Федеральное государственное образовательное бюджетное
учреждение высшего образования
**«Финансовый университет при Правительстве Российской Федерации»**
**(Финуниверситет)**

**Самарский финансово-экономический колледж**
**(Самарский филиал Финуниверситета)**

УТВЕРЖДАЮ
Заместитель директора по учебно-методической работе
_____Л.А Косенкова
«_21_» _февраля_____20 _22_г.

# МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ОРГАНИЗАЦИИ И ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ ЗАНЯТИЙ ПО ПРОФЕССИОНАЛЬНОМУ МОДУЛЮ «ПМ.11 РАЗРАБОТКА, АДМИСТРИРОВАНИЕ И ЗАЩИТА БАЗ ДАННЫХ»

**СПЕЦИАЛЬНОСТЬ: 09.02.07 ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ПРОГРАММИРОВАНИЕ**

Самара – 2022

Методические указания по организации и выполнению практических занятий разработаны на основе рабочей программы по профессиональному модулю «Разработка, администрирование и защита баз данных», в соответствии с федеральным государственным образовательным стандартом среднего профессионального образования по специальности 38.02.06 Финансы, утвержденного приказом Министерства образования науки Российской Федерации от 09.12.2016 года № 1547, с учетом Профессионального стандарта, утвержденного приказом Министерства труда и социальной защиты Российской Федерации от 11 февраля 2014 г. № 647н «Об утверждении профессионального стандарта 06.011 Администратор баз данных» (зарегистрирован Министерством юстиции Российской Федерации 24 ноября 2014 г., регистрационный № 34846)
Присваиваемая квалификация: администратор баз данных

Разработчики:

Платковская Е.А. _____ Преподаватель Самарского филиала Финуниверситета

Методические указания по организации и выполнению практических занятий рассмотрены и рекомендованы к утверждению на заседании предметной (цикловой) комиссии естественно-математических дисциплин

Протокол от «24» ноября 20 22 г. № 5

Председатель ПЦК _____ М.В. Писцова

## ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Методические указания по выполнению практических работ по профессиональному модулю ПМ.11 Разработка, администрирование и защита баз данных разработаны с целью оказания помощи студентам специальности 09.02.07 Информационные системы и программирование и преподавателям по организации практических занятий по изучаемой дисциплине, в соответствии с требованиями федерального государственного стандарта среднего профессионального образования.

Методические разработка включает в себя краткие теоретические сведения, указания по выполнению практических работ, контрольные вопросы, формы контроля.

В соответствии с учебным планом на практические работы для студентов отводится 146 часов.

В результате изучения профессионального модуля студент должен освоить основной вид деятельности Осуществление интеграции программных модулей и соответствующие ему общие компетенции и профессиональные компетенции:

**Перечень общих компетенций**

| **Код** | **Наименование общих компетенций** |
|---|---|
| ОК 01 | Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам |
| ОК 02 | Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности. |
| ОК 03 | Планировать и реализовывать собственное профессиональное и личностное развитие. |
| ОК 04 | Работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами. |
| ОК 05 | Осуществлять устную и письменную коммуникацию на государственном языке с учетом особенностей социального и культурного контекста. |
| ОК 06 | Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных общечеловеческих ценностей |
| ОК 07 | Содействовать сохранению окружающей среды, ресурсосбережению, эффективно действовать в чрезвычайных ситуациях. |
| ОК 08 | Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания необходимого уровня физической подготовленности |
| ОК 09 | Использовать информационные технологии в профессиональной деятельности. |
| ОК 10 | Пользоваться профессиональной документацией на государственном и иностранном языке |
| ОК 11 | Планировать предпринимательскую деятельность в профессиональной сфере |

***Перечень профессиональных компетенций***

| **Код** | **Наименование видов деятельности и профессиональных компетенций** |
|---|---|
| **ВД 11** | **Разработка, администрирование и защита баз данных** |
| ПК 11.1 | Осуществлять сбор, обработку и анализ информации для проектирования баз данных |
| ПК 11.2 | Проектировать базу данных на основе анализа предметной области |
| ПК 11.3 | Разрабатывать объекты базы данных в соответствии с результатами анализа предметной области |

| | |
|---|---|
| ПК 11.4 | Реализовывать базу данных в конкретной системе управления базами данных |
| ПК 11.5 | Администрировать базы данных |
| ПК 11.6 | Защищать информацию в базе данных с использованием технологии защиты информации |

1.1.3. В результате освоения профессионального модуля студент должен:

| | |
|---|---|
| **Иметь практический опыт** | В работе с объектами базы данных в конкретной системе управления базами данных;<br>использовании стандартных методов защиты объектов базы данных; работе с документами отраслевой направленности |
| **Уметь** | – работать с современными case-средствами проектирования баз данных; проектировать логическую и физическую схемы базы данных;<br>– создавать хранимые процедуры и триггеры на базах данных;<br>– применять стандартные методы для защиты объектов базы данных;<br>– выполнять стандартные процедуры резервного копирования и мониторинга выполнения этой процедуры;<br>– выполнять процедуру восстановления базы данных и вести мониторинг выполнения этой процедуры;<br>– обеспечивать информационную безопасность на уровне базы данных. |
| **Знать** | – основные положения теории баз данных, хранилищ данных, баз знаний;<br>– основные принципы структуризации и нормализации базы данных;<br>– основные принципы построения концептуальной, логической и физической модели данных;<br>– методы описания схем баз данных в современных системах управления базами данных;<br>– структуры данных систем управления базами данных, общий подход к организации представлений, таблиц, индексов и кластеров; |

**Количество практических работ по разделам**

| | |
|---|---|
| Раздел 1. Разработка, администрирование и защита баз данных | 146 |

**Перечень практических занятий**

| **№п.п.** | **НАЗВАНИЕ ПРАКТИЧЕСКИХ ЗАНЯТИЙ** |
|---|---|
| **1** | Практическая работа № 1 «Сбор и анализ информации» |
| **2** | Практическая работа № 2 «Проектирование реляционной схемы базы данных в среде СУБД» |
| **3** | Лабораторная работа № 1 «Приведение БД к нормальной форме 3НФ» |
| 4 | Лабораторная работа 2 Создание базы данных в среде разработки |
| **5** | Лабораторная работа 3 Создание и использование фильтров |
| **6** | Лабораторная работа 4 Создание многотабличной базы данных. Установление взаимосвязей между таблицами |
| **7** | Лабораторная работа 5 Создание экранной формы |
| **8** | Лабораторная работа 6 Создание элементов управления на форме. |
| **9** | Лабораторная работа 7 Создание главной кнопочной формы |

| | |
|---|---|
| **10** | Лабораторная работа 8 Создание отчета |
| **11** | Лабораторная работа 9 Создание подчиненного отчета. Вычисления в отчетах |
| **12** | Лабораторная работа 10 Создание и управление базой данных с помощью SQL – операторов |
| **13** | Лабораторная работа 11 Построение запросов вычисления и подведения итогов к учебной базе данных |
| **14** | Лабораторная работа 12 Организация локальной сети. Настройка локальной сети |
| **15** | Лабораторная работа 13 Установка и настройка SQL-сервера |
| **16** | Лабораторная работа 14 Создание пользовательских баз данных |
| **17** | Лабораторная работа 15 Создание ограничений |
| **18** | Лабораторная работа 16 Использование диаграмм баз данных |
| **19** | Лабораторная работа 17 Экспорт и импорт данных базы в документы пользователя |
| **20** | Лабораторная работа 18 «Выполнение резервного копирования» |
| **21** | Лабораторная работа 19 «Восстановление базы данных из резервной копии» |
| **22** | Лабораторная работа 20 «Реализация доступа пользователей к базе данных» |
| **23** | Лабораторная работа 21 «Мониторинг безопасности работы с базами данных» |
| **24** | Лабораторная работа 22 «Установка приоритетов» |
| **25** | Лабораторная работа 23 «Развертывание контроллеров домена» |
| **26** | Лабораторная работа 24 «Мониторинг сетевого трафика» |

## МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ СТУДЕНТОВ ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ ЗАНЯТИЙ

### Практическая работа №1 Сбор и анализ информации (6 часов)

**Цель практической работы:**

Получить теоретические знания и практические навыки реализации баз данных (БД). Осуществить анализ предметной области. Освоить концептуальное проектирование и научиться определять сущности и атрибуты БД. Научиться разрабатывать инфологическую модель БД в виде ER-диаграмм. Получить теоретические знания и практические навыки при физическом проектировании баз данных (БД). Научиться создавать даталогическую модель БД.

**Теоретические сведения:**

**Понятие БД и СУБД**

**Информационная система** - система, реализующая автоматизированный сбор, обработку и манипулирование данными и включающая технические средства обработки данных, программное обеспечение и соответствующий персонал.

Цель любой информационной системы - обработка данных об объектах реального мира. Основой информационной системы является **база данных**. В широком смысле слова **база данных** - это совокупность сведений о конкретных объектах реального мира вкакой-либо предметной области.

Под **предметной областью** принято понимать часть реального мира, подлежащего изучению для организации управления и в конечном счете автоматизации, например, предприятие, вуз и т.д.

Создавая базу данных, пользователь стремится упорядочить информацию по различным признакам и быстро производить выборку с произвольным сочетанием признаков. Большое значение при этом приобретает структурирование данных.

**Структурирование данных** - это введение соглашений о способах представления данных.

***Неструктурированными*** называют данные, записанные, например, в текстовом файле.

Ниже приведен пример неструктурированных и структурированных данных, содержащих сведения о студентах (номер личного дела, фамилию, имя, отчество и год рождения).

**Неструктурированные данные:**

| |
|---|
| Личное дело № 16493. Сергеев Петр Михайлович, дата рождения 1 января 1976 г.; Л/д № 16593, Петрова Анна Владимировна, дата рожд. 15 марта 1975 г.; № личн.дела 16693, д.р. 14.04.76, Анохин Андрей Борисович |

Легко убедиться, что сложно организовать поиск необходимых данных, хранящихся в неструктурированном виде.

**Структурированные данные:**

| № личного | Фамилия | Имя | Отчество | Дата рождения |
|---|---|---|---|---|
| 16493 | Сергеев | Петр | Михайлович | 01.01.76 |
| 16593 | Петрова | Анна | Владимировна | 15.03.75 |
| 16693 | Анохин | Андрей | Борисович | 14.04.76 |

В современной технологии баз данных предполагается, что создание базы данных, ее поддержка и обеспечение доступа пользователей к ней осуществляются централизованно с помощью специального программного инструментария - **системыуправлении базами данных (СУБД).**

**База данных (БД) - это поименованная совокупность данных, отражающая состояние объектов и их отношений в рассматриваемой предметной области.**

**Объектом** называется элемент предметной области, информацию о котором мы сохраняем. Объект может быть **реальным** (например, человек, изделие; или населенный пункт) и

**абстрактным** (например, событие, счет покупателя или изучаемый студентами курс).

Так, в области продажи автомобилей примерами **объектов** могут служить **МОДЕЛЬ АВТОМОБИЛЯ**, **КЛИЕНТ** и **СЧЕТ**. На товарном складе - это **ПОСТАВЩИК**, **ТОВАР, ОТПРАВЛЕНИЕ** и т. д.

Понятие базы данных тесно связано с такими понятиями структурных элементов, как **поле, запись, файл (таблица)** (рис.1).

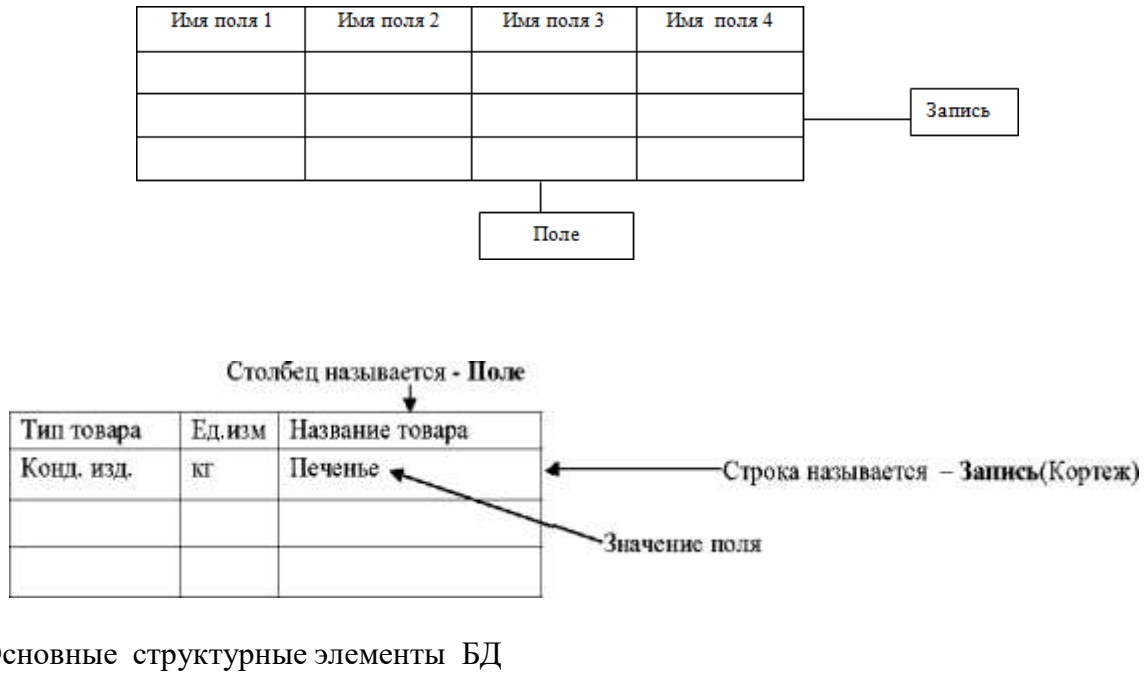



Рис.1 Основные структурные элементы БД

**Структурные элементы базы данных**

**Поле** - элементарная единица логической организации данных, которая соответствует неделимой единице информации - реквизиту. Для описания поля используются следующие характеристики:

- ***имя***, например, **Фамилия, Имя, Отчество, Дата рождения**;
- ***тип***, например, символьный, числовой, денежный;
- ***длина***, например, 15 байт, причем будет определяться максимально возможным количеством символов;
- ***точность*** для числовых данных, например два десятичных знака для отображения дробной части числа,

**Запись** - совокупность логически связанных полей.

**Экземпляр записи** - отдельная реализация записи, содержащая конкретные значения ее полей.

**Файл (таблица)** - совокупность экземпляров записей одной структуры.

Описание логической структурызаписи файла содержит последовательность расположения полей записи и их основные характеристики, как это показано на рис. 2 и3.

| Имя файла | | | | | |
|---|---|---|---|---|---|
| Поле | | Iризнак ключа | Формат поля | | |
| Имя (обозначение) | Полное наименование | | Тип | Длина | ность (для чисел) |
| имя 1 | | | | | |
| | | | | | |
| имя N | | | | | |

Рис. 2. Описание логической структуры записи файла

В структуре записи файла указываются поля, значения которых являются ключами:
**первичными (ПК) и внешними (ВК),**

**Первичный ключ** (ПК) - это одно или несколько полей, однозначно идентифицирующих запись. Если первичный ключ состоит из одного поля, он называется **простым**, если из нескольких полей - **составным** ключом.

**Внешний ключ** (ВК) - это одно или несколько полей, которые выполняют роль поисковых или группировочных признаков. В отличие от первичного, значение внешнегоключа может повторяться в нескольких записях файла, то есть он не является уникальным. Если по значению первичного ключа может быть найден один единственный экземпляр записи, то по внешнему - несколько.

| Имя файла: СТУДЕНТ | | | | | |
|---|---|---|---|---|---|
| Поле | | Iризнак ключа | Формат поля | | |
| Обозначение | Наименование | | Тип | Длина | Точность |
| Номер | № личного дела | • | Симв | 5 | |
| Фамилия | Фамилия студента | | Симв | 15 | |
| Имя | Имя студента | | Симв | 10 | |
| Отчество | Отчество студента | | Симв | 15 | |
| Дата | Дата рождения | | Дата | 8 | |

Рис. 3. Описание логической структуры записи файла **СТУДЕНТ**

**Понятие модели данных**

Для того, чтобы спроектировать структуру базы данных, необходима исходная информация о предметной области. Желательно, чтобы эта информация была представлена в формализованном виде.

Такое формализованное описание предметной области будем называть **инфологической** (infological) **моделью предметной области** (ИЛМ) или **концептуальной моделью** (КМ).

Ядром любой базы данных является модель данных. **Модель данных** представляет собой множество структур данных, ограничений целостности и операций манипулирования данными. С помощью модели данных могут быть представлены объекты предметной области и взаимосвязи между ними.

**Модель данных** - совокупность структур данных и операций их обработки.

СУБД основывается на использовании **иерархической, сетевой или реляционной модели**, на комбинации этих моделей или на некотором их подмножестве.

Самой распространенной моделью данных является - **реляционная**.

Иерархическая модель данных

**Иерархическая модель** организует данные в виде древовидной структуры

К основным понятиям иерархической структуры относятся: **уровень, элемент (узел),**

**связь**. Дерево представляет собой иерархию элементов, называемых узлами. **Узел** - это совокупность атрибутов данных, описывающих некоторый объект. На самом верхнем уровне иерархии имеется один и только один узел - **корень**. Каждый узел, кроме корня, связан с одним узлом на более высоком уровне, называемым **исходным** для данного узла. Ни один элемент не имеет более одного исходного. Каждый элемент может быть связан с одним или несколькими элементами на более низком уровне. Они называются **порожденными** (рис. 4).

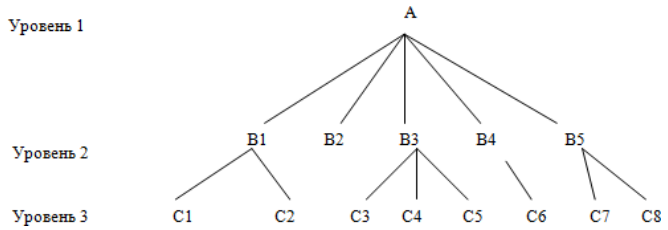



Рис. 4. Графическое изображение иерархической структуры БД

К каждой записи базы данных существует только один (иерархический) путь от корневой записи. Например, как видно из рис. 4, для записи С4 путь проходит через записи А и В3.

Сетевая модель данных

**Сетевая модель** организует данные в виде сетевой структуры.

Структура называется сетевой, если в отношениях между данными порожденныйэлемент имеет более одного исходного.

В сетевой структуре при тех же основных понятиях (уровень, узел, связь) каждыйэлемент может быть связан с любым другим элементом.

На рис. 5 изображена сетевая структура базы данных в виде графа.

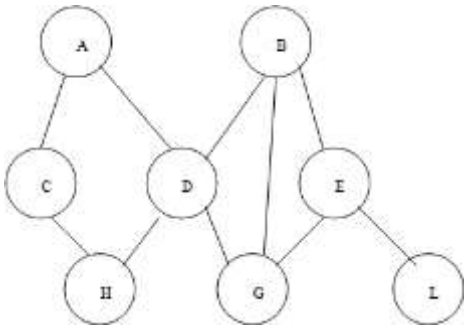



Рис. 5. Графическое изображение сетевой структуры

Реляционная модель данных

**Реляционная модель** данных является совокупностью взаимосвязанных двумерных таблиц объектов модели.

Например, реляционной таблицей можно представить информацию о студентах,обучающихся в вузе (рис. 6).

| № личного дела | Фамилия | Имя | Отчество | Дата рождения | Группа |
|---|---|---|---|---|---|
| 16493 | Сергеев | Петр | Михайлович | 01.01.76 | 111 |
| 16593 | Петрова | Анна | Владимировна | 15.03.75 | 112 |
| 16693 | Анохин | Андрей | Борисович | 14.04.76 | 111 |

Рис. 6. Пример реляционной таблицы

Связи между двумя логически связанными таблицами в реляционной моделиустанавливаются по равенству значений одинаковых атрибутов этих таблиц.

Каждая реляционная таблица представляет собой двумерный массив и обладает следующими

*свойствами:*

- - каждый элемент таблицы - один элемент данных;
- - все столбцы в таблице однородные, т.е. все элементы в столбце имеют одинаковый тип (числовой, символьный и т.д.) и длину;
- - каждый столбец имеет уникальное имя;
- - одинаковые строки в таблице отсутствуют;
- - порядок следования строк и столбцов может быть произвольным.

При описании реляционной модели часто используют следующие термины: **отношение, кортеж, домен.**

**Отношения** представлены в виде таблиц, строки которых соответствуют **записям (кортежам)**, а столбцы полям, атрибутам отношений (**доменам**).

**Поле, каждое значение которого однозначно определяет соответствующую запись, называется простым ключом (ключевым полем)**. Если записи однозначно определяются значениями нескольких полей, то такая таблица базы данных имеет **составной ключ.**
В примере, показанном на рис.6, ключевым полем таблицы является "№ личного дела".

Между двумя реляционными таблицами могут быть сформированы **связи**.
Различные таблицы, могут быть **связаны между собой через общее поле данных**.
На рис. 7 показан пример реляционной модели, построенной на основе отношений: **СТУДЕНТ, СЕССИЯ, СТИПЕНДИЯ**.

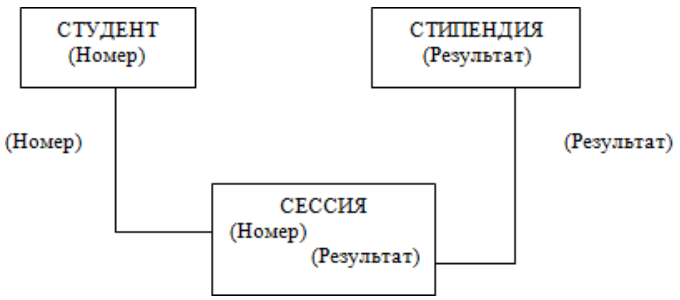



Рис.7. Пример реляционной модели

Таблица **СТУДЕНТ** имеет поля: Номер, Фамилия, Имя, Отчество, Дата рождения,Группа;
**СЕССИЯ** - Номер, Оценка 1, Оценка 2, Оценка 3, Оценка 4, Результат;
**СТИПЕНДИЯ** - Результат, Процент

Таблицы **СТУДЕНТ** И **СЕССИЯ** имеют совпадающие ключи (**Номер**), что даетвозможность легко организовать связь между ними.

Таблица **СЕССИЯ** имеет **первичный ключ Номер** и содержит **внешний ключ Результат**, который обеспечивает ее связь с таблицей **СТИПЕНДИЯ**.

Благодаря имеющимся связям достигаются следующие преимущества:

1. Удается избежать дублирования информации. Все необходимые данные можно хранить только в одной таблице. Так, например, нет необходимости в таблице **СЕССИЯ** хранить номер группы каждого студента, сдающего экзамены, достаточно задать связь с таблицей **СТУДЕНТ**.
2. В реляционных базах данных легко производить изменения Если в таблице

**СЕССИЯ** изменить какие-нибудь значения, то правильная информация автоматически будет связана с другими таблицами, ссылающимися на первую (например, таблица **СТИПЕНДИЯ**).

3. В нереляционных базах данных сложно передать все имеющиеся зависимости,

т.е. связать друг с другом данные из различных таблиц Реляционная база данных выполняет все эти действия автоматически.

4. В реляционных базах данных удается легко избежать установления ошибочных связей между различными таблицами данных, а необходимый объем памяти сокращен до минимума.

# 1. ПРОЕКТИРОВАНИЕ БАЗ ДАННЫХ

## Анализ и описание предметной области БД

***Предметной областью*** называется фрагмент реальности, который описывается или моделируется с помощью БД и ее приложений. В предметной области выделяются информационные объекты – идентифицируемые объекты реального мира, процессы, системы, понятия и т.д., сведения о которых хранятся в БД.

**Анализ предметной области** выполняется в следующем порядке: название предприятия, цель деятельности предприятия, структура предприятия, информационные потребностей пользователей. Для описания ПО следует привести основные решаемые задачи БД. Дать словесное описание процесса функционирования ПО, проанализироватьосновные хозяйственные операции, которые совершаются в ПО. Привести анализ структуры предприятия, перечислить задачи, решаемые отдельными подразделениями.

**Анализ предметной области целесообразно разбить па три фазы:**

1. анализ концептуальных требований и информационных потребностей;
2. выявление информационных объектов и связей между ними;
3. построение концептуальной модели предметной области и проектирование концептуальной схемы БД.

### Анализ концептуальных требований и информационных потребностей

Требования пользователей к разрабатываемой БД представляют собой список запросов с указанием их интенсивности и объемов данных. Эти сведения разработчики БД получают в диалоге с ее будущими пользователями. Здесь же выясняются требованияк вводу, обновлению и корректировке информации. Требования пользователей уточняются и дополняются при анализе имеющихся и перспективных задач.

Рассмотрим примерный состав вопросника при анализе различных предметных областей.

***Пример***. Предлагается разработать БД для учета студентов вуза. Анализ предметной области:

1. Сколько студентов учится в вузе? 2. Сколько факультетов и отделений в вузе? 3. Как распределены студенты по факультетам отделений и курсам? 4. Сколько дисциплин читается на каждом курсе по каждой специальности? 5. Как частообновляется информация в БД? 6. Сколько преподавателей в вузе? 7. Сколько иногородних студентов живет в общежитии, на частных квартирах? 8. Сколько лекционных аудиторий и аудиторий для проведения практических занятий, лабораторий? 9. Какая преемственность существует между читаемыми курсами?

10. Как информация, представленная в п.п. 1-9, используется в настоящее время (расписание занятий, экзаменов, зачетов и т.д.) и как ее собираются использовать?

11. Сколько раз в день, сколько человек и кто пользуются БД?

### Выявление информационных объектов и связей между ними

Вторая фаза анализа предметной области состоит в выборе информационных объектов, задании необходимых свойств для каждого объекта, выявлении связей между объектами, определении ограничений, накладываемых на информационные объекты, типы связей между ними, характеристики информационных объектов- Проанализируем предметную область на примере БД "Видеомагнитофоны".

При выборе информационных объектов постараемся ответить на ряд вопросов:

1. На какие классы можно разбить данные, подлежащие хранению в БД?
2. Какое имя можно присвоить каждому классу данных?
3. Какие наиболее интересные характеристики (с точки зрения пользователя) каждого класса данных можно выделить?
4. Какие имена можно присвоить выбранным наборам характеристик?

***Пример.*** БД "Видеомагнитофоны", рассчитанной на пользователей, которые хотят приобрести данный вид техники.

После беседы с различными пользователями и просмотра каталогов было выяснено, что интерес представляют три информационных объекта: видеомагнитофон, видеоплеер, видеокассета. Рассмотрим наиболее существенные характеристики каждого информационного объекта.

**Объект** - ВИДЕОМАГНИТОФОН.

**Атрибуты** - страна-изготовитель, фирма-изготовитель, № модели, телевизионные системы, число кассетных гнезд, ресурс непрерывной работы, система автопоиска, напряжение в сети, наличие таймера, число программ, габаритные размеры, масса, цена вдолларах, год выпуска.

**Объект** - ВИДЕОПЛЕЙЕР,

**Атрибуты** - страна-изготовитель, фирма-изготовитель, № модели, телевизионные системы, число воспроизводящих головок, ресурс непрерывной работы, напряжение в сети, наличие таймера, габаритные размеры, масса, цена в долларах, год выпуска.

**Объект - ВИДЕОКАССЕТА**.

**Атрибуты** - наименование, страна-изготовитель, фирма-изготовитель, тип кассеты, время проигрывания, цена в долларах.

Далее выделим связи между информационными объектами. В ходе этого процессапостараемся ответить на следующие вопросы:

1. Какие типы связей между информационными объектами?
2. Какое имя можно присвоить каждому типу связей?
3. Каковы возможные типы связей, которые могут быть использованы впоследствии?
4. Имеют ли смысл какие-нибудь комбинации типов связей?

Попытаемся задать ограничения на объекты и их характеристики.

Под **ограничением целостности** обычно понимают логические ограничения, накладываемые на данные. Ограничение целостности - это такое свойство, которое мы задаем для некоторого информационного объекта или его характеристики и которое должно сохраняться для каждого их состояния.

Введем следующие ограничения:

1. Значение атрибута "число кассетных гнезд" изменяется от 1 до 2.
2. Значение атрибута "ресурс непрерывной работы" изменяется от 4 до 24.
3. Значение атрибута "напряжение в сети" изменяется от 110 до 240 В.
4. Значение атрибута "число программ" изменяется от 1 до 20 и т.д.

Связи между различными классами объектов.

Помимо классов объектов в ИЛМ отображают связи между различными классами объектов. Такие связи моделируют отношения между объектами различных видов в реальном мире. При отборе связей помещаемых в ИЛМ следует руководствоваться информационными потребностями пользователей базы данных.

Каждая связь характеризуется именем, типом, классом принадлежности и

направлением. Имя связи должно быть глагольным оборотом, например

**«Принадлежит», «Закреплены за», «Входит в»** и т.д.

Все информационные объекты предметной области связаны между собой. Соответствия, отношения, возникающие между объектами предметной области,

называются связями. Различаются связи нескольких типов, для которых введены следующие обозначения:

Различают четыре типа связи:

- **«один к одному» (1:1);**
- **«один ко многим» (1:М);**
- **«многие к одному» (М:1)**
- **«многие ко многим» (М:М).**

Рассмотрим эти типы связей на примерах.

***Пример.*** Дана совокупность информационных объектов, отражающих учебныйпроцесс в вузе:

СТУДЕНТ (Номер, Фамилия, Имя, Отчество, Пол, Дата рождения, Группа)СЕССИЯ (Номер. Оценка 1, Оценка 2, Оценка 3, Оценка 4, Результат) СТИПЕНДИЯ (Результат, Процент)

ПРЕПОДАВАТЕЛЬ (Код преподавателя, Фамилия, Имя, Отчество)

**Связь один к одному** (1:1) предполагает, что в каждый момент времени одному экземпляру информационного объекта А соответствует не более одного экземпляра информационного объекта В и наоборот. Рис. 8 иллюстрирует указанный тип отношений.


Рис. 8. Графическое изображение реального отношения 1:1

Примером связи 1:1 может служить связь между информационными объектамиСТУДЕНТ и СЕССИЯ:

**СТУДЕНТ <-> СЕССИЯ**

Каждый студент имеет определенный набор экзаменационных оценок в сессию. При связи **один ко многим (1 : М**) одному экземпляру информационного объекта

А соответствует 0, 1 или более экземпляров объекта В, но каждый экземпляр объекта В связан не более чем с 1 экземпляром объекта А. Графически данное соответствие имеет вид, представленный на рис. 9.

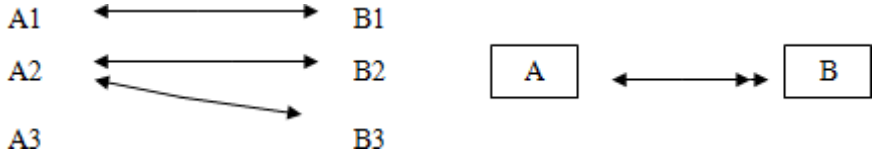



Рис. 9. Графическое изображение реального отношения

1:М Примером связи 1: М служит связь между информационными объектами СТИПЕНДИЯ и СЕССИЯ:

**СТИПЕНДИЯ < -- >> СЕССИЯ**

Установленный размер стипендии по результатам сдачи сессии может повторяться многократно для различных студентов.

Связь **многие ко многим** (M:M) предполагает, что в каждый момент времени одному экземпляру информационного объекта A соответствует 0, 1 или более экземпляров объекта B и наоборот. На рис. 10 графически представлено указанное соответствие.

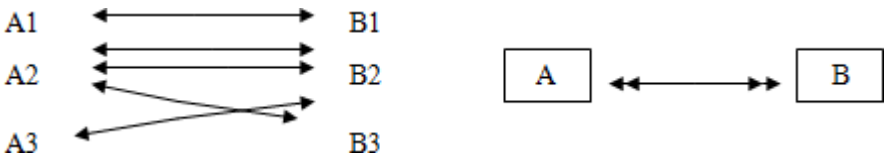



Рис. 10. Графическое изображение реального отношения M : M

Примером данного отношения служит связь между информационными объектамиСТУДЕНТ и ПРЕПОДАВАТЕЛЬ:

**СТУДЕНТ <<—>>ПРЕПОДАВАТЕЛЬ**

Один студент обучается у многих преподавателей, один преподаватель обучает многих студентов.

Связь «многие к одному» при создании БД физически обычно организуется путем введения дополнительного поля в таблицу со стороны «много». Это поле называется **внешний ключ.** На рис.
10. код группы - внешний ключ.

Студент

| Фамилия | Имя | ... | Код группы |
|---|---|---|---|
| Петров | Иван | ... | 2 |
| Сидоров | Сергей | ... | 2 |
| Лаптева | Марина | ... | 1 |
| Киров | Андрей | ... | 3 |
| Жоголева | Ирина | ... | 3 |

Группа

| Код группы | Название группы | Код факультета |
|---|---|---|
| 1 | КТ171 | |
| 2 | МТ181 | |
| 3 | ЕК382 | |

Рис. 10. Введение внешнего ключа в БД

**Построение инфологической (концептуальной модели) предметной области**

Заключительная фаза анализа предметной области состоит в проектировании ее инфологической структуры или концептуальной модели.

**Концептуальная модель** включает описания объектов и их взаимосвязей, представляющих интерес в рассматриваемой предметной области (ПО) и выявляемых в результате анализа данных.

Концептуальная модель применяется для структурирования предметной области с учетом информационных интересов пользователей системы. Она дает возможность систематизировать информационное содержание предметной области, позволяет как бы "подняться вверх" над ПО и увидеть ее отдельные элементы. При этом уровень детализации зависит от выбранной модели.

Концептуальная модель является представлением точки зрения пользователя на предметную область и не зависит ни от программного обеспечения СУБД, ни от технических решений.

Концептуальная модель должна быть стабильной. Могут меняться прикладные программы, обрабатывающие данные, может меняться организация их физического хранения, концептуальная модель остается неизменной или увеличивается с целью включения дополнительных данных.

Одной из распространенных моделей концептуальной схемы является модель **«сущность - связь» (*ER-моделей*** (или ER-диаграмм))**.** Основными конструкциями данной модели являются сущности и связи.

Под **сущностью** понимают основное содержание объекта ПО, о котором собирают информацию. В качестве сущности могут выступать место, вещь, личность, явление.

Экземпляр **сущности** - конкретный **объект**. **Например:**

| |
|---|
| сущность (объект) - служащий экземпляр<br>сущности - Иванов А.В.;сущность<br>(объект) - институт<br>экземпляр сущности - МГУ. |

Сущность принято определять **атрибутами** - поименованными характеристиками.

**Например:**

| |
|---|
| сущность - служащий<br>атрибуты: ФИО, год рождения, адрес, образование и т.д. |

Чтобы задать атрибут в модели, ему надо присвоить имя и определить область допустимых значений. Одно из назначении атрибута - идентифицировать сущность.

**Связь** определяет отношения между сущностями. **Типы связей: один к одному, один ко многим, многие ко многим**.

При построении модели «сущность - связь» используют **графические диаграммы**. При этом обозначают: сущности - прямоугольниками, атрибуты - овалами, связи - ромбами, см. рис.11.

На практике приходится строить несколько вариантов моделей, из которых выбирается одна, наиболее полно отображающая предметную область.

**Классом объектов** называют совокупность объектов, обладающих одинаковым набором свойств. Например, для объектов класса «СТУДЕНТ» таким набором свойств являются: «ГОД_РОЖДЕНИЯ»,

«ПОЛ» и др.

Объекты могут быть реальными, как названный выше объект «СТУДЕНТ», и абстрактными, как, например, «ПРЕДМЕТЫ», которые изучают студенты.

***Пример.*** Спроектировать БД "Сессия". База данных должна выдавать оперативную информацию об успеваемости студентов на факультетах в семестре. Результатами сессии считать только экзамены.

По сути дела *и* БД исходя из формулировки задания можно выделить лишь одно приложение. Речь идет об успеваемости студентов разных факультетов по тем или иным дисциплинам. Более конкретно речь идет о выдаче справок по результатам сессии каждого студента, учебной группы, курса, факультета, а также об автоматизированном составлении ведомости

Выберем следующие сущности:

**ИНСТИТУТ, ФАКУЛЬТЕТ, СТУДЕНТ, ПРЕПОДАВАТЕЛЬ, ДИСЦИПЛИНА.**

В данном примере можно выделить сущность ЭКЗАМЕН или ВЕДОМОСТЬ, но можно не выделять, а сформировать ведомость из имеющихся данных посредством связей.

Зададим каждую сущность набором атрибутов:

**ИНСТИТУТ** (<u>название</u>, подчиненность, адрес, телефон, ФИО ректора)

**ФАКУЛЬТЕТ** (<u>название,</u> код специальности, данные о кафедрах, число выпускников, декан).

**СТУДЕНТ** (<u>ФИО</u>, <u>группа,</u> курс, номер текущего семестра, пол).

**ПРЕПОДАВАТЕЛЬ** (<u>ФИО</u>, должность, звание, <u>кафедра, стаж).</u>

**ДИСЦИПЛИНА** (<u>название,</u> число часов, код дисциплины, виды занятий, число читаемых семестров, номера текущих семестров, на каких курсах преподается)

В каждом наборе атрибутов, характеризующих сущность, необходимо выбрать ключевые атрибуты, т.е. атрибуты, делающие сущность уникальной. При задании атрибутов ключевые атрибуты подчеркивались.

Определим связи между сущностями.

| **Название связи** | **Связи между сущностями** |
|---|---|
| учится | студент, факультет |
| изучает | студент, дисциплина |
| имеет | институт, факультет |
| работает | преподаватель, факультет |
| преподает | преподаватель, дисциплина |
| экзамен | студент, дисциплина, преподаватель |

После выбора сущностей, задания атрибутов и анализа связей можно перейти кпроектированию информационной (концептуальной) схемы БД.

Концептуальная схема БД "Успеваемость» представлена на рис.11 (атрибутысущностей на диаграмме не показаны).

Рассмотрим некоторые **ограничения** в рассматриваемом примере:

1 Значение атрибута "телефон" (сущность - ИНСТИТУТ) задается целым положительным шестизначным числом.

2 Значение атрибута "код факультета" (сущность - ФАКУЛЬТЕТ) лежит в интервале 1-10.

3. Значение атрибута "курс" (сущность - СТУДЕНТ) лежит в интервале 1 - 6

4. Значение атрибута "семестр" (сущность - СТУДЕНТ, ДИСЦИПЛИНА) лежит в интервале 1-12.

5. Значение атрибута "число часов" (сущность - ДИСЦИПЛИНА) лежит в интервале 1-300.

6. Одному студенту может быть приписана только одна группа.

7. Один студент может учиться только на одном факультете.

8. Один студент в семестре сдает от 3 до 5 дисциплин

9. Один студент изучает в семестре от 6 до 12 дисциплин.

10. Одному преподавателю приписывается только одна кафедра.

11. Один студент может пересдавать одну дисциплину не более трех раз.

12. Ключи: название института, название факультета, ФИО и группа студента, ФИО и кафедра преподавателя, название дисциплины.

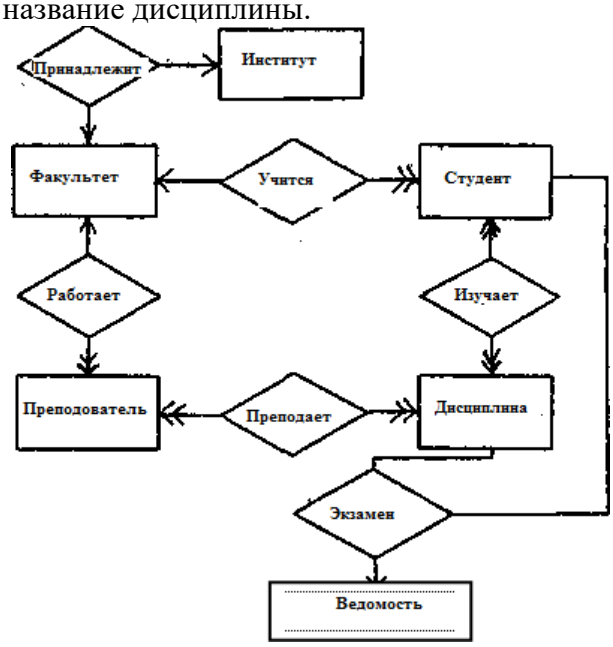


Рис 11. Концептуальная схема БД «Успеваемость»

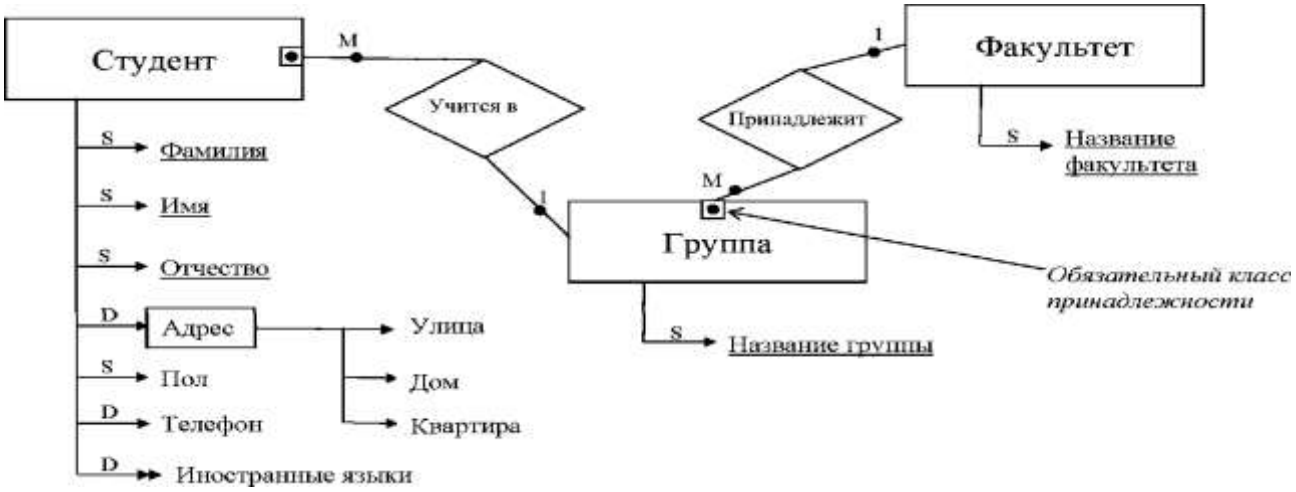



Рис. 12. Пример построения инфологической модели данных.

## 2. ФИЗИЧЕСКОЕ ПРОЕКТИРОВАНИЕ

### Даталогическое проектирование

Инфологическая модель является исходной для построения ***даталогической*** модели БД и служит промежуточной моделью для специалистов предметной области (для которой создается банк данных (БнД)) и администратора БД в процессе проектированияи разработки конкретной БД.

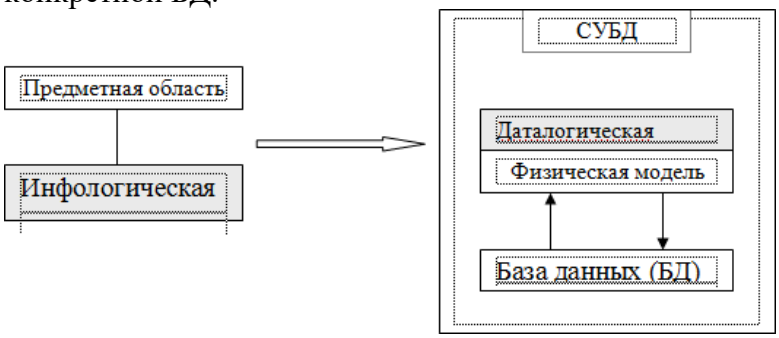



Рис. 13. Структура проектирования БД

Под ***даталогической*** понимается модель, отражающая логические взаимосвязи между элементами данных безотносительно их содержания и физической организации. При этом даталогическая модель разрабатывается с учетом конкретной реализации СУБД, также с учетом специфики конкретной предметной области на основе ее инфологической модели.

Инфологическая модель предметной области строится первой. Предварительная инфологическая модель строится еще на предпроектной стадии и затем уточняется на более поздних стадиях проектирования баз данных. Затем на ее основе строятся концептуальная (логическая), внутренняя (физическая) и внешняя модели.

Конечным результатом даталогического проектирования является описание логической структуры базы данных на языке программирования. Однако если проектирование выполняется

«вручную», то для большей наглядности сначала строится схематическое графическое изображение структуры базы данных. При этом должно быть обеспечено однозначное соответствие между конструкциями языка описания данных и графическими обозначениями информационных единиц и связей между ними.

Графическое представление используется и при автоматизированном проектировании

структуры базы данных как интерфейсное средство общения с проектировщиком, и при документировании проекта.

Спроектировать логическую структуру базы данных означает определить все информационные единицы и связи между ними, задать их имена; если для информационных единиц возможно использование разных типов, то определить их тип. Следует также задать некоторые количественные характеристики, например, длину поля.

Описание датологической модели.

Даталогическая модель представляет собой описание базы данных, выполненное в терминах используемой СУБД. Наиболее часто при разработках баз данных применяют реляционные СУБД. Для СУБД этого типа даталогическая удобно представить в виде набора таблиц специальной формы (табл. 1.4.).

Такая таблица составляется для каждого отношения, используемого в базе банных. Отношения в базе соответствуют классам объектов из инфологической модели. Кроме того, отношения могут представлять некоторые связи предметной области.

Каждой таблице нужно поставить в соответствие ее **ключи**. Схема ключа представляет собой перечисление атрибутов отношения, составляющих ключ.

Различают простые и составные ключи. **Простой ключ** строится на основе одного атрибута.

**Составной ключ** строится на базе использования нескольких атрибутов.

Ключи принято разделять на **первичные, внешние** и **вспомогательные**.

**Первичный индекс** должен быть только один для каждой таблицы. Значения атрибутов, используемых для формирования первичного ключа, должны быть

уникальными для каждой записи в таблице. Значения первичного ключа уникально идентифицируют каждую запись. Не может быть двух записей в таблице с одинаковым значением первичного ключа. Например, в качестве первичного ключа для отношения

«Сотрудники» можно выбрать атрибут «Табельный номер», значение которого является уникальным для каждой записи о сотруднике.

**Внешние ключи** используются для реализации связей типа **1:М** между отношениями. Внешний ключ строится для отношения, находящегося на стороне

**«много»** связи 1:М. Для каждого такого отношения на даталогической модели должен быть показан внешний ключ. Внешний ключ всегда должен иметь соответствующий ему первичный ключ отношения, находящегося на стороне «один» связи типа 1:М.

Для связанных ключа «внешний - первичный» соединяются на схеме даталогической модели ломаной линией. Например, если в таблице «Сотрудники» имеется атрибут

«Шифр категории», то этот атрибут можно использовать в качестве внешнего ключа для связи с таблицей «Категории». В последней таблице должен быть первичный ключ по полю «Шифр категории». Внешних ключей может быть несколько для одной таблицы.

Следует отметить, что первичные и внешние ключи строятся как правило на основе целочисленных атрибутов, а не атрибутов, имеющих строковый или вещественный тип.

Кроме первичных и внешних ключей часто используют **вспомогательные индексы**. Они применяются для реализации связей, получения нужного упорядочения при выводе на экран и создании отчетов и т.д. В каждом случае использования вспомогательного индекса, его необходимость должна быть обоснована. Применение большого количества индексов замедляет работу СУБД, т.к. операции над записями отношения требуют корректировки всех индексов.

## II. ЗАДАНИЕ ВЫПОЛНЕНИЯ ПРАКТИЧЕСКОЙ РАБОТЫ

*Практическая работа №1 выполняется письменно и в конце занятия сдается на проверку. После проверки будет выставлена оценка.*

**Выбор задания**

Выбрать из таблицы «**Варианты заданий для лаб.работы №1.doc**» вариант задания, соответствующий **номеру студента в списке учебной группы**. Для всех последующих практических работ вариант остается неизменным. Каждому студенту предоставляется свой вариант предметной области (ПО), который он будет использовать в процессе выполнения всех практических работ.

**Анализ предметной области**.

На основании выбранного варианта привести: название предприятия, цель деятельности предприятия, структура предприятия, информационные потребностейпользователей (кратко).

**Описание основных сущностей ПО**.

Здесь следует привести описание основных сущностей (объектов) ПО. Отбор сущностей производится на основе анализа информационных потребностей. Необходимо привести таблицы описания сущностей (сущностей должно быть не менее 3-х)

Таблица 1.1. Список сущностей предметной области.

| N п.п. | Наименование сущности | Краткое описание |
|---|---|---|
| | | |
| | | |

Здесь же приводится отбор атрибутов (не менее 5-ти) для каждого экземпляра сущности. Отбираются только те атрибуты сущностей, которые необходимы для формирования ответов на регламентированные и непредусмотренные запросы. Для каждого объекта следует привести таблицы его атрибутов.

Таблица 1.2. Список атрибутов.

| N п.п. | именование атрибута | Краткое описание |
|---|---|---|
| | | |
| | | |

На основе анализа информационных запросов следует выявить связи между сущностями. Для выявленных связей также нужно заполнить таблицу 1.3.

Таблица 1.3. Список связей ПО.

| N п.п. | енование связи | участвующие в связи | Краткое описание |
|---|---|---|---|
| | | | |
| | | | |

**Построение инфологической модели.**

На основании ранее выбранного варианта и таблиц 1.1-1.3:

- описать классы объектов (сущностей) и их свойства,
- расставить существующие связи между ними,
- на основании табл. 1.3. в письменной форме обосновать типы связей (1:1, 1:M и т.д.).

При графическом построении ИЛМ следует придерживаться единого масштаба для всей схемы. Все прямоугольники, обозначающие классы объектов, должны быть одного размера. Аналогично, все ромбы с именами связей также должны иметь одинаковый размер.

**Построение даталогической модели.**

На основании ранее выбранного варианта и таблиц 1.1-1.3, инфологической модели и нормализации БД необходимо:

- провести соответствие ключей для каждой таблицы 1.1-1.3,
- заполнить для каждой таблицы БД форму, согласно табл. 1.4. Таблица 1.4. Структура таблицы для даталогической модели.

| N п.п. | именование реквизита | Идентификатор | Тип | Длина | Формат изображения | граничения и комментарий |
|---|---|---|---|---|---|---|
| | | | | | | |
| | | | | | | |

## III. СОДЕРЖАНИЕ ОТЧЕТА

1. Название и цель работы.
2. Словесный и схематический анализ предметной области (ПО), включая схему структуры предприятия.
3. Заполненные таблицы 1.1 - 1.3. с описанием основных сущностей ПО.
4. Инфологическая модель БД, согласно варианту.
5. Обоснование типов связи в инфологической модели данных.
6. Даталогическая модель БД (табл. 1.4.).

## IV. ПРИМЕР ОФОРМЛЕНИЯ

**Пример. Разработать базу данных «Учеба студентов». Решение.**

**Шаг первый**. **Анализ предметной области.**

Студенты учатся на одном из факультетов, возглавляемом деканатом, в функции которого входит контроль за учебным процессом. В учебном процессе участвуют преподаватели кафедр, административно относящиеся к одному из факультетов. Каждому факультету могут принадлежать несколько кафедр. Студенты кафедр организованные в группы.

Преподаватели кафедр характеризуются фамилией именем и отчеством, должностью, научным званием, ставкой и стажом работы, адресом проживания,возрастом.

Каждая кафедра читает определенный набор закрепленных за ней дисциплин. Каждая дисциплина характеризуется своим полным названием, указанием общего количества часов и формы контроля (зачет, экзамен).

В конце каждого семестра составляется экзаменационно-зачетные ведомости, в которых указываются дисциплины и для каких групп проводится форма контроля, фамилия преподавателя и учебный год и семестр. В каждой такой ведомости составляется список студентов и выставляется оценка.

**Шаг второй. Описание основных сущностей ПО.**

В результате проведенного анализа предметной области базы данных «Учеба студентов» легко перечислить основные сущности этой БД. Так как на физическом уровне сущности соответствует таблица, то просто перечислим основные таблицы БД.

В реляционную модель проектированной БД будут входить следующие таблицы (сущности): Факультет, Кафедра, Преподаватели, Группы, Студенты, Дисциплины, Ведомости.

**Список сущностей.**

| № | Название | Назначение |
|---|---|---|
| 1 | Факультет | Описание факультета и его деканата |
| 2 | Кафедра | Описание кафедры |
| 3 | Преподаватели | Описание состава сотрудников кафедр |
| 4 | Группы | Перечень групп, закрепленных за каждой кафедрой |
| 5 | Студенты | Перечень студентов каждой группы |
| 6 | Дисциплины | Перечень дисциплин, закрепленных за каждой кафедрой |
| 7 | Ведомости | Экзаменнационно-зачетные ведомости с перечнем студентов и их |

| | | |
|---|---|---|
| | | оценками |
| 8 | Подчиненная ведомость | Это таблица внутри таблицы ведомости. Отражает связь один-ко-многим. Так как каждая ведомость выписывается каждой конкретной группе, а студентов в ней много. |

Для каждой таблицы (сущности) приведем описание ее атрибутов. Атрибут нафизическом уровне – это колонки таблицы и выражает определенное свойство объекта.

**Список атрибутов таблицы «Факультеты»**

| **Ключевое поле** | **Название** | **Назначение** |
|---|---|---|
| ПК (первичный ключ) | Код факультета | Ключевое поле, предназначенное для однозначной идентификации каждой записи в таблице. Представляет собой первичный ключ. Это уникальное значение, соответствующее каждому факультету.<br>Это целое число. Т.е. для идентификации каждого факультета будет применятся не названия самих факультетов, а определенный номер. Этот номер может быть случайным целым числом или счетчик по порядку. |
| | Название факультета | |
| | ФИО декана | |
| | Номер комнаты деканата | |
| | Телефон деканата | |

**Список атрибутов таблицы «Кафедра»**

| **Ключевое поле** | **Название** | **Назначение** |
|---|---|---|
| ПК (первичный ключ) | Код кафедры | Ключевое поле. Представляет собой первичный ключ. Это уникальное значение, соответствующее каждой кафедре.<br>Однако для идентификации каждой кафедры первичного ключа недостаточно, так как каждая кафедра принадлежит определенному факультету.<br>Для этого будем использовать внешний ключ. |
| ВК (внешний ключ) | Код факультета | Внешний ключ – это атрибут отношения, который является первичным ключом другого отношения. В нашем случае это атрибут таблицы факультеты. С помощью внешнего ключа будет определено к какому факультету принадлежит каждая кафедра. |
| | Название кафедры | |
| | ФИО заведующего | |
| | Номер комнаты кафедры | |
| | Телефон кафедры | |

**Список атрибутов таблицы «Преподаватели»**

| Ключевое поле | Название | Назначение |
|---|---|---|
| ПК (первичный ключ) | Код преподавателя | Ключевое поле. Представляет собой первичный ключ. Это уникальное значение, соответствующее каждому преподавателю. Это например, может быть его табельный номер. Однако для идентификации каждого преподавателя первичного ключа недостаточно, так как каждый сотрудник принадлежит определенной кафедры. Для этого будем использовать внешний ключ. |
| ВК (внешний ключ) | Код кафедры | С помощью данного внешнего ключа будет определено к какой кафедры принадлежит каждый преподаватель. |
| | ФИО | |
| | должность | Ассистент, доцент, процессор, ст. преподаватель |
| | научное звание | К.т.н., проф., магистр, ст.н.с., м.н.с. |
| | ставка | |
| | стаж работы, | |
| | адрес проживания | |
| | возраст | |

**Список атрибутов таблицы «Группы»**

| Ключевое поле | Название | Назначение |
|---|---|---|
| ПК (первичный ключ) | Код группы | Ключевое поле. Представляет собой первичный ключ. Это уникальное значение, соответствующее каждой группе. Однако для идентификации каждой группы первичного ключа недостаточно, так как каждая группа принадлежит определенной кафедре. Для этого будем использовать внешний ключ. |
| ВК (внешний ключ) | Код кафедры | С помощью данного внешнего ключа будет определено к какой кафедре принадлежит каждая группа. |
| | Номер группы | |
| | Год поступления | |
| | Курс обучения | |

**Список атрибутов таблицы «Студенты»**

| Ключевое поле | Название | Назначение |
|---|---|---|

| | | |
|---|---|---|
| ПК (первичный ключ) | Код студента | Ключевое поле. Представляет собой первичный ключ. Это уникальное значение, соответствующее каждому студенту. Однако для идентификации каждого студента первичного ключа недостаточно, так как каждый студент принадлежит определенной группе. Для этого будем использовать внешний ключ. |
| ВК (внешний ключ) | Код группы | С помощью данного внешнего ключа будет определено к какой группе принадлежит каждый студент. |
| | ФИО | |
| | Год рождения | |
| | Адрес проживания | |

**Список атрибутов таблицы «Дисциплины»**

| **Ключевое поле** | **Название** | **Назначение** |
|---|---|---|
| ПК (первичный ключ) | Код дисциплины | Ключевое поле. Представляет собой первичный ключ. Это уникальное значение, соответствующее каждой дисциплине. Однако для идентификации каждой дисциплины первичного ключа недостаточно, так как каждая дисциплина принадлежит определенной кафедре. Для этого будем использовать внешний ключ. |
| ВК (внешний ключ) | Код кафедры | С помощью данного внешнего ключа будет определено к какой кафедре принадлежит каждая дисциплина. |
| | Название дисциплины | |
| | Расчасовка | |
| | Форма контроля | |

**Список атрибутов таблицы «Ведомости»**

| Ключевое поле | Название | Назначение |
|---|---|---|
| ПК (первичный ключ) | Код ведомости | Ключевое поле. Представляет собой первичный ключ. Это уникальное значение, соответствующее каждой учебной ведомости. Однако для идентификации каждой ведомости первичного ключа недостаточно, так как каждая ведомость выписывается для определенной учебной группе по определенной дисциплине и преподавателя. Для этого будем использовать внешние ключи. |
| ВК (внешний ключ) | Код группы | С помощью данного внешнего ключа будет определено для какой группы выписывается ведомость. |
| ВК (внешний ключ) | Код дисциплины | С помощью данного внешнего ключа будет определено для какой дисциплины выписывается ведомость. |
| ВК (внешний ключ) | Код преподавателя | С помощью данного внешнего ключа будет определено какому преподавателю выписывается ведомость. |
| | Учебный год | |
| | Семестр | |

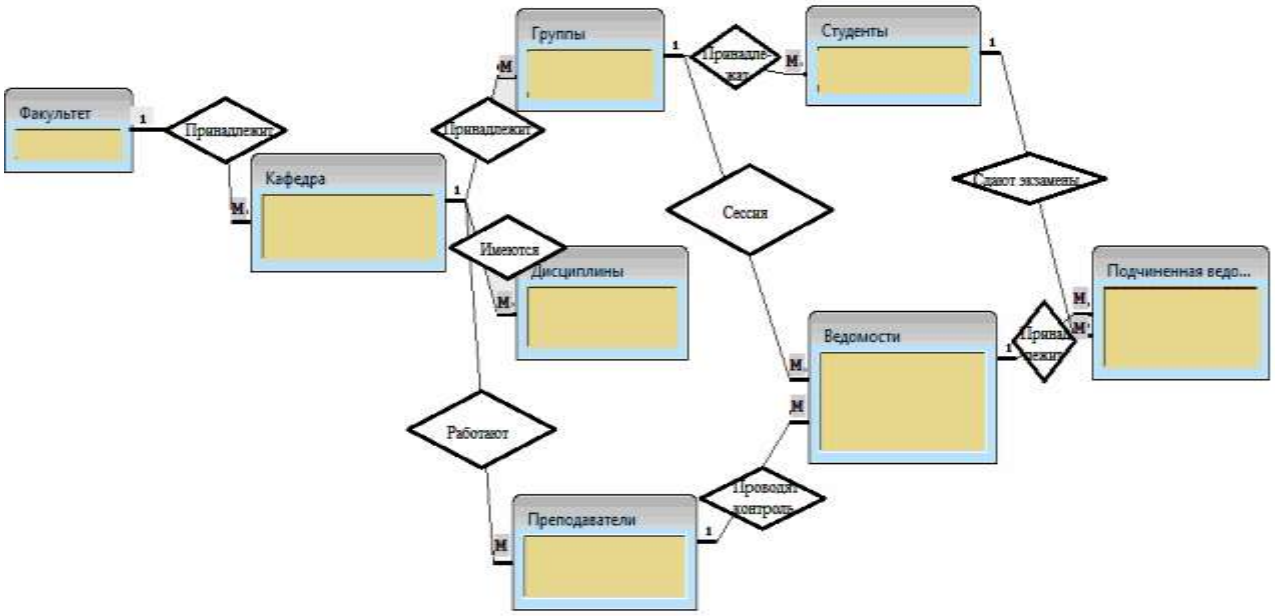
**Список атрибутов таблицы «Подчиненная таблица Ведомости»**

| Ключевое поле | Название | Назначение |
|---|---|---|
| ПК (первичный ключ) | Код под_ведомости | Ключевое поле. Представляет собой первичный ключ. Это уникальное значение, соответствующее каждой подведомости. Однако для идентификации каждой подведомости первичного ключа недостаточно, так как каждая подведомость принадлежит определенной ведомости. Для этого будем использовать внешний ключ. |
| ВК (внешний ключ) | Код ведомости | С помощью данного внешнего ключа будет осуществлена связь с таблицей ведомости. |
| ВК (внешний ключ) | Код студента | С помощью данного внешнего ключа будет определен студент |
| | Оценка | |

**Шаг третий.** Построение **инфологической модели**.

Инфологическую модель лучше представить графически, где будут изображены

всетаблицы и связи между ними. В нашем случае схема связей представлена на рисунке.



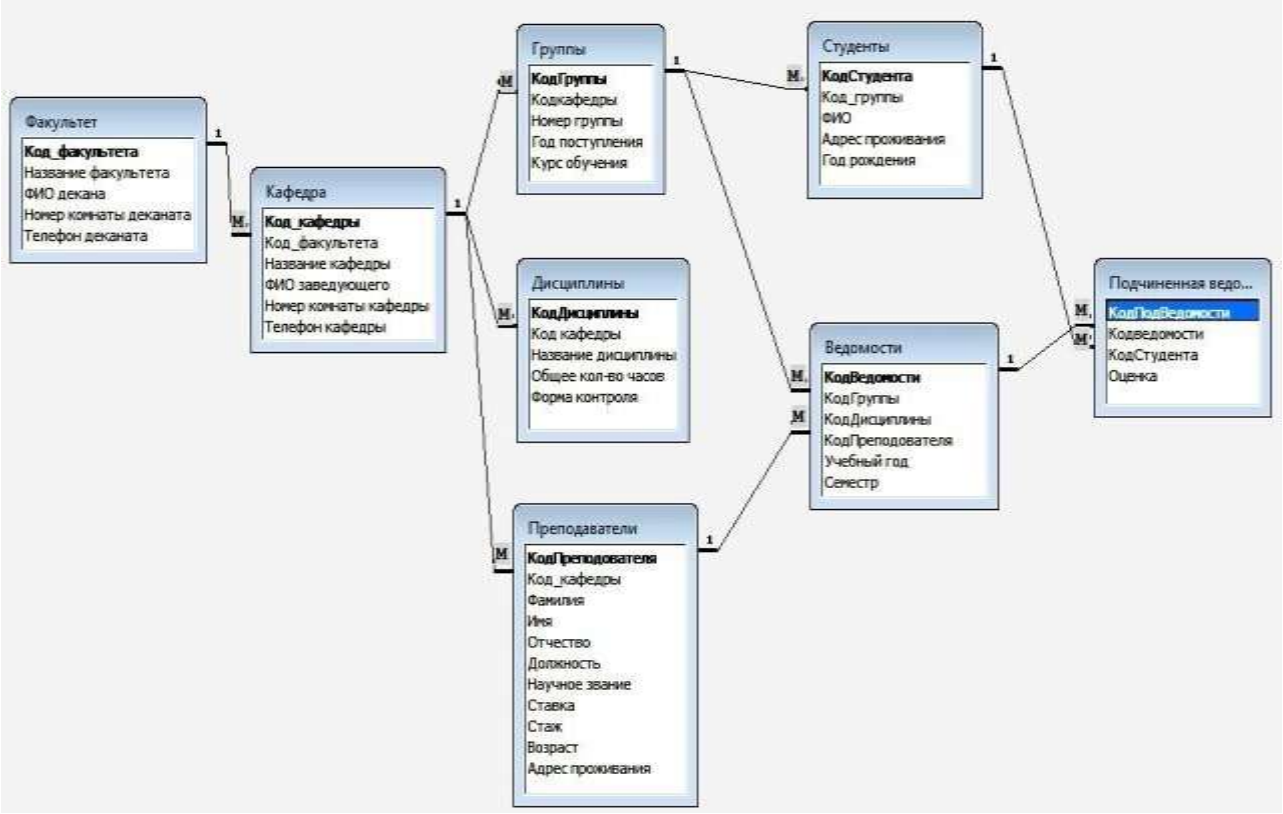

Для выявленных связей заполним таблицу

**Список связей.**

| № | Название связи | Сущности, участвующие в связи | Назначение |
|---|---|---|---|
| 1 | 1:M | Факультет-Кафедра | Одному факультету могут принадлежать несколько кафедр |
| 2 | 1:M | Кафедра - Группа | Одной кафедре может принадлежать несколько групп |
| 3 | 1:M | Кафедра - Дисциплины | Одной кафедре могут принадлежать несколько читаемых дисциплин |
| 4 | 1:M | Кафедра - Преподаватели | На одной кафедре работает более одного преподавателя |
| 5 | 1:M | Группа-Студенты | В каждой группе учится множество студентов |
| 6 | 1:M | Группа - Ведомость | Каждой группе выписывают несколько ведомостей |
| 7 | 1:M | Дисциплины - Ведомость | Ведомость выписывается из множества дисциплин |
| 8 | 1:M | Преподаватели - Ведомость | Ведомость выписывается конкретному преподавателю |
| 9 | 1:M | Ведомость-Подчиненная ведомость | Подчиненная ведомость принадлежит одной конкретной ведомости |
| 10 | 1:M | Студенты-Подчиненная ведомость | В подчиненной ведомости перечислены все студенты группы |

**Шаг четвертый. Построение даталогической модели БД.**

Даталогическая модель отражается графически в виде схемы базы данных, гдеуказываются имена сущностей, их атрибуты и связи между сущностями.
В нашем случае схема связей представлена на рисунке.




Даталогическая модель БД представляется в виде набора таблиц специальной формы, в которых указываются наименование атрибута, идентификатор, тип, длина, формат,ограничения.

**Таблица «Факультеты»**

| № | Название | Идентификатор | Тип | Не пусто | Ограничение |
|---|---|---|---|---|---|
| 1 | Код факультета | Kod_fakulteta | Числовой | Да | ПК (первичный ключ) |
| 2 | Название факультета | Name_fakulteta | Текстовый | Нет | |
| 3 | ФИО декана | FIO | Текстовый | нет | |
| 4 | Номер комнаты деканата | N_komnatu_dekanata | Текстовый | Нет | Например, 123/а |
| 5 | Телефон деканата | Telefon_dekanata | Текстовый | Нет | Например, 41-69-99 |

**Список атрибутов таблицы «Кафедра»**

| № | Название | Идентификатор | Тип | Не пусто | Ограничение |
|---|---|---|---|---|---|
| 1 | Код кафедры | Kod_kafedru | Числовой | Да | ПК (первичный ключ) |

| | | | | | |
|---|---|---|---|---|---|
| 2 | Код факультета | Kod_fakulteta | Числовой | Да | ВК (внешний ключ) |
| 3 | Название кафедры | Name_kafedru | Текстовый | | |
| 4 | ФИО | FIO | Текстовый | нет | |
| | заведующего | | | | |
| 5 | Номер комнаты кафедры | N_komnatu_kafedru | Текстовый | Нет | Например, 123/а |
| 6 | Телефон кафедры | Telefon_kafedru | Текстовый | Нет | Например, 41-69-99 |

**Список атрибутов таблицы «Преподаватели»**

| № | Название | Идентификатор | Тип | Не пусто | Ограничение |
|---|---|---|---|---|---|
| 1 | Код преподавателя | Kod_prepodavately | Числовой | Да | ПК (первичный ключ) |
| 2 | Код кафедры | Kod_kafedru | Числовой | Да | ВК (внешний ключ) |
| 3 | ФИО | FIO | Текстовый | Нет | |
| 4 | должность | Dolgnost | Текстовый | Нет | |
| 5 | научное звание | Zvanie | Текстовый | Нет | |
| 6 | ставка | Stavka | Числовой | Нет | Вещественное число Например, 0.5, 0.75, 1 |
| 7 | стаж работы, | Stag | Числовой | Нет | Вещественное число |
| 8 | адрес проживания | Address | Текстовый | Нет | |
| 9 | возраст | Vozrast | Числовой | нет | |

**Список атрибутов таблицы «Группы»**

| № | Название | Идентификатор | Тип | Не пусто | Ограничение |
|---|---|---|---|---|---|
| 1 | Код группы | Kod_grupu | Числовой | Да | ПК (первичный ключ) |
| 2 | Код кафедры | Kod_kafedru | Числовой | Да | ВК (внешний ключ) |
| 3 | Номер группы | N_grupu | Текстовый | Нет | Например, МТ-461 |
| 4 | Год поступления | God_post | Числовой | нет | |
| 5 | Курс обучения | Kurs | Числовой | Нет | Вычисляемое поле, как разность между текущей датой и годом поступления |

**Список атрибутов таблицы «Студенты»**

| № | Название | Идентификатор | Тип | Не пусто | Ограничение |
|---|---|---|---|---|---|
| 1 | Код студента | Kod_studenta | Числовой | Да | ПК (первичный ключ) |
| 2 | Код группы | Kod_grupu | Числовой | Да | ВК (внешний ключ) |
| 3 | ФИО | FIO | Текстовый | Нет | |
| 4 | Год рождения | God_rogdeniya | Числовой | нет | |
| 5 | Адрес проживания | Address | Текстовый | Нет | |

**Список атрибутов таблицы «Дисциплины»**

| № | Название | Идентификатор | Тип | Не пусто | Ограничение |
|---|---|---|---|---|---|
| 1 | Код дисциплины | Kod_disciplinu | Числовой | Да | ПК (первичный ключ) |
| 2 | Код кафедры | Kod_kafedru | Числовой | Да | ВК (внешний ключ) |
| 3 | Название дисциплины | Name_dis | Текстовый | Нет | |
| 4 | Расчасовка | Raschasovka | Числовой | нет | |
| 5 | Форма контроля | Kontrol | Текстовый | Нет | Два значения – экзамен или зачет |

**Список атрибутов таблицы «Ведомости»**

| № | Название | Идентификатор | Тип | Не пусто | Ограничение |
|---|---|---|---|---|---|
| 1 | Код ведомости | Kod_vedomopsti | Числовой | Да | ПК (первичный ключ) |
| 2 | Код группы | Kod_grupu | Числовой | Да | ВК (внешний ключ) |
| 3 | Код дисциплины | Kod_disciplinu | Числовой | Да | ВК (внешний ключ) |
| 4 | Код преподавателя | Kod_prepodavately | Числовой | Да | ВК (внешний ключ) |
| 5 | Учебный год | God | Числовой | Нет | |
| 6 | Семестр | Semester | Числовой | Нет | Диапазон от 1-10 |

**Список атрибутов таблицы «Подчиненная таблица Ведомости»**

| № | Название | Идентификатор | Тип | Не пусто | Ограничение |
|---|---|---|---|---|---|
| 1 | Код под_ведомости | Kod_pod_vedomopsti | Числовой | Да | ПК (первичный ключ) |
| 2 | Код ведомости | Kod_ edomopsti | Числовой | Да | ВК (внешний ключ) |
| 3 | Код студента | Kod_studenta | Числовой | Да | ВК (внешний ключ) |
| 4 | Оценка | Osenka | Числовой | Нет | Диапазон от 0-12 |

Таблица 1. Варианты заданий для практической работы №1

| № варианта | Условие |
|---|---|

| | |
|---|---|
| **Вариант №1** | На основании выбранного варианта выполнить следующее:<br>Выполнить анализ предметной области исследуемой организации;<br>Описать основные сущности предметной области;<br>3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями;<br>Построить инфологическую модель базы данных организации;<br>Построить даталогическую модель базы данных организации.<br>**БД – успеваемость студентов ВУЗА**. БД состоит из следующих таблиц: факультеты, кафедры, учебные группы, студенты, ведомости успеваемости. |
| | **Таблица факультеты** имеет следующие атрибуты: название факультета, ФИО декана, номер комнаты, номер корпуса, телефон.<br>**Таблица кафедра** имеет следующие атрибуты: название кафедры, факультет, ФИО заведующего, номер комнаты, номер корпуса, телефон, кол-во преподавателей.<br>**Таблица учебные группы** имеет следующие атрибуты: название группы, год поступления, курс обучения, кол-во студентов в группе.<br>**Таблица студенты** имеет следующие атрибуты: студента, фамилия, имя, отчество, группа, год рождения, пол, адрес, город, телефон.<br>**Таблица ведомости успеваемости** имеет следующие атрибуты: группа, студент, предмет, оценка. |
| **Вариант №2** | На основании выбранного варианта выполнить следующее:<br>Выполнить анализ предметной области исследуемой организации;<br>Описать основные сущности предметной области;<br>3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями;<br>Построить инфологическую модель базы данных организации;<br>5. Построить даталогическую модель базы данных организации. **БД – информационная система супермаркета**. БД состоит из следующих таблиц: отделы, сотрудники, товары, продажа товаров, должности.<br>**Таблица отделы** имеет следующие атрибуты: название отдела, кол-во прилавков, кол-во продавцов, номер зала.<br>**Таблица сотрудники** имеет следующие атрибуты: фамилия, имя, отчество, отдел, год рождения, год поступления на работу, стаж, должность, пол, адрес, город, телефон.<br>**Таблица должности** имеет следующие атрибуты: название должности, сумма ставки.<br>**Таблица товары** имеет следующие атрибуты: название товара, отдел, страна производитель, условия хранения, сроки хранения .<br>**Таблица продажа товаров** имеет следующие атрибуты: сотрудника являющегося продавцом, товара дата, время, кол-во, цена, сумма. |

| | |
|---|---|
| **Вариант №3** | На основании выбранного варианта выполнить следующее:<br>Выполнить анализ предметной области исследуемой организации;<br>Описать основные сущности предметной области;<br>3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями;<br>Построить инфологическую модель базы данных организации;<br>Построить даталогическую модель базы данных организации.<br>**БД – информационная система военного округа**. БД состоит из следующих таблиц: места дислокации, вид войск, части, роты, личный состав.<br>**Таблица вид войск** имеет следующие атрибуты: название.<br>**Таблица места дислокации** имеет следующие атрибуты: страна, город, адрес, занимаемая площадь.<br>**Таблица части** имеет следующие атрибуты: номер части, место |
| | дислокации, вид войск, кол-во рот.<br>**Таблица роты** имеет следующие атрибуты: название роты, кол-во служащих.<br>**Таблица личный состав** имеет следующие атрибуты: фамилия, рота, должность, год рождения, год поступления на службу, выслуга лет, награды, участие в военных мероприятиях. |
| **Вариант №4** | На основании выбранного варианта выполнить следующее:<br>Выполнить анализ предметной области исследуемой организации;<br>Описать основные сущности предметной области;<br>3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями;<br>Построить инфологическую модель базы данных организации;<br>Построить даталогическую модель базы данных организации.<br>**БД – информационная система библиотеки**. БД состоит из следующих таблиц: библиотеки, фонд библиотеки, тип литературы, сотрудники, пополнение фонда.<br>**Таблица библиотеки** имеет следующие атрибуты: название, адрес, город.<br>**Таблица фонд библиотеки** имеет следующие атрибуты: название фонда, библиотека, кол-во книг, кол-во журналов, кол-во газет, кол-во сборников, кол-во диссертаций, кол-во рефератов.<br>**Таблица тип литературы** имеет следующие атрибуты: название типа.<br>**Таблица сотрудники** имеет следующие атрибуты: фамилия сотрудника, библиотека, должность, год рождения, год поступления на работу, образование, зарплата.<br>**Таблица пополнение фонда** имеет следующие атрибуты: фонд, сотрудник, дата, название источника литературы, тип литературы, издательство, дата издания, кол-во экземпляров. |

| | |
|---|---|
| **Вариант №5** | На основании выбранного варианта выполнить следующее:<br>Выполнить анализ предметной области исследуемой организации;<br>Описать основные сущности предметной области;<br>3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями;<br>Построить инфологическую модель базы данных организации;<br>Построить даталогическую модель базы данных организации.<br>**БД – информационная система туристического агентства**. БД состоит из следующих таблиц: пансионаты, туры, клиенты, путевки, вид жилья. **Таблица пансионаты** имеет следующие атрибуты: название пансионата, адрес, город, страна, телефон, описание территории, кол-во комнат, наличие бассейна, наличие медицинских услуг, наличие спа-салона, уровень пансионата, расстояние до моря.<br>**Таблица вид жилья** имеет следующие атрибуты: название (дом, бунгало, квартира, 1-я комната, 2-я комната и т.д.), категория жилья (люкс, полулюкс, и т.д.), пансионат, описание условий проживания, цена за номер в сутки.<br>**Таблица туры** имеет следующие атрибуты: название тура (Европа, средняя Азия, тибет и т.д.), вид транспорта, категория жилья на ночь |
| | (гостиница, отель, палатка и т.д.), вид питания (одноразовое, двухразовое, трехразовое, завтраки), цена тура в сутки.<br>**Таблица клиенты** имеет следующие атрибуты: фамилия, имя, отчество, паспортные данные, дата рождения, адрес, город, телефон.<br>**Таблица путевки** имеет следующие атрибуты: клиент, пансионата, вид жилья, дата заезда, дата отъезда, наличие детей, наличие мед. страховки, кол-во человек, цена, сумма. |

| | |
|---|---|
| **Вариант №6** | На основании выбранного варианта выполнить следующее:<br>Выполнить анализ предметной области исследуемой организации;<br>Описать основные сущности предметной области;<br>3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями;<br>Построить инфологическую модель базы данных организации;<br>Построить даталогическую модель базы данных организации.<br>**БД – информационная система автопредприятия города**. БД состоит из следующих таблиц: автотранспорт, водители, маршруты, обслуживающий персонал, гаражное хозяйство.<br>**Таблица автотранспорт** имеет следующие атрибуты: название транспорта (автобусы, такси, маршрутные такси, прочий легковой транспорт, грузовой транспорт и т.д.), кол-во наработки, пробег, кол-во ремонтов, характеристика.<br>**Таблица маршруты** имеет следующие атрибуты: название маршрута, транспорт, водитель, график работы.<br>**Таблица водители** имеет следующие атрибуты: фамилия, имя, отчество, год рождения, год поступления на работу, стаж, должность, пол, адрес, город, телефон.<br>**Таблица обслуживающий персонал** имеет следующие атрибуты: должность (техники, сварщики, слесари, сборщики и др.), фамилия, имя, отчество, год рождения, год поступления на работу, стаж, пол, адрес, город, телефон.<br>**Таблица гаражное хозяйство** имеет следующие атрибуты: название гаража, транспорт на ремонте, вид ремонта, дата поступления, дата выдачи после ремонта, результат ремонта, персонал, производящего ремонт. |
| **Вариант №7** | На основании выбранного варианта выполнить следующее:<br>Выполнить анализ предметной области исследуемой организации;<br>Описать основные сущности предметной области;<br>3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями;<br>Построить инфологическую модель базы данных организации;<br>5. Построить даталогическую модель базы данных организации. **БД – информационная система поликлиники**. БД состоит из следующих таблиц: врачи, пациенты, история болезней, отделения, обслуживающий персонал.<br>**Таблица отделения** имеет следующие атрибуты: название отделения (хирургия, терапия, нервология и т.д.), этаж, номера комнат, ФИО заведующего.<br>**Таблица врачи** имеет следующие атрибуты: фамилия, имя, отчество, |

| | |
|---|---|
| | должность, стаж работы, научное звание, адрес, номер отделения, в котором он работает.<br>**Таблица пациенты** имеет следующие атрибуты: фамилия, имя, отчество, адрес, город, возраст, пол.<br>**Таблица диагнозы** имеет следующие атрибуты: название диагноза, признаки болезни, период лечения, назначения.<br>**Таблица история болезни** имеет следующие атрибуты: пациент, врач, диагноз, лечение, дата заболевания, дата вылечивания, вид лечения (амбулаторное, стационарное). |
| **Вариант №8** | На основании выбранного варианта выполнить следующее:<br>Выполнить анализ предметной области исследуемой организации;<br>Описать основные сущности предметной области;<br>3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями;<br>Построить инфологическую модель базы данных организации;<br>Построить даталогическую модель базы данных организации.<br>**БД – информационная система больницы**. БД состоит из следующих таблиц: врачи, пациенты, история болезней, операции, лист лечения.<br>**Таблица врачи** имеет следующие атрибуты:, фамилия, имя, отчество, должность, стаж работы, научное звание, адрес.<br>**Таблица пациенты** имеет следующие атрибуты: фамилия, имя, отчество, адрес, город, возраст, пол.<br>**Таблица история болезни** имеет следующие атрибуты:, пациента врач, диагноз, дата заболевания, дата вылечивания, вид лечения (амбулаторное, стационарное), код операции.<br>**Таблица лист лечения** имеет следующие атрибуты: дата лечения, история болезни, лекарства, температура, давление, состояние больного (тяжелое, среднее, и т.д.).<br>**Таблица операции** имеет следующие атрибуты: описание операции (удаление аппендицита, пластическая операция и т.д.), врач, дата операции, пациент, результат операции. |

| | |
|---|---|
| **Вариант №9** | На основании выбранного варианта выполнить следующее:<br>Выполнить анализ предметной области исследуемой организации;<br>Описать основные сущности предметной области;<br>Расставить существующие связи между сущностями;<br>Построить инфологическую модель базы данных организации;<br>Построить даталогическую модель базы данных организации.<br>**БД – информационная система библиотек города**. БД состоит из следующих таблиц: библиотеки, читальные залы, литература, читатели, выдача лит-ры.<br>**Таблица библиотеки** имеет следующие атрибуты: название, адрес, город.<br>**Таблица читальные залы** имеет следующие атрибуты: название читального зала, библиотека, кол-во единиц лит-ры, кол-во посадочных мест, время работы, этаж, кол-во сотрудников.<br>**Таблица читатели** имеет следующие атрибуты: фамилия, имя, отчество, категория читателя, место работы или обучения, возраст, дата регистрации в библиотеке. |
| | **Таблица литература** имеет следующие атрибуты: название, категория литературы, авторы, издательство, год издательства, кол-во страниц, читальный зал.<br>**Таблица выдача литературы** имеет следующие атрибуты: читатель, литература, дата выдачи, срок выдачи, вид выдачи, наличие залога. |
| **Вариант №10** | На основании выбранного варианта выполнить следующее:<br>Выполнить анализ предметной области исследуемой организации;<br>Описать основные сущности предметной области;<br>3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями;<br>Построить инфологическую модель базы данных организации;<br>Построить даталогическую модель базы данных организации.<br>**БД – информационная система автосалона**. БД состоит из следующих таблиц: автомобили, марка автомобиля, сотрудники, продажа автомобилей, покупатели.<br>**Таблица марка автомобиля** имеет следующие атрибуты: название марки, страна производитель, завод производитель, адрес.<br>**Таблица автомобиля** имеет следующие атрибуты: название автомобиля, марка, год производства, цвет, категория, цена.<br>**Таблица покупатели** имеет следующие атрибуты: фамилия, имя, отчество, паспортные данные, адрес, город, возраст, пол.<br>**Таблица сотрудника** имеет следующие атрибуты: фамилия, имя, отчество, стаж, зарплата.<br>**Таблица продажа автомобилей** имеет следующие атрибуты: дата, сотрудник, автомобиль, покупатель. |

| | |
|---|---|
| **Вариант №11** | На основании выбранного варианта выполнить следующее:<br>Выполнить анализ предметной области исследуемой организации;<br>Описать основные сущности предметной области;<br>3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями;<br>Построить инфологическую модель базы данных организации;<br>Построить даталогическую модель базы данных организации.<br>**БД – успеваемость студентов кафедры**. БД состоит из следующих таблиц: кафедры, дисциплины, преподаватели, студенты, ведомости успеваемости.<br>**Таблица кафедра** имеет следующие атрибуты: название кафедры, факультет, ФИО заведующего, номер комнаты, номер корпуса, телефон, кол-во преподавателей.<br>**Таблица преподаватели** имеет следующие атрибуты: фамилия, имя, отчество, кафедра, год рождения, год поступления на работу, стаж, должность, пол, адрес, город, телефон.<br>**Таблица студенты** имеет следующие атрибуты: фамилия, имя, отчество, кафедра, год рождения, пол, адрес, город, телефон.<br>**Таблица дисциплины** имеет следующие атрибуты: название дисциплины, кафедра, читаемой эту дисциплину, кол-во часов, вид итогового контроля. |
| | **Таблица ведомости успеваемости** имеет следующие атрибуты: преподаватель, дисциплина, студент, оценка. |

| | |
|---|---|
| **Вариант №12** | На основании выбранного варианта выполнить следующее:<br>Выполнить анализ предметной области исследуемой организации;<br>Описать основные сущности предметной области;<br>3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями;<br>Построить инфологическую модель базы данных организации;<br>Построить даталогическую модель базы данных организации.<br>**БД – торговая организация**. БД состоит из следующих таблиц: торговая организация, торговая точка, продавцы, поставщики, заказы поставщикам.<br>**Таблица торговая организация** имеет следующие атрибуты: название торговой организации, адрес, ФИО директора, налоговый номер.<br>**Таблица торговая точка** имеет следующие атрибуты: название торговой точки, тип (универмаги, магазины, киоски, лотки и т.д.), торговая организация, адрес, ФИО заведующего.<br>**Таблица продавцы** имеет следующие атрибуты: фамилия, имя, отчество, торговая точка, должность, год рождения, пол, адрес проживания, город.<br>**Таблица поставщики** имеет следующие атрибуты: название поставщика, тип деятельности, страна, город, адрес.<br>**Таблица заказы поставщикам** имеет следующие атрибуты: дата заказа, торговая точка, поставщик, название товара, кол-во, цена. |
| **Вариант №13** | На основании выбранного варианта выполнить следующее:<br>Выполнить анализ предметной области исследуемой организации;<br>Описать основные сущности предметной области;<br>3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями;<br>Построить инфологическую модель базы данных организации;<br>5. Построить даталогическую модель базы данных организации. **БД – проектная организация**. БД состоит из следующих таблиц: отделы, сотрудники, организации, договора, проектные работы.<br>**Таблица отделы** имеет следующие атрибуты: название отдела, этаж, телефон, начальник отдела.<br>**Таблица сотрудники** имеет следующие атрибуты: ФИО, должность (конструкторы, инженеры, техники, лаборанты, прочий обслуживающий персонал), номер отдела, в котором работает, пол, адрес, дата рождения.<br>**Таблица организации** имеет следующие атрибуты: название организации, тип деятельности, страна, город, адрес, ФИО директора.<br>**Таблица договора** имеет следующие атрибуты: номер договора, дата заключения договора, организация, стоимость договора.<br>**Таблица проектные работы** имеет следующие атрибуты: дата начала проектной работы, дата завершения проектной работы, номер договора, отдел, осуществляющий разработку. |

| **Вариант** | На основании выбранного варианта выполнить следующее: |
|---|---|
| **№14** | Выполнить анализ предметной области исследуемой организации;<br>Описать основные сущности предметной области;<br>3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями;<br>Построить инфологическую модель базы данных организации;<br>Построить даталогическую модель базы данных организации.<br>**БД – информационная система военно-морского флота**. БД состоит из следующих таблиц: базы, части, личный состав, корабли, учения.<br>**Базы военно-морского флота** имеет следующие атрибуты: название базы, георгафическое расположение, кол-во частей.<br>**Таблица части** имеет следующие атрибуты: номер части, база флота, место базирования, вид войск (морская авиация, морская пехота и т.д.). **Таблица личный состав** имеет следующие атрибуты: фамилия, часть, должность, год рождения, год поступления на службу, выслуга лет, награды,<br>**Таблица корабли** имеет следующие атрибуты: идентификационный номер корабля, название корабля, тип корабля, дата создания, наработка, кол-во посадочных мест, устройство двигателя (парусное, гребное, пароход, теплоход, турбоход, и т.д. ), тип привода (самоходное, несамоходное), размещение корпуса (подводная лодка, ныряющее, полупогружное, и т.д.)<br>**Таблица учения**: часть, корабль, дата учения, место проведения, оценка. |

| | |
|---|---|
| **Вариант №15** | На основании выбранного варианта выполнить следующее:<br>Выполнить анализ предметной области исследуемой организации;<br>Описать основные сущности предметной области;<br>3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями;<br>Построить инфологическую модель базы данных организации;<br>Построить даталогическую модель базы данных организации.<br>**БД – туристическая фирма**. БД состоит из следующих таблиц: туристы, туристичексая группа, состав групп, гостиницы, ведомости продажей.<br>**Таблица туристы** имеет следующие атрибуты: ФИО, паспортные данные, пол, возраст, дети.<br>**Таблица туры** имеет следующие атрибуты: название, страна, города, тип передвижения, тип питания, цена тура, тип проживания.<br>**Таблица туристическая группа** имеет следующие атрибуты: название, дата отправления, дата прибытия, тур, кол-во туристов.<br>**Таблица состав групп** имеет следующие атрибуты: дата продажи, турист, группа, цена билета.<br>**Таблица гостиницы** имеет следующие атрибуты: название гостиницы, страна, город, адрес, кол-во мест, тип гостиницы.<br>**Таблица ведомость продажей** имеет следующие атрибуты: дата, туристическая группа, гостиница, общая стоимость. |
| **Вариант №16** | На основании выбранного варианта выполнить следующее:<br>Выполнить анализ предметной области исследуемой организации; |

| | |
|---|---|
| | Описать основные сущности предметной области;<br>3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями;<br>Построить инфологическую модель базы данных организации;<br>Построить даталогическую модель базы данных организации.<br>**БД – цирк**. БД состоит из следующих таблиц: работники цирка, представления, расписание гастролей, труппа цирка, программа цирка. **Таблица работники цирка** имеет следующие атрибуты: фамилия, имя, отчество, год рождения, год поступления на работу, стаж, должность (акробат, клоун, гимнаст, музыкант, постановщик, служащий и т.д.), пол, адрес, город, телефон.<br>**Таблица представления** имеет следующие атрибуты: название, режиссер-постановщик, художник-постановщик, дирижер-постановщик, автор, жанр, тип.<br>**Таблица расписание гастролей** имеет следующие атрибуты: представление, дата начала, дата окончания, места проведения гастроли. **Таблица труппа представления цирка** имеет следующие атрибуты: представление, актер цирка, название роли.<br>**Таблица программа цирка** имеет следующие атрибуты: представление, дата премьеры, период проведения, дни и время, цена билета. |
| **Вариант №17** | На основании выбранного варианта выполнить следующее:<br>Выполнить анализ предметной области исследуемой организации;<br>Описать основные сущности предметной области;<br>3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями;<br>Построить инфологическую модель базы данных организации;<br>Построить даталогическую модель базы данных организации.<br>**БД – аптека**. БД состоит из следующих таблиц: лекартства, покупатели, продавцы, рецепты, продажа лекарств.<br>**Таблица лекарства** имеет следующие атрибуты: название, тип (готовое, изготовляемое), вид (таблетки, мази, настойки), цена.<br>**Таблица покупатели** имеет следующие атрибуты: фамилия, имя, отчество, адрес, город, телефон.<br>**Таблица продавцы** имеет следующие атрибуты: фамилия, имя, отчество, дата поступления, дата рождения, образование.<br>**Таблица рецепты** имеет следующие атрибуты: номер рецепта, дата выдачи, ФИО больного (покупатель), ФИО врача, диагноз пациента. **Таблица продажа лекарств** имеет следующие атрибуты: дата, лекарство, кол-во, рецепт, продавец. |

| | |
|---|---|
| **Вариант №18** | На основании выбранного варианта выполнить следующее:<br>Выполнить анализ предметной области исследуемой организации;<br>Описать основные сущности предметной области;<br>3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние |
| | ключи между сущностями;<br>Построить инфологическую модель базы данных организации;<br>Построить даталогическую модель базы данных организации.<br>**БД – городская телефонная сеть**. БД состоит из следующих таблиц: АТС, абонент, ведомость звонков, прайс АТС, ведомость абонентской платы.<br>**Таблица АТС** имеет следующие атрибуты: название АТС, вид (городские, ведомственные и учрежденческие), адрес, город, кол-во абонентов.<br>**Таблица абоненты** имеет следующие атрибуты: фамилия, имя, отчество, вид телефона (основной, параллельный или спаренный), номер телефона, межгород (открыт/закрыт), льгота (да/нет), адрес: индекс, район, улица, дом, квартира.<br>**Таблица ведомость звонков** имеет следующие атрибуты: абонент, дата звонка, время начала, время окончания, межгород (да/нет).<br>**Таблица прайс АТС** имеет следующие атрибуты: АТС, цена на городские, цена на межгород.<br>**Таблица ведомость абонентской платы** имеет следующие атрибуты: абонент, месяц, год, кол-во минут на городские, кол-во минут на межгород, стоимость, сумма льготы, общая стоимость. |

| | |
|---|---|
| **Вариант №19** | На основании выбранного варианта выполнить следующее:<br>Выполнить анализ предметной области исследуемой организации;<br>Описать основные сущности предметной области;<br>3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями;<br>Построить инфологическую модель базы данных организации;<br>Построить даталогическую модель базы данных организации.<br>**БД – аэропорт**. БД состоит из следующих таблиц: работники аэропорта, расписание вылетов, самолеты, бригады самолетов, ведомость продаж билетов.<br>**Таблица работники аэропорта** имеет следующие атрибуты: фамилия, имя, отчество, год рождения, год поступления на работу, стаж, должность (пилотов, диспетчеров, техников, кассиров, работников службы безопасности, справочной службы и других,), пол, адрес, город, телефон.<br>**Таблица расписание вылетов** имеет следующие атрибуты: самолет, дата вылета, время вылета, место выбытия, место прибытия, маршрут (начальный и конечный пункты назначения, пункт пересадки), стоимость билета.<br>**Таблица самолеты** имеет следующие атрибуты: номер, год выпуска, кол-во посадочных место, грузоподъемность.<br>**Таблица бригады самолетов** имеет следующие атрибуты: номер бригады, самолет, работник аэропорта (пилоты, техники и обслуживающий персонал)ю<br>**Таблица ведомость продажи билетов** имеет следующие атрибуты: дата и время продажи, ФИО пассажира, паспортные данные, номер рейса, кол- во билетов, наличие льгот (пенсионеры, дети-сироты и т.д.), багаж (да/нет), стоимость. |

| | |
|---|---|
| **Вариант №20** | На основании выбранного варианта выполнить следующее:<br>Выполнить анализ предметной области исследуемой организации;<br>Описать основные сущности предметной области;<br>3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями;<br>Построить инфологическую модель базы данных организации;<br>Построить даталогическую модель базы данных организации.<br>**БД – театр**. БД состоит из следующих таблиц: работники театра, спектакли, расписание гастролец, труппа спектакля, репертуар театра. **Таблица работники театра** имеет следующие атрибуты: фамилия, имя, отчество, год рождения, год поступления на работу, стаж, должность (актеров, музыкантов, постановщиков и служащих), пол, адрес, город, телефон.<br>**Таблица спектакли** имеет следующие атрибуты: название, режисер-постановщик, художник-постановщик, дирижер-постановщик, автор, жанр (музыкальная комедия, трагедия, оперетта и пр), тип (детские, молодежные и пр.).<br>**Таблица расписание гастролей** имеет следующие атрибуты: название, дата начала, дата окончания, места проведения гастроли, спектакль.<br>**Таблица труппа спектакля** имеет следующие атрибуты: спектакль, актер, название роли.<br>**Таблица репертуар театра** имеет следующие атрибуты: спектакль, дата премьеры, период проведения, дни и время, цена билета. |

| | |
|---|---|
| **Вариант №21** | На основании выбранного варианта выполнить следующее:<br>Выполнить анализ предметной области исследуемой организации;<br>Описать основные сущности предметной области;<br>3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями;<br>Построить инфологическую модель базы данных организации;<br>Построить даталогическую модель базы данных организации.<br>**БД – железнодорожный вокзал**. БД состоит из следующих таблиц: работники ж.д.вокзала, расписание движения поездов, поезда, бригады поездов, ведомость продаж билетов.<br>**Таблица работники ж.д.вокзала** имеет следующие атрибуты: фамилия, имя, отчество, год рождения, год поступления на работу, стаж, должность (машинист, диспетчеров, проводник, ремонтников подвижного состава, путей, кассиров, работников службы подготовки составов, справочной службы и других,), пол, адрес, город, телефон.<br>**Таблица расписание движения поездов** имеет следующие атрибуты: поезд, дата отправления, время отправления, место отправления, дата прибытия, время прибытия, место прибытия, маршрут ((начальный и конечный пункты назначения, основные узловые станции), стоимость билета.<br>**Таблица поезда** имеет следующие атрибуты: номер, год выпуска, кол-во вагонов, тип поезда (общий, скоростной, высокоскоростной). |
| | **Таблица бригады поездов** имеет следующие атрибуты: номер бригады, поезд, работник ж.д.вокзала (машинисты, техники, проводники и обслуживающий персонал).<br>**Таблица ведомость продажи билетов** имеет следующие атрибуты: дата и время продажи, ФИО пассажира, паспортные данные, номер рейса, кол- во билетов, наличие льгот (пенсионеры, дети-сироты и т.д.), стоимость. |

| | |
|---|---|
| **Вариант №22** | На основании выбранного варианта выполнить следующее:<br>Выполнить анализ предметной области исследуемой организации;<br>Описать основные сущности предметной области;<br>3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями;<br>Построить инфологическую модель базы данных организации;<br>Построить даталогическую модель базы данных организации.<br>**БД – информационная система ВУЗА**. БД состоит из следующих таблиц: факультеты, кафедры, преподаватели, дисциплины, учебная нагрузка.<br>**Таблица факультеты** имеет следующие атрибуты: название факультета, ФИО декана, номер комнаты, номер корпуса, телефон.<br>**Таблица кафедра** имеет следующие атрибуты: название кафедры, ФИО заведующего, номер комнаты, номер корпуса, телефон, кол-во преподавателей.<br>**Таблица дисциплины** имеет следующие атрибуты: название дисциплины, кол-во часов, цикл дисциплин.<br>**Таблица преподаватели** имеет следующие атрибуты: фамилия, имя, отчество, кафедра, год рождения, год поступления на работу, стаж, должность, пол, город.<br>**Таблица учебная нагрузка** имеет следующие атрибуты: преподаватель, дисциплина, учебный год, семестр, группы, кол-во студентов, вид итогового контроля. |
| **Вариант №23** | На основании выбранного варианта выполнить следующее:<br>Выполнить анализ предметной области исследуемой организации;<br>Описать основные сущности предметной области;<br>3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями;<br>Построить инфологическую модель базы данных организации;<br>Построить даталогическую модель базы данных организации.<br>**БД – информационная система военного округа**. БД состоит из следующих таблиц: места дислокации, вид войск, части, роты, личный состав.<br>**Таблица вид войск** имеет следующие атрибуты: название вида войск.<br>**Таблица места дислокации** имеет следующие атрибуты: страна, город, адрес, занимаемая площадь, кол-во сооружений.<br>**Таблица части** имеет следующие атрибуты: номер части, место дислокации, вид войск, кол-во рот, кол-во техники, кол-во вооружений. **Таблица техника** имеет следующие атрибуты: название техники, часть, характеристики.<br>**Таблица вооружения** имеет следующие атрибуты: название вооружения, |
| | часть, характеристики. |

| | |
|---|---|
| **Вариант №24** | На основании выбранного варианта выполнить следующее:<br>Выполнить анализ предметной области исследуемой организации;<br>Описать основные сущности предметной области;<br>3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями;<br>Построить инфологическую модель базы данных организации;<br>5. Построить даталогическую модель базы данных организации. **БД – информационная система супермаркета**. БД состоит из следующих таблиц: отделы, клиенты, товары, продажа товаров, поставщики.<br>**Таблица отделы** имеет следующие атрибуты: название отдела, кол-во прилавков, кол-во продавцов, номер зала.<br>**Таблица клиенты** имеет следующие атрибуты: название клиента, адрес, вид оплаты.<br>**Таблица поставщики** имеет следующие атрибуты: название поставщика, адрес, страна, вид транспорта, вид оплаты.<br>**Таблица товары** имеет следующие атрибуты: название товара, отдел, поставщик, условия хранения, сроки хранения .<br>**Таблица продажа товаров** имеет следующие атрибуты: клиент, товар, дата, время, кол-во, цена, сумма. |
| **Вариант №25** | На основании выбранного варианта выполнить следующее:<br>Выполнить анализ предметной области исследуемой организации;<br>Описать основные сущности предметной области;<br>3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями;<br>Построить инфологическую модель базы данных организации;<br>Построить даталогическую модель базы данных организации.<br>**БД – информационная система больницы**. БД состоит из следующих таблиц: врачи, пациенты, история болезней, отделения, лист лечения.<br>**Таблица отделения** имеет следующие атрибуты: название отделения (хирургия, терапия, нервология и т.д.), этаж, номера комнат, ФИО заведующего.<br>**Таблица врачи** имеет следующие атрибуты: фамилия, имя, отчество, должность, стаж работы, научное звание, адрес.<br>**Таблица пациенты** имеет следующие атрибуты: фамилия, имя, отчество, адрес, город, возраст, пол.<br>**Таблица история болезни** имеет следующие атрибуты: пациент, врач, диагноз, дата заболевания, дата вылечивания, вид лечения (амбулаторное, стационарное).<br>**Таблица лист лечения** имеет следующие атрибуты: дата лечения,<br>история болезни, лекарства, температура, давление, состояние больного (тяжелое, среднее, и т.д.). |

**Практическая работа № 2 Проектирование реляционной схемы базы данных в среде СУБД** Цель работы: **приобретение практических навыков анализа предметной области, информационных задач и построения концептуальной модели базы данных. (10часов)**

**Проектирование базы данных (БД)**

Проектирование базы данных (БД) – одна из наиболее сложных и ответственных задач, связанных с созданием информационной системы (ИС). В результате еѐ решения должны быть определены содержание БД, эффективный для всех еѐ будущих пользователей способ организации данных и инструментальные средства управления данными.

Основная цель процесса проектирования БД состоит в получении такого проекта, который удовлетворяет следующим требованиям:

- корректность схемы БД, т.е. база должна быть гомоморфным образом моделируемой предметной области (ПО), где каждому объекту предметной области соответствуют данные в памяти ЭВМ, а каждому процессу – адекватные процедуры обработки данных;
- обеспечение ограничений ;
- эффективность функционирования ;
- защита данных (от аппаратных и программных сбоев и несанкционированного доступа);
- простота и удобство эксплуатации;
- гибкость, т.е. возможность развития и адаптации к изменениям предметной области и/или требований пользователей.

Внимание! Базы данных всегда проектируются под конкретное назначение системы.

Техника проектирования баз данных может измениться в целом и в деталях в зависимости от назначения системы. Например, следует различать проектирование систем складирования данных и проектирование так называемых OLTP-систем, ориентируемых на оперативную обработку транзакций. В данном учебном курсе рассматривается проектирование баз данных в основном для OLTP-систем. Именно на таких системах исторически сложилась техника проектирования баз данных.

**Этапы проектирования базы данных**

Процесс проектирования включает в себя следующие этапы:

Концептуальное проектирование – это процедура конструирования информационной модели, не зависящей от каких-либо физических условий реализации.

Логическое проектирование – это процесс конструирования информационной модели на основе существующих моделей данных, не зависимо от используемой СУБД и других условий физической реализации.

Физическое проектирование – это процедура создания описания конкретной реализации БД с описанием структуры хранения данных, методов доступа к данным.

Концептуальное проектирование

Основными задачами концептуального проектирования являются определение предметной области системы и формирование взгляда на ПО с позиций сообщества будущих пользователей БД, т.е. инфологической модели ПО.

Концептуальная модель ПО представляет собой описание структуры и динамики ПО, характера информационных потребностей пользователей в терминах, понятных пользователю и не зависимых от реализации БД. Это описание выражается в терминах не отдельных объектов ПО и связей между ними, а их типов, связанных с ними ограничений целостности и тех процессов, которые приводят к переходу предметной области из одного состояния в другое.

Рассмотрим основные подходы к созданию концептуальной модели предметной области.

1. Функциональный подход к проектированию БД. Этот метод реализует принцип "от задач" и применяется тогда, когда известны функции некоторой группы лиц и/или комплекса задач, для обслуживания информационных потребностей которых создаѐтся рассматриваемая БД.

2. Предметный подход к проектированию БД
   Предметный подход к проектированию БД применяется в тех случаях, когда у разработчиков есть чѐткое представление о самой ПО и о том, какую именно информацию они хотели бы хранить в БД, а структура запросов не определена или определена не полностью. Тогда основное внимание уделяется исследованию ПО и наиболее адекватному еѐ отображению в БД с учѐтом самого широкого спектра информационных запросов к ней.
3. Проектирование с использованием метода "сущность-связь"
   Метод "сущность–связь" (entity–relation, ER–method) является комбинацией двух предыдущих и обладает достоинствами обоих. Этап инфологического проектирования начинается с моделирования ПО. Проектировщик разбивает еѐ на ряд локальных областей, каждая из которых (в идеале) включает в себя информацию, достаточную для обеспечения запросов отдельной группы будущих пользователей или решения отдельной задачи (подзадачи). Каждое локальное представление моделируется отдельно, затем они объединяются.

Выбор локального представления зависит от масштабов ПО. Обычно она разбивается на локальные области таким образом, чтобы каждая из них соответствовала отдельному внешнему приложению и содержала 6-7 сущностей.

Сущность – это объект, о котором в системе будет накапливаться информация. Сущности бывают как физически существующие (например, СОТРУДНИК или АВТОМОБИЛЬ), так и абстрактные (например, ЭКЗАМЕН или ДИАГНОЗ). Для сущностей различают тип сущности и экземпляр. Тип характеризуется именем и списком свойств, а экземпляр – конкретными значениями свойств. Типы сущностей можно классифицировать как сильные и слабые. Сильные сущности существуют сами по себе, а существование слабых сущностей зависит от существования сильных. Например, читатель библиотеки – сильная сущность, а абонемент этого читателя – слабая, которая зависит от наличия соответствующего читателя. Слабые сущности называют подчинѐнными (дочерними), а сильные – базовыми (основными, родительскими).

Для каждой сущности выбираются свойства (атрибуты). Различают:

- Идентифицирующие и описательные атрибуты. Идентифицирующие атрибуты имеют уникальное значение для сущностей данного типа и являются потенциальными ключами. Они позволяют однозначно распознавать экземпляры сущности. Из потенциальных ключей выбирается один первичный ключ (ПК). В качестве ПК обычно выбирается потенциальный ключ, по которому чаще происходит обращение к экземплярам записи. Кроме того, ПК должен включать в свой состав минимально необходимое для идентификации количество атрибутов. Остальные атрибуты называются описательными и заключают в себе интересующие свойства сущности.
- Составные и простые атрибуты. Простой атрибут состоит из одного компонента, его значение неделимо. Составной атрибут является комбинацией нескольких компонентов, возможно, принадлежащих разным типам данных (например, ФИО или адрес). Решение о том, использовать составной атрибут или разбивать его на компоненты, зависит от характера его обработки и формата пользовательского представления этого атрибута.
- Однозначные и многозначные атрибуты (могут иметь соответственно одно или много значений для каждого экземпляра сущности).
- Основные и производные атрибуты. Значение основного атрибута не зависит от других атрибутов. Значение производного атрибута вычисляется на основе значений других атрибутов (например, возраст студента вычисляется на основе даты его рождения и текущей даты).

Спецификация атрибута состоит из его названия, указания типа данных и описания ограничений целостности – множества значений (или домена), которые может принимать данный атрибут. Далее осуществляется спецификация связей внутри локального представления. Связи могут иметь различный содержательный смысл (семантику).

Различают связи типа "сущность- сущность", "сущность-атрибут" и "атрибут-атрибут" для отношений между атрибутами, которые характеризуют одну и ту же сущность или одну и ту же связь типа "сущность-сущность". Каждая связь характеризуется именем, обязательностью, типом и степенью. Различают факультативные и обязательные связи. Если вновь порождѐнный объект одного типа оказывается по необходимости связанным с объектом другого типа, то между этими типами объектов существует обязательная связь (обозначается двойной линией). Иначе связь является факультативной. По типу различают множественные связи "один к одному" (1:1), "один ко многим" (1:N) и "многие ко многим" (M:N). Степень связи определяется количеством сущностей, которые охвачены данной связью. Пример бинарной связи – связь между отделом и сотрудниками, которые в нѐм работают. Примером тернарной связи является связь типа экзамен между сущностями ДИСЦИПЛИНА, СТУДЕНТ, ПРЕПОДАВАТЕЛЬ. Из последнего примера видно, что связь также может иметь атрибуты (в данном случае это Дата проведения и Оценка). Пример ER–диаграммы с указанием сущностей, их атрибутов и связей приведен на рис. 1.


Рисунок 1. Пример ER–диаграммы с однозначными и многозначными атрибутами

**Пример проектирования реляционной базы данных**

В качестве примера возьмем базу данных компании, которая занимается издательской деятельностью. База данных создаѐтся для информационного обслуживания редакторов, менеджеров и других сотрудников компании. БД должна содержать данные о сотрудниках компании, книгах, авторах, финансовом состоянии компании и предоставлять возможность получать разнообразные отчѐты. В соответствии с предметной областью система строится с учѐтом следующих особенностей:

- каждая книга издаѐтся в рамках контракта;
- книга может быть написана несколькими авторами;
- контракт подписывается одним менеджером и всеми авторами книги;
- каждый автор может написать несколько книг (по разным контрактам);
- порядок, в котором авторы указаны на обложке, влияет на размер гонорара;
- если сотрудник является редактором, то он может работать одновременно над несколькими книгами;
- у каждой книги может быть несколько редакторов, один из них – ответственный редактор;
- каждый заказ оформляется на одного заказчика;в заказе на покупку может быть перечислено несколько книг. Выделим базовые сущности этой предметной области:

1. Сотрудники компании. Атрибуты сотрудников – ФИО, табельный номер, пол, дата рождения, паспортные данные, ИНН, должность, оклад, домашний адрес и телефоны. Для редакторов необходимо хранить сведения о редактируемых книгах; для менеджеров – сведения о подписанных контрактах.

2. Авторы. Атрибуты авторов – ФИО, ИНН (индивидуальный номер налогоплательщика), паспортные данные, домашний адрес, телефоны. Для авторов необходимо хранить сведения о написанных книгах.

3. Книги. Атрибуты книги – авторы, название, тираж, дата выхода, цена одного экземпляра, общие затраты на издание, авторский гонорар.

4. Контракты будем рассматривать как связь между авторами, книгами и менеджерами. Атрибуты контракта – номер, дата подписания и участники.

5. Для отражения финансового положения компании в системе нужно учитывать заказы на книги. Для заказа необходимо хранить номер заказа, заказчика, адрес заказчика, дату поступления заказа, дату его выполнения, список заказанных книг с указанием количества экземпляров.

ER–диаграмма издательской компании приведена на рис. 2 (базовые сущности на рисунках выделены полужирным шрифтом).

Анализ информационных задач и круга пользователей системы Система создаѐтся для обслуживания следующих групп пользователей:

- администрация (дирекция);
- менеджеры;
- редакторы;
- сотрудники компании, обслуживающие заказы.

Определим границы информационной поддержки пользователей:

1) Функциональные возможности:

- ведение БД (запись, чтение, модификация, удаление в архив);
- обеспечение логической непротиворечивости БД;
- обеспечение защиты данных от несанкционированного или случайного доступа (определение прав доступа);
- реализация наиболее часто встречающихся запросов в готовом виде;
- предоставление возможности сформировать произвольный запрос на языке манипулирования данными.

2) Готовые запросы:

- получение списка всех текущих проектов (книг, находящихся в печати и в продаже);
- получение списка редакторов, работающих над книгами;
- получение полной информации о книге (проекте);
- получение сведений о конкретном авторе (с перечнем всех книг);
- получение информации о продажах (по одному или по всем проектам);
- определение общей прибыли от продаж по текущим проектам;
- определение размера гонорара автора по конкретному проекту.


Рисунок 2. ER–диаграмма издательской компании

**Задание для практической работы**

**По заданному описанию предметной области построить концептуальную модель базы данных**

Выделите типы сущностей;

- Выделите типы связей и определите для них показатели кардинальности и степень участия сторон;
- Выделите атрибуты и свяжите их типами сущностей и связей;
- Определите потенциальные и первичные ключи сущностей;
- Нарисуйте ER-диаграмму.
- и проанализируйте информационные задачи и группы пользователей.

**Индивидуальные задания к практической работе**

Вариант 1.

Задача – организация учебного процесса в вузе:

- Студенты: паспортные данные, адрес, дата зачисления, номер приказа, факультет, группа, является ли старостой, кафедра (специализация), изучаемые (изученные) предметы, оценки, задолженности, стипендия.
- Учебные курсы: название, факультет(ы), групп(ы), кафедра, семестр(ы), форма отчѐтности, число часов.
- Преподаватели: паспортные данные, адрес, телефон, фотография, кафедра, должность, учѐная степень, начальник (зав. кафедрой), предмет(ы), число ставок, зарплата.

Вариант 2.

Учет и выдача книг в библиотеке вуза:

- Книги: авторы, название, раздел УДК, раздел (техническая, общественно-политическая и т.п.), место и год издания, издательство, количество страниц, иллюстрированность, цена, дата покупки, номер сопроводительного документа (чек, счѐт/накладная), вид издания (книги, учебники, брошюры, периодические издания), инвентарный номер (есть только для книг и некоторых учебников), длительность использования читателями (год, две недели, день), электронная версия книги или ее реферата (отсканированный текст).
- Читатели: номер читательского билета, ФИО, год рождения, адрес, дата записи, вид (студент, аспирант, преподаватель, сотрудник), курс, номер группы, названия взятых книг и даты их выдачи.

Вариант3.

Отдел кадров некоторой компании.

- Сотрудники: ФИО, паспортные данные, фотография, дом. и моб. телефоны, отдел, комната, раб. телефоны (в т.ч. местный), подчинѐнные сотрудники, должность, тип(ы)

работы, задание(я), проект(ы), размер зарплаты, форма зарплаты (почасовая, фиксированная).

- Отделы: название, комната, телефон(ы), начальник, размер финансирования, число сотрудников.
- Проекты: название, дата начала, дата окончания, размер финансирования, тип финансирования (периодический, разовый), задачи и их исполнители, структура затрат и статьи расходов.

Вариант 4.

Отдел поставок некоторого предприятия.

- Поставщики: название компании, ФИО контактного лица, расчѐтный счѐт в банке, телефон, факс, поставляемое оборудование (материалы), даты поставок (по договорам и реальные), метод и стоимость доставки.
- Сырьѐ: тип, марка, минимальный запас на складе, время задержки, цена, продукты, при производстве которых используется, потребляемые объемы (необходимый, реальный, на единицу продукции).

Вариант5.

Пункт проката видеозаписей (внутренний учет).

- Видеокассеты: идентификационный номер видеокассеты, тип видеокассет, дата его создания, компания-поставщик, число штук данного типа (общее, в магазине, выдано в настоящее время, выдано всего, выдано в среднем за месяц), общая длительность записей; записи видеокассет: название, длительность, категория, год выпуска и производитель (оригинала).
- Клиенты: ФИО, паспортные данные, адрес, телефон; заказы, т.е. взятые видеокассеты (сейчас и в прошлом): номер, дата выдачи, дата возвращения, общая стоимость заказа.

Вариант 6.

Пункт проката видеозаписей (информация для клиентов).

- Видеокассеты: краткое описание, внешний вид (этикетка), марка (пустой) видеокассеты, цена за единицу прокатного времени (например: 1 день, 3 дня, неделя), есть ли в наличии, общая длительность записей; записи на видеокассете: название, длительность, жанр (категория), тема, год и страна выпуска (оригинала), кинокомпания, описание, актеры, режиссер.
- Заказы: идентификационные номера и названия выданных видеокассет, дата выдачи, дата возвращения (продления), общая стоимость заказа, возвращены ли кассеты заказа.

Вариант 7.

Кинотеатры (информация для зрителей).

- Фильмы: название, описание, жанр (категория), длительность, популярность (рейтинг, число проданных билетов в России и в мире), показывается ли сейчас (сегодня, на текущей неделе), в каких кинотеатрах показывается, цены на билеты (в т.ч. средние).
- Кинотеатры: название, адрес, схема проезда, описание, число мест (в разных залах, если их несколько), акустическая система, широкоэкранность, фильмы и цены на них: детские и взрослые билеты в зависимости от сеанса (дневной, вечерний и т.п.) и от категории мест (передние, задние и т.п.); сеансы показа фильмов (дата и время начала).

Вариант 8.

Ресторан (информация для посетителей).

- Меню: дневное или вечернее, список блюд по категориям.
- Блюда: цена, название, вид кухни, категории (первое, второе и т.п.; мясное, рыбное, салат и т.п.), является ли вегетарианским, компоненты блюда, время приготовления, есть ли в наличии.
- Компоненты блюд: тип (гарнир, соус, мясо и т.п.), калорийность, цена, рецепт, время приготовления, есть ли в наличии, ингредиенты (продукты) и их расходы на порцию.

Вариант 9.

Задача - информационная поддержка деятельности склада.

База данных должна содержать информацию о наименовании товара, его поставщике, количестве, цене товара, конечном сроке реализации, сроке хранения на складе. Торговый склад производит уценку хранящейся продукции. Если продукция хранится на складе дольше 10 месяцев, то она уценивается в 2 раза, а если срок хранения превысил 6 месяцев, но не достиг 10, то в 1,5 раза. Ведомость уценки товаров должна содержать информацию: наименование товара, количество товара(шт.), цена товара до уценки, срок хранения товара, цена товара после уценки, общая стоимость товаров после уценки.

Вариант10.

Задача – информационная поддержка деятельности адвокатской конторы. БД должна осуществлять:

- ведение списка адвокатов;
- ведение списка клиентов;
- ведение архива законченных дел.

- Необходимо предусмотреть:
- получение списка текущих клиентов для конкретного адвоката;
- определение эффективности защиты (максимальный срок минус полученный срок) с учѐтом оправданий, условных сроков и штрафов;
- определение неэффективности защиты (полученный срок минус минимальный срок);
- подсчѐт суммы гонораров (по отдельных делам) в текущем году;
- получение для конкретного адвоката списка текущих клиентов, которых он защищал ранее (из архива, с указанием полученных сроков и статей).

Вариант11.

Задача – информационная поддержка деятельности гостиницы. БД должна осуществлять:

- ведение списка постояльцев;
- учѐт забронированных мест;
- ведение архива выбывших постояльцев за последний год.
- Необходимо предусмотреть:
- получение списка свободных номеров (по количеству мест и классу);
- получение списка номеров (мест), освобождающихся сегодня и завтра;
- выдачу информации по конкретному номеру;
- автоматизацию выдачи счетов на оплату номера и услуг;
- получение списка забронированных номеров;
- проверку наличия брони по имени клиента и/или названию организации

Вариант12.

Описание предметной области:

- В компании несколько отделов.
- В каждом отделе есть некоторое количество сотрудников, занятых в нескольких проектах и размещающихся в нескольких офисах.
- Каждый сотрудник имеет план работы, т.е. несколько заданий, которые он должен выполнить. Для каждого такого задания существует ведомость, содержащая перечень денежных сумм, полученных сотрудником за выполнение этого задания.
- В каждом офисе установлено несколько телефонов.
- В базе данных должна храниться следующая информация.
- Для каждого отдела: номер отдела ( уникальный), его бюджет и личный номер сотрудника, возглавляющего отдел ( уникальный).
- Для каждого сотрудника: личный номер сотрудника ( уникальный), номер текущего проекта, номер офиса, номер телефона, название выполняемого задания вместе с датой и размером выплат, проведенных в качестве оплаты за выполнение данного задания.
- Для каждого проекта: номер проекта ( уникальный) и его бюджет.
- Для каждого офиса: номер офиса ( уникальный), площадь в квадратных футах, номера всех установленных в нем телефонов.

Вариант13.

Задача – информационная поддержка деятельности спортивного клуба. БД должна осуществлять:

- ведение списков спортсменов и тренеров;
- учѐт проводимых соревнований (с ведением их архива);
- учѐт травм, полученных спортсменами. Необходимо предусмотреть:
- возможность перехода спортсмена от одного тренера к другому;
- составление рейтингов спортсменов;
- составление рейтингов тренеров;
- выдачу информации по соревнованиям;
- выдачу информации по конкретному спортсмену;

- подбор возможных кандидатур на участие в соревнованиях (соответствующего уровня мастерства, возраста и без травм).

Вариант14.

Задача – информационная поддержка деятельности аптечного склада.

В аптечном складе хранятся лекарства. Сведения о лекарствах содержатся в специальной ведомости: наименование лекарственного препарата; количество (в шт.); цена; срок хранения на складе (в месяцах). Лекарства поступают на склад ежедневно от разных поставщиков, отпускаются два раза в неделю по предварительным заказам аптек. Выяснить, сколько стоит самый дорогой и самый дешевый препарат; сколько препаратов хранится на складе более 3 месяцев; сколько стоят все препараты, хранящиеся на складе, отыскать препараты, остаток которых равен нулю , ниже требуемого по заказам.

Вариант15.

-Электронный журнал посещаемости"

Предметная область представлена следующими документами:

- Список студентов
- Журнал посещаемости
- Расписание занятий
- Предусмотреть учет пропусков по уважительным, неуважительным причинам. Подсчет
- пропусков по каждому студенту, за неделю, месяц, заданный период, по конкретному
- предмету. Вариант16.

«Итоги сессии»

База данных должна содержать информацию о двух последних сессиях студентов. Источником информации являются экзаменационные ведомости. Необходимо проводить анализ успеваемости по специальностям, формам обучения, курсам, группам, предметам, вычислять средний балл по указанным критериям, а также число каждых оценок .

**Лабораторная работа № 1 Приведение БД к нормальной форме 3НФ (4 часа)**

**Цель работы:** освоить основные приемы заполнения и редактирования таблиц; познакомиться с простой сортировкой данных и с поиском записей по образцу; научиться сохранять и загружать базы данных.

**Microsoft Office Access**

Access является наиболее сложной программой из всех офисных приложений Microsoft Office. Чтобы начать работу с этой программой, вначале необходимо создать структуру базы данных, подробно ее описать, а затем создать различные формы.

ACCESS – это реляционная СУБД. Это означает, что с ее помощью можно работать одновременно с несколькими таблицами базы данных, эти таблицы между собой связаны. Таблицу ACCESS можно связать с данными, хранящимися на другом компьютере. Данные ACCESS очень просто комбинировать с данными EXCEL, WORD и другими программами Office.

Access во многом похож на Excel. Основное различие между таблицей БД и электронной таблицей – в системе адресации: в электронной таблице адресуется каждая ячейка, а в таблице БД – только поля текущей записи.

Таблицы – основные объекты базы данных (БД). В них хранятся данные. Реляционная база данных может иметь много взаимосвязанных таблиц. Сведения по разным вопросам следует хранить в разных таблицах.

Запрос – это средство, с помощью которого извлекается из базы данных информация, отвечающая определенным критериям. Результаты запроса представляют не все записи из таблицы, а только те, которые удовлетворяют запросу.

Формы – Обеспечивают более наглядную работу с таблицами, с помощью форм в базу

вводят новые данные или просматривают имеющиеся.

Отчеты – средство представления данных таблиц. Отчеты могут быть оформлены надлежащим образом и распечатаны в том виде, в котором требуется пользователю.

Макросы – набор из одной или более макрокоманд, выполняющих определенные операции (открытие форм, печать отчетов)

Модули - это программы, написанные на языке программирования Visual Basic.

**Задание для практической работы Заполните таблицы по образцу**

1. Вызвать программу Access 2007 (Access 2016).
2. В окне системы управления базы данных щелкнуть по значку «Новая база данных». Справа в появившемся окне дать имя новой базе данных «Анкета ГС-31» и щелкнуть по значку папки, находящемуся справа от окна названия  . Откроется окно сохранения, найдите свою папку и сохраните в нее новый файл базы данных «Анкета ГС-31» (вместо ГС-31 укажите номер вашей группы). Затем нажмите на кнопку «Создать».
3. Появится окно «Таблица» (Рисунок 5.1).

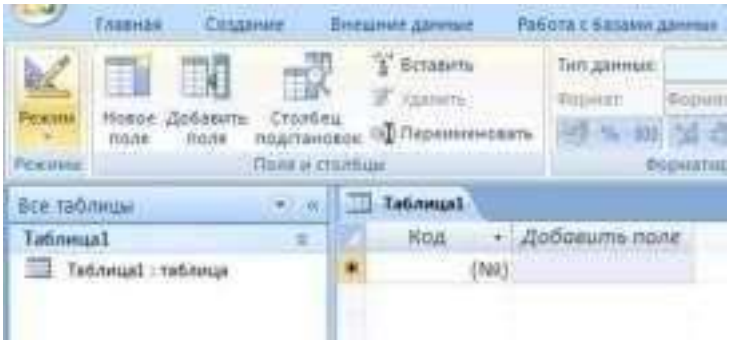



Рисунок 5.1. Окно пустой базы данных

4. В появившемся окне откройте меню команды <Режим> и выберите вариант <Конструктор>  и сохраните будущую таблицу под названием <Ведомость успеваемости>. Появится окно Конструктора.
5. Заполните поля в Конструкторе данными из *рисунка 5.2.* Тип данных можно выбрать из меню, появившемся при нажатии на кнопку  в ячейке справа.

Обратите внимание: *ключевое поле «Счетчик» внесен в таблицу автоматически. Если напротив поля отсутствует значок ключа, то на панели инструментов щелкните по этому значку.* 

Рисунок 5.2. Создание таблицы через конструктор

6. Перейдите в режим таблицы, щелкнув по кнопке **Режим** на панели инструментов, Введите данные в этом режиме, заполняя клетки таблицы. Значение поля **Код** будет меняться автоматически.

7. Заполните базу данных значениями из *таблицы 5.1*. Напротив каждой фамилии выставьте по всем дисциплинам оценки от 2 до 5

Таблица 5.1.

| Код | Фамилия | Имя | Математика | Менеджмент | Сервисная деятельность | Информационные технологии | Стандартизация | Гостиничная индустрия | Пропуски по неуважительной причине | Пропуски по уважительной причине |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Иваникова | Анна | | | | | | | | |
| 2 | Баранова | Ирина | | | | | | | | |
| 3 | Корнилова | Ольга | | | | | | | | |
| 4 | Воробьев | Алексей | | | | | | | | |
| 5 | Воробьев | Олег | | | | | | | | |
| 6 | Скоркин | Алекс | | | | | | | | |
| 7 | Володина | Нина | | | | | | | | |
| 8 | Новоселов | Алексей | | | | | | | | |
| 9 | Петрова | Елена | | | | | | | | |
| 10 | Чернова | Кристина | | | | | | | | |
| 11 | Терещинка | Инна | | | | | | | | |
| 12 | Истратов | Максим | | | | | | | | |
| 13 | Бондарь | Ольга | | | | | | | | |
| 14 | Ревин | Олег | | | | | | | | |
| 15 | Шарова | Оксана | | | | | | | | |

8. Выполните редактирование ячеек:

– Замените фамилию Иванникова на Иванова.

9. Отсортируйте:

а) *фамилии* – по алфавиту (поставьте маркер на любую фамилию в столбце Фамилия и щелкнете мышкой по кноп на панели инструментов или произведите сортировку с помощью контекстного меню)

б) *имя* – по алфавиту

10. Сохраните текущую таблицу, щелкнув по кнопке «крестик» в правом верхнем углу окна таблицы.

11. Откройте снова свою базу данных.

12. Выполните поиск записей по образцу: *найти студентку по фамилии Володина*. Для этого установите курсор в поле фамилия, щелкните на кнопке <Бинокль> на панели инструментов меню Главная и в появившемся диалоговом окне введите в поле <Образец> фамилию *Володина* и щелкните по кнопке <Найти>.

Примечание: Если требуется найти следующую подобную запись, то щелкните мышкой по кнопке <Найти далее>. По окончании работы щелкните по кнопке <Отмена>.

13. Переименуйте поле «Математика» на «Информатика» с помощью контекстного меню. (Верните все как было назад).

14. Скройте столбец Пр н/пр., потом отобразите его назад.

Войдите в режим *Конструктора* и назначьте полю Пр н/пр и Пр ув/пр. *Маску ввода* формы.

«часов». Заполните эти поля данными от 0 до 99.

15. Завершите работу с Access.

**Лабораторная работа № 3 Создание базы данных в среде разработки (4 часа)**

**Цель работы:** Познакомиться с основными принципами создания базы данных в MS SQL Server.

На сегодняшний день известно более двух десятков серверных СУБД, из которых наиболее популярными являются Oracle, Microsoft SQL Server, Informix, DB2, Sybase, InterBase, MySQL.

Для выполнения лабораторных работ будет использоваться сервер " Microsoft SQL Server 2008", установленный на сервере кафедры компьютерных систем и сетей (компьютер pi_srv).


**Microsoft® SQL Server™** — это система анализа и управления реляционными базами данных в решениях электронной коммерции, производственных отраслей и хранилищ данных.

**Microsoft SQL Server** — система управления реляционными базами данных (СУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов — **Transact-SQL**, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими

СУБД в этом сегменте рынка.

В SQL Server 2008 имеется большой набор интегрированных служб, расширяющих возможности использования данных: вы можете составлять запросы, выполнять поиск, проводить синхронизацию, делать отчеты, анализировать данные. Все данные хранятся на основных серверах, входящих в состав центра обработки данных. К ним осуществляется доступ с настольных компьютеров и мобильных устройств. Таким образом, вы полностью контролируете данные независимо от того, где вы их сохранили.

Система SQL Server 2008 позволяет обращаться к данным из любого приложения, разработанного с применением технологий Microsoft .NET и Visual Studio, а также в пределах сервисно-ориентированной архитектуры и бизнес-процессов — через Microsoft BizTalk Server. Сотрудники, отвечающие за сбор и анализ информации, могут работать с данными, не покидая привычных приложений, которыми они пользуются каждый день, например приложений выпуска 2007 системы Microsoft Office.

В Microsoft SQL базы данных хранятся в виде обычных файлов на диске. Как минимум на одну БД приходится таких **файлов 2**: ***.mdf и *.ldf.** В первом хранятся сами данные, таблицы, индексы и пр., а во втором находится т.н. transaction log, в котором находится информация необходимая для восстановления БД.

**Файл с базой данных** представляет собой набор страниц одинакового размера. Размер страницы задается при создании базы данных и может быть изменен только при ее восстановлении из резервной копии. Чтение и запись данных в базе данных осуществляется постранично.

**Все операции с базой данных должны производиться только посредством команд к SQL-серверу**. Для клиентских приложений эти файлы абсолютно бесполезны и при правильной организации доступа пользователей к файлам в сети, вообще не должны быть доступны.

**Сервер СУБД не имеет интерфейса пользователя** и для выполнения операций с базой данных ему необходимо посылать команды либо с помощью командной строки или с помощью какой-либо прикладной программы.

Для выполнения операций с базой данных при проведении лабораторных работ предлагается использовать программу " **SQL Server Management Studio 2008 Rus**"(рис. 1), представляющую собой наиболее распространенное и удобное средствоадминистрирования баз данных под управлением MS SQL Server

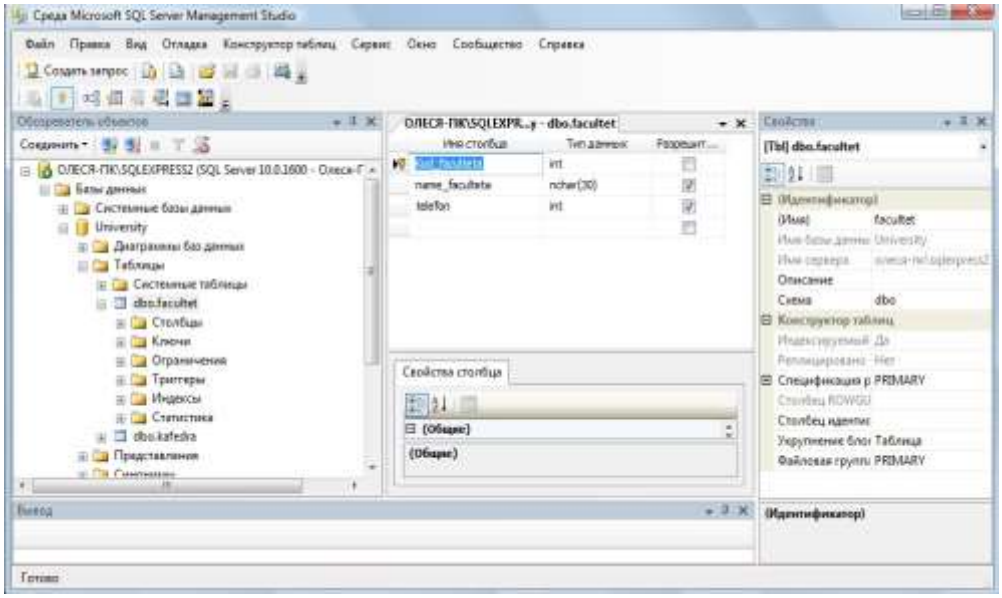



*Рис. 1. Программа* SQL Server Management Studio

**Среда SQL Server Management Studio** — это интегрированная среда для доступа, настройки, управления, администрирования и разработки всех компонентов SQL Server.

Среда SQL Server Management Studio объединяет большое число графических средств с набором полнофункциональных редакторов сценариев для доступа к SQL Server разработчиков и администраторов с любым опытом работы.

Среда SQL Server Management Studio обеспечивает следующие основные возможности:

- поддерживает большинство административных задач для SQL Server;
- единая интегрированная среда для управления SQL Server Database Engine и разработки;
- новые управляющие диалоговые окна для управления объектами в компоненте SQL Server Database Engine, службах Analysis Services, Reporting Services, Notification Services и выпуске SQL Server Compact 3.5 с пакетом обновления 1 (SP1), позволяющие выполнять действия немедленно, направлять их в редактор кода или включать эти действия в сценарий для последующего выполнения;
- экспорт и импорт регистрации сервера среды SQL Server Management Studio из одной среды Management Studio в другую;
- сохранение и печать XML-файлов плана выполнения и взаимоблокировок, созданных приложением SQL Server Profiler, просмотр их в любое время и отправка для анализа администратору;
- новые окна сообщений об ошибках и информационных сообщений, предоставляющие гораздо больше сведений и позволяющие отправлять в Майкрософт комментарии о сообщениях, копировать сообщения в буфер обмена и отправлять их по электронной почте в службу поддержки;
- встроенный веб-обозреватель для быстрого обращения к библиотеке MSDN или получения интерактивной справки;
- встроенная справка от сообществ в Интернете и т.д.

Большинство действий с базой данной MS SQL Server в среде Среда SQL Server Management Studio может быть осуществлено двумя способами: **либо выполнением операторов языка SQL** в окнах "**Script Execute**" (подключение к базе данных не обязательно) и "**SQL Editor**" (требуется подключение к базе данных), либо с использованием меню и диалоговых окон. В последнем случае операторы SQL, которые требуются для выполнения данного действия, будут сгенерированы и выполнены средой SQL Server Management Studio автоматически.

### 2.1. Задание

Лабораторную работу следует выполнять в следующем порядке:

1. Создать на сервере pi_srv (или на локальном компьютере, если нет сервера) рабочую папку для хранения файлов, получаемых при выполнении лабораторной работы. Эта папка должна располагаться в папке **\Базы данных\Группа\Студент** и соответствовать номеру выполняемой лабораторной работы.
2. На основании индивидуального задания выбрать имя файла создаваемой базы данных. Для имени лучше всего выбрать одно или несколько английских слов, соответствующих наименованию предметной области. Использование для имени русских слов, записанных латинскими буквами, не допускается.
3. Открыть приложение " Среда SQL Server Management Studio ". Для этого можно либо воспользоваться меню Пуск (**Пуск/Программы/ Microsoft SQL Server 2008 / Среда SQL Server Management Studio**).
4. Создать соединение с локальным или удаленным сервером.
5. Создать базу данных для своей предметной области с помощью диалога, выбрав сервер "pi_srv" или локальный сервер "**Имя_компьютера\SQLEXPRESS**"
6. Создать базу данных и указать в качестве имени файла **"\Базы данных\Группа\ФИО_студента\Название_БД"**.
7. Извлечь метаданные для автоматической генерации команды создания базы данных.

8. Удалить базу данных, выполнив команду "**Database/Drop Database**" (База данных/Удалить базу данных).

9. Создать базу данных вторым способом, выполнив в окне "**Script Executive**" операторы, полученные при извлечении метаданных перед предыдущим удалением.

10. Создать резервную копию базы данных.

11. Удалить базу данных.

12. Восстановить базу данных из резервной копии.

13. Сохранить файл сценария на сервере в папке "Студент", дав ему имя «лаб.№1» истандартное расширение "***.sql**".

## 2.2. Ход работы

### 2.2.1. Создание соединения с сервером

Выполните следующие инструкции:

Работа с приложением **SQL Server Management Studio** начинается с создания соединения с установленным сервером. Убедитесь вначале, что сервер Microsoft SQL Server (2008) на локальной машине или на сервере компьютерного класса установлен и работает.

Откройте приложение " SQL Server Management Studio ". Для этого можно либо воспользоваться меню Пуск (**Пуск/Программы/ Microsoft SQL Server 2008 / Среда SQL Server Management Studio**).

В диалогом окне **Соединение с сервером** подтвердите заданные по умолчанию параметры и нажмите кнопку **Соединить**, см. рис.2.

Для соединения необходимо, чтобы поле **Имя сервера** содержало имя компьютера,на котором установлен SQL Server.

Если компонент **Database Engine** является именованным экземпляром, то поле **Имясервера** должно также содержать имя экземпляра в формате

**<имя_компьютера>\<имя_экземпляра>**.


Рис. 2. Создание соединения с серверомВ параметрах указываем:

**Тип сервера –** Компонент **Database Engine.**

**Имя сервера**. Подключение может быть ***локальным*** или ***удаленным***. Представляет собой название компьютера в сети, на котором установлен сервер СУБД. Если сервер установлен на том же компьютере, где сейчас работает пользователь, то в качестве имени используется имя компьютера и идентификатор сервера;

**проверка подлинности** – Windows (по умолчанию),

**имя пользователя** –имя пользователя по умолчанию, зарегистрированного на сервере MS SQL Server (задается при установке сервера),

**пароль** – пусто или пароль для пользователя, заданного для сервера MS SQL Server;

Нажмите кнопку **Соединить**. Если соединение будет совершенно успешно, то на экране появятся данные сервера.

Среда Management Studio представляет данные в виде окон, выделенных для отдельных типов данных. Сведения о базе данных отображаются в обозревателе объектов и окнах документов.

Обозреватель объектов является представлением в виде дерева, в котором отображаются все объекты базы данных на сервере. Он может содержать базы данных компонента SQL Server Database Engine, служб Analysis Services, служб Reporting Services, служб Integration Services и SQL Server Compact 3.5 с пакетом обновления 1 (SP1).

Обозреватель объектов включает сведения по всем серверам, к которым он подключен. При открытии среды Management Studio пользователю предлагается применить при подключении обозревателя объектов параметры, которые использовалисьв прошлый раз. Чтобы подключиться к любому из серверов, следует дважды щелкнуть его в компоненте «**Зарегистрированные серверы**», однако регистрировать его не обязательно, см. рис.1.

Окно документов представляет собой наиболее крупную часть среды Management Studio. В окнах документов могут размещаться редакторы запросов и окна обзора. По умолчанию отображается страница «**Сводка**», подключенная к экземпляру компонента Database Engine на текущем компьютере.

### 2.2.2. Общие сведения о базах данных MS SQL Server

Кроме четырех системных баз, SQL Server может обрабатывать до **32 734** баз данных, определяемых пользователем.

**База данных представляет собой:**

- набор взаимосвязанных таблиц;
- связанный набор страниц, выделенных для хранения данных MS SQL Server;
- совокупность данных при архивации;
- два и более файла;
- важную совокупность данных для целей защиты и управления.

**Файлы базы данных**

База данных состоит из двух и более файлов, каждый из которых может использоваться лишь одной базой.

У файлов существуют два имени: **логическое и физическое**. **Логическое имя** подчиняется стандартным правилам выбора имен объектов SQL Server. **Физическое имя** представляет собой полное имя любого локального или сетевого файла. Максимальное число файлов в базе данных — 32 768. **Файлы делятся на три типа:**

- **Первичные файлы**. Используются для хранения данных и информации, определяющих начальные действия с базой. База данных содержит лишь один первичный файл. Стандартное расширение — **.mdf**.
- **Вторичные файлы**. Одна или несколько вспомогательных областей для хранения данных. Могут использоваться для распределения операций чтения/записи по нескольким дискам. Стандартное расширение — **.ndf**.
- **Файлы журналов**. Содержат журналы транзакций базы данных. База данных содержит по крайней мере один файл журнала. Стандартное расширение — **.ldf**. Перед непосредственной записью транзакций в файл данных все вносимые изменения записываются в журнал.

**Группы файлов**

Группы файлов предназначены для объединения нескольких файлов. Каждый файл может входить не более чем в одну группу. Файлы журналов не могут принадлежать никаким группам. Группы файлов используются для распределения операций

чтения/записи по нескольким дискам. Если группа содержит более одного файла, операции записи распределяются между файлами группы. Базы данных могут содержать до 32 768 групп файлов.

У каждой базы данных имеется **первичная группа файлов**. Она содержит первичный файл данных и все файлы, которые не были явно назначены в другую группу файлов. Имя первичной группы файлов — **PRIMARY**.

### 2.2.3. Создание и регистрация базы данных

Для создания базы данных можно использовать один из **двух способов**:

**Первый способ создания БД**. Выполнить команду "**База данных/Создать базу данных**..." в программе SQL Server Management Studio, ввести параметры создаваемой базы данных в диалоговом окне "**Создание базы данных**" (рис. 3) и нажать кнопку [OK].

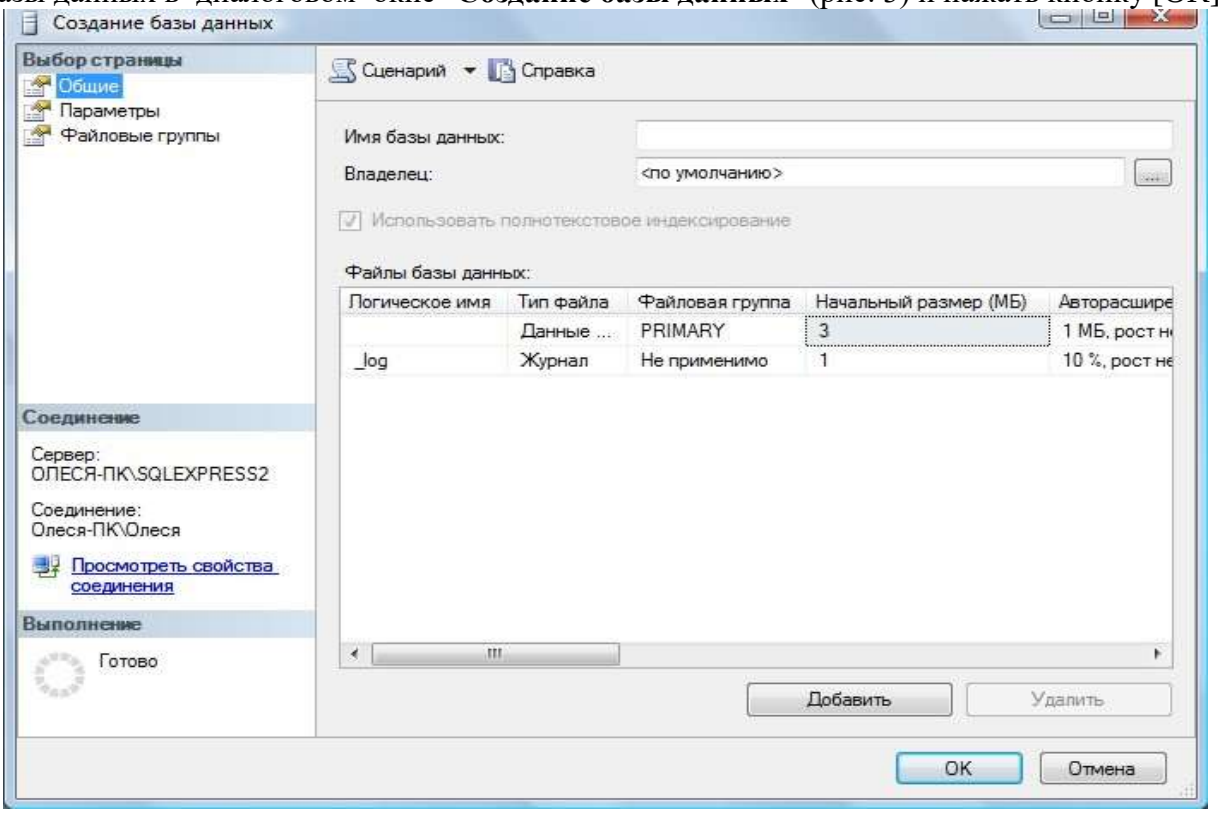



*Рис. 3. Диалоговое окно создания базы данных*

В поле **Имя базы данных** введите имя нашей будущей базы данных, например –

**University**.

Поле **Владелец** - задан по умолчанию, в зависимости от настройки сервера.

Папка с базой данных будет создана по умолчанию на диске **C:\ ProgramFiles\Microsoft SQL Server\MSSQL10.SQLEXPRESS2\MSSQL\DATA \.**

Прежде чем нажать кнопку **Добавить**, просмотрите **Параметры** и **Файловые группы** для создаваемой базы данных.

После нажатия на кнопку [**OK**] программа " SQL Server Management Studio " создаст базу данных, имя которой вы увидите в обозревателе объектов, а также сгенерирует необходимый SQL-код для создания базы данных с теми свойствами, которые указаны в этом диалоговом окне и передаст его серверу СУБД для выполнения.

Пример этих операторов приведен на рис. 4. (нажмите на имени базы данных **University** правой клавишей и из контекстного меню выберите **Создать скрипт как.. CREATE**). Если параметры введены правильно, база данных будет создана.

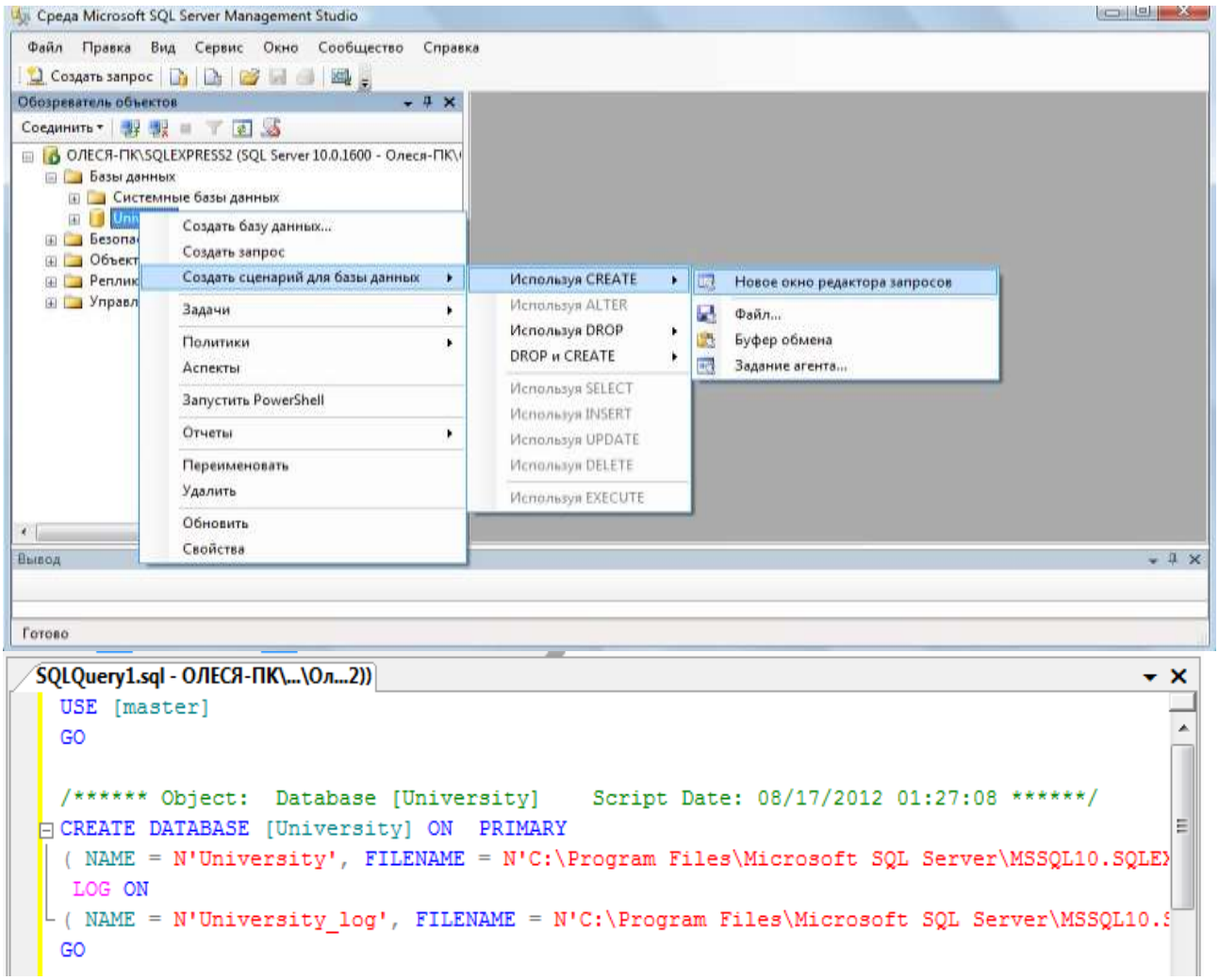



Рис. 4. Сгенерированный sql-код созданной базы данных

Содержащиеся в сценарии операторы отделяются друг от друга символом "**;**". Сценарий может содержать поясняющие комментарии двух видов:

многострочный комментарий (начинается символами "**/***" и заканчивается символами "***/**") и однострочный комментарий, который начинается символами "**--**" и продолжается до конца строки.

При создании базы данных возможны следующие типичные ошибки:

1. На целевом компьютере не запущен или не установлен сервер СУБД –т.е. выполнять команду создания базы данных просто некому.
2. На целевом компьютере нет каталога, в котором предполагается создать базу данных.
3. Файл, в котором должна будет находиться база данных на сервере, уже существует.

После создания базы данных вся введенная о базе данных информация запоминается программой SQL Server Management Studio и в окно редактора в дерево на вкладке "Проводник" добавляется узел с зарегистрированной базой данных (рис. 5).

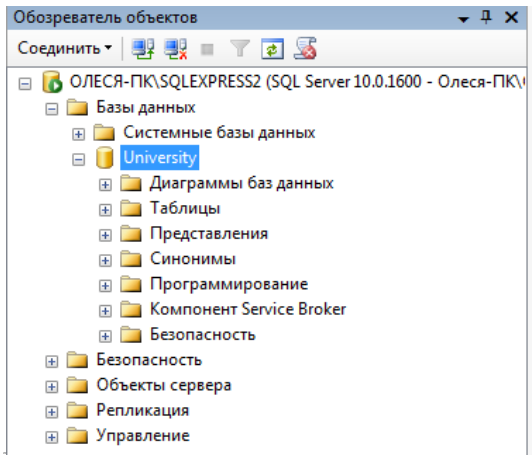
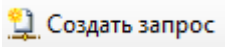
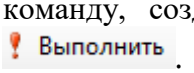



*Рис. 5. Перечень зарегистрированных баз данных в* SQL Server Management Studio

**Второй способ создания БД**. Выполнить в программе SQL Server Management Studio команду "**Создать запрос**" 

на панели инструментов, затем ввести команду, создающую базу данных в окне "Script Execute" (рис. 3) и нажать кнопку 

.

**Команда CREATE DATABASE - Создание базы данных MS SQL Server**

Базы данных создаются командой **CREATE DATABASE**. Создание баз данных разрешается любому пользователю с ролью системного администратора или всем, кому системный администратор предоставил такое право. Команда **CREATE DATABASE** имеет следующий синтаксис:

```
01. CREATE DATABASE имя_базы
02. [ ON [PRIMARY] ([ NAME = логическое_имя_файла, ]
03. FILENAME = 'имя_файла_ОС'
04. [, SIZE = размер]
05. [, MAXSIZE = { максимальный_размер | UNLIMITED } ]
06. [, FILEGROWTH = приращение] )
07. | {FILEGROUP имя_группы_файлов FILEDEFINITIONS}
08. [,...n] ]
09. [LOG ON {[ NAME = логическое_имя_файла, ]
10. [FILENAME = 'имя_файла_ОС'
11. [, SIZE = размер]
12. [, MAXSIZE = { максимальный_размер | UNLIMITED } ]
13. [, FILEGROWTH = приращение] } [,...n]
14. [FOR LOAD | FOR ATTACH]
```

Если при создании базы не указан первичный файл данных и/или файл журнала, то отсутствующий файл (или файлы) создается с именем по умолчанию.

Физические файлы будут находиться в стандартном каталоге.

Первичному файлу присваивается имя **имя_базы.mdf**, а файлу журнала — **имя_базы_log.ldf**.

Если размер файлов не задан, то при создании размер первичного файла совпадает с размером первичного устройства базы **model**, а размер файла журнала и вторичных файлов данных равен 1 Мбайт. Он может быть и больше, если размер первичного файла базы данных model превышает 1 Мбайт. Хотя имена и размеры файлов указывать не обязательно, на практике это всегда следует делать. SQL Server создает базу данных за два этапа. На первом этапе база model копируется в новую базу данных, а на втором этапе инициализируется все неиспользуемое пространство.

Команда **CREATE DATABASE** имеет следующие параметры

- **PRIMARY** — файл определяется как первичное устройство.
- **NAME** — логическое имя; по умолчанию совпадает с именем файла.

- **FILENAME** — полное имя файла на диске.
- **SIZE** — исходный размер файла. Минимальный размер файла журнала равен 512Кбайт.
- **MAXSIZE** — максимальный размер файла.
- **UNLIMITED** — размер файла не ограничивается.
- **FILEGROWTH** — приращение размера в мегабайтах (MB), килобайтах (KB) илипроцентах (%). По умолчанию приращение равно 10%.
- **FOR LOAD** — обеспечивает обратную совместимость со сценариями SQL,написанными для предыдущих версий SQL Server.
- **FOR ATTACH** — указывает, что файлы базы данных уже существуют.

Пользователь, создавший базу данных, является ее владельцем. Все параметры конфигурации базы копируются из базы model, если только при создании базы не был указан параметр **FOR ATTACH.** В этом случае параметры конфигурации читаются из существующей базы данных. Рассмотрим некоторые примеры команды **CREATE DATABASE**:

/* База данных со стандартным размером и именами файлов */

```
01. CREATE DATABASE test1
02. /* Данные – 2 Мбайт, файл журнала – по умолчанию */
03. CREATE DATABASE test2
04. ON (FILENAME = 'c:\d1.mdf', SIZE = 2, NAME = 'd1')
05. /* Первичный файл – 10 Мбайт, одна группа файлов
06. g1 и журнал размером 10 Мбайт */
07. CREATE DATABASE test3
08. ON PRIMARY (FILENAME ='c:\test3.mdf',
09. SIZE = 10 , NAME = 'd1'),
10. FILEGROUP g1 (FILENAME – 'c:\g1.mdf',
11. SIZE = 10 , NAME = 'g1')
12. LOG ON (FILENAME ='c:\test3.ldf',
13. SIZE = 10, NAME = 'log1')
```

**Задача 1.** Создайте sql-скрипт создания новой базы данных под именем **Educator** на **"D:\Базы данных\Группа\ФИО_студента\Название_БД.mdf**, с первичным устройством, с исходным размером файла в 10 Мбайт и запустите на выполнение скрипт (кнопка ! на панели инструментов). Выполните в окне обозревателя объектов **Обновление**. Сохраните созданный скрипт в текущую папку под именем **1.sql**.

После успешного выполнения и обновления проводника у вас должна появится новая база данных.

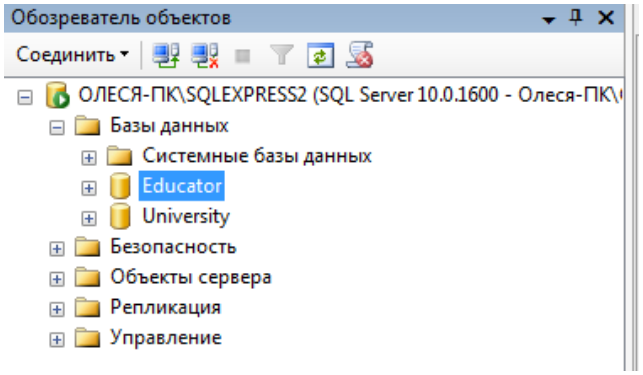



*Рис. 6. Окно проводника после выполнения сценария создания базы данных*

**Лабораторная работа № 4 Создание и использование фильтров (4 часа)**

**Цели работы:** Изучить используемый в реляционных СУБД оператор извлечения данных из таблиц. Получить навыки работы с оператором SELECT в программе ‘SQL Server Managmant Studio’.

**Задание:**

В SQL имеется единственный оператор, который предназначен для выборки данных из базы данных. Оператор относится к подмножеству **DML**.

Ниже приведен почти полный синтаксис оператора **SELECT**.

**SELECT** [**DISTINCT** | **ALL**]
**{*** | <величина> [, <величина> ...]**}** [**INTO** :Переменная [, :Переменная ...]] **FROM** <tableref> [, <tableref> ...] [**WHERE** <условие поиска>]
[**GROUP BY** Колонка [, Колонка ...]][**HAVING** <условие поиска>] [**UNION** [**ALL**] <select_expr>] [**ORDER BY** <список сортировки>];

**<величина>** = **{**Колонка | :Переменная | <константа>
| <выражение> | <функция>
| udf **(**[<величина> [, <величина> ...]]**)**
| **NULL** | **USER}** [**AS** Псевдоним]

**<константа> =** Число | 'Строка'

**<выражение>** = SQL выражение, возвращающее единичное значение

<функция> =
**COUNT (*** | [**ALL**] <величина> | **DISTINCT** <величина>**)**
| **SUM (**[**ALL**] <величина> | **DISTINCT** <величина>**)**
| **AVG** ([**ALL**] <величина> | **DISTINCT** <величина>**)**
| **MAX (**[**ALL**] <величина> | **DISTINCT** <величина>**)**
| **MIN (**[**ALL**] <величина> | **DISTINCT** <величина>**)**
| **CAST(**<величина> **AS** <тип данных>**)**
| **UPPER (**<величина>**)**
| **GEN_ID (**Имя_Генератора, <величина>**)**
**<tableref> = {**<joined_table> | table | view
| procedure[**(**<величина> [, <величина> ...]**)**]**}**[Псевдоним]

**<joined_table> =** <tableref> <join_type> **JOIN** <tableref>
**ON** <условие поиска> | **(**<joined_table>**)**

<join_type> = [**INNER**] | **{LEFT** | **RIGHT** | **FULL }** [**OUTER**]

**<условие поиска> =**
<величина> <оператор сравнения>
**{**<величина> | **(**<select_one>**)}**
| <величина> [**NOT**] **BETWEEN** <величина> **AND** <величина>
| <величина> [**NOT**] **LIKE** <величина>
| <величина> [**NOT**] **IN**
**(**<величина> [, <величина> ...] | <select_list>**)**
| <величина> **IS** [**NOT**] **NULL**
| <величина> **{**>= | <=**}** <величина>
| <величина> [**NOT**] **{**= | < | >**}** <величина>
| **{ALL** | **SOME** | **ANY} (**<select_list>**)**
| **EXISTS (**<select_expr>**)**
| **SINGULAR (**<select_expr>**)**
| <величина> [**NOT**] **CONTAINING** <величина>

| <величина> [**NOT**] **STARTING** [**WITH**] <величина>
| **(**<условие поиска>**)**
| **NOT** <условие поиска>
| <условие поиска>**OR** <условие поиска>
| <условие поиска>**AND** <условие поиска>

**<оператор сравнения> =**
**{**= | < | > | <= | >= | !< | !> | <> | !=**}**

**<select_one>** = оператор SELECT, выбирающий одну колонку и возвращающий ровно одно значение

**<select_list>** = оператор SELECT, выбирающий одну колонку, возвращающий ноль или много значений

**<select_expr>** = оператор SELECT, выбирающий несколько величин и возвращающий ноль или много значений

**<список сортировки> =**
**{**Колонка | Номер**}**[**ASC** | **DESC**]
[, <список сортировки> ...]

Некоторые параметры, входящие в этот оператор, описаны в табл. 5.1.
Таблица 5.1
**Описание параметров оператора SELECT**

| **Параметр** | **Описание** |
|---|---|
| **DISTINCT \| ALL** | **DISTINCT** – предотвращает дублирование данных, которыебудут извлечены.<br>**ALL** (по умолчанию) – приведет к извлечению всех |
| **{* \|** <величина> [, <величина> ...]**}** | Звездочка (*****) означает, что надо извлекать все колонки изуказанных таблиц.<br><величина> [, <величина> ...] – извлекает список указанных |
| **INTO** :Переменная [,<br>:Переменная ...] | Используется только в триггерах и хранимых процедурах для операторов SELECT, возвращающих не более одной строки.<br>Указывается список переменных, в которые извлекаютсявеличины |
| **FROM** <tableref> [, <tableref> ...] | Указывает список таблиц, просмотров и хранимых процедур, из которых извлекаются данные. Список можетвключать соединения и соединения могут быть |
| **table** | Имя существующей в базе данных таблицы |
| **view** | Имя существующего базе данных просмотра |
| **procedure** | Имя существующей хранимой процедуры, предназначеннойдля использования в операторе SELECT |
| **Псевдоним** | Короткое альтернативное имя для таблицы, просмотра иликолонки. После описания в <tableref>, псевдоним может использоваться для ссылок на таблицу или просмотр |

| | |
|---|---|
| **join_type** | Задает тип соединения, которое может быть внутренним иливнешним |
| **WHERE** <условие поиска> | Указывает условие, которое ограничивает количество извлекаемых строк |
| **GROUP BY** Колонка [,Колонка ...] | Разбивает результат запроса на группы, содержащие |
| **HAVING** <условие поиска> | Использует совместно с GROUP BY. Задает условие, котороеограничивает количество возвращаемых групп |
| **UNION** [**ALL**] | Объединяет результаты нескольких запросов. Все запросы должны извлекать одинаковое количество столбцов, тип данных каждого столба первого запроса должен совпадать с типом данных других запросов, имена столбцов в разных запросах мо- гут отличаться. Необязательный параметр ALL |
| **ORDER BY** <список сортировки> | Указывает колонки, по которым будет производиться сортировка извлекаемых строк. Можно указывать либо имена коло- нок, либо их порядковые номера в списке извлекаемых колонок. Если указать ASC, то строки будут выдаваться в порядке возрастания значений сортируемых полей, если DESC |

Как видно из синтаксиса оператора **SELECT,** обязательными являются только предложение **SELECT** с перечнем выдаваемых колонок и предложение **FROM**.

**Пример простейшего оператора SELECT**:

```
-- Выдать перечень всех служащих:
SELECT * FROM Employee;
```

Ниже приведено несколько упрощенных вариантов синтаксиса оператора **SELECT**, помогающих научиться составлять простые запросы.

**Упрощенный синтаксис внутреннего соединения** (стандарт SQL-92):

```
SELECT Колонка [, Колонка ...] | *
FROM <tableref_left> [INNER] JOIN <tableref_right>[ON <условие поиска>]
[WHERE <условие поиска>];
```

**Упрощенный синтаксис внешнего соединения**:

```
SELECT Колонка [, Колонка ...] | *FROM <tableref_left>
{LEFT | RIGHT | FULL} [OUTER] JOIN
<tableref_right>
[ON <условие поиска>] [WHERE <условие поиска>];
```

**Упрощенный синтаксис использования подзапроса**:

```
SELECT [DISTINCT] Колонка [, Колонка ...]
FROM <tableref> [, <tableref> ...]
WHERE
{expression {[NOT] IN | <оператор сравнения>}
| [NOT] EXISTS
}
(SELECT [DISTINCT] Колонка [, Колонка ...]
FROM <tableref> [, <tableref> ...]
```

**WHERE** <условие поиска>
);

### 5.1. Задание

Лабораторную работу следует выполнять в следующем порядке:

1. Изучить синтаксис оператора **SELECT** и примеры запросов к учебной базе данных '**University.mdf**'.
2. Выполнить в окне 'SQL Editor' **27** запроса к базе данных, согласно приведенным в практической работе образцам выполнения запросов и сохранять каждый под именами '**Lab5-k.sql'**, где k – номер запроса по порядку, в своей рабочей папке. Каждый запрос должен иметь комментарии с описанием, а файл в целом должен иметь комментарии со сведениями об авторе и дате создания.

### 5.2. Ход работы

**Примечание.** У вас должны быть перед выполнением этой лабораторной работы созданы все таблицы базы данных университета, созданы ключи, а также заполнены данными.

**Выполнение sql-запросов**

Для выполнения запросов **SELECT** в программе 'SQL Server Managmant Studio' необходимо выполнить следующие действия:

Подключиться к базе данных и выполнить команду '**Создать запрос**'. В результате откроется окно '**Конструктора запросов**' (рис. 1).

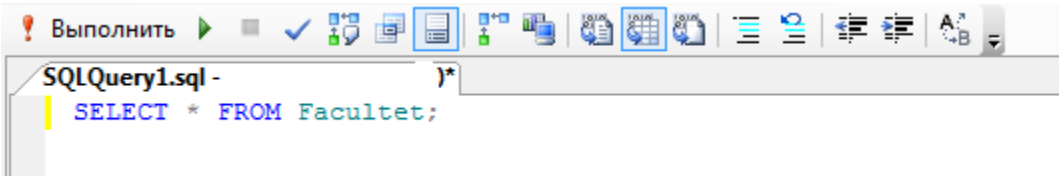
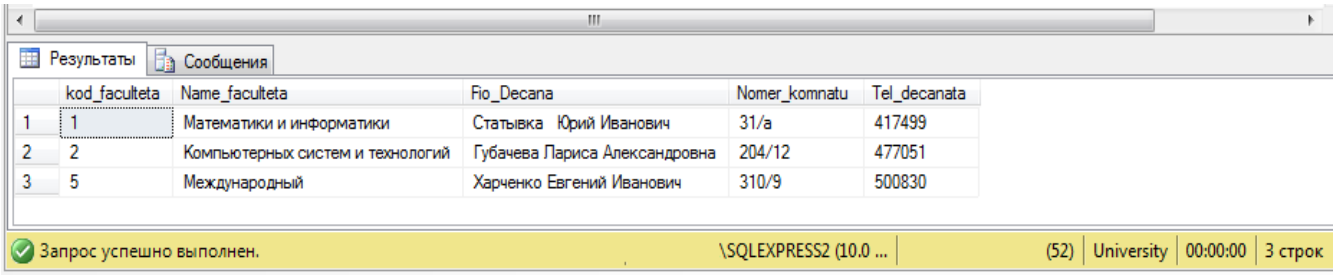



*Рис. 1. Окно выполнения запросов*

1. Ввести текст запроса согласно рис.1.
2. Нажать на панели инструментов кнопку  [**Выполнить**] .
3. Если запрос правильный, то в результате произойдет его выполнение и результат будет отображен на вкладке '**Результаты**' (рис. 2).


Результаты | Сообщения

| | kod_faculteta | Name_faculteta | Fio_Decana | Nomer_komnatu | Tel_decanata |
|---|---|---|---|---|---|
| 1 | 1 | Математики и информатики | Статывка Юрий Иванович | 31/a | 417499 |
| 2 | 2 | Компьютерных систем и технологий | Губачева Лариса Александровна | 204/12 | 477051 |
| 3 | 5 | Международный | Харченко Евгений Иванович | 310/9 | 500830 |

Запрос успешно выполнен. | \SQLEXPRESS2 (10.0 ... | (52) | University | 00:00:00 | 3 строк

*Рис. 2. Окно с результатом выполнения запроса*

4. Количество извлеченных в результате выполнения запроса строк отображается над сеткой с данными справа. На рис.2 там содержится строка '3 строк'. В ТР РТ анном примере извлечено столько строк, сколько требуется, чтобы заполнить сетку (в ней помещается только 3 строки) * .
5. Чтобы узнать, сколько всего строк соответствуют выполненному оператору, надо перейти в конец отображаемого набора данных.

Чтобы выполнить другой запрос, надо вернуться на вкладку ‘**Редактора**’, создать новый запрос и повторить те же действия.

К тексту ранее выполнявшихся правильных запросов можно вернуться, если перейти на вкладку ‘**История**’.

**Примеры создания запросов с отбором строк по условию**.

SQL дает возможность определить критерии отбора необходимых строк во фразе WHERE предложения SELECT. В этом случае строки исходных таблиц будут включены в результирующую только если строка соответствует указанным критериям. *Условие* - это выражение, которое может быть истинным или ложным (логическое выражение или предикат), то есть принимать логические значения TRUE или FALSE соответственно. В результирующую таблицу включаются только те строки, для которых указанное во фразе WHERE условие равно TRUE (иными словами, которые удовлетворяют заданному условию).

**В случае одной таблицы механизм работы предложения SELECT с фразой WHERE следующий.**

1. Из таблицы, указанной во фразе **FROM**, выбирается очередная строка.
2. Она проверяется на соответствие условию во фразе **WHERE**.
3. Если результат равен **TRUE**, строка включается в результирующую таблицу иформатируется в соответствии с фразой **SELECT**, а если он равен **FALSE**, строка пропускается.

Далее будут рассмотрены основные выражения, допустимые для условия во фразе WHERE.

Использование простейших условий

Простейшими считаются условия, в которых используются операторы сравнения и логические операторы.

Хотя такие условия являются простейшими в смысле семантики конструкций, они могут иметь довольно сложную структуру из многих операторов сравнений и вложенных друг в друга логических связок.

Операторы сравнения

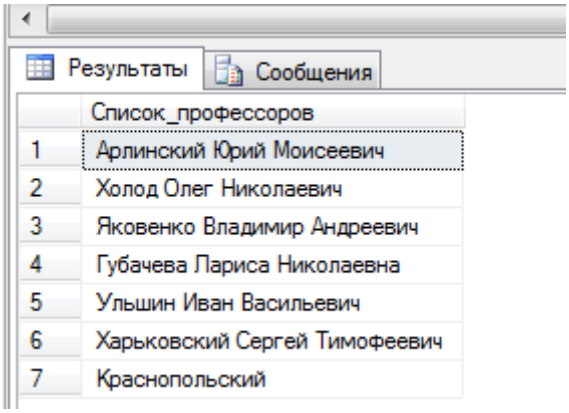
Особенностью операторов сравнения является то, что независимо от типов операндов их результатом являются логические значения. Предположим, вы хотите получить список всех профессоров.

Создайте новый запрос, введите sql-запрос, выполните его, сохраните его врабочую папку ЛАБ6_SQL под именем 1.sql.

Запрос 1. Вывести фамилии профессоров.

```
SELECT NAME_TEACHER AS ‘Список профессоров’FROM TEACHER
WHERE  DOLGNOST = 'профессор';
```

**Чтобы выполнить sql-команду нажмите на панели редактора кнопку  . В результате выполнения данного кода будут выданы все профессора. Например,**




Слово 'профессор' в запросе является строковой константой, поэтому ее следует заключить в кавычки. Обратите внимание, что мы указали фразу SELECT без ключевого слова DISTINCT, так как тогда от нас была бы скрыта информация о существовании среди профессоров однофамильцев. Чтобы при выводе результирующий столбец имел содержательный заголовок, мы поименовали егокак Список профессоров.

Это первый пример использования предиката над строковым типом данных. Здесь столбец строкового типа сравнивается со строковой константой. Запрос выполнен правильно, однако нужно всегда помнить о том, что предикаты над строками являются чувствительными к регистру букв. Например, предикат 'ИВАНОВ' = 'Иванов' будет ложным. Поэтому, если для некоторого профессора его должность была введена в таблицу TEACHER как 'Профессор', он не будет найден по условию WHERE DOLGNOST = 'профессор'.

Чтобы на предикаты над строками не влиял регистр букв, нужно использовать обычно имеющиеся в СУБД функции преобразования букв в прописные и строчные. В стандарте SQL, например, указаны функции UPPER и LOWER, выполняющие такие преобразования. Следовательно, для предыдущего запроса правильней будет записать условие фразы WHERE одним из следующих способов:

WHERE LOWER(DOLGNOST) = 'профессор' WHERE UPPER(DOLGNOST) = 'ПРОФЕССОР'

Самостоятельно измените в исходном запросе строку условия с использованием функции изменения регистра.

Чтобы сохранить запрос нажмите правую кнопу и из контекстного менювыберите Сохранить в файл. Присвойте имя 1.sql и сохраните в папку.

Запрос 2. **Найти всех студентов с стипендией, превышающим 300 грн. В sql-редакторе создайте новый запрос и введите следующий код: SELECT SUTNAME, SUTFNAME**

**FROM STUDENT WHERE STIPEND > 300;**

Выполните его.

Приведем несколько примеров использования операторов сравнения для столбцов строкового и временного типа. Обратите внимание, что сравнивать можно не только значение столбца с константой, но и значения столбцов между собой.

**Запрос 3. Вывести фамилии и должности преподавателей, принятых на работу после 01.01.2002.**

**SELECT NAME_TEACHER AS 'Фамилия', DOLGNOST AS 'Должность'FROM TEACHER**

**WHERE DATA_HIRE > '1/01/2002';**

**Запрос 4. Вывести фамилии и должности преподавателей, фамилии которых в**

**алфавитном порядке располагаются после фамилии Сычева.**

```
SELECT NAME_TEACHER, DOLGNOSTFROM      TEACHER
WHERE  UPPER(NAME_TEACHER) > 'Сычева';
```

Самостоятельно создайте запрос 5**. Вывести фамилии преподавателей, укоторых надбавка меньше ставки в 2,5 и более раз.**

Логические операторы

Операндами и результатом логических операторов являются логические значения. Стандартными логическими операторами являются **AND, OR** и **NOT**. Их действие показано в так называемых истинностных таблицах. Использование операторов сравнения вместе с логическими операторами предоставляет возможность

формулировать составные условия для отбора строк таблиц.

**Использование логического оператора AND**

Логический оператор **AND** во многих случаях действует как связка ‘и’ в русском языке. Рассмотрим несколько примеров с использованием этого оператора.

**Запрос 6. Вывести фамилии студентов, проживающих в городе Макарово и имеющих стипендию больше 100 грн.**

```
SELECT SUTFNAMEFROM      STUDENT
WHERE  CITY = 'Макарово' AND STIPEND >100;
```

**Самостоятельно создайте запрос 7.** Вывести фамилии преподавателей, которые являются профессорами и ставка которых превышает 4500.

**Самостоятельно создайте запрос 8.** Вывести фамилии студентов учащихся на кафедре под порядковым номером 2 (Прикладная математика) с стипендией в диапазоне 100-500 грн.

**Использование логического оператора OR**

Логический оператор OR во многих случаях действует как связка ‘или’ в русском языке. Рассмотрим несколько примеров.

**Запрос 9. Вывести названия кафедр, расположенных либо в 1 либо в 8 корпусе.**

```
SELECT NAME_KAFEDRU, NUM_KORPUSAFROMKAFEDRA
WHERE NUM_KORPUSA =1 OR NUM_KORPUSA =8;
```

Использование логического оператора NOT

Логический оператор NOT в русском языке передается словами ‘не’ и ‘кроме’.

Запрос 10. Вывести названия всех факультетов, кроме факультета математики и информатики.

```
SELECT NAME_FACULTETA FROM      FACULTET
WHERE      NOT LOWER(NAME_FACULTETA)  =      'математики и
информатики';
```

Обратите внимание, что оператор NOT должен предшествовать выражению сравнения, а не ставиться перед оператором сравнения. То есть запись **LOWER(NAME_FACULTETA) NOT ="математики и информатики"** будет неверной. Учитывая, что отрицанием оператора = является оператор <>**,** вместо указанного условия можно было бы записать **LOWER(NAME_FACULTETA) <> "математики и информатики"**. Это относится ко всем операторам сравнения, так как каждый из них имеет оператор, являющийся его отрицанием.

**Комбинирование логических операторов**

Логические операторы можно объединять, формируя составные условия. Возможность комбинирования обеспечивается тем, что любой логический оператор возвращает истинностное значение, а значит, его результат может использоваться в другом логическом операторе. Рассмотрим несколько примеров.

**Запрос 11. Вывести фамилии, должность, ставку и надбавку ассистентов, у которых либо ставка меньше 550, либо надбавка больше 60.**

```
SELECT NAME_TEACHER, DOLGNOST, Salary, RiseFROM  TEACHER
WHERE LOWER(DOLGNOST) ='ассистент' AND
(Salary < 550 OR Rise > 60);
```

Использование выражений над столбцами

До сих пор в выражениях фразы **WHERE** мы использовали в качестве значения одного или обоих аргументов имена столбцов. Однако аргументами могут быть и выражения над столбцами.

**Запрос 12. Показать фамилии преподавателей, чья зарплата (ставка плюс надбавка) превышает 3500.**

```
SELECT NAME_TEACHER AS ‘Фамилия преподавателя’, Salary + Rise AS ‘Его зарплата’
FROM  TEACHER
WHERE Salary + Rise > 3500;
```

**Запрос 13. Показать фамилии преподавателей, половина зарплаты которых превышает пятикратную надбавку.**

```
SELECT NAME_TEACHERFROM        TEACHER
WHERE (Salary + Rise) / 2 > 5 * Rise;
```

**Использование специальных операторов**

В SQL имеются операторы сравнения, позволяющие проверять значения столбцов и выражений над ними на соответствие некоторым специальным условиям:

- **принадлежность множеству;**
- **принадлежность диапазону;**
- **соответствие шаблону;**
- **соответствие регулярному выражению;**
- **неопределенное значение.**

В этом разделе вы узнаете, как их использовать и как с их помощью создавать составные условия.

**Проверка на принадлежность множеству**

Оператор **IN** позволяет проверить, входит ли значение в указанное множество значений. В простейшем случае этот оператор имеет следующий синтаксис:

**имя_столбца [NOT] IN (список_значений)**

Здесь список значений представляет собой перечень разделенных запятыми констант, тип которых должен соответствовать типу столбца, чье имя приведено слева. Семантика этого предиката такова: он принимает значение TRUE, если значение столбца соответствует одной из констант списка. Приведем пример.

**Запрос 14. Вывести названия и номер корпуса кафедр, расположенных в корпусах 1, 3, 12.**

```
SELECT Name_Kafedru, NUM_KORPUSAFROM       KAFEDRA
WHERE NUM_KORPUSA IN ('1', '3', '12');
```

**Использование отрицания**

Так как предикат IN возвращает истинностное значение, к нему можно применить логическое отрицание. Для этого следует воспользоваться нотацией NOT IN. В этомслучае предикат будет истинным, если значение столбца не входит в указанный список.

**Запрос 15. Вывести названия и номер корпуса кафедр, расположенных в любых корпусах, кроме 1, 3, или 12.**

```
SELECT Name_Kafedru AS ‘Название кафедры’,NUM_KORPUSA AS “Корпус’
FROM        KAFEDRA
WHERE  NUM_KORPUSA NOT IN (‘1', '3', '12');
```

**Использование выражений над столбцами**

В левой части оператора **IN** вместо имени столбца можно использовать любое допустимое над столбцами таблицы выражение языка.

**Запрос 16. Вывести фамилии преподавателей, зарплата которых (ставка + надбавка) равна 800, 900, 1000, 1100 или 1200.**

**SELECT NAME_TEACHER AS ‘Фамилия преподавателя’, Salary + Rise AS ‘Зарплата преподавателя’**

**FROM TEACHER**

**WHERE Salary + Rise IN (1150, 2400, 3150, 4300);**

Более того, элементами списка в правой части оператора **IN** тоже могут быть выражения над столбцами, как это показано в следующем примере:

**Запрос 17**.

**SELECT NAME_TEACHER, Salary, Salary + RiseFROM TEACHER**

**WHERE Salary + Rise IN (Salary + 100, Salary + 200, Salary + 300, Salary + 400, Salary + 500);**

Проверка на принадлежность диапазону значений

Еще одной формой проверки вхождения элемента во множество является проверка на его принадлежность диапазону значений. Для этого применяется предикат **BETWEEN**, который определяет нахождение значения столбца между указанными минимальным и максимальным значениями. Синтаксис предиката следующий:

**имя_столбца [NOT] BETWEEN минимум AND максимум**

Проверять можно значения числовых, строковых и временных типов (для строк символов предполагается алфавитное упорядочение). Оператор **BETWEEN** является включающим - это означает, что крайние значения диапазона включаются в допустимые.

**Запрос 18. Вывести фамилии преподавателей со ставкой в диапазоне 1000** 2000.

**SELECT NAME_TEACHERFROM TEACHER**

**WHERE Salary BETWEEN 1000 AND 2000;**

**Использование строковых значений**

Использование в операторе **BETWEEN** в качестве границ диапазона строковых значений имеет особенности, связанные с упорядочением (это же относится и к другим операторам сравнения).

**Запрос 19. Вывести фамилии преподавателей, начинающиеся на буквы от ‘З’ до ‘Л’.**

**SELECT NAME_TEACHER**FROM TEACHER

**WHERE UPPER(NAME_TEACHER) BETWEEN 'З' AND 'Л';**

Среди строк результата нет фамилий, начинающихся на букву ‘Л’. Дело в том, что при сравнении строк символов разной длины SQL предварительно дополняет более короткую строку символами пробела, а он в упорядочениях символов предшествует всем остальным. Поэтому строка, состоящая из буквы ‘Л’ (дополненная пробелами), всегда будет меньше любой другой строки, в которой за начальной буквой 'Л’ следуют отличающиеся от пробела символы.

Чтобы это учесть, в качестве верхнего значения диапазона лучше всего указывать следующую по алфавиту букву (в данном случае — ‘М’).

**Использование отрицания**

Так как предикат **BETWEEN** возвращает истинностное значение, к нему можно применить логическое отрицание. Для этого следует воспользоваться нотацией **NOT BETWEEN**, в которой предикат будет истинным, только если значение столбца не входит в указанный диапазон. Представление отрицания нотацией NOT BETWEEN введено в язык для большей наглядности, так как с предикатом BETWEEN можно стандартным образом использовать логический оператор NOT (то есть ставить отрицание ко всему выражению, а не к предикату):

**NOT (имя_столбца BETWEEN минимум AND максимум)**

Круглые скобки в данном случае можно и опустить, так как они не меняют порядка исполнения операторов. В нотации NOT BETWEEN крайние значения в диапазон не включаются.

**Запрос 20. Вывести названия и номер корпуса кафедр, которые не расположены в корпусах 1 и 3.**

```
SELECT Name_Kafedru, NUM_KORPUSAFROM KAFEDRA
WHERE NUM_KORPUSA NOT BETWEEN ‘1’ AND '3';

SELECT Name_Kafedru, NUM_KORPUSAFROM KAFEDRA
WHERE NOT (NUM_KORPUSA BETWEEN ‘1’ AND '3');
```

**Использование выражений над столбцами**

Как и в предикате IN, вместо имени столбца и границ диапазона можно использовать любое допустимое в языке выражение над столбцами таблицы, включая и функции.

**Запрос 21. Показать фамилии преподавателей, принятых на работу между 01.01.2000 и 12.12.2001.**

```
SELECT NAME_TEACHER, DATA_HIREFROM TEACHER
WHERE DATA_HIRE BETWEEN ‘01/01/2000’ AND '12/12/2001';
```

**Запрос 22. Вывести данные преподавателей, зарплата которых (ставка + надбавка) находится в диапазоне от удвоенной величины надбавки до утроенной надбавки плюс 50.**

```
SELECT NAME_TEACHER, Salary + Rise, 2 * Rise, 3 * Rise + 50
FROM TEACHER
WHERE Salary + Rise BETWEEN 2 * Rise AND 3 * Rise + 50;
```

Проверка на соответствие шаблону

Когда необходимо отобрать строки таблицы, в которых значение некоторого столбца совпадает с заданной строкой символов, следует использовать обычное сравнение, как это показано выше. Однако во многих случаях можно не знать точное представление в базе данных интересующего значения. Название одной и той же кафедры, например, может храниться в одном из следующих вариантов: ‘базы данных’, ‘организация баз данных’, ‘информационные системы и базы данных’, ‘базы данных и знаний’.

Такая же ситуация возникает, когда не известно точное написание фамилии преподавателя, название дисциплины, факультета и т. п. Специально для таких случаев предназначен оператор сравнения **LIKE**, позволяющий отобрать из таблицы строки на основе частичного соответствия. Упрощенный синтаксис оператора следующий:

**имя_столбца [NOT] LIKE шаблон [ESCAPE символ_пропуска]**

Его можно использовать только с символьными значениями. Использование шаблона

Оператор **LIKE** сравнивает значение столбца с множеством значений,

определяемых **шаблоном**. Он представляет собой строку, в которой помимо обычных символов, составляющих основу поискового выражения, можно использовать так называемые **подстановочные символы** (иногда они называются групповыми символами). Имеется всего два подстановочных символа, различающихся тем, что именно на их месте может стоять:

**%** — любая последовательность символов, включая их отсутствие;

**_** — один любой символ.

Подстановочные символы могут находиться в любом месте шаблона в любом наборе.

Например, шаблону **'%Иван%'** соответствуют строки 'Иван'**, 'Иванов',**

**'Иванченко', 'Петр Иванович'**, а шаблону **'л_с_' - 'лист', 'леса', 'лоск' (ноне 'лес', 'листок', 'плес').**

**Оператор LIKE**, как и все другие, работающие с символьными строками, чувствителен к регистру букв, поэтому при его использовании мы рекомендуем использовать уже известные вам функции **UPPER()** и **L0WER().**

**Запрос 23. Найти фамилии преподавателей на букву 'М'. SELECT NAME_TEACHER**
**FROM TEACHER**
**WHERE UPPER(NAME_TEACHER) LIKE 'М%';**

Имейте в виду, что если вы запишете условие фразы WHERE как
**UPPER(NAME_TEACHER) = 'М%' или даже как 'М%' LIKE UPPER(NAME_TEACHER)**, фамилии преподавателей будут сравниваться со строкой ' М%'. Во втором случае выражение является синтаксически правильным оператором **LIKE**, однако в нем строка 'М%' не выступает в качестве шаблона, так как расположена перед ключевым словом **LIKE**.

**Запрос 24. Указать преподавателей, в фамилиях которых первой буквой является 'М', а четвертой – 'ы'.**
**SELECT NAME_TEACHERFROM TEACHER**
**WHERE NAME_TEACHER LIKE 'М_____ы%';**

**Запрос 25. Вывести названия кафедр, в которых присутствует словосочетание 'анализ' (в различных грамматических формах).**
**SELECT Name_KafedruFROM KAFEDRA**
**WHERE LOWER(Name_Kafedru) LIKE '%анализ%';**

В левой части оператора **LIKE** может находиться не только имя столбца, но и любое допустимое над столбцами выражение, как это показано в следующем примере.

**Запрос 26. Указать преподавателей, в фамилию и название должности которых входит в сумме не меньше пяти букв 'о'.**

**SELECT NAME_TEACHER, DOLGNOST FROMTEACHER** WHERE LOWER(NAME_TEACHER+ DOLGNOST) LIKE
**'%о%о%о%о%о%';**

**Проверка на неопределенное значение**

Как мы уже отмечали, наличие значения NULL во фразе WHERE приводит к тому, что условие принимает истинностное значение UNKNOWN и соответствующая строка не включается в результат. Детальное описание работы с неопределенным значением вы можете найти в уроке 10, а здесь мы покажем, как обрабатывать значение NULL во фразе WHERE.

Чтобы проверить столбец на неопределенное значение, следует применитьунарный оператор **IS NULL**, имеющий такой синтаксис:

**имя_столбца IS [NOT] NULL**

Этот оператор принимает истинностное значение TRUE, если столбец имеет неопределенное значение, и FALSE — в противном случае. В нотации **IS NOT NULL** его действие обратное.

**Запрос 27. Вывести фамилии преподавателей, у которых не задан номер телефона или идентификационный код.**
**SELECT NAME_TEACHER, INDEF_KOD, TEL_TEACHERFROM TEACHER**
**WHERE INDEF_KOD IS NULL OR TEL_TEACHER IS NULL;**

Для созданной базы данных, согласно номеру варианта, самостоятельно создать

**наязыке Transact-SQL 15 запросов с отбором строк по условию:**

- 3 простейших запроса с использованием операторов сравнения;
- 3 запроса с использованием логических операторов AND, OR и NOT;
- 1 запрос на использование комбинации логических операторов;
- 1 запрос на использование выражений над столбцами;
- 2 запроса с проверкой на принадлежность множеству;
- 2 запроса с проверкой на принадлежность диапазону значений;
- 2 запроса с проверкой на соответствие шаблону;
- - 1 запрос с проверкой на неопределенное значение.
- Все программные инструкции команд SQL сохранять в файлах с расширением
- ***.sql** в папке ***ФИО_студента/Лаб5***.
- Для каждого запроса сформулировать текстовое задание, которое должно быть выполнено к базе данных.
- Создать текстовый отчет, в котором отобразить sql-команды разработанных запросов и скриншоты результатов работы из СУБД **SQL Server Management Studio**.

## Лабораторная работа № 5 Создание многотабличной базы данных. Установление взаимосвязей между таблицами (6 часов)

**Цели работы:** приобретение навыков применение инструментальных средств разработки и программирования объектов создаваемых баз данных.

**Инструкция CREATE TABLE (Transact-SQL)**

**Для создания таблиц** используется оператор "**CREATE TABLE**", который приводит к созданию **пустой** таблицы без строк. При создании таблиц задается имя таблицы, описание набора столбцов с их именами, типами и размерами, а также ограничения на хранящуюся в таблице информацию.

Имена таблиц в пределах базы данных должны быть уникальны.

Каждый столбец в таблице должен иметь имя, уникальное в пределах таблицы, а также либо тип данных, ограничения целостности, либо выражение для вычисления значения столбца.

**Пример создания таблицы без ограничений.**

Для выполнения данных запросов предварительно откройте **среду Micrpsoft SQL Server Managmant Studio**, выполните соединение с сервером и откройте базу данных **Educator**. Нажмите на панели инструментов команду **Создать запрос**.

**Пример 1. Создание родительской таблицы Товар без ограничений. use Educator**

```
CREATE TABLE Товар (КодТовара INT IDENTITY(1,1),
Название VARCHAR(50) , Цена MONEY ,
Тип VARCHAR(50) , Сорт VARCHAR(50) ,Город VARCHAR(50) ,
Остаток INT );
```

**Выполните sql-код**. Обновите базу данных и просмотрите созданную таблицу.

Сохраните sql-запрос под именем **Пример1.sql** в папке **ФИО_студента/Лаб4**.

Автоматическая генерация значения столбца **КодТовара** достигается за счет использования свойства **IDENTITY**, по умолчанию начальное значение, генерируемое с помощью **IDENTITY** равно 1, так же как и его приращение. Таким образом, следующее значение будет равно 2. Значения в **IDENTITY**-столбцах обязательно последовательные, то есть если приращение положительное, то следующее значение всегда больше предыдущего, если приращение отрицательное, то – всегда меньше. Приращение и начальное значение могут быть заданы, однако этот механизм чрезвычайно редко используется в реальных проектах.

**Создание ограничений**

В качестве ограничений используются ограничения столбца и ограничения

таблицы. Различие между ними в том, что ограничение столбца применяется только к определенному полю, а ограничение таблицы - к группам из одного или более полей.

***<ограничение_столбца>::=***

**[ CONSTRAINT имя_ограничения ]**
**{ [ NULL | NOT NULL ]**
**| [ {PRIMARY KEY | UNIQUE }**
**[ CLUSTERED | NONCLUSTERED ]**
**[ WITH FILLFACTOR=фактор_заполнения ] [ ON {имя_группы_файлов | DEFAULT } ] ] ]**
**| [ [ FOREIGN KEY ]**
**REFERENCES имя_род_таблицы[(имя_столбца_род_таблицы) ]**
**[ ON DELETE { CASCADE | NO ACTION } ][ ON UPDATE { CASCADE | NO ACTION } ][ NOT FOR REPLICATION ]]**
**| CHECK [ NOT FOR REPLICATION](<лог_выражение>) }**

***<ограничение_таблицы>::=***

**[CONSTRAINT имя_ограничения ]**
**{ [ {PRIMARY KEY | UNIQUE }**
**[ CLUSTERED | NONCLUSTERED ]**
**{(имя_столбца [ASC | DESC][,...n])}**
**[WITH FILLFACTOR=фактор_заполнения ] [ON {имя_группы_файлов | DEFAULT } ]]**
**|FOREIGN KEY[(имя_столбца [,...n])]**
REFERENCES имя_род_таблицы [(имя_столбца_род_таблицы [,...n])]
**[ ON DELETE { CASCADE | NO ACTION } ][ ON UPDATE { CASCADE | NO ACTION } ]**
**| NOT FOR REPLICATION ]**
**| CHECK [ NOT FOR REPLICATION ] (лог_выражение) }**

Рассмотрим отдельные параметры представленных конструкций, связанные с ограничениями целостности данных. Ограничения целостности имеют приоритет над триггерами, правилами и значениями по умолчанию. К ограничениям целостности относятся ограничение первичного ключа **PRIMARY KEY**, ограничение внешнего ключа **FOREIGN KEY**, ограничение уникальности **UNIQUE**, ограничение значения **NULL**, ограничение на проверку **CHECK**.

**Ограничение первичного ключа (PRIMARY KEY)**

Таблица обычно имеет столбец или комбинацию столбцов, значения которых уникально идентифицируют каждую строку в таблице. Этот столбец (или столбцы) называется **первичным ключом таблицы** и нужен для обеспечения ее **целостности**. Если в первичный ключ входит более одного столбца, то значения в пределах одного столбца могут дублироваться, но любая комбинация значений всех столбцов первичного ключа должна быть уникальна.

При создании первичного ключа SQL Server автоматически создает **уникальный индекс** для столбцов, входящих в первичный ключ. Он ускоряет доступ к данным этих столбцов при использовании первичного ключа в запросах.

Таблица может иметь только **одно ограничение PRIMARY KEY**, причем ни один из включенных в первичный ключ столбцов не может принимать значение **NULL**. При попытке использовать в качестве первичного ключа столбец (или группу столбцов), для

которого ограничения первичного ключа не выполняются, первичный ключ создан не будет, а система выдаст сообщение об ошибке.

Поскольку ограничение **PRIMARY KEY** гарантирует уникальность данных, оно часто определяется для столбцов-счетчиков. Создание ограничения целостности **PRIMARY KEY** возможно как при создании, так и при изменении таблицы. Одним из назначений первичного ключа является обеспечение ссылочной целостности данных нескольких таблиц. Естественно, это может быть реализовано только при определении соответствующих внешних ключей в других таблицах.

**Пример 2. Создание таблицы Товар с ограничением первичного ключа.**

```
CREATE TABLE Товар
(КодТовара INT IDENTITY(1,1) PRIMARY KEY,
Название VARCHAR(50)  ,Цена MONEY  ,
Тип VARCHAR(50)  , Сорт VARCHAR(50)  ,Город VARCHAR(50) ,
Остаток INT  );
```

***Примечание.*** Прежде чем выполнять sql-код удалите ранее созданную таблицу **Товар** из базы данных и обновите ее.

**Выполните sql-код**. Обновите базу данных и просмотрите созданную таблицу.

Сохраните sql-запрос под именем **Пример2.sql** в папке **ФИО_студента/Лаб4**.

**Первичные ключи более чем одного поля**

Ограничение **PRIMARY KEY** может также быть применено для многочисленных полей, составляющих уникальную комбинацию значений. Предположим что ваш первичный ключ - это имя, и вы имеете первое имя и последнее имя сохраненными в двух различных полях ( так что вы можете организовывать данныес помощью любого из них ). Очевидно, что ни первое ни последнее имя нельзя заставить быть уникальным самостоятельно, но мы можем каждую из этих двух комбинаций сделать уникальной.

Мы можем применить ограничение таблицы **PRIMARY KEY** для пар:

**Пример 3. Создание таблицы Сотрудники с ограничением первичного ключа.**

```
CREATE TABLE Сотрудники( Фамилия        char (10),
Имя                     char (10) ,
Город                   char (10),
PRIMARY KEY  ( Фамилия, Имя ));
```

**Выполните sql-код**. Обновите базу данных и просмотрите созданную таблицу.

Сохраните sql-запрос под именем **Пример3.sql** в папке **ФИО_студента/Лаб4**.

Одна проблема в этом подходе та, что мы можем вынудить появление уникальности - например, введя **Иванов Андрей** и **Иванов А.** Это может ввести в заблуждение, потому что ваши служащие могут не знать кто из них кто.

Обычно более надежный способ чтобы определять числовое поле которое могло бы отличать одну строку от другой, это иметь первичный ключ, и применять ограничение **UNIQUE** для двух имен полей.

**Использование ограничений для исключения пустых( NULL )указателей**

Вы можете использовать команду **CREATE TABLE** чтобы предохранить поле от разрешения в нем пустых (**NULL**) указателей с помощью ограничения **NOT NULL**. Это

ограничение накладывается только для разнообразных столбцов.

**NULL** - это специальное обозначение которое отмечает поле как пустое. **NULL** может быть полезен, когда имеются случаи, когда вы хотите быть от них гарантированы. Очевидно, что первичные ключи никогда не должны быть пустыми , поскольку это будет подрывать их функциональные возможности. Кроме того, такие поля как имена, требуют в большинстве случаев, определенных значений. Например, вы вероятно захотите иметь информацию о **должности** занимаемым каждым сотрудником в таблице **Сотрудники**.

Если вы поместите ключевые слова **NOT NULL** сразу после типа данных ( включая размер ) столбца, любая попытка поместить значение **NULL** в это поле будет отклонена. В противном случае, SQL понимает, что **NULL** раз решен.

Например, давайте улучшим наше определение таблицы Сотрудники, не позволяя помещать NULL значения в столбец **Должность** :

**Пример 4. Создание таблицы Сотрудники с ограничением пустых значений.**

```
CREATE TABLE Сотрудники( Фамилия   char (10),
Имя                      char (10) ,
Город                    char (10),
Должность ……char (10) NOT NULL,PRIMARY KEY  ( Фамилия, Имя ));
```

**Выполните sql-код**. Обновите базу данных и просмотрите созданную таблицу. Сохраните sql-запрос под именем **Пример4.sql** в папке **ФИО_студента/Лаб4**.

Важно помнить, что любому столбцу с ограничением **NOT NULL** должно быть установлено значение в каждом предложении **INSERT** воздействующем на таблицу. При отсутствии **NULL**, SQL может не иметь значений для установки в эти столбцы, если конечно значение по умолчанию, описанное ра нее в этой главе, уже не было назначено.

Если ваша система поддерживает использование **ALTER TABLE** чтобы добавлять новые столбцы к уже существующей таблице, вы можете вероятно помещать ограничение столбцов, типа **NOT NULL**, для этих новых столбцов. Однако, если вы предписываете новому столбцу значение **NOT NULL**, текущая таблица должна быть **пустой**!!!!.

**Использование ограничений для уникальности значений**

В предыдущей лаб.работе мы обсудили использование уникальных индексов чтобы зас тавить поля иметь различные значения для каждой строки. Эта практика - осталась с прежних времен, когда SQL поддерживал ограничение **UNIQUE**.

**Уникальность** - это свойство данных в таблице, и поэтому его более логично назвать как ограничение этих данных, а не просто как свойство логического отличия, связывающее объект данных ( индекс ).

Несомненно, уникальные индексы - один из самых простых и наиболее эффективных методов предписания уникальности. По этой причине, некото рые реализации ограничения **UNIQUE** используют уникальные индексы; то-есть они создают индекс не сообщая вам об этом. Остается фактом, что вероятность беспорядка в базе данных достаточно мала, если вы предписываете уникальность вместе с ограничением.

**Уникальность как ограничение столбца**

Время от времени, вы хотите убедиться, что все значения введеные в столбец отличаются друг от друга. Например, первичные ключи достаточно ясно это показывают. Если вы помещаете ограничение столбца **UNIQUE** в поле при создании таблицы, база данных отклонит любую попытку ввода в это поле для одной из строк, значения, которое уже представлено в другой строке. Это ограничение может применяться только к полям которые были обьявлены как непустые (**NOT NULL**), так как не имеет смысла позволить одной строке таблицы иметь значение **NULL**, а затем исключать другие строки с **NULL** значениями как дубликаты. Имеется дальнейшее

усовершенствование нашей команды создания таблицы Сотрудники :

**Пример 5. Создание таблицы Сотрудники с ограничением уникальности.**

```
CREATE TABLE  Сотрудники
( Фамилия                    char (10) NOT NULL UNIQUE,
Имя                          char (10) NOT NULL UNIQUE,
Город                        char (10),
…….Должность ……char (10) NOT NULL,PRIMARY KEY  ( Фамилия, Имя
));
```

**Выполните sql-код**. Обновите базу данных и просмотрите созданную таблицу. Сохраните sql-запрос под именем **Пример5.sql** в папке **ФИО_студента/Лаб4**.

Когда вы обьявляете поля **Фамилия** уникальным, убедитесь, что в вашей базе данных не будет двух Ивановых или Петровых. В то же время это не так уж необходимос функциональной точки зрения - потому что поле **Имя** в качестве первичного ключа, все равно обеспечит отличие этих двух строк - что проще для людей использующих данные в таблицах, чем помнить, что эти Ивановы не идентичны.

Столбцы ( не первичные ключи ) чьи значения требуют уникальности, называются **ключами-кандидатами** или **уникальными ключами**.

**Ограничение по умолчанию (DEFAULT)**

Столбцу может быть присвоено значение по умолчанию. Оно будет актуальным в том случае, если пользователь не введет в столбец никакого иного значения.

Отдельно необходимо отметить пользу от использования значений по умолчанию при добавлении нового столбца в таблицу. Если для добавляемого столбца не разрешено хранение значений **NULL** и не определено значение по умолчанию, то операция добавления столбца закончится неудачей.

Когда вы вставляете строку в таблицу без указания значений в ней для каждого поля, SQL должен иметь значение по умолчанию для включения его в определенное поле, или же команда будет отклонена. Наиболее общим значением по умолчанию является - **NULL**. Это - значение по умолчанию для любого столбца, которому не было дано ограничение **NOT NULL** или который имел другое назначение по умолчанию.

Значение **DEFAULT(ПО УМОЛЧАНИЮ)** указывается в команде **CREATE TABLE** тем же способом что и ограничение столбца, хотя, с технической точки зрения, значение **DEFAULT** не ограничительного свойства - оно не ограничивает значения которые вы можете вводить, а просто определяет, что может случиться если вы не введете любое из них.

Предположим, что вы работаете в г. Москва и подавляющее большинство ваших сотрудников живут в этом городе. Вы можете указать г. Москва в качестве значения поля **Город**, по умолчанию, для вашей таблицы **Сотрудников**:

**Пример 7. Создание таблицы Сотрудники с значением по умолчанию.**

```
CREATE TABLE  Сотрудники
( Фамилия                    char (10) NOT NULL UNIQUE,Имя      char      (10)
NOT NULL UNIQUE,
Город                        char (10) DEFAULT  'Москва',
…….Должность ……char (10) NOT NULL,PRIMARY KEY  ( Фамилия, Имя
));
```

**Выполните sql-код**. Обновите базу данных и просмотрите созданную таблицу.

Сохраните sql-запрос под именем **Пример7.sql** в папке **ФИО_студента/**

Конечно, вводить значение Москва в таблицу каждый раз когда назна чается новый сотрудник, не такая уж необходимость, и можно просто пре небречь им ( не вводя его ) даже если оно должно иметь некоторое зна чение.

Другой способ использовать значение по умолчанию - это использовать его как альтернативу для **NULL**. Так как **NULL** (фактически) неверен при любом сравнении, ином чем **IS NULL**, он может быть исключен с помощью большинства предикатов. Иногда, вам нужно видеть пустые значения ваших полей не обрабатывая их каким-то определенным образом. Вы можете уста-

новить значение по умолчанию, типа нуль или пробел, которые функцио нально меньше по значению чем просто не установленное значение - пус тое значение(**NULL**). Различие между ними и обычным **NULL** в том, что SQL будет обрабатывать их также как и любое другое значение.

**Ограничение проверочное (CHECK)**

Данное ограничение используется для проверки допустимости данных, вводимых в конкретный столбец таблицы, т.е. ограничение **CHECK** обеспечивает еще один уровень защиты данных.

Ограничения целостности **CHECK** задают диапазон возможных значений для столбца или столбцов. В основе ограничений целостности **CHECK** лежит использование логических выражений.

Допускается применение нескольких ограничений **CHECK** к одному и тому же столбцу. В этом случае они будут применимы в той последовательности, в которой происходило их создание. Возможно применение одного и того же ограничения к разным столбцам и использование в логических выражениях значений других столбцов.

**Проверка значений полей**

Конечно, имеется любое число ограничений, которые можно устанавливать для данных вводимых в ваши таблицы, чтобы видеть, например, находятся ли данные в соответствующем диапазоне или правильном формате, о чем SQL естественно не может знать заранее. По этой причине, SQL обеспечивает вас ограничением **CHECK**, которое позволяет вам установить условие которому должно удовлетворять значение вводимое в таблицу, прежде чем оно будет принято.

Ограничение **CHECK** состоит из ключевого слова **CHECK** сопровождаемого предложением предиката, который использует указанное поле. Любая попытка модифицировать или вставить значение поля которое могло бы сделать этот предикат неверным - будет отклонена.

Давайте рассмотрим таблицу **Продавцы**. Столбец комиссионных выражается десятичным числом и поэтому может быть умножен непосредственно на сумму приобретений в результате чего будет получена сумма комиссионных(в долларах) продавца с установленым справа значком доллара( $ ) . Кто-то может использовать понятие процента, однако ведь, можно об этом и не знать. Если человек введет по ошибке 14 вместо .14 чтобы указать в процентах свои комиссионные, это будет расценено как 14.0 , что является законным десятичным значением, и будет нормально воспринято системой. Чтобы предотвратить эту ошибку, мы можем наложить ограничение столбца - **CHECK** чтобы убедиться, что вводимое значение меньше чем 1.

**Пример 8. Создание таблицы Продавцы и с проверкой значений полей**

```
CREATE TABLE Продавцы
( КодПродавца          integer NOT NULL PRIMARY KEY,
Фамилия                char(10) NOT NULL UNIQUE,Город      char(10),
Комиссионные           decimal CHECK (Комиссионные < 1 ));
```

**Использование - CHECK, чтобы предопределять допустимое вводимое значение**

Мы можем также использовать ограничение **CHECK** чтобы защитить от ввода в поле определенных значений, и таким образом предотвратить ошибку.

Например, предположим, что единственными городами в которых мы имели ведомства сбыта являются Лондон, Барселона, Сан Хосе, и Нью Йорк. Если вам известны все продавцы работающие в каждом из этих ведомств, нет необходимости позволять ввод других значений. Если же нет, использова ние ограничения может предотвратить опечатки и другие ошибки.

**Пример 8. Создание таблицы Продавцы и с проверкой значений полей, чтобы предопределять допустимое вводимое значение**

**CREATE TABLE Продавцы**
**( КодПродавца integer NOT NULL PRIMARY KEY, Фамилия char(10) NOT NULL UNIQUE,**
**Город char(10)**
**CHECK (Город IN (' Лондон ', 'Барселона', ' Сан Хосе ', ' Нью Йорк ')), Комиссионные decimal CHECK (Комиссионные < 1 ));**

**Выполните sql-код**. Обновите базу данных и просмотрите созданную таблицу. Сохраните sql-запрос под именем **Пример8.sql** в папке **ФИО_студента/Лаб4**.

Конечно, если вы собираетесь сделать это, вы должны быть уверены что ваша компания не открыла уже новых других ведомств сбыта. Большинство программ баз данных поддерживают команду **ALTER TABLE** , которая позволяет вам изменять определение таблицы, даже когда она находится в использовании. Однако, изменение или удаление ограничений не всегда возможно для этих команд, даже там где это вроде бы поддерживается.

Если вы использовали систему, которая не может удалять ограничения, вы будете должны создавать (**CREATE**) новую таблицу и передавать информацию из старой таблицы в нее всякий раз, когда вы хотите изменить ограничение. Конечно же Вы не захотите делать это часто, и со временем вообще перестанете это делать.

**Проверка условий, базирующийся на многочисленых полях**

Вы можете также использовать **CHECK** в качестве табличного ограничения. Это полезно в тех случаях, когда вы хотите включить более одного поля строки в условие. Предположим что комиссионные 0.15 и выше, будут разрешены только для продавца из Барселоны. Вы можете указать это со следующим табличным ограничением **CHECK** :

**Пример 9. Создание таблицы Продавцы и с проверкой значений полей, базирующийся на многочисленых полях**

**CREATE TABLE Продавцы2**
**( КодПродавца integer NOT NULL PRIMARY KEY, Фамилия char(10) NOT NULL UNIQUE,**
**Город char(10) ,Комиссионные decimal,**
**CHECK (Комиссионные < 0.15 OR Город='Барселона'));**

**Выполните sql-код**. Обновите базу данных и просмотрите созданную таблицу. Сохраните sql-запрос под именем **Пример9.sql** в папке **ФИО_студента/**.

Как вы можете видеть, два различных поля должны быть проверены чтобы определить, верен предикат или нет. Имейте в виду, что это - два разных поля одной и той же строки. Хотя вы можете использовать многочис леные поля, SQL не может проверить более одной строки одновременно. Вы не можете, например использовать ограничение **CHECK** чтобы удостовериться что все комиссионные в данном городе одинаковы. Чтобы сделать это, SQL должен всякий раз просматривая другие строки таблицы, когда вы модифицируете или вставляете строку, видеть, что значение комиссионных указано для текущего города. SQL этого делать не умеет.

Фактически, вы могли бы использовать сложное ограничение **CHECK** для вышеупомянутого, если бы знали заранее, каковы должны быть комиссионные в разных городах.

**Самостоятельно** измените ограничение в примере 9 на следующее:

- Если комиссионные равны 0.15 , то будут разрешены только для продавца из Лондана
- Если комиссионные равны 0.14 , то будут разрешены только для продавца из Барселоны
- Если комиссионные равны 0.13 , то будут разрешены только для продавца из Сан-Хосе
- Если комиссионные равны 0.12 , то будут разрешены только для продавца из Нью-Йорка

Вы получили идею. Чем налагать такой комплекс ограничений, вы могли бы просто использовать представление с предложением **WITH CHECK OPTION,** которое имеет все эти условия в своем предикате. Пользователи могут обращаться к представлению таблицы вместо самой таблицы. Одним из преимуществ этого будет то, что процедура изменения в ограничении не будет такой болезненной или трудоемкой. Представление с **WITH CHECK OPTION** - хороший заменитель ограничению **CHECK**.

**Пример 10. Создание таблицы Клиент с ограничениями.**

```
CREATE TABLE  Клиент
(КодКлиента INT IDENTITY(1,1) PRIMARY KEY,Фирма    VARCHAR(50)
                NOT NULL, Фамилия    VARCHAR(50)       NOT NULL,
Город          VARCHAR(50)       NOT NULL,Телефон              CHAR(10)
               NOT NULL
CHECK (Телефон LIKE  '[1-9][0-9]-[0-9][0-9]-[0-9][0-9]'));
```

**Выполните sql-код**. Обновите базу данных и просмотрите созданную таблицу. Сохраните sql-запрос под именем **Пример10.sql** в папке **ФИО_студента/**

Ограничение внешнего ключа (FOREIGN KEY)

**Ограничение внешнего ключа** - это основной механизм для поддержания ссылочной целостности между таблицами реляционной базы данных.

**Столбец дочерней таблицы**, определенный в качестве **внешнего ключа** в параметре **FOREIGN KEY**, применяется для ссылки на **столбец родительской таблицы**, являющийся в ней первичным ключом.

Имя родительской таблицы и столбцы ее первичного ключа указываются в предложении **REFERENCES**.

Данные в столбцах, определенных в качестве внешнего ключа, могут принимать только такие же значения, какие находятся в связанных с ним столбцах первичногоключа родительской таблицы.

Совпадение имен столбцов для связи дочерней и родительской таблиц необязательно.

Первичный ключ может быть определен для столбца с одним именем, в то время как столбец, на который наложено ограничение **FOREIGN KEY**, может иметь совершенно другое имя. Единственным требованием остается соответствие столбцов по типу и размеру данных.

На первичный ключ могут ссылаться не только столбцы других таблиц, но и столбцы, расположенные в той же таблице, что и собственно первичный ключ; это позволяет создавать рекурсивные структуры.

Внешний ключ может быть связан не только с первичным ключом другой таблицы. Он может быть определен и для столбцов с ограничением **UNIQUE** второй таблицы или любых других столбцов, но таблицы должны находиться в одной базе данных.

Столбцы внешнего ключа **могут содержать значение NULL**, однако проверка на ограничение **FOREIGN KEY** игнорируется. Внешний ключ может быть проиндексирован, тогда сервер будет быстрее отыскивать нужные данные. Внешнийключ определяется как при создании, так и при изменении таблиц.

Ограничение ссылочной целостности задает требование, согласно которому для каждой записи в дочерней таблице должна иметься запись в родительской таблице. При этом изменение значения столбца связи в записи родительской таблицы при наличии дочерней записи блокируется, равно как и удаление родительской записи (запрет каскадного изменения и удаления), что гарантируется параметрами **ON DELETE NO ACTION и ON UPDATE NO ACTION**, принятыми по умолчанию. Для разрешения каскадного воздействия следует использовать параметры **ON DELETE CASCADE** и **ON UPDATE CASCADE**.

Если пользователь предпринимает попытку удалить из родительской таблицы строку, на которую ссылается одна или более строк дочерней таблицы, язык SQL предоставляет следующие возможности:

**CASCADE** - выполняется удаление строки из родительской таблицы, сопровождающееся автоматическим удалением всех ссылающихся на нее строк дочерней таблицы;

**SET NULL** - выполняется удаление строки из родительской таблицы, а во внешние ключи всех ссылающихся на нее строк дочерней таблицы записывается значение NULL;

**SET DEFAULT** - выполняется удаление строки из родительской таблицы, а во внешние ключи всех ссылающихся на нее строк дочерней таблицы заносится значение, принимаемое по умолчанию;

**NO ACTION** - операция удаления строки из родительской таблицы отменяется. Именно это значение используется по умолчанию в тех случаях, когда в описании внешнего ключа фраза ON DELETE опущена.

Те же самые правила применяются в языке SQL и тогда, когда значение потенциального ключа родительской таблицы обновляется.

Определитель **MATCH** позволяет уточнить способ обработки значения **NULL** во внешнем ключе.

При определении таблицы предложение **FOREIGN KEY** может указываться произвольное количество раз.

В операторе **CREATE TABLE** используется необязательная фраза **DEFAULT**, которая предназначена для задания принимаемого по умолчанию значения, когда в операторе **INSERT** значение в данном столбце будет отсутствовать.

Фраза **CONSTRAINT** позволяет задать имя ограничению, что позволит впоследствии отменить то или иное ограничение с помощью оператора **ALTER TABLE**.

**Пример 11. Создание таблицы Склад с ограничениями первичного ключа и внешнего ключа.**

**CREATE TABLE Сделка**
**(КодСделки INT IDENTITY(1,1) PRIMARY KEY,**
**КодТовара INT NOT NULL, КодКлиента INT NOT NULL,**
**Количество INT NOT NULL DEFAULT 0,**
**Дата DATETIME NOT NULL DEFAULT GETDATE(),**
**Остаток INT,**
**CONSTRAINT fk_Товар FOREIGN KEY(КодТовара) REFERENCES Товар, CONSTRAINT fk_Клиент FOREIGN KEY(КодКлиента) REFERENCES**
**Клиент);**

**Выполните sql-код**. Обновите базу данных и просмотрите созданную таблицу. Сохраните sql-запрос под именем **Пример11.sql** в папке **ФИО_студента/Лаб4**.

**Изменение таблиц**

Для внесения изменений в уже созданные таблицы стандартом SQL предусмотрен оператор **ALTER TABLE**, предназначенный для выполнения следующих действий:

- добавление в таблицу нового столбца;
- удаление столбца из таблицы;
- добавление в определение таблицы нового ограничения;
- удаление из определения таблицы существующего ограничения;
- задание для столбца значения по умолчанию;
- отмена для столбца значения по умолчанию.

**Оператор изменения таблицы имеет следующий обобщенный формат**:

***<изменение_таблицы> ::=***
**ALTER TABLE имя_таблицы**
**[ADD [COLUMN]имя_столбца тип_данных[ NOT NULL ][UNIQUE]**
**[DEFAULT <значение>][ CHECK (<условие_выбора>)]]**
**[DROP [COLUMN] имя_столбца [RESTRICT | CASCADE ]]**
**[ADD [CONSTRAINT [имя_ограничения]] [{PRIMARY KEY (имя_столбца [,...n])**
**|[UNIQUE (имя_столбца [,...n])}**
**|[FOREIGN KEY (имя_столбца_внешнего_ключа [,...n])**
**REFERENCES имя_род_таблицы [(имя_столбца_род_таблицы [,...n])],**
**[ MATCH {PARTIAL | FULL}**
**[ON UPDATE {CASCADE| SET NULL |SET DEFAULT | NO ACTION}]**
**[ON DELETE {CASCADE| SET NULL |SET DEFAULT | NO ACTION}]**
**|[CHECK(<условие_выбора>)][,...n]}] [DROP CONSTRAINT имя_ограничения**
**[RESTRICT | CASCADE]]**
**[ALTER [COLUMN] SET DEFAULT <значение>] [ALTER [COLUMN] DROP DEFAULT]**

Здесь параметры имеют то же самое назначение, что и в определении оператора
**CREATE TABLE**.
Оператор **ALTER TABLE** реализован не во всех диалектах языка SQL. В некоторых диалектах он поддерживается, однако не позволяет удалять из таблицы уже существующие столбцы.

В дополнение к уже названным параметрам определим параметр **{ENABLE | DISABLE } TRIGGER ALL_,** предписывающий задействовать или отключить конкретный триггер или все триггера, связанные с таблицей.

Создать **на языке Transact-SQL** файл базы данных согласно номеру варианта (присвоить ей новое имя, несовпадающие с именем базы данных созданной в лаб.№3). База данных разрабатывается на основе спроектированной концептуальной модели данных в лаб.№1.

**Создать программно на языке SQL** все таблицы, с указанием первичных и внешних ключей и ограничения целостности.

Все программные инструкции команд SQL сохранять в файлах с расширением ***.sql** в папке ***ФИО_студента/Лаб4***.

Заполнить таблицы данными по 5 записей в каждой.

Создать текстовый отчет, в котором отобразить sql-команды разработанных запросов и скриншоты результатов работы из СУБД **SQL Server Management Studio**

**Самостоятельно** заполните вручную данными таблицы **Студент** и **Преподаватель** согласно рис. 1-2, приведенным ниже.

Также ранее должны были введены следующие данные:

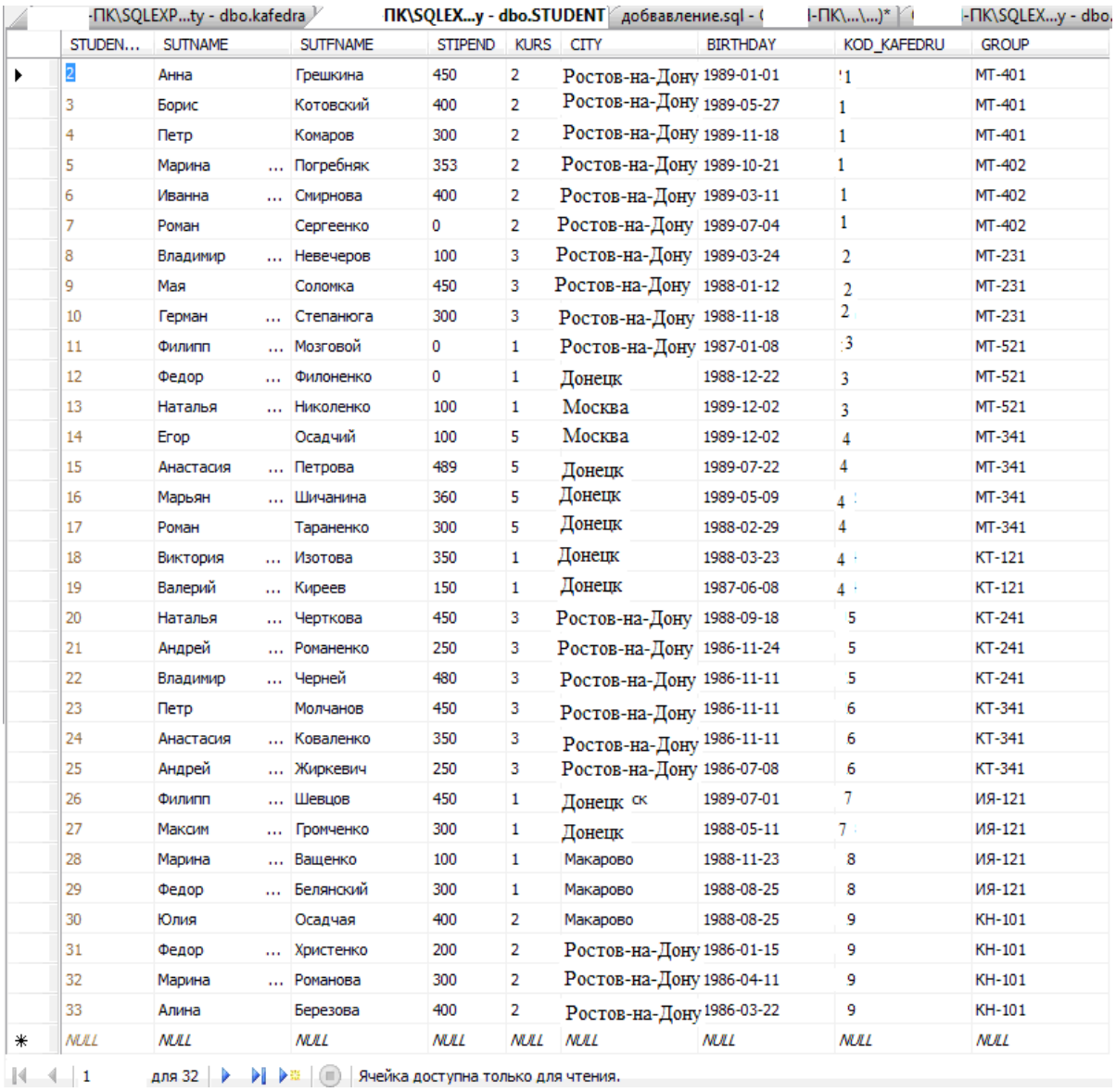
I-ПК\SQLEXP... - dbo.Facultet* | I-ПК\SQLEXPR...y - dbo.Facultet

| kod_faculteta | Name_faculteta | Fio_Decana | Nomer_komnatu | Tel_decan... |
|---|---|---|---|---|
| 1 | Математики и информатики | Статывка Юрий Иванович | 31/a | 417499 |
| 2 | Компьютерных систем и технологий | Губачева Лариса Александровна | 204/12 | 477051 |
| 3 | Международный | Харченко Евгений Иванович | 310/9 | 500830 |
| NULL | NULL | NULL | NULL | NULL |

\COMP\SQLEX...15 - dbo.Kafedra

| Kod_kafe... | kod_faculteta | Name_kafedru | Fio_zavkaf | Nomer_ko... | num_korpusa | Tel_kafedru |
|---|---|---|---|---|---|---|
| 1 | 1 | Компьютерные системы и сети | Соловьев | 414 | 1 | 899028 |
| 2 | 1 | Прикладная математика | Кочевский | 205 | 1 | 425262 |
| 3 | 1 | Математического анализа | Арлинский | 61 | 1 | 627272 |
| 4 | 1 | Информатики | Пожидаев | 404 | 1 | 738328 |
| 5 | 2 | Автоматизации компьютерных технологий | Малахов | 123 | 12 | 637327 |
| 6 | 2 | Компьютерных технологий на промышленном транспорте | Губачева | 342 | 12 | 373728 |
| 7 | 2 | Системная инженерия | Ульшин | 234 | 12 | 727287 |
| 8 | 3 | Иностранных языков | Краснопольский | 123 | 2 | 762728 |
| 9 | 3 | Компьютерных наук | Дядечев | 703 | 9 | 563272 |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL |

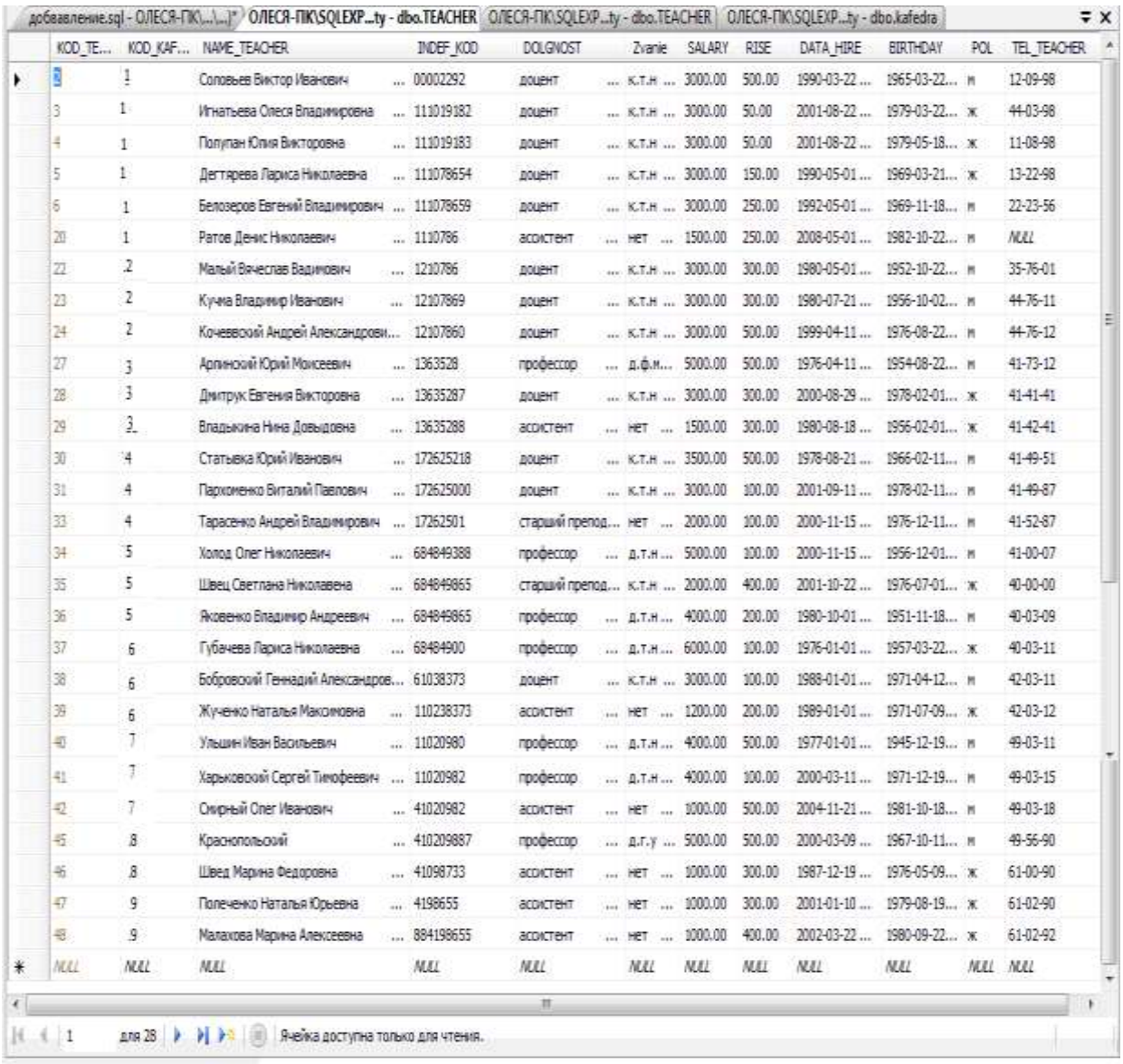
**Новые данные ввести вручную.**



-ПК\SQLEXP...ty - dbo.kafedra | ПК\SQLEX...y - dbo.STUDENT | добвавление.sql - ( | -ПК\...\...)* | -ПК\SQLEX...y - dbo.

| STUDEN... | SUTNAME | SUTFNAME | STIPEND | KURS | CITY | BIRTHDAY | KOD_KAFEDRU | GROUP |
|---|---|---|---|---|---|---|---|---|
| 2 | Анна | Грешкина | 450 | 2 | Ростов-на-Дону | 1989-01-01 | 1 | МТ-401 |
| 3 | Борис | Котовский | 400 | 2 | Ростов-на-Дону | 1989-05-27 | 1 | МТ-401 |
| 4 | Петр | Комаров | 300 | 2 | Ростов-на-Дону | 1989-11-18 | 1 | МТ-401 |
| 5 | Марина ... | Погребняк | 353 | 2 | Ростов-на-Дону | 1989-10-21 | 1 | МТ-402 |
| 6 | Иванна ... | Смирнова | 400 | 2 | Ростов-на-Дону | 1989-03-11 | 1 | МТ-402 |
| 7 | Роман | Сергеенко | 0 | 2 | Ростов-на-Дону | 1989-07-04 | 1 | МТ-402 |
| 8 | Владимир ... | Невечеров | 100 | 3 | Ростов-на-Дону | 1989-03-24 | 2 | МТ-231 |
| 9 | Мая | Соломка | 450 | 3 | Ростов-на-Дону | 1988-01-12 | 2 | МТ-231 |
| 10 | Герман ... | Степанюга | 300 | 3 | Ростов-на-Дону | 1988-11-18 | 2 | МТ-231 |
| 11 | Филипп ... | Мозговой | 0 | 1 | Ростов-на-Дону | 1987-01-08 | 3 | МТ-521 |
| 12 | Федор ... | Филоненко | 0 | 1 | Донецк | 1988-12-22 | 3 | МТ-521 |
| 13 | Наталья ... | Николенко | 100 | 1 | Москва | 1989-12-02 | 3 | МТ-521 |
| 14 | Егор | Осадчий | 100 | 5 | Москва | 1989-12-02 | 4 | МТ-341 |
| 15 | Анастасия ... | Петрова | 489 | 5 | Донецк | 1989-07-22 | 4 | МТ-341 |
| 16 | Марьян ... | Шичанина | 360 | 5 | Донецк | 1989-05-09 | 4 | МТ-341 |
| 17 | Роман | Тараненко | 300 | 5 | Донецк | 1988-02-29 | 4 | МТ-341 |
| 18 | Виктория ... | Изотова | 350 | 1 | Донецк | 1988-03-23 | 4 | КТ-121 |
| 19 | Валерий ... | Киреев | 150 | 1 | Донецк | 1987-06-08 | 4 | КТ-121 |
| 20 | Наталья ... | Черткова | 450 | 3 | Ростов-на-Дону | 1988-09-18 | 5 | КТ-241 |
| 21 | Андрей ... | Романенко | 250 | 3 | Ростов-на-Дону | 1986-11-24 | 5 | КТ-241 |
| 22 | Владимир ... | Черней | 480 | 3 | Ростов-на-Дону | 1986-11-11 | 5 | КТ-241 |
| 23 | Петр | Молчанов | 450 | 3 | Ростов-на-Дону | 1986-11-11 | 6 | КТ-341 |
| 24 | Анастасия ... | Коваленко | 350 | 3 | Ростов-на-Дону | 1986-11-11 | 6 | КТ-341 |
| 25 | Андрей ... | Жиркевич | 250 | 3 | Ростов-на-Дону | 1986-07-08 | 6 | КТ-341 |
| 26 | Филипп ... | Шевцов | 450 | 1 | Донецк ск | 1989-07-01 | 7 | ИЯ-121 |
| 27 | Максим ... | Громченко | 300 | 1 | Донецк | 1988-05-11 | 7 | ИЯ-121 |
| 28 | Марина ... | Ващенко | 100 | 1 | Макарово | 1988-11-23 | 8 | ИЯ-121 |
| 29 | Федор ... | Белянский | 300 | 1 | Макарово | 1988-08-25 | 8 | ИЯ-121 |
| 30 | Юлия | Осадчая | 400 | 2 | Макарово | 1988-08-25 | 9 | КН-101 |
| 31 | Федор ... | Христенко | 200 | 2 | Ростов-на-Дону | 1986-01-15 | 9 | КН-101 |
| 32 | Марина ... | Романова | 300 | 2 | Ростов-на-Дону | 1986-04-11 | 9 | КН-101 |
| 33 | Алина | Березова | 400 | 2 | Ростов-на-Дону | 1986-03-22 | 9 | КН-101 |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

1 для 32 | Ячейка доступна только для чтения.



добавление.sql - ОЛЕСЯ-ПК\...\...)* | ОЛЕСЯ-ПК\SQLEXP...ty - dbo.TEACHER | ОЛЕСЯ-ПК\SQLEXP...ty - dbo.TEACHER | ОЛЕСЯ-ПК\SQLEXP...ty - dbo.kafedra

| | KOD_TE... | KOD_KAF... | NAME_TEACHER | INDEF_KOD | DOLGNOST | Zvanie | SALARY | RISE | DATA_HIRE | BIRTHDAY | POL | TEL_TEACHER |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ▶ | 2 | 1 | Соловьев Виктор Иванович | 00002292 | доцент | к.т.н | 3000.00 | 500.00 | 1990-03-22 | 1965-03-22 | м | 12-09-98 |
| | 3 | 1 | Игнатьева Олеся Владимировна | 111019182 | доцент | к.т.н | 3000.00 | 50.00 | 2001-08-22 | 1979-03-22 | ж | 44-03-98 |
| | 4 | 1 | Полупан Юлия Викторовна | 111019183 | доцент | к.т.н | 3000.00 | 50.00 | 2001-08-22 | 1979-05-18 | ж | 11-08-98 |
| | 5 | 1 | Дегтярева Лариса Николаевна | 111078654 | доцент | к.т.н | 3000.00 | 150.00 | 1990-05-01 | 1969-03-21 | ж | 13-22-98 |
| | 6 | 1 | Белозеров Евгений Владимирович | 111078659 | доцент | к.т.н | 3000.00 | 250.00 | 1992-05-01 | 1969-11-18 | м | 22-23-56 |
| | 20 | 1 | Ратов Денис Николаевич | 1110786 | ассистент | нет | 1500.00 | 250.00 | 2008-05-01 | 1982-10-22 | м | NULL |
| | 22 | 2 | Малый Вячеслав Вадимович | 1210786 | доцент | к.т.н | 3000.00 | 300.00 | 1980-05-01 | 1952-10-22 | м | 35-76-01 |
| | 23 | 2 | Кучма Владимир Иванович | 12107869 | доцент | к.т.н | 3000.00 | 300.00 | 1980-07-21 | 1956-10-02 | м | 44-76-11 |
| | 24 | 2 | Кочевский Андрей Александрови | 12107860 | доцент | к.т.н | 3000.00 | 500.00 | 1999-04-11 | 1976-08-22 | м | 44-76-12 |
| | 27 | 3 | Арлинский Юрий Моисеевич | 1363528 | профессор | д.ф.м | 5000.00 | 500.00 | 1976-04-11 | 1954-08-22 | м | 41-73-12 |
| | 28 | 3 | Дмитрук Евгения Викторовна | 13635287 | доцент | к.т.н | 3000.00 | 300.00 | 2000-08-29 | 1978-02-01 | ж | 41-41-41 |
| | 29 | 3 | Владыкина Нина Довыдовна | 13635288 | ассистент | нет | 1500.00 | 300.00 | 1980-08-18 | 1956-02-01 | ж | 41-42-41 |
| | 30 | 4 | Статывка Юрий Иванович | 172625218 | доцент | к.т.н | 3500.00 | 500.00 | 1978-08-21 | 1966-02-11 | м | 41-49-51 |
| | 31 | 4 | Пархоменко Виталий Павлович | 172625000 | доцент | к.т.н | 3000.00 | 100.00 | 2001-09-11 | 1978-02-11 | м | 41-49-87 |
| | 33 | 4 | Тарасенко Андрей Владимирович | 17262501 | старший препод | нет | 2000.00 | 100.00 | 2000-11-15 | 1976-12-11 | м | 41-52-87 |
| | 34 | 5 | Холод Олег Николаевич | 684849388 | профессор | д.т.н | 5000.00 | 100.00 | 2000-11-15 | 1956-12-01 | м | 41-00-07 |
| | 35 | 5 | Швец Светлана Николавена | 684849865 | старший препод | к.т.н | 2000.00 | 400.00 | 2001-10-22 | 1976-07-01 | ж | 40-00-00 |
| | 36 | 5 | Яковенко Владимир Андреевич | 684849865 | профессор | д.т.н | 4000.00 | 200.00 | 1980-10-01 | 1951-11-18 | м | 40-03-09 |
| | 37 | 6 | Губачева Лариса Николаевна | 6848900 | профессор | д.т.н | 6000.00 | 100.00 | 1976-01-01 | 1957-03-22 | ж | 40-03-11 |
| | 38 | 6 | Бобровский Геннадий Александров | 61038373 | доцент | к.т.н | 3000.00 | 100.00 | 1988-01-01 | 1971-04-12 | м | 42-03-11 |
| | 39 | 6 | Жученко Наталья Макимовна | 110238373 | ассистент | нет | 1200.00 | 200.00 | 1989-01-01 | 1971-07-09 | ж | 42-03-12 |
| | 40 | 7 | Ульин Иван Васильевич | 11020980 | профессор | д.т.н | 4000.00 | 500.00 | 1977-01-01 | 1945-12-19 | м | 49-03-11 |
| | 41 | 7 | Харьковский Сергей Тимофеевич | 11020982 | профессор | д.т.н | 4000.00 | 100.00 | 2000-03-11 | 1971-12-19 | м | 49-03-15 |
| | 42 | 7 | Смирный Олег Иванович | 41020982 | ассистент | нет | 1000.00 | 500.00 | 2004-11-21 | 1981-10-18 | м | 49-03-18 |
| | 45 | 8 | Краснопольский | 410209887 | профессор | д.г.у | 5000.00 | 500.00 | 2000-03-09 | 1967-10-11 | м | 49-56-90 |
| | 46 | 8 | Швед Марина Федоровна | 41098733 | ассистент | нет | 1000.00 | 300.00 | 1987-12-19 | 1976-05-09 | ж | 61-00-90 |
| | 47 | 9 | Полеченко Наталья Юрьевна | 4198655 | ассистент | нет | 1000.00 | 300.00 | 2001-01-10 | 1979-08-19 | ж | 61-02-90 |
| | 48 | 9 | Малахова Марина Алексеевна | 884198655 | ассистент | нет | 1000.00 | 400.00 | 2002-03-22 | 1980-09-22 | ж | 61-02-92 |
| * | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

1 для 28 | Ячейка доступна только для чтения.

**Лабораторная работа № 6 Создание экранной формы (6 часов)**

**Цели работы:** Научиться создавать формы ввода-вывода; научиться создавать кнопочные

**Форма** – это средство, упрощающее ввод, редактирование и отображение информации,хранящейся в таблицах базы данных. Она представляет собой окно с набором элементов управления.

Форма сама по себе не хранит информацию, она просто обеспечивает удобный способ доступа к информации, хранящейся в одной или нескольких таблицах. Формы по сравнению с обработкой данных в режиме таблицы обладают следующими преимуществами:

- Форма позволяет в каждый момент сфокусировать внимание на отдельной записи;
- Элементы управления на форме можно расположить логичным образом, облегчающим чтение и работу с данными;
- Отдельные элементы управления обладают возможностями облегчить ввод и изменение отдельных данных;
- Некоторые объекты баз данных, такие как рисунки, анимации, звуки и видеоклипы, могут отображаться только в режиме формы, но не в режиме таблицы.

Создание кнопочной формы.

Кнопочное меню представляет собой форму, на которой расположены элементы управления – кнопки с поясняющими надписями. Щелчок на кнопке открывает соответствующую таблицу, запрос, форму или отчет. Меню - удобный инструмент работы с базами данных, и он практически всегда присутствует в базах созданных для предприятий или фирм.

Кнопочное меню создают с помощью Диспетчера кнопочных форм.

Отчет – это гибкое и эффективное средство для организации просмотра и распечатки итоговой информации. В отчете можно получить результаты сложных расчетов, статистических сравнений, а также поместить в него рисунки и диаграммы. Пользователь имеет возможность разработать отчет самостоятельно (в режиме *Конструктора*) или создать отчет с помощью *Мастера*, т.е. полуавтоматически.

**Задание для практической работы Задание 1. Создайте формы к базе данных**

1) Откройте свою базу данных.
2) Создайте форму с помощью <Мастера форм> на базе таблицы <Ведомость успеваемости>.

- Откройте таблицу <Ведомость успеваемости>.
- Выберите закладку <Формы >, щелкните мышкой по кнопке <Другие формы>.

- В появившемся диалоговом окне выберите <Мастер форм>.
- В поле <Таблицы/Запросы> выберите таблицу <Ведомость успеваемости>, в поле

<Доступные поля> выберите поля <Фамилия>, <Имя> и перенесите их стрелкой в поле

<Выбранные поля>. Также перенесите поля с названием предметов, щелкните по кнопке <Далее>.

- Выберите внешний вид формы – Табличный, щелкните по кнопке <Далее>.
- Выберите требуемый стиль (н-р, Обычная), щелкните по кнопке <Далее>.
- Задайте имя формы <Успеваемость> и щелкните по кнопке <Готово>. В результате получите форму, в которой можно менять данные и вводить новые значения.
- Закройте форму.

3) Создайте форму на основе таблицы <Преподаватели>.

- Откройте таблицу <Преподаватели>.
- Выберите закладку <Формы >, щелкните мышкой по кнопке <Другие формы>.

- В появившемся диалоговом окне выберите <Мастер форм> .
- Выберите внешний вид формы - < ленточный>.
- Выберите любой стиль.
- Получите готовую форму. Сохраните ее под именем <Преподаватели>.
- Закройте форму.
- Создайте форму <Личные данные> с помощью инструмента <Пустая форма>
- На вкладке Создание в группе Формы щелкните Пустая форма. 
- Access открывает пустую форму в режиме макета и отображает область Список полей.
- В области Список полей щелкните знак плюс (+) рядом с таблицей или таблицами, содержащими поля, которые нужно включить в форму.
- Чтобы добавить поле к форме, дважды щелкните его или перетащите его на форму. Чтобы добавить сразу несколько полей, щелкните их последовательно, удерживая нажатой клавишу CTRL. Затем перетащите выбранные поля на форму.
- Закройте окно списка полей.
- Перейдите в режим Конструктора

**Примечание 1.** Размер окошка для названия поля и для его значений меняются мышкой.

Для этого выделите черный квадратик рамки (рамка станет цветной), установите курсор на границу рамки и с помощью двунаправленной стрелки измените размеры рамки.

**Примечание 2.** С помощью кнопок панели инструментов Шрифт меняйте соответственно цвет фона, текста, линии/границы и т.д.

- Расположите элементы удобно по полю.
- Задайте размер текста поля <Фамилия> равным 24 пт, шрифт - синего цвета.
- Увеличьте в высоту рамку поля <Фотография>.
- Сохраните форму с именем <Данные студентов>.
- Посмотрите все способы представления форм: в режиме Конструктора, режиме Макета и режиме Форм.
- Закройте форму.

4) Добавьте в таблицу <Личные данные> логическое поле <Институт> (т.е., собирается ли в дальнейшем учащийся поступать в институт). Значение этого поля <ДА> или

<НЕТ>.

- Откройте таблицу <Личные данные> в режиме Конструктор. Добавьте поле с именем

<Институт> и типом Логический. Закройте таблицу.

- Перейдите на закладку Формы и откройте форму <Данные студентов> в режиме Конструктор
- Щелкните по кнопке <Список полей> на панели инструментов, выделите название

<Институт> и перетащите его мышкой в область данных, появиться значок

и надпись <Институт>.

- Расположите новые элементы по правилам оформления формы (с помощью мыши).
- Закройте <Список полей>

**Примечание 3.** Если флажок установлен, поле в таблице имеет значение <ДА>, если снят, то <НЕТ>.

- Перейдите в режим <Раздельная форма> и посмотрите записи. Установите флажки у восьми разных учащихся.
- Закройте форму, ответив утвердительно на вопрос о сохранении.

5) Создайте кнопочную форму <Заставка> с помощью Конструктора.

- Щелкните по кнопке <Создать>.
- Выберите <Конструктор>. Появиться пустая форма. Задайте мышкой ширину формы, равную 10см, а высоту – 7см. Сохраните работу с именем <Заставка>.
- Откройте созданную форму <Заставка> в режиме Конструктора.
- Выберите на панели инструментов <Элементы управления> кнопку Аа – <Надпись>. Курсор мышки примет вид крестика с «приклеенной» буквой А. Щелкните мышкой по месту начала надписи и введите:

*База данных*

*«Гостиница» группа ГС - 31*

(после слов База данных нажмите одновременно комбинацию клавиш Shift+Enter.)

- Нажмите клавишу <Enter>. Выберите размер букв 18, а выравнивание - по центру. Цвет фона – голубой. Растяните мышкой надпись на ширину окна.
- Выберите на панели элементов значок Кнопка. Щелкните мышкой по тому месту области данных, где должна быть кнопка. Появиться диалоговое окно <Создание кнопок>.
- Выберите категорию <Работа с формой>, а действие <Открыть форму>, и щелкните по кнопке <Далее>.
- Выберите форму <Успеваемость> , открываемую этой кнопкой щелкните по кнопке

<Далее>. В следующем окне также щелкните по кнопке <Далее>.

- В следующем окне поставьте переключатель в положение <Текст>, наберите в поле слово <Успеваемость> (Рисунок8.1) и щелкните по кнопке <Далее>.

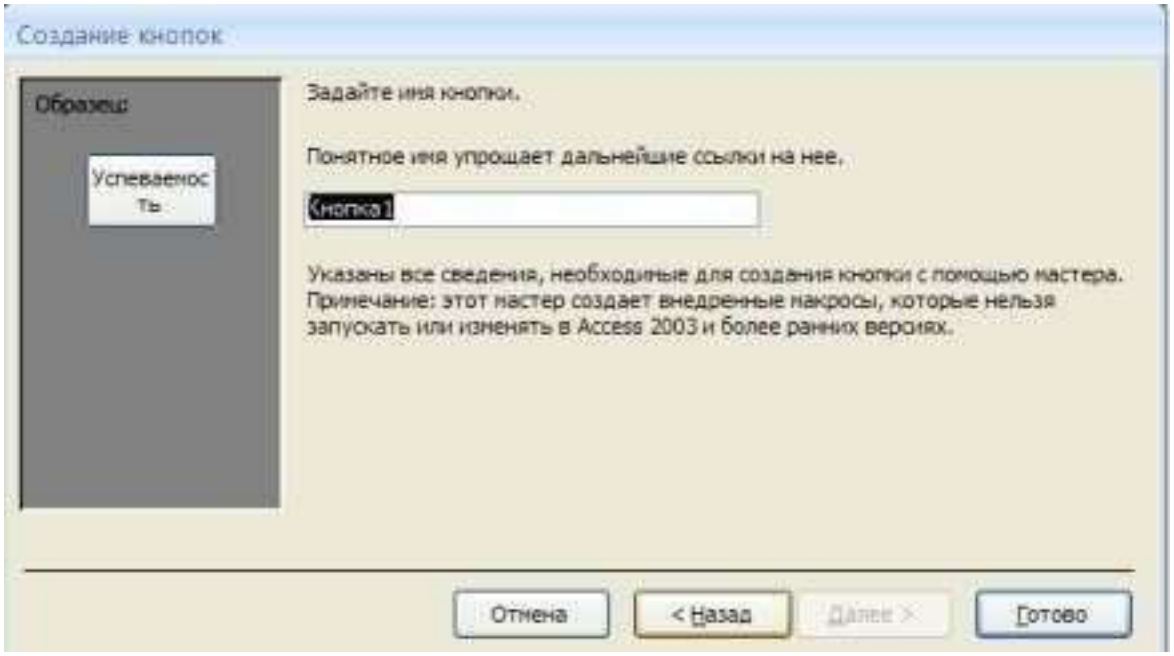



Рисунок 8.1. Создание кнопок

- Задайте имя кнопки <Успеваемость> и щелкните по кнопке <Готово>.

**Примечание 4.** Размер и расположение кнопок можно менять мышкой в режиме Конструктор.

Самостоятельно создайте кнопки для форм <Личные данные> и <Преподаватели>.

- Перейдите в режим формы (Рисунок 8.2). Теперь при щелчке мышью по соответствующим кнопкам будут открываться соответствующие формы для работы.
- Закройте форму.


Рисунок 8.2. Окно формы

6) Создайте кнопочную форму при помощи Диспетчера кнопочных форм.

- Откройте вкладку Работа с базами данных, команда - Диспетчер кнопочных форм.

Вы получите диалоговое окно, представленное на Рисунке 8.3.

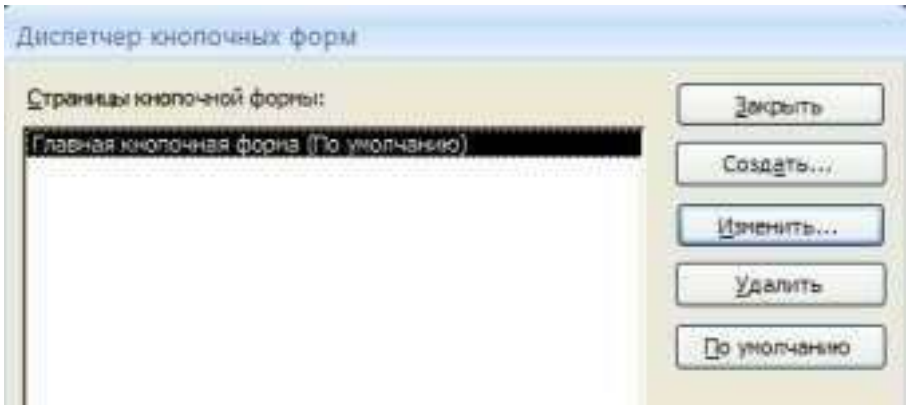



Рисунок 8.3. Диспетчер кнопок

- Щелкните в этом окне по кнопке <Изменить>.
- В следующем окне щелкните по кнопке <Создать> и в появившемся окне измените содержимое полей в соответствии с Рисунком 8.4 (Команду и Форму выбирайте из списка, а не набирайте вручную). Щелкните по кнопке <ОК>.

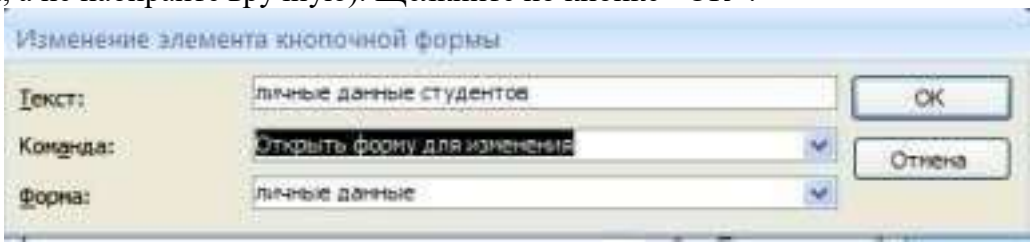



Рисунок 8.4. Изменение элементов кнопочной формы

- Аналогично создайте еще три элемента кнопочной формы: <Успеваемость>,

<Преподаватели > и <Заставка>.

- Добавьте кнопку закрытия базы данных. Для этого щелкните по кнопке <Создать>, наберите в поле Текст слово <Выход>, а в поле Команда выберите <Выйти из приложения>. Закройте диалоговые окна.
- Откройте окно <Кнопочная форма> в режиме Конструктора или Макета, измените цвет надписи и название вашей базы данных на ГОСТИНИЦА, сохраните форму.
- Украсьте вашу форму рисунком. Для этого щелкните по значку Эмблема и выберите в открывшемся окне папку с рисунками, выберите понравившийся и вставьте в

свою кнопочную форму.

- Перейдите в режим формы, проверьте работу всех кнопок кнопочной формы. Завершите работу с базой данных, нажав на кнопку <Выход>.

**Задание 2. Создайте отчет с помощью *Мастера отчетов.***

- Откройте вкладку *Создание,* меню *Отчеты*.
- Выберите *Мастер отчетов* и таблицу «Личные данные».
- Выберите нужные поля, которые будут участвовать в отчете, нажмите кнопку «Далее».
- В новом окне выберите поля для группировки так, чтобы сначала было указано поле

«Фамилия», нажмите кнопку «Далее».

- На этом шаге отсортируйте данные по алфавиту, нажмите кнопку «Далее».
- Выберите вид макета *Ступенчатый* и щелкните по кнопке «Далее».
- Выберите стиль отчета: *Открытая* и щелкните по кнопке «Далее».
- Задайте имя отчета: «Отчет1» и щелкните по кнопке «Готово». Вы попадете в режим просмотра отчета.
- Закройте отчет согласившись с сохранением.

Самостоятельно. Составьте еще два отчета по запросам – «Запрос 3» и «Запрос 5», выбирая из разных макетов: *блок*; *структура*, выбирая из разных стилей. Сохраните отчеты под именами «Отчет 2» и «Отчет 3».

**Задание 3. Создайте Пустой отчет** в столбец на базе таблицы «Ведомость успеваемости» и сохраните его с именем «Успеваемость».

С помощью Конструктора измените цвет букв заголовка, их размер и шрифт.

**Задание 4. Создайте почтовые наклейки.**

- Откройте вкладку *Создание,* меню *Отчеты*.
- Выберите таблицу «Личные данные», команда Наклейки.
- В следующем окне щелкните по кнопке «Далее».
- В следующем окне выберите шрифт, размер шрифта, насыщенность и цвет, вновь щелкните по кнопке «Далее».
- В следующем окне создайте прототип наклейки, напечатав слово ЛИЧНОСТЬ и выбрав соответствующие поля, щелкните по кнопке «Далее».
- В следующем окне укажите поля для сортировки (Фамилия, Имя), щелкните по кнопке

«Далее».

- Введите имя отчета «Наклейки» и щелкните по кнопке «Готово».
- Просмотрите Наклейки (Рисунок 8.5).

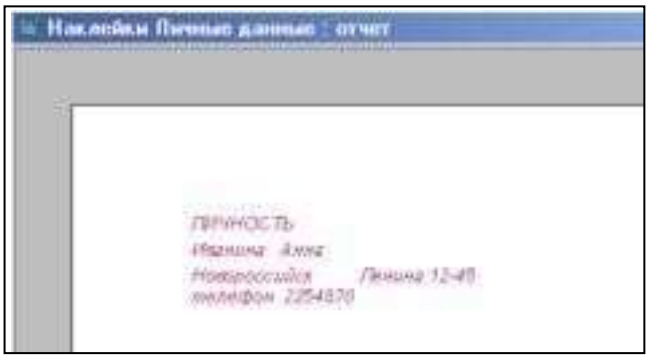


Рисунок 8.5. Отчет

**Лабораторная работа № 7 Создание элементов управления на форме.**

**Цель занятия:** изучить технологию создания форм и упраляющих кнопок

**Теоретические сведения:**

В VisualFoxPro для просмотра, ввода и редактирования данных, хранящихся в таблицах, используются формы, являющиеся более наглядным средством представления информации. Важным преимуществом форм является, то, что они позволяют работать не с одной, а с несколькими связанными таблицами, что, в свою очередь, также увеличивает наглядность.

При создании форм в VisualFoxPro разработчик может использовать следующие средства: FormWizard (мастер форм), FormBuilder (построитель формы), Builder (построитель объектов формы), AutoFormatBuilder (построитель автоформата), FormDesigner (конструктор форм).

Чтобы создать форму для одной или связанных таблиц с возможностью задания отображаемых в форме полей, стиля их отображения и указания типа кнопок управления, можно использовать FormWizard (мастер создания форм).

Для самостоятельной разработки формы с заданными свойствами или изменения формы, созданной с помощью мастера, вам необходимо использовать FormDesigner (конструктор форм).

Для облегчения размещения в конструкторе форм полей и надписей, оформленных в соответствии с выбранным стилем, можно использовать FormBuilder (Построитель формы).

**Создание формы с помощью конструктора форм**

Любая форма в VisualFoxPro состоит из объектов, каждый из которых имеет характерные свойства. Для любого объекта вы можете указать действия, выполняемые написанной разработчиком программой при наступлении определѐнных событий. Процесс создания формы в конструкторе форм состоит в размещении в форме объектов и определении свойств, а также связанных с ними событий и выполняемых действий.

Процесс создания формы включает следующие действия:

- настройка параметров формы
- определение среды окружения, то есть выбор используемых в форме таблиц и установка связей между ними
- размещение в форме объектов: текста, полей различных типов, линий, рисунков, кнопок управления.

**Задание для практической работы**

**Задание 1. Создание формы средствами мастера форм**

Запустить программу MicrosoftVisualFoxPro

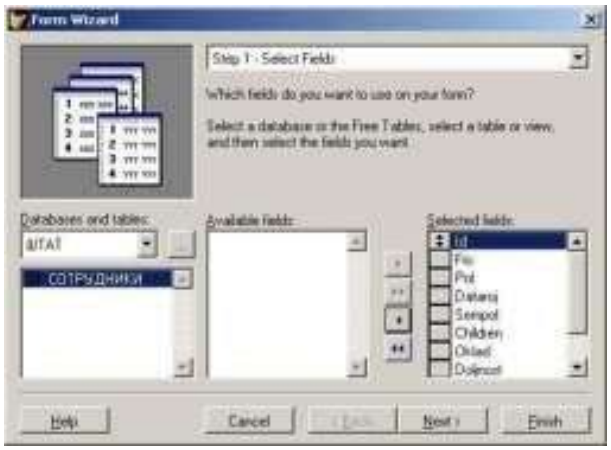
Открыть созданный проект: File/Open/Информационная система

Открыть БД штат: в окне ProjectManager (Менеджер проекта) вкладка Data/Databases/штат/Open

Создать форму Сотрудники, щелкнув клавишей мыши на вкладке Documents/Forms/New/FormWizard/FormWizard, нажать клавишуOk


Рисунок 10.1. Диалоговое окно WizardSelection

В появившемся окне FormWizard в области Databasesandtables (Базы данных и таблицы) выбрать необходимую БД штат и указать таблицу Сотрудники, для которой создается форма. Из области Availablefields (Имеющиеся поля) выбрать поля, которые будут размещены в форме, в соответствии с рис.10.2, используя кнопки расположенные между




списками для перехода к следующему шагу нажать кнопку Next(Далее)

Рисунок 10.2. Диалоговое окно выбора полей для отображения.

В появившемся диалоговом окне в области Style установить стиль отображения объектов и в области Button type типы кнопок управления в соответствии с рис.10.3, нажать кнопку Next

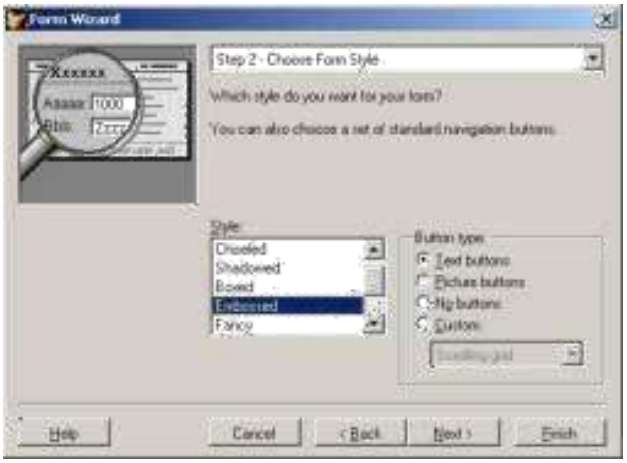



Рисунок 10.3. Окно выбора стиля отображения полей и управляющих кнопок.

В появившемся диалоговом окне задать критерий сортировки данных, указав поля по которым будет осуществляться упорядочивание в соответствии рис.10.4. нажать кнопку Next

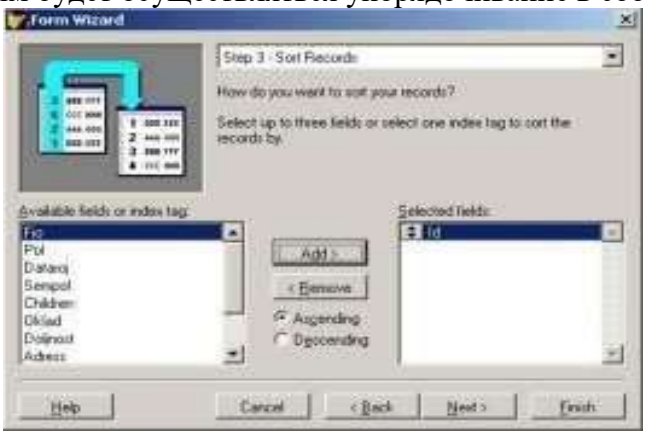



Рисунок 10.4. Установка критерия упорядочения данных

На заключительном шаге создания форм в области Typeatitleforyourform(Тип заголовка формы) указать имя формы

Сотрудники. Воспользовавшись кнопкой Preview (Просмотр) просмотрите, как будет выглядеть создаваемая форма, после просмотра нажмите кнопку ReturntoWizard.


Рисунок 10.5.Окно задания заголовка формы.

Если что-то не так, вернитесь к предыдущим шагам, воспользовавшись кнопкой Back. Нажать кнопку Finish (Готово) и в появившемся диалоговом окне SaveAs (Сохранить как) указать папку, в которой будет размещена форма Сотрудники

Запустить в окне ProjectManager форму Сотрудники: Documents/Forms/Сотрудники/Run

Рисунок 10.6. Окно ProjectManager


Или с помощью меню программы VisualFoxPro: Program/Dозадать имя формыСотрудникии тип файлаForm, нажать кнопкуDo. С помощью кнопок (см. таблицу 10.1.), находящихся в нижней части формы можно вводить или редактировать записи в таблице сотрудники.

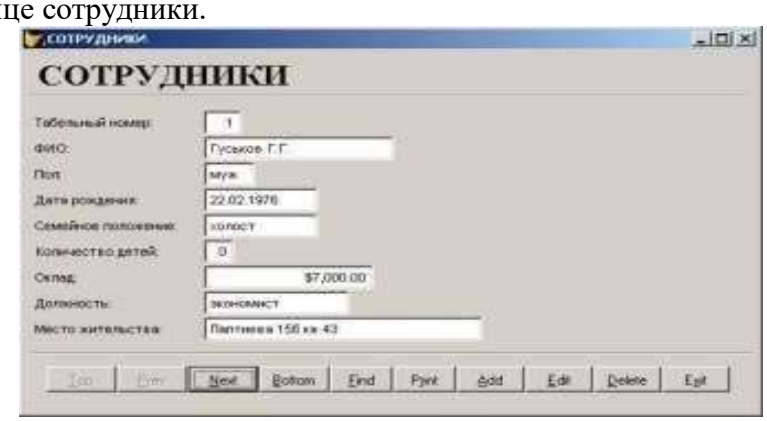



Рисунок 10.7.Форма Сотрудники

Ввести в таблицу Сотрудники дополнительно шесть записей, одну удалить, две отредактироватьТаблица 10.1.Кнопки управления и редактирования данных

**Индивидуальные задания к практической работе**

Создать форму для таблицы Книги БД Библиотека

1. В диалоговом окне выбора полей для отображения выбрать все поля

| Top | Prev | Next | Buttom | Find | Save |
|---|---|---|---|---|---|
| Первая запись | Предыдущая запись | Следующая запись | Последняя запись | Найти | Сохранить |
| Print | Add | Edit | Delete | Exit | Revent |
| Вывести на печать | Добавить запись | Редактировать запись | Удалить запись | Выход | Отменить |

таблицы Книги

2. В окне выбора стиля отображения полей и управляющих кнопок (см. рис.3.) в области Style (стиль) выбрать стиль Embossed, а в областиButtontype (Типы кнопок управления) выбрать пункт NoButtons

3. Установить критерий упорядочения данных по полю kodknigi

4. В окне задания заголовка формы в области Typeatitleforyourform(Тип заголовка формы) указать имя формы Книги. Просмотреть вид создаваемой формы нажатием кнопки Preview (Просмотр).

5. После сохранения запустить форму Книги, она должна иметь вид как на рис.10.8.

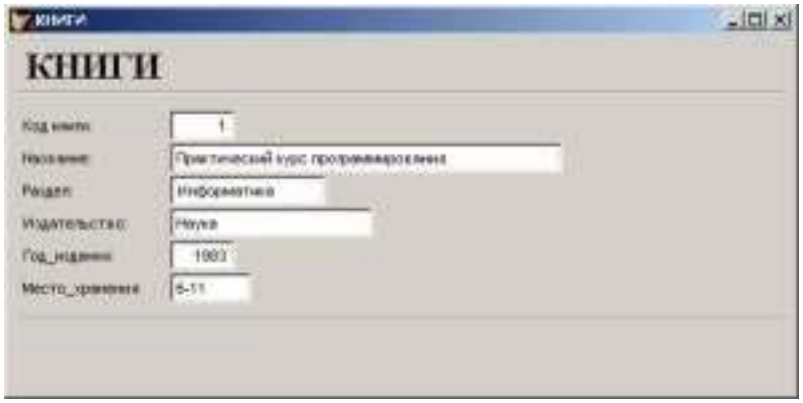



Рисунок 10.8. Форма Книги

**Задание 2. Создание формы средствами конструктора форм**

Запустить программу MicrosoftVisualFoxPro Открыть проект Информационная система

Открыть БД штат Выбрать в окне Project Manager вкладку Documents/Forms/New/New Form. Откроется окно FormDesigner(Конструктор форм) для создания новой формы

Открыть окно окружения формы DataEnvironment (Среда окружения): пункт меню View/ DataEnvironment. Для размещения таблицы в среде окружения выбрать команду Add(Добавить) из меню DataEnvironment (Среда окружения) или вызвать контекстное меню на окне окружения формы DataEnvironmentи выбрать пункт Add. В появившемся диалоговом окне AddTableorView (Добавить таблицу или представление данных), выбрать из списка таблиц открытой базы штат таблицу Сотрудники, для которой создаѐтся форма, и нажать кнопку OK

Открыть окно свойств таблицы, размещѐнной в окне окруженияDataEnvironment. Для этого установить на таблицу Сотрудники курсор, вызвать контекстное меню правой клавишей мыши и выбрать команду Properties (Свойства)

В окне Properties (свойства) выделить свойство Order (Упорядочение). Для упорядочения данных в форме в поле коррекции свойства нажмите кнопку раскрытия списка и из списка индексов таблицы выберите индекс (id), по которому хотите упорядочить данные.

Закройте окно определения среды окружения DataEnvironment.

Для задания свойств формы установить курсор в форму, вызвать контекстное меню, выбрать команду Properties (Свойства). Откроется окно Properties(Свойства). В его верхнем списке, указывающем название объекта, для которого осуществляется настройка свойств

содержится текст Form1 (Форма1)

В окне Properties (Свойства) скорректировать свойство Caption (Надпись), введя в текстовое поле заголовок формыСотрудники тестСвойство формы AutoCenter (Автоцентр) должно иметь значение True(Истина), чтобы форма располагалась в центре экрана

Изменить свойства FontName (Наименование шрифта), указав шрифт TimesNewRomanu FontSize (Размер шрифта) указав размер шрифта 12 После того как вы определили параметры формы, разместили в окружении используемые таблицы, можно приступать к размещению объектов в форме. Осуществим размещение полей таблицы Сотрудники и надписей к ним, используя FormBuilder (построитель формы)

Для запуска построителя форм установить курсор в форму, вызвать контекстное меню, выбрать команду Builder (Построитель), откроется диалоговое окноFormBuilder (построитель формы), содержащее две вкладки: FieldSelection (Выбор поля) и Styles (Стиль). В вкладке FieldSelection (Выбор поля) из области Databasesandtables (Базы данных и таблицы) выбрать БД штат и таблицу Сотрудники, затем из списка Availablefields (Имеющиеся поля) перенести в Selectedfields(Выбранные поля) все необходимые поля используя кнопки расположенные между списками. Перейти на вкладку Styles (Стиль), выбрать из списка стилей стиль Embossed и нажать кнопку OK. На форме будут размещены объекты формы (см. рис. 10.9.).

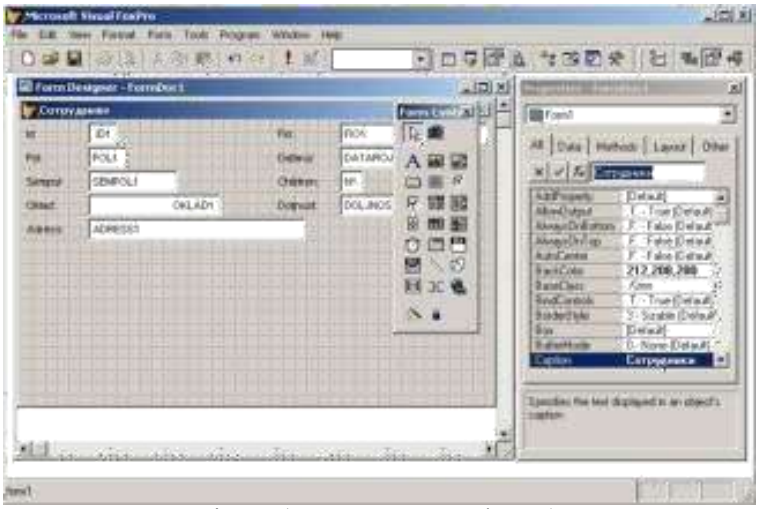
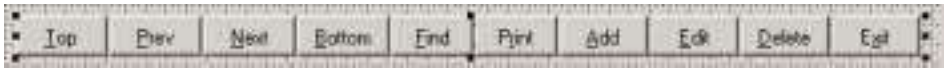


Рисунок 10.9. Окно FormDesigner(Конструктор форм) Рисунок 10.10. Объект - блок кнопок управления




Для размещения на создаваемой форме кнопок управления воспользуемся ранее созданной формой Сотрудники. Открыть форму Сотрудники в режиме редактирования: окно программы ProjectManager/Documents/Forms/сотрудники/Modify. В появившемся окне FormDesigner формы Сотрудники нажатием левой клавиши мыши выделить объект BUTTONSET1 (блок кнопок управления, см. рис.10.10.) и нажать комбинацию клавиш Ctrl+Сна клавиатуре или установить курсор в форму, вызвать контекстное меню, выбрать команду Copy. Закрыть окно FormDesigner формы Сотрудники.

В окне FormDesigner формы Сотрудники тест установить курсор в форму, вызвать контекстное меню, выбрать команду Paste или нажать комбинацию клавиш Ctrl+V. Блок кнопок управления записями таблицы (см. рис.10.10) в создаваемой форме будет скопирован.

Задать цвет фона формы. Выделить в окнеProperties (Свойства) свойство формы BackColor (цвет фона), нажать расположенную с правой стороны поля редактирования свойства кнопку и в открывшемся диалоговом окне Цвет выберите цвет, который вы хотите использовать для фона

Расположить объекты формы в соответствии с рис.10.11. Для перемещения объектов формы можно использовать метод перетаскивания мышью или кнопками управления курсором на клавиатуре, предварительно выделив необходимый объект, а также комбинацией клавиш Ctrl+[кнопки управления курсором], использовать все выше перечисленные способы, выяснить их различия. Для изменения размеров формы или объектов формы необходимо выделить объект (форму), подвести курсор к углу объекта (формы), когда курсор примет вид разнонаправленной стрелки нажать левую клавишу мыши и удерживая еѐ, изменить размеры объекта (формы)

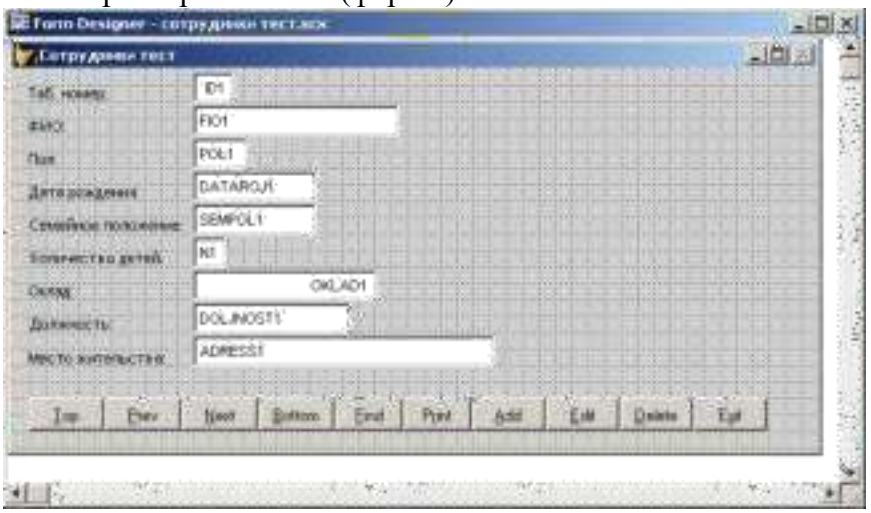



Рисунок 10.11 Вид отредактированной формы

Запустить отредактированную форму: установить курсор в форму, вызвать контекстное меню, выбрать команду RunForm или нажать кнопку [ ] на панели инструментов. Система попросить сохранить созданную форму, нажать кнопку Yes, в появившемся окне указать папку и имя файла (Сотрудники тест), сохранить.

**Индивидуальные задания к практической работе**

Создать кнопки навигации в форме Книги

Открыть форму Книги в окне FormDesigner (Конструктор форм)

Нажать кнопку CommandGroup (Группа кнопок) на панели инструментов FormControls (элементы управления формы) и щѐлкнуть в месте их предполагаемого размещения в форме

В Окне Properties (свойства) размещѐнного объекта выделить свойство ButtonCount (Количество кнопок) и указать необходимое количество размещаемых в форме кнопок, указав число 5

Увеличить с помощью мыши размеры рамки, окружающей данный объект. Перейти в режим редактирования: установить на объект курсор, вызвать контекстное меню и выбрать команду Edit (Редактировать). Выделяя поочерѐдно кнопки, переместить их, расположив горизонтально, в одну линию

Выйти из режима редактирования, щѐлкнув вне области объекта CommandGroup (Группа кнопок)

Скорректировать размер рамки, окружающей объект: в свойстве AutoSize (Авторазмер) установить значение True (Истина)

Открыть окно свойств объекта CommandGroup (Группа кнопок), нажать кнопку раскрытия списка в верхней части данного окна, поочерѐдно выбрать из списка элементы Command1, Command2, Command3, Command4, Command5 и, используя свойство

Caption(Надпись) задать названия кнопок соответственно: Первая, Следующая, Предыдущая, Последняя, Выход (см. рис.10.12.)

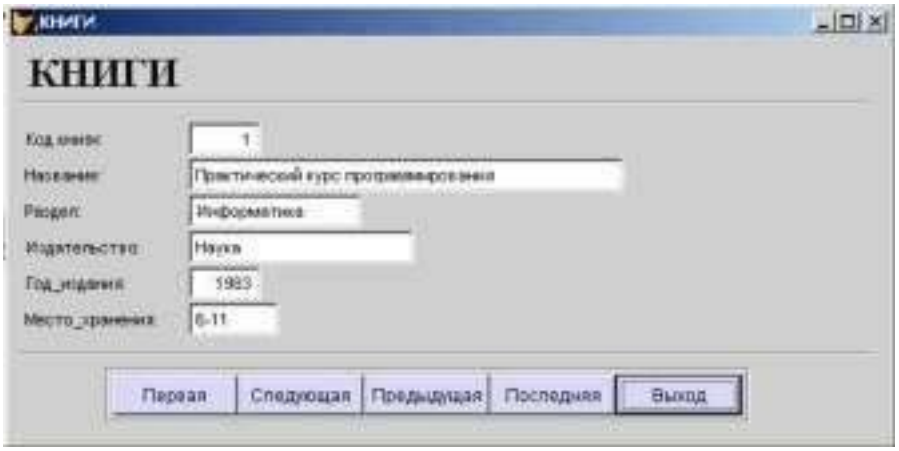



Рисунок 10.12. Форма Книги

Определить команды, которые будут выполняться при нажатии на созданные кнопки. Установить на объект CommandGroup (Группа кнопок) курсор, вызвать контекстное меню и выбрать команду Edit (Редактировать), выделить кнопку Первая, вызвать контекстное меню выбрать команду Code (Код программы), в появившемся окне ввести следующие команды:

Для кнопки Первая:

Переходим на первую запись и обновляем информацию в форме IF !BOF()

GO TOP ENDIF

_screen.ActiveForm.Refresh()

После ввода кода программы закрыть окно.

Ввести аналогично команды для остальных кнопок:

Для кнопки Следующая:

Переходим на следующую запись и обновляем информацию в форме IF !EOF()

SKIP ENDIF

_screen.ActiveForm.Refresh() Для кнопки Предыдущая:

Переходим на предыдущую запись и обновляем информацию в форме IF !BOF()

SKIP - 1 ENDIF

_screen.ActiveForm.Refresh() Для кнопки Последняя:

Переходим на последнюю запись и обновляем информацию в форме IF !BOF()

GO BOTTOM ENDIF

_screen.ActiveForm.Refresh() Для кнопки Выход:

Запрашиваем и выходим, если Да

IFMESSAGEBOX("Выходить из формы?", 4+32+256, "Выход")=6

_screen.ActiveForm.Release() ELSE

_screen.ActiveForm.Refresh() ENDIF

После ввода команд закрыть окно процедур

Запустить форму на выполнение. Проверить результат проделанной работы

**Лабораторная работа № 8 Создание главной кнопочной формы (6 часов)**

**Цель работы:** научиться создавать пользовательский интерфейс (главная кнопочная форма, простые ленточные формы для работы с данными)

**Теоретические сведения:**

Перейдём теперь к созданию пользовательского интерфейса. Его создание начнём с создания главной кнопочной формы. Запустите «Microsoft Visual Studio 2010» и откройте созданный ранее проект «StudentsDB», щёлкнув по его значку в области «Последние проекты» стартовой страницы «Начальная страница».

После появления стандартного окна среды разработки в рабочей области на форму поместите надпись (Label) и четыре кнопки (Button) как показано на рисунке 8.1.

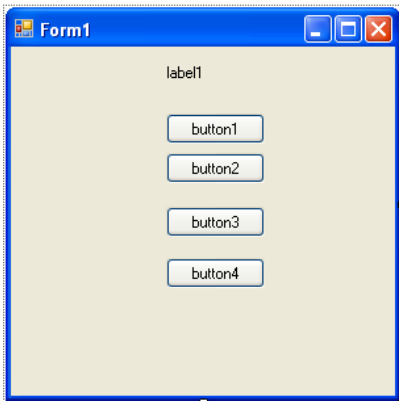



Рис.8.1

После создания объектов перейдём к настройке их свойств. Начнём с настройки свойств формы. Выделите форму, щёлкнув ЛКМ в пустом месте формы. На панели свойств задайте свойства формы как представлено ниже:

• FormBorderStyle (Стиль границы формы): Fixed3D;
• MaximizeBox (Кнопка развёртывания формы во весь экран): False;
• MinimizeBox (Кнопка свёртывания формы на панель задач): False;
• Text (Текст надписи в заголовке формы): База данных «Студент».

На форме выделите надпись, щёлкнув 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): Тёмно синий;
• Text (Текст надписи): База данных «Студент»;
• TextAlign (Выравнивание текста): MiddleCenter.

У кнопок задайте надписи (свойство «Text») как показано на рисунке 8.2.

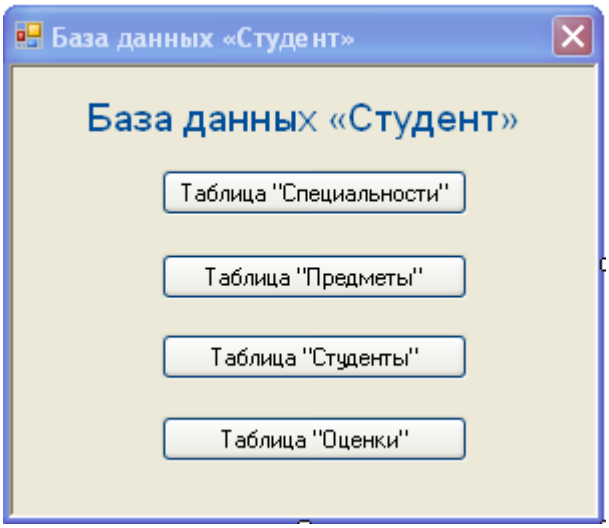



Рис.8.2

После настройки свойств вышеперечисленных объектов форма примет вид представленный на рисунке 8.2.

Теперь перейдём 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 «Проект/Добавить Windows Form». Появится окно «Добавление нового элемента- StudentsDB» (Рис.8.3).

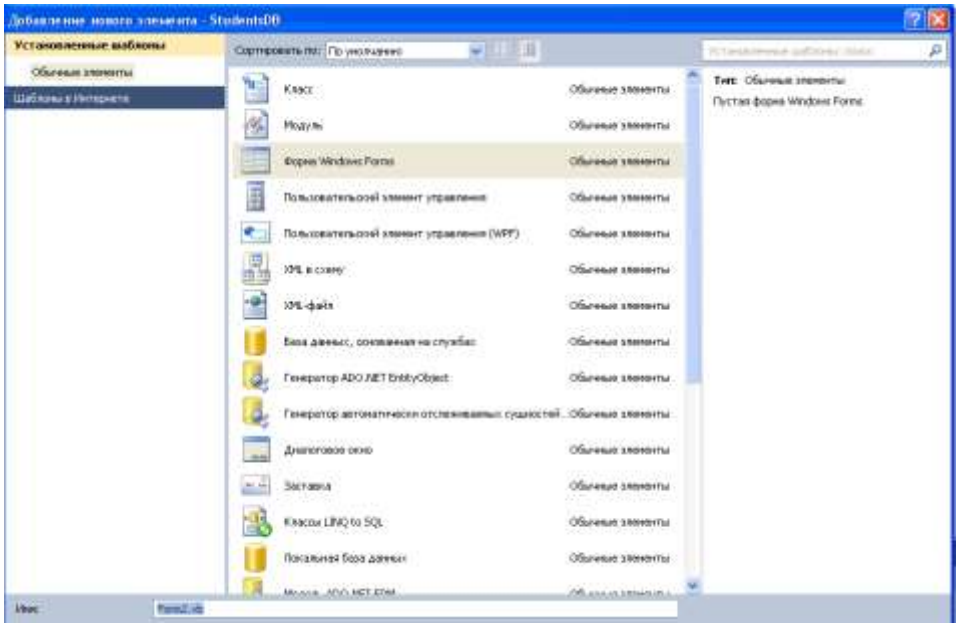


Рис.8.3

В разделе «Шаблоны» выберите «Форма Windows Form» и нажмите кнопку «Добавить». Новая пустая форма появится в рабочей области среды разработки.

В верхней части новой формы создайте надпись (Label)

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

• FormBorderStyle (Стиль границы формы): Fixed3D;
• MaximizeBox (Кнопка развёртывания формы во весь экран): False;
• MinimizeBox (Кнопка свёртывания формы на панель задач): False;
• Text (Текст надписи в заголовке формы): Таблица «Специальности».

На форме выделите надпись, щёлкнув 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): Тёмно синий;
• Text (Текст надписи): Таблица «Специальности»;
• TextAlign (Выравнивание текста): MiddleCenter.

После настройки всех вышеперечисленных свойств форма будет выглядеть следующим образом (Рис.8.4):

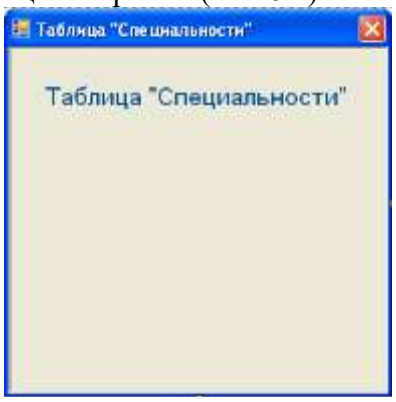



Рис.8.4

Теперь поместим на форму поля таблицы «Специальности». Сначала откройте панель «Источники данных». На панели «Источники данных» отобразите поля таблицы «Специальности», щёлкнув по значку «+», расположенному слева от имени таблицы (Рис.8.5).

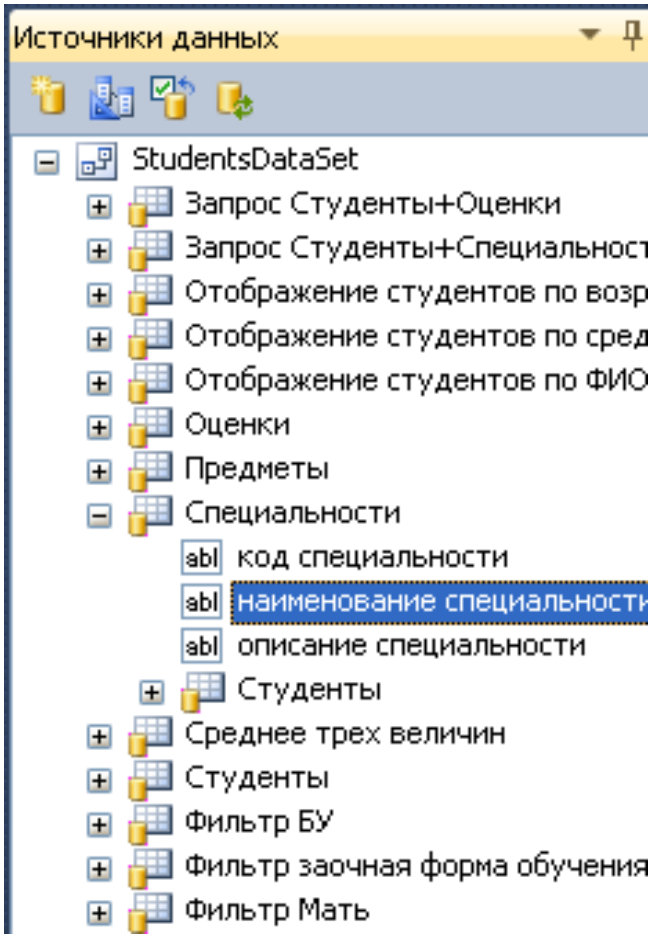



Рис.8.5

**Замечание:** Под полями таблицы специальности в виде подтаблицы располагается таблица «Студенты» (Рис.8.5). 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. 8.5), позволяющего выбирать объект, отображающий содержимое выделенного поля (Рис. 8.6).

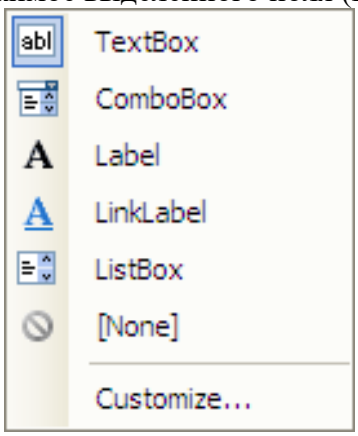



Рис. 8.6

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 рисунке 8.7

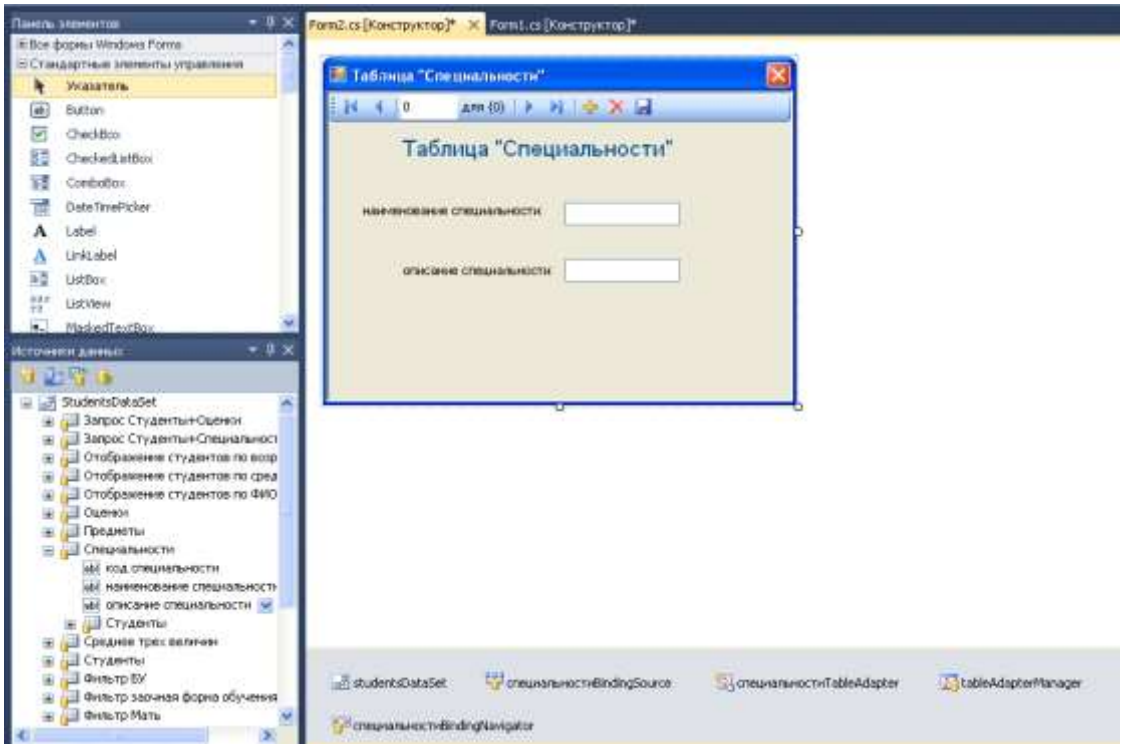



Рис. 8.7

**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» её 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, щёлкнув по вкладке «Form1.cs» в верхней части рабочей области. Для подключения новой формы «Специальности» к главной кнопочной форме дважды щёлкните ЛКМ по кнопке «Таблица «Специальности»», расположенной на главной кнопочной форме (рис.9.3). В появившемся окне кода формы в процедуре «Button1_Click» наберите команду «Form2.Show()», предназначенную для открытия формы «Таблица «Специальности»» (Form2) как это показано на рисунке 8.8.

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

Рис.8.8.

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.8.9).

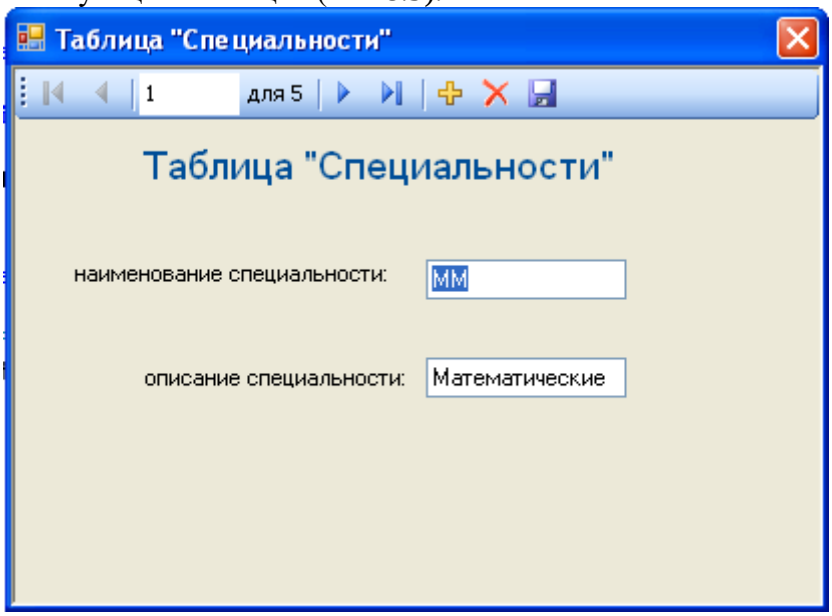



Рис.8.9

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.8.10):

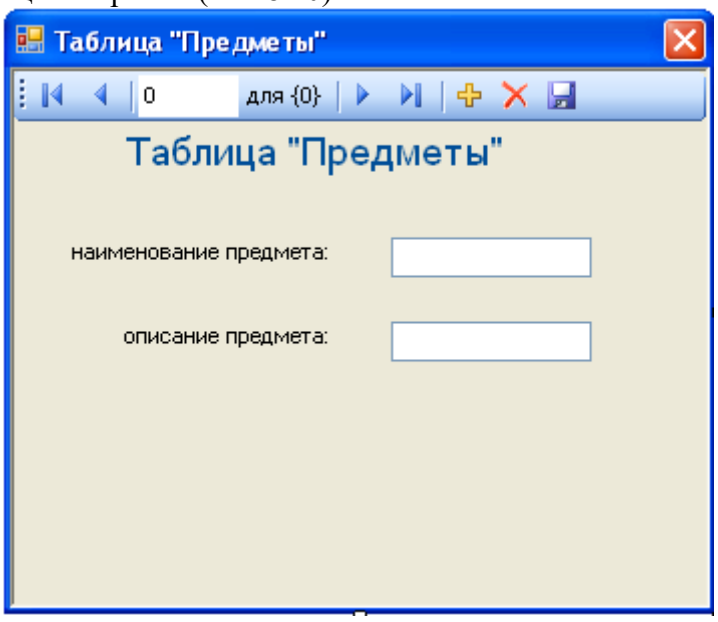


Рис.8.10

На главной кнопочной форме дважды щёлкните ЛКМ по кнопке «Таблица «Предметы»» и в появившемся окне кода в процедуре «Button2_Click» наберите «Form3.Show()»

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.8.11):

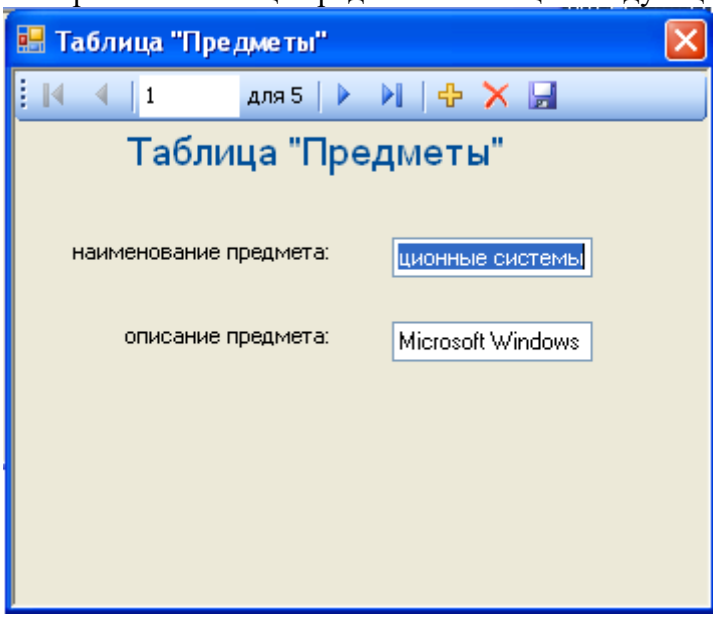



Рис.8.11

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», щёлкнув ЛКМ по знаку «+», расположенному слева от названия таблицы.

Отобразятся все поля таблицы «Студенты»

**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) .

Создайте новую форму и поместите в её 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.8.12):

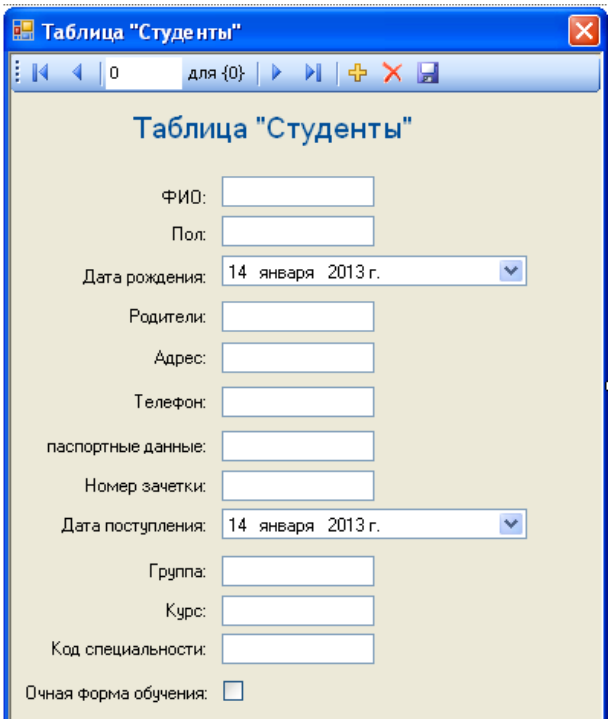



Рис.8.12

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»

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.8.13).

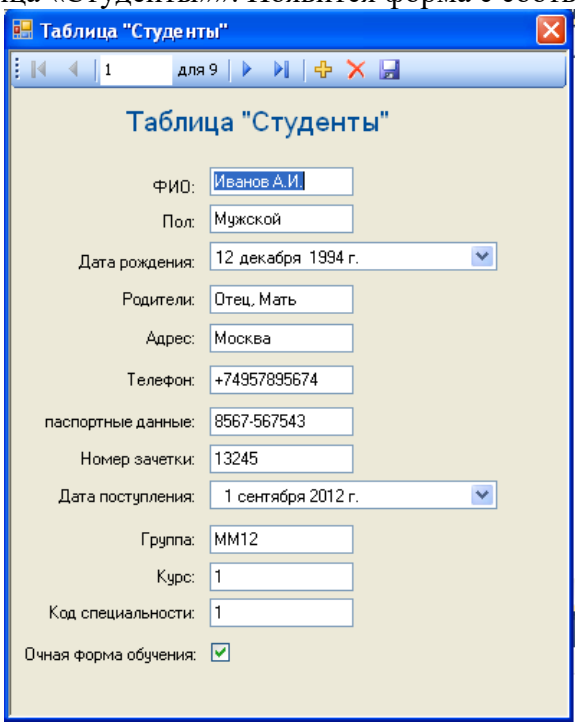



Рис.8.13

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.8.14):

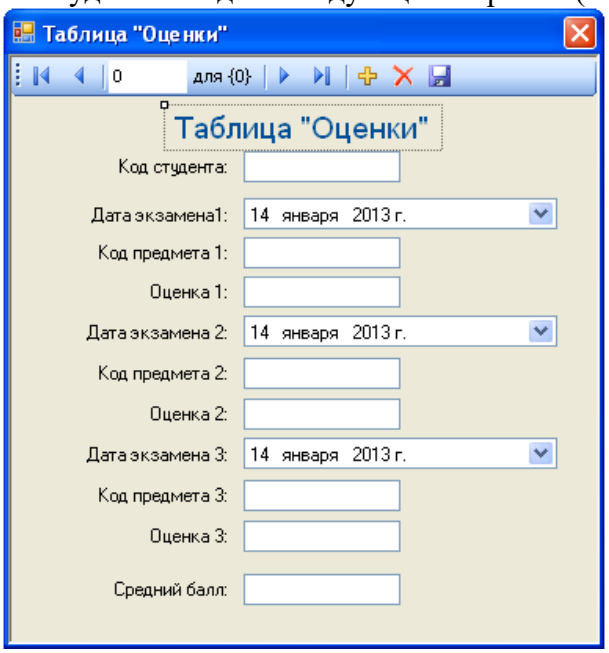



Рис.8.14

Подключите вновь созданную форму таблицы «Оценки» к главной кнопочной форме. Для этого отобразите главную кнопочную форму и на ней дважды щёлкните ЛКМ по кнопке «Таблица «Оценки»». В появившемся окне с кодом, в процедуре «Button4_Click» наберите команду «Form5.Show»

Проверьте работу формы таблицы «Оценки», запустив проект, и на главной кнопочной форме нажмите кнопку «Таблица «Оценки»». Появится вновь созданная форма (Рис.8.15).

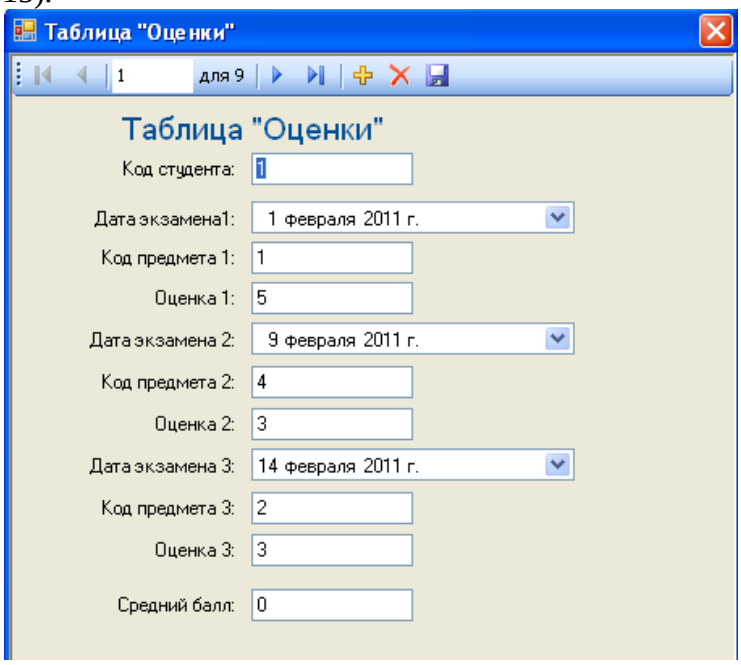
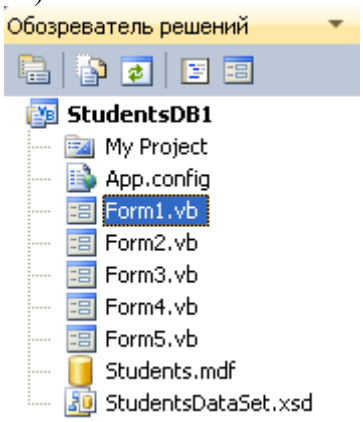



Рис.8.15

В заключение, откройте обозреватель проекта щёлкнув по его вкладке в правой части окна среды разработки. На данной панели должны отобразиться все выше созданные формы

(Рис.8.16).




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 рисунке 9.1.

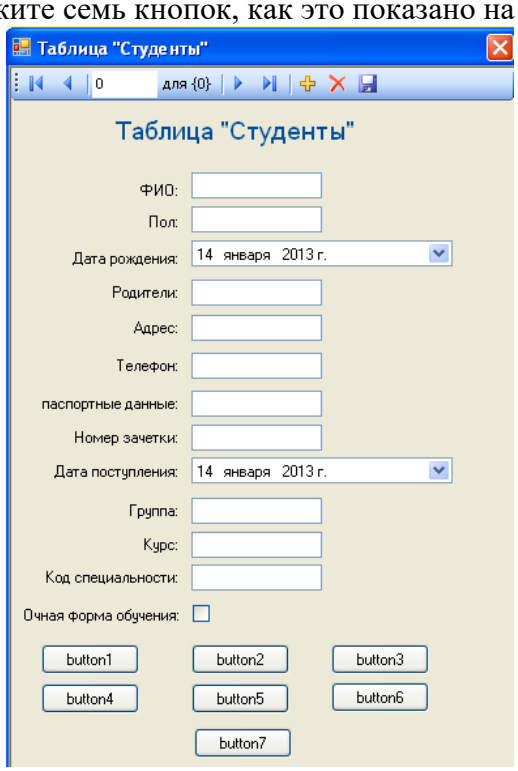



Рис.9.1

В качестве надписей на созданных кнопках (Свойство «Caption») задайте как:

«Первая», «Предыдущая», «Добавить», «Последняя», «Следующая», «Удалить» и «Сохранить» (Рис.9.2).

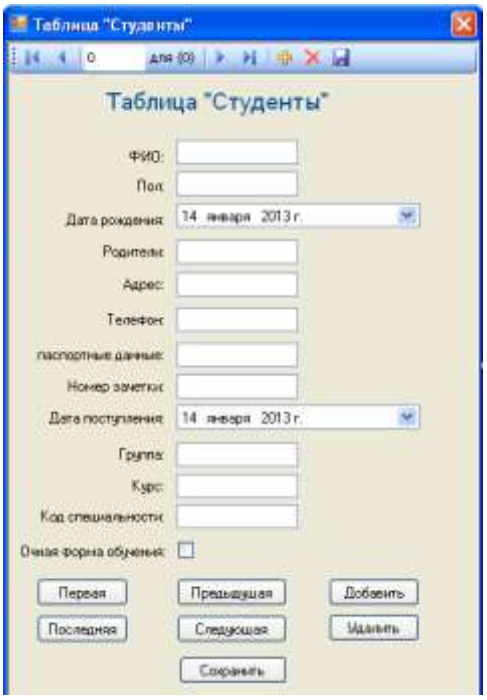



Рис.9.2

Дважды щёлкните ЛКМ по кнопке «Первая» и в появившемся окне кода формы «Form4» в процедуре «Button1_Click» наберите команду для перехода к первой записи «СтудентыBindingSource.MoveFirst()»

Дважды щёлкните ЛКМ по кнопке «Предыдущая» и в появившемся окне кода формы «Form4» в процедуре «Button2_Click» наберите команду для перехода к предыдущей записи «СтудентыBindingSource.MovePrevious()»

Дважды щёлкните ЛКМ по кнопке «Добавить» и в появившемся окне кода формы «Form4» в процедуре «Button3_Click» наберите команду для добавления новой записи «СтудентыBindingSource.AddNew()»

Дважды щёлкните ЛКМ по кнопке «Последняя» и в появившемся окне кода формы «Form4» в процедуре «Button4_Click» наберите команду для перехода к последней записи «СтудентыBindingSource.MoveLast()»

Дважды щёлкните ЛКМ по кнопке «Следующая» и в появившемся окне кода формы «Form4» в процедуре «Button5_Click» наберите команду для перехода к следующей записи «СтудентыBindingSource.MoveNext()»

Дважды щёлкните ЛКМ по кнопке «Удалить» и в появившемся окне кода формы «Form4» в процедуре «Button6_Click» наберите команду для удаления текущей записи «СтудентыBindingSource.RemoveCurrent()»

Дважды щёлкните ЛКМ по кнопке «Сохранить» и в появившемся окне кода формы «Form4» в процедуре «Button7_Click» наберите команду для сохранения изменений, отображённую на рисунке9.3.

```
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
```

Рис.9.3

Рассмотрим последнюю процедуру более подробно. Она содержит следующие команды:

• Me.Validate() – проверяет введённые 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 зачётки», «Курс» и «Код специальности». После удаления, перечисленных полей форма, отображающая таблицу «Студенты» примет следующий вид (Рис.9.4):

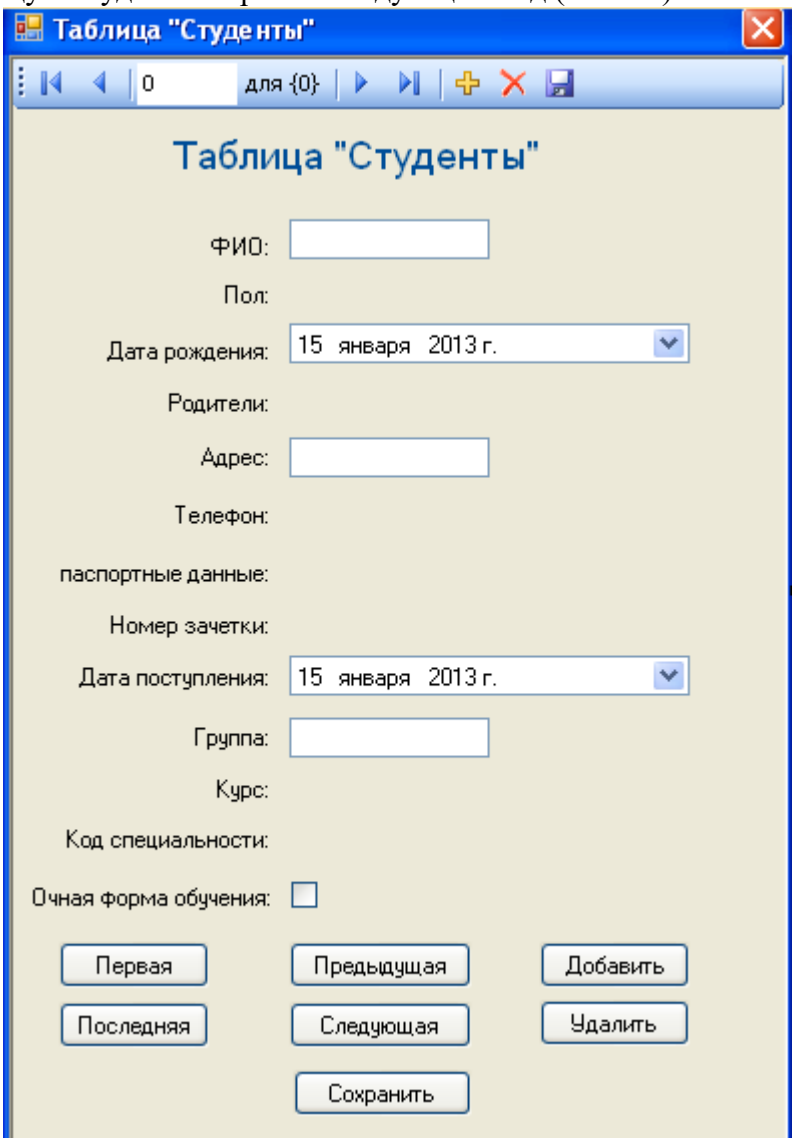



Рис.9.4

Для отображения полей «Телефон», «Паспортные данные» и «Номер зачётки» будем использовать текстовые поля ввода по маске (MaskedTexBox). 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 MaskedTextBox . Создайте текстовые поля ввода по маске справа от надписей «Телефон»,

«Паспортные данные» и «Номер зачётки».

Теперь у созданных объектов настроим маски ввода. Начнём с объекта, отображающего номер зачётки. На форме выделите соответствующее полю «Номер зачётки» текстовое поле ввода по маске. Для задания маски в меню действий с объектом выберите пункт «Установка маски».

После выбора пункта «Установка маски» на экране появится окно задания маски «Маска ввода» (Рис.9.5).

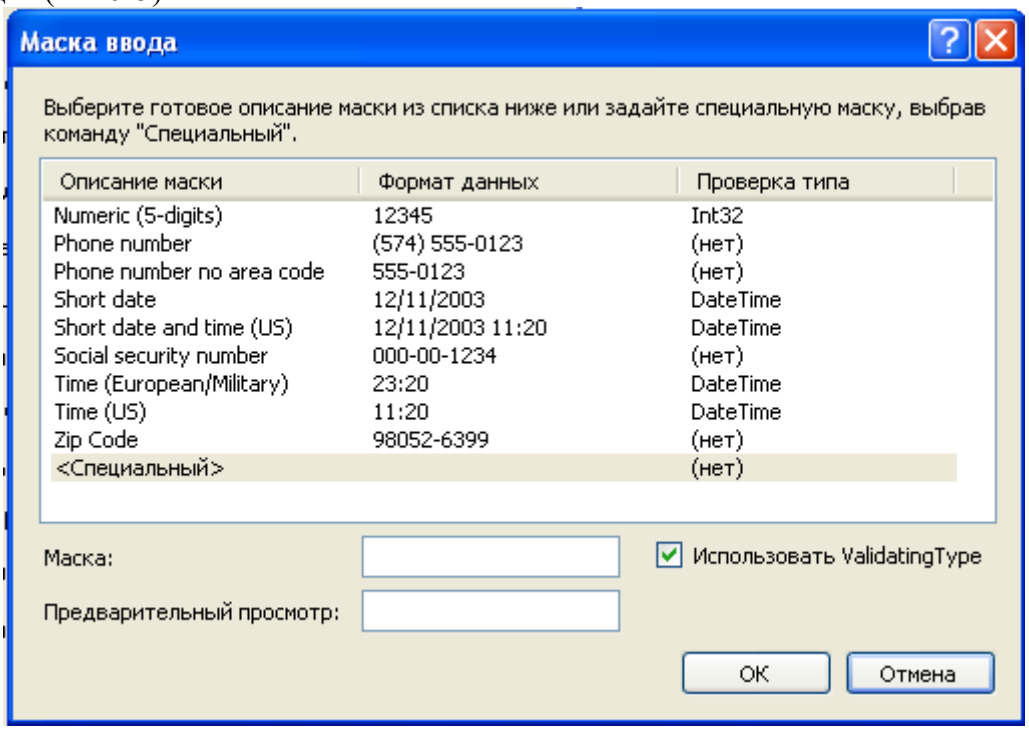



Рис.9.5.

В окне «Маска ввода» выберите маску «Numeric (5-digits)» (Числовое (5-цыфр)) и нажмите кнопку «Ok» (Рис.9.5).

Для текстового поля ввода по маске для поля «Паспортные данные» задайте маску как показано на рисунке 9.6.

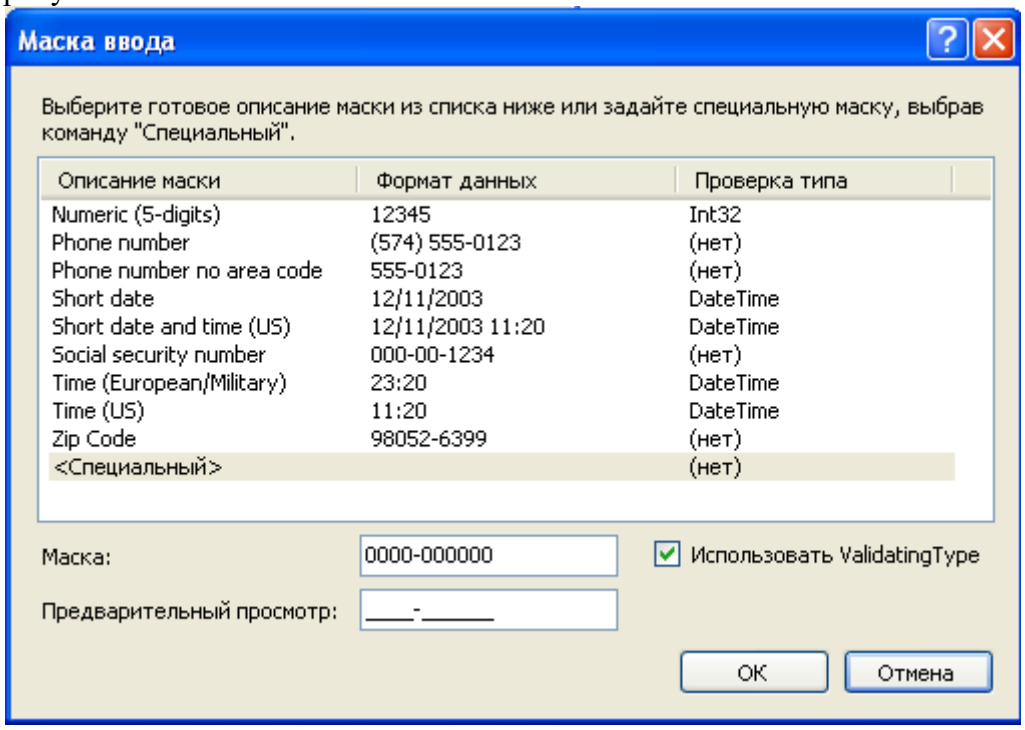


Рис.9.6

Замечание: Обратите внимание, что паспортные данные отображаются как четыре числа, тире, шесть чисел. То есть в поле «Маска» нужно задать «0000-000000».

Знак «0» обозначает цифру. В поле «Предварительный просмотр» отображается вид текстового поля ввода по маске на форме.

После определения маски для поля «Паспортные данные» в окне «Маска ввода» нажмите кнопку «Ok».

Теперь зададим маску для текстового поля ввода по маске отображающего поле «Телефон». Задайте маску как показано на рисунке 9.7.

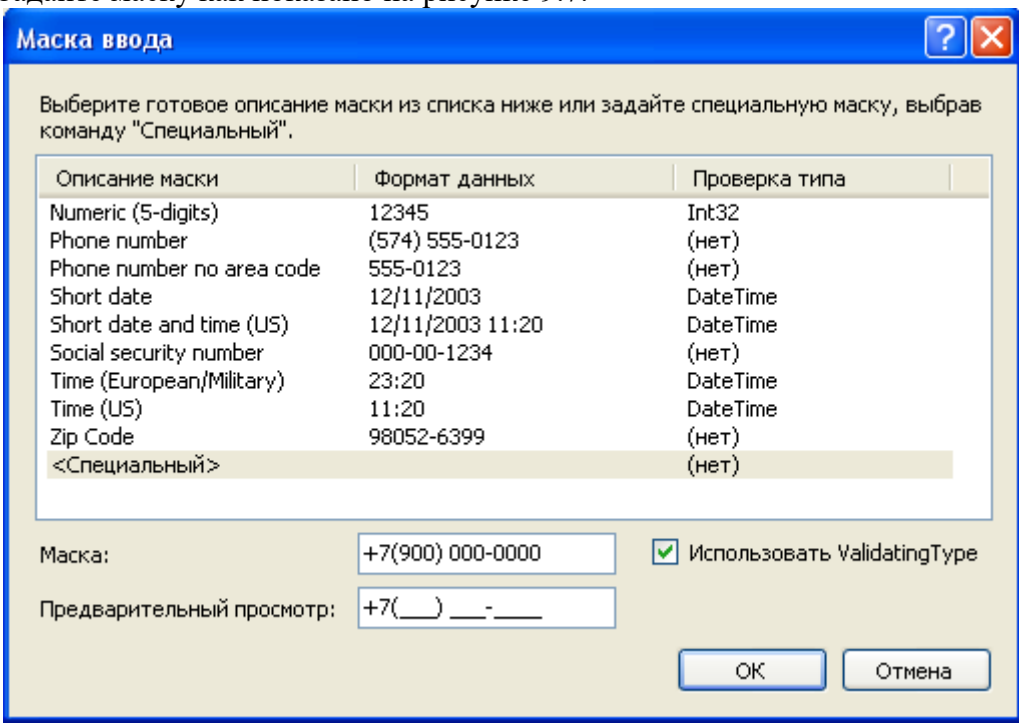



Рис.9.7

Теперь нам необходимо подключить созданные текстовые поля ввода по маске к соответствующим полям. Для этого с панели «Источники данных» (DataSources) перетащите поле «Номер зачётки» на текстовое поле ввода по маске, расположенное справа от надписи «Номер зачётки». 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 счётчика (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»

Затем перетащите поле на форму мышью, расположив, его справа от надписи «Курс».

Замечание: После перетаскивания поля «Курс» на форму слева от него появится ещё одна надпись «Курс». Удалите ее, щёлкнув 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»

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 и нажмите кнопку в свойстве «Изменить

элементы». Появится окно «Редактор коллекции строк» (Рис.9.8).

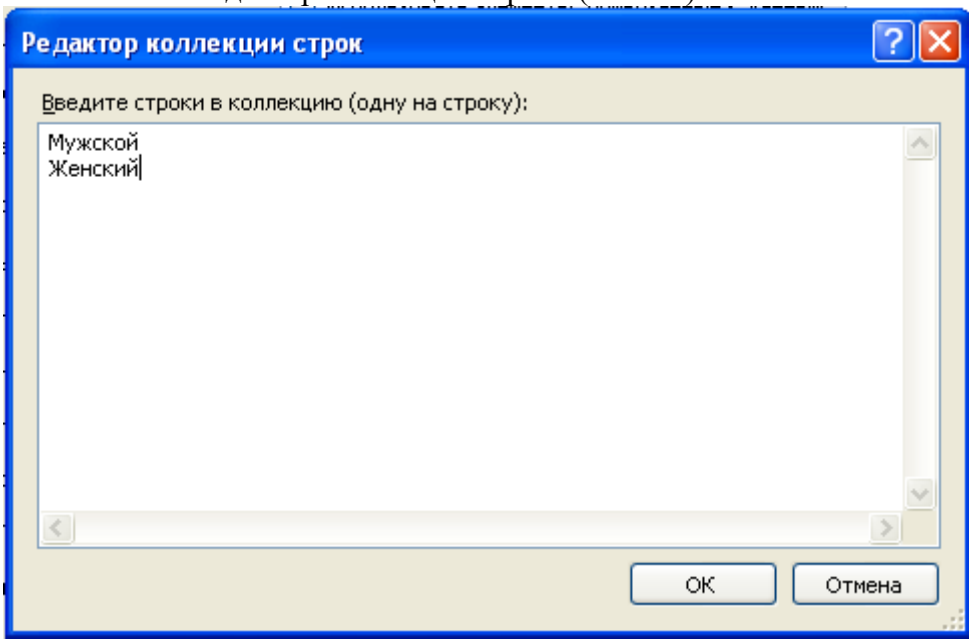



Рис.9.8

В появившемся окне в отдельных строках наберите элементы выпадающего списка: «Мужской» и «Женский» (Рис.9.8). 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.9.9).

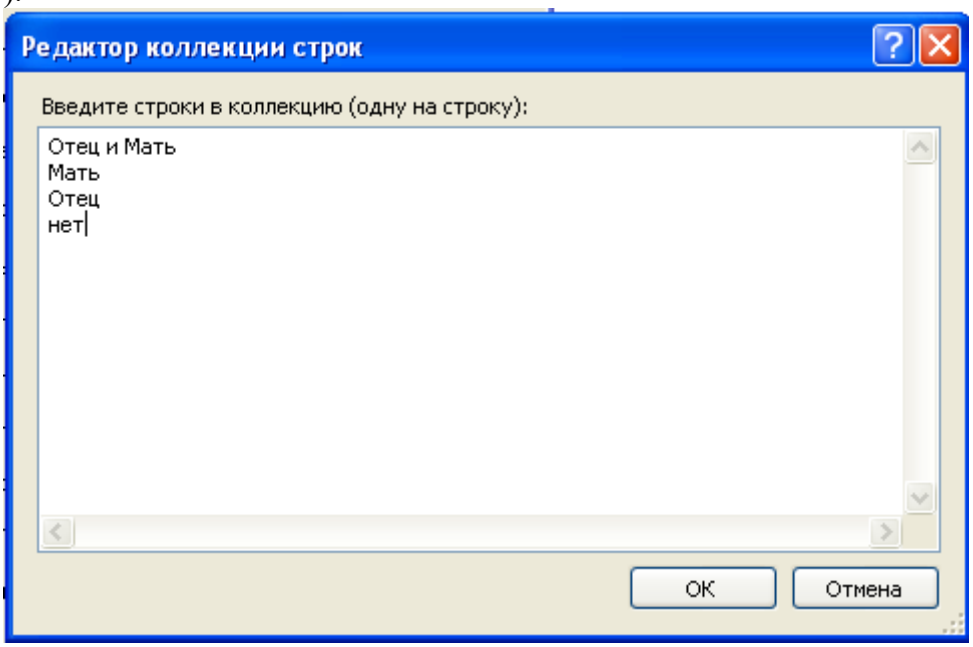



Рис.9.9

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 её код будет автоматически подставляться в поле «Код специальности» таблицы «Студенты».

Поместите справа от надписи «Код специальности», неподключённый ни к каким полям выпадающий список. Для создания выпадающего списка на панели объектов воспользуйтесь кнопкой  ComboBox .

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 «Использовать элементы, привязанные к данным» (Рис.9.10).

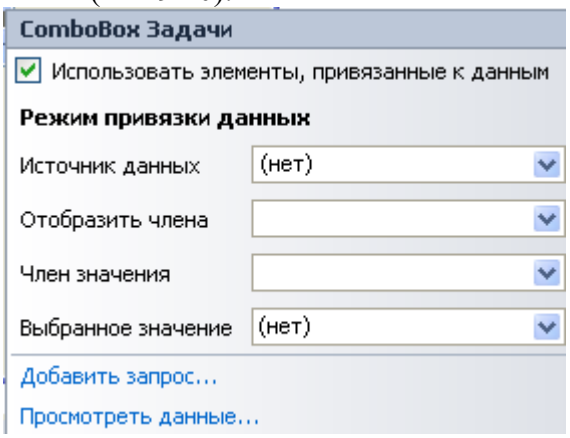



Рис.9.10

В панели действий под опцией «Использовать элементы, привязанные к данным» расположены следующие параметры:

• Источник данных – определяет таблицу или запрос из которого заполняется список;

• Отобразить члена – определяет поле значениями которого заполняется список;

• Член значений – определяет значения какого поля подставляются в связанное с выпадающим списком поле;

• Выбранное значение – определяет связанное с выпадающим списком поле.

Для изменения параметров необходимо нажать кнопку в нутрии поля параметра.

Появится древовидная структура выбора источника данных (Рис.9.11).

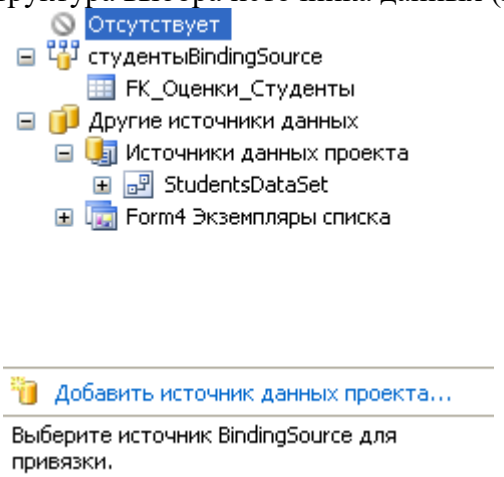



Рис.9.11

В нашем случае зададим вышеперечисленные параметры следующим образом:

• Параметр « Источник данных » задайте как «Other Data Sources\Project Data Sources\StudentsDataSet\Специальности»;

• Параметр «Отобразить члена» задайте как «Наименование специальности»;

• Параметр «Член значений» задайте как «Код специальности»;

• Параметр «Выбранное значение» задайте как «СтудентыBindingSource\Код специальности».

После задания всех вышеперечисленных параметров панель действий выпадающего списка примет вид (Рис.9.12):

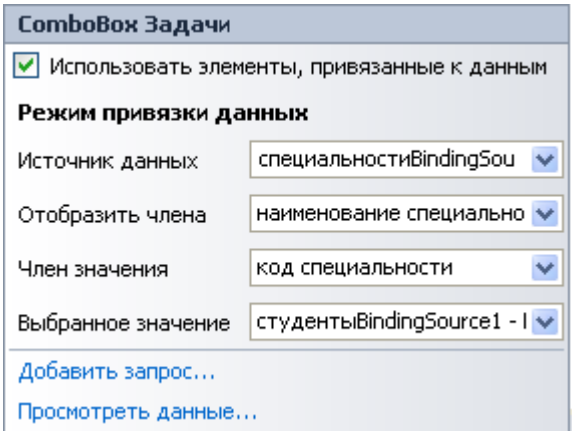



Рис.9.12

Обратите внимание на то, что на панели невидимых объектов, расположенной в нижней части рабочей области среды разработки, появилось два новых объекта: «СпециальностиBindingSource» и «СпециальностиTableAdapter»

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 рисунке 9.13

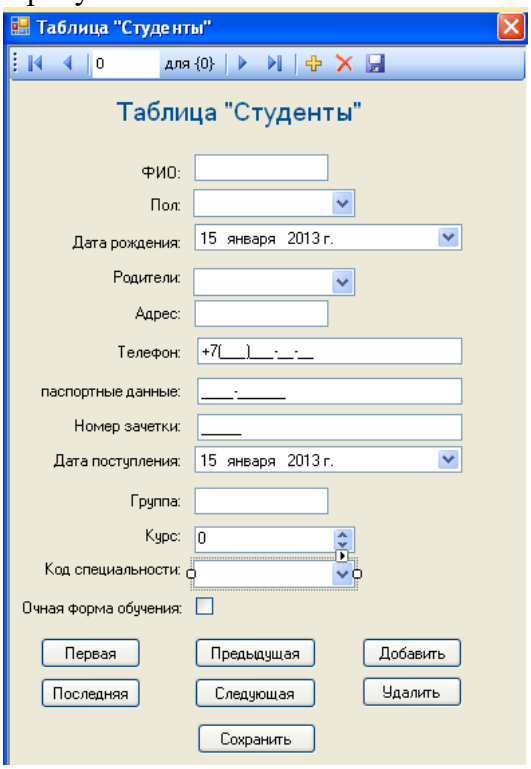



Рис.9.13

Проверим работу формы, отображающей таблицу «Студенты». Запустите проект и на главной кнопочной форме нажмите кнопку «Таблица «Студенты»». Появится форма, имеющая следующий вид (Рис.9.14):

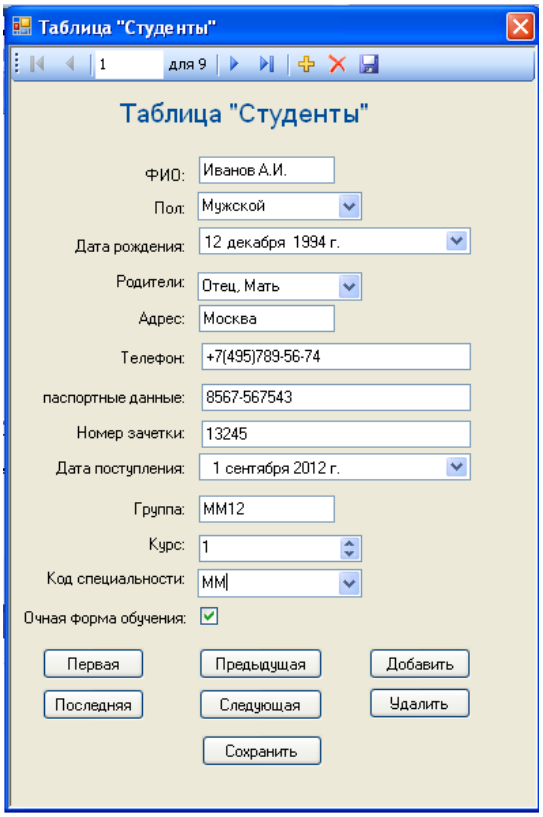



Рис.9.14

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 трёх полей:

Отобразите форму для таблицы «Оценки», щёлкнув ЛКМ по её вкладке в верхней части рабочей области среды разработки. На форму, справа от поля «Средний балл» поместите кнопку.

Задайте свойство «Text» у вновь созданной кнопки как «Вычислить» (Рис.9.15).

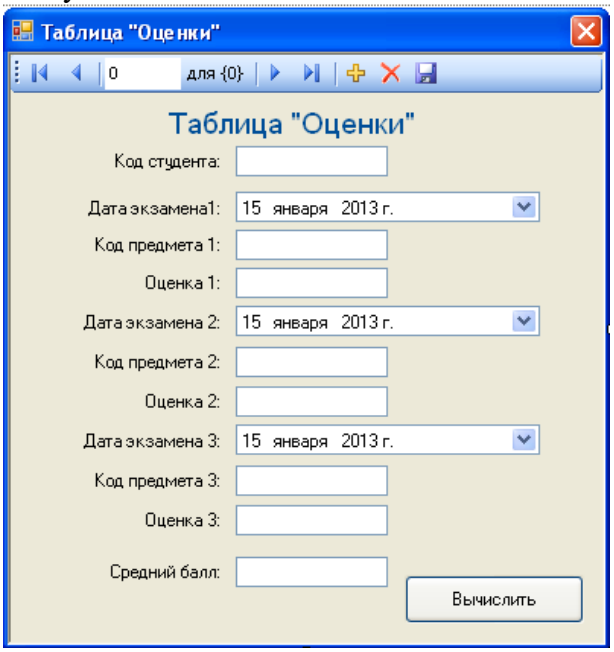



Рис.9.15

Теперь дважды щёлкните ЛКМ по кнопке «Вычислить» и в появившемся коде процедуры «Button1_Click» наберите код, представленный на рисунке 9.16, 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
```

Рис.9.16

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.9.17).

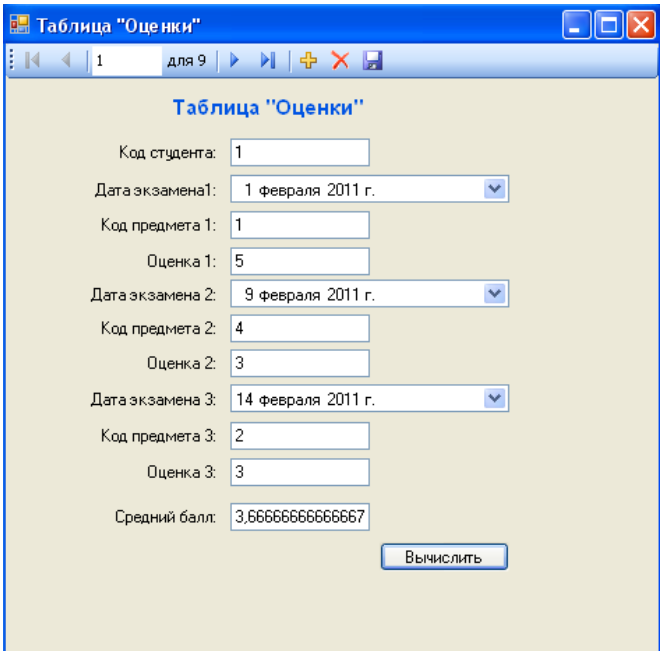



Рис.9.17

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

## Лабораторная работа № 9 Создание отчета

**Цель занятия:** научиться создавать запросы и отчёты в базах данных

Перейдём к созданию статических запросов. В обозревателе объектов «Microsoft SQL Server 2008» все запросы БД находятся в папке «Представления» (Рис 3.1).

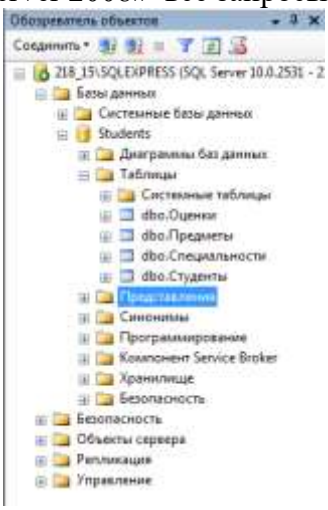



Рис. 3.1.

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» щёлкнуть ПКМ по папке «Представления», затем в появившемся меню выбрать пункт «Создать представление». Появиться окно «Добавление таблицы», предназначенное для выбора таблиц и запросов, участвующих в новом запросе (Рис.3.2).

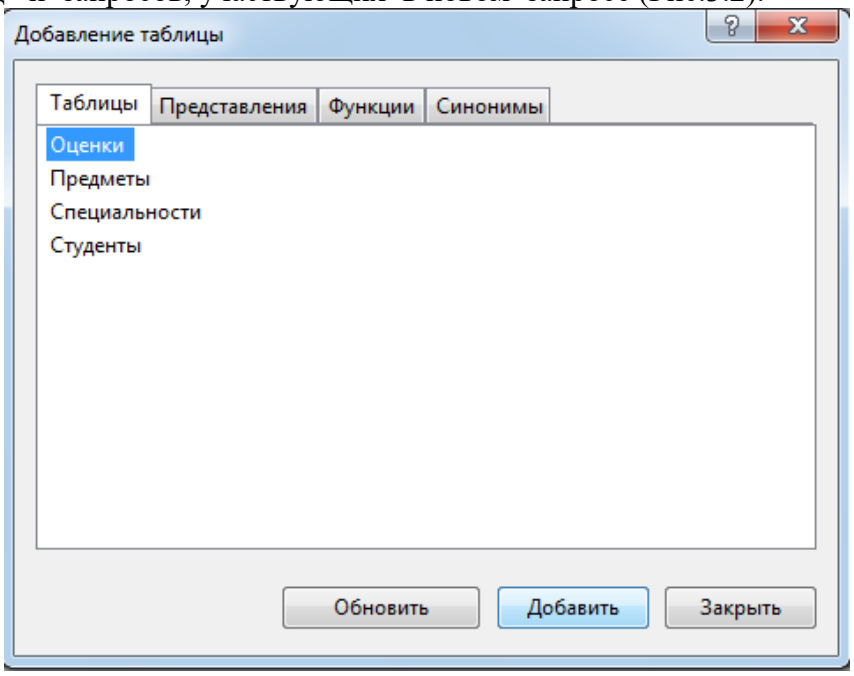



Рис.3.2.

Добавим в новый запрос таблицы «Студенты» и «Специальности». Для этого в окне «Добавление таблицы» выделите таблицу «Студенты» и нажмите кнопку «Добавить».

Аналогично добавьте таблицу «Специальности». После добавления таблиц участвующих в запросе закройте окно «Добавление таблицы» нажав кнопку «Закрыть». Появится окно конструктора запросов (Рис.3.3).

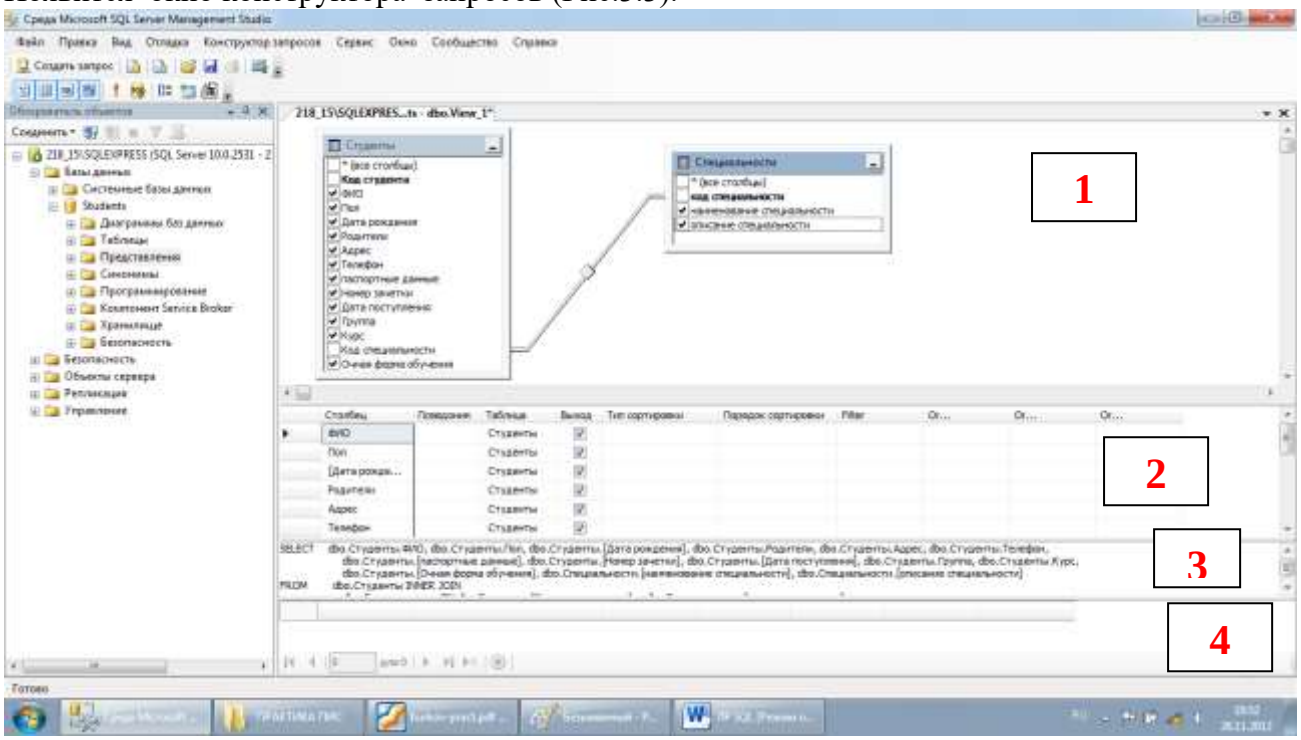


Рис.3.3.

***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 «Столбец»), позволяет задавать им псевдонимы (столбец «Псевдоним»), позволяет устанавливать тип сортировки записей по одному или нескольким полям (столбец «Тип сортировки»), позволяет задавать порядок сортировки (столбец «Порядок сортировки»), 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 «Добавление таблицы» 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 щёлкнуть ПКМ и в появившемся меню выбрать пункт «Удалить».

Теперь перейдём 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 отобразиться в виде ломаной линии соединяющей эти два поля связи (Рис.3.3).

***Замечание:*** Если необходимо удалить связь, то для этого необходимо щёлкнуть по ней ПКМ и в появившемся меню выбрать пункт «Удалить».

***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 щёлкнуть 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 щёлкните мышью по галочке.

***Замечание:*** Если необходимо отобразить все поля таблицы, то необходимо установить галочку слева от пункта «* Все поля», принадлежащего соответствующей таблице на схеме данных.

Определите отображаемые поля нашего запроса, как это показано на рисунке 3.3 (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 щёлкните ПКМ в любом месте окна конструктора запросов и в появившемся меню выберите пункт «Выполнить код SQL». Результат выполнения запроса появиться в виде таблицы в

области результата (Рис.3.3).

***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 её необходимо удалить и создать заново, как это описано выше.

Если запрос выполняется правильно, то необходимо сохранить. Для сохранения запроса закройте окно конструктора запросов, щёлкнув мышью по кнопке закрытия ✕, расположенной в верхнем правом углу окна конструктора (над схемой данных).

Появиться окно с вопросом о сохранении запроса (Рис.3.4).

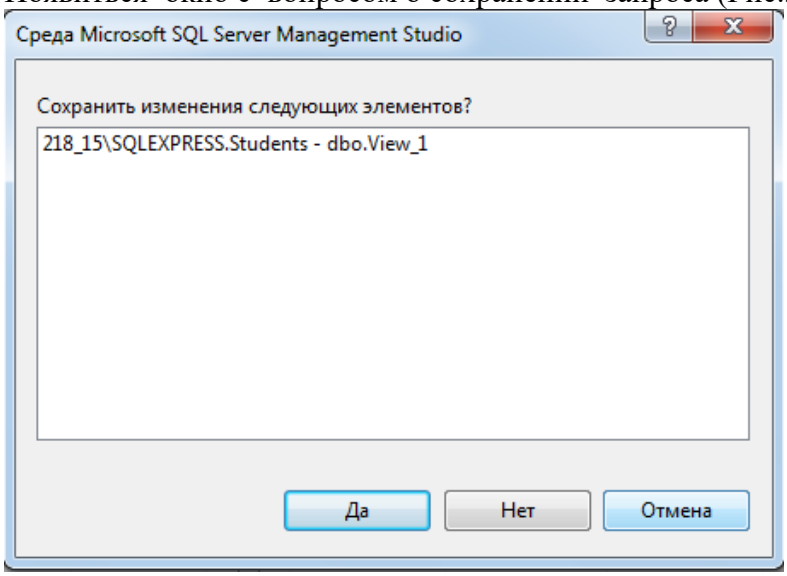



Рис.3.4

В данном окне необходимо наддать кнопку «Да». Появится окно «Выбор имени» (Рис.3.5).

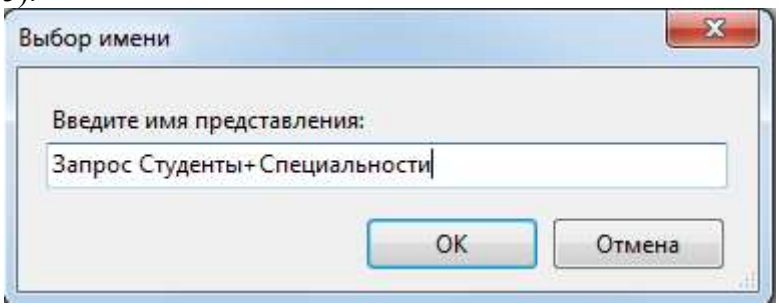



Рис. 3.5.

В данном окне зададим имя нового запроса « Запрос Студенты+ Специальности» и нажмём кнопку «Ok». Запрос появиться в папке «Представления» БД «Students» в обозревателе объектов (Рис.3.6).

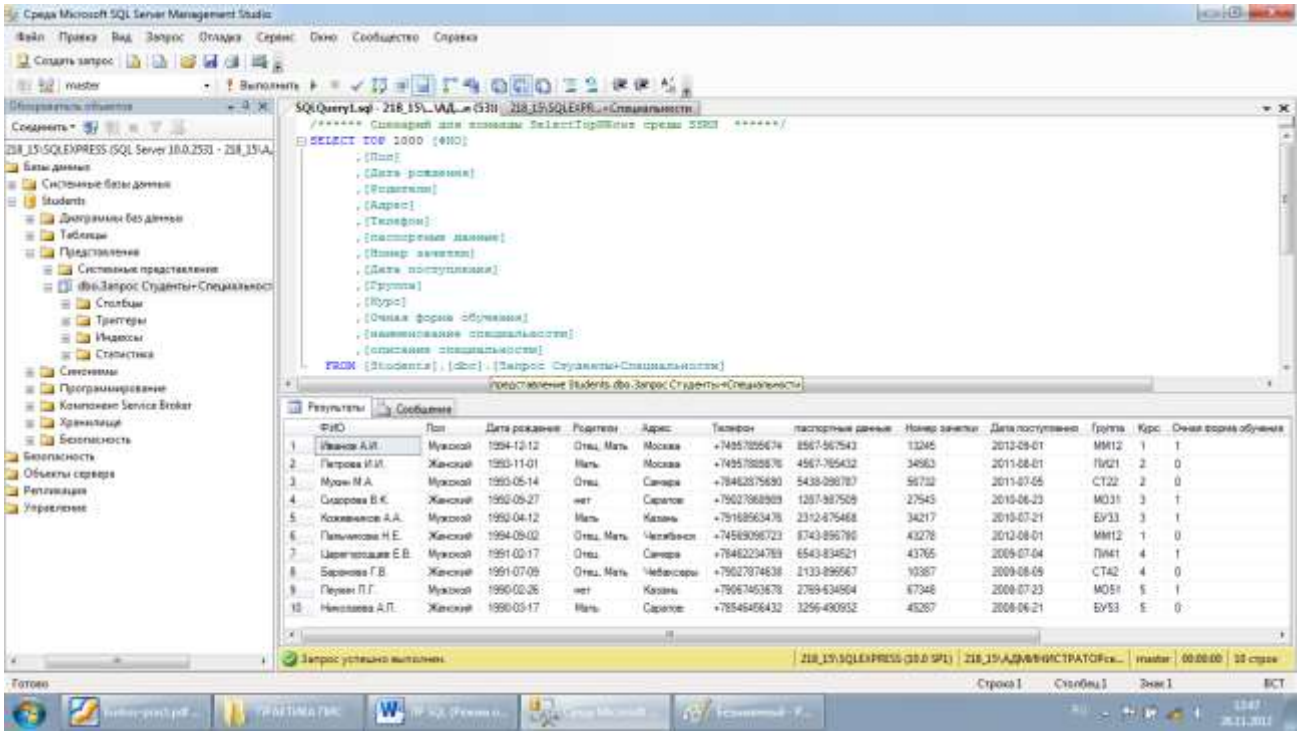


Рис.3.6

Проверим работоспособность созданного запроса вне конструктора запросов.

Запустим вновь созданный запрос «Запрос Студенты+ Специальности» без использования конструктора запросов. Для выполнения уже сохранённого запроса необходимо щёлкнуть ПКМ по запросу и в появившемся меню выбрать пункт «Отобразить первые 1000 записей». Выполните эту операцию для запроса «Запрос Студенты+ Специальности». Результат представлен на рисунке 3.6.

Перейдём к созданию запроса «Запрос Студенты+ Оценки». В обозревателе объектов в БД «Students» щелкните ПКМ по папке «Представление», затем в появившемся меню выберите пункт «Создать представление». Появиться окно «Добавление таблицы» (Рис.3.2).

В запросе «Запрос Студенты+ Оценки» мы связываем таблицы «Студенты» и «Оценки» по полям связи «Код студента». Следовательно, в окне «Добавление таблицы» 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 трём полям. То есть поля «Код предмета 1», «Код предмета 2» и «Код предмета 3» таблицы «Оценки» связаны с полем «Код предмета » таблицы «Предметы». По этому 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 «Добавление таблицы», появится окно конструктора запросов.

В окне конструктора запросов установите связи между таблицами и определите отображаемые поля, как показано на рисунке 3.7.

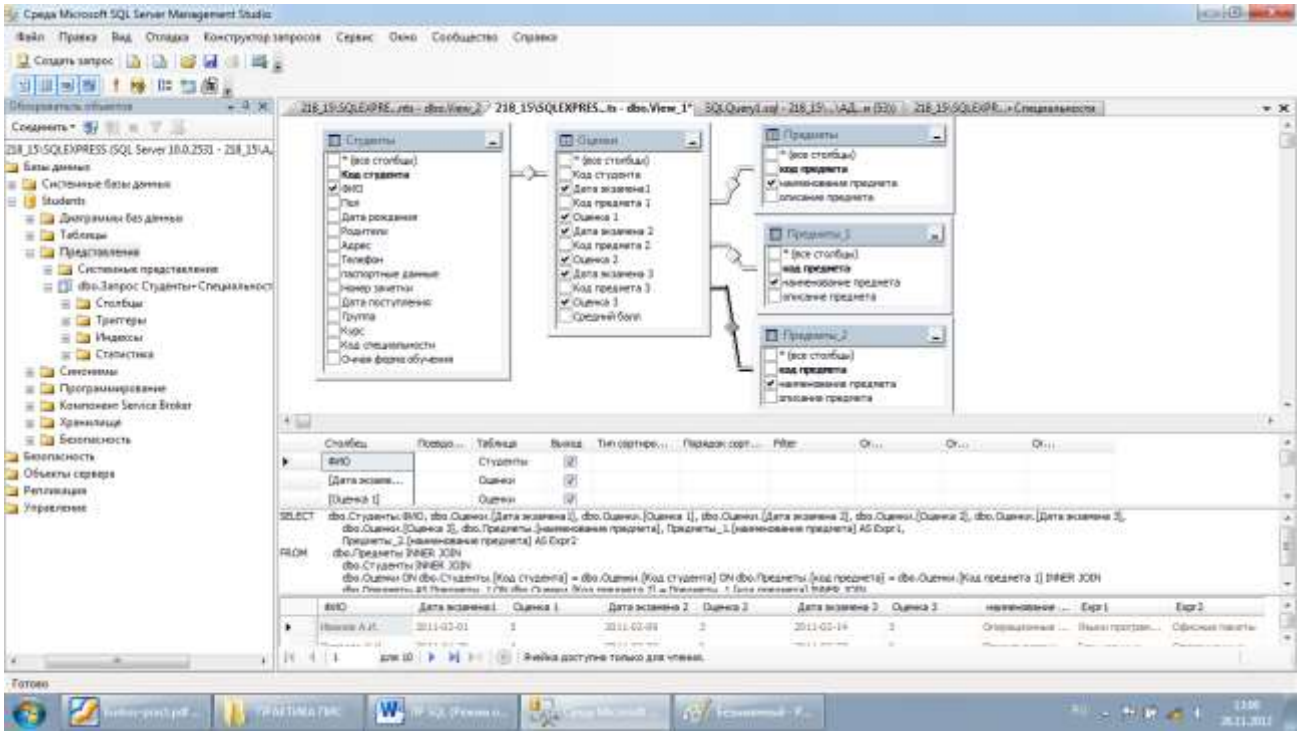
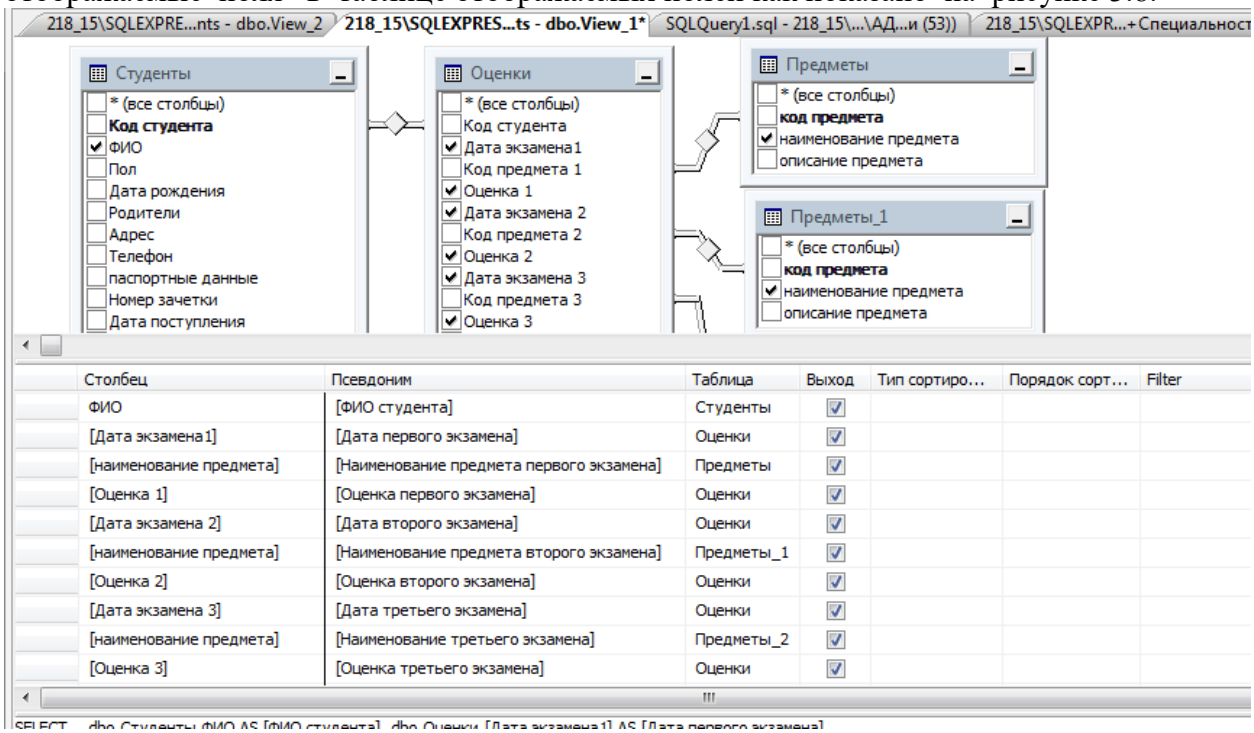


Рис.3.7

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 «Псевдоним»). Расположите отображаемые поля в таблице отображаемых полей как показано на рисунке 3.8.




| Столбец | Псевдоним | Таблица | Выход | Тип сортиро... | Порядок сорт... | Filter |
|---|---|---|---|---|---|---|
| ФИО | [ФИО студента] | Студенты | ☑ | | | |
| [Дата экзамена1] | [Дата первого экзамена] | Оценки | ☑ | | | |
| [наименование предмета] | [Наименование предмета первого экзамена] | Предметы | ☑ | | | |
| [Оценка 1] | [Оценка первого экзамена] | Оценки | ☑ | | | |
| [Дата экзамена 2] | [Дата второго экзамена] | Оценки | ☑ | | | |
| [наименование предмета] | [Наименование предмета второго экзамена] | Предметы_1 | ☑ | | | |
| [Оценка 2] | [Оценка второго экзамена] | Оценки | ☑ | | | |
| [Дата экзамена 3] | [Дата третьего экзамена] | Оценки | ☑ | | | |
| [наименование предмета] | [Наименование третьего экзамена] | Предметы_2 | ☑ | | | |
| [Оценка 3] | [Оценка третьего экзамена] | Оценки | ☑ | | | |

Рис.3.8.

Задайте псевдонимы для каждого из полей, просто записав псевдонимы в столбце «Псевдоним» таблицы отображаемых полей, как на рисунке 3.8.

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 «Введите имя представления» задайте имя нового запроса «Запрос Студенты+ Оценки» (Рис.3.9).

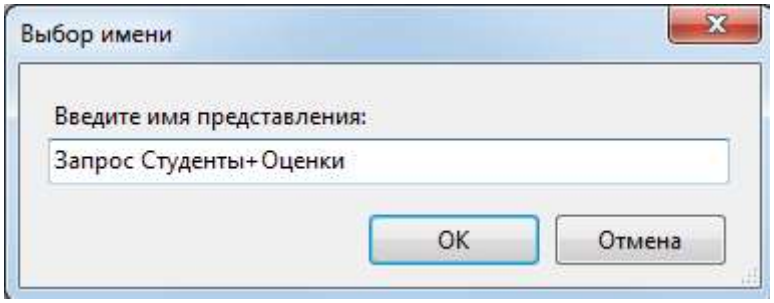



Рис.3.9

Проверьте работоспособность нового запроса вне конструктора . Для этого запустите запрос. Результат выполнения запроса « Запрос Студенты+ Оценки» должен выглядеть как на рисунке 3.10

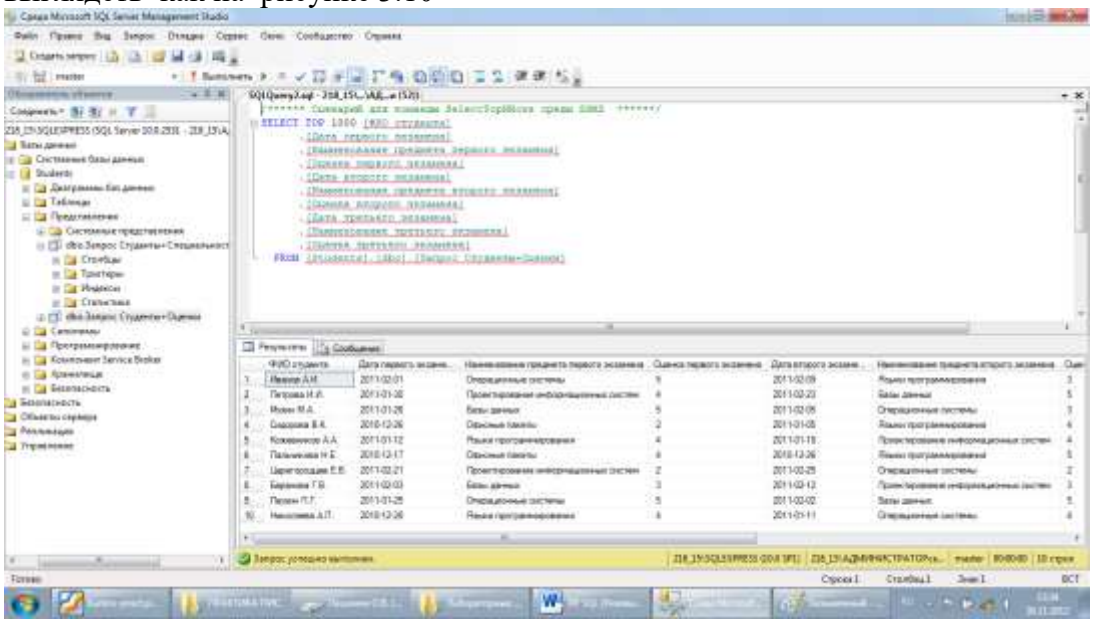


Рис.3.10

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 «Добавления таблицы» перейдите на вкладку «Представления» и добавьте в новый запрос «Запрос Студенты+ Специальности» (Рис.3.11). Затем закройте окно «Добавления таблицы»

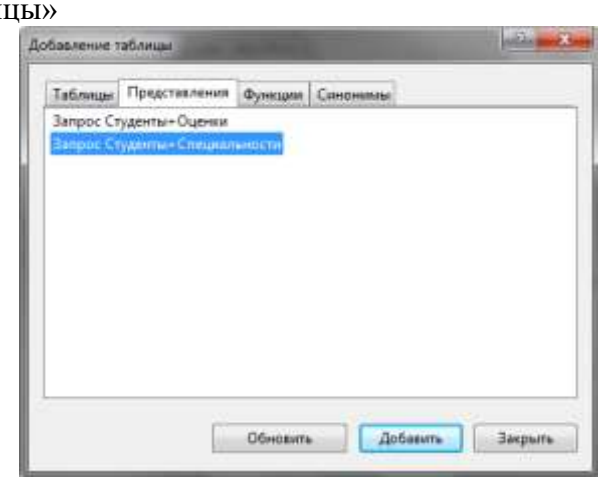



Рис.3.11

В появившемся окне конструктора запросов определите в качестве отображаемых полей все поля запроса «Запрос Студенты+Специальности» (Рис.3.12).

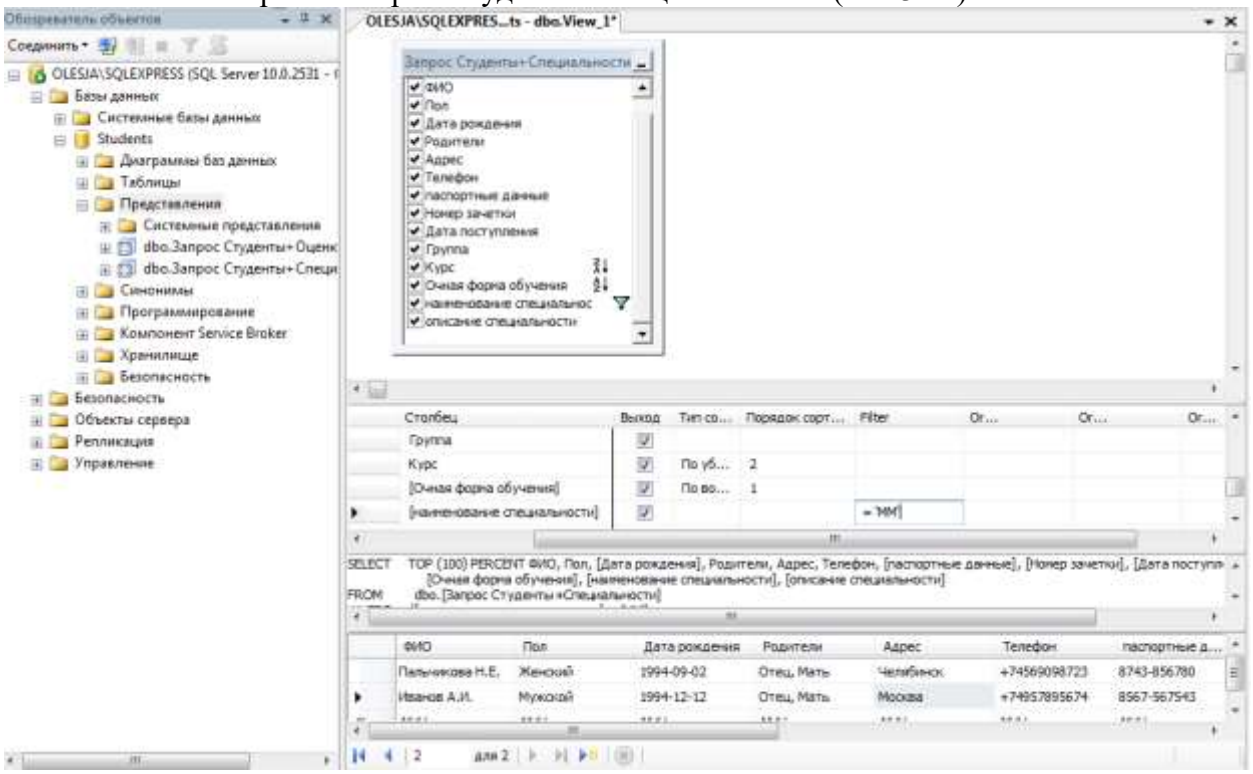


Рис.3.12

*Замечание:* Для отображения всех полей запроса, в данном случае, мы не можем использовать пункт «* (все столбцы)». 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.3.12).

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 «Тип сортировки», задайте «По возрастанию» , а в строке для поля «Курс» - задайте «По убыванию». Для определения порядка сортировки для поля «Очная форма обучения» в столбце «Порядок сортировки) поставьте 1, а для поля «Курс» поставьте 2 (Рис.3.12). 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 рисунке 3.12.

Закройте окно конструктора запросов. В качестве имени нового фильтра в окне «Выбор имени» задайте «Фильтр ММ» (Рис.3.13) и нажмите кнопку «Ok».

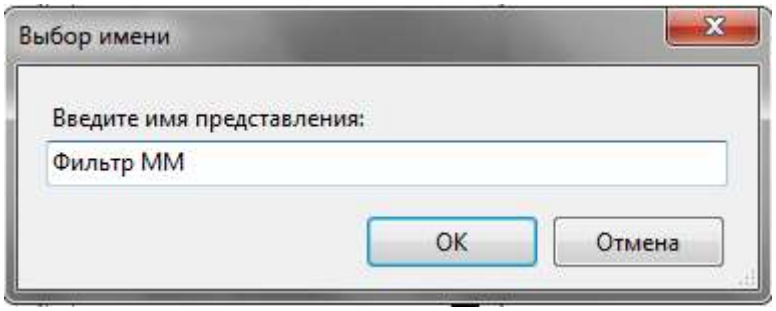



Рис.3.13

Фильтр «Фильтр ММ» появиться в обозревателе объектов. Выполните созданный фильтр вне окна конструктора запросов. Результат должен быть таким же как на рисунке 3.14.

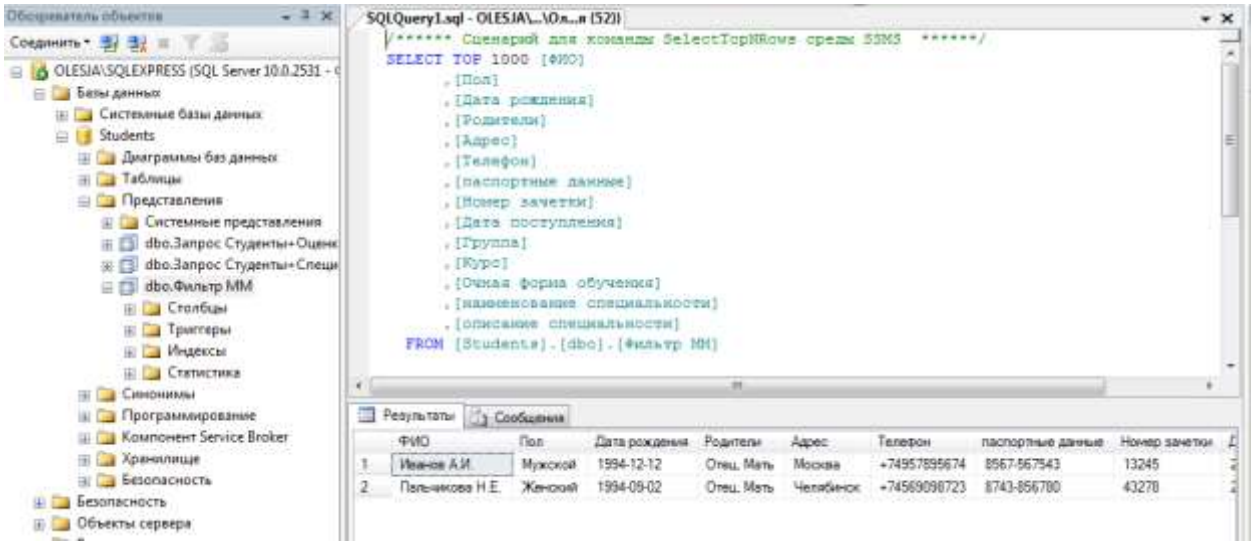



Рис.3.14

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 задаём 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.3.11). После закрытия окна «Добавление таблицы» сделайте отображаемыми все поля запроса (Рис.3.15).

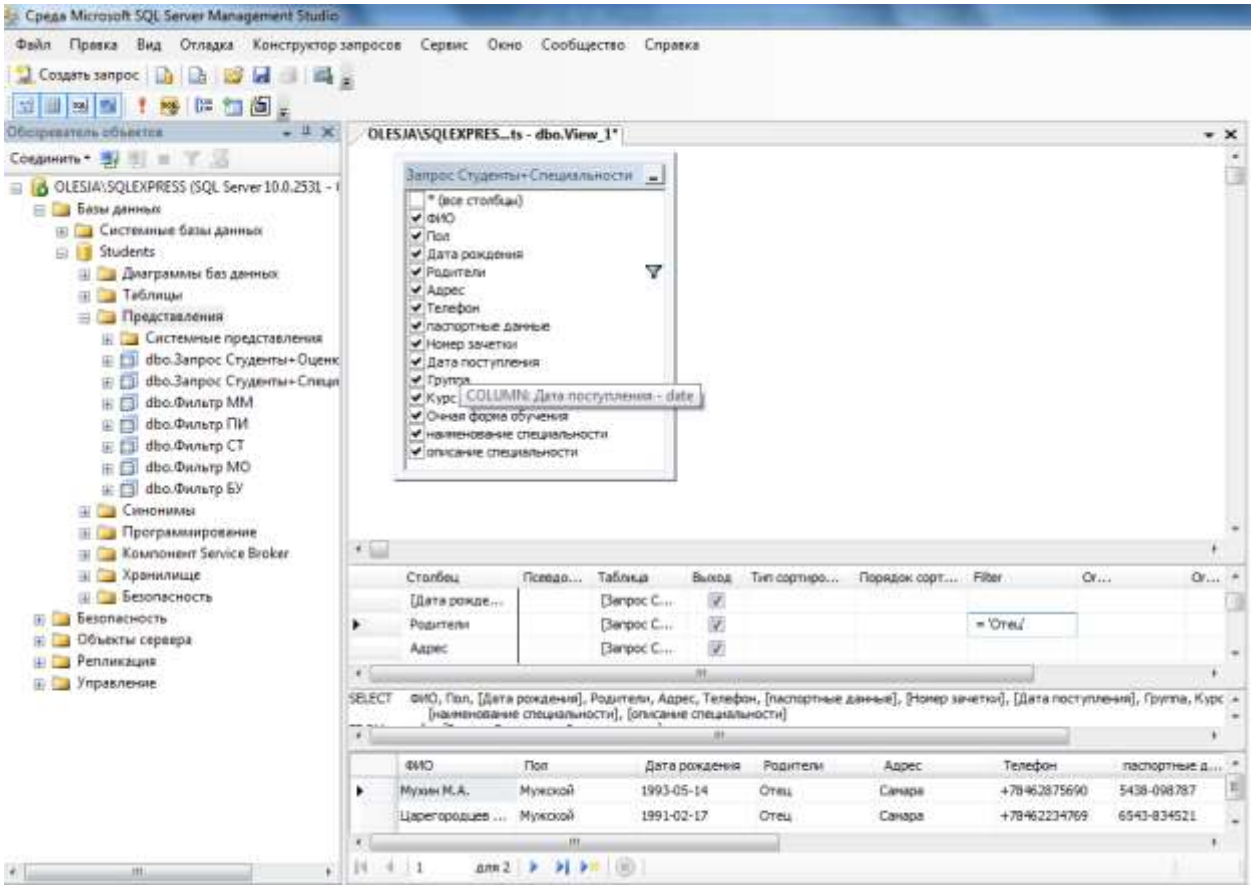


Рис.3.15

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 рисунке 3.15.

Закройте окно конструктора запросов. В окне «Выбор имени» задайте имя нового фильтра как «Фильтр Отец».

Выполните фильтр «Фильтр Отец» вне конструктора запросов. Результат должен быть аналогичен рисунку 3.16.

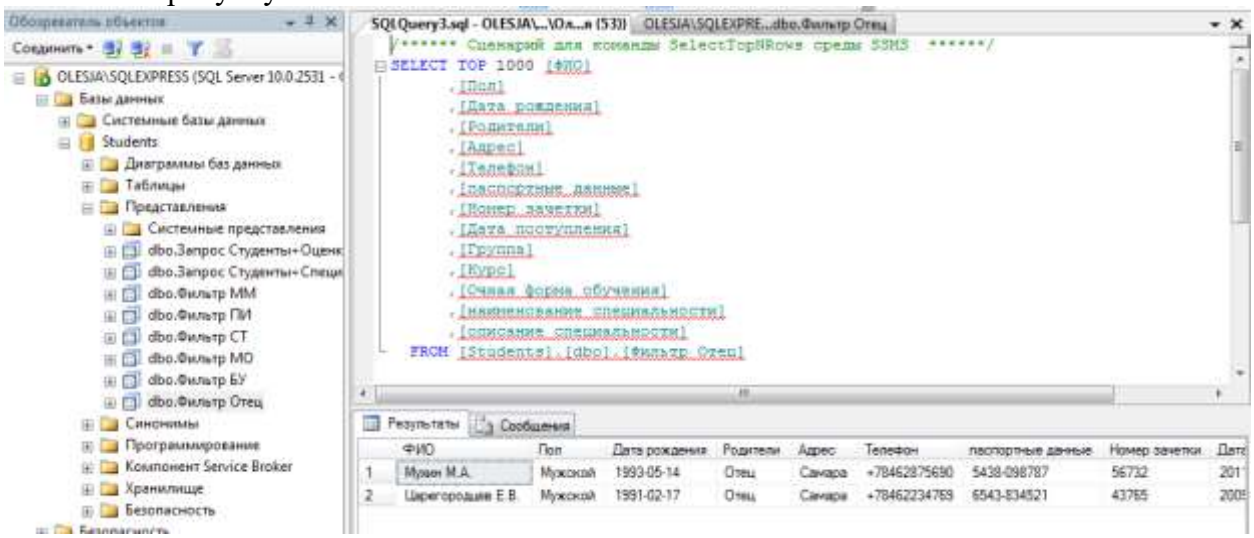


Рис.3.16

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 задаём 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. Начнём с очной формы обучения. Создайте новый запрос и добавьте в него запрос «Запрос Студенты+Специальности». Как и ранее сделайте все поля запроса отображаемыми (Рис.3.17).

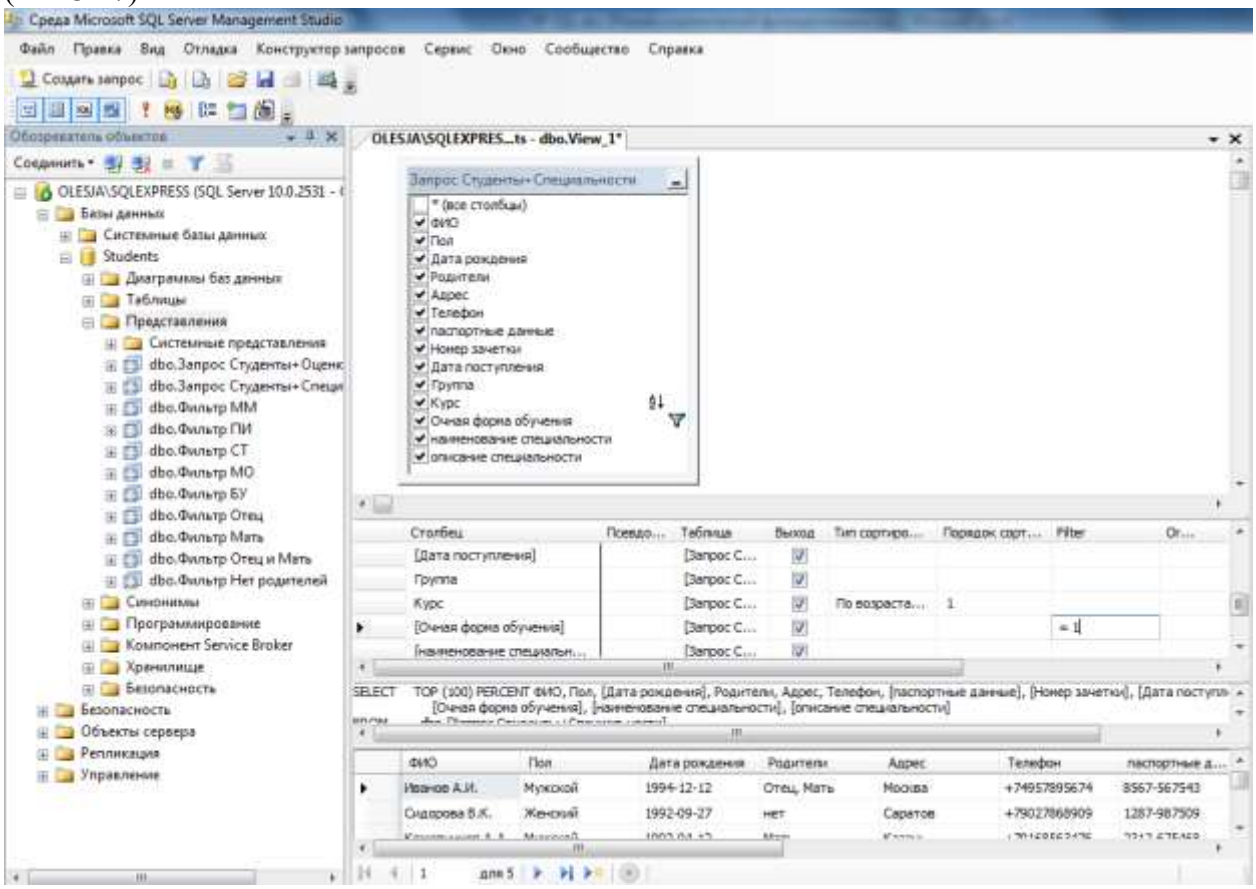



Рис.3.17

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 «Тип сортировки», значение «По возрастанию».

Проверьте работу фильтра, выполнив его. После выполнения фильтра окно конструктора запросов должно выглядеть точно также как на рисунке 3.17.

Закройте окно конструктора запросов. Сохраните фильтр под именем «Фильтр очная форма обучения».

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 рисунке 3.18.

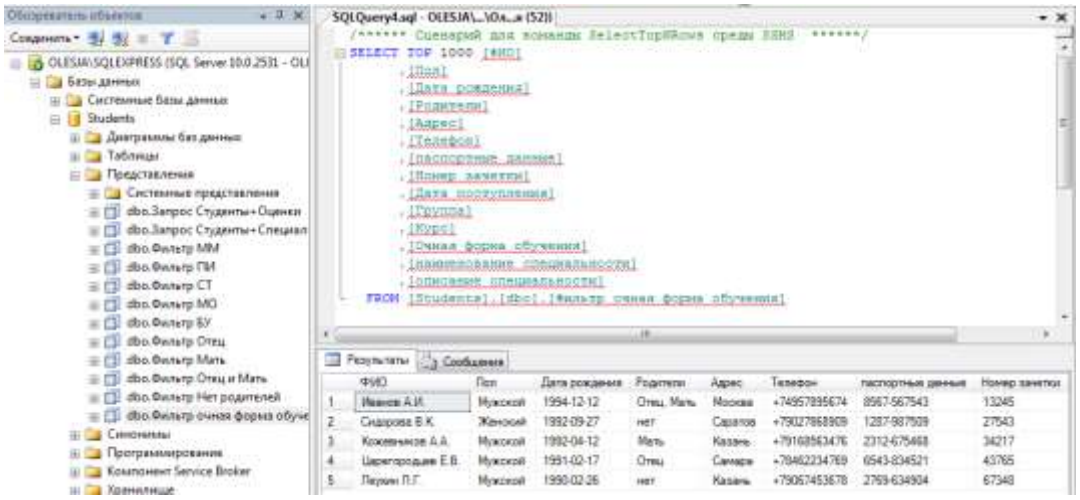


Рис.3.18

Самостоятельно создайте фильтр для отображения студентов заочной формы обучения. Данный фильтр создаётся 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.3.19):

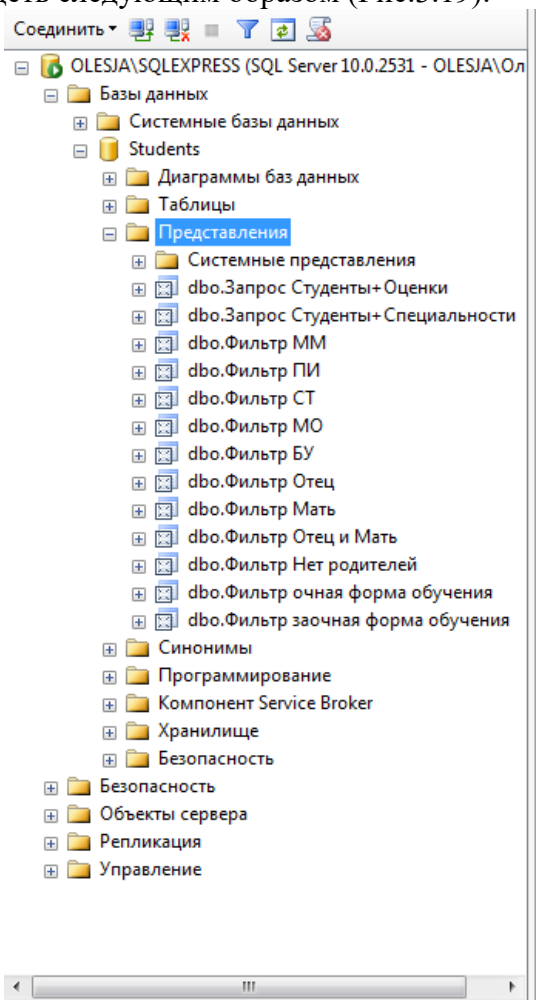



Рис.3.19

**Лабораторная работа № 10** Создание подчиненного отчета. Вычисления в отчетах

**Цель занятия:** Научиться создавать формы отчетных документов по данным БД

Термин «отчет» понимается в информационных системах шире, чем это традиционно принято. Под отчетом здесь понимается любой выходной документ: список (например, сотрудников), письмо, адрес, печатающийся на конверте (почтовая этикетка), отчет в традиционном понимании этого слова. Создание отчетов (выходных документов) является одной из наиболее важных функций информационных систем.

Чаще всего, когда отчеты обсуждаются в контексте баз данных, речь идет об извлечении информации из каких-либо источников (однотипных или разнотипных) и представлении их в виде, удобном для дальнейшего восприятия и анализа. Это делает рассматриваемые вопросы близкими к проблематике OLAP-систем[1].

Для создания отчетов используются высокоуровневые средства автоматизации - генераторы отчетов. Генераторы отчетов, так же как и генераторы форм ввода-вывода, являются компонентами языков 4-го поколения. Они включены в состав большинства СУБД. Кроме того, генераторы отчетов представлены и как самостоятельный класс программного обеспечения. Существует даже англоязычный термин «reporting», объединяющий все вопросы, относящиеся к процессу получения отчетов.

Генераторы отчетов представляют неотъемлемый элемент концепции Business Intelligence.

Business Intelligence или сокращенно BI (бизнес-анализ, бизнес-аналитика) - это инструменты, используемые для преобразования, хранения, анализа, моделирования, доставки и трассировки информации в ходе работы над задачами, связанными с принятием решений на основе фактических данных. При этом с помощью этих средств лица, принимающие решения, должны при использовании подходящих технологий получать нужные сведения и в нужное время. Помимо отчётности в BI входят инструменты интеграции и очистки данных (ETL), аналитические хранилища данных и средства Data Mining.

BI-технологии позволяют анализировать большие объёмы информации, заостряя внимание пользователей лишь на ключевых факторах эффективности, моделируя исход различных вариантов действий, отслеживая результаты принятия тех или иных решений.

Среда Business Intelligence Development Studio входит в состав используемого в лабораторных работах MS SQL Server 2008.

Источниками для получения отчетов могут быть не только таблицы баз данных, но и запросы, а также записи, отобранные с помощью фильтров. Некоторые генераторы отчетов позволяют проводить отбор данных, включаемых в отчет, непосредственно пользуясь средствами самого генератора отчетов.

**Отчеты позволяют выполнять следующие действия:**

- проводить группировку данных;
- вычислять многоуровневые промежуточные и общие итоги по отдельным полям;вводить в отчеты вычисляемые поля;
- выводить в отчеты данные из разных источников;
- включать в отчеты данные, отобранные по заданным критериям;

Отчеты имеют много общего с формами. Однако отчеты в отличие от форм не предназначены для ввода и правки данных в таблицах. Они позволяют только выводить данные в различном виде. Вывод отчета может быть осуществлен на экран, на печать, а также в файл. Чаще всего отчеты используются для вывода информации на печать (т.е. для получения так называемых твердых копий). Для документов, даже одинаковых по содержанию, могут использоваться разные приемы их оформления в зависимости от того, куда осуществляется вывод информации. Например, при выдаче информации на экран могут использоваться специальные эффекты (мигание, динамические изображения, полосы

прокрутки и т. п.).

Различают отчеты анкетной и табличной формы. При анкетной форме данные об одном объекте (сотруднике, товаре и т. п.) обычно размещаются один под другим, причем слева указывается название атрибута (поля), а справа - его значение. После вывода информации об одном объекте выводится информация о следующем объекте, как в форме с простой привязкой - см. рис. 6.14.

Документы табличной формы включают в себя привычные таблицы с названиями атрибутов в заголовках столбцов; данные о каждом объекте представляются в одной строке, как в форме со сложной привязкой - см. выше рис. 6.18.

Табличные документы могут включать либо одну таблицу - однотабличные документы, либо несколько таблиц (обычно разной структуры) - многотабличные документы.

В зависимости от объема и выбранного способа оформления документы могут занимать одну страницу (одностраничные) или несколько страниц (многостраничные).

В любом документе могут быть выделены самостоятельные разделы (зоны, «полосы», области), выполняющие разную смысловую нагрузку. В реальном документе те или иные из перечисленных ниже зон могут отсутствовать. Различают зоны, относящиеся целиком к документу; это заголовок отчета и примечание документа («итоговая» зона, оформительская часть). Заголовок отчета и примечание отчета могут размещаться на первой и последней страницах отчета соответственно либо выноситься на отдельные листы.

Для каждой страницы выделяют верхний и нижний колонтитулы. Ну и, естественно, главное место принадлежит области данных. В этой области размещаются данные из БД.

Кроме того, в документе может быть обеспечена группировка данных, причем группировка может быть одноуровневой и многоуровневой (иерархической). В последнем случае для каждого уровня группировки могут быть созданы зоны заголовка и примечания группы.

Группировка обычно используется в целях подсчета каких-либо итоговых показателей для каждой группы (суммы, количества элементов в группе и т.п.). При этом возможно получение итоговых документов, включающих только итоговые значения, детальных документов, имеющих только детальные строки, и смешанных, содержащих как детальные строки, так и итоговые.

Отчеты с подведением итогов могут быть использованы и при создании документов анкетной формы. Например, при выдаче в анкетной форме сведений о сотрудниках в конце документа можно подсчитать общее количество сотрудников. Но поскольку наиболее часто эти возможности используются в табличных документах, то соответствующие классификационные группировки отнесены именно к этому видудокументов.

Источниками информации для отчетов могут быть либо реальные таблицы базы данных, либо предварительно созданные запросы, отбирающие информацию, выводимую в отчет. Кроме того, в отчет могут включаться вычисляемые поля. Вычисляемые поля, как, впрочем, и реальные поля БД, могут входить в любую зону документа.

В последнее время в отчеты, наряду с символьной информацией, часто включается деловая графика.

Кроме документов, содержащих главным образом фактографическую информацию из баз данных, можно создавать и документы, которые в основном, напротив, включают какой-то текст (документографические), в который вкраплены данные из БД (документы типа письма).

В документах фактографического типа можно различать просто какой-то текст, не имеющий жесткой связи с элементами БД (например, название документа, поясняющий текст), названия элементов из БД (например, «Фамилия») и значения этих элементов (например, Иванов, Петров), элементы оформления (линия, рисунки).

Генераторы отчетов разных СУБД различаются по своим возможностям и особенностям выполнения идентичных функций. Так, например, в некоторых системах в итоговую зону документа по умолчанию включаются суммарные величины всех числовых полей, включенных в отчет.

Из генераторов отчетов, встроенных в СУБД или среды разработки, можно назвать, например, генераторы отчетов Microsoft Access и 1С:Предприятие. Аналогичные инструменты имеются в Delphi и других средах разработки.

Далее рассмотрим наиболее известные самостоятельные генераторы отчетов.

**Crystal Reports**

Среди генераторов отчетов - самостоятельных программ самым мощным генератором отчетов, ставшим де-факто стандартом, является Crystal Reports.

Возможности Crystal Reports:

поддержка очень большого количества источников данных, в том числе нестандартных (логи Web-сервера, журналы событий и т.п.);

- развитая среда программирования, встроенная библиотека функций; интегрированы многие продукты прочих фирм - для создания диаграмм, географических карт и т. п.;
- исключительно богатые возможности оформления (в том числе и условного - в зависимости от значений);
- экспорт отчетов в самые разнообразные форматы (документов Office, XML, компилируемые файлы, HTML, PDF) - в том числе и динамические, реагирующие на изменения на источнике данных;
- интеграция с MS Visual Studio и готовые объекты для использования в пользовательских приложениях. Crystal Reports поставляется в составе Visual Studio;
- встраивание отчетов, сгенерированных в Crystal Reports, в клиентские части ИС, созданные при помощи других средств разработки (Power Builder, Delphi, Centura и т.п.).

Средства Crystal Reports интегрированы в продукты более чем 250 производителей программного обеспечения, в том числе Microsoft, IBM, Lotus, SAP, PeopleSoft, Computer Associates, и активно используется разработчиками в России (например, Crystal Reports - основное средство создания отчетов в программных пакетах Parus, Scala, Platinum, ExactSoftware и т.п.).

**FastReport**

Достойным представителем семейства генераторов отчетов является продукция Российской компании Fast Reports Inc. - генератор отчетов FastReport. Он позволяет генерировать отчеты из программ, написанных на Visual C++, Visual Basic, C#, Microsoft Access, Delphi и др. Иерархия включает компоненты управления отчетами, страницами, запросами к базам данных, а также набор компонентов для построения интерфейсов интерактивных отчетов. Благодаря этим компонентам, создаваемое приложение может генерировать отчеты как «самостоятельно», так и с использованием дизайнера отчетов FastReport Designer во время выполнения.

Перечень отчетов, которые способен генерировать FastReport Studio включает как простые отчеты (красиво оформленные данные из таблиц), так и отчеты с многоуровневыми группами, отчеты с отношениями master-detail, перекрестные отчеты, отчеты с вложенными подотчетами, диаграммы и многие другие. Помимо распечатки отчетов на принтере, FastReport Studio поддерживает экспорт отчетов в форматы PDF, RTF, простой текст, HTML, CSV, таблицы Excel, передаваемые с помощью OLE или XML, а также в форматы изображений BMP, JPEG и TIFF. Дизайнер отчетов (FastReport Designer) представляет приложение, предназначенное как для вызова из других программ, так и для самостоятельного построения отчетов.

FastReport Designer может играть роль и визуального редактора отчетов, и

самостоятельного приложения БД. При этом дизайнер предоставляет конечному пользователю не только возможность управления процессом построения отчёта, но и широкие возможности обработки данных. Не зря читатели журнала Delphi Informant наградили в 2004 году продукцию компании Fast Reports призом «продукт года». Главное же заключается в том, что FastReport Studio действительно оправдывает свое название - отчеты создаются быстро.

В марте 2009 года выпущен релиз FastReport.Net для Microsoft Visual Studio, написаний полностью на C#. Отличительной особенностью FastReport.Net от других генераторов является высокая скорость создания отчетов, стабильная работа при обработке гигантского количества данных (до 20 тыс. страниц отчета)

**SQL Server Reporting Services**

SQL Server Reporting Services (сокр. SSRS, Службы отчетности SQL Server) - программная серверная система создания отчетов, разработанная корпорацией Microsoft. Она может быть использована для подготовки множества интерактивных и печатных отчетов. Система администрируется через веб-интерфейс. Reporting services используют интерфейс веб-служб[1] для поддержки разработки обычных отчетных приложений.

SSRS соперничает с Crystal Reports и другими BI-инструментами, и входит в состав Microsoft SQL Server в качестве устанавливаемого дополнения. Reporting Servicesбыли впервые выпущены в 2004 году как дополнение для SQL Server 2000. Вторая версия была выпущена в виде составной части SQL Server 2005 в ноябре 2005 года. Последняя на момент составления пособия версия была выпущена как часть SQL Server 2008 в августе 2008.

В SSRS отчеты описываются при помощи Report Definition Language (RDL) на языке разметки XML. Отчеты могут проектироваться при помощи последних версий Microsoft Visual Studio с входящим в них дополнением Business Intelligence Projects или при помощи входящего в комплект Report Builder — упрощенного инструмента, не предлагающего полного функционала Visual Studio. Отчеты, определенные при помощи RDL, могут создаваться во множестве различных форматов, включая Excel, PDF, CSV, XML, TIFF (и других графических форматах), а также HTML Web Archive. SQL Server 2008 SSRS также может подготавливать отчеты в формате Microsoft Word (DOC).

Пользователи могут работать с веб-службой Report Server напрямую или использовать Report Manager - веб-приложение, взаимодействующее с веб-службой Report Server. При помощи Report Manager могут просматривать и управлять отчетами, также как и управлять и оперировать источниками данных и настройками безопасности. Отчеты могут рассылаться по электронной почте или записываться на файловую систему как обычный файл. Защита выполняется на основе ролей и может накладываться на отдельные элементы, как например, отчет или источник данных, каталог элементов или сайт вообще. Роли безопасности и права являются наследуемыми и могут быть переопределены.

В дополнение к использованию отдельного Report Server, поставляемого с SQL Server, RDL-отчеты можно просматривать при помощи веб-контрола ASP.NET ReportViewer или Windows Forms-контрола ReportViewer. Это позволяет встраивать отчеты прямо в веб-страницы или .NET-приложения.

Контрол ReportViewer обрабатывает отчеты одним из двух способов: (a) на стороне сервера, где отчет обрабатывается Report Server; и (b) локальная обработка, где соответствующий контрол самостоятельно обрабатывает RDL-файл.

Службы MicrosoftSQL Server 2008 Reporting Services обеспечивают широкий спектр готовых к использованию средств и служб для создания, разворачивания и управления отчетами организации, а также функции программирования, которые позволяют расширить и настроить функциональность отчетов.

Инструменты служб Reporting Services работают в окружении Microsoft Visual Studio и полностью интегрированы с инструментами и компонентами SQL Server.

**2. Создание и публикация отчета средствами Microsoft SQL Server 2008 Reporting Services**

Рассмотрим создание простого табличного отчета (на основе базы данных) с помощью конструктора отчетов служб Microsoft SQL Server 2008 Reporting Services.

Чтобы создать отчет в SQL Server, необходимо сначала создать проект сервера отчетов, в котором будет сохранен файл определения отчета и другие файлы ресурсов, необходимые для отчета. Затем будет создан действительный файл определения отчета, определен источник данных для этого отчета, набор данных и макет отчета. При выполнении отчета происходит получение и объединение фактических данных с макетом, затем осуществляется его подготовка к просмотру на экране, после чего может быть выполнен их импорт, печать или сохранение.

Инструкции к созданию и публикации отчета средствами Microsoft SQL Server 2008 Reporting ServicesДля создания отчетов используем пример базы данных с названием University.mdf, которая была создана в лабораторной работе №9.

При выполнении примеров и заданий обращайте внимание на соответствие названий БД, таблиц и других объектов проекта.

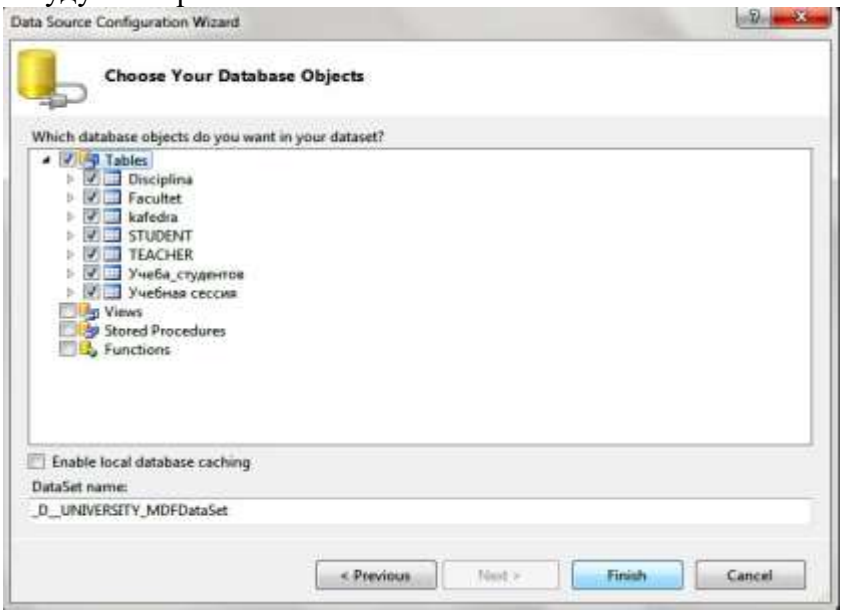
Отчеты во многом похожи на формы и тоже позволяют получить результаты работы запросов в наглядной форме, но только не на экране, а в виде распечатки на принтере. Таким образом, в результате работы отчета создается бумажный документ.

На Visual C# 2010 есть несколько способов создания отчетов. Один из способов создание отчетов это использование встроенного генератора отчета *Reporting Services*.

1. Откройте проект созданный в лабораторной работе №9. В этом проекте мы создали отображение таблиц базы данных с помощью визуальных объектов без программирования.

Особенностью данного проекта было создание источника данных нашей базы данных через команду Add Project Data Source.

Целью данного диалога было создание строки соединения, в которой были описаны параметры соединения для механизма ADO, такие как тип базы данных, ее местонахождение, имена пользователей, средства безопасности и все объекты базы данных, которые будут отображены.




В нашем случае мы добавили все таблицы базы данных University.mdf. У вас должен был появиться новый компонент UniversityDataSet.

Затем мы создали в этом проекте два элемента DataGridView, где отобразили две взаимосвязанных таблицы – Кафедры и Студенты.

Теперь продолжим работу с этим проектом и создадим несколько разных отчетов к базе данных University.mdf.

**2. Рассмотрим создание простого табличного отчета о преподавателях университета** (на основе базы данных **University.mdf**) с помощью конструктора отчетов служб **Microsoft SQL Server 2008 Reporting Services**.

Чтобы создать **отчет в SQL Server**, необходимо сначала **создать файл определения отчета** и другие файлы ресурсов, необходимые для отчета. Затем **определить источник данных для этого отчета, набор данных и макет отчета.** При выполнении отчета происходит получение и объединение фактических данных с макетом, затем осуществляется его подготовка к просмотру на экране, после чего может

быть выполнен их импорт, печать или сохранение.

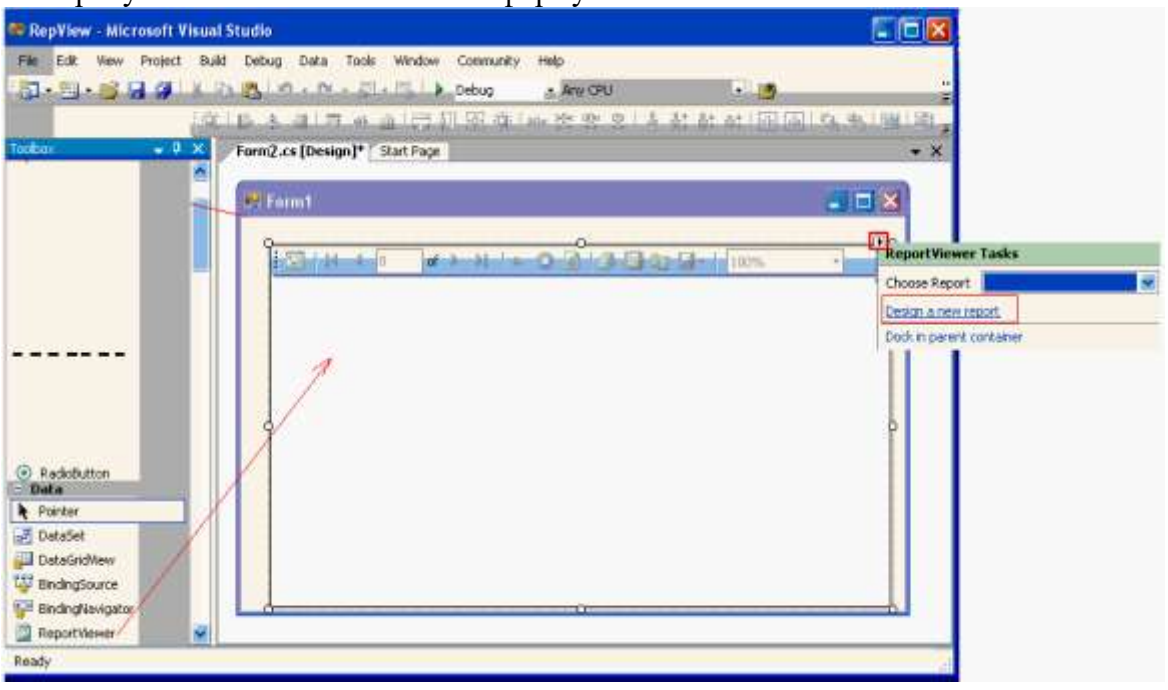
**3. Размещать отчет мы будем на новой диалоговой форме**. Поэтому в нашу первую форму добавим внизу формы кнопку **Button** с панели инструментов **Toolbox**. Переименуем подпись кнопки на «**Отчет по преподавателям**».

Затем добавьте еще одну диалоговую форму **Project/Add Windows Form**. Чтобы при нажатии на кнопку из первой формы открывалась вторая, вернитесь **в конструктор первой формы** и вызовите **обработчик нажатия кнопки** и введите программный код:

```
Form2 f2 = new Form2(); f2.Show();
```

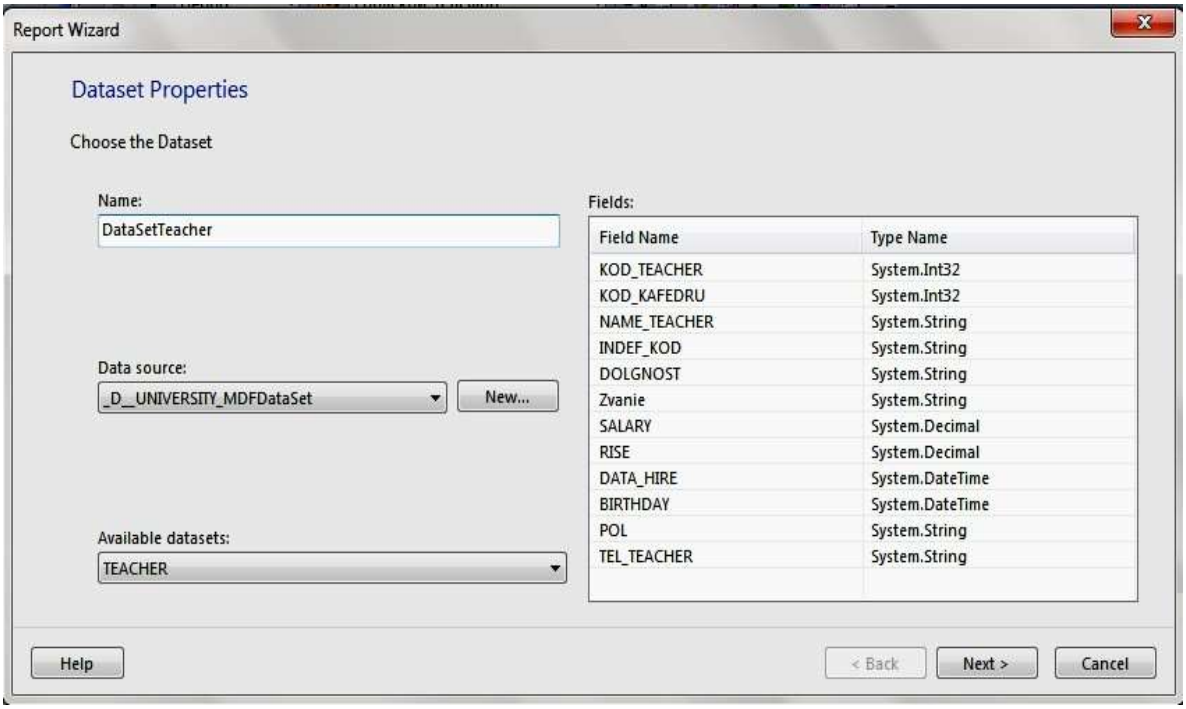
**4. Перейдите на вторую форму**. Переименуйте подпись второй формы **Отчет по преподавателям**.

Поместим на форму объект **ReportViewer** с панели инструментов **Toolbox**, как показано на рисунке. Растяните его на всю форму.

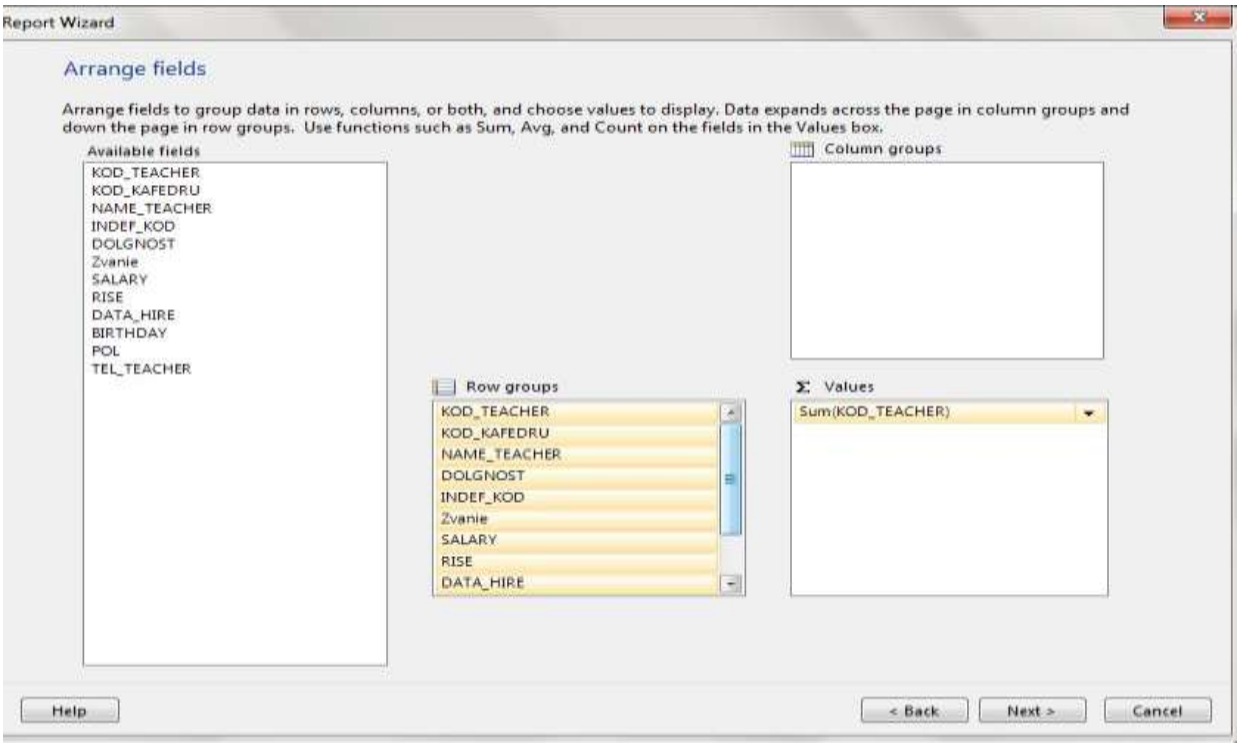



**5. Настроим источник данных для отображения**. Нажимаем в правом верхнем углу контрола **ReportViewer** треугольничек и выбираем пункт **Design a new Report** (Рис.7.). В появившемся окне редактора **ReportViewer**, нажимаем кнопку "**Add New Source..**"(Рис.8.)

В появившемся окне выберем уже созданный нами ранее общий источник данных – **UniversityDataSet**, таблица – **Teacher**, переименуем имя на **DataSetTeacher.**

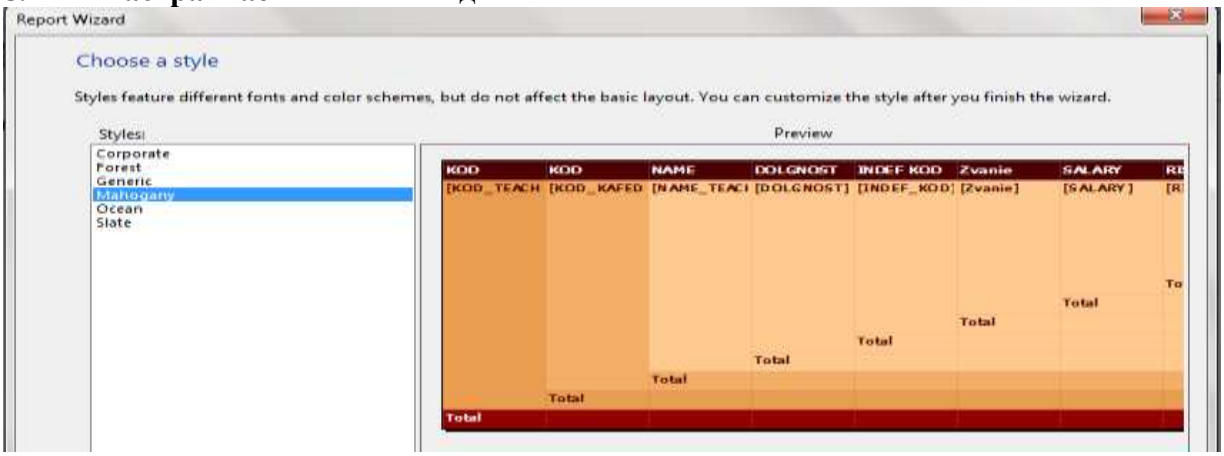



**6. Нажимаем кнопку NEXT** и переходим на второе диалоговое окно. В этом диалоговом окне, с помощью мастера можно настроить, как данные будут размещены в отчете. Например, добавим в отчет следующие группы:

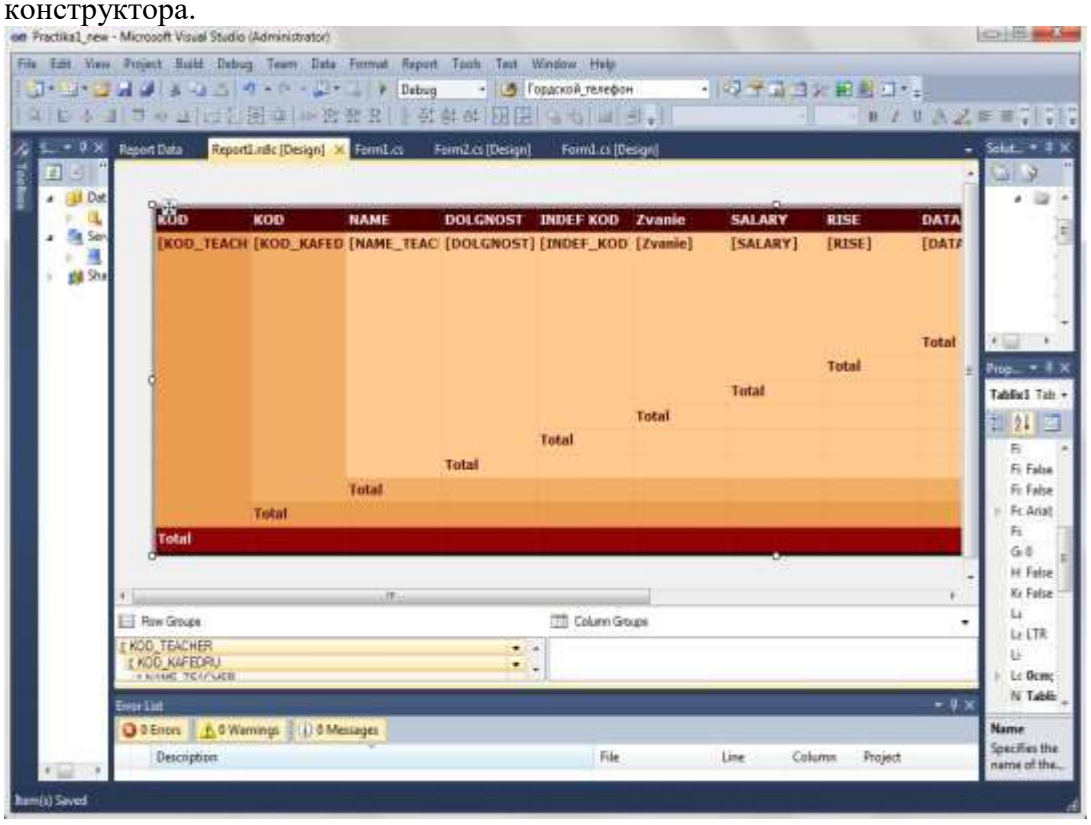



7. **Настраиваем**, как будут отображаться группы в отчете:


**8. Настраиваем** внешний вид отчета.

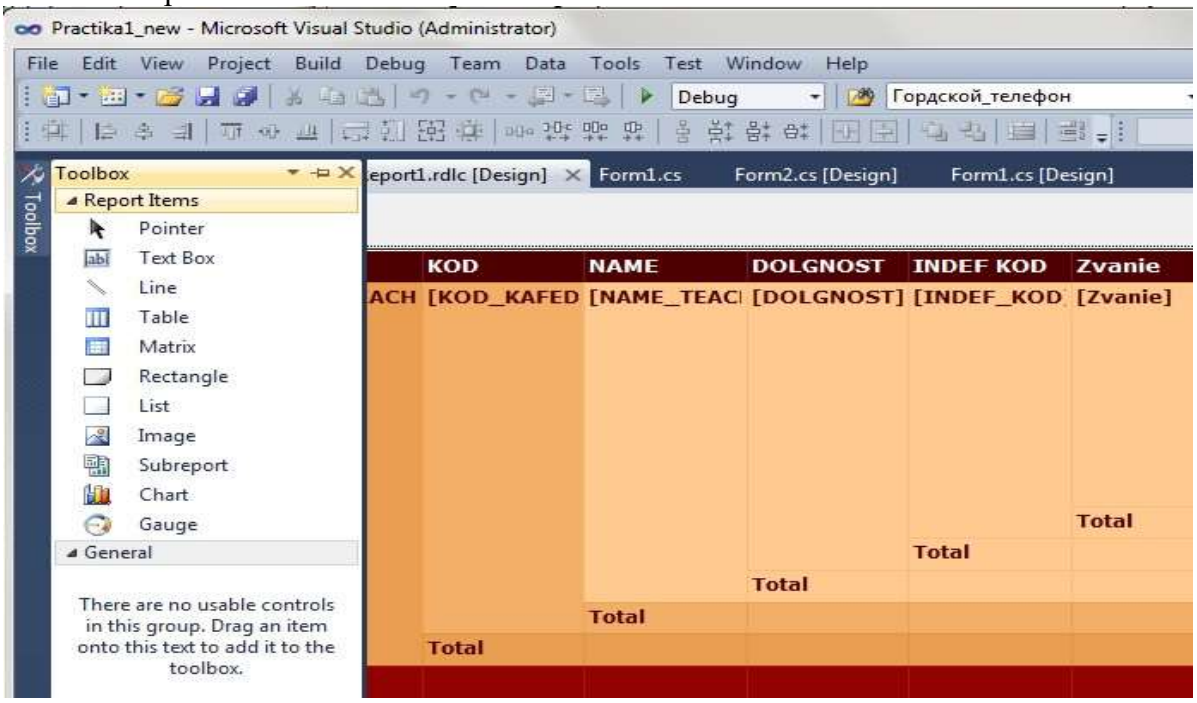



9. **На последнем шаге мастера** нажимаем кнопку **FINISH**. Итак, в результате у вас появиться новый объект проекта – **Report1.rdls**, который будет вам открыт в режиме конструктора. Сохранитесь.



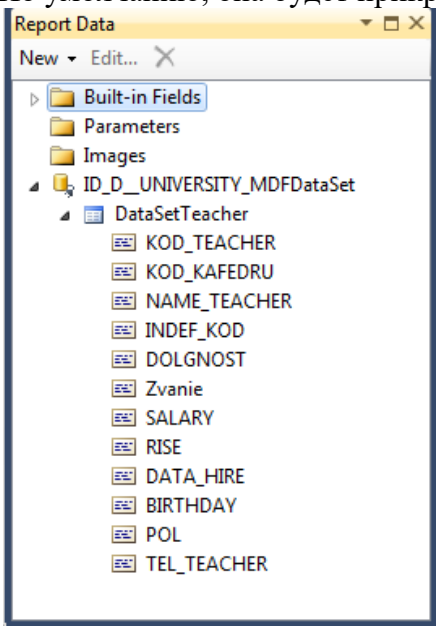

**10. Отображение данных**

Кликнем мышкой в **поле дизайнера Report1.rdlc** и в **меню View** выберем **ToolBox**. В результате в проекте отобразится **ToolBox**, с контролами, которые доступны для использования в **ReportViewer** (Рис.14.). Здесь мы видим все доступные для отображения в **ReportViever** контролы.




В наш отчет уже добавлена таблица для отображения. Поэтому в этот отчет мы не будем добавлять элемент таблица.

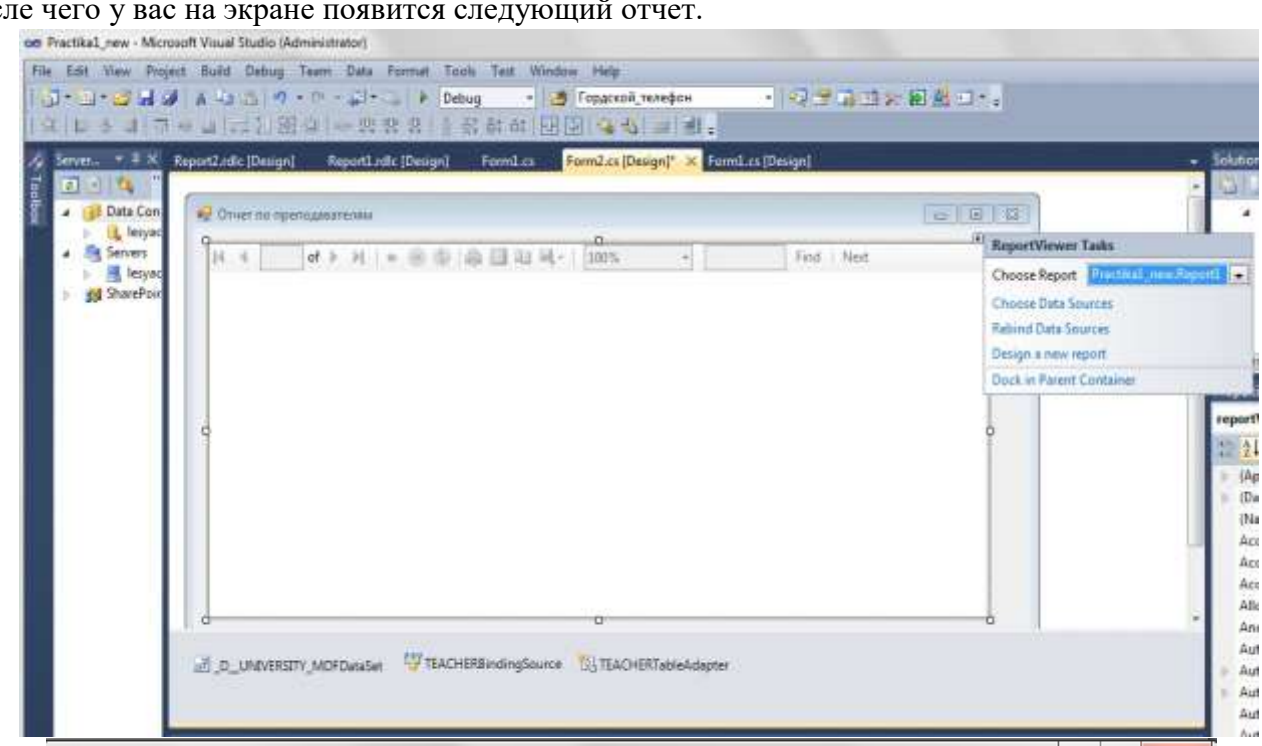
Откройте панель **ReportDate**, чтобы увидеть какие данные были добавлены в отчет. По умолчанию, она будет прикреплена к проекту.

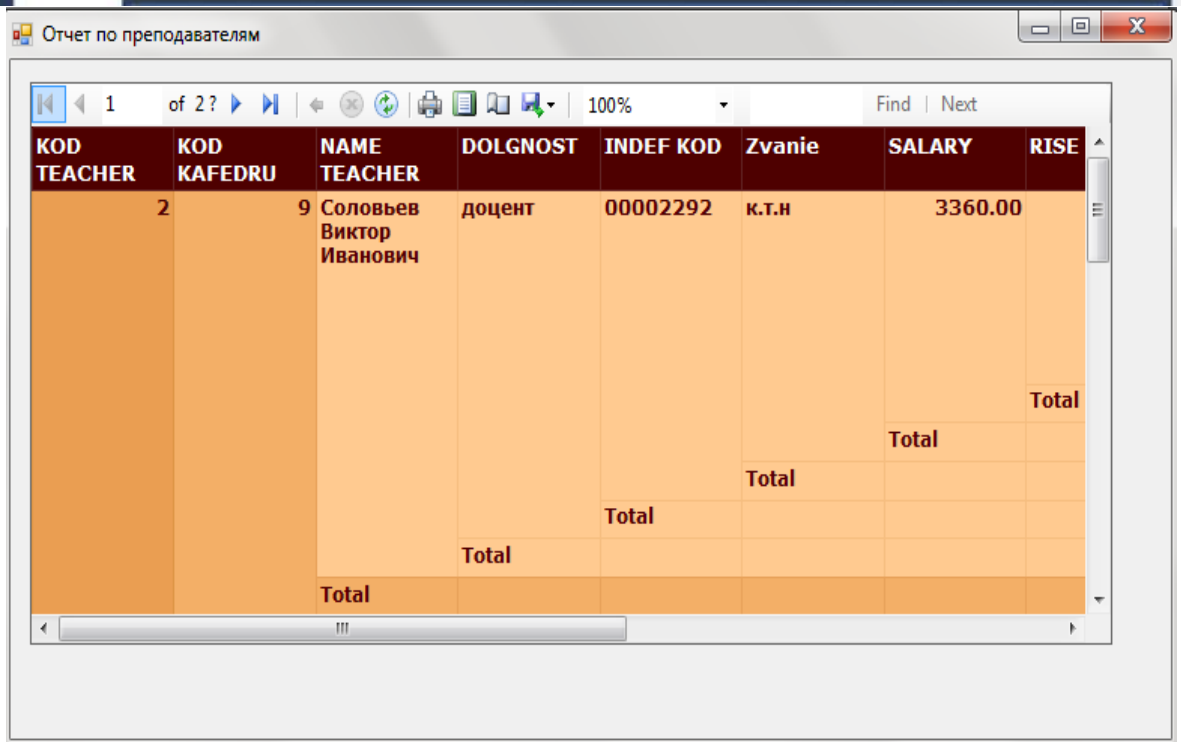



**11. Сохранитесь**. Чтобы отобразить отчет на экран необходимо настроить параметры нашей второй формы. Поэтому вернитесь во вторую форму. Выделите объект **ReportViewer** , нажимаем на треугольничек и выбираем пункт **Report1**.

На форму будут добавлены элементы **UniversityDataSet, TeacherBindingSourse, TeacgerTableAdapter.**

**12. Запускаем проект на выполнение** и нажимаем на первой форме на кнопку. После чего у вас на экране появится следующий отчет.



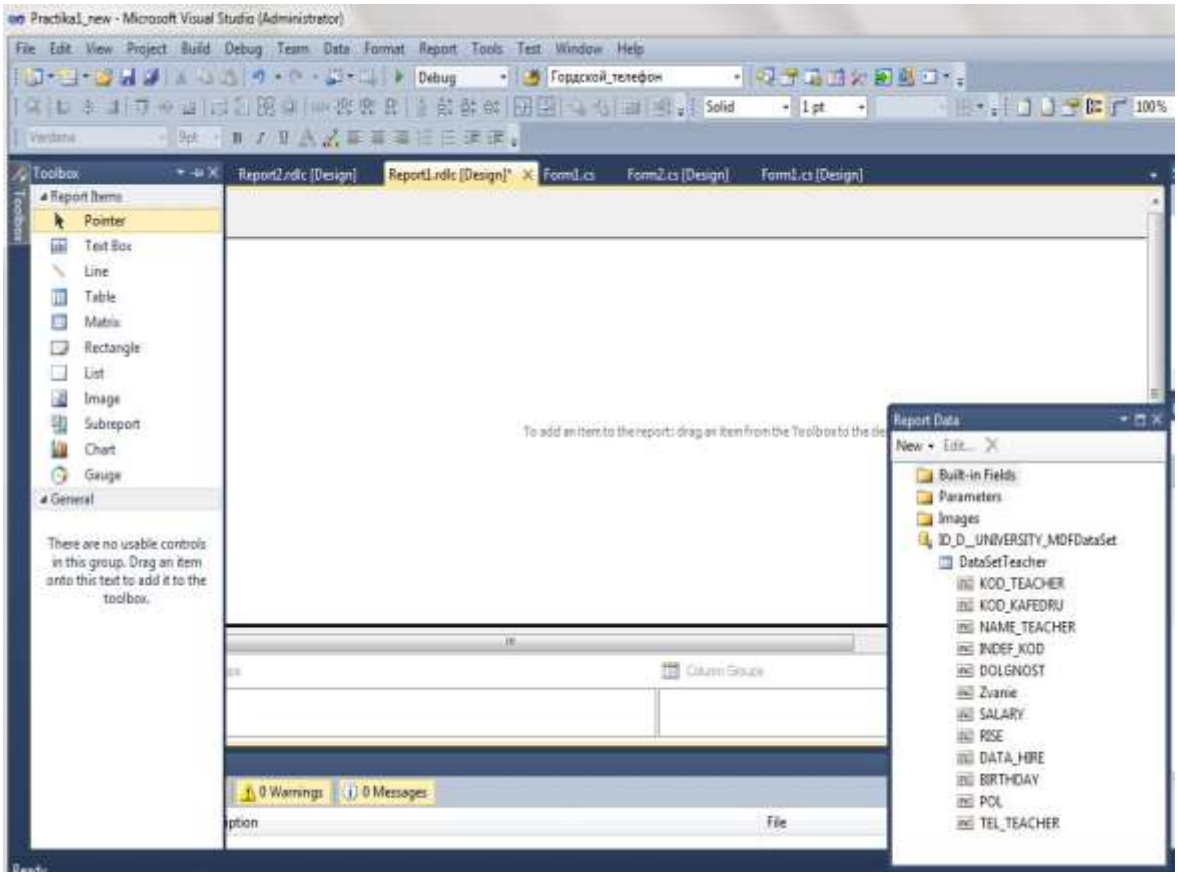



| KOD TEACHER | KOD KAFEDRU | NAME TEACHER | DOLGNOST | INDEF KOD | Zvanie | SALARY | RISE |
|---|---|---|---|---|---|---|---|
| 2 | 9 | Соловьев Виктор Иванович | доцент | 00002292 | к.т.н | 3360.00 | |
| | | | | | | | Total |
| | | | | | | Total | |
| | | | | | Total | | |
| | | | | Total | | | |
| | | | Total | | | | |
| | | Total | | | | | |

**13. Редактирование отчета**. Закройте режим просмотра и перейдите в режим конструктора **Report1.rdlc.**

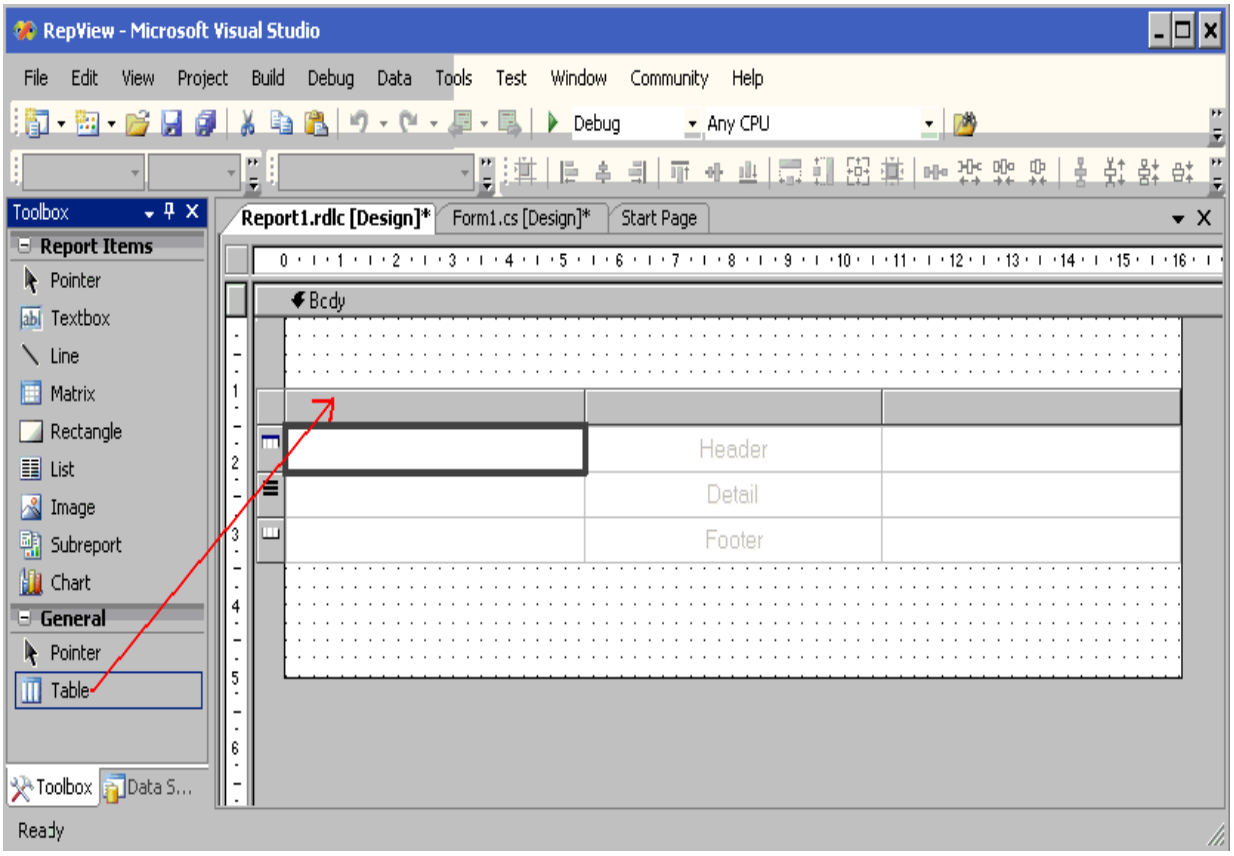
В нашем отчете мы много добавили групп, просматривать такой отчет тяжело.

Выделите таблицу в конструкторе отчетов и удалите ее. На экране появится пустая рабочая область.

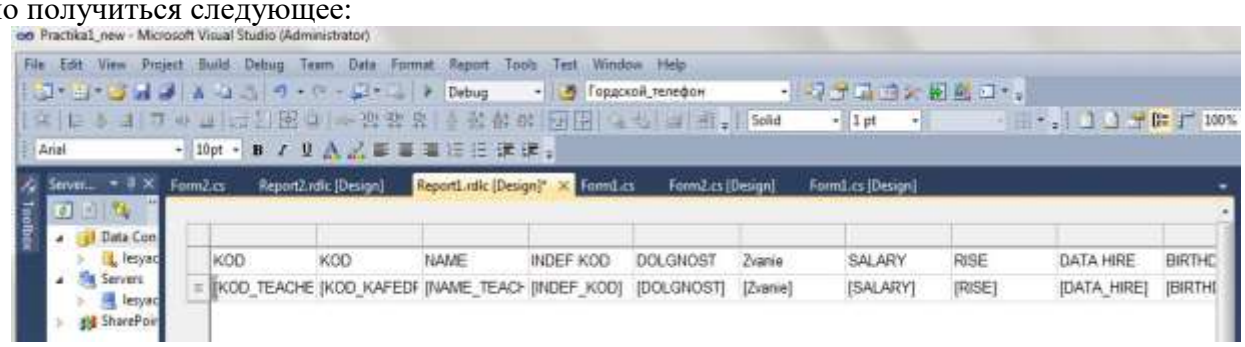



Вначале будем использовать элемент таблица **Table** из панели инструментов **ToolBox**. Перетащим таблицу на панель дизайнера:

Добавление таблицы в дизайнер ReportViever

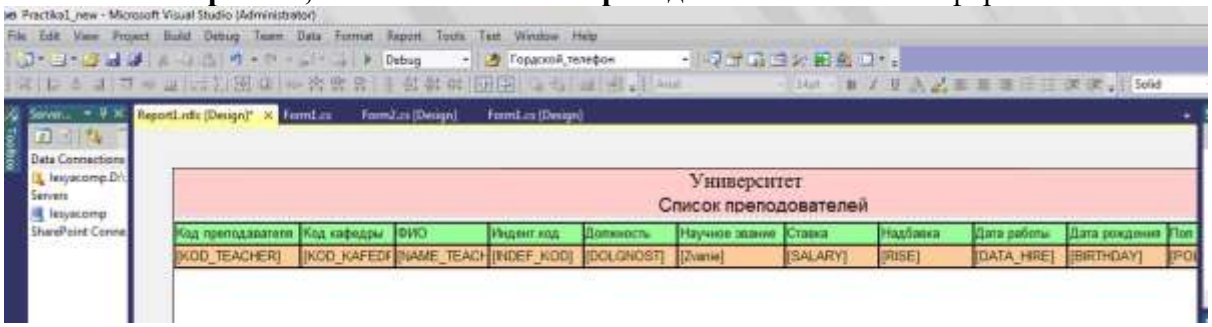


Теперь последовательно добавляем столбцы таблицы из панели **ReportDate**. У вас должно получиться следующее:

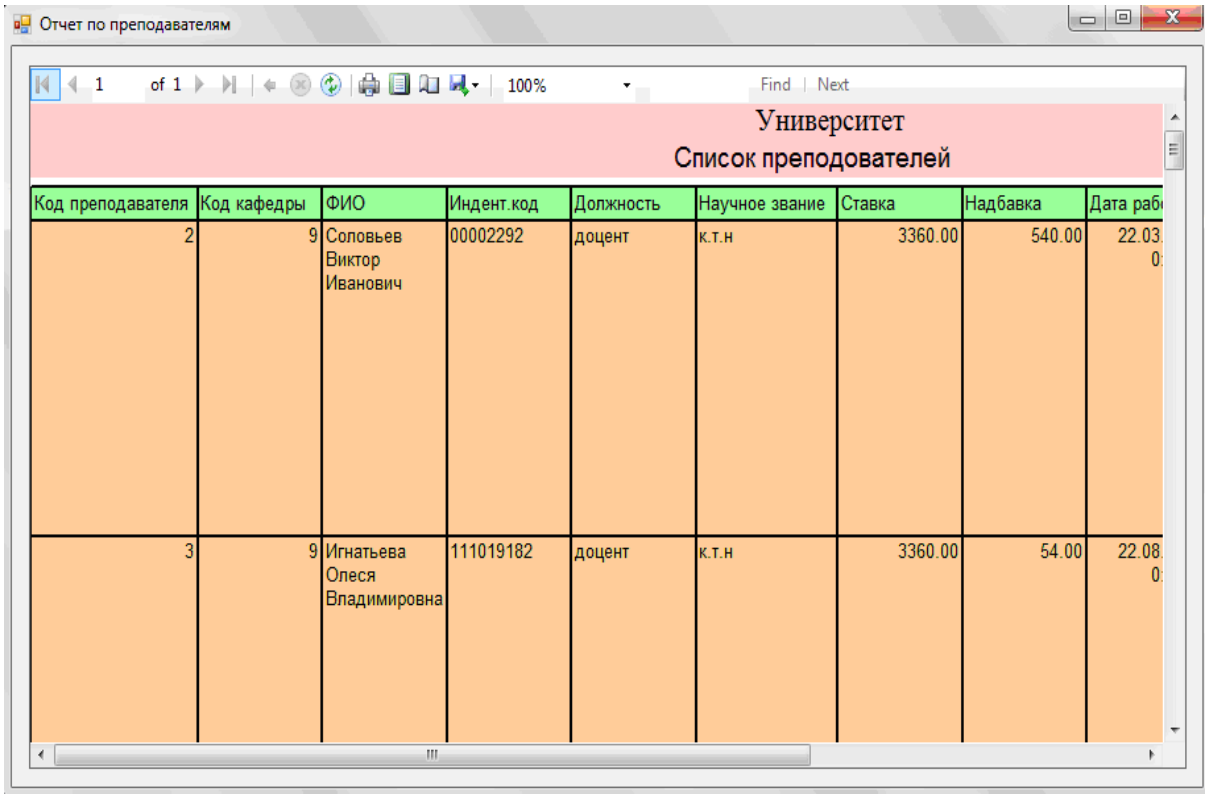



Затем, изменим подписи к столбцам, перепишем их по-русски, сделаем заливку строк, добавим границу всех ячеек таблицы.

Сместите таблицу немного вниз и добавим заголовок к таблице. Для этого добавьте с панели инструментов **ToolBox** элемент **TextBox** и введите туда текс «**Университет, список преподавателей**». Оформите заголовок.



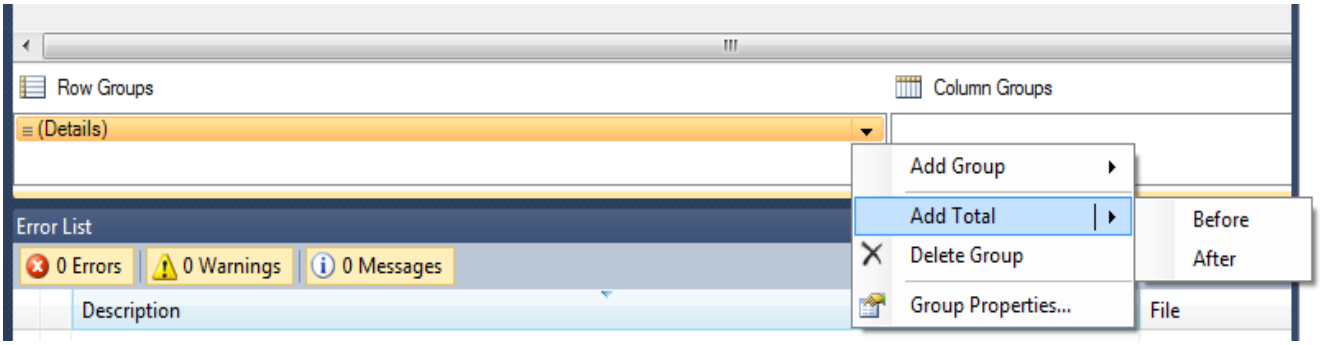

Сохранимся. И запустим на выполнение проект. Просмотрите новый вид отчета.



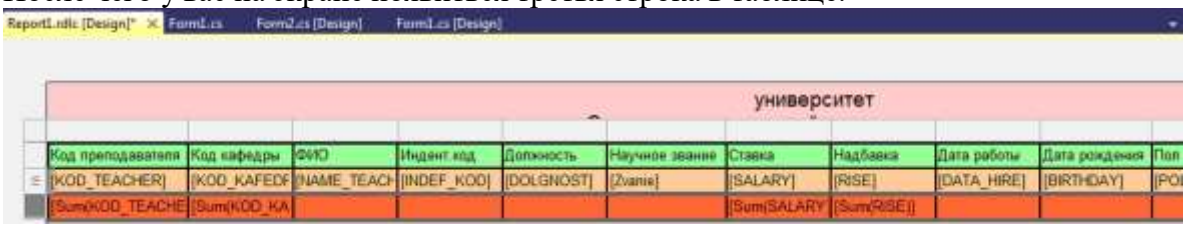

## 14. Добавление итогов в отчет.

Для добавления итогов в отчет перейдите в режим конструктора нашего отчета,

выделите таблицу и внизу выберите из группы **Row Groups** команду из списка =**Details/ Add Total/ Before**.




После чего у вас на экране появиться третья строка в таблице.



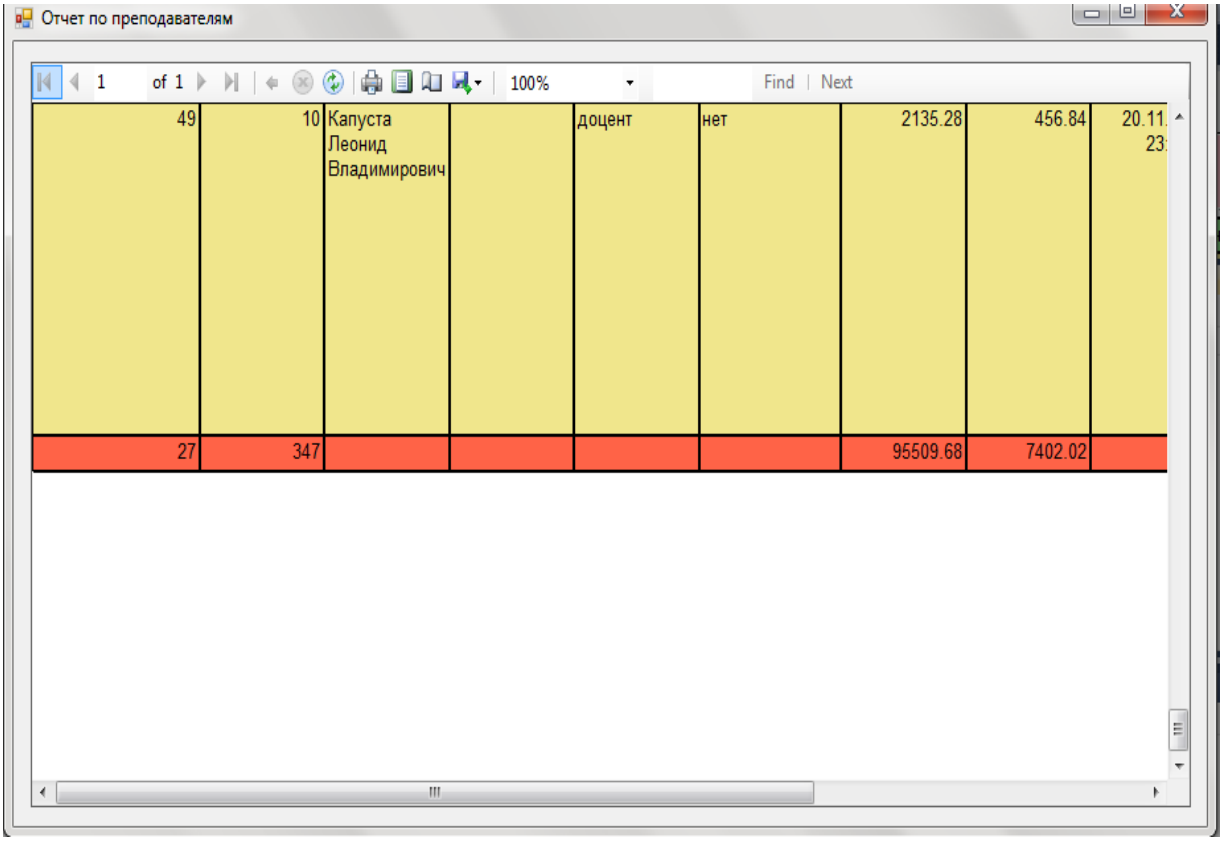

При этом, по умолчанию, ячейка таблицы будет восприниматься как сумма содержимого ячеек данного столбца:

=Sum(Kod_teacher)

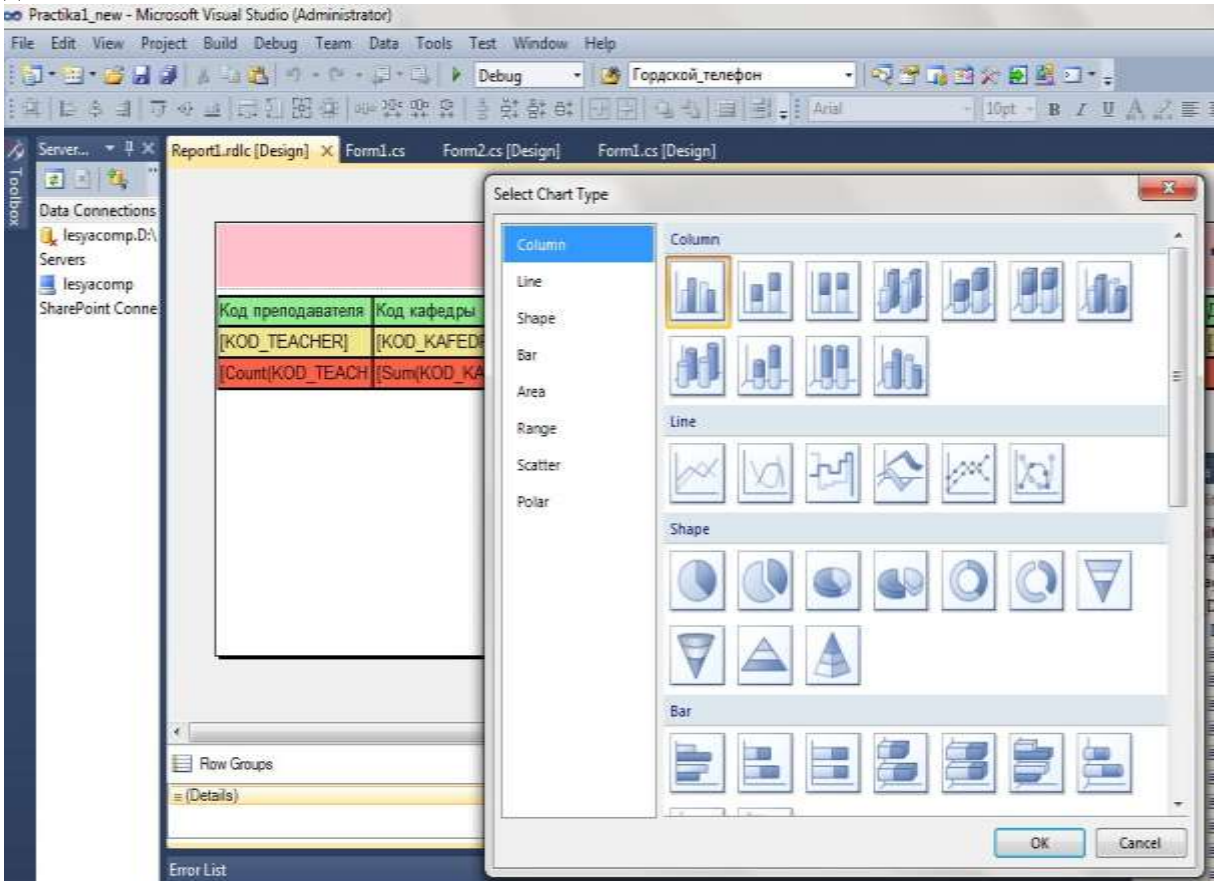
Достаточно кликнуть правой кнопкой мышки по данной ячейке и выбрать пункт **"fx Expression..."** мы отобразим окно **"Edit Expression",** где можно подобрать любую другую функцию из множества доступных.

Например, измените агрегатную функцию для первого столбца на **Count(Kod_teacher)**.

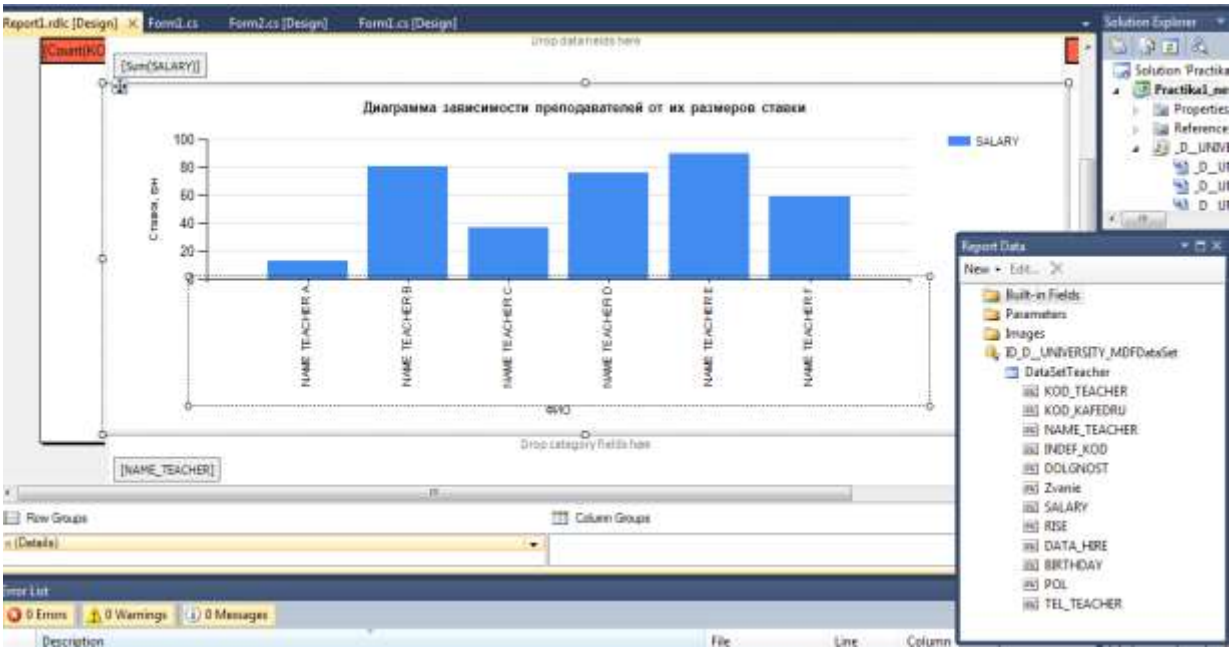
Сохранитесь и просмотрите результат. Пролистайте отчет до самого конца и в последней строке будут выведены итоги.




15. **Теперь добавим контрол Chart** с вкладки **Tools ReportViewer**. И, как показано на рис.17. перетащим поля в своеобразные ушки контрола, которые появляются при двойном клике мышкой в поле контрола.



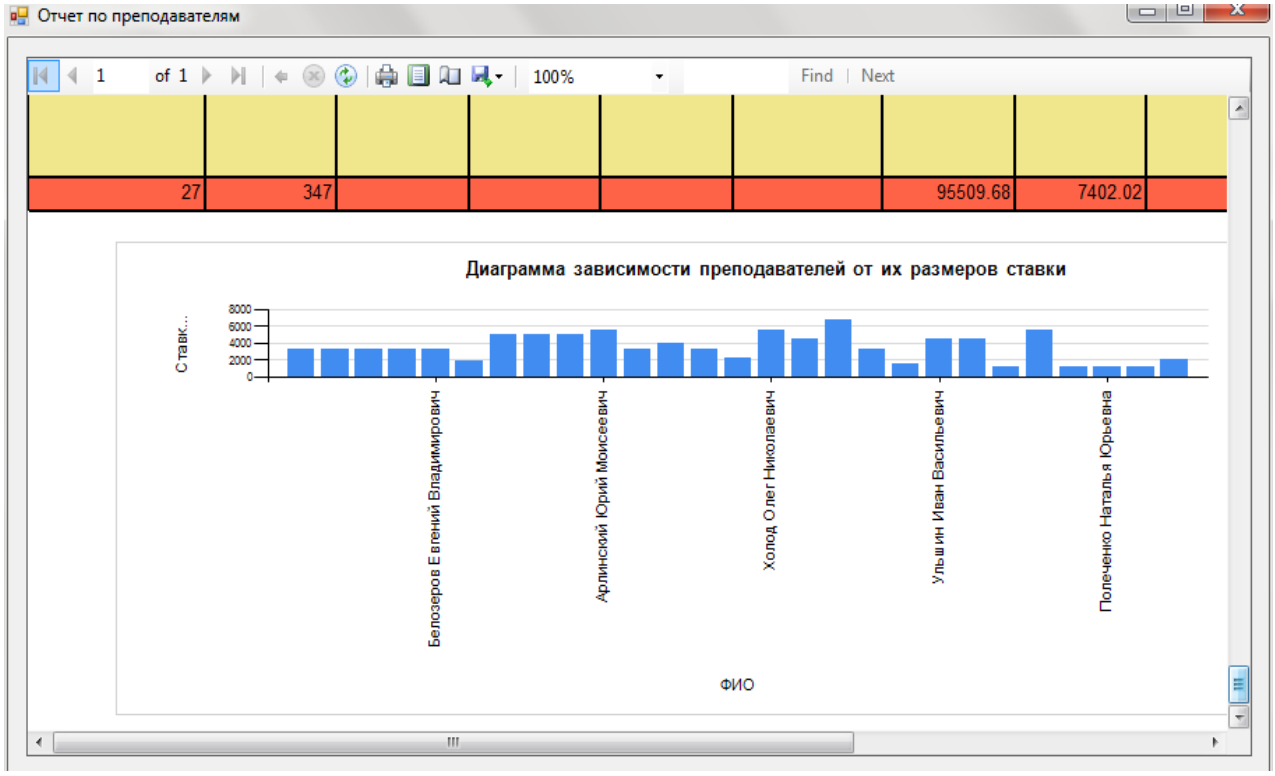

Выберите первую гистограмму и нажимаем на кнопку Ок. Диаграмма добавиться под таблицей. Но на ней пока не добавлены данные, по каким будет строиться диаграмма.




В качестве данных по оси **OX** перетащите из панели **ReportDate** поле **NameTeacher** и разместите в нижнюю область под осью X.В качестве данных оси **OУ** перетащите из панели **ReportDate** поле **Salary** и разместите в область оси У.

Измените подписи заголовков. Измените угол наклона на фамилии. Сохраните и запустите на выполнение. Результат показан на рис.18:

## 16. Определение запроса Transact-SQL для данных отчета




Бывает так, что отчет нужно построить не на основе таблиц базы данных, а на определенном **запросе,** где в качестве источника будут использоваться несколько взаимосвязанных таблиц с каким-то условием отбора.

Например, создадим еще один отчет для отображения **всех факультетов и**

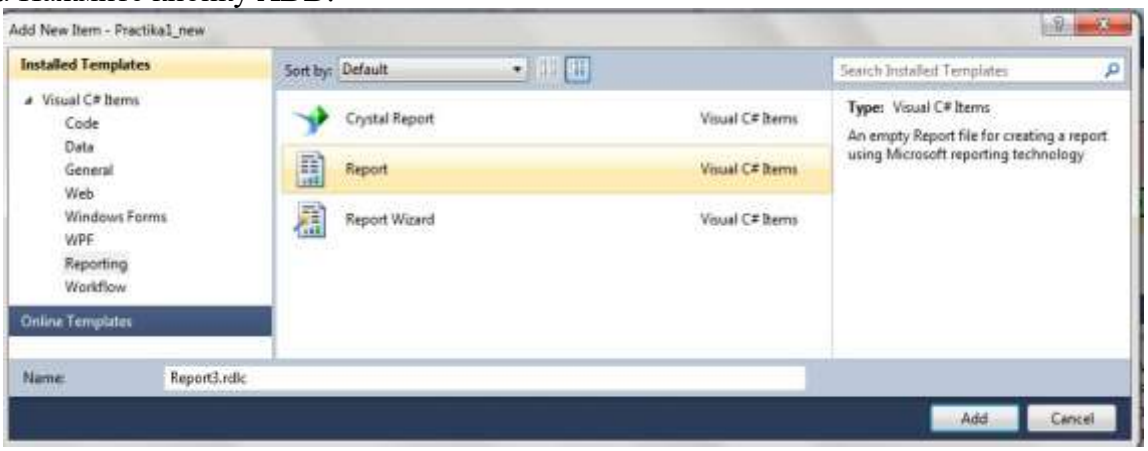
**имеющихся на них кафедрах.**

1). Предварительно **создадим новую диалоговую форму** и изменим подпись ее на «**Факультеты и кафедры**».

**2). Добавьте еще одну кнопку** на первую форму для открытия третьей формы.

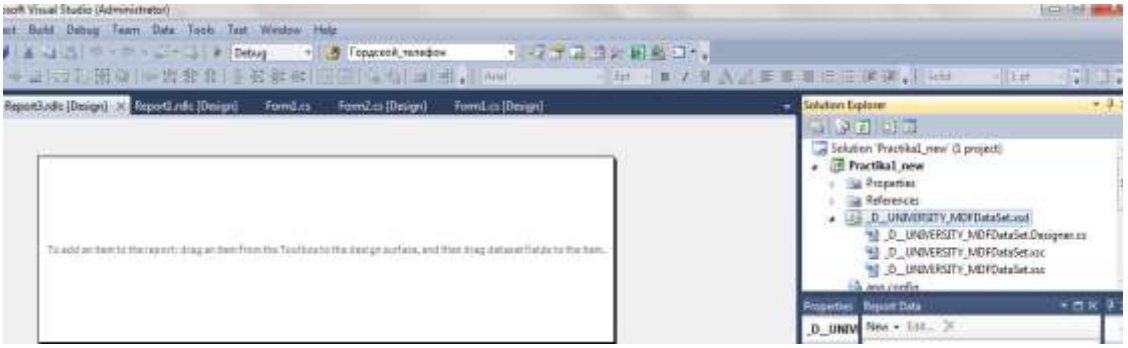
3). Добавляем на новую форму элемент ReportViewer и пока не будем ничего настраивать.

4). Создадим пустой бланк отчета, для этого выполним команду меню **Project/Add windows Form** и в окне проекта выберем слева категорию **Reporting** и выберите элемент **Report**. Нажмите кнопку **ADD**.

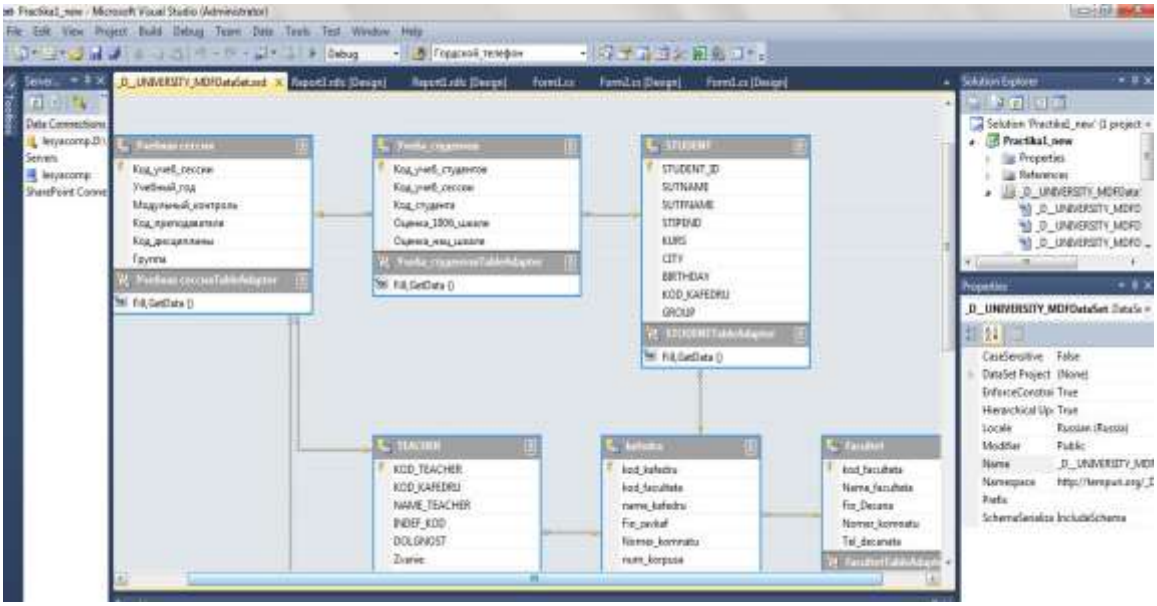



5). У вас создаться пустой бланк нового отчета. Нам необходимо теперь создать источник данных для этого отчета.

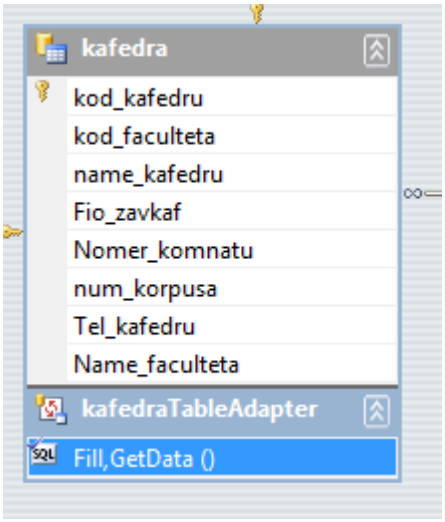
6). Чтобы создать запрос для отчета перейдем в область **Solution Explorer**. В проводнике выберите элемент **University_DataSet.** Нажмите правой кнопкой мыши на нем и выберите из меню команду **Open**.



Вам будет отображена **схема базы данных**, см. рис.21.



Выбираем на схеме таблицу **Kafedra**, внизу ее вы увидите команду **SQL, Fill, GetDate()**.




Нажимаем на этом запросе правой кнопкой мыши и выбираем из контекстного меню команду Configure.

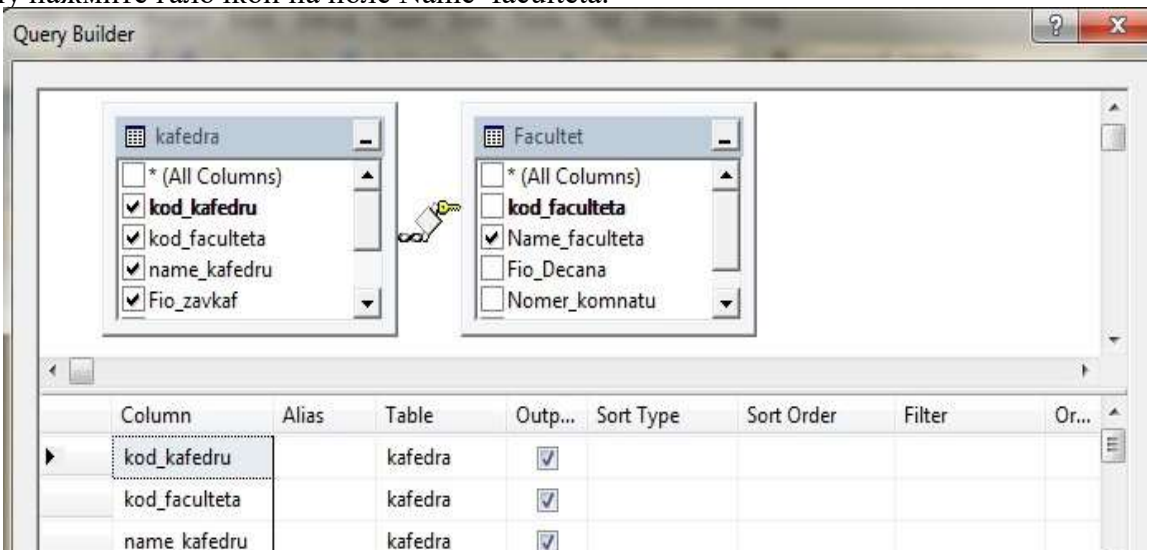
7). У вас откроется конструктор запросов. Нажимаем на кнопку Query Builder.

8). Вам будет открыт конструктор запросов, где зная язык SQL вы сможете легко создать наш запрос, с помощью которого необходимо вывести все факультеты и их кафедры.

В окне был уже сформирован запрос на вывод всех кафедр.

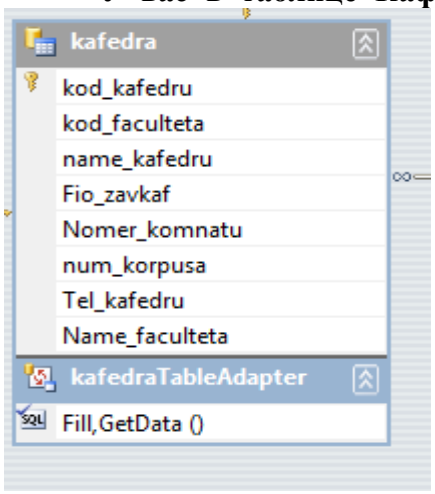
Изменим запрос, добавьте в конструктор новую таблицу Faculty. Для этого вы нажимаете на пустом месте правой клавишей мыши и выбираете ADD table. И добавляем таблицу с факультетами.

Из таблицы факультетов нас интересует только поле с названиями факультетов, поэтому нажмите галочкой на поле Name  faculteta.

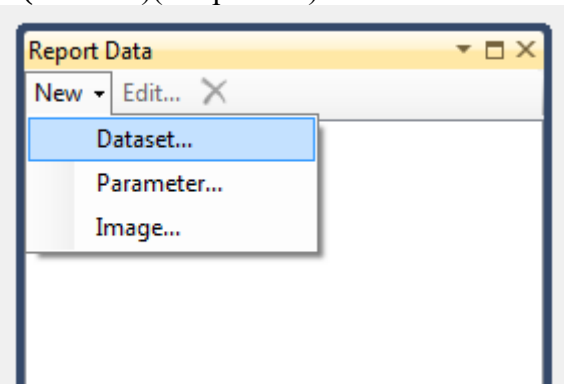


**Можно просмотреть результат выполнения запроса, для этого нажмите кнопку Execute Query.**

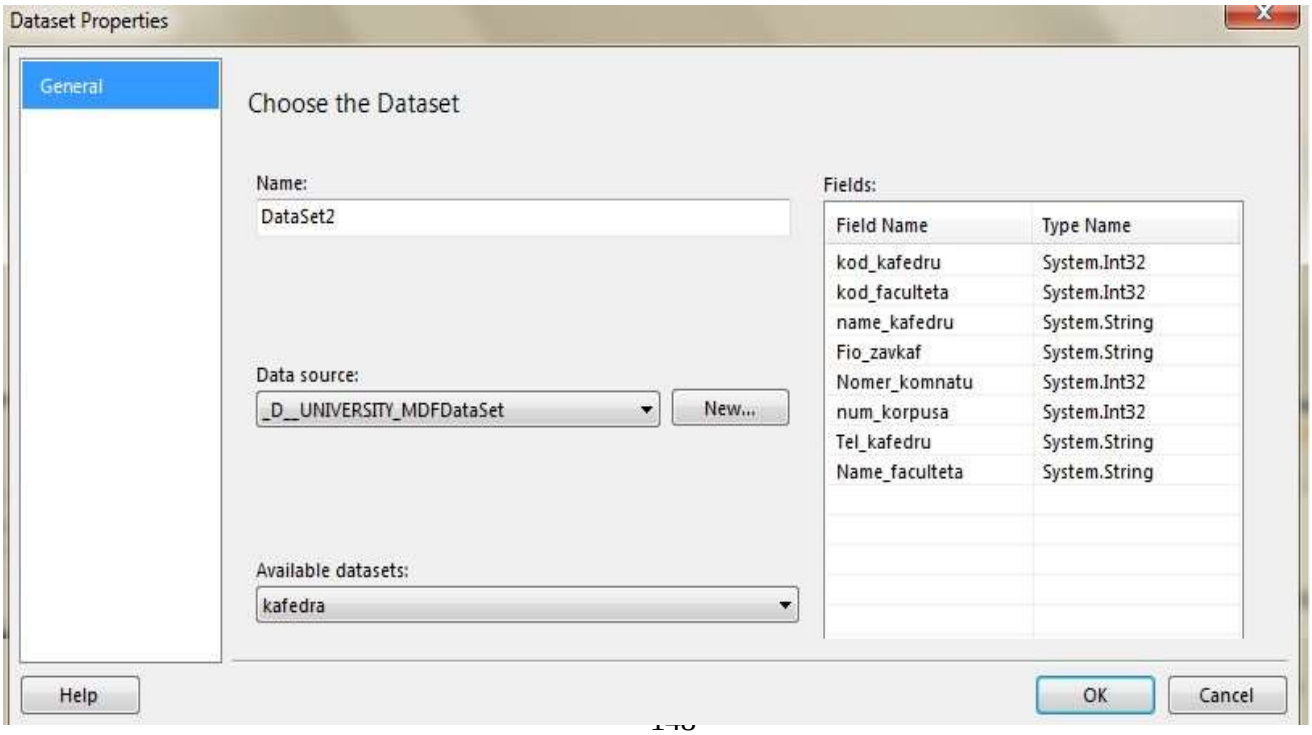
**Затем нажимаем кнопку Ок. Затем нажимаем кнопку Finish. Сохранитесь.**
**У вас в таблице Кафедры теперь появилось поле с названием факультетов.**




9). Возвращаемся в режим конструктора нашего второго отчета **Report2.rdls** . В области **Данные отчета** (**Report Date** ) нажмите кнопку **Создать (Add)** и выберите **Набор данных** (**DataSet**)(см. рис. 25).




Откроется диалоговое окно **Свойства набора данных**.
**Установите источник – University_DataSet** и таблицу для вывода **Kafedra.**



Нажмите кнопку ОК для выхода из диалогового окна **Свойства набора данных**. Поля набора данных **DataSet** появятся в области **Данные отчета**.Определен запрос, получающий данные для **отчета**. Далее предстоит создать **формат отчета**.

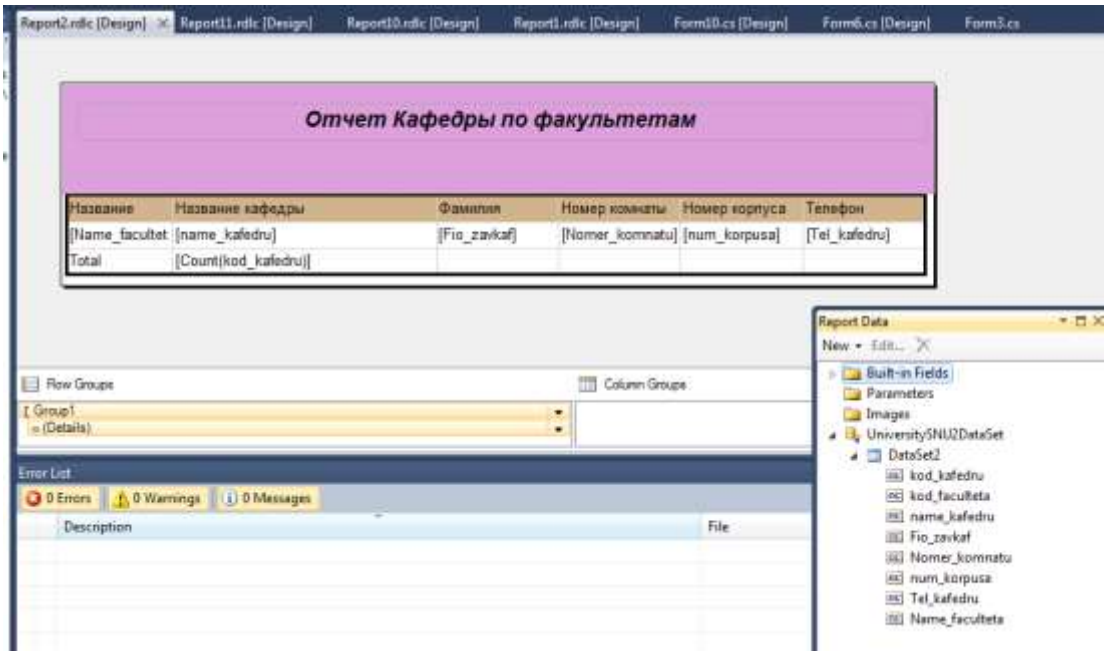

**12). Добавление табличной области данных и полей в макет отчета**

После определения набора данных можно приступать к определению макета отчета. Макет отчета создается путем перетаскивания в область конструктора областей данных, текстовых полей, изображений и других элементов, которые необходимо включить в отчет.

Элементы, содержащие повторяющиеся строки данных из базовых наборов данных, называются **областями данных.** Обычно отчеты имеют только одну область данных, но можно добавить больше, например, если требуется добавить диаграмму в табличный отчет. После добавления области данных можно добавлять в нее поля.

На вкладке Панель элементов щелкните элемент Таблица, а затем щелкните область конструктора. В конструкторе отчетов будет отображена табличная область данных с тремя столбцами.

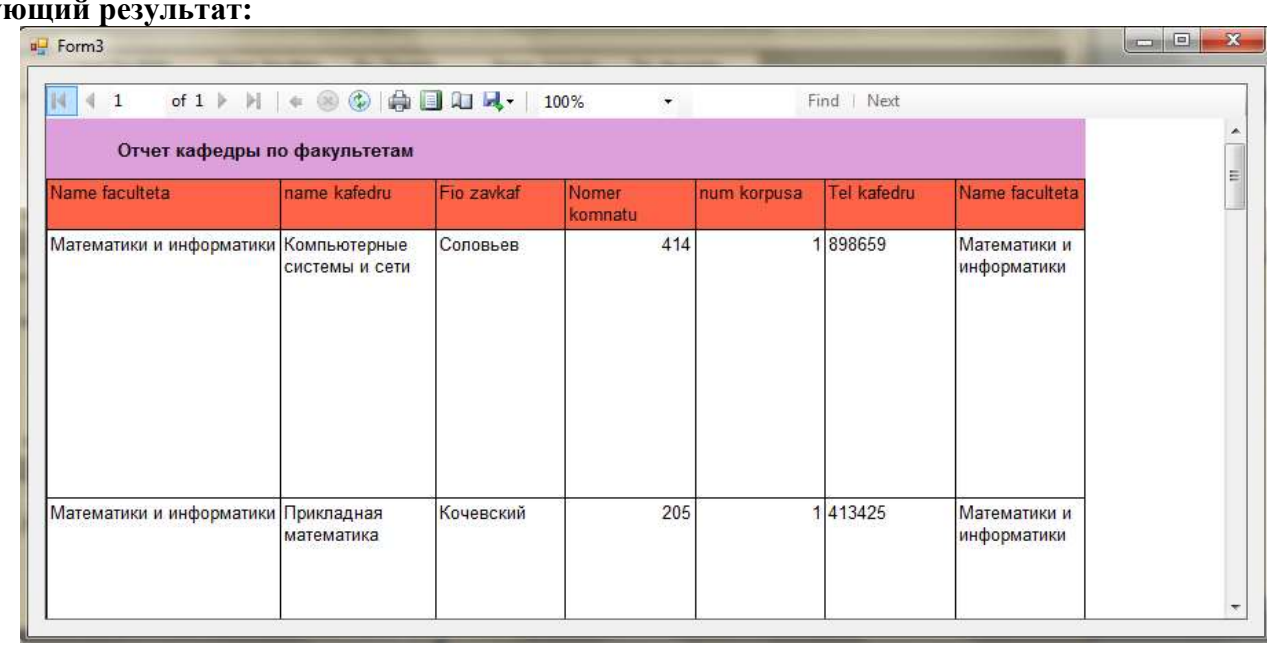
Измените формат отчета на такой, какой показан на рис.26. Добавьте подведение итогов по строке.



**Далее, вернитесь в третью форму, выделите объект ReportView и из треугольничка выберите Report2.**

Сохранитесь и запустите проект на выполнение.

**В главной форме нажмите вторую кнопку и на экране должен появиться следующий результат:**




**Задание к лабораторной работе**

Для созданной базы данных, согласно номеру варианта, продолжить разрабатывать **клиентское приложение в среде Visual Studio C#**, сделанного в лаб.работе №9.

Добавить две новых диалоговых формы для отображения отчетов.

Один отчет создать как **простой табличный отчет** на основе одной из таблиц базы данных с помощью конструктора отчетов служб **Microsoft SQL Server 2008 Reporting Services**. Изменить макет отчета, настроить подведение итогов и внешнее оформление (цвет, фон, шрифты и т.д.).

Второй отчет создать на основе **запроса Transact-SQL** с помощью встроенного конструктора запросов **Query Builder.** Данный запрос должен извлекать данные из двух взаимосвязанных таблиц базы данных.

Клиентское приложение сохранить в папке ***ФИО_студента/Лаб9***.

Создать текстовый отчет, в котором отобразить скриншоты результатов работы в среде **Visual Studio C#.**

**Лабораторная работа № 11 Создание и управление базой данных с помощью SQL – операторов (6 часов)**

**Цель занятия:** Программное подключение к базе данных с использованием событий Connection ADO.NET.

**Использование событий объекта Connection.**

**Самостоятельно:** используя приведенный ниже пример создайте новое приложение и выполните загрузку данных таблицы **Производители** при нажатии кнопки с использованием событий объекта **Connection**.

Создадим новое Windows-приложение, работающее с базой данных BDTur_firmSQL.mdf.

На форме разместим объект dataGridView со свойством Dock=Top, кнопку btnFill с надписью «Заполнить» справа под ним, и две метки (объекты типа Label). Свойству AutoSize каждой метки назначим значение False, свойству Dock второй метки (label2) значение Top, а свойству Dock первой (label1) None, и поместим ее под второй меткой слева, увеличив ее размеры (рис. 61).

Подключаем пространство имен для работы с базой в файле Form1.cs:

```
using System.Data.SqlClient;
```

В классе формы создаем строки connectionString и commandText:

```
string connectionString = @"Data Source=.\SQLEXPRESS; AttachDbFilename=" +
        @"D:\ВМИ\For ADO\BDTur_firmSQL.mdf" +
        ";Integrated Security=True;Connect Timeout=30";

string commandText = "SELECT * FROM Туристы";
```

Объекты ADO будем создавать в обработчике события Click кнопки «Заполнить»:

```
private void btnFill_Click(object sender, System.EventArgs e)
{
        SqlConnection  conn = new SqlConnection();
        conn.ConnectionString = connectionString;
//Делегат EventHandler связывает метод-обработчик conn_Disposed
//с событием Disposed объекта conn
        conn.Disposed+=new EventHandler(conn_Disposed);
//Делегат StateChangeEventHandler связывает метод-обработчик conn_StateChange
//с событием StateChange объекта conn
        conn.StateChange+= new StateChangeEventHandler(conn_StateChange);
        SqlDataAdapter dataAdapter = new SqlDataAdapter(commandText, conn);
        DataSet ds = new DataSet();
        dataAdapter.Fill(ds);
        dataGrid1.DataSource = ds.Tables[0].DefaultView;
//Метод Dispose, включающий в себя метод Close,
//разрывает соединение и освобождает ресурсы.
        conn.Dispose();
}
```

Для создания методов-обработчиков дважды нажимаем клавишу TAB при вводе соответствующей строки как на рис. 60.

В методе conn_Disposed просто будем выводить текстовое сообщение в надпись label2:

```
private void conn_Disposed(object sender, EventArgs e)
{
  label2.Text+="Событие Dispose";
}
```

При необходимости в этом методе могут быть определены соответствующие события.

В методе conn_StateChange будем получать информацию о текущем и исходном состояниях соединения:

```
private void conn_StateChange(object sender, StateChangeEventArgs e)
{
  label1.Text+="\nИсходное состояние: "+e.OriginalState.ToString() +
               "\nТекущее состояние: "+ e.CurrentState.ToString();
}
```

Запускаем приложение. До открытия соединения состояние объекта conn было закрытым (Closed). В момент открытия текущим состоянием становится Open, а предыдущим – Closed. Этому соответствуют первые две строки, выведенные в надпись (рис. 61). После закрытия соединения (вызова метода Dispose) текущим состоянием становится закрытое (Closed), а предыдущим – открытое (Open). Этому соответствуют последние две строки, выводимые в надпись.

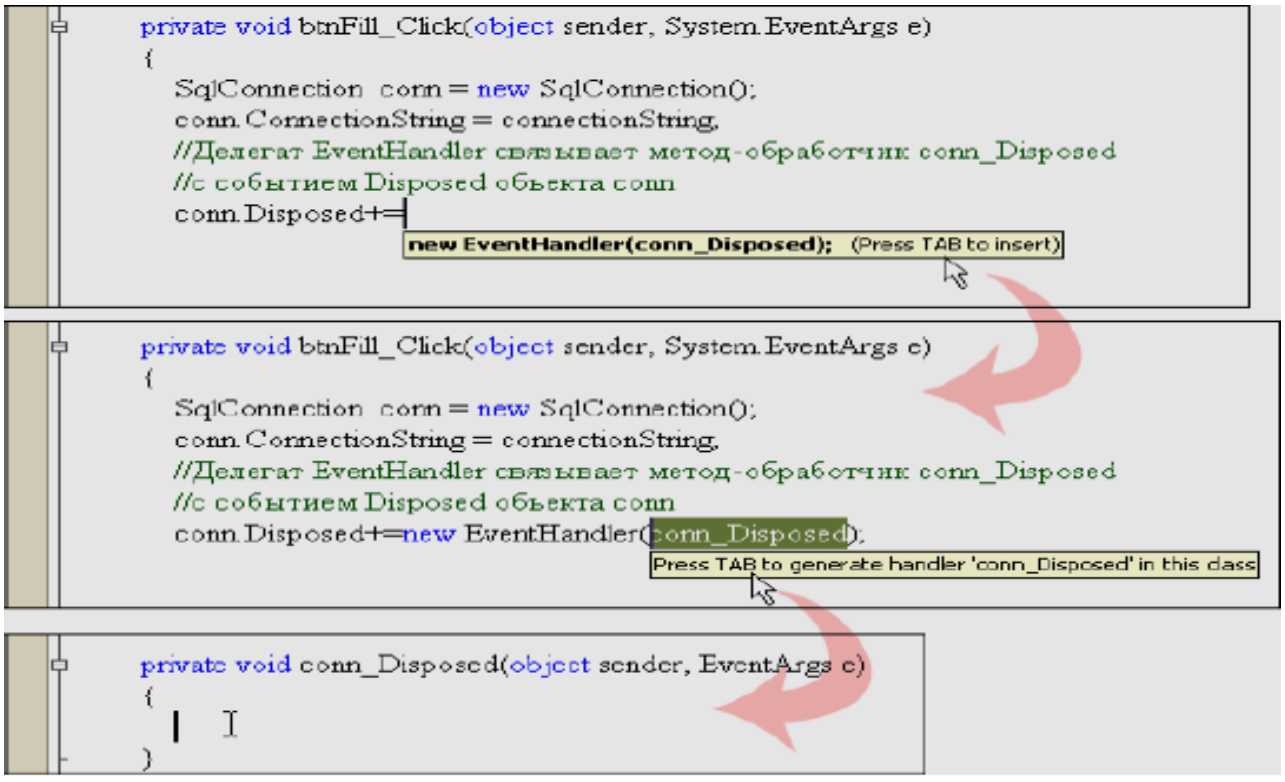



Рис. 60. Автоматическое создание методов-обработчиков при помощи Intelligence


Рис. 61. Изменения состояния соединения с базой данных

Конечно, в таком предельно простом приложении статус соединения очевиден. Но сама идея может применяться в любых, сколь угодно сложных приложениях, когда необходимо определять статус одного из нескольких подключений.

**Задание:**

Изменить приложение так, чтобы в начале появлялась пустая форма с одним текстовым полем и кнопкой **Загрузить**.

В текстовое поле пользователь будет вводить полный путь с именем подключаемой базы данных.

При нажатии кнопки **Загрузить** должно осуществляться соединение с указанной базой данных. В случае не успешного соединения выдать сообщение об отсутствии соединения с базой данных.

После успешного соединения на экране должны появиться ранее спрятанные наборы вкладок формы. В каждой отобразить объекты **DataGridView**, в которых отобразить таблицы **Договоры, Поставщики, Отделы, Производитель, Товары**.

В каждой вкладке разместить по одной кнопке, с помощью которой выполнить запрос произвольного типа (**SELECT, INSERT, DELETE, UPDATE**) выполняющий действие с данными соответствующей таблицы.

Также в каждой вкладке добавить по текстовому полю, в которые отобразить результат расчетов произвольной агрегированной функции (**COUNT, MAX, MIN, AVG**).

**Лабораторная работа № 12 Построение запросов вычисления и подведения итогов к учебной базе данных( 4 часа)**

**Цель занятия:** Изучить синтаксис языка модификации данных. Научится использовать встроенные функции в запросах.

**Числовые функции над числами**

Эти функции возвращают числовые значения на основании заданных в аргументе значений того же типа. Числовые функции используются для обработки данных, а также в условиях их поиска. Стандарт SQL предлагает ряд числовых функций с очевидной семантикой. Часть функций перечислены ниже:

**ABS** абсолютное значение

**DEGREES** Возвращает для значения угла в радианах соответствующее значение в градусах.

**RAND** – Возвращает псевдослучайное значение типа float от 0 до 1.

**EXP** экспонента

**ROUND** - Возвращает числовое значение, округленное до указанной длины или точности.

**FLOOR** Возвращает наибольшее целое число, меньшее или равное указанному числовому выражению.

**LOG** логорифм

**SIN** - синус

**LOG10** десятичный логорифм

**SQRT** – корень квадратный

**PI** число 3.14

**SQUARE** – квадрат числа

**POWER** Возвращает значение указанного выражения, возведенное в заданную степень.

**TAN** - тангенс

Особой функцией является **WIDTH_BUCKET**, с помощью которой можно легко строить гистограммы:

**WIDTH_BUCKET(число, миминум, максимум, количество)**

Некоторые СУБД расширяют приведенный выше набор функций, включая другие числовые функции, например вычисления обычных и гиперболических тригонометрических функций.

**Временные функции**

Эти функции используют в качестве аргумента типы даты, времени, временной отметки или временного промежутка. Тип возвращаемого значения не всегда соответствует типу аргумента.

Функции даты и времени в Transact-SQL деляться на Функции, получающие значения системной даты и времени Функции, получающие компоненты даты и времени

Функции, получающие значения даты и времени из их компонентов Функции, получающие разность даты и времени

Функции, изменяющие значения даты и времени

Функции, устанавливающие или получающие формат сеанса Функции, проверяющие значения даты и времени. числа.

**Функции, получающие компоненты даты и времени.**

Функция извлекает из операнда указанный компонент и возвращает его в виде

**DATENAME ( datepart , date )**Здесь **date** - это выражение временного типа, а **datepart** - временная единица, которая может иметь одно из следующих значений: **YEAR, MONTH, DAY, HOUR, MINUTE, SECOND** и т.д.

**DATEPART ( *datepart,date* )** - Возвращает целое число, представляющее указанный компонент*datepart* указанной даты *date*.

**DAY (*date*)** - Возвращает целое число, представляющее день указанной даты *date*.

**MONTH ( *date* )** - Возвращает целое число, представляющее месяц указанной даты *date*.

**YEAR (*date*)** - Возвращает целое число, представляющее год указанной даты *date*.

Рассмотрим пример.

**Запрос 12. Вывести фамилии всех преподавателей родившихся в 1979 году.**

```
SELECT Name_teacher, BIRTHDAY
FROM TEACHER
WHERE DATENAME(YEAR, BIRTHDAY)=1979;
```

**Функции, получающие значения системной даты и времени**

**Функция CURRENT_TIMESTAMP** - Возвращает значение типа datetime2(7), которое содержит дату и время компьютера, на котором запущен экземпляр SQL Server. Смещение часового пояса не включается.

Эта функция возвращает текущую дату. Аргументов она не имеет.

**Функция GETDATE ( )** Возвращает значение типа datetime2(7), которое содержит дату и время компьютера, на котором запущен экземпляр SQL Server. Смещение часового пояса не включается.

**Функция GETUTCDATE ( )** Возвращает значение типа datetime2(7), которое содержит дату и время компьютера, на котором запущен экземпляр SQL Server. Возвращаемые дата и время отображаются в формате UTC.

Многие СУБД существенно расширяют список функций, оперирующих датой и временем. Далее мы приведем некоторые из важных функций этого типа, которые используются в Oracle. Напоминаем, что тип DATE в Oracle содержит в себе как дату, так и время.

**Функции, получающие значения даты и времени из их компонентов**

**Функция DATEADD (datepart, number , date )** Возвращает новое значение datetime, добавляя интервал к указанной части datepart заданной даты date.

Добавляет к дате, указанной в первом аргументе, количество месяцев второго аргумента.

Dateadd (компонент, кол-во , дата)

Здесь кол-во - это количество прибавляемых лет, месяцев, дней и т.д., а компонент

- временная единица, которая может иметь одно из следующих значений: **YEAR,**

**MONTH, DAY, HOUR, MINUTE, SECOND**.

Например, **DATEADD(month, 1, '2006-08-30')**

**Запрос 13. Осуществить пересчет даты приема на работу преподавателя на фамилию начинающуюся на букву С в сторону увеличения на 3 месяца.**

```
SELECT NAME_TEACHER, DATA_HIRE AS ' Дата приема ',
DATEADD(month, 3, DATA_HIRE) AS ' Плюс 3 месяца ' FROM TEACHER
WHERE (NAME_TEACHER) LIKE 'C%';
```

**Функция EOMONTH**

**EOMONTH (start_date [, month_to_add ])**

Возвращает дату последнего дня того месяца, который указан в аргументе. Обычно используется для определения, сколько дней осталось до конца месяца.

**LAST_DAY(дата)**

**Функция DATEDIFF**

**DATEDIFF ( datepart , startdate , enddate )**

Возвращает количество пересеченных границ (целое число со знаком), указанных аргументом datepart, за период времени, указанный аргументами startdate и enddate.

**Запрос 14. Например, если вы хотите узнать, сколько месяцев уже проработал Статывка, можно выполнить такой запрос:**

```
SELECT 'Статывка проработал ' || R0UND(DATEDIFF(MONTH,GETDATE(),
DATA_HIRE),1) ||
' месяцев' AS "Стаж Статывки" FROM TEACHER
WHERE NAME_TEACHER LIKE 'Статыв%';
```

**Функция NEXT_DAY**

Возвращает ближайшую к первому параметру дату, в которой название дня недели совпадает с указанным во втором параметре.

**NEXT_DAY(дата, день_недели) Функции преобразования**

Стандарт SQL предлагает единственную функцию преобразования данных из одного типа в другой — это функция CAST.

**Функция CAST и CONVERT**

Производит преобразование выражения, заданного первым аргументом, в тип, заданный вторым аргументом. Преобразование допускается только для определенных пар типов данных.

**CAST ( expression AS data_type [ ( length ) ] ) CONVERT ( data_type [ ( length ) ] , expression [ , style ] )** Например

**CAST(10.3496847 AS money) CAST(10.6496 AS int)**

**Тема 2. Группировка и сортировка**

В этом уроке мы рассмотрим еще три фразы предложения SELECT, а именно:

HAVING, GROUP BY и ORDER BY.

Первая из них позволяет группировать строки таблицы и применять к созданным группам агрегатные функции.

Рассмотрим простейшие варианты группировки. Фраза HAVING используется вместе с фразой GROUP BY и позволяет формулировать условия на группах строк для дополнительного отбора.

Наконец, фраза ORDER BY позволяет сортировать строки результирующей таблицы.

Запросы с группировкой строк

Часто при создании отчетов появляется необходимость в формировании промежуточных итоговых значений, то есть относящихся к данным не всей таблицы, а ее частей.

Именно для этого предназначена фраза GROUP BY. Она позволяет все множество строк таблицы разделить на группы по признаку равенства значений одного или нескольких столбцов (и выражений над ними).

Фраза GROUP BY должна располагаться вслед за фразой WHERE (если она отсутствует, то за фразой FROM).

Общий синтаксис фразы GROUP BY следующий:

**GROUP BY выражение[, выражение]...**

При наличии фразы GROUP BY фраза SELECT применяется к каждой группе, сформированной фразой группировки. В этом случае и действие агрегатных функций, указанных во фразе SELECT, будет распространяться не на всю результирующую таблицу, а только на строки в пределах каждой группы. Каждое выражение в списке фразы SELECT должно принимать единственное значение для группы, то есть оно может быть:

константой;

- агрегатной функцией, которая оперирует всеми значениями аргумента в пределах группы и агрегирует их в одно значение (например, в сумму); выражением, идентичным стоящему во фразе GROUP BY;

выражением, объединяющим приведенные выше варианты.

Рассмотрим возможности фразы GROUP BY, переходя от простых вариантов ее использования к более сложным.

**Задание**

Лабораторную работу следует выполнять в следующем порядке:

1. Изучить синтаксис создания запросов с использованием функций, группировки и сортировки данных, язык манипулирования данных на примерах запросов, использовать встроенные функции к учебной базе данных "**University.mdf**".

2. Выполнить в окне "SQL Editor" **41** запросов к базе данных "**University.mdf**", согласно приведенным в практической работе образцам выполнения запросов и сохранить их в файле "**Lab7.sql**" в своей рабочей папке.

**Лабораторная работа № 13 Организация локальной сети. Настройка локальной сети ( 6 часов)**

**Цель работы:** изучить способы организации локальных сетей и их настройку для ра-боты с базой данных.

**ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

Многие коммерческие предприятия имеют данные, доступ к которым предлагается пользователям вне компании - другим предприятиям, покупателям или производителям Напри- мер, предприятие может предлагать потенциальным покупателям доступ к дополнительной ин-формации о продуктах предприятия в надежде увеличить сбыт Должны быть учтены и нужды служащих предприятия Например, доступной может быть служебная информация о графиках работы и отпусков, обучении персонала, политике компании и т п Подобная база данных может быть создана и сделана легко доступной для пользователей средствами SQL и Internet

**Прикладная часть** клиент-серверного приложения (back-end application) состоит из сер- вера базы данных, источников данных и соответствующего промежуточного программного обеспечения, используемого для подключения приложения к Web или удаленной базе данных по локальной сети.

Еще раз напомним, что к наиболее распространенным серверам баз данных относят Oracle, Informix, Sybase, Microsoft SQL Server и Borland InterBase Именно с сервера начинается разворачивание приложения баз данных либо на все предприятие в рамках локальной сети (LAN) или сети intranet предприятия, либо в Internet Разворачивание (porting) представляет со- бой процесс внедрения приложения в среду, доступную пользователям Сервер базы данных должен быть установлен на рабочем месте администратора базы данных, который, в свою оче- редь, должен понимать и производственные потребности предприятия, и требования самого приложения

Промежуточное программное обеспечение приложения состоит из сервера Web и средств, позволяющих подключить сервер Web к серверу базы данных Главной целью в данном случае является наличие в Web приложения, предоставляющего доступ к корпоративным дан-ным

**Интерфейсная часть**

Интерфейсная часть приложения (front-end application) - это та часть приложения, с ко- торой имеет дело пользователь Интерфейсная часть приложения может быть коммерческим продуктом какой-нибудь компании, производящей программное обеспечение на продажу, либопродуктом, разработанным внутри предприятия с применением различных программных средств.

До того, как на рынке сложилось имеющееся сегодня разнообразие приложений, пред-лагающих интерфейс пользователя базы данных, пользователю необходимо было уметь про-граммировать на языках типа C HTML или любом другом из множества процедурных языков программирования, с помощью которых разрабатывались приложения для Web Языки типа ANSI C, COBOL, FORTRAN или Pascal использовались для разработки интерфейсной части внутри предприятия, и соответствующий интерфейс пользователя был, как правило, текстовым Сегодня большинство новых приложений интерфейсной части предлагают графический поль-зовательский интерфейс (GUI)

Интерфейсная часть приложения призвана обеспечить пользователю простоту доступа к базе данных и работы с ней Внутренние процессы программный код и происходящие в ней события должны быть незаметными для пользователя Интерфейсная

часть приложения должна быть разработана для того чтобы по возможности избавить пользователя от необходимости те- ряться в догадках и интуитивно чувствовать систему в целом Новые технологии позволяют сделать приложения более понятными и простыми в применении что дает пользователю воз- можность сосредоточиться на решении своих конкретных задач повышая в конечном итоге эф- фективность своего труда.Имеющиеся на сегодня средства разработки приложений достаточно просты в примене- нии и объектно-ориентированны, что достигается использованием в них пиктограмм, возмож- ностей перетаскивания объектов с помощью мыши, а также различных мастеров, автоматиче- ски генерирующих объекты с заданными свойствами. Среди наиболее популярных средств разработки Web-приложений следует упомянуть C++Builder, IntraBuilder фирмы Borland и Visual J++, C++ фирмы Microsoft. Для разработки программ, предназначенных для работы в рамках локальной сети предприятия, используют PowerBuilder фирмы Powersoft, Developer/2000 фирмы Oracle Corporation, Visual Basic фирмы Microsoft и Delphi фирмы Borland.

Прикладная часть располагается на сервере, там же размещается и сама база данных. Пользователями прикладной части являются разработчики базы данных, программисты, адми- нистраторы базы данных, системные администраторы и системные аналитики. Интерфейсная часть приложения размещается на машинах-клиентах, которыми обычно являются персональ- ные компьютеры конечных пользователей. Интерфейсная часть приложения рассчитана на са- мую широкую аудиторию пользователей, включающую операторов ввода данных, бухгалтеров и т. д. Пользователь должен иметь возможность доступа к базе данных по сети-и такая сеть может быть как локальной (LAN), так и глобальной (WAN). Для предоставления пользователю такой возможности используется промежуточное программное обеспечение (например, драй- вер ODBC).

**Удаленный доступ к базе данных**

База данных может быть локальной, и тогда вы имеете возможность подключится к ней непосредственно. Но, как правило, пользователю требуется доступ к базе данных, которая нахо- дится на некотором удалении от его системы. Удаленная база данных - это некоторая база дан- ных, не являющаяся локальной, т. е. расположенной на том сервере, к которому вы подключены в данный момент, и предполагающая для доступа к ней использование сети и определенных сетевых протоколов.

Доступ к удаленной базе данных можно осуществить несколькими способами. Говоря в общем, доступ к удаленной базе данных осуществляется с помощью Подключения к сети или Internet посредством использования промежуточного программного обеспечения (например, стандартных средств ODBC).

Локальный сервер базы данных и главный локальный сервер часто оказываются одним и тем же объектом, поскольку база данных, как правило, размещается на главном локальном сервере. Но подключиться к удаленной базе данных с главного локального сервера можно и без подключения к локальной базе данных. Для конечного пользователя чаще всего предлагается подключение к удаленной базе данных средствами интерфейсного приложения. Во всех этих случаях используется передача запросов к базе данных по сети.

Технология ODBC (Open Database Connectivity - открытый интерфейс доступа к базам данных) обеспечивает возможность доступа к удаленным базам данных с помощью подходя- щего драйвера. Драйвер ODBC используется интерфейсным приложением для получения до- ступа к прикладной части базы данных для взаимодействия с данными нижнего уровня. Для доступа к удаленной базе данных, кроме того, может понадобиться и сетевой драйвер. Прило- жение вызывает функции ODBC, а соответствующий драйвер обеспечивает загрузку драйвера ODBC. Драйвер ODBC обрабатывает вызов функции, пересылает запрос к базе данных и воз- вращает результат этого запроса. На сегодня ODBC является стандартом, используемым целым рядом продуктов, в частности, PowerBuilder, FoxPro, Visual C++, Visual Basic, Delphi, MicrosoftAccess и многими другими.

Как часть ODBC, любой производитель систем управления базами данных предлагает со своими базами данных программный интерфейс приложения (API). Для примера из таких предлагаемых продуктов можно отметить Open Call Interface (OCI) фирмы Oracle и SQLGateway или SQLRouter фирмы Centura.

Другие интерфейсы доступа к данным

В дополнение к драйверу ODBC многие производители систем управления базами дан- ных предлагают свое программное обеспечение, предназначенное для организации доступа к

удаленным базам данных. Каждый из таких продуктов оказывается специфическим для си- стемы управления базами данных конкретного производителя и, вообще говоря, не предпола- гает переносимости на другие типы серверов баз данных.

Oracle Corporation для организации доступа к удаленным базам данных предлагает свой продукт под именем Net8. Net8 можно использовать практически с любыми сетевыми протоко- лами, в частности, TCP/IP, OSI, SPX/IPX и многими другими. Кроме того, Net8 может работать под управлением почти любой из наиболее распространенных операционных систем.

Sybase Incorporated предлагает свой продукт под именем Open Client/C Developers Kit, поддерживающий продукты других производителей, в частности Net8 фирмы Oracle.

Доступ к удаленным базам данных с помощью интерфейса Web

Доступ к удаленным базам данных посредством интерфейса Web подобен доступу к ба- зам данных по локальной сети Главное отличие состоит в том, что в Web все запросы к базе данных направляются через Web-сервер.

Конечный пользователь инициирует доступ к удаленной базе данных с помощью бро- узера Web. Броузер Web используется для связывания с заданным URL или адресом IP в Internet, соответствующим нужному серверу Web. Сервер Web, проверив имя и пароль пользо- вателя, пересылает пользовательский запрос базе данных, которая, в свою очередь, тоже может потребовать проверки имени и пароля. Затем сервер базы данных вернет результаты запроса серверу Web, а последний отобразит эти результаты в окне пользовательского броузера Web. Несанкционированный доступ к конкретному серверу может пресекаться с помощью бранд- мауэра (firewall).

**Брандмауэр** (firewall) - это аппаратно-программная система межсетевой защиты от не- санкционированного доступа к серверу. Для зашиты от несанкционированного доступа к сер- веру может использоваться как одна, так и несколько таких систем. Точно также одна или не- сколько систем защиты могут использоваться для управления доступом к серверу базы данных и к самой базе данных.

При пересылке информации по Web следует принять все возможные меры безопасности на всех уровнях. К таким уровням можно отнести сервер Web, главный локальный сервер и удаленную базу данных Частные данные, такие как идентификационные номера служащих, всегда должны быть защищены от доступа к ним случайных лиц и не должны распространяться в Web.

**SQL и Internet**

SQL можно использовать в рамках приложений, создаваемых средствами C или COBOL. Точно так же SQL можно использовать и в Internet-приложениях, создаваемых средствами та- ких языков программирования, как Java. Текст HTML тоже можно транслировать в запрос SQL, чтобы потом с помощью интерфейсного приложения Web переслать этот запрос удаленной базе данных. Возвращенный базой данных результат затем транслируется обратно в текст HTML и отображается броузером Web на экране пользователя, пославшего запрос. В следующих разде- лах использование SQL в Internet обсуждается подробнее.

С изобретением и распространением Internet по всему миру данные стали доступными покупателям и производителям в любой стране. Такие данные обычно бывают доступными

только для чтения с помощью соответствующего интерфейсного приложения.

Предназначенные для покупателей данные могут состоять из общей информации для покупателей, информации о конкретных продуктах, бланков заказов, информации о текущих и выполненных ранее заказах и т.д. Частная информация, например, информация о корпоратив- ной стратегии и о служащих компании доступной быть не должна.

Наличие информационной странички в Web стало почти обязательным для компаний, стремящихся успешно конкурировать с другими в своем бизнесе. Страничка Web оказываетсявесьма эффективным средством информирования большого числа потенциальных покупателейоб услугах, продуктах и других аспектах деятельности компании, не требуя при этом чрезмер-ных затрат.

Предоставление доступа к данным служащим и привилегированным клиентам

С помощью Internet или сети intranet компании базу данных можно сделать доступной для служащих этой компании или ее клиентов. Использование технологий Internet оказывается весьма удобным для информирования служащих о политике компании, преимуществах работыв ней, обучающих программах и т. п.

**Интерфейсные приложения Web, использующие SQL**

Имеется целый ряд приложений, обеспечивающих доступ к базам данных. Многие из таких приложений предлагают графический интерфейс пользователя, так что пользователю даже нет необходимости понимать SQL, чтобы составить запрос к базе данных. В таких прило- жениях пользователю предлагается указывать и щелкать мышью на объектах, представляющих таблицы, манипулировать данными этих объектов, задавать критерии отбора возвращаемых данных и т. д. Такие приложения часто разрабатываются и настраиваются в полном соответ- ствии с конкретными требованиями конкретной компании.

SQL и intranet

IBM изначально создавала SQL для доступа к базам данных, размещенным на мэйнфрей- мах, с клиентских машин пользователей. Пользователи при этом связывались с мэйнфреймами по локальной сети. Позже SQL стал стандартным языком коммуникации пользователей с ба- зами данных. Intranet, по сути, является миниатюрным аналогом Internet. Основным различием между ними является то, что intranet предна-шачлегся для использования внутри некоторой ор- ганизации, а Internet открыта для доступа всем. Пользовательский (клиентский) интерфейс в intranet остается тем же, что и в модели клиент/сервер. Запросы SQL направляются базе данных сервером Web с использованием соответствующего языка (например, HTML).

Безопасность в рамках базы данных значительно выше, чем в Internet. Поэтому всегда используйте средства безопасности, предлагаемые вашим сервером базы данных.

**ХОД РАБОТЫ**

**Задание 1.** Войдите в Internet и ознакомьтесь с информационными страницами несколь- ких из представленных там компаний. Если ваша компания тоже имеет информационную стра- ницу в Web, сравните ее с информационными страницами конкурентов. Ответьте для себя на следующие вопросы в отношении просмотренных страниц.

а. Открывается ли страница быстро или ее открытие тормозится наличием слишком большого числа графических изображений?

б. Интересно ли читать представленную на странице информацию?

в. Получили ли вы в результате чтения имеющейся на странице информации представ- ление о предлагаемых компанией услугах и продуктах и о компании в целом?

г. Если на странице предлагается доступ к некоторой базе данных, то достаточно ли быстро осуществляется такой доступ?

д. Можно ли сделать вывод об использовании на данной странице Web каких-либо средств безопасности?

**Задание 2**. Если в вашей компании используется intranet, войдите в сеть и посмотрите,

какая информация о компании там представлена. Доступна ли там какая-нибудь база данных? Если да, то кто является производителем соответствующей системы управления базами дан- ных? Какого типа интерфейсные приложения предлагаются при этом конечному пользова- телю?

**Лабораторная работа № 14 Установка и настройка SQL-сервера (6 часов)**

**Цель работы:** изучить этапы установки и настройка SQl-сервера

Для установки нам потребуется дистрибутив SQL Server Express with Tools с сайта Microsoft. Данная сборка уже включает в себя графическое средство управления — среду SQLServer Management Studio Express.

1. Запустить установщик с правами администратора на данном компьютере. В разделе

«Планирование» нажать пункт «Средство проверки конфигурации»: Нажать кнопку «Показать подробности» и убедиться, что все проверки успешно пройдены. Если будут обнаружены ка- кие-то проблемы, то необходимо их устранить и запустить повторную проверку кнопкой

«Включить заново». Затем закрыть данное окно кнопкой «ОК».

2. Нажать на раздел «Установка» и затем пункт «Новая установка изолированного SQLServer или добавление компонентов …».

3. Нажать кнопку «Показать подробности» и убедиться, что все проверки успешно пройдены. Если будут обнаружены какие-то проблемы, то необходимо их устранить и запу- стить повторную проверку кнопкой «Включить заново». Затем нажать кнопку «ОК».

4. Ввести приобретенный ключ продукта (для бесплатной версии не требуется) и нажать кнопку «Далее». Прочитать лицензию, установить галочку и нажать кнопку «Далее». Нажать кнопку «Установить».

5. Нажать кнопку «Показать подробности» и убедиться, что все проверки успешно пройдены. Если будут обнаружены какие-то проблемы, то необходимо их устранить и запу- стить повторную проверку кнопкой «Включить заново». Затем нажать кнопку «Далее»

**Примечание**. Если появится предупреждение в строке «Брандмауэр Windows», то его можно проигнорировать – оно просто акцентирует Ваше внимание на том, что потребуется до- полнительная настройка «Брандмауэра Windows» для доступа к SQL Server с других компью- теров (см. ниже).

6. Выбрать компоненты для установки (можно воспользоваться кнопкой «Выделить все»), и нажать кнопку «Далее»:

Выбрать опцию «Экземпляр по умолчанию» и нажать кнопку «Далее»Нажать кнопку «Далее»

Выбрать опции, как показано на рисунке, и перейти на закладку «Параметры сорти- ровки»:

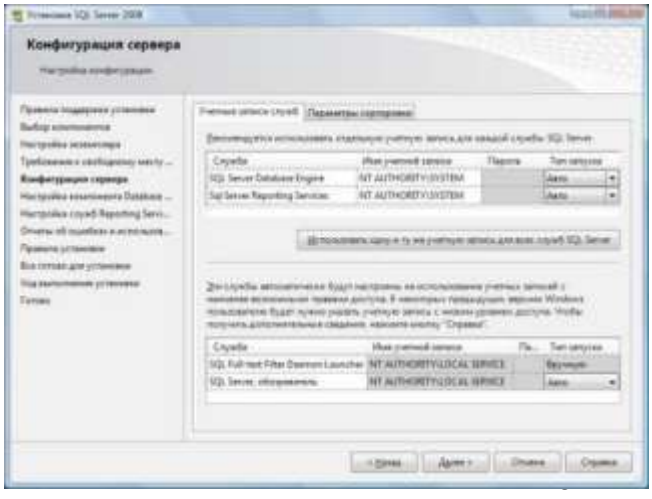


*Рисунок 8*

Выбрать опции, как показано на рис. 8, и нажать кнопку «Далее»:

Примечание: в данном пункте мы указываем кодовую страницу для не-Unicode типов данных (char, varchar, text) и порядок сортировки текстовых данных.

Выбрать опцию «Смешанный режим» и задать пароль для встроенной учетной записи администратора «sa» (эта учетная запись обладает максимальными правами доступа ко всем функциям и объектам на SQL-сервере). Дополнительно можно указать учетные записи пользо- вателей Windows или целые группы пользователей Windows, которые должны обладать макси- мальными правами доступа к SQL Server (например, встроенную группу «Администраторы»). Затем перейти на закладку «Каталоги данных»

В поле «Корневой каталог данных» ввести путь к папке, где будут размещаться файлы баз данных (рекомендуется использовать отдельный от ОС физический диск), и нажать кнопку

«Далее»

Нажать кнопку «Показать подробности» и убедиться, что все проверки успешно прой- дены. Если будут обнаружены какие-то проблемы, то необходимо их устранить и запустить повторную проверку кнопкой «Включить заново». Затем нажать кнопку «Далее»

Нажать кнопку «Установить»

После завершения установки нажать кнопку «Далее»Нажать кнопку «Закрыть».

Установка Microsoft SQL Server 2008 Express завершена!

**Лабораторная работа № 15 Создание пользовательских баз данных (10 часов)**

**Цель работы:** изучить принципы создания базы данных при помощи SQL - операторов

Язык SQL стал фактически стандартным языком доступа к базам данных. Все СУБД, претендующие на название "реляционные", реализуют тот или иной диалект SQL. Многие не- реляционные системы также имеют в настоящее время средства доступа к реляционным дан- ным. Целью стандартизации является переносимость приложений между различными СУБД.

Нужно заметить, что в настоящее время, ни одна система не реализует стандарт SQL в полном объеме. Кроме того, во всех диалектах языка имеются возможности, не являющиеся стандартными. Таким образом, можно сказать, что каждый диалект - это надмножество неко- торого подмножества стандарта SQL. Это затрудняет переносимость приложений, разработан-ных для одних СУБД в другие СУБД.

Язык SQL оперирует терминами, несколько отличающимися от терминов реляционной теории, например, вместо "отношений" используются "таблицы", вместо "кортежей" - "строки",вместо "атрибутов" - "колонки" или "столбцы".

Стандарт языка SQL, хотя и основан на реляционной теории, но во многих местах от- ходит он нее. Например, отношение в реляционной модели данных не допускает наличия оди- наковых кортежей, а таблицы в терминологии SQL могут иметь одинаковые строки. Имеются и другие отличия.

Язык SQL является реляционно полным. Это означает, что любой оператор реляцион- ной алгебры может быть выражен подходящим оператором SQL.

**Операторы SQL**

Основу языка SQL составляют операторы, условно разбитые не несколько групп по выполняемым функциям.

Можно выделить следующие группы операторов (перечислены не все операторы SQL): Операторы DDL (Data Definition Language) - операторы определения объектов базы данных

CREATE SCHEMA - создать схему базы данных DROP SHEMA - удалить схему базы

данных CREATE TABLE - создать таблицу

ALTER TABLE - изменить таблицу DROP TABLE - удалить таблицу CREATE DOMAIN - создать домен ALTER DOMAIN - изменить домен DROP DOMAIN - удалить домен

CREATE COLLATION - создать последовательность DROP COLLATION - удалить последовательность CREATE VIEW - создать представление

DROP VIEW - удалить представление

Операторы DML (Data Manipulation Language) - операторы манипулирования данными SELECT - отобрать строки из таблиц

INSERT - добавить строки в таблицу UPDATE - изменить строки в таблицеDELETE - удалить строки в таблице

COMMIT - зафиксировать внесенные изменения ROLLBACK - откатить внесеи

CREATE ASSERTION - создать ограничение DROP ASSERTION - удалить ограничение

GRANT - предоставить привилегии пользователю или приложению на манипулирование объектами

REVOKE - отменить привилегии пользователя или приложения

Кроме того, есть группы операторов установки параметров сеанса, получения информации о базе данных, операторы статического SQL, операторы динамического SQL.

Наиболее важными для пользователя являются операторы манипулирования данными (DML).

Примеры использования операторов манипулирования данными INSERT - вставка строк в таблицу

**Задание 1.** Вставка одной строки в таблицу:

INSERT INTO

P (PNUM, PNAME) VALUES (4, "Иванов");

**Задание 2**. Вставка в таблицу нескольких строк, выбранных из другой таблицы (в таблицу TMP_TABLE вставляются данные о поставщиках из таблицы P, имеющие номера, боль-шие 2):

INSERT INTO

TMP_TABLE (PNUM, PNAME)SELECT PNUM, PNAME FROM P

WHERE P.PNUM>2;

UPDATE - обновление строк в таблице

**Задание 3**. Обновление нескольких строк в таблице:

UPDATE P

SET PNAME = "Пушников"WHERE P.PNUM = 1;

DELETE - удаление строк в таблице

Пример 4. Удаление нескольких строк в таблице:

DELETE FROM P WHERE P.PNUM = 1;

**Задание 5**. Удаление всех строк в таблице:

DELETE FROM P;

Примеры использования оператора SELECT

Оператор SELECT является фактически самым важным для пользователя и самым сложным оператором SQL. Он предназначен для выборки данных из таблиц, т.е. он, собственно, и реализует одно их основных назначение базы данных - предоставлять информациюпользователю.

Оператор SELECT всегда выполняется над некоторыми таблицами, входящими в базу данных.

Замечание. На самом деле в базах данных могут быть не только постоянно хранимые таблицы, а также временные таблицы и так называемые представления. Представления - это просто хранящиеся в базе данные SELECT-выражения. С точки зрения

пользователей представ-ления - это таблица, которая не хранится постоянно в базе данных, а "возникает" в момент об- ращения к ней. С точки зрения оператора SELECT и постоянно хранимые таблицы, и времен- ные таблицы и представления выглядят совершенно одинаково. Конечно, при реальном выпол- нении оператора SELECT системой учитываются различия между хранимыми таблицами и представлениями, но эти различия *скрыты* от пользователя.

Результатом выполнения оператора SELECT всегда является таблица. Таким образом, по результатам действий оператор SELECT похож на операторы реляционной алгебры. Любой оператор реляционной алгебры может быть выражен подходящим образом сформулированным оператором SELECT. Сложность оператора SELECT определяется тем, что он содержит в себе

все возможности реляционной алгебры, а также дополнительные возможности, которых в ре-ляционной алгебре нет.

**Отбор данных из одной таблицы**

**Задание 6**. Выбрать все данные из таблицы поставщиков (ключевые слова ***SELECT***… ***FROM***…):

```
SELECT *FROM P;
```

Замечание. В результате получим новую таблицу, содержащую полную копию данных из исходной таблицы P.

**Задание 7**. Выбрать все строки из таблицы поставщиков, удовлетворяющих некоторому условию (ключевое слово ***WHERE***…):

```
SELECT *FROM P
WHERE P.PNUM > 2;
```

Замечание. В качестве условия в разделе WHERE можно использовать сложные логические выражения, использующие поля таблиц, константы, сравнения (>, <, = и т.д.), скобки, союзы AND и OR, отрицание NOT.

## Лабораторная работа № 16 Создание ограничений

**Цель занятия:** получение практических навыков обработки транзакций и защиты баз данных при помощи функций.

Изменения БД часто требуют выполнения нескольких запросов, например при покупке в электронном магазине требуется добавить запись в таблицу заказов и уменьшить число товарных позиций на складе. В промышленных БД одно событие может затрагивать большее число таблиц и требовать многочисленных запросов.

Если на этапе выполнения одного из запросов происходит сбой, это может нарушить целостность БД (товар может быть продан, а число товарных позиций на складе не обновлено). Чтобы сохранить целостность БД, все изменения должны выполняться как единое целое. Либо все изменения успешно выполняются, либо, в случае сбоя, БД принимает состояние, которое было до начала изменений. Это обеспечивается средствами обработки транзакций.

*Транзакция* – последовательность операторов SQL, выполняющихся как единая операция, которая не прерывается другими клиентами. Пока происходит работа с записями таблицы (обновление или удаление), никто другой не может получить доступ к этим записям, т. к. MySQL автоматически блокирует доступ к ним.

Таблицы *ISAM, MyISAM* и *HEAP* не поддерживают транзакции. В настоящий момент их поддержка осуществляется только в таблицах *BDB* и *InnoDB.*

Транзакции позволяют объединять операторы в группу и гарантировать, что все операторы группы будут выполнены успешно. Если часть транзакции выполняется со сбоем, результаты выполнения всех операторов транзакции до места сбоя отменяются, приводя БД к виду, в котором она была до выполнения транзакции.

По умолчанию MySQL работает в режиме автоматического завершения транзакций, т.

е. как только выполняется оператор обновления данных, который модифицирует таблицу, изменения тут же сохраняются на диске. Чтобы объединить операторы в транзакцию, следует отключить этот режим: *SET AUTOCOMMIT=0;*

После отключения режима для завершения транзакции необходимо ввести оператор *COMMIT*, для отката – *ROLLBACK*.

Включить режим автоматического завершения транзакций для отдельной последовательности операторов можно оператором *START TRANSACTION*.

Для таблиц *InnoDB* есть операторы *SAVEPOINT* и *ROLLBACK TO SAVEPOINT*,

```
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO catalogs VALUES(NULL,'Периферия');
Query OK, 1 row affected (0.00 sec)

mysql> SAVEPOINT point1;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO catalogs VALUES(NULL,'Разное');
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM catalogs;
+--------+--------------------+
| cat_ID | cat_name           |
+--------+--------------------+
|      1 | Программирование   |
|      2 | Интернет           |
|      3 | Базы данных        |
|      4 | Сети               |
|      5 | Мультимедиа        |
|     12 | Периферия          |
|     13 | Разное             |
+--------+--------------------+
7 rows in set (0.00 sec)

mysql> ROLLBACK TO SAVEPOINT point1;
Query OK, 0 rows affected (0.02 sec)
```

которые позволяют работать с именованными точками начала транзакции.

```
mysql> SELECT * FROM catalogs;
+--------+--------------------+
| cat_ID | cat_name           |
+--------+--------------------+
|      1 | Программирование   |
|      2 | Интернет           |
|      3 | Базы данных        |
|      4 | Сети               |
|      5 | Мультимедиа        |
|     12 | Периферия          |
+--------+--------------------+
6 rows in set (0.00 sec)
```

Оператор *SAVEPOINT* устанавливает именованную точку начала транзакции с именем *point1*. Оператор *ROLLBACK TO SAVEPOINT point1* откатывает транзакцию к состоянию, в котором находилась БД на момент установки именованной точки. Все точки сохранения транзакций удаляются, если выполняются операторы *COMMIT* или *ROLLBACK* без указания имени точки сохранения.

**Управление правами пользователей**

СУБД MySQL является многопользовательской средой, поэтому для доступа к таблицам БД могут быть созданы различные учетные записи с разным уровнем привилегий. Учетной записи редактора можно предоставить привилегии на просмотр таблицы, добавление новых записей и обновление уже существующих. Администратору БД можно предоставить более широкие полномочия (возможность создания таблиц, редактирования и удаления уже существующих). Для пользователя БД достаточно лишь просмотра таблиц.

Рассмотрим следующие вопросы:

- создание, редактирование и удаление учетных записей пользователей;
- назначение и отмена привилегий.

Учетная запись является составной и принимает форму *'username' @ 'host'*, где *username* – имя пользователя, а *host* – наименование хоста, с которого пользователь может обращаться к серверу. Например, записи *'root' @ '127.0.0.1'* и *'wet' @ '62.78.56.34'* означают, что пользователь с именем *root* может обращаться с хоста, на котором расположен сервер, а *wet* – только с хоста с IP- адресом 62.78.56.34.

IP-адрес 127.0.0.1 всегда относится к локальному хосту. Если сервер и клиент

установлены на одном хосте, то сервер слушает соединения по этому адресу, а клиент отправляет на него SQL- запросы.

IP-адрес 127.0.0.1 имеет псевдоним *localhost*, поэтому учетные записи вида *'root' @ '127.0.0.1'* можно записывать в виде *'root' @ 'localhost'*.

Число адресов, с которых необходимо обеспечить доступ пользователю, может быть значительным. Для задания диапазона в имени хоста используется специальный символ "%". Так, учетная запись *'wet' @ '%'* позволяет пользователю *wet* обращаться к серверу MySQL с любых компьютеров сети.

Все учетные записи хранятся в таблице *user* системной базы данных с именем *mysql*.

```
mysql> SELECT Host,User,Password FROM mysql.user;
+-----------------------+------+----------+
| Host                  | User | Password |
+-----------------------+------+----------+
| localhost             | root |          |
| production.mysql.com  | root |          |
| 127.0.0.1             | root |          |
| localhost             |      |          |
| production.mysql.com  |      |          |
+-----------------------+------+----------+
5 rows in set (0.27 sec)
```

После первой инсталляции содержимое таблицы *user* выглядит так, как показано в листинге.

**Создание новой учетной записи**. Создать учетную запись позволяет оператор

*CREATE USER 'username' @ 'host' [IDENTIFIED BY [PASSWORD] 'пароль'];*

Оператор создает новую учетную запись с необязательным паролем. Если пароль не указан, в его качестве выступает пустая строка. Разумно хранить пароль в виде хэш-кода, полученного в результате необратимого шифрования. Чтобы воспользоваться этим механизмом авторизации, необходимо поместить между ключевым словом *IDENTIFIED BY* и паролем ключевое слово *PASSWORD*.

**Удаление учетной записи**. Удалить учетную запись позволяет оператор

*DROP USER 'username' @ 'host';*

**Изменение имени пользователя в учетной записи**. Осуществляется с помощью оператора

*RENAME USER старое_имя TO новое_имя;***Назначение привилегий**. Рассмотренные выше операторы позволяют создавать, удалять и редактировать учетные записи, но они не позволяют изменять привилегии пользователя – сооб- щать MySQL, какой пользователь имеет право только на чтение информации, какой на чтение и редактирование, а кому предоставлены права изменять структуру БД и создавать учетные записи.

Для решения этих задач предназначены операторы *GRANT* (назначает привилегии) и *REVOKE* (удаляет привилегии). Если учетной записи, которая показана в операторе *GRANT*, не существует, то она автоматически создается. Удаление всех привилегий с помощью оператора *REVOKE* не приводит к автоматическому уничтожению учетной записи. Для удаления пользователя необходимо воспользоваться оператором *DROP USER*.

```
mysql> GRANT ALL ON *.* TO 'wet'@'localhost' IDENTIFIED BY 'pass';
Query OK, 0 rows affected (0.17 sec)
```

В простейшем случае оператор *GRANT* выглядит следующим образом:

Данный запрос создает пользователя с именем *wet* и паролем *pass*, который может обращаться к серверу с локального хоста (*localhost*) и имеет все права (*ALL*) для всех баз данных (*.*). Если такой пользователь существует, то его привилегии будут изменены на *ALL*.

Вместо ключевого слова *ALL* можно использовать любое из ключевых слов, представленных в табл. 17.1. Ключевое слово *ON* в операторе *GRANT* задает уровень привилегий, которые могут быть заданы на одном из четырех уровней, представленных в табл. 10. Для таблиц можно установить только следующие типы привилегий: *SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, GRANT OPTION, INDEX* и *ALTER*. Это следует учитывать при использовании конструкции *GRANT ALL*, которая назначает привилегии на текущем уровне. Так, запрос уровня базы данных *GRANT ALL ON db.** не предоставляет

никаких глобальных привилегий.

**Отмена привилегий**. Для отмены привилегий используется оператор *REVOKE*:

```
mysql> REVOKE DELETE, UPDATE ON *.* FROM 'wet'@'localhost';
Query OK, 0 rows affected (0.02 sec)
```

Оператор *REVOKE* отменяет привилегии, но не удаляет учетные записи, для их удаления необходимо воспользоваться оператором *DROP USER*.

Таблица 17.1. Вместо ключевого слова ALL

| **Привилегия** | **Операция, разрешенная привилегией** |
|---|---|
| *ALL [PRIVILEGES]* | Комбинация всех привилегий, за исключением привилегии *GRANT OPTION*, которая задается отдельно |
| *ALTER* | Позволяет редактировать таблицу с помощью оператора *ALTER TABLE* |
| *ALTER ROUTINE* | Позволяет редактировать или удалять хранимую процедуру |
| *CREATE* | Позволяет создавать таблицу при помощи оператора *CREATE TABLE* |
| *CREATE ROUTINE* | Позволяет создавать хранимую процедуру |
| *CREATE TEMPORARY TABLES* | Позволяет создавать временные таблицы |
| *CREATE USER* | Позволяет работать с учетными записями с помощью *CREATE USER, DROP USER, RENAME USER* и *REVOKE ALL PRIVILEGES* |
| *CREATE VIEW* | Позволяет создавать представление с помощью оператора *CREATE VIEW* |
| *DELETE* | Позволяет удалять записи при помощи оператора *DELETE* |
| *DROP* | Позволяет удалять таблицы при помощи оператора *DROP TABLE* |
| *EXECUTE* | Позволяет выполнять хранимые процедуры |
| *INDEX* | Позволяет работать с индексами, в частности, использовать операторы *CREATE INDEX* и *DROP INDEX* |
| *INSERT* | Позволяет добавлять в таблицу новые записи оператором *INSERT* |
| *LOCK TABLES* | Позволяет осуществлять блокировки таблиц при помощи операторов *LOCK TABLES* и *UNLOCK TABLES*. Для вступления в действие этой привилегии должна быть установлена привилегия *SELECT* |
| *SELECT* | Позволяет осуществлять выборки таблиц оператором *SELECT* |
| *SHOW DATABASES* | Позволяет просматривать список всех таблиц на сервере при помощи оператора *SHOW DATABASES* |
| *SHOW VIEW* | Позволяет использовать оператор *SHOW CREATE VIEW* |
| *UPDATE* | Позволяет обновлять содержимое таблиц оператором *UPDATE* |
| *USAGE* | Синоним для статуса «отсутствуют привилегии» |
| *GRANT OPTION* | Позволяет управлять привилегиями других пользователей, *REVOKE* |

Таблица 17.2. Вместо ключевого слова ON

| **Ключевое слово *ON*** | **Уровень** |
|---|---|

| | |
|---|---|
| *ON *.** | Глобальный уровень – пользователь с полномочиями на глобальном уровне может обращаться ко всем БД и таблицам, входящим в их состав |
| *ON db.** | Уровень базы данных – привилегии распространяются на таблицы базы данных *db* |
| *ON db.tbl* | Уровень таблицы – привилегии распространяются на таблицу *tbl* базы данных *db* |
| *ON db.tbl* | Уровень столбца – привилегии касаются отдельных столбцов в таблице *tbl* базы данных *db*. Список столбцов указывается в скобках через запятую после<br>ключевых слов *SELECT, INSERT, UPDATE* |

**Задание для практической работы**

- При выполнении практической работы необходимо:
- создать транзакцию, произвести ее откат и фиксацию;
- составить отчет по практической работе.

**Задание 1. Обработка транзакций**

Для выполнения задания объединим несколько операций по добавлению в таблицу *catalogs* новых каталогов, а затем произведем откат транзакции, т. е. отмену произведенных действий. Отключаем режим автоматического завершения, добавляем новые записи и проверяем, добавились записи или нет.

```
mysql> SET AUTOCOMMIT=0;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO catalogs VALUES(NULL,'Аппаратура');
Query OK, 1 row affected (0.06 sec)

mysql> INSERT INTO catalogs VALUES(NULL,'Безопасность');
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM catalogs;
+--------+--------------------+
| cat_ID | cat_name           |
+--------+--------------------+
|      1 | Программирование   |
|      2 | Интернет           |
|      3 | Базы данных        |
|      4 | Сети               |
|      5 | Мультимедиа        |
|      6 | Аппаратура         |
|      7 | Безопасность       |
+--------+--------------------+
7 rows in set (0.02 sec)
```

Откатываем транзакцию оператором *ROLLBACK* (изменения не сохранились).

```
mysql> ROLLBACK;
Query OK, 0 rows affected (0.03 sec)

mysql> SELECT * FROM catalogs;
+--------+--------------------+
| cat_ID | cat_name           |
+--------+--------------------+
|      1 | Программирование   |
|      2 | Интернет           |
|      3 | Базы данных        |
|      4 | Сети               |
|      5 | Мультимедиа        |
+--------+--------------------+
5 rows in set (0.00 sec)
```

Воспроизведем транзакцию и сохраним действия оператором *COMMIT*.

```
mysql> INSERT INTO catalogs VALUES(NULL,'Аппаратура');
Query OK, 1 row affected (0.00 sec)

mysql> INSERT INTO catalogs VALUES(NULL,'Безопасность');
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM catalogs;
+--------+--------------------+
| cat_ID | cat_name           |
+--------+--------------------+
|      1 | Программирование   |
|      2 | Интернет           |
|      3 | Базы данных        |
|      4 | Сети               |
|      5 | Мультимедиа        |
|      8 | Аппаратура         |
|      9 | Безопасность       |
+--------+--------------------+
7 rows in set (0.00 sec)
```

```
mysql> COMMIT;
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT * FROM catalogs;
+--------+-----------------+
| cat_ID | cat_name        |
+--------+-----------------+
|      1 | Программирование |
|      2 | Интернет        |
|      3 | Базы данных     |
|      4 | Сети            |
|      5 | Мультимедиа     |
|      8 | Аппаратура      |
|      9 | Безопасность    |
+--------+-----------------+
7 rows in set (0.00 sec)
```

**Задание 2. Защита данных**

Для работы выберем два компьютера, подключенных к локальной сети. На одном необходимо развернуть сервер MySQL, на другой – скопировать клиент командной строки *mysql.exe*. Определим IP-адрес сервера:

```
C:\Documents and Settings\admin>ipconfig

Настройка протокола IP для Windows

Подключение по локальной сети - Ethernet адаптер:

        DNS-суффикс этого подключения . . :
        IP-адрес  . . . . . . . . . . . . : 192.168.67.6
        Маска подсети . . . . . . . . . . : 255.255.255.0
        Основной шлюз . . . . . . . . . . : 192.168.67.254
```

Создадим новую учетную запись, позволив пользователю *user1* обращаться к серверу MySQL с любых компьютеров сети:

```
mysql> CREATE USER 'user1'@'%' IDENTIFIED BY '123';
Query OK, 0 rows affected (0.01 sec)
```

Назначим этому пользователю привилегии глобального уровня:

```
mysql> GRANT ALL ON *.* TO 'user1'@'%' IDENTIFIED BY '123';
Query OK, 0 rows affected (0.00 sec)
```

На клиентском компьютере в командной строке (например, с помощью FAR), запустим клиент командной строки в следующем формате:

```
The FAR manager, version 1.70 beta 5 (build 1634)
Copyright (C) 1996-2000 Eugene Roshal, Copyright (C) 2000-2003 FAR Group
Зарегистрирован: xUSSR регистрация
C:\>mysql -u user1 -p123 -h 192.168.67.6
```

Наблюдаем отклик удаленного сервера и работаем с ним как обычно:

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.0.51b-community-nt MySQL Community Edition (GPL)

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| book               |
| mysql              |
| test               |
+--------------------+
4 rows in set (0.00 sec)
```

**Варианты заданий к практическим работам по MySQL**

1. *Страховая компания*. Страховая компания имеет филиалы, которые характеризуются наименованием, адресом и телефоном. В филиалы обращаются клиенты с целью заключения договора о страховании. В зависимости от принимаемых на страхование объектов и страхуемых рисков договор заключается по определенному виду страхования (страхование автотранспорта от угона, страхование домашнего имущества, добровольное медицинское страхование). При заключении договора фиксируются: дата заключения, страховая сумма, вид страхования, тарифная ставка и филиал, в котором заключался договор. Договоры заключают страховые агенты. Помимо информации об агентах (фамилия, имя, отчество, адрес, телефон) нужно хранить филиал, в котором они работают. Необходимо иметь возможность рассчитывать заработную плату агентам. Заработная плата составляет

некоторый процент от страхового платежа (платеж – страховая сумма, умноженная на тарифную ставку). Процент зависит от вида страхования, по которому заключен договор.

2. *Гостиница*. Гостиница предоставляет номера клиентам. Каждый номер характеризуется вместимостью, комфортностью (люкс, полулюкс, обычный) и ценой. О клиентах собирается
определенная информация (фамилия, имя, отчество, паспортные данные, адрес жительства и некоторый комментарий). Сдача номера клиенту производится при наличии свободных мест в номерах, подходящих клиенту по указанным выше параметрам. При заселении фиксируется дата заселения. При выезде из гостиницы для каждого места запоминается дата освобождения. Необходимо также осуществлять бронирование номеров. Для постоянных клиентов, а также для определенных категорий клиентов предусмотрена система скидок. Скидки могут суммироваться.

3. *Ломбард*. В ломбард обращаются различные лица с целью получения денежных средств под залог товаров. Клиент сообщает фамилию, имя, отчество и другие паспортные данные. После оценивания стоимости принесенного в качестве залога товара работник ломбарда определяет сумму, которую готов выдать на руки клиенту, а также свои комиссионные. Кроме того определяется срок возврата денег. Если клиент согласен, то договоренности фиксируются в виде документа, деньги выдаются клиенту, а товар остается в ломбарде. Если в указанный срок не происходит возврата денег, товар переходит в собственность ломбарда. После перехода прав собственности на товар, ломбард может продавать товары по цене, меньшей или большей, чем была заявлена при сдаче. Цена может меняться несколько раз, в зависимости от ситуации на рынке (например, владелец ломбарда может устроить распродажу зимних вещей в конце зимы). Помимо текущей цены нужно хранить все возможные значения цены для данного товара.

4. *Оптово-розничная продажа товаров*. Компания торгует товарами из определенного спектра. Каждый товар характеризуется наименованием, оптовой ценой, розничной ценой и справочной информацией. В компанию обращаются покупатели, для каждого из которых в базе данных фиксируются стандартные данные (наименование, адрес, телефон, контактное лицо). По каждой сделке составляется документ, в котором наряду с покупателем фиксируются количество купленного им товара и дата покупки. Обычно покупатели в рамках одной сделки покупают не один товар, а сразу несколько. Также компания решила предоставлять скидки в зависимости от количества закупленных товаров и их общей стоимости.

5. *Ведение заказов*. Компания занимается оптовой продажей различных товаров. Каждый из товаров характеризуется ценой, справочной информацией и признаком наличия или отсутствия доставки. В компанию обращаются заказчики. Для каждого из них в базе данных запоминаются стандартные данные (наименование, адрес, телефон, контактное лицо). По каждому заказу составляется документ, в котором наряду с заказчиком фиксируются количество купленного им товара и дата покупки. Доставка товаров может производиться способами, различными по цене и скорости. Нужно хранить информацию о том, какими способами может осуществляться доставка каждого товара, и информацию о том, какой вид доставки (и какую стоимость доставки) выбрал клиент при заключении сделки.

6. *Бюро по трудоустройству*. Бюро готово искать работников для различных работодателей и вакансии для ищущих работу специалистов различного профиля. При обращении в бюро работодателя его стандартные данные (название, вид деятельности, адрес, телефон) фиксируются в базе данных. При обращении в бюро соискателя его стандартные данные (фамилия, имя, отчество, квалификация, профессия, иные данные) также фиксируются в базе данных. По каждому факту удовлетворения интересов обеих сторон составляется документ. В документе указываются соискатель, работодатель, должность и комиссионные (доход бюро). В базе должна фиксироваться не только сделка, но

и информация по открытым вакансиям. Кроме того для автоматического поиска вариантов необходимо вести справочник «Виды деятельности».

*7. Нотариальная контора*. Нотариальная контора готова предоставить клиенту определенный комплекс услуг. Услуги формализованы, т. е. составлен их список с описанием каждой услуги. При обращении клиента его стандартные данные (название, вид деятельности, адрес, телефон) фиксируются в базе данных. По каждому факту оказания услуги клиенту составляется документ, в котором указываются дата, услуга, сумма сделки, комиссионные (доход конторы), описание сделки. В рамках одной сделки клиенту может быть оказано несколько услуг. Стоимость каждой услуги фиксирована. Кроме того компания предоставляет в рамках одной сделки различные виды скидок. Скидки могут суммироваться.

*8. Фирма по продаже запчастей*. Фирма продает запасные части для автомобилей. Фирма имеет определенный набор поставщиков, по которым известны название, адрес и телефон. У поставщиков приобретаются детали. Каждая деталь характеризуется названием, артикулом и ценой. Некоторые из поставщиков могут поставлять одинаковые детали (один артикул). Каждый факт покупки запчастей у поставщика фиксируется в базе данных, причем обязательными для запоминания являются дата покупки и количество приобретенных деталей. Цена детали может
меняться от поставки к поставке. Поставщики заранее ставят фирму в известность о дате изменения цены и ее новом значении. Нужно хранить не только текущее значение цены, но и всю историю изменения цен.

*9. Курсы по повышению квалификации*. В учебном заведении организованы курсы повышения квалификации. Группы слушателей формируются в зависимости от специальности и отделения. В каждую из них включено определенное количество слушателей. Проведение занятий обеспечивает штат преподавателей, для каждого из которых в базе данных зарегистрированы стандартные анкетные данные (фамилия, имя, отчество, телефон) и стаж работы. В результате распределения нагрузки получена информация о том, сколько часов занятий проводит каждый преподаватель с соответствующими группами. Хранятся также сведения о виде занятий (лекция, практика), дисциплине и оплате за 1 час. Размер почасовой оплаты зависит от предмета и типа занятия. Кроме того каждый преподаватель может вести не все предметы, а только некоторые.

*10. Определение факультативов для студентов*. Преподаватели кафедры в высшем учебном заведении обеспечивают проведение факультативных занятий по некоторым предметам. Имеются сведения о студентах, включающие стандартные анкетные данные (фамилия, имя, отчество, группа, адрес, телефон). По каждому факультативу существует определенное количество часов и вид проводимых занятий (лекции, практика, лабораторные работы). В результате работы со студентами появляется информация о том, кто из них записался на какие факультативы. Существует некоторый минимальный объем факультативных предметов, которые должен прослушать каждый студент. По окончании семестра в базу данных заносится информация об оценках, полученных студентами на экзаменах. Некоторые из факультативов могут длиться более одного семестра. В каждом семестре для предмета устанавливается объем лекций, практик и лабораторных работ в часах. В качестве итоговой оценки за предмет берется последняя оценка, полученная студентом.

*11. Распределение учебной нагрузки*. Необходимо распределять нагрузку между преподавателями кафедры. Имеются сведения о преподавателях, включающие наряду с анкетными данными сведения об их ученой степени, занимаемой должности и стаже работы. Преподаватели кафедры должны обеспечить проведение занятий по некоторым дисциплинам. По каждой из них существует определенное количество часов. В результате распределения нагрузки необходимо получить информацию следующего рода: «Такой-то преподаватель проводит занятия по такой-то дисциплине с такой-то группой». Все проводимые занятия делятся на лекционные и практические. По каждому виду занятий

устанавливается свое количество часов. Кроме того данные по нагрузке нужно хранить несколько лет.

*12.* *Распределение дополнительных обязанностей*. Кафедра вуза имеет штат сотрудников, каждый из которых получает определенный оклад. Каждый месяц возникает потребность в выполнении некоторой дополнительной работы, не входящей в круг основных обязанностей сотрудников. Для наведения порядка в этой сфере классифицированы все виды дополнительных работ и определена сумма оплаты по факту их выполнения. При возникновении дополнительной работы назначается ответственный, фиксируется дата начала. По факту окончания фиксируется дата и выплачивается дополнительная сумма к зарплате с учетом классификации. Необходимо учесть разделение сотрудников на преподавателей и учебно-вспомогательный персонал. Для первых нужно хранить сведения об ученой степени и ученом звании, для вторых – о должности. Некоторые работы являются трудоемкими и срочными, что требует привлечения к их выполнению нескольких сотрудников. Длительность работ различна. Нужно заранее планировать длительность работы и количество сотрудников, занятых для выполнения работы.

*13.* *Техническое обслуживание станков*. Компания занимается ремонтом станков и другого оборудования. Клиентами компании являются промышленные предприятия. Ремонтные работы организованы следующим образом: все станки классифицированы по типам, странам- производителям, годам выпуска и маркам. Все виды ремонта отличаются названием, продолжительностью в днях, стоимостью. Исходя из этих данных, по каждому факту ремонта фиксируется вид станка, дата начала и дата окончания ремонта. Анализ показал, что нужно не просто подразделять станки по типам, а иметь информацию о том, сколько раз ремонтировался тот или иной станок.

*14.* *Туристическая фирма*. Фирма продает путевки клиентам. У каждого клиента запрашиваются стандартные данные – фамилия, имя, отчество, адрес, телефон. После этого сотрудники компании выясняют у клиента, куда он хотел бы поехать отдыхать. Ему демонстрируются различные варианты, включающие страну проживания, особенности климата,отель. Обсуждается длительность пребывания и стоимость путевки. Если удалось найти приемлемый вариант, регистрируется факт продажи путевки (или путевок, если клиент покупает сразу несколько), фиксируется дата отправления. Иногда клиенту предоставляется скидка (скидки фиксированы и могут суммироваться). Фирма работает с несколькими отелями (название, категория, адрес) в нескольких странах. Путевки продаются на одну, две или четыре недели. Стоимость путевки зависит от длительности тура и отеля.

*15.* *Грузовые перевозки*. Компания осуществляет перевозки грузов по различным маршрутам. Необходимо отслеживать стоимость перевозок с учетом заработной платы водителей. Для каждого маршрута определено название, вычислено примерное расстояние и установлена некоторая оплата для водителя. Информация о водителях включает фамилию, имя, отчество и стаж. Для проведения расчетов хранится полная информация о перевозках (маршрут, водитель, даты отправки и прибытия). По факту некоторых перевозок водителям выплачивается премия. Фирма решила ввести гибкую систему оплаты. Оплата водителям должна зависеть не только от маршрута, но и от стажа водителя. Кроме того, нужно учесть, что перевозку могут осуществлять два водителя.

*16.* *Учет телефонных переговоров*. Телефонная компания предоставляет абонентам телефонные линии для междугородних переговоров. Абонентами компании являются юридические лица, имеющие телефонную точку, ИНН, расчетный счет в банке. Стоимость переговоров зависит от города, в который осуществляется звонок, и времени суток (день, ночь). Каждый звонок абонента автоматически фиксируется в базе данных. При этом запоминаются город, дата, длительность разговора и время суток. Компания решила ввести гибкую систему скидок. Так, стоимость минуты теперь уменьшается в зависимости от длительности разговора. Размер скидки для каждого города разный.

*17.* *Учет внутриофисных расходов*. Сотрудники частной фирмы могут

осуществлять мелкие покупки для нужд фирмы, предоставляя в бухгалтерию товарный чек. Бухгалтерия отслеживает внутриофисные расходы. Фирма состоит из отделов, каждый из которых имеет название. В каждом отделе работает определенное количество сотрудников. Сотрудники могут осуществлять покупки в соответствии с видами расходов. Каждый вид расходов имеет название, некоторое описание и предельную сумму средств, которые могут быть потрачены по данному виду расходов в месяц. При каждой покупке сотрудник оформляет документ, где указывает вид расхода, дату, сумму и отдел. Нужно хранить данные о расходах не только в целом по отделу, но и по отдельным сотрудникам. Нормативы по расходованию средств устанавливаются не в целом, а по каждому отделу за каждый месяц. Неиспользованные в текущем месяце деньги могут быть использованы позже.

*18.* *Библиотека*. Библиотека решила зарабатывать деньги, выдавая напрокат книги, имеющиеся в небольшом количестве экземпляров. У каждой книги, выдаваемой в прокат, есть название, автор, жанр. В зависимости от ценности книги для каждой из них определена залоговая стоимость (сумма, вносимая клиентом при взятии книги напрокат) и стоимость проката (сумма, которую клиент платит при возврате книги, получая назад залог). Читатели регистрируются в картотеке, которая содержит стандартные анкетные данные (фамилия, имя, отчество, адрес, телефон). Каждый читатель может обращаться в библиотеку несколько раз. Все обращения читателей фиксируются, при этом по каждому факту выдачи книги запоминаются дата выдачи и ожидаемая дата возврата. Стоимость проката книги должна зависеть не только от самой книги, но и от срока ее проката. Кроме того, необходимо добавить систему штрафов за вред, нанесенный книге, и систему скидок для некоторых категорий читателей.

*19.* *Прокат автомобилей*. Фирма, занимающаяся прокатом автомобилей, имеет автопарк, содержащий некоторое количество автомобилей различных марок, стоимостей и типов. Каждый автомобиль имеет свою стоимость проката. В пункт проката обращаются клиенты. Клиенты проходят обязательную регистрацию, в ходе которой о них собирается стандартная информация (фамилия, имя, отчество, адрес, телефон). Каждый клиент может обращаться в пункт проката несколько раз. Обращения клиентов фиксируются, при этом по каждой сделке запоминаются дата выдачи и ожидаемая дата возврата. Стоимость проката автомобиля должна зависеть не только от самого автомобиля, но и от срока его проката, а также от года выпуска. Также нужно ввести систему штрафов за возвращение автомобиля в ненадлежащем виде и систему скидок для постоянных клиентов.

*20.* *Выдача банком кредитов*. Коммерческий банк выдает кредиты юридическим лицам. В зависимости от условий получения кредита, процентной ставки и срока возврата все кредитные
операции делятся на несколько основных видов. Каждый из этих видов имеет свое название. Кредит может получить юридическое лицо (клиент), при регистрации предоставивший следующие сведения: название, вид собственности, адрес, телефон, контактное лицо. Каждый факт выдачи кредита регистрируется банком, при этом фиксируются сумма кредита, клиент и дата выдачи. Чтобы отслеживать динамику возврата кредитов принято решение учитывать в системе еще и дату фактического возврата денег. Нужно еще учесть, что кредит может гаситься частями, и за задержку возврата кредита начисляются штрафы.

*21.* *Инвестиционная компания*. Компания занимается вложением денежных средств в ценные бумаги, которые характеризуются рейтингом, доходностью за прошлый год, минимальной суммой сделки и некоторой дополнительной информацией. Клиентами компании являются предприятия, которые доверяют ей управлять их свободными денежными средства на определенный период. Необходимо выбрать вид ценных бумаг, которые позволят получить прибыль и компании и клиенту. При работе с клиентом существенной является информация о предприятии – название, вид собственности, адрес и телефон. Каждая инвестиция характеризуется информацией о клиенте, информацией о ценной бумаге, котировкой бумаги, датой ее покупки и датой ее продажи. Необходимо

хранить историю котировок каждой ценной бумаги. Кроме того, помимо вложений в ценные бумаги существует возможность вкладывать деньги в банковские депозиты.

*22. Занятость актеров театра*. Коммерческий директор театра организует привлечение актеров и заключение контрактов. Каждый год театр осуществляет постановку различных спектаклей. Каждый спектакль имеет определенный бюджет. Для участия в конкретных постановках в определенных ролях привлекаются актеры. С каждым из актеров заключается персональный контракт на определенную сумму. Каждый актер имеет стаж работы, некоторые из них удостоены различных званий. В рамках одного спектакля на одну и ту же роль привлекается несколько актеров. Контракт определяет базовую зарплату актера, а по итогам реально отыгранных спектаклей актеру назначается премия. В базе данных нужно хранить информацию за несколько лет.

*23. Платная поликлиника*. В поликлинике работают врачи различных специальностей, имеющие разную квалификацию. Каждый день в поликлинику обращаются пациенты. Все пациенты проходят обязательную регистрацию, при которой в базу данных заносятся стандартные анкетные данные (фамилия, имя, отчество, год рождения, адрес). При обращении в поликлинику пациент обследуется и проходит лечение у разных специалистов. Каждый пациент может обращаться в поликлинику несколько раз, нуждаясь в различной медицинской помощи. Все обращения пациентов фиксируются, при этом устанавливается диагноз, определяется стоимость лечения, запоминается дата обращения. Общая стоимость лечения зависит от стоимости консультаций и процедур, назначенных пациенту. Для определенных категорий граждан предусмотрены скидки.

*24. Анализ динамики показателей финансовой отчетности предприятий*. Информационно-аналитический центр крупного холдинга отслеживает динамику показателей предприятий холдинга. В структуру холдинга входят несколько предприятий. Каждое предприятие имеет стандартные характеристики (название, реквизиты, телефон, контактное лицо). Работа предприятия может быть оценена следующим образом: в начале каждого отчетного периода на основе финансовой отчетности вычисляется определенный набор показателей. Важность показателей характеризуется некоторыми числовыми константами. Значение каждого показателя измеряется в некоторой системе единиц. Некоторые показатели считаются в рублях, некоторые в долларах, некоторые в евро. Для удобства работы с показателями нужно хранить изменения курсов валют относительно друг друга.

*25. Учет телекомпанией стоимости прошедшей в эфире рекламы*. Работа коммерческой службы телевизионной компании построена следующим образом: заказчики просят поместить свою рекламу в определенной передаче в определенный день. Каждый рекламный ролик имеет определенную продолжительность. Для каждой организации-заказчика известны банковские реквизиты, телефон и контактное лицо для проведения переговоров. Передачи имеют определенный рейтинг. Стоимость минуты рекламы в каждой передаче определяется, исходя из рейтинга передачи и прочих соображений. Необходимо хранить информацию об агентах, заключивших договоры на рекламу. Зарплата рекламных агентов составляет некоторый процент от общей стоимости рекламы, прошедшей в эфире.*IT-компания*. Компания оказывает IT-услуги организациям и предприятиям. В компании работают сотрудники, о которых должна сохраняться стандартная информация и данные о квалификации (владение языками и системами программирования, знание СУБД, операционных систем). В компанию обращаются клиенты, о которых собираются стандартные данные (наименование и адрес организации, телефон, адрес электронной почты, фамилия, имя и отчество контактного лица для связи). Задание для клиента выполняет определенный сотрудник, при этом фиксируется дата выдачи задания и трудоемкость выполнения (в часах). При повторном обращении клиент переходит в категорию постоянных и получает скидку. С ростом компании возникла необходимость разделения ее на отделы. Увеличились масштабы проектов, и теперь задание клиента

поручается отделу. В рамках одного договора может выполняться несколько заданий разными отделами компании.

*26. Ювелирная мастерская*. Ювелирная мастерская осуществляет изготовление ювелирных изделий для частных лиц на заказ. Мастерская работает с определенными материалами (платина, золото, серебро, драгоценные камни). При обращении потенциального клиента выясняется, какое именно изделие ему необходимо. Все изготавливаемые изделия принадлежат к некоторому типу (серьги, кольца, броши, браслеты), выполняются из определенного материала, имеют некоторый вес и цену (включающую стоимость материалов и работы). Ювелирное изделие может состоять из нескольких материалов. Кроме того, постоянным клиентам мастерская предоставляет скидки.

*27. Парикмахерская*. Парикмахерская стрижет клиентов в соответствии с их пожеланиями и некоторым каталогом различных видов стрижки. Для каждой стрижки определены название, категория (мужская, женская, детская), стоимость работы. Для наведения порядка составляется база данных клиентов, где запоминаются их анкетные данные (фамилия, имя, отчество). Начиная с 5-ой стрижки, клиент переходит в категорию постоянных и получает скидку в 3 % при каждой последующей стрижке. После того, как закончена очередная работа, в БД фиксируются стрижка, клиент и дата производства работ. Кроме того, у парикмахерской появился филиал и необходима раздельная статистика по филиалам. Стоимость стрижки может меняться с течением времени. Нужно хранить не только последнюю цену, но и все данные по изменению цены стрижки.

*28. Химчистка*. Химчистка осуществляет прием у населения вещей для выведения пятен. Для наведения порядка составляется база данных клиентов, в которой запоминаются их анкетные данные (фамилия, имя, отчество, адрес, телефон). Начиная с 3-го обращения, клиент переходит в категорию постоянных клиентов и получает скидку в 3 % при чистке каждой последующей вещи. Все оказываемые услуги подразделяются на виды, имеющие название, тип и стоимость, зависящую от сложности работ. Работа с клиентом первоначально состоит в определении объема работ, вида услуги и, соответственно, ее стоимости. Если клиент согласен, он оставляет вещь (при этом фиксируется услуга, клиент и дата приема) и забирает ее после обработки (при этом фиксируется дата возврата). Химчистка заключает с клиентом договор. Клиент может одновременно сдавать в чистку несколько вещей. У химчистки появились филиалы, и необходима раздельная статистика по филиалам. Введены надбавки за срочность и сложность.

*29. Сдача в аренду торговых площадей*. Торговый центр сдает в аренду коммерсантам свои торговые площади. В результате планирования определено некоторое количество торговых точек в пределах здания, которые могут сдаваться в аренду. Для каждой из торговых точек важными данными являются этаж, площадь, наличие кондиционера и стоимость аренды в день. С потенциальных клиентов собираются стандартные данные (название, адрес, телефон, реквизиты, контактное лицо). При появлении потенциального клиента ему показывают имеющиеся свободные площади. При достижении соглашения оформляется договор и в базе данных фиксируется торговая точка, клиент, период (срок) аренды. Некоторые клиенты в рамках одного договора арендуют сразу несколько торговых точек, причем для каждой точки возможен свой срок аренды. Дата заключения договора может не совпадать с датой начала аренды. Необходимо собирать информацию об ежемесячных платежах, поступающих от арендаторов.

**Лабораторная работа № 17 Использование диаграмм баз данных**

**Цель работы**: научиться создавать диаграммы и триггеры

Перейдём теперь к созданию диаграмм. В БД «Microsoft SQL Server 2008» все диаграммы находятся в папке «Диаграммы базы данных» обозревателя объектов.

Создадим диаграмму, обеспечивающую целостность данных нашей БД «Stuudents».

Для создания новой диаграммы в БД «Students» щёлкните ПКМ по папке «Диаграмма БД» и в появившемся меню выберем пункт «Создать диаграмму БД». Сначала появится окно с вопросом о добавлении нового объекта «Диаграмма». В этом окне нужно нажать кнопку «Да». Затем появится окно «Добавление таблицы» предназначенное для добавления таблиц в новую диаграмму.

В окне добавления таблиц выделите все таблицы нашей БД и нажмите кнопку «Добавить» Закройте окно «Добавление таблицы» нажатием на кнопку «Закрыть».

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 «Таблицы и столбцы» (Рис.6.1).

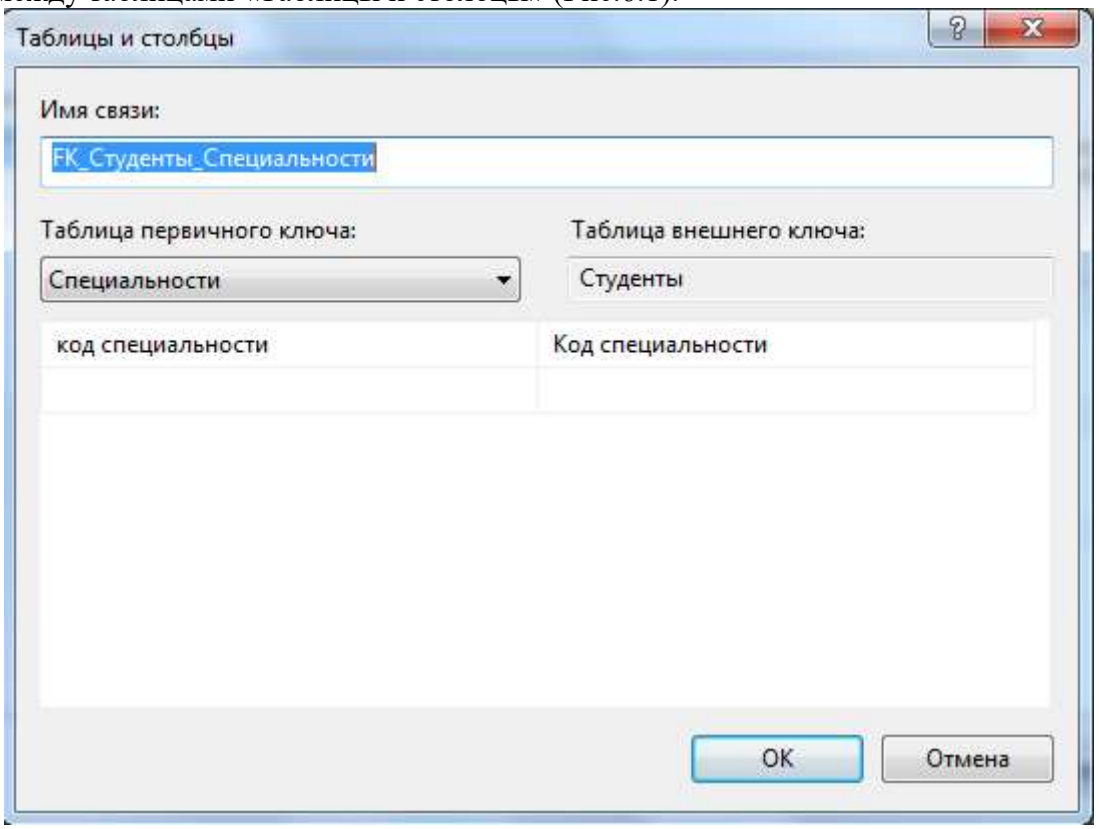



Рис.6.1

В окне создания связи нажмите кнопку «Ok». Появится окно настройки свойств связи «Связь по внешнему ключу» (Рис.6.2).

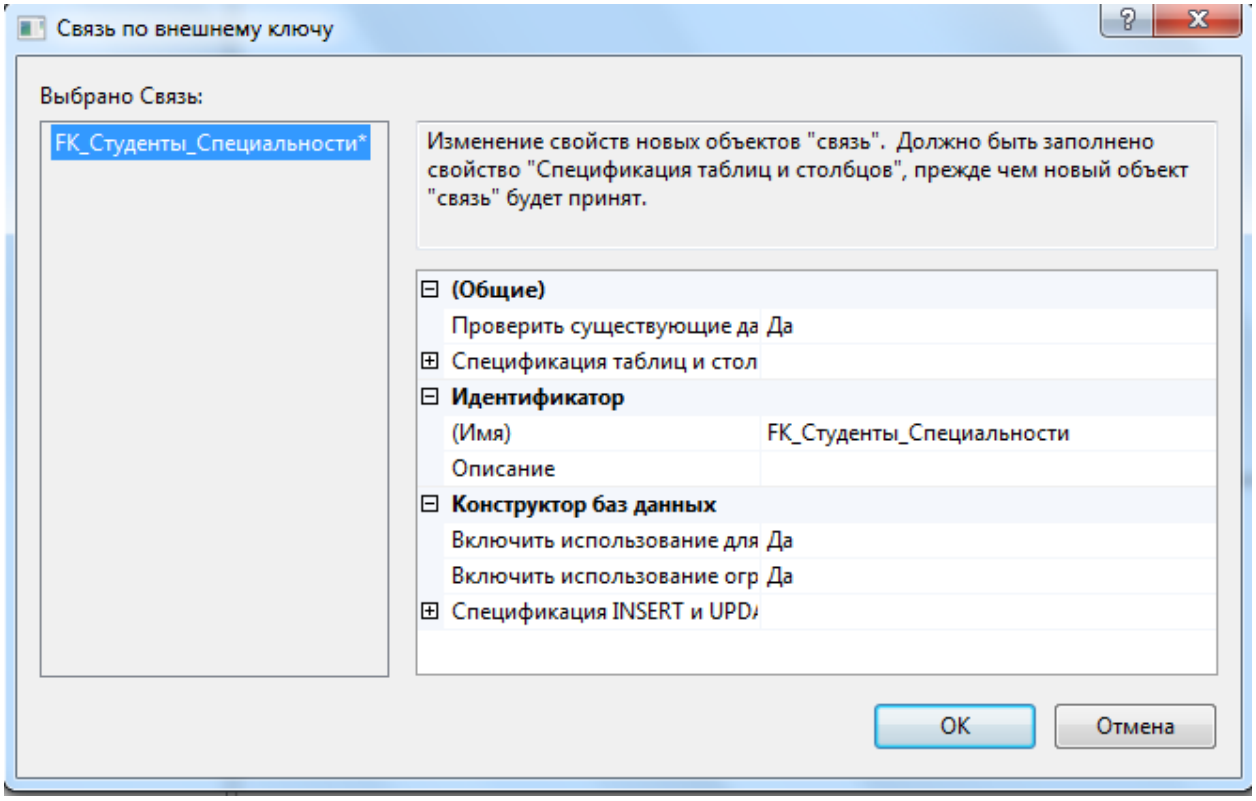



Рис.6.2

Оставьте свойства связи без изменений и в окне свойств связи нажмите кнопку «Ok».В диаграмме между таблицами «Студенты» и «Специальности» появится связь в виде ломанной линии (Рис.6.3).

Аналогичным образом создайте связь таблицы «Студенты» с таблицей «Оценки», перетащив поле «Код студента» из таблицы «Студенты» на одноимённое 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.6.3).

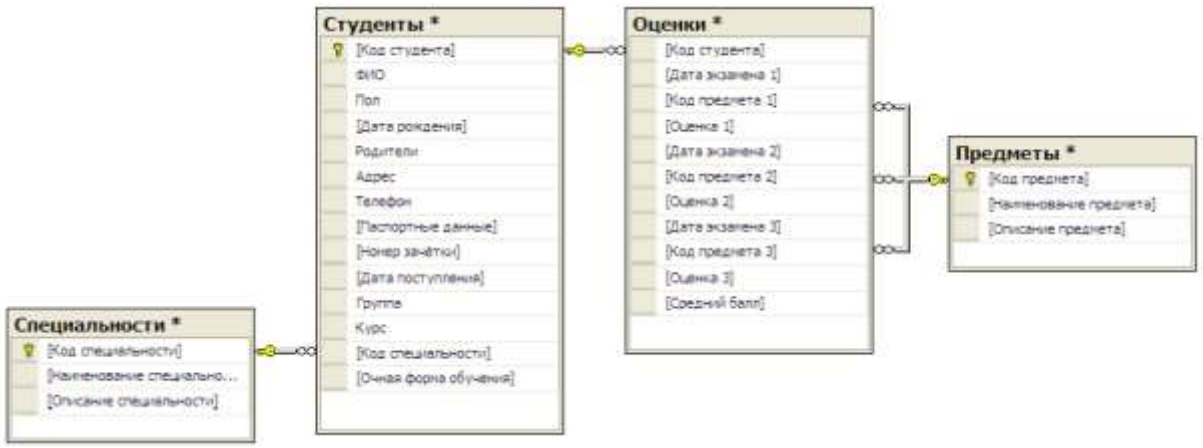



Рис.6.3

Закройте окно с диаграммой, щёлкнув мышью по кнопке закрытия, расположенной в верхнем правом углу окна с диаграммой. Появится окно с вопросом о сохранении новой диаграммы, где необходимо нажать кнопку «Да».

В окне определения имени, задайте имя диаграммы как «Диаграмма БД Студенты» и нажмите кнопку «Ok».

Появится окно «Сохранить» с запросом сохранения таблиц, входящих в диаграмму. В

данном окне необходимо нажать кнопку «Да».

Перейдём к созданию триггеров. Создадим триггеры для таблицы «Студенты».

Триггеры создаются отдельно для каждой таблицы и располагаются в обозревателе объектов в папке «Триггеры ». В нашем случае, папка «Триггеры» входит в состав таблицы «Студенты».

Для начала создадим триггер, выводящий сообщение «Запись добавлена» при добавлении записи в таблицу «Студенты». Создадим новый триггер, щёлкнув ПКМ по папке «Триггеры» в таблице «Студенты» и в появившемся меню выбрав пункт «Создать триггер». Появится следующее окно с новым триггером (Рис.6.4).

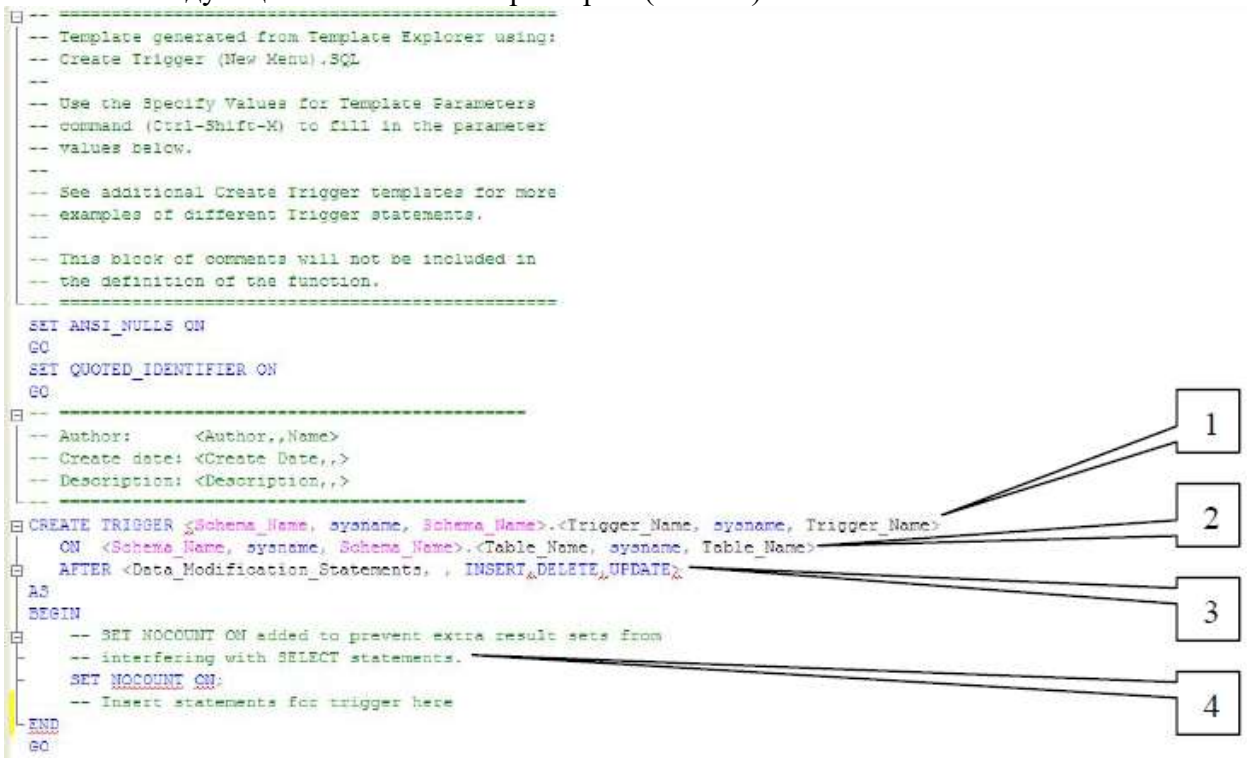



Рис.6.4.

Рассмотрим структуру триггеров:

1) Область определения имени функции (Trigger_Name);

2) Область, показывающая для какой таблицы создаётся 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 (ALTER – после выполнения операции, INSTEAD OF – вместо выполнения операции);

4) Тело триггера, содержит команды языка программирования запросов TSQL.

В окне нового триггера наберите код как показано на рисунке 6.5.

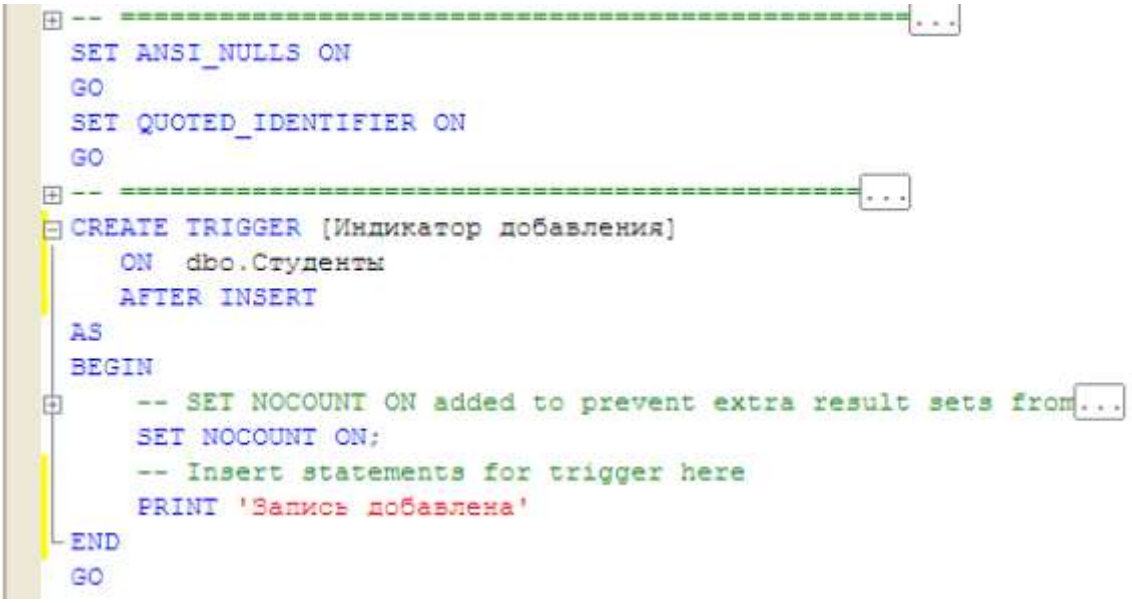



Рис.6.5.

Из рисунка 6.5. 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 Выполнить на панели инструментов. В нижней части окна с кодом появиться сообщение «Выполнение команд успешно завершено».

Проверим, как работает новый триггер. Создайте новый пустой запрос и в нём наберите следующую команду для добавления новой записи в таблицу «Студенты» (Рис.6.6.):

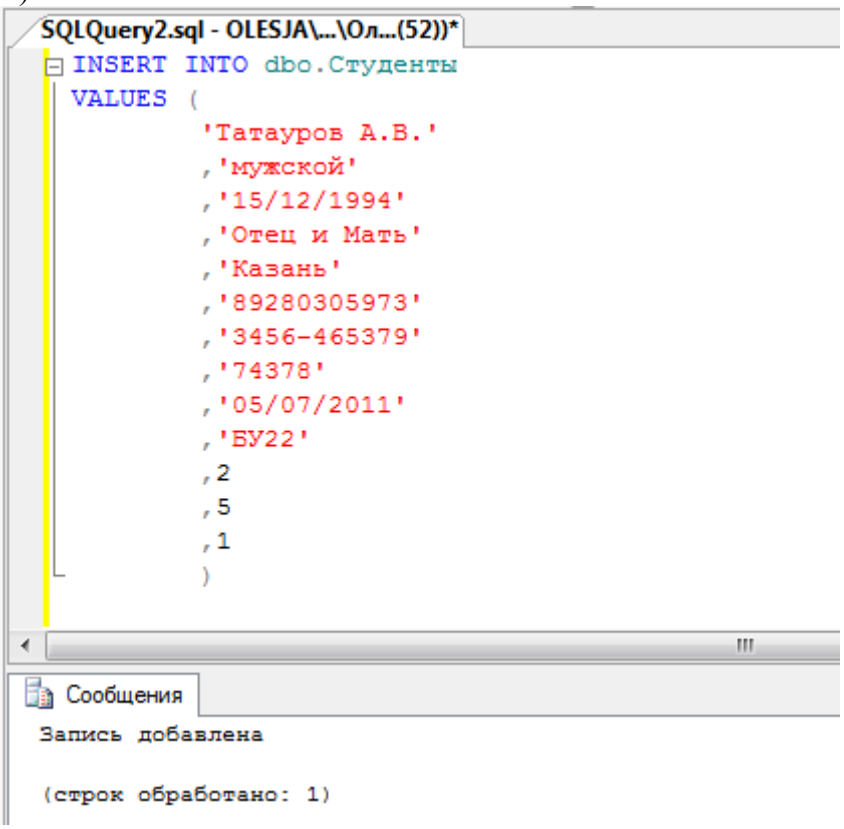



Рис.6.6

Выполните набранную команду, нажав кнопку Выполнить на панели инструментов.

В таблицу будет добавлена новая запись, и триггер выведет сообщение «Запись

добавлена» (Рис.6.6).

Теперь создадим триггер отображающий сообщение «Запись изменена». Создайте новый триггер, как в предыдущем случае. В окне нового триггера наберите следующий код (Рис.6.7):

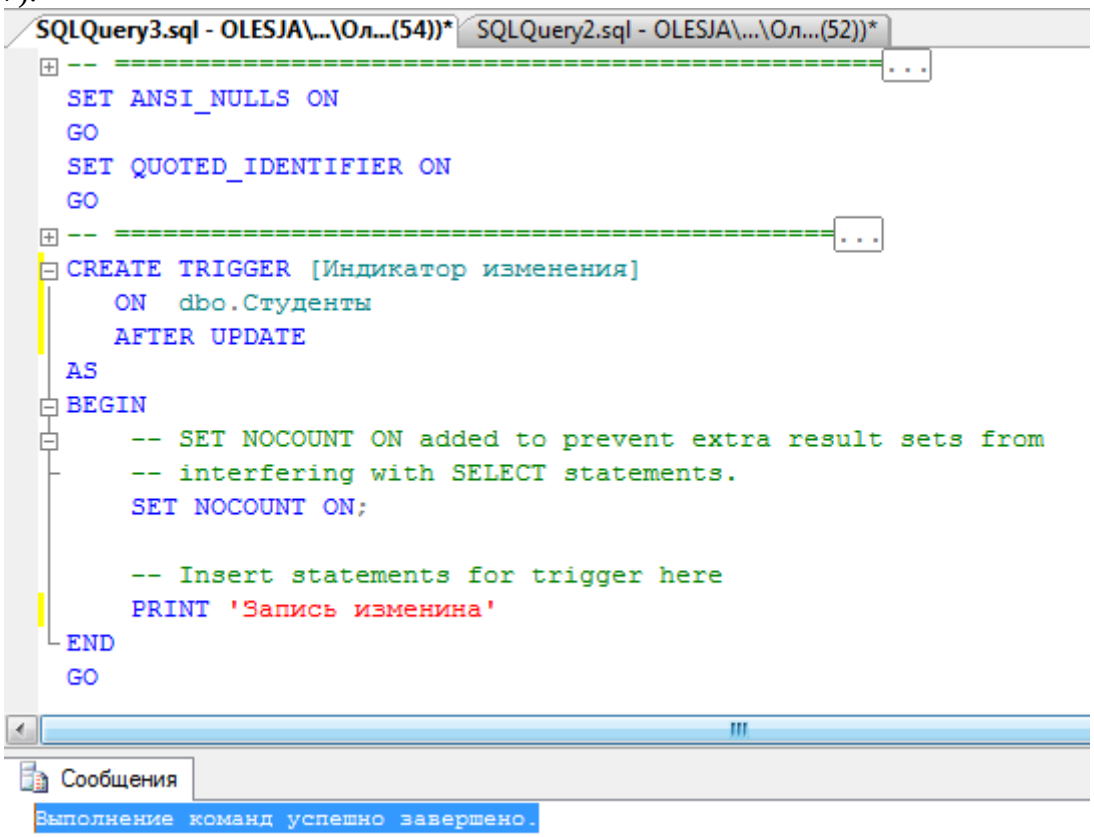



Рис.6.7.

Из рисунка 6.7. видно, что новый триггер «Индикатор добавления» 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 появиться сообщение «Выполнение команд успешно завершено».

Проверим работоспособность созданного триггера. Создайте новый запрос и в нём наберите команду, представленную на рисунке 6.8

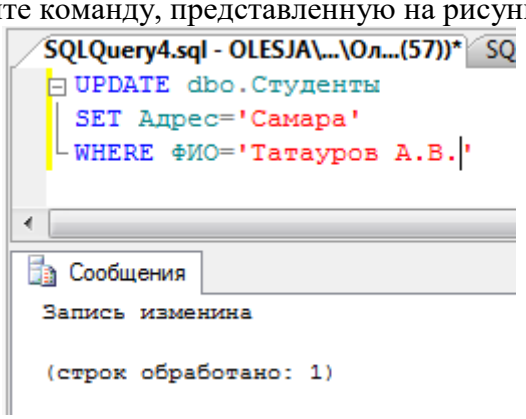



Рис.6.8

Выполните набранную команду, нажав кнопку на панели инструментов.

В таблицу будет добавлена новая запись, и триггер выведет сообщение «Запись добавлена» (Рис.6.8).

Для полноты картины создадим триггер, выводящий сообщение при удалении записи

из таблицы «Студенты». Создайте новый триггер и в нём наберите код, показанный на рисунке 6.9.

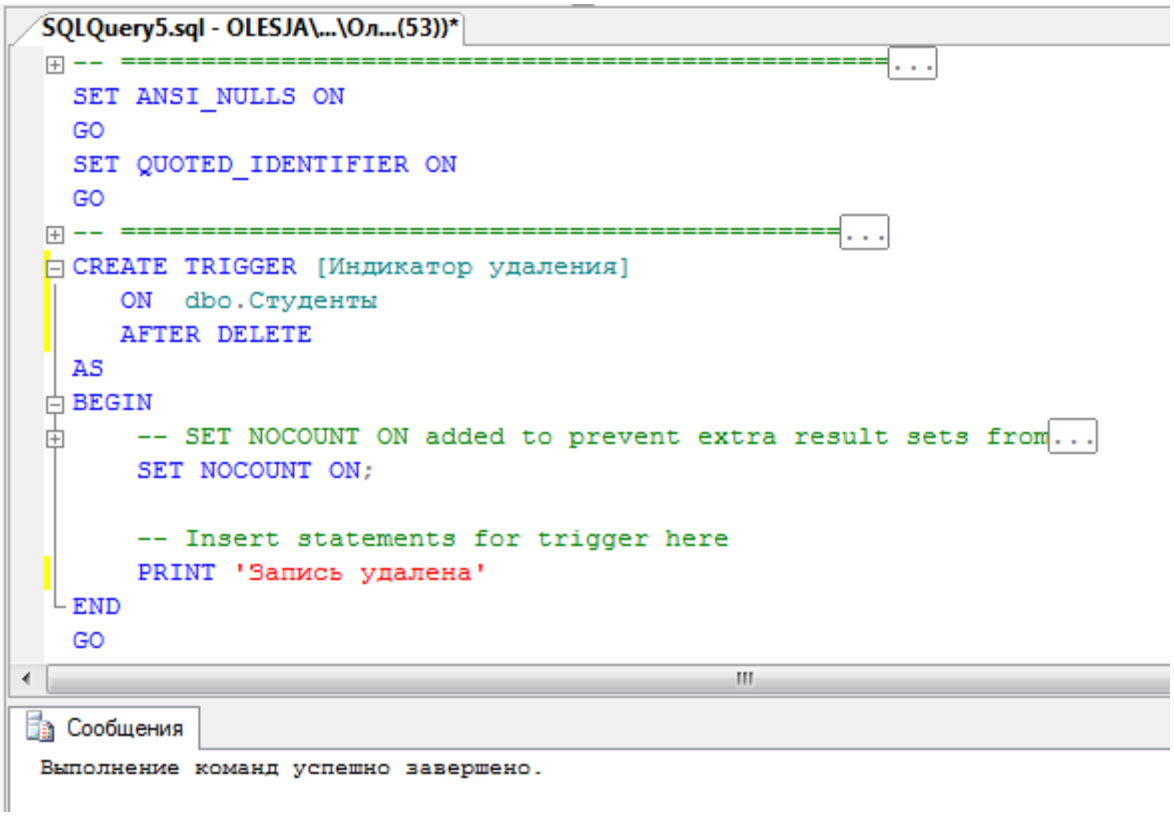



Рис 6.9.

Создаваемый триггер «Индикатор удаления» выполняется после удаления записи (ALTER DELETE) из таблицы студенты (ON dbo.Студенты). После удаления записи триггер выводит сообщение «Запись удалена» (PRINT ‘Запись удалена’).

Проверим работу триггера «Индикатор удаления» удалив созданную ранее запись из таблицы «Студенты». Для этого создайте новый запрос и в нём наберите следующую команду (Рис.6.10):

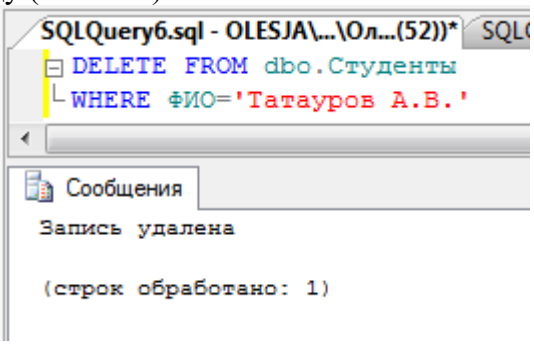



Рис.6.10

Выполните вышеприведённую команду. После удаления записи триггер «Индикатор удаления» отобразит сообщение «Запись удалена» (Рис.6.10).

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 нём наберите следующий код (Рис.6.11):

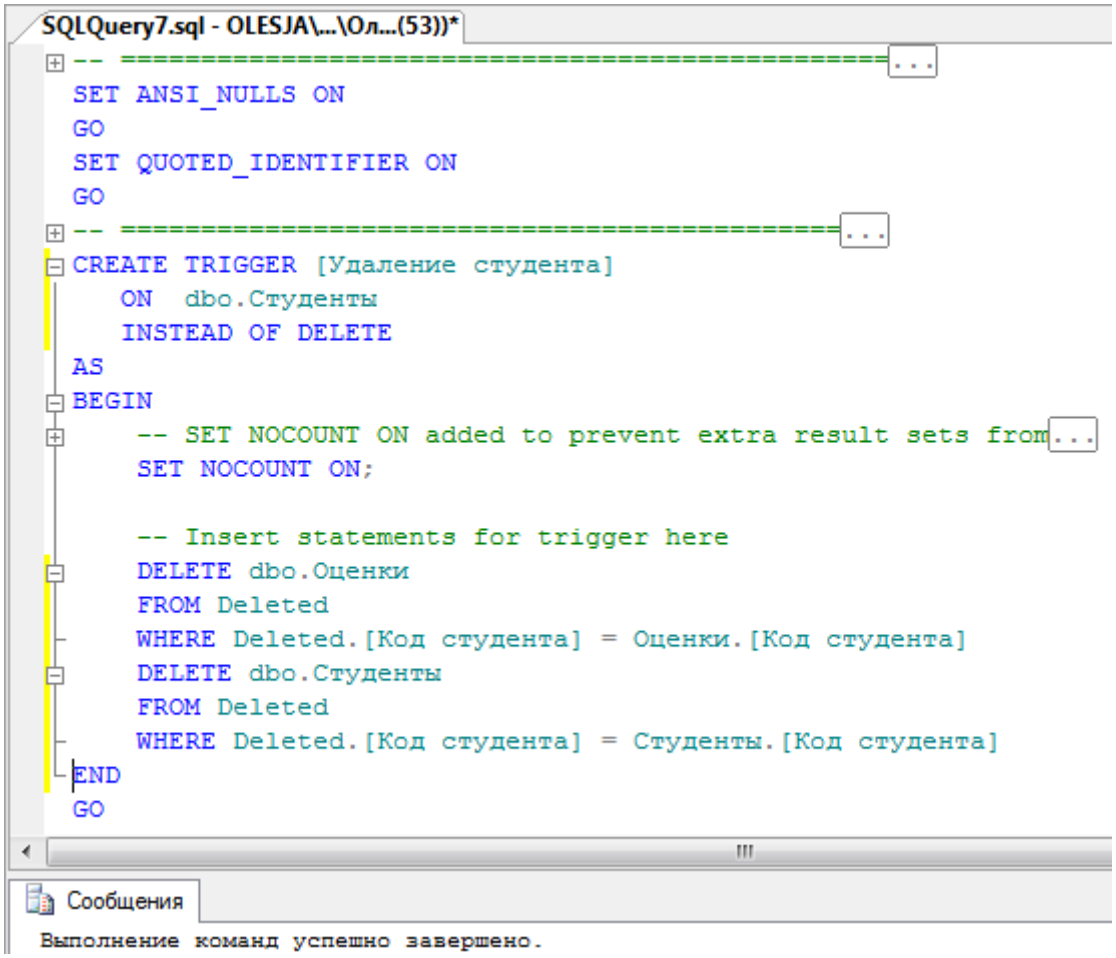



Рис.6.11.

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

**Замечание:** При срабатывании триггера вместо удаления записи создаётся 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]
```

Проверим, как работает триггер «Удаление студента». Для этого создайте новый запрос и в нём наберите следующий код (Рис6.12):

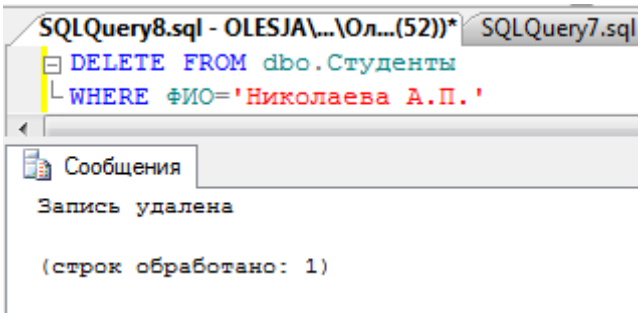



Рис.6.12

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.6.13):

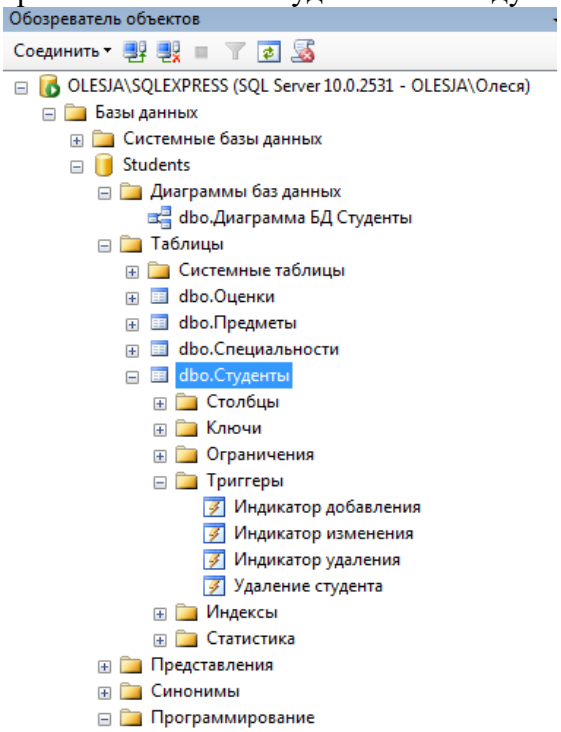



Рис. 6.13.

## Лабораторная работа № 18 Экспорт и импорт данных базы в документы пользователя (4 часа)

**Цель работы: Цель работы:** изучить технологию импорта данных пользователя в базу данных

Импорт - копирует данные в БД СУБД из внешних файлов, которые были созданы ути-литой экспорта .

**Import** - дополнительная утилита в основном применяется для резервного копирования и миграции БД (между серверами либо из более старой версии в более новую). Ниже приведены дру-гие возможности утилиты Import :

1. хранение данных в файлах ОС для архивирования;
2. выборочное резервное копирование частей БД;

3. перемещение данных из одной пользовательской схемы в другую;

4. перемещение данных с одной аппаратной платформы или ОС в другую.

5. экономия пространства и повышение производительности за счет уменьшения фрагмен-тации.

FILE expdat.dmp Имя файла, из которого будут импортироваться данные. По умолчанию именем файла, из которого осуществляется импорт, будет expdat.dmp.

FROMUSER Если указан этот параметр, то импортируются только те объекты, владельцем которых является пользователь с иден-тификационным кодом FROMUSER

FULL N Если FULL=Y, то импортироваться будет вся БД.

GRANTS Y Указывает, будут ли заданы все полномочия для экспортированных объектов.

HELP N Если задано HELP=Y, то другие параметры не требуются. На экране будет выведена справочная информация.

IGNORE NO Если задано IGNORE=Y, то ошибки при создании объектов игнорируются и строки вставляются в таблицу. Будьте внимательны, поскольку, если для таблицы не определено ограничение уникальности значений, то могут появиться дублирующие записи. Отметим, что о других ошибках, не связанных с созданием объектов (например, проблемах с ОС), пользователь будет информирован в обычном режиме.

INDEXES Y Указывает, экспортируются ли определяемые пользователем ин-дексы. Системные индексы, созданные посредством опре-деления ограничений (первичный ключ, уникальный ключ) импортируются независимо от значения параметра ISDEXES.

LOG Имя файла, в который будет записан журнал импорта. Если не указано иное, Oracle даст файлу расширение LOG.

TABLES Указывает список таблиц (с запятой в качестве разделителя), кото-рые должны быть импортированы. Этот параметр используется совместно с параметром FROMUSER. В не-UNIX среде, как, например, Windows, необходимо заключать список таблиц в круглые скобки.

TOUSER Параметр TOUSER указывает имя пользователя, который будет владельцем импортируемых объектов. Данный параметр необходимо использовать совместно с параметром FROMUSER.

USERID Указывает имя и пароль пользователя, который осуществляет про-цесс импорта. Формат параметра — «имя пользователя/пароль@сервер».

**ХОД РАБОТЫ**

1. Подключитесь к учебной БД под учетной записью student. Создайте двух новых поль-зователей (USER1) и (USER2). Создайте новую роль. Присвойте роли права подключаться, со-здавать таблицы, создавать последовательности, создавать триггеры и роль DBA.

2. Создайте таблицу-справочник стран: ID (первичный ключ), название страны (символьное, уникальное). Добавьте в таблицу две-три записи.

3. Создайте таблицу-справочник городов: ID (первичный ключ), страна (внешний ключ к таблице стран), название города (символьный). Создайте последовательность. Создайте триг-гер к таблице, который перед вставкой записи заполняет первичный ключ. Добавьте в таблицу пять записей.

4. Импортируйте таблицу из файла в схему пользователя USER1. Подключитесь к учеб-ной БД под учетной записью USER1. Напишите запрос, который бы возвращал все записи таб-лицы стран. Добавьте три записи в таблицу. Удалите таблицу стран.

5. Подключитесь под учетной записью student. Создайте хранимую процедуру, которая бы возвращала список городов с привязанными странами, т.е. город и страна. Для этого приме-ните курсорный цикл. Вызовите хранимую процедуру.

6. Импортируйте схему пользователя student из файла в схему пользователя USER2. Под-ключитесь к учебной БД под учетной записью USER2. Напишите запрос, который бы

возвра-щал все записи таблицы городов. Добавьте три записи в таблицу. Удостоверьтесь, что триггер заполнил первичный ключ.

7. Создайте текстовый файл или новый документ в EXCEL. Заполните десять строк для переноса в таблицу городов с помощью SQL*Loader. Создайте в текстовом редакторе управля-ющий файл и импортируйте в таблицу городов данные. Напишите запрос, который бы возвра-щал все записи таблицы городов. Убедитесь, что после импорта появились новые записи.

## Лабораторная работа № 19 Выполнение настроек для автоматизации обслуживания базы данных

**Цель работы:** Освоение некоторых возможностей автоматизации управления базой данных

**ХОД РАБОТЫ**

При работе с базой данных часто приходится многократно выполнять одинаковые порой рутинные операции. Вполне естественно было бы автоматизировать их выполнение. Для этого СУБД располагает достаточными средствами, позволяющими во многом автоматизировать и упорядочить работу с базой данных. К числу таких средств относятся:

– пользовательские меню и инструментальные панели;
– кнопочные формы управления базой данных;
– средства настройки параметров запуска базы данных;
– макросы и модули.

**Задание 1.** Создайте пользовательское меню для управления базой данных, содержащее категории Формы и Отчеты с пунктами (командами) для открытия ранее составленных форм иотчетов.

1. Для создания новой строки меню откройте окно Настройка. Для этого выполните команду ВИД/Панели инструментов/Настройка или, щелкнув правой клавишей по любой па- нели инструментов, выберите в контекстном меню пункт Настройка.
2. В окне Настройка на вкладке Панели инструментов щелкните по кнопке Создать.
3. В окне Создание панели инструментов введите имя панели инструментов: Управле-ние базой данных. Нажмите кнопку Ok. В окне БД появится небольшая миниатюра панели. Пе-ретащите созданную панель инструментов в верхнюю часть окна установив над строкой меню.
4. В окне Настройка нажмите кнопку Свойства и определите тип созданной панели

- Строка меню. Закройте окно установки свойств.

5. Добавьте в строку созданного меню категорию Формы. Для этого в окне Настройка откройте вкладку Команды и в списке категорий щелкните по категории Новое меню. Перетащите команду Новое меню из списка команд в правом подокне на строку меню Управление базой данных. Не закрывая окно Настройка, щелкните правой клавишей в строке меню по категории Новое меню и в контекстном меню замените имя категориина Формы.
6. Добавьте в меню категорию Отчеты аналогично пункту 5.
7. Аналогично добавьте в меню Формы новое подменю, назвав его Простые.
8. В окне Настройка на вкладке Команды выделите категорию Все формы. Перета-щите строку с названием одной из созданных ранее форм - Студент простая в область команд (пунктов) категории Формы строки меню Управление базой данных. Включив контекстное меню новой команды, установите стиль отображения - Только текст.
9. Аналогично добавьте в область команд категории Формы/Простые пункты с назва- нием форм – Группа и Простая форма по запросу.

10. Добавьте в категорию Отчеты меню Управление базой данных пункты с названи- ями отчетов. Закройте окно Настройка. Проверьте работу меню.

11. Выполните команду СЕРВИС/Параметры запуска и установите в окне Параметры запуска следующие параметры запуска при открытии базы данных:

– введите в качестве заголовка приложения название Академия;

– выберите в качестве строки меню строку Управление базой данных.

– отмените вывод на экран окна базы данных, строки состояния, полного набора меню встро-енных панелей инструментов.

Закройте окно Параметры запуска. Закройте базу данных, затем повторно откройте. От- кроется окно базы данных, содержащее только одну строку пользовательского меню Управле-ние базой данных с категориями Формы и Отчеты. Убедитесь в правильной работе команд меню.

12. Восстановите для базы данных Академия отображение окна базы данных, полного набора меню, встроенных панелей инструментов. Для этого перезагрузите базу данных и при повторном открытии держите нажатой клавишу SHIFT. Выполните команду СЕРВИС/Пара- метры запуска и восстановите исходное состояние флажков.

**Задание 2.** Создайте макрос для автоматического формирования экзаменационных ведомостей, рассмотренных ранее. Отдельные таблицы должны быть созданы для каждой группыстудентов, имеющейся в базе данных, и для выбранной дисциплины.

Целью разработки макроса является исключение необходимости каждый раз перед созданием новой таблицы (экзаменационной ведомости) вручную переименовывать ранее создан- ную таблицу, чтобы предотвратить ее удаление. Макрос должен сам создавать таблицы с име- нами, соответствующими номерам групп и кодам дисциплин, по схеме: Ведомость NNN-K,

где NNN - номер группы, введенный в диалоговом окне, K - код дисциплины.

Для того, чтобы передать параметры создаваемых таблиц (номер группы и код дисциплины) можно, например, использовать форму, созданную на основе временной таблицы и вы-бирать эти параметры из первой строки этой формы.

Кроме того, для того чтобы избежать лишних остановок при выполнении макроса, поручите макросу не выводить вспомогательные служебные сообщения и сообщения-предупре- ждения. При выполнении макроса пользователь должен будет вводить только номер группы и код дисциплины.

При конструировании макроса можно использовать ранее созданный запрос с именем Запрос на создание экзаменационной ведомости.

Поскольку условия выполнения макрокоманд могут определяться только значениями полей или элементов управления форм и отчетов предварительно следует создать вспомогательную табличную форму на основании таблицы Ведомость 1.

Автоматически создайте новую табличную форму на основании таблицы Ведомость Для этого в окне База данных выберите объект Формы и щелкните по кнопке Создать. В диалоговом окне Новая форма выберите способ создания - Автоформа: табличная, а в качестве источника данных - таблицу Ведомость Щелкните по кнопке Ok. После появления на экране формы закройте ее и сохраните под именем Форма для макроса.

Для того, чтобы форма не зависела от таблицы Ведомость 1 замените имя источника за- писей в окне свойств формы на Ведомость 00. Для этого откройте форму в режиме конструк- тора и щелкните по кнопке Свойства, расположенной инструментальной панели Конструктор форм. Отредактируйте значение свойства Источник записей на вкладке Данные.

В окне База данных выберите объект Макросы и щелкните по кнопке Создать. Появится окно конструктора макросов. Добавьте в окне еще один столбец - Условия. Для этого выпол- ните команду ВИД/Условия или щелкните по кнопке инструментальной панели

с соответству-ющим названием.

Щелкните внутри ячейки первой строки и столбца Макрокоманда. Появится поле со списком макрокоманд. Выберите макрокоманду: УстановитьСообщения. В нижней части окна появится аргумент этой команды: Нет. Оставьте его без изменения. Введите в графу Примеча- ние краткий комментарий: Отключение системных и предупреждающих сообщений.

Перейдите на следующую строку и выберите для нее макрокоманду Открыть Запрос. Определите аргументы макрокоманды в нижней части окна. Раскройте список в поле Имя запроса и выберите в нем имя Запрос на создание экзаменационной ведомости. Сохраните значение аргументов Режим - таблица и Режим данных - изменение. Введите коммен- тарий к этой макрокоманде: на создание экзаменационной ведомости. При выполнении данноймакрокоманды будет создана таблица Ведомость 1.

После того, как запрос на создание ведомости отработал, его можно закрыть, поэтому в третьей строке выберите макрокоманду Закрыть. Определите аргументы макрокоманды: Тип объекта - Запрос, Имя объекта - Запрос на создание экзаменационной ведомости, Сохранение-Подсказка. Введите комментарий: Закрытие запроса.

Перед открытием созданной ранее Формы для макроса необходимо произвести копиро- вание таблицы Ведомость1 в другую таблицу, которая и будет использована для открытия формы - Ведомость 00. Для этого выберите макрокоманду КопироватьОбъект. Определите ар- гументы макрокоманды: Новое имя - Ведомость 00, Тип объекта - таблица, Имя объекта - Ве- домость 1. Введите комментарий: копирование таблицы Ведомость 1 в Ведомость 00.

Примечание. При определении для макрокоманды в качестве аргументов имен объектовсовершенно не обязательно, чтобы соответствующие объекты существовали в базе данных в момент конструирования макроса. Важно, чтобы они были в базе данных к моменту выполне- ния этой макрокоманды.

В следующей строке выберите макрокоманду ОткрытьФорму. Определите аргументы макрокоманды: Имя формы - Форма для макроса, Режим - Форма, Режим окна - Обычное. Вве-дите комментарий: Открыть форму на основе таблицы Ведомость 00.

Следующая макрокоманда должна осуществить переход на первую строку таблицы-формы для проверки значений полей N группы и Код дисциплины, введенных при выполнениизапроса в диалоговые окна. Поэтому в очередной строке выберите макрокоманду НаЗапись. Выберите из списка для аргумента Запись значение Первая. Введите комментарий: Перейти на1-ую запись табличной формы.

Выберите для следующей строки макрокоманду Переименовать для переименования ве- домости для группы 851 и дисциплины с кодом 1. Определите аргументы макрокоманды: Новоеимя - Ведомость 851_1, Тип объекта - Таблица, Старое имя - Ведомость 1. В графу Усло- вие введите с помощью построителя для этой строки выражение :

[Формы]![Форма для макроса]![N группы]=851 And [Формы]![Форма для макроса]![Коддисциплины]=1

Введите комментарий: Переименовать Ведомость 1 в Ведомость 851_1

Введите следующую макрокоманду для переименования ведомости для группы 851 и дисциплины с кодом 2. Определите аргументы макрокоманды: Новое имя - Ведомость 851_2, Тип объекта - Таблица, Старое имя - Ведомость 1. Введите Условие:

[Формы]![Форма для макроса]![N группы]=851 And [Формы]![Форма для макроса]![Коддисциплины]=2

Примечание. Для ввода условий и комментариев к макрокомандам целесообразно ис-пользовать приемы копирования в буфер, а затем вставки с последующей корректировкой в окне области ввода, открываемом при нажатии сочетания клавиш SHIFT и F2.

Введите следующие строки макроса по аналогии с двумя предыдущими для переимено- вания ведомости для группы 851 и дисциплины с кодом 3, а также для других

групп и дисци- плин. При 3-х дисциплинах должно быть 3 макрокоманды на одну группу студентов.

Введите последнюю макрокоманду Закрыть с аргументами: Тип объекта - Форма, Имя объекта - Форма для макроса, Сохранение - Подсказка. Окончательный вид макроса представ-лен на рис. 7.1

Сохраните макрос, щелкнув по кнопке Сохранить инструментальной панели, под именем Макрос для создания ведомостей. Запустите макрос на выполнение в пошаговом режиме. Для этого, находясь в режиме конструктора щелкните по кнопке инструментальной панели По шагам, а затем по кнопке Запуск. Проследите по шагам правильность исполнения макрокоманд.В случае неверного выполнения или аварийного завершения найдите ошибку и исправьте мак-рос.

После отладки макроса отключите пошаговый режим запуска, повторно щелкнув по кнопке По шагам, закройте макрос. Произведите запуск макроса из окна базы данных. Для этого в окне База данных выберите объект Макросы, выделите Макрос для создания ведомостей и щелкните по кнопке Запуск на инструментальной панели окна. Запустите макрос для формирования экзаменационных ведомостей по разным группам и дисциплинам.

## Перечень учебных изданий, Интернет ресурсов, дополнительной литературы

### Основные источники

### Основная литература

1. Агальцов, В. П. Базы данных : в 2 книгах. Книга 2. Распределенные и удаленные базы данных : учебник / В.П. Агальцов. - Москва : ФОРУМ : ИНФРА-М, 2021. - 271 с. - (Высшее образование: Бакалавриат). - URL: https://znanium.com/catalog/product/1514118 (дата обращения: 11.09.2023). - . Режим доступа: ЭБС Znanium.com, для зарегистрир. пользователей. -ISBN 978-5-8199-0713-9. - Текст : электронный.

2. Казарин, О. В. Программно-аппаратные средства защиты информации. Защита программного обеспечения : учебник и практикум для среднего профессионального образования / О. В. Казарин, А. С. Забабурин. - Москва : Юрайт, 2023. - 312 с. - (Профессиональное образование). - URL: https://urait.ru/bcode/519364 (дата обращения: 04.04.2023). - Режим доступа: ЭБС Юрайт, для зарегистрир. пользователей. - ISBN 978-5-534-13221-2. - Текст : электронный.

3. Кумскова, И. А. Базы данных : учебник / Кумскова И. А. - Москва : КноРус, 2022. - 400 с. - URL: URL: https://book.ru/book/943244 (дата обращения: 04.04.2023). - Режим доступа: ЭБС Book.ru, для зарегистрир. пользователей. - ISBN 978-5-406-09667-3. - Текст : электронный.

4. Полищук, Ю. В. Базы данных и их безопасность : учебное пособие / Ю. В. Полищук, А. С. Боровский. - Москва : ИНФРА-М, 2023. - 210 с. - (Среднее профессиональное образование). - URL: https://znanium.com/catalog/product/1899319 (дата обращения: 04.04.2023). - Режим доступа: ЭБС Znanium.com, для зарегистрир. пользователей. - ISBN 978-5-16-016151-8. - Текст : электронный.

5. Шаньгин, В. Ф. Информационная безопасность компьютерных систем и сетей : учебное пособие / В. Ф. Шаньгин. - Москва : ФОРУМ : ИНФРА-М, 2023. - 416 с. - (Среднее профессиональное образование). - URL: https://znanium.com/catalog/product/1910870 (дата обращения: 30.03.2023). - Режим доступа: ЭБС Znanium.com, для зарегистрир. пользователей. - ISBN 978-5-8199-0754-2. - Текст : электронный.

6. Шустова, Л. И. Базы данных : учебник / Л. И. Шустова, О. В. Тараканов. - Москва : ИНФРА-М, 2021. - 304 с. + Доп. материалы - (Среднее профессиональное образование). - URL: https://znanium.com/catalog/product/1189322 (дата обращения: 04.04.2023). - Режим доступа: ЭБС Znanium.com, для зарегистрир. пользователей. - ISBN 978-5-16-014161-9. - Текст : электронный.

### Дополнительная литература

1. Мартишин, С. А. Базы данных. Практическое применение СУБД SQL- и NoSQL-типа для проектирования информационных систем : учебное пособие / С.А. Мартишин, В.Л. Симонов, М.В. Храпченко. - Москва : ФОРУМ : ИНФРА-М, 2023. - 368 с. - (Среднее профессиональное образование). - URL: https://znanium.com/catalog/product/1912454 (дата обращения: 16.09.2023). - Режим доступа: ЭБС Znanium.com, для зарегистрир. пользователей. - ISBN 978-5-8199-0785-6. - Текст : электронный.

2. Голицына, О. Л. Базы данных : учебное пособие / О. Л. Голицына, Н. В. Максимов, И. И. Попов. - 4-е изд., перераб. и доп. - Москва : ФОРУМ : ИНФРА-М, 2020. - 400 с. - (Среднее профессиональное образование). - URL: https://znanium.com/catalog/product/1091314 (дата обращения: 04.04.2023). - Режим доступа: ЭБС Znanium.com, для зарегистрир. пользователей. - ISBN 978-5-00091-601-8. - Текст : электронный.

3. Гордеев, С. И. Организация баз данных в 2 ч. Часть 1 : учебник для среднего профессионального образования / С. И. Гордеев, В. Н. Волошина. - 2-е изд., испр. и доп. - Москва : Юрайт, 2023. - 310 с. - (Профессиональное образование). - URL:

https://urait.ru/bcode/518510 (дата обращения: 04.04.2023). - Режим доступа: ЭБС Юрайт, для зарегистрир. пользователей. - ISBN 978-5-534-11626-7. - Текст : электронный.

4. Гордеев, С. И. Организация баз данных в 2 ч. Часть 2 : учебник для среднего профессионального образования / С. И. Гордеев, В. Н. Волошина. - 2-е изд., испр. и доп. - Москва : Юрайт, 2023. - 513 с. - (Профессиональное образование). - URL: https://urait.ru/bcode/518511 (дата обращения: 04.04.2023). - Режим доступа: ЭБС Юрайт, для зарегистрир. пользователей. - ISBN 978-5-534-11625-0. - Текст : электронный.

5. Казарин, О. В. Основы информационной безопасности: надежность и безопасность программного обеспечения : учебное пособие для среднего профессионального образования / О. В. Казарин, И. Б. Шубинский. - Москва : Юрайт, 2023. - 342 с. - (Профессиональное образование). - URL: https://urait.ru/bcode/518005 (дата обращения: 04.04.2023). - Режим доступа: ЭБС Юрайт, для зарегистрир. пользователей. - ISBN 978-5-534-10671-8. - Текст : электронный.

6. Стасышин, В. М. Базы данных: технологии доступа : учебное пособие для среднего профессионального образования / В. М. Стасышин, Т. Л. Стасышина. - 2-е изд., испр. и доп. - Москва : Юрайт, 2023. - 164 с. - (Профессиональное образование). - URL: https://urait.ru/bcode/516927 (дата обращения: 04.04.2023). - Режим доступа: ЭБС Юрайт, для зарегистрир. пользователей. - ISBN 978-5-534-09888-4. - Текст : электронный