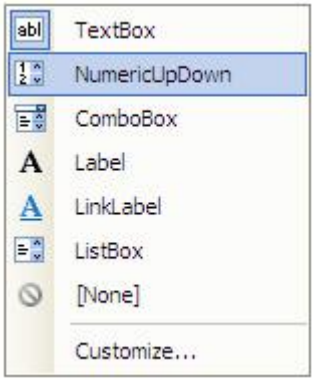


Рис. 20.16.
Затем перетащите поле на форму мышью, расположив, его справа от надписи **"Курс"**.

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).

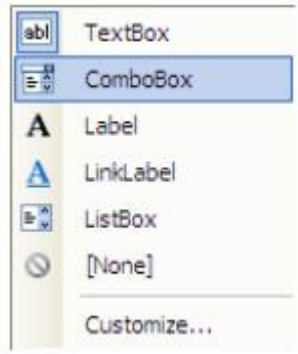


Рис. 20.17.
Такую же операцию проделайте с полем **"Родители"**. Затем перетащите мышью поля на форму, расположив их напротив соответствующих надписей. Удалите лишние надписи.
Теперь заполним выпадающие списки. Выделите выпадающий список, отображающий поле **"Пол"**. На панели свойств (**Properties**)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

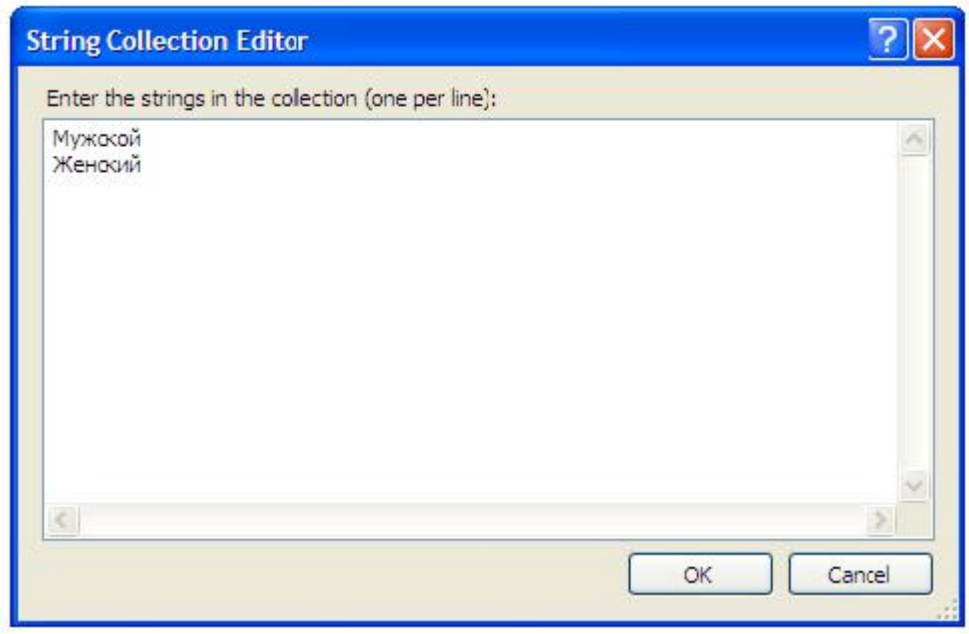


Рис. 20.18.
В появившемся окне в отдельных строках наберите элементы выпадающего списка: **"Мужской"** и **"Женский"** (рис. 20.18). Затем нажмите кнопку **"Ok"**.
Для выпадающего списка, отображающего поле **"Родители"**, проделайте аналогичную операцию, только в качестве пунктов списка задайте: **"Отец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

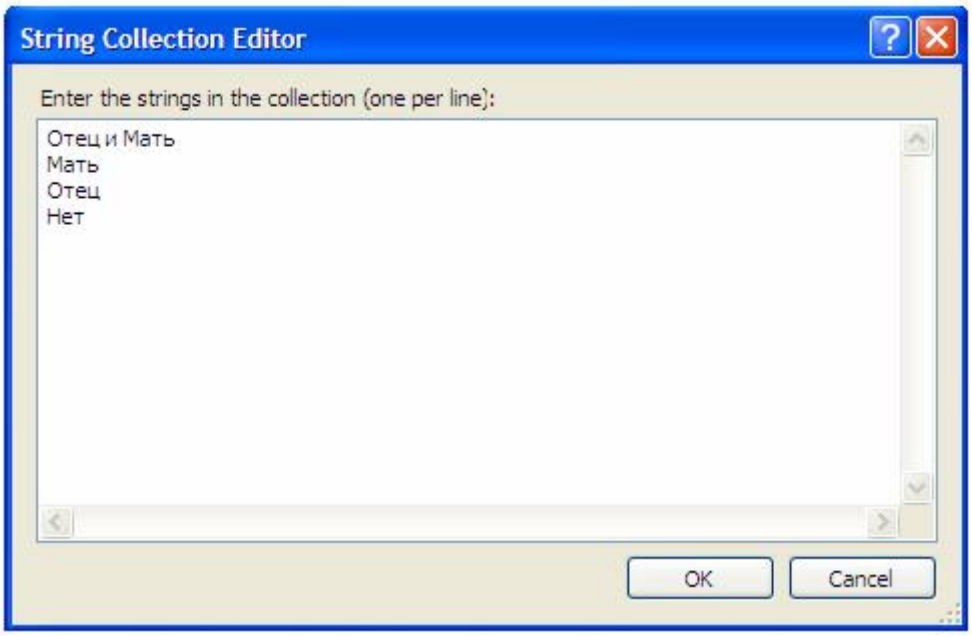
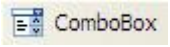


Рис. 20.19.

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).

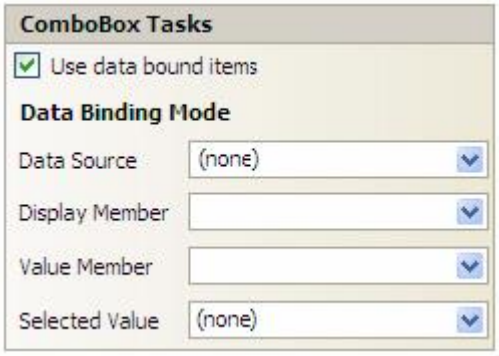


Рис. 20.20.

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).

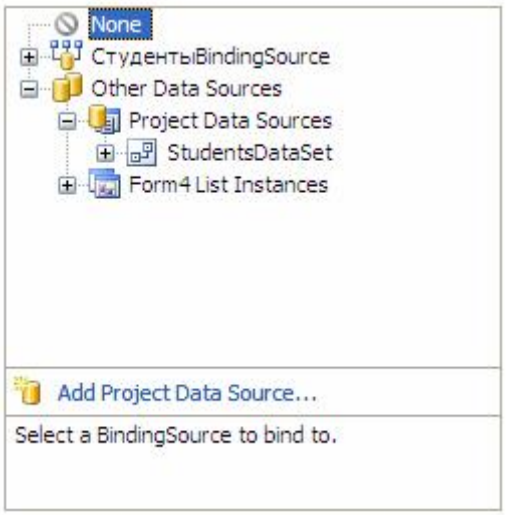


Рис. 20.21.

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):

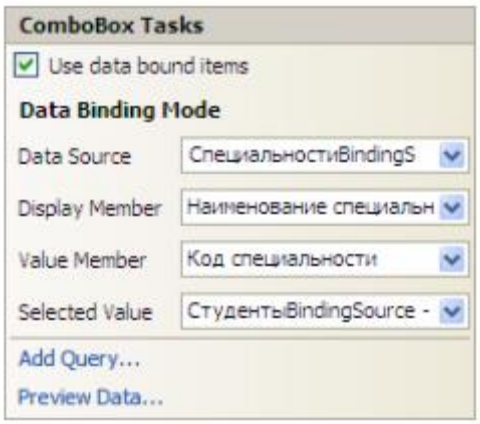
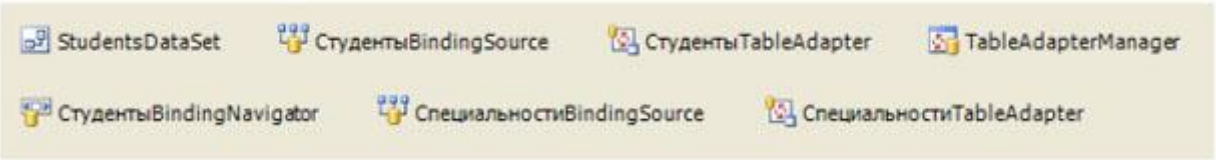


Рис. 20.22.

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.

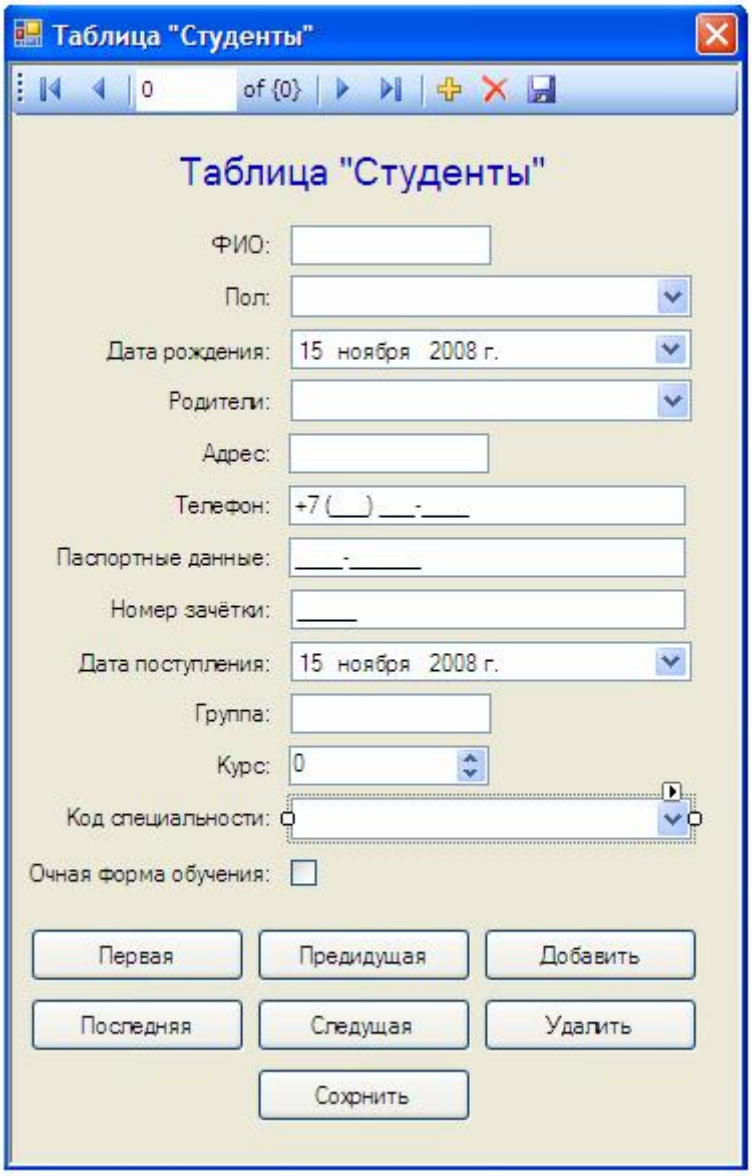


Рис. 20.24.

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):

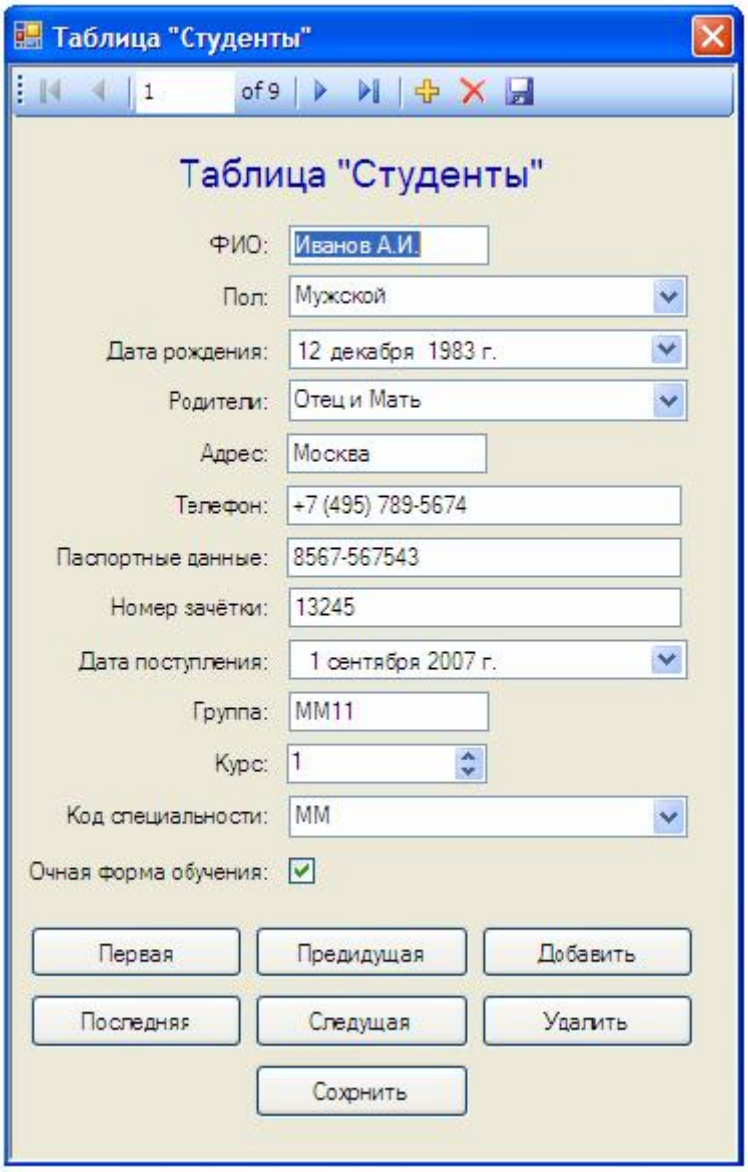


Рис. 20.25.

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).

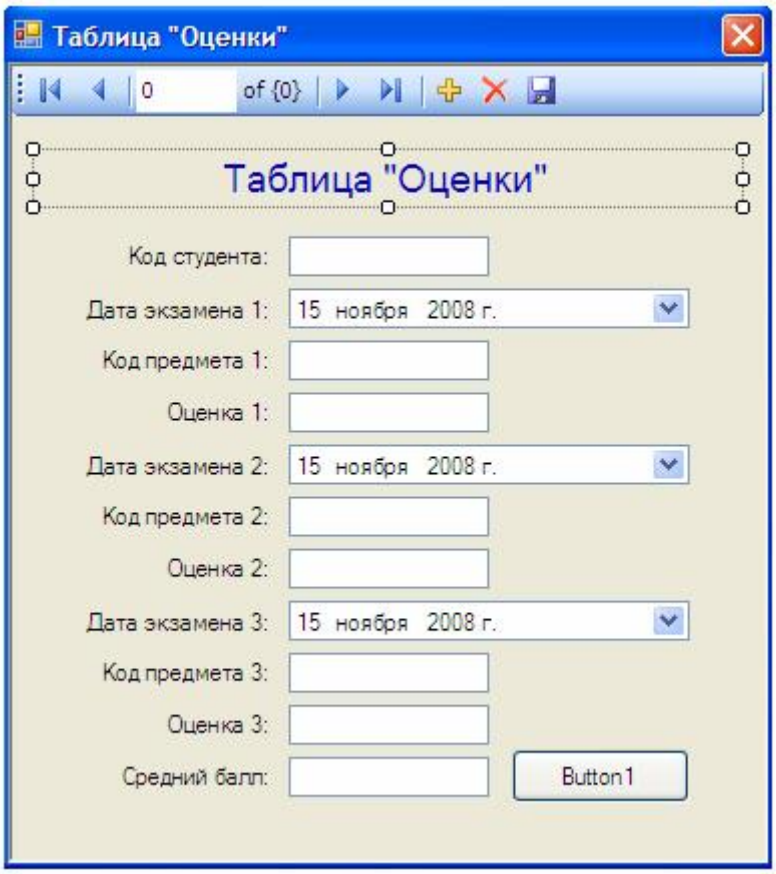


Рис. 20.26.

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

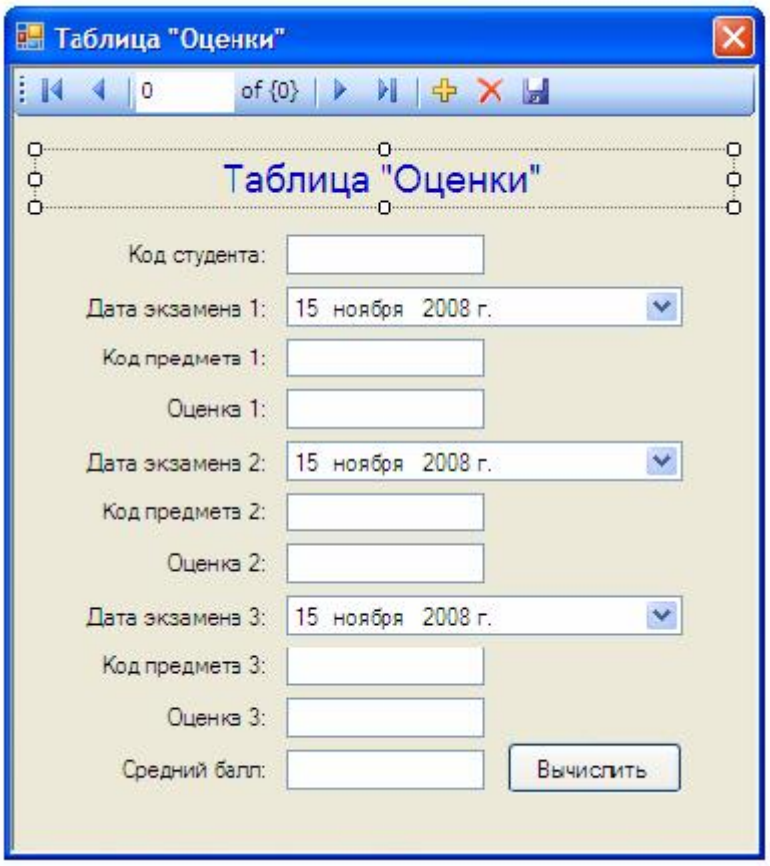


Рис. 20.27.

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).

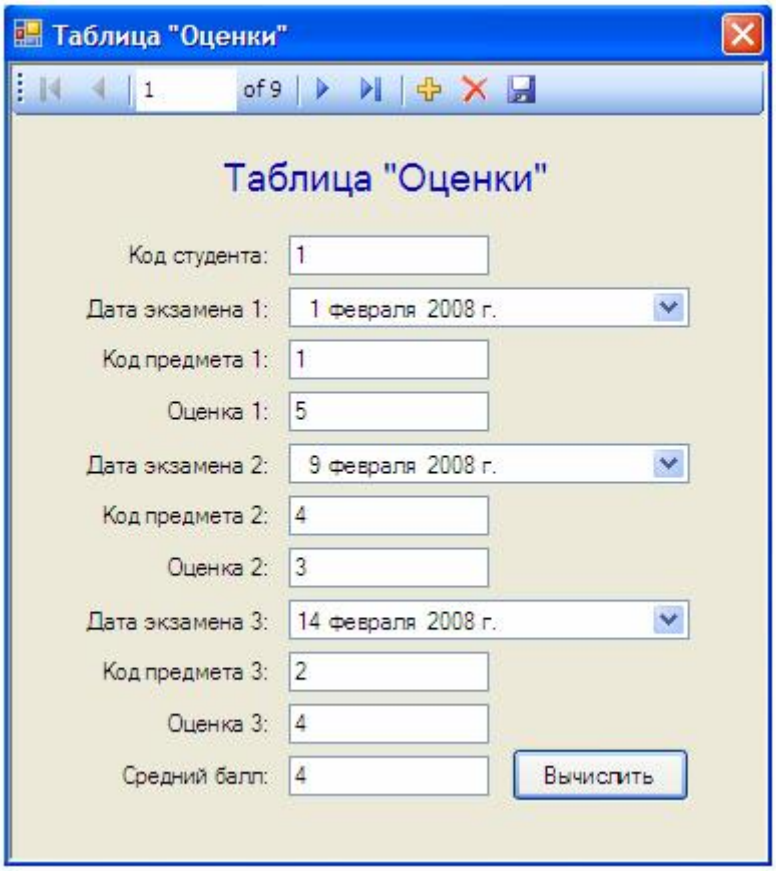


Рис. 20.29.

На этом мы заканчиваем рассмотрение ленточных форм и переходим к рассмотрению табличных форм.

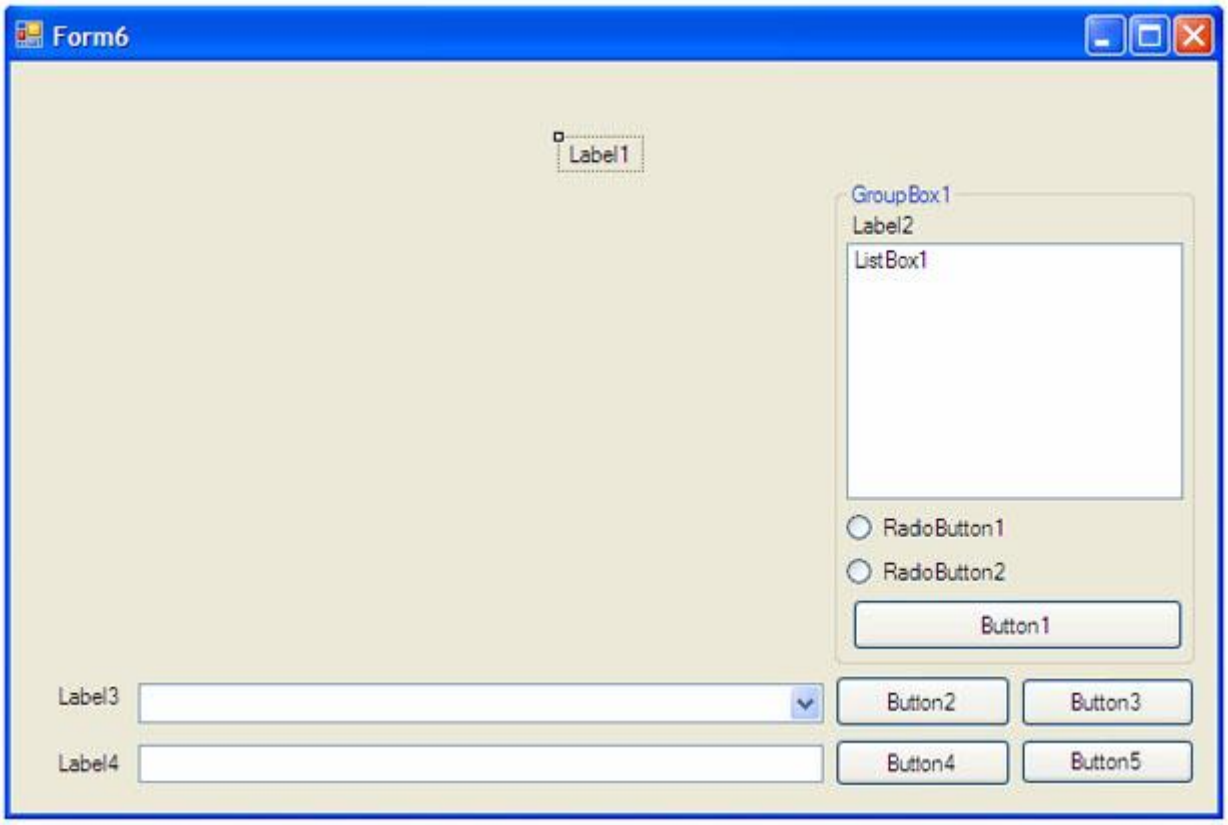
Перейдем теперь к созданию табличных форм для отображения данных. В данной главе также затрагиваются вопросы фильтрации и сортировки данных, а также реализуется поиск информации в таблице.

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).

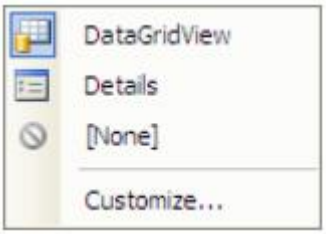
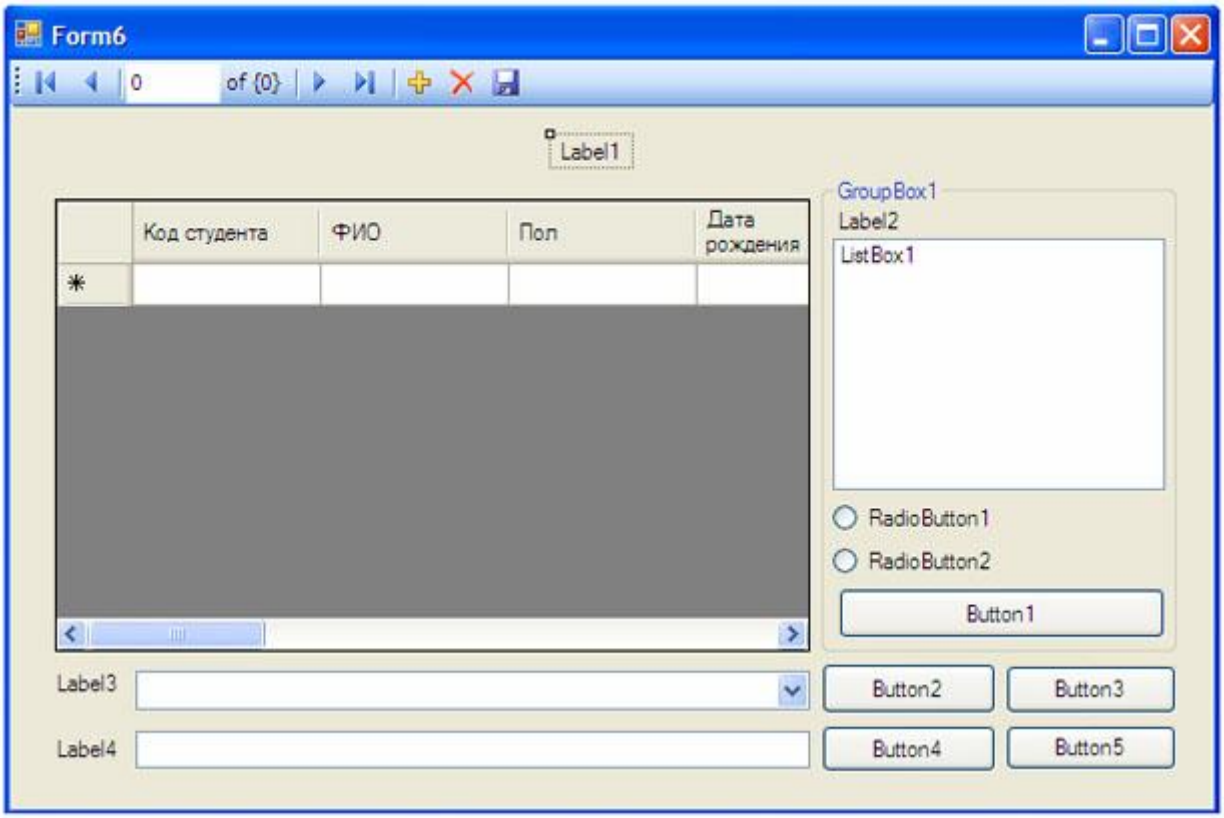


Рис. 22.2.

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"**.

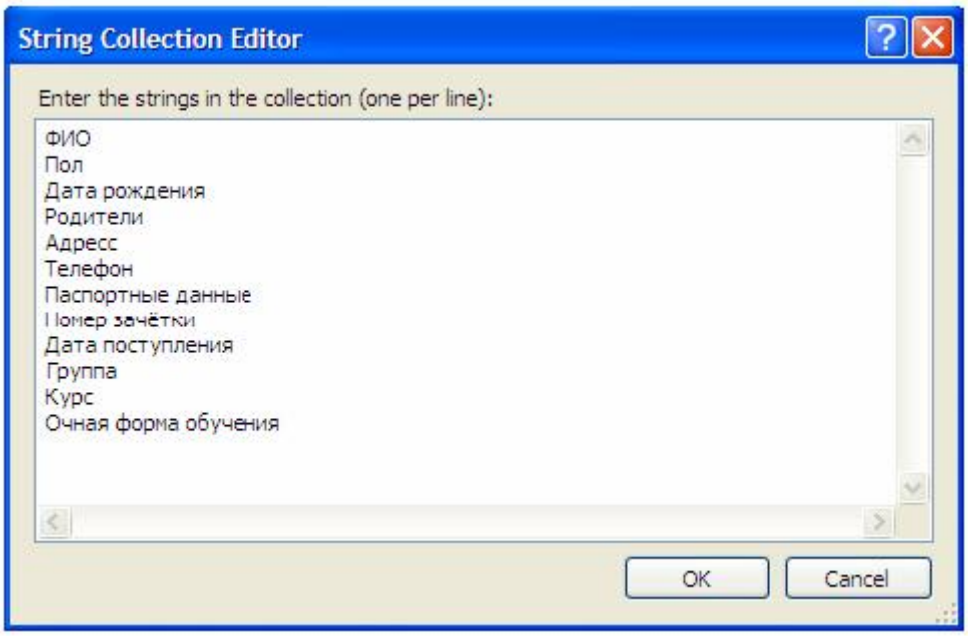


Рис. 22.5.
Настроим таблицу для отображения данных, удалив из нее поля с кодами. Выделите таблицу на форме и отобразите ее меню действий, щелкнув **ЛКМ** по кнопке


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

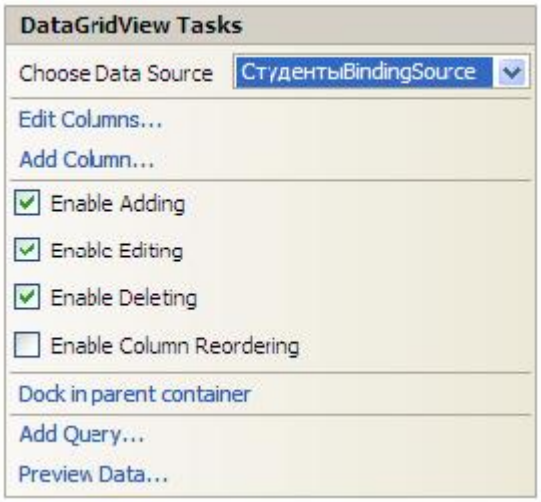


Рис. 22.6.
Появится окно настройки свойств полей таблицы **"Edit Columns"** (рис. 22.7).

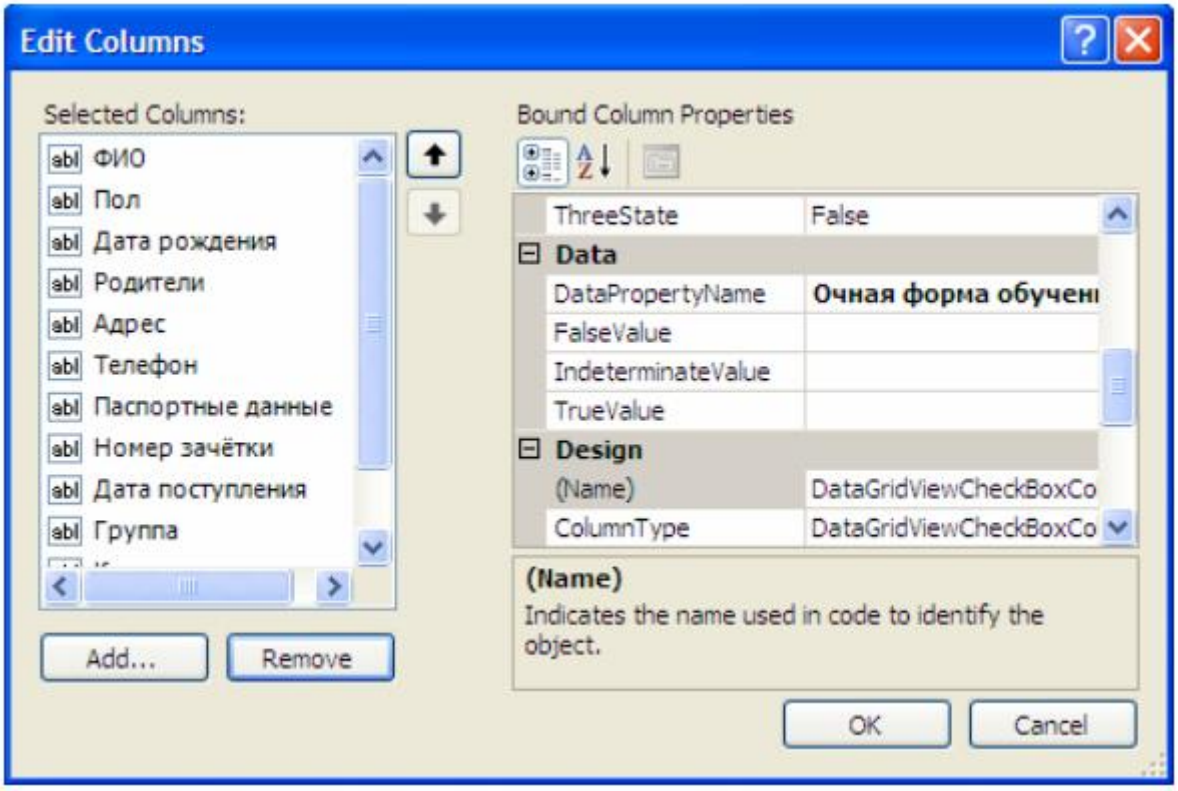


Рис. 22.7.

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).

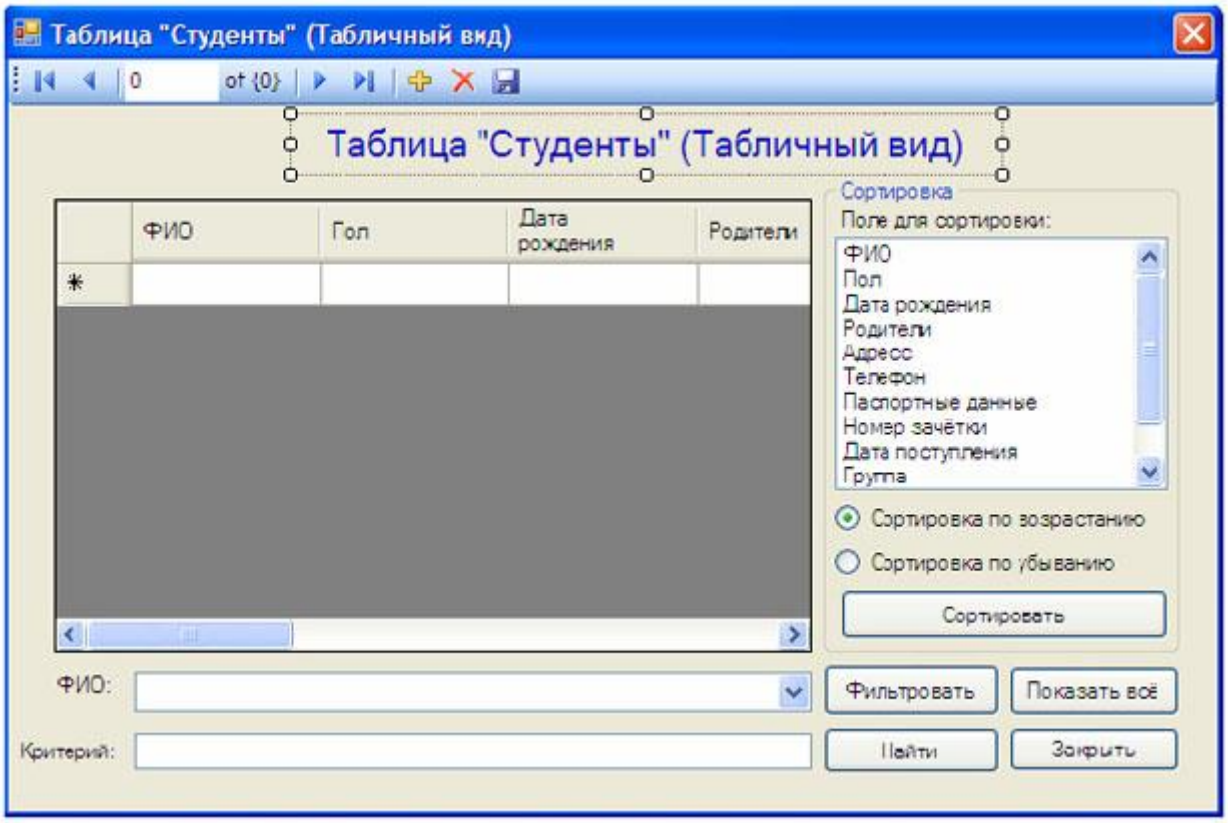
Рис. 22.8.

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.

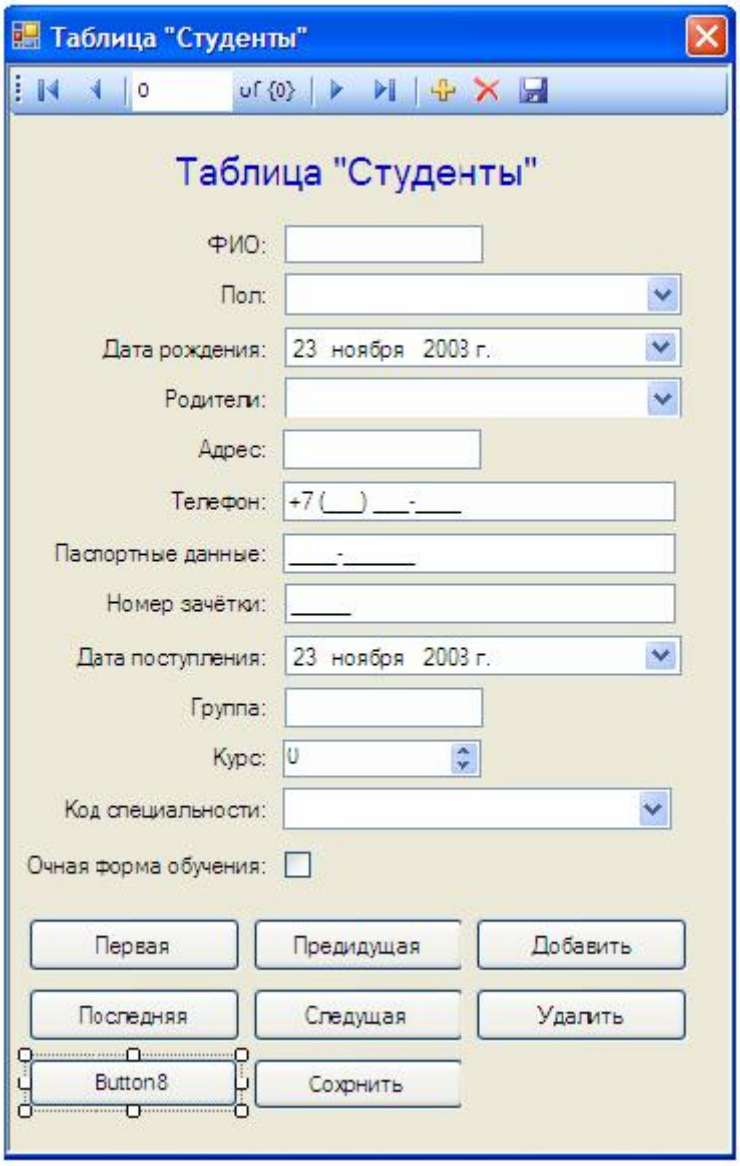


Рис. 22.17.

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

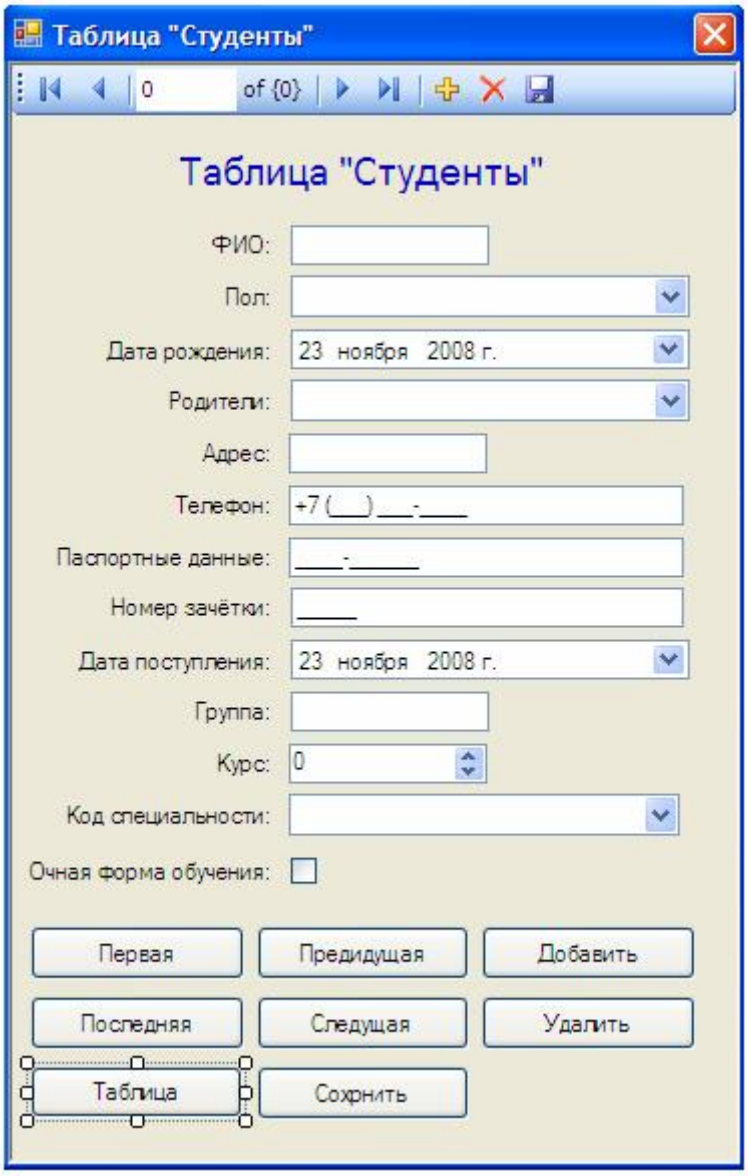


Рис. 22.18.

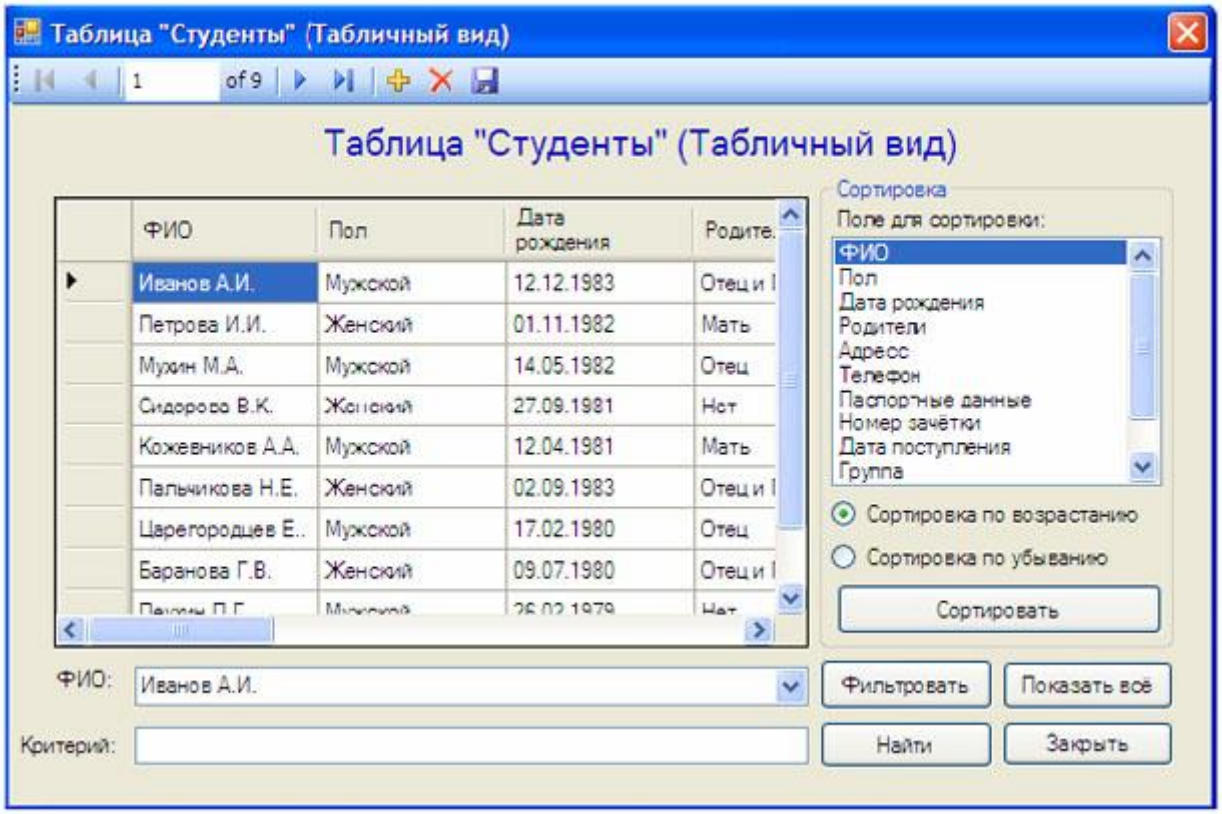
Подключим к кнопке **"Таблица"** созданную ранее табличную форму (**Form6**). Для этого дважды щелкните **ЛКМ** по кнопке **"Таблица"** и в появившейся процедуре **"Button8_Click"** наберите команду "Form6.Show" (рис. 22.19).

```
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):

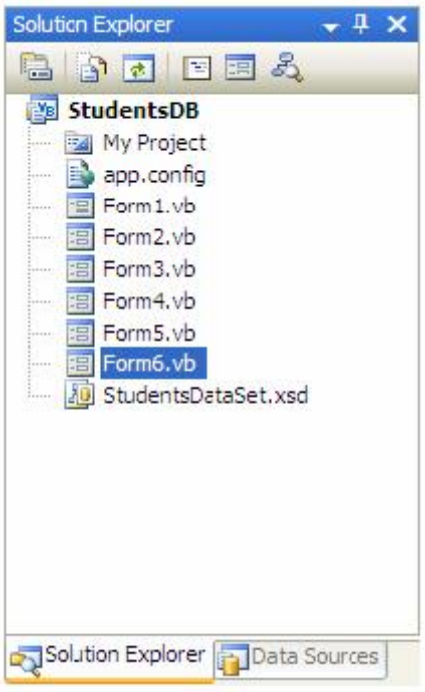


Рис. 22.21.

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

Задание.

В программе Microsoft Visual Studio создать приложение для работы с базой данных по учету успеваемости студентов, созданной в предыдущей лабораторной работе.
Создать главную кнопочную форму, простые и сложные ленточные формы для работы с данными, табличные формы и отчет «Студенты».

Контрольные вопросы

1. Настройки свойств формы.
2. Главная кнопочная форма.
3. Как настроить маски ввода.
4. Как создать выпадающий список
5. Как создать новую табличную форму.
6. Как обеспечить фильтрацию и сортировку данных.
7. Как реализуется поиск информации в таблице.

Работа с литературой:

Рекомендуемые источники информации (№ источника)			
Основная	Дополнительная	Методическая	Интернет-ресурсы
1-2	1-2	1-2	1-3

Оценочные средства: устный отчет к лабораторной работе (См.: Фонд оценочных средств)

Лабораторная работа № 3. «Выполнение индивидуальных заданий по проектированию информационных систем в Visual Studio 2012. Создание отчетов и диаграмм.»

Форма проведения лабораторная работа (3 часа)

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).

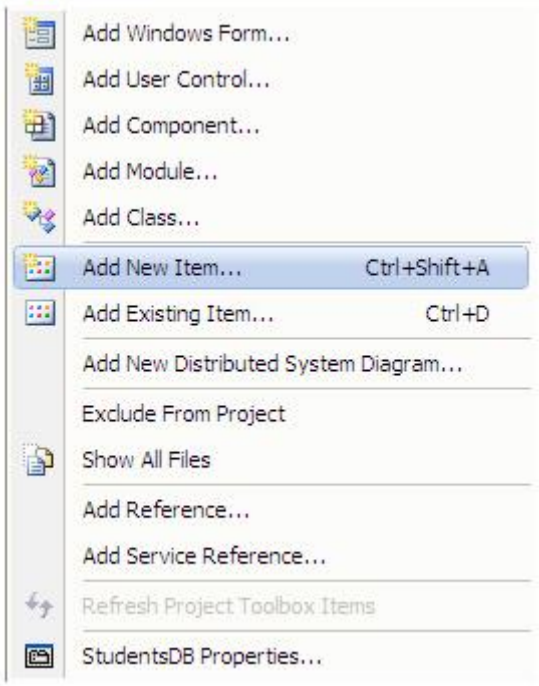
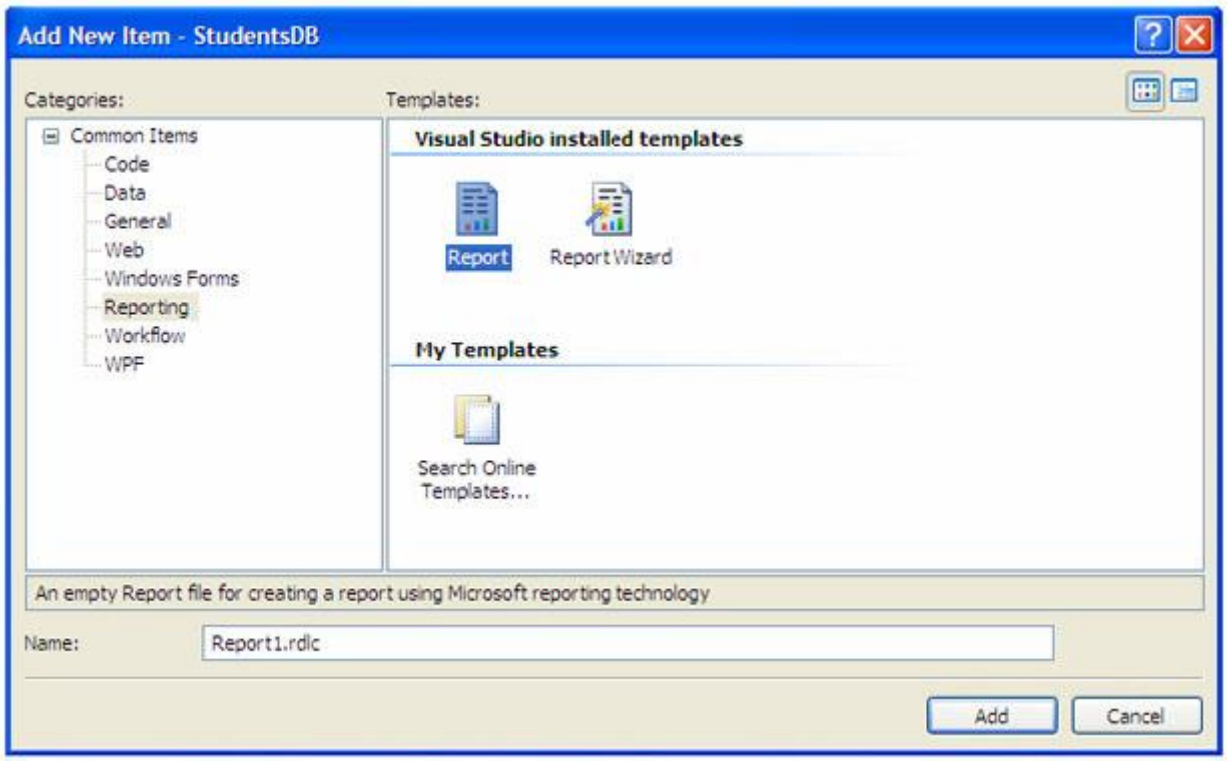


Рис. 24.1.

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).

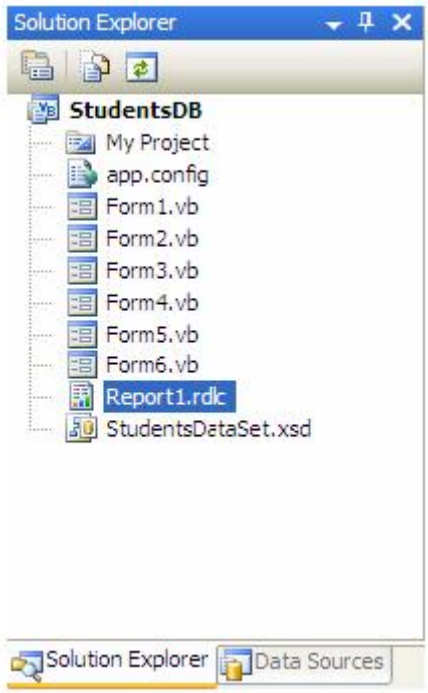
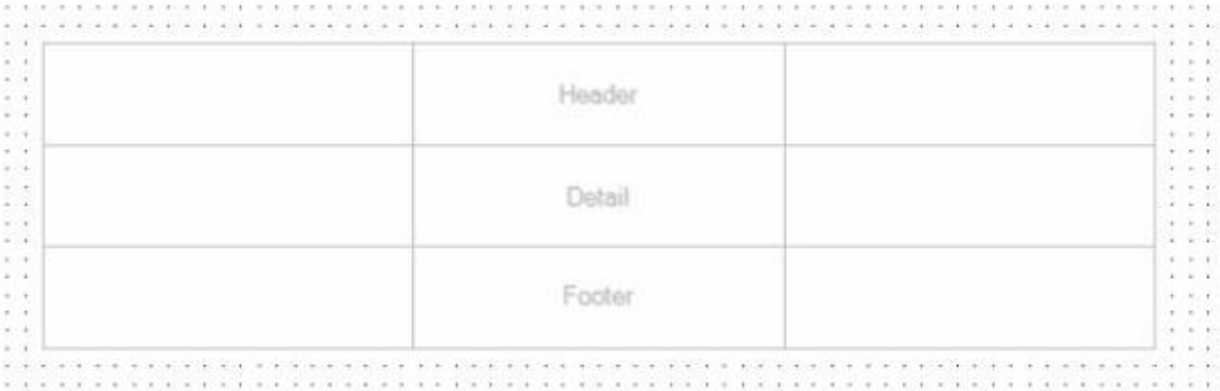


Рис. 24.3.

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

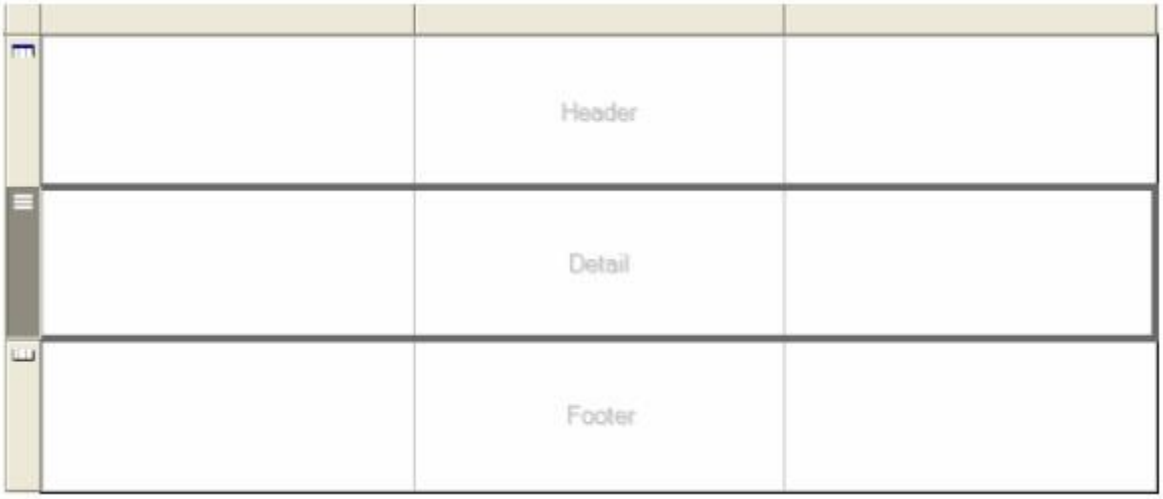


Рис. 24.5.
Для вставки новой строки щелкните **ПКМ** по заголовку выделенной строки


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

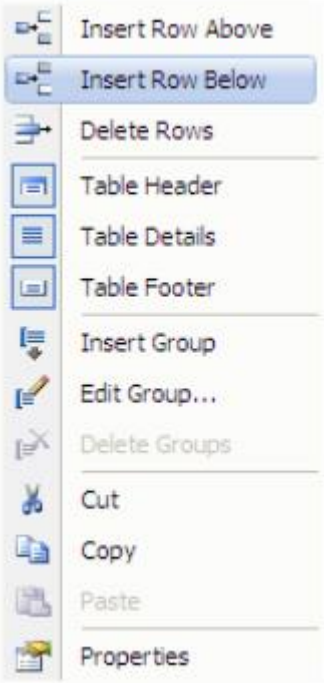


Рис. 24.6.
Проделайте эту операцию одиннадцать раз. Таблица примет вид (рис. 24.7)

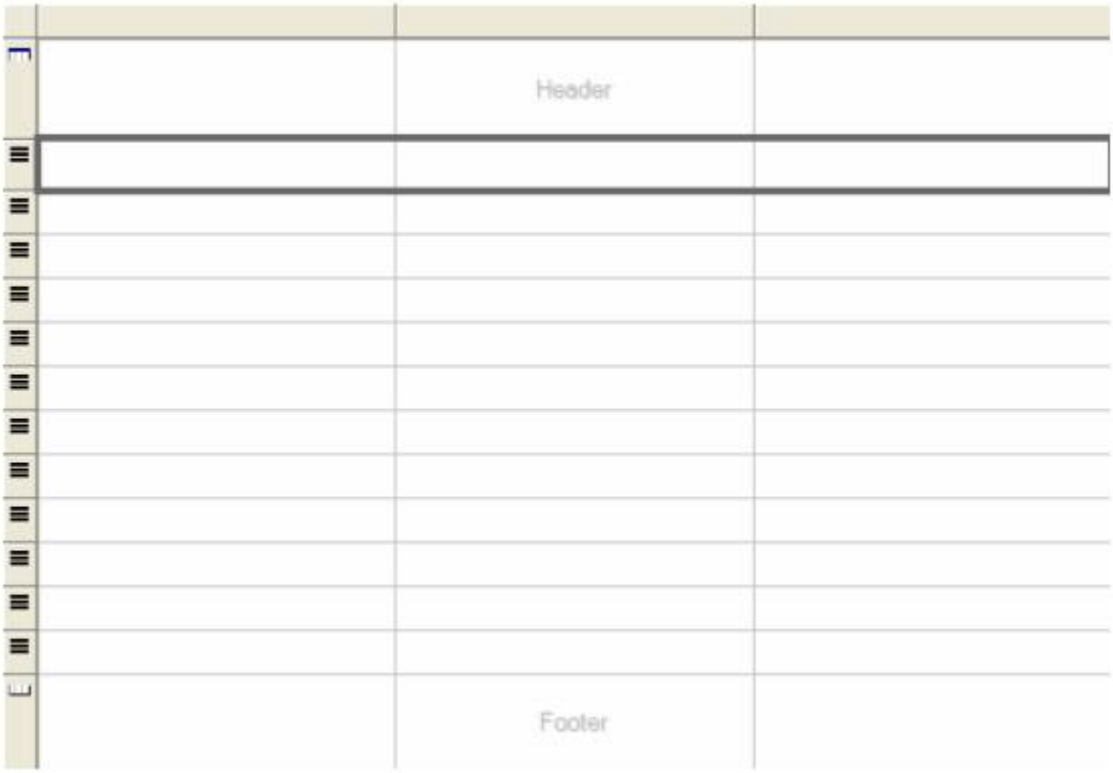


Рис. 24.7.
Наша таблица имеет три столбца. Один столбец является лишним, так как нам необходимо всего два столбца: один для отображения полей таблицы **"Студенты"**, а другой для подписей к ним. Поэтому удалим первый столбец таблицы. Щелкните **ПКМ** по заголовку первого столбца таблицы и в появившемся меню выберите пункт **"Delete Columns"** (Удалить строки) (рис. 24.8).

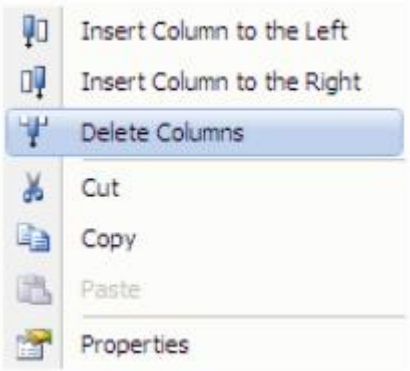


Рис. 24.8.
Теперь объединим ячейки в заголовке и примечании таблицы. Выделите ячейки заголовка таблицы, обведя их курсором мыши удерживая **ЛКМ**. Затем щелкните по выделению **ПКМ** и в появившемся меню выберите пункт **"Merge Cells"** (Объединить ячейки) (рис. 24.9).

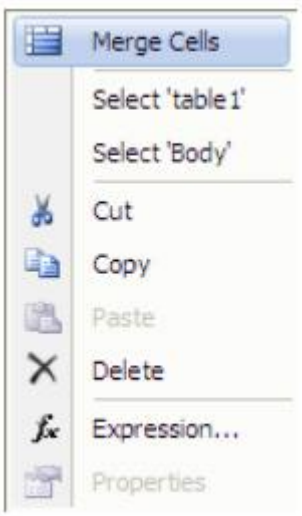


Рис. 24.9.

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

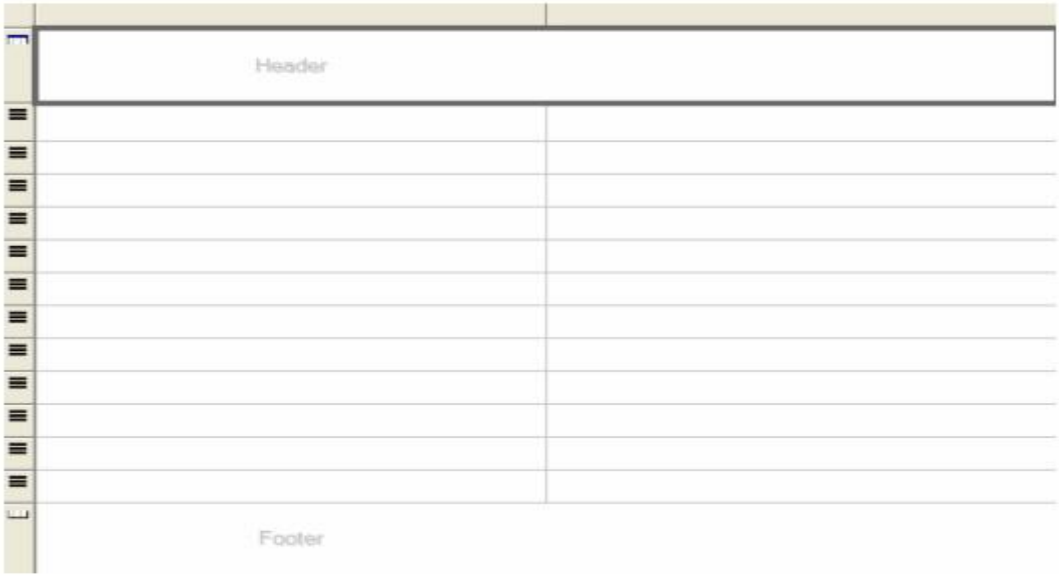


Рис. 24.10.

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.

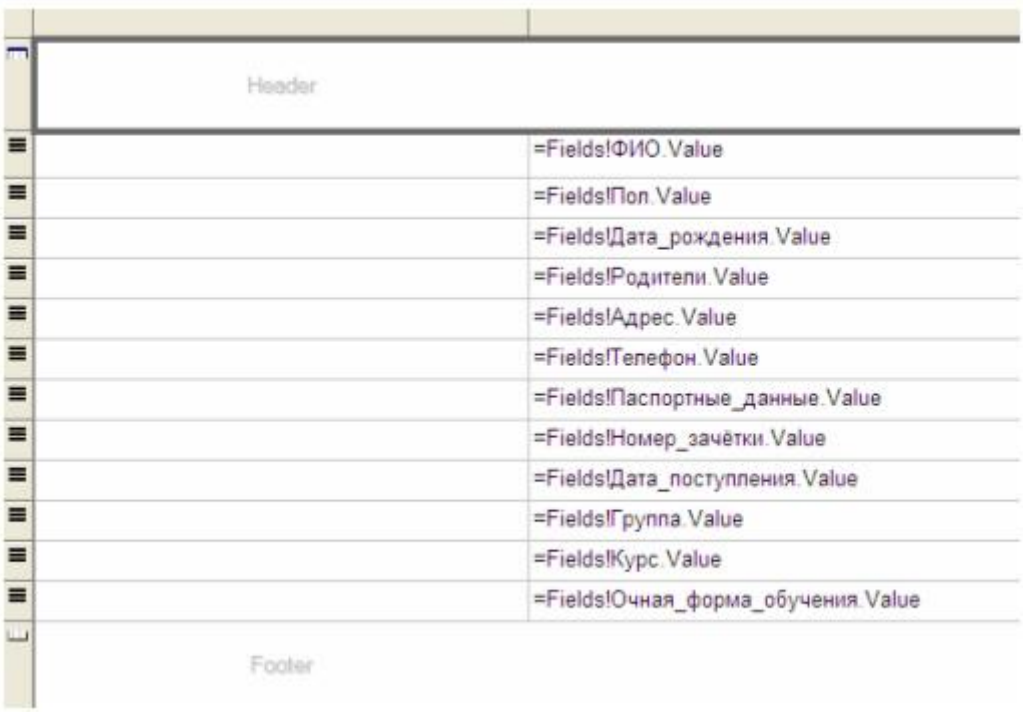


Рис. 24.11.

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).

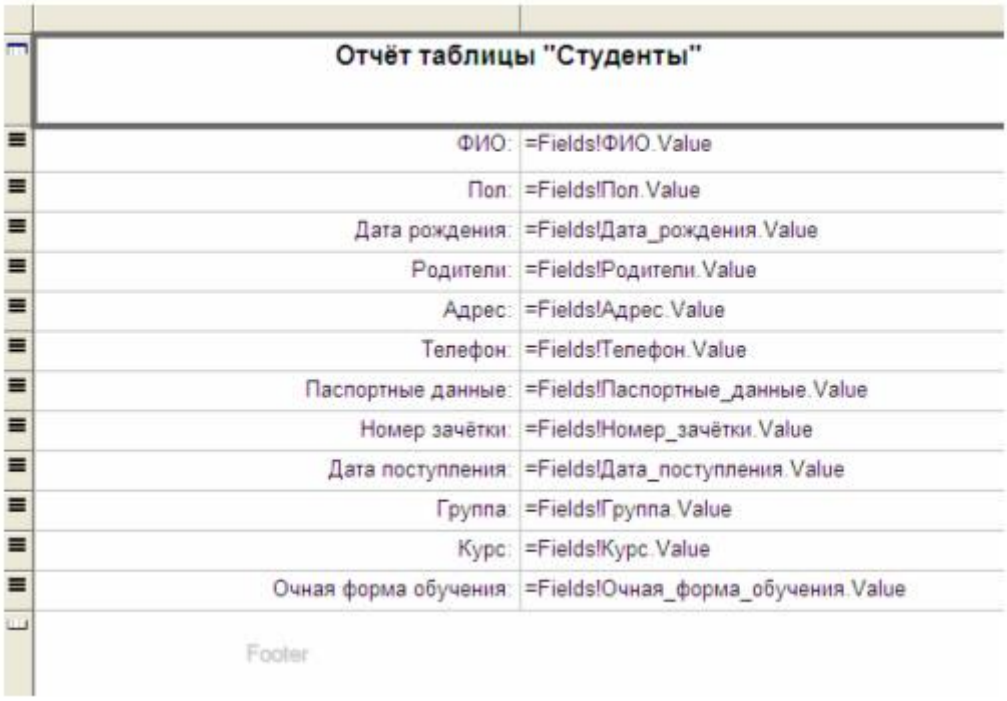


Рис. 24.12.

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).

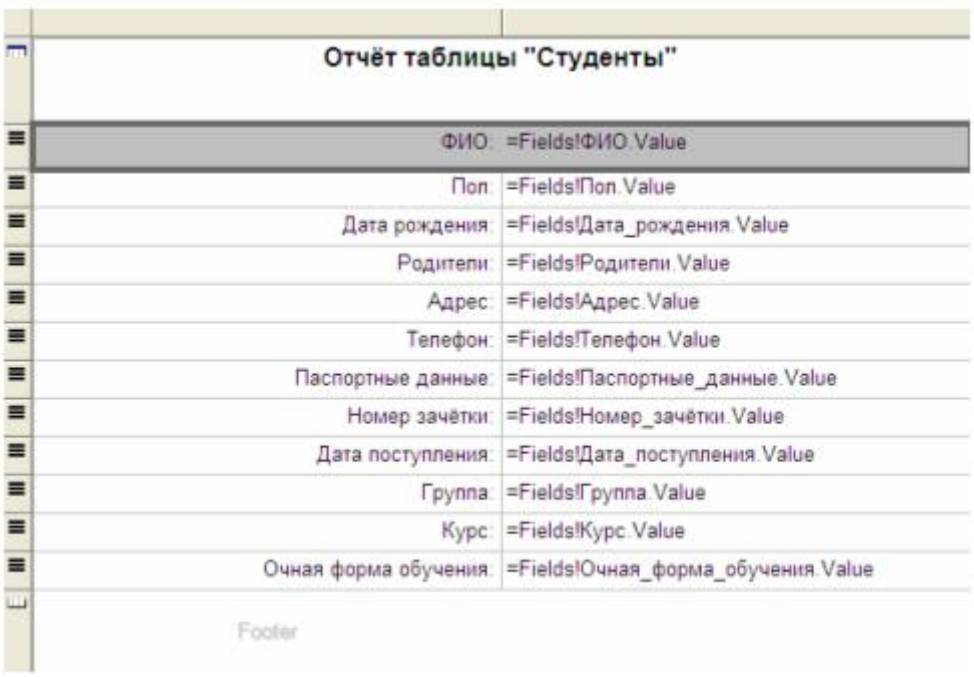


Рис. 24.13.

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).

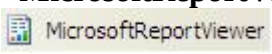
Замечание: Если кнопка

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):

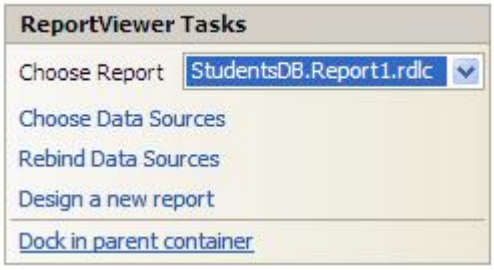
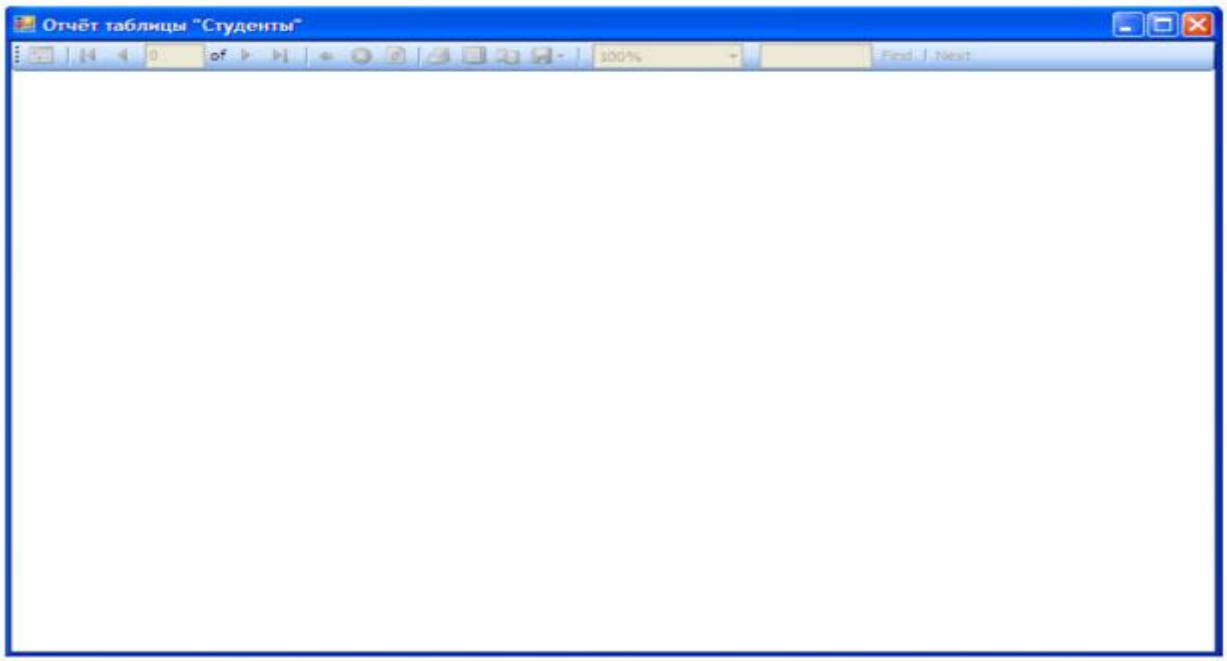


Рис. 24.15.

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).

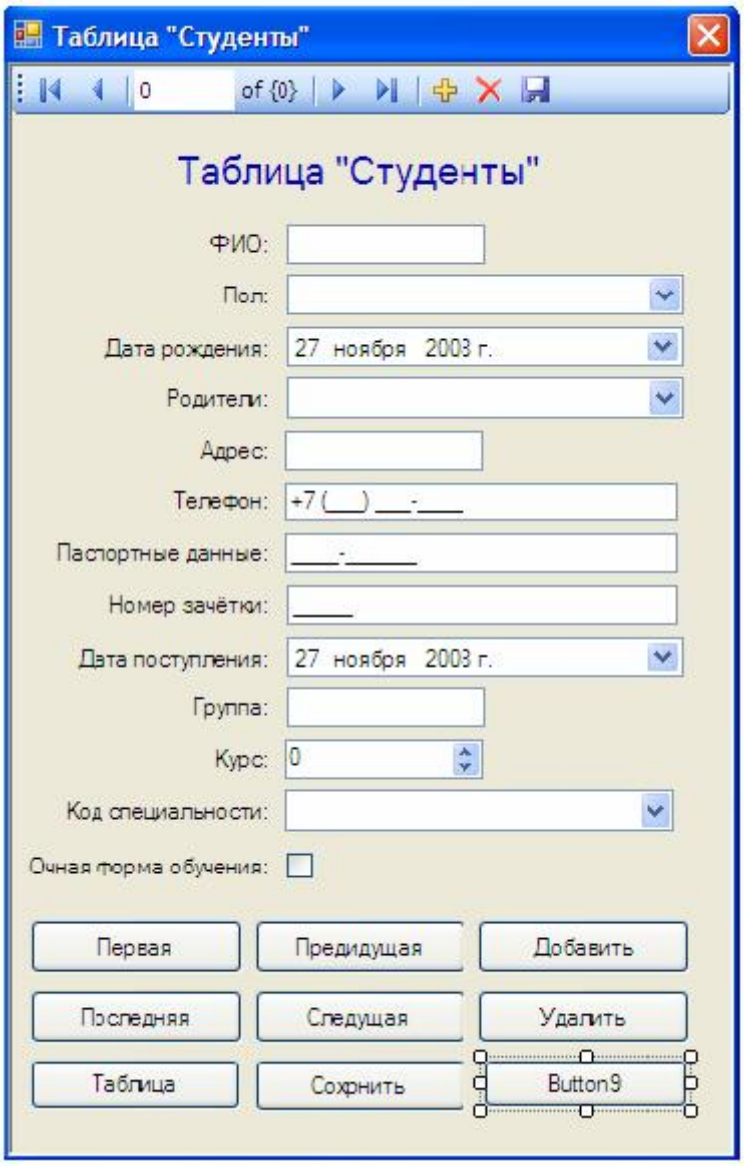


Рис. 24.18.

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

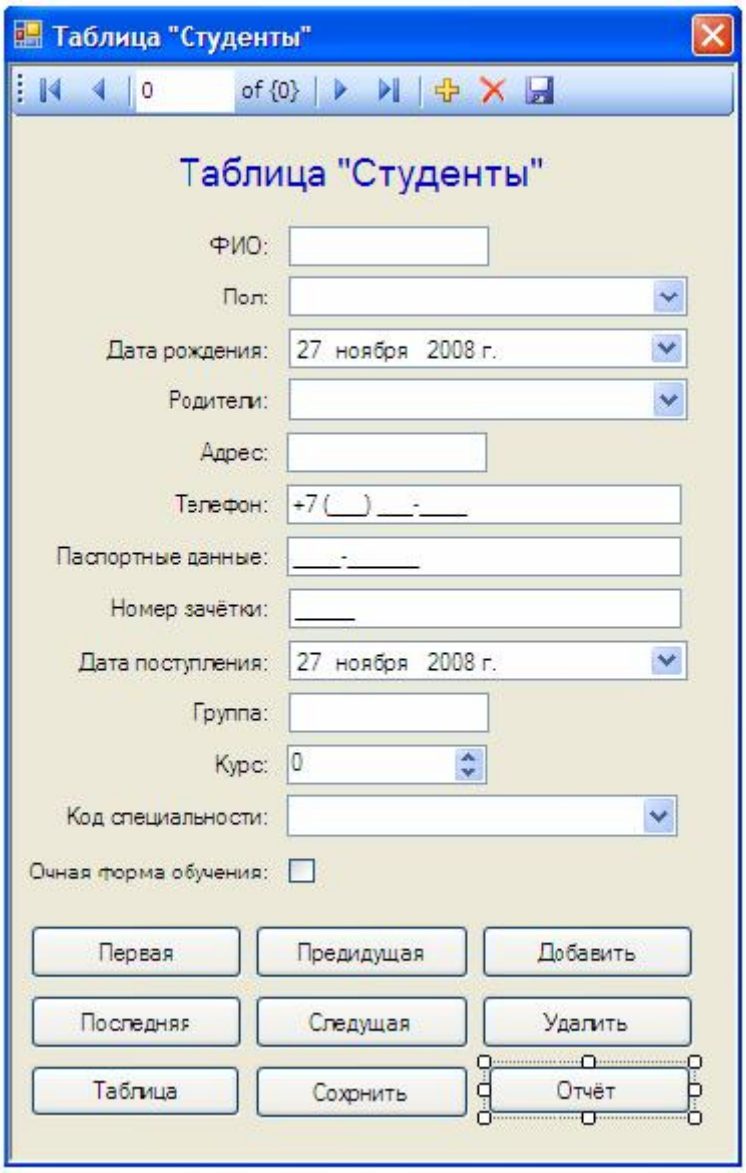


Рис. 24.19.

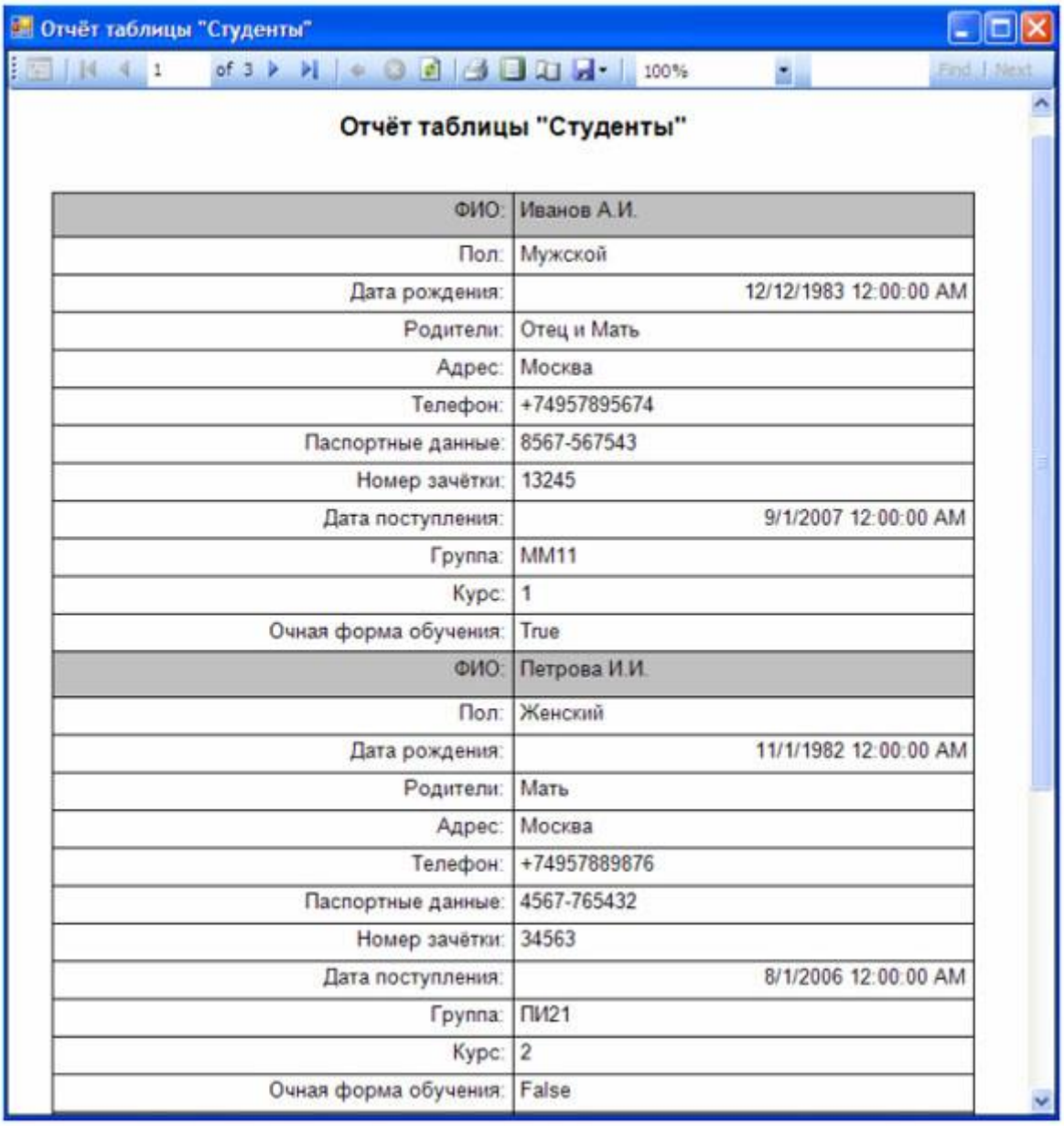
Теперь определим код обработчика события нажатия кнопки. Дважды щелкните **ЛКМ** по кнопке **"Отчет"** и в появившейся процедуре **"Button9_Click"** наберите команду "Form7.Show()" (рис. 24.20).

```
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"**.

Задание.

В программе Microsoft Visual Studio для работы с базой данных по учету успеваемости студентов, созданной в лабораторной работе № 1.

Создать отчет «Студенты».

Контрольные вопросы

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

Работа с литературой:

Рекомендуемые источники информации (№ источника)

Основная	Дополнительная	Методическая	Интернет-ресурсы
1-2	1-2	1-2	1-3

Оценочные средства: устный отчет к лабораторной работе (См.: Фонд оценочных средств)

Лабораторная работа № 4. «Подготовка документации IT проекта.»

Форма проведения лабораторная работа (3 часа)

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.

Работа с литературой:

Рекомендуемые источники информации (№ источника)			
Основная	Дополнительная	Методическая	Интернет-ресурсы
1-2	1-2	1-2	1-3

Оценочные средства: устный отчет к лабораторной работе (См.: Фонд оценочных средств)

Лабораторная работа № 5. «Расчет экономической эффективности проекта.»
Форма проведения: лабораторная работа (1,5 часа).

Цель работы:

Научиться проводить стоимостный анализ.

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*.

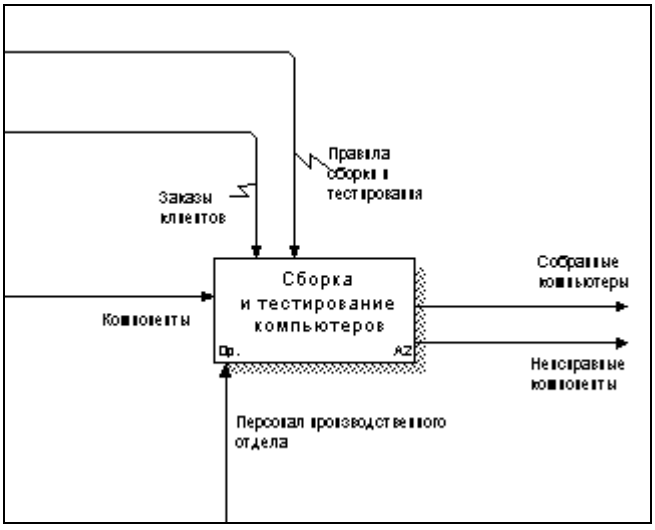


Рисунок 1 - Иллюстрация терминов ABC

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).

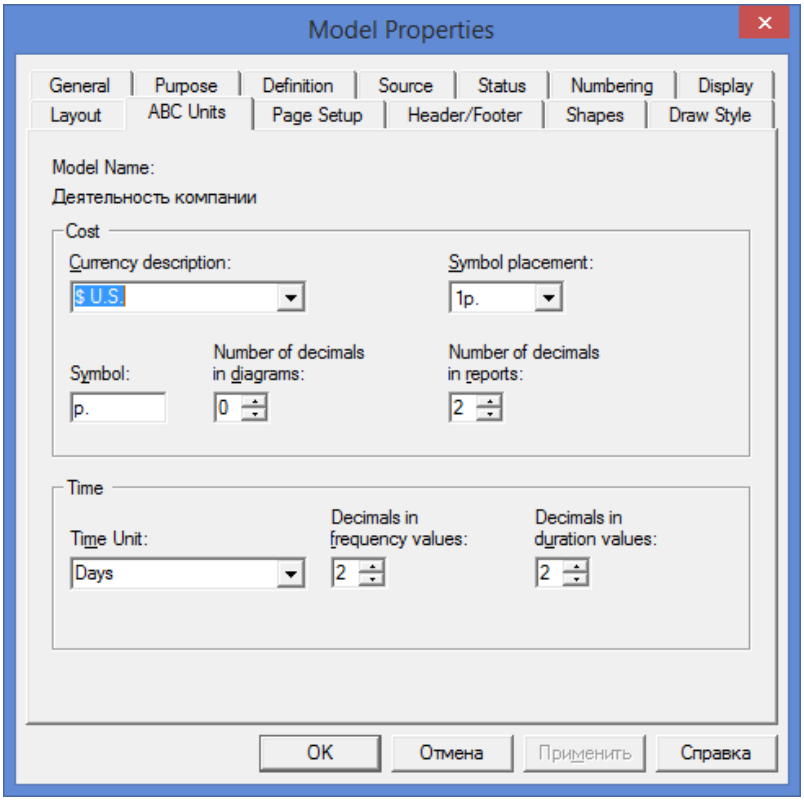


Рисунок 2 - Настройка единиц измерения валюты и времени

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).

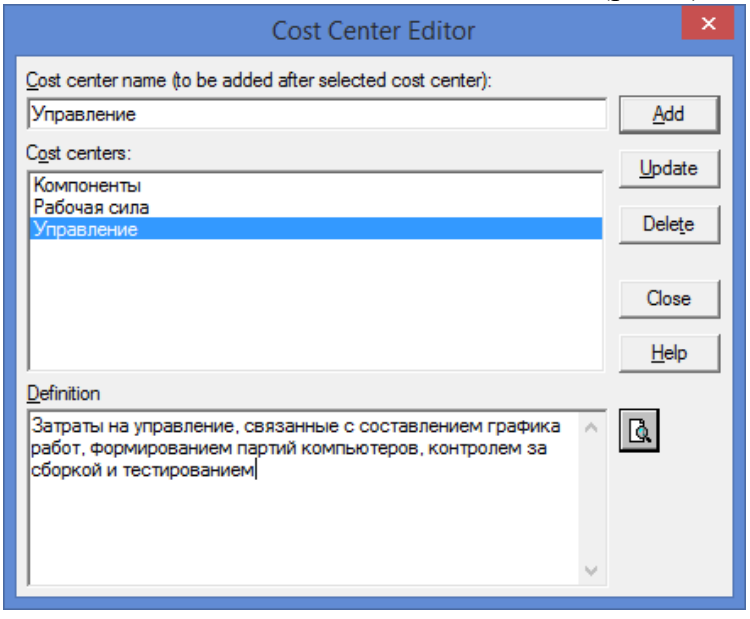


Рисунок 3 - Диалог Cost Center Editor

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.

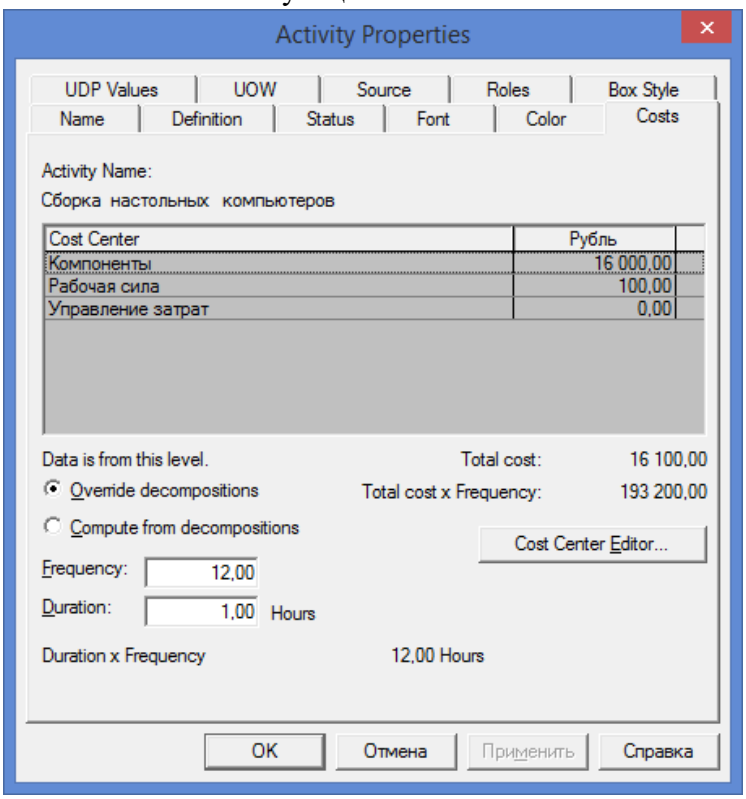


Рисунок 4 - Задание стоимости работ в диалоге Activity Properties/Cost

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).

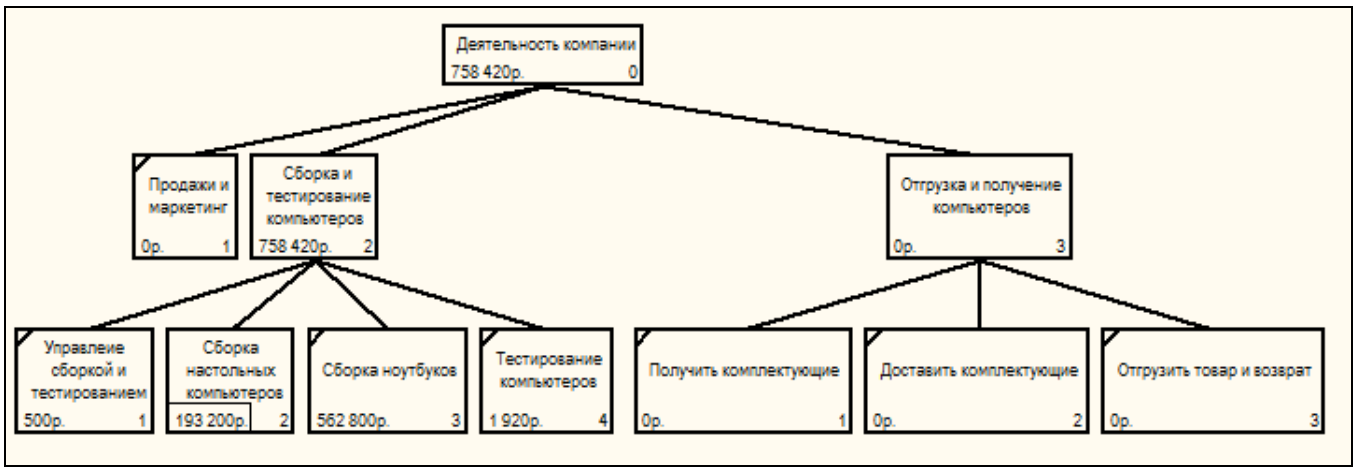


Рисунок 5 - Вычисление затрат родительской работы

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.

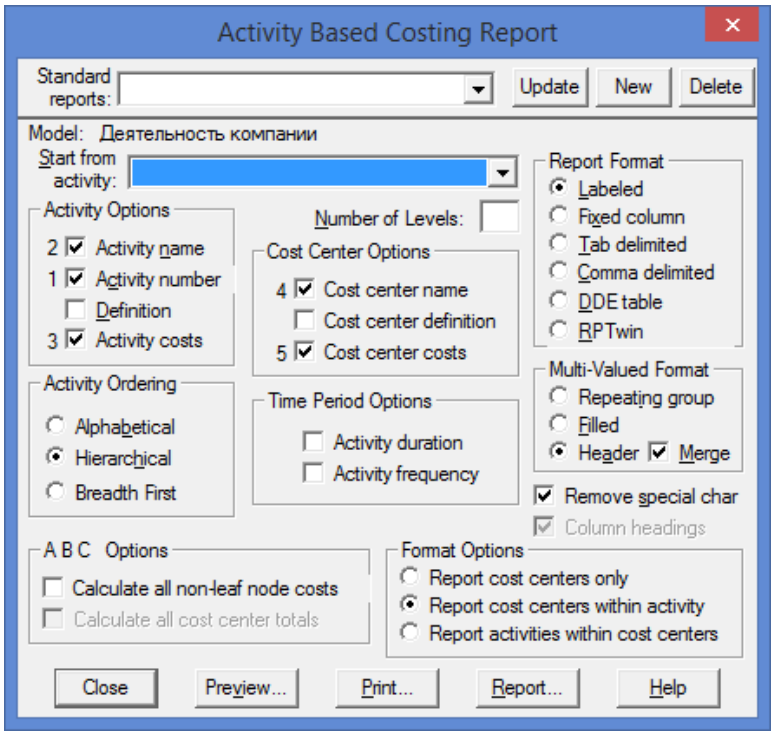


Рисунок 6 - Диалог настройки отчета по стоимости работ

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. ABC-анализ.

Работа с литературой:

Рекомендуемые источники информации (№ источника)			
Основная	Дополнительная	Методическая	Интернет-ресурсы
1-2	1-2	1-2	1-3

Оценочные средства: устный отчет к лабораторной работе (См.: Фонд оценочных средств)

8. КРИТЕРИИ ОЦЕНИВАНИЯ КОМПЕТЕНЦИЙ

Оценка «отлично» выставляется студенту, если он продемонстрировал глубокие, исчерпывающие знания и творческие способности в понимании, изложении и использовании учебно-программного материала; логически последовательные, содержательные, полные, правильные и конкретные ответы на все поставленные вопросы и дополнительные вопросы преподавателя; свободное владение основной и дополнительной литературой, рекомендованной учебной программой.

Оценка «хорошо» выставляется студенту, если он продемонстрировал твердые и достаточно полные знания всего программного материала, правильное понимание сущности и взаимосвязи рассматриваемых процессов и явлений; последовательные, правильные, конкретные ответы на поставленные вопросы при свободном устранении

замечаний по отдельным вопросам; достаточное владение литературой, рекомендованной учебной программой.

Оценка «удовлетворительно» выставляется студенту, если он продемонстрировал твердые знания и понимание основного программного материала; правильные, без грубых ошибок ответы на поставленные вопросы при устранении неточностей и несущественных ошибок в освещении отдельных положений при наводящих вопросах преподавателя; недостаточное владение литературой, рекомендованной учебной программой.

Оценка «неудовлетворительно» выставляется студенту, если он продемонстрировал неправильные ответы на основные вопросы, допущены грубые ошибки в ответах, непонимание сущности излагаемых вопросов; неуверенные и неточные ответы на дополнительные вопросы.

9. МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ, ОПРЕДЕЛЯЮЩИЕ ПРОЦЕДУРЫ ОЦЕНИВАНИЯ ЗНАНИЙ, УМЕНИЙ, НАВЫКОВ И (ИЛИ) ОПЫТА ДЕЯТЕЛЬНОСТИ, ХАРАКТЕРИЗУЮЩИХ ЭТАПЫ ФОРМИРОВАНИЯ КОМПЕТЕНЦИЙ

Текущая аттестация студентов проводится преподавателями, ведущими лабораторные занятия по дисциплине, в следующей форме: отчет письменный по заданию преподавателя.

Допуск к лабораторным работам происходит при наличии у студентов печатного варианта отчета. Защита отчета проходит в форме доклада студента по выполненной работе и ответов на вопросы преподавателя.

Отчет включает в себя следующие разделы: титульный лист с названием работы; цель работы; краткие теоретические сведения; описание результатов лабораторной работы (скриншоты); вывод из работы, включающий в себя описание проделанной работы.

Оценку «отлично» студент получает, если оформление отчета соответсвует установленным требованиям, правильно отвечает на предложенные преподавателем контрольные вопросы, правильно отвечает на дополнительные вопросы по теме лабораторной работы.

Оценку «хорошо» студент получает, если оформление отчета соответсвует установленным требованиям, правильно отвечает на предложенные преподавателем контрольные вопросы.

Оценку «удовлетворительно» студент получает без беседы с преподавателем, если оформление отчета соответсвует установленным требованиям.

Отчет может быть отправлен на доработку в следующих случаях:

- польностью не соовествует установленным требованиям;
- не раскрыта суть работы.

10. УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ

10.1. Рекомендуемая литература

10.1.1. Основная литература

1. Проектирование информационных систем. Проектный практикум / А.В. Платёнкин, И.П. Рак, А.В. Терехов, В.Н. Чернышов ; Министерство образования и науки Российской Федерации, Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Тамбовский государственный технический университет». – Тамбов : Издательство ФГБОУ ВПО «ТГТУ», 2015. – 81 с. : ил., схем. – Режим доступа: по подписке. – URL: http://biblioclub.ru/index.php?page=book&id=444966 (дата обращения: 21.10.2019). – Библиогр. в кн. – ISBN 978-5-8265-1409-2. – Текст : электронный.

2. Бурков, А.В. Проектирование информационных систем в Microsoft SQL Server 2008 и Visual Studio 2008 / А.В. Бурков. – Москва : Интернет-Университет Информационных Технологий, 2010. – 273 с. – Режим доступа: по подписке. – URL: http://biblioclub.ru/index.php?page=book&id=233750 (дата обращения: 21.10.2019). – Текст : электронный.

3. Лазицкас, Е.А. Базы данных и системы управления базами данных : [12+] / Е.А. Лазицкас, И.Н. Загумённикова, П.Г. Гилевский. – Минск : РИПО, 2016. – 267 с. : ил. – Режим доступа: по подписке. – URL: http://biblioclub.ru/index.php?page=book&id=463305 (дата обращения: 21.10.2019). – Библиогр. в кн. – ISBN 978-985-503-558-0. – Текст : электронный.

10.1.2. Дополнительная литература:

1. Соболева, М.Л. Информационные системы. Лабораторный практикум / М.Л. Соболева, А.С. Алфимова. – Москва : Прометей, 2011. – 88 с. – Режим доступа: по подписке. – URL: http://biblioclub.ru/index.php?page=book&id=212836 (дата обращения: 21.10.2019). – Библиогр. в кн. – ISBN 978-5-4263-0025-5. – Текст : электронный.

2. Щелоков, С.А. Разработка и создание баз данных средствами СУБД Access и SQL Server / С.А. Щелоков ; Министерство образования и науки Российской Федерации, Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Оренбургский государственный университет», Кафедра программного обеспечения вычислительной техники и автоматизированных систем. – Оренбург : Оренбургский государственный университет, 2014. – 109 с. – Режим доступа: по подписке. – URL: http://biblioclub.ru/index.php?page=book&id=260754 (дата обращения: 21.10.2019). – Текст : электронный.

3. Сенченко, П.В. Организация баз данных / П.В. Сенченко ; Министерство образования и науки Российской Федерации, Томский Государственный Университет Систем Управления и Радиоэлектроники (ТУСУР), Факультет дистанционного обучения. – Томск : ТУСУР, 2015. – 170 с. : схем., табл., ил. – Режим доступа: по подписке. – URL: http://biblioclub.ru/index.php?page=book&id=480906 (дата обращения: 21.10.2019). – Библиогр.: с. 163-164. – Текст : электронный.

10.1.3. Методическая литература:

1. Методические указания по выполнению лабораторных работ по дисциплине «Проектный практикум».

2. Методические рекомендации для студентов по организации самостоятельной работы по дисциплине «Проектный практикум».

10.1.4. Интернет-ресурсы:

1. http://www.intuit.ru – сайт дистанционного образования в области информационных технологий

2. http://window.edu.ru – образовательные ресурсы ведущих вузов

3. http:// www.informika.ru – сервер Министерства образования РФ и ГосНИИ Информационных технологий и телекоммуникаций. На сервере представлена разнообразная информация по всем аспектам образования (нормативная и законодательная база, обучающие ресурсы, информационные технологии).