

Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Саратовский государственный технический университет  
имени Гагарина Ю.А.»

Филиал федерального государственного бюджетного образовательного  
учреждения высшего образования  
«Саратовский государственный технический университет  
имени Гагарина Ю.А.» в г. Петровске



УТВЕРЖДАЮ

Директор филиала СГТУ  
имени Гагарина Ю.А. в г.Петровске  
Е.А.Бесшапошникова  
«30» июня 2025 г.

## МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ

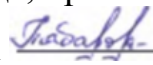
по дисциплине

МДК.11.01 «Технология разработки и защиты баз данных»

направление подготовки

09.02.07 «Информационные системы и программирование»

Методические указания рассмотрены  
на заседании предметной (цикловой) комиссии  
общепрофессиональных дисциплин и  
профессиональных модулей  
«16» июня 2025 года, протокол №13

Председатель ПЦК  /Ю.А.Табарова/

## Пояснительная записка

Методические указания по выполнению практических работ подготовлены на основе рабочей программы учебной дисциплины МДК.11.01 «Технология разработки и защиты баз данных», разработанной на основе ФГОС СПО по специальности 09.02.07 «Информационные системы и программирование» и соответствующих общих (ОК) и профессиональных (ПК) компетенций:

ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам.

ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности

ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста

ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках.

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

При выполнении практических работ студент должен *знать*:

- методы описания схем баз данных в современных СУБД;
- основные положения теории баз данных, хранилищ данных, баз знаний;
- основные принципы структуризации и нормализации базы данных;
- основные принципы построения концептуальной, логической и физической модели данных;
- структуры данных СУБД, общий подход к организации представлений, таблиц, индексов и кластеров;
- методы организации целостности данных;
- технологии передачи и обмена данными в компьютерных сетях;
- алгоритм проведения процедуры резервного копирования;
- алгоритм проведения процедуры восстановления базы данных;
- способы контроля доступа к данным и управления привилегиями;
- основы разработки приложений баз данных;
- основные методы и средства защиты данных в базе данных

При выполнении практических работ студент должен *уметь*:

- работать с документами отраслевой направленности;
- собирать, обрабатывать и анализировать информацию на предпроектной стадии;
- работать с современными case-средствами проектирования баз данных;
- создавать объекты баз данных в современных СУБД;

- применять стандартные методы для защиты объектов базы данных.
- выполнять стандартные процедуры резервного копирования и мониторинга выполнения этой процедуры.
- выполнять процедуру восстановления базы данных и вести мониторинг выполнения этой процедуры
- выполнять установку и настройку программного обеспечения для обеспечения работы пользователя с базой данных.
- обеспечивать информационную безопасность на уровне базы данных

Содержание практических занятий определено рабочей программой и тематическим планированием, соответствует теоретическому материалу изучаемых разделов учебной дисциплины. Объем практических занятий по дисциплине определяется учебным планом по данной специальности. Продолжительность практического занятия – 2 академических часа. Перед проведением практического занятия преподавателем организуется инструктаж, а по ее окончании – обсуждение итогов.

Комплект методических указаний по выполнению практических работ по дисциплине МДК.11.01 «Технология разработки и защиты баз данных» содержит 68 практических занятий.

## **Перечень практических работ**

**по дисциплине МДК.11.01 «Технология разработки и защиты баз данных»**

### **ПРАКТИЧЕСКАЯ РАБОТА № 1.**

Тема: Сбор и анализ данных

### **ПРАКТИЧЕСКАЯ РАБОТА № 2.**

Тема: Сбор и анализ данных

### **ПРАКТИЧЕСКАЯ РАБОТА № 3.**

Тема: Создание концептуальной модели БД

### **ПРАКТИЧЕСКАЯ РАБОТА № 4.**

Тема: Создание концептуальной модели БД

### **ПРАКТИЧЕСКАЯ РАБОТА № 5.**

Тема: Построение логической схемы БД

### **ПРАКТИЧЕСКАЯ РАБОТА № 6.**

Тема: Построение логической схемы БД

### **ПРАКТИЧЕСКАЯ РАБОТА № 7.**

Тема: Приведение БД к нормальной форме 3НФ

### **ПРАКТИЧЕСКАЯ РАБОТА № 8.**

Тема: Приведение БД к нормальной форме 3НФ

### **ПРАКТИЧЕСКАЯ РАБОТА № 9.**

Тема: Проектирование реляционной схемы базы данных в среде СУБД.

### **ПРАКТИЧЕСКАЯ РАБОТА № 10.**

Тема: Проектирование реляционной схемы базы данных в среде СУБД.

### **ПРАКТИЧЕСКАЯ РАБОТА № 11.**

Тема: Модификация отношений БД

### **ПРАКТИЧЕСКАЯ РАБОТА № 12.**

Тема: Работа с первичными, вторичными ключами отношений БД

### **ПРАКТИЧЕСКАЯ РАБОТА № 13**

Тема: Установка и настройка SQL-сервера

### **ПРАКТИЧЕСКАЯ РАБОТА № 14**

Тема: Создание базы данных в среде разработки на языке SQL

### **ПРАКТИЧЕСКАЯ РАБОТА № 15**

Тема: Взаимосвязи между отношениями БД

### **ПРАКТИЧЕСКАЯ РАБОТА № 16**

Тема: Организация локальной сети. Настройка локальной сети

### **ПРАКТИЧЕСКАЯ РАБОТА № 17**

Тема: Обработка данных БД в модели «Клиент-Сервер» с использованием простых SQL запросов

### **ПРАКТИЧЕСКАЯ РАБОТА № 18**

Тема: Обработка данных БД в модели «Клиент-Сервер» с использованием простых SQL запросов

### **ПРАКТИЧЕСКАЯ РАБОТА № 19**

Тема: Обработка данных БД в модели «Клиент-Сервер» с использованием многотабличных SQL запросов

### **ПРАКТИЧЕСКАЯ РАБОТА № 20**

Тема: Обработка данных БД в модели «Клиент-Сервер» с использованием многотабличных SQL запросов



## **ПРАКТИЧЕСКАЯ РАБОТА № 21**

Тема: Создание запросов на группировку и Сортировку данных. Запросы на изменение. Использование встроенных функций

## **ПРАКТИЧЕСКАЯ РАБОТА № 22**

Тема: Создание запросов на группировку и Сортировку данных. Запросы на изменение. Использование встроенных функций

## **ПРАКТИЧЕСКАЯ РАБОТА № 23**

Тема: Создание БД в среде разработки

## **ПРАКТИЧЕСКАЯ РАБОТА № 24**

Тема: Создание БД в среде разработки

## **ПРАКТИЧЕСКАЯ РАБОТА № 25**

Тема: Создание БД в среде разработки

## **ПРАКТИЧЕСКАЯ РАБОТА № 26**

Тема: Создание БД в среде разработки

## **ПРАКТИЧЕСКАЯ РАБОТА № 27**

Тема: Программирование баз данных на языке C#. Технология ADO.NET. Соединение с базой данных

## **ПРАКТИЧЕСКАЯ РАБОТА № 28**

Тема: Программирование баз данных на языке C#. Технология ADO.NET. Соединение с базой данных

## **ПРАКТИЧЕСКАЯ РАБОТА № 29**

Тема: Программирование баз данных на языке C#. Технология ADO.NET. Соединение с базой данных

## **ПРАКТИЧЕСКАЯ РАБОТА № 30**

Тема: Программное подключение к БД в Visual studio c#. Создание запросов

## **ПРАКТИЧЕСКАЯ РАБОТА № 31**

Тема: Программное подключение к БД в Visual studio c#. Создание запросов

## **ПРАКТИЧЕСКАЯ РАБОТА № 32**

Тема: Программное подключение к БД в Visual studio c#. Создание запросов

## **ПРАКТИЧЕСКАЯ РАБОТА № 33**

Тема: Управление событиями в программировании БД

## **ПРАКТИЧЕСКАЯ РАБОТА № 34**

Тема: Управление событиями в программировании БД

## **ПРАКТИЧЕСКАЯ РАБОТА № 35**

Тема: Определение класса для подключения к базе данных. Создание многооконного интерфейса

## **ПРАКТИЧЕСКАЯ РАБОТА № 36**

Тема: Определение класса для подключения к базе данных. Создание многооконного интерфейса

## **ПРАКТИЧЕСКАЯ РАБОТА № 37**

Тема: Определение класса для подключения к базе данных. Создание многооконного интерфейса

## **ПРАКТИЧЕСКАЯ РАБОТА № 38**

Тема: Параметризованные запросы на C# ADO.NET

## **ПРАКТИЧЕСКАЯ РАБОТА № 39**

Тема: Параметризованные запросы на C# ADO.NET

## **ПРАКТИЧЕСКАЯ РАБОТА № 40**

Тема: Параметризованные запросы на C# ADO.NET

#### **ПРАКТИЧЕСКАЯ РАБОТА № 41**

Тема: Поиск и фильтрация данных в базе данных

#### **ПРАКТИЧЕСКАЯ РАБОТА № 42**

Тема: Поиск и фильтрация данных в базе данных

#### **ПРАКТИЧЕСКАЯ РАБОТА № 43**

Тема: Поиск и фильтрация данных в базе данных

#### **ПРАКТИЧЕСКАЯ РАБОТА № 44**

Тема: Создание отчетных форм в клиентских приложениях

#### **ПРАКТИЧЕСКАЯ РАБОТА № 45**

Тема: Создание отчетных форм в клиентских приложениях

#### **ПРАКТИЧЕСКАЯ РАБОТА № 46**

Тема: Создание отчетных форм в клиентских приложениях

#### **ПРАКТИЧЕСКАЯ РАБОТА № 47.**

Тема: Экспорт данных базы в документы пользователя

#### **ПРАКТИЧЕСКАЯ РАБОТА № 48.**

Тема: Импорт данных пользователя в базу данных

#### **ПРАКТИЧЕСКАЯ РАБОТА № 49.**

Тема: Выполнение настроек для автоматизации обслуживания БД

#### **ПРАКТИЧЕСКАЯ РАБОТА № 50**

Тема: Выполнение резервного копирования

#### **ПРАКТИЧЕСКАЯ РАБОТА № 51**

Тема: Выполнение резервного копирования

#### **ПРАКТИЧЕСКАЯ РАБОТА № 52**

Тема: Восстановление БД из резервной копии

#### **ПРАКТИЧЕСКАЯ РАБОТА № 53**

Тема: Восстановление БД из резервной копии

#### **ПРАКТИЧЕСКАЯ РАБОТА № 54.**

Тема: Реализация доступа пользователей к БД. Назначение/отмена привилегий пользователя для доступа к объектам БД

#### **ПРАКТИЧЕСКАЯ РАБОТА № 55.**

Тема: Реализация доступа пользователей к БД. Назначение/отмена привилегий пользователя для доступа к объектам БД

#### **ПРАКТИЧЕСКАЯ РАБОТА № 56.**

Тема: Поиск требуемой информации в БД с использованием операторов объединения таблиц

#### **ПРАКТИЧЕСКАЯ РАБОТА № 57.**

Тема: Поиск требуемой информации в БД с использованием операторов объединения таблиц

#### **ПРАКТИЧЕСКАЯ РАБОТА № 58.**

Тема: Поиск требуемой информации в БД с использованием операторов объединения таблиц

#### **ПРАКТИЧЕСКАЯ РАБОТА № 59.**

Тема: Поиск требуемой информации в БД с использованием операторов лево/правостороннего объединения таблиц и хранимых процедур

#### **ПРАКТИЧЕСКАЯ РАБОТА № 60.**

Тема: Поиск требуемой информации в БД с использованием операторов лево/правостороннего объединения таблиц и хранимых процедур

**ПРАКТИЧЕСКАЯ РАБОТА № 61.**

Тема: Поиск требуемой информации в БД с использованием операторов лево/правостороннего объединения таблиц и хранимых процедур

**ПРАКТИЧЕСКАЯ РАБОТА № 62.**

Тема: Мониторинг безопасности работы с базами данных

**ПРАКТИЧЕСКАЯ РАБОТА № 63.**

Тема: Мониторинг безопасности работы с базами данных

**ПРАКТИЧЕСКАЯ РАБОТА № 64.**

Тема: Резервное копирование БД, журнализация транзакций пользователя

**ПРАКТИЧЕСКАЯ РАБОТА № 65.**

Тема: Резервное копирование БД, журнализация транзакций пользователя

**ПРАКТИЧЕСКАЯ РАБОТА № 66.**

Тема: Установка приоритетов

**ПРАКТИЧЕСКАЯ РАБОТА № 67.**

Тема: Установка приоритетов

**ПРАКТИЧЕСКАЯ РАБОТА № 68.**

Тема: Мониторинг сетевого трафика

## **ИНСТРУКЦИИ ДЛЯ ОБУЧАЮЩИХСЯ ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ**

Прежде чем приступить к выполнению заданий, внимательно прочитайте данные рекомендации. Практические работы включают в себя задания следующих видов.

### **1. Работа за компьютером**

В ходе выполнения практических работ студент должен:

- выполнять требования по охране труда
- соблюдать инструкцию по правилам и мерам безопасности в кабинете информационных технологий
- строго выполнять весь объем работы, указанный в задании
- соблюдать требования эксплуатации компьютерной техники (правила включения и выключения)
- предоставить отчет о проделанной работе по окончании выполненной работы, который должен содержать:

1. Название работы.
2. Цель работы.
3. Задание и его решение.
4. Вывод о проделанной работе.

Текст отчета по практической работе должен быть набран на компьютере шрифтом Times New Roman размером 14 пт. (при оформлении текста используется текстовый редактор MS Word). Шрифт, используемый в иллюстративном материале (таблицы и рисунки), рекомендуется уменьшить до 12 пт. Межстрочный интервал в основном тексте - полуторный. В иллюстративном материале межстрочный интервал рекомендуется сделать одинарным. Поля страницы должны быть: левое поле - 30 мм; правое поле – 15 мм; верхнее и нижнее поле - 20 мм.

Каждый абзац должен начинаться с красной строки. Отступ абзаца – 1,25 см от левой границы текста.

Студент должен выполнить практическую работу самостоятельно (или в группе, если это предусмотрено заданием). Практическая работа выполняется согласно заданию и методическим рекомендациям. После выполнения практической работы обучающийся самостоятельно себя контролирует путем ответов на вопросы. Результат работы представляется преподавателю в виде файла (файлов) в личном каталоге, защищается обучающимися.

По ходу выполнения работы при возникновении вопросов обучающийся может получить консультацию у преподавателя или самостоятельно воспользоваться лекционным материалом, рекомендуемой литературой.

### **2. Ответ на поставленные вопросы (с аргументацией)**

Прочитайте вопрос и вникните в него.

Для удобства подчеркните ту, фразу, которая, по вашему мнению, является главной. Это поможет вам быстрее сориентироваться при ответе на вопрос.

Если вы считаете, что можете ответить на вопрос без помощи лекции и дополнительной литературы – приступайте. Если же вопрос заставляет вас сомневаться, откройте лекционную тетрадь (учебник или дополнительную литературу), прочитайте необходимый пункт, вникните в содержание и после этого приступайте за работу.

**ГЛАВНОЕ!** Не переписывайте отрывки лекции в рабочую тетрадь! Четко отвечайте на ПОСТАВЛЕННЫЙ вопрос!

Не забудьте привести аргументацию (обоснование) вашей позиции, если вопрос предполагает личностное отношение к проблеме.

### **3. Заполнение таблиц и схем**

Прочитайте название таблицы или схемы.

Исходя из названия, вы поймете цель предстоящей работы.

Воспользуйтесь материалами лекций или другими источниками, чтобы заполнить таблицу (схему).

Используйте цветные графические материалы для выделения строк, столбцов или элементов схем.

Особое внимание обращайтесь на четкость при отборе материала: делайте записи кратко и четко!

**4. Поиск информации в сети** – использование web-браузеров, баз данных, пользование информационно-поисковыми и информационно-справочными системами, автоматизированными библиотечными системами, электронными журналами. Поиск и обработка информации включает подготовку фрагмента практического занятия.

## ПРАКТИЧЕСКАЯ РАБОТА № 1

### Тема: Сбор и анализ данных

**Цель работы:** ознакомиться с процессом описания БД и получить навыки по использованию основных методов анализа данных.

**Оборудование:** ПК, интернет, программное обеспечение – программа для онлайн-диаграмм, MS Word, инструкции по выполнению работы.

#### Содержание работы:

##### Задание 1.

1. Ознакомиться с предложенным вариантом описания предметной области (согласно заданию индивидуального задания).
2. Проанализировать предметную область, уточнив и дополнив ее, руководствуясь собственным опытом, консультациями и любыми источниками (книгами, учебниками или Интернет-источниками).
3. Выполнить структурное разбиение предметной области на отдельные подразделения (подсистемы) согласно выполняемым ими функциям.
4. Определить задачи и функции системы в целом и функции каждого подразделения (подсистемы).
5. Продумать подробное описание работы каждого подразделения (подсистемы), алгоритмов и сценариев выполнения ими отдельных работ. Продумать виды входной и выходной информации для каждого подразделения (подсистемы).
6. Описать схему работы будущей БД, учитывая выделенные и описанные ранее подсистемы.
7. Определить группу пользователей, для которой данная система будет более востребована. Описать перечень функций системы, которые будут доступны данной группе пользователей.
8. Создать физическую диаграмму в соответствии с описанием деятельности предметной области.

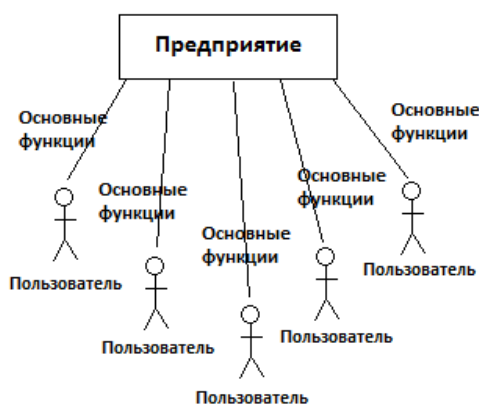


Рисунок 1 – Схематическое отображение физической диаграммы предметной области

9. Осуществить идентификацию опорных точек зрения, построив диаграмму идентификации точек зрения.

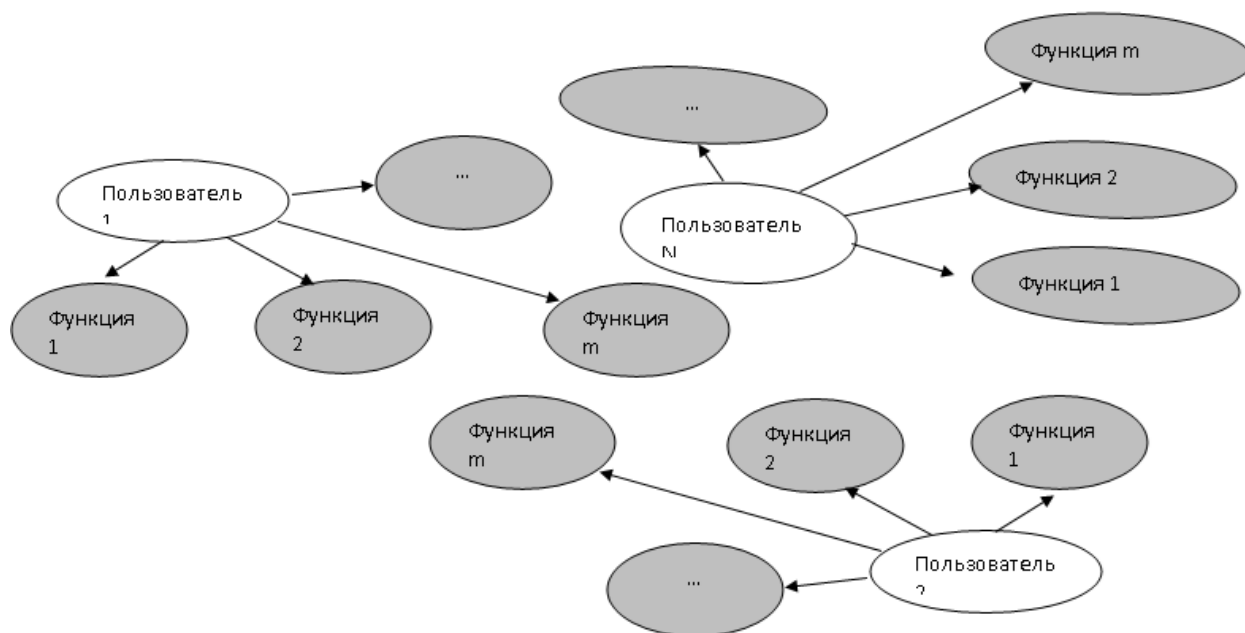


Рисунок 2 – Пример диаграммы идентификации точек зрения

10. Построить таблицу «Соотнесение точек зрения с выполняемыми функциями».

Точка зрения 1	Точка зрения 2	...	Точка зрения N
функция 1_1	функция 2_1		функция N_1
функция 1_2	функция 2_2		функция N_2
...	...		...
функция 1_m	функция 2_m		функция N_m

11. Расписать основные функциональные возможности администратора системы, как одного из пользователей системы.

12. Подготовить отчет по работе

### Индивидуальные задания:

Вариант 1. Страховая медицинская компания.

*Сущность задачи:* Страховая медицинская компания (СМК) заключает договора добровольного медицинского страхования с населением и договора с лечебными учреждениями на лечение застрахованных клиентов. При возникновении страхового случая клиент подает заявку на оказание медицинских услуг по условиям договора инспектору, который работает с данным клиентом. Инспектор направляет данного клиента в лечебное учреждение. Отчеты о своей деятельности инспектор предоставляет в бухгалтерию. Бухгалтерия проверяет оплату договоров, перечисляет денежные средства за оказанные услуги лечебным учреждениям, производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики. СМК не только оплачивает лечение застрахованного лица при возникновении с ним страхового случая, но и, при возникновении каких-либо осложнений после лечения, оплачивает лечение этих осложнений.

Вариант 2. Агентство недвижимости

*Сущность задачи:* Агентство недвижимости занимается покупкой, продажей, сдачей в аренду объектов недвижимости по договорам с их собственниками. Агентство управляет объектами недвижимости как физических, так и юридических лиц. Собственник может иметь несколько объектов. В случае покупки или аренды клиент может произвести осмотр объекта. В качестве одной из услуг, предлагаемых агентством, является проведение инспектирования текущего состояния объекта для адекватного определения его рыночной цены. По

результатам своей деятельности агентство производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики.

### Вариант 3. Компания по разработке программных продуктов

*Сущность задачи:* Компания заключает договор с клиентом на разработку программного продукта согласно техническому заданию. После утверждения технического задания определяется состав и объем работ, составляется предварительная смета. На каждый проект назначается ответственный за его выполнение – куратор проекта, который распределяет нагрузку между программистами и следит за выполнением технического задания. Когда программный продукт готов, то его внедряют, производят обучение клиента и осуществляют дальнейшее сопровождение. По результатам своей деятельности компания производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики.

### Вариант 4: Кадровое агентство

*Сущность задачи:* Кадровое агентство способствует трудоустройству безработных граждан. Агентство ведет учет и классификацию данных о безработных на основании резюме от них. От предприятий города поступают данные о свободных вакансиях, на основании которых агентство предлагает различные варианты трудоустройства соискателям. В случае положительного исхода поиска вакансии считается заполненной, а безработный становится трудоустроенным. По результатам своей деятельности кадровое агентство производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики.



## ПРАКТИЧЕСКАЯ РАБОТА № 2

### Тема: Сбор и анализ данных

**Цель работы:** ознакомиться с процессом описания БД и получить навыки по использованию основных методов анализа данных.

**Оборудование:** ПК, интернет, программное обеспечение – программа для онлайн-диаграмм, MS Word, инструкции по выполнению работы.

#### Содержание работы:

##### Задание 1.

1. Ознакомиться с предложенным вариантом описания предметной области (согласно заданию индивидуального задания).
2. Проанализировать предметную область, уточнив и дополнив ее, руководствуясь собственным опытом, консультациями и любыми источниками (книгами, учебниками или Интернет-источниками).
3. Выполнить структурное разбиение предметной области на отдельные подразделения (подсистемы) согласно выполняемым ими функциям.
4. Определить задачи и функции системы в целом и функции каждого подразделения (подсистемы).
5. Продумать подробное описание работы каждого подразделения (подсистемы), алгоритмов и сценариев выполнения ими отдельных работ. Продумать виды входной и выходной информации для каждого подразделения (подсистемы).
6. Описать схему работы будущей БД, учитывая выделенные и описанные ранее подсистемы.
7. Определить группу пользователей, для которой данная система будет более востребована. Описать перечень функций системы, которые будут доступны данной группе пользователей.
8. Создать физическую диаграмму в соответствии с описанием деятельности предметной области.

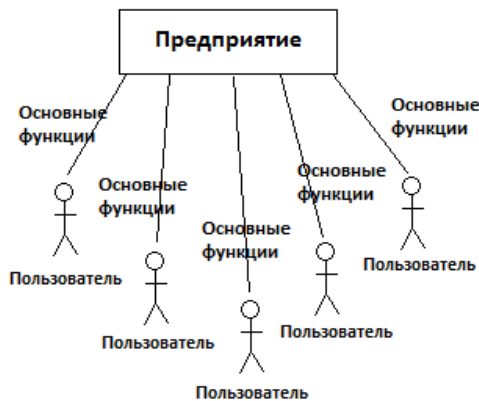


Рисунок 1 – Схематическое отображение физической диаграммы предметной области

9. Осуществить идентификацию опорных точек зрения, построив диаграмму идентификации точек зрения.

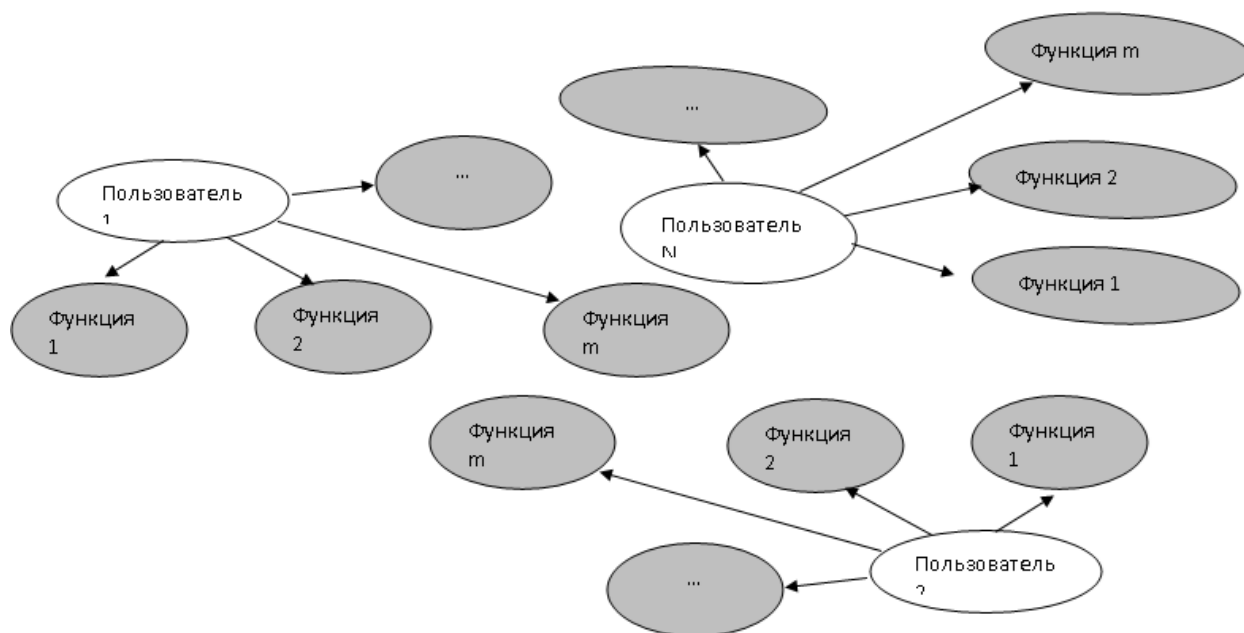


Рисунок 2 – Пример диаграммы идентификации точек зрения

10. Построить таблицу «Соотнесение точек зрения с выполняемыми функциями».

Точка зрения 1	Точка зрения 2	...	Точка зрения N
функция 1_1	функция 2_1		функция N_1
функция 1_2	функция 2_2		функция N_2
...	...		...
функция 1_m	функция 2_m		функция N_m

11. Расписать основные функциональные возможности администратора системы, как одного из пользователей системы.

12. Подготовить отчет по работе

### Индивидуальные задания:

#### Вариант 1. Строительная организация

*Сущность задачи:* Строительная организация занимается строительством объектов по заказам клиентов. Сначала заказ проходит предварительную стадию: сбор различных разрешений на строительство, составление эскиза объекта, расчет объема и закупка строительных материалов. Сами строительные материалы доставляются на объект партиями. По мере поступления очередной партии стройматериалов закладывается фундамент объекта, строится каркас здания. По результатам данной работы происходит согласование с заказчиком, после чего утепляется контур, вставляются окна, устанавливается крыша. Далее идет обсуждение с клиентом внутренней отделки здания, закупаются отделочные материалы. После того, как объект проходит технический контроль, он передается заказчику. В дополнительные услуги строительной организации входят: услуги дизайнера по интерьеру, закупка и доставка мебели, сотрудничество с охранным предприятием по установке сигнализации. По результатам своей деятельности строительная организация производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики.

#### Вариант 2. Компьютерная компания

*Сущность задачи:* Компьютерная компания занимается продажей, ремонтом, сборкой, тестированием компьютерной техники. Также, специалисты компании предоставляют услуги по разработке и монтажу локальных вычислительных сетей. Вся техника и комплектующие закупаются оптом у дилеров и хранятся на складе.

Клиент, который хочет приобрести товар, оформляет заказ в торговом зале, а забирает технику со склада или оставляет заявку на ее доставку. Клиент, который хочет отремонтировать технику, приносит ее в сервисный отдел, откуда, по прошествии некоторого времени, забирает как от-ремонтированную или как технику, не подлежащую ремонту. По желанию клиента, специалисты компании могут выехать к клиенту для общей диагностики возникшей проблемы с техникой. По результатам своей деятельности компьютерная компания производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики.

Вариант 3. Компания по предоставлению телекоммуникационных услуг

*Сущность задачи:* Компания занимается оказанием телекоммуникационных услуг абонентам. Клиент делает заявку на подключение к телекоммуникационным услугам и ему, по необходимости, устанавливают соответствующее оборудование. Оплата за услуги вносится путем авансовых платежей. Каждый факт предоставления услуги фиксируется соответствующим оборудованием и является основанием для списания соответствующей суммы с личного счета абонента. Клиент в любое время суток может получить отчет об оказанных ему услугах, их стоимости и остатку на личном счете абонента. По результатам своей деятельности компания производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики.

Вариант 4. Спортивный комплекс

*Сущность задачи:* Спортивный комплекс предоставляет услуги по проведению спортивных тренировок. Тренировки, относящиеся к одному виду спорта, объединяются в спортивные секции. Клиент обращается в спортивный комплекс, где получает абонемент на посещение спортивной секции. На основе купленных абонементов составляется расписание тренировок на следующий месяц. Также, в зависимости от загруженности спортивного комплекса, распределяются тренеры спортивных секций. По результатам своей деятельности спортивный комплекс производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики.

### ПРАКТИЧЕСКАЯ РАБОТА № 3.

#### Тема: Создание концептуальной модели БД

**Цель работы:** научиться создавать концептуальную схему базы данных для решения конкретной экономической задачи в соответствии с индивидуальным вариантом.

**Оборудование:** ПК, интернет, программное обеспечение – программа для онлайн-диаграмм, MS Word, инструкции по выполнению работы.

#### Содержание работы:

**Задание 1.** Разработать концептуальную модель БД «Курсовая работа»

Список потенциальных сущностей для рассматриваемого примера будет представлен таблицей вида

Варианты заданий
График
Графическая часть
Задание
Замечания, дополнения
Курсовая работа
Литература
Методические указания
Оценка за курсовую работу
Положение о курсовом проектировании
Пояснительная записка
Преподаватель
Расчеты
Список литературы
Студент

Теперь из этого списка необходимо выделить сущности, остальные интерфейсные дуги будут преобразованы в атрибуты сущностей. В качестве сущностей выделим следующие:

- 1) задание;
- 2) пояснительная записка;
- 3) курсовая работа;
- 4) положение о курсовом проектировании;
- 5) студент;
- 6) преподаватель;
- 7) график;
- 8) методические указания.

Запустите любую онлайн-программу для создания диаграмм, например, SmartDraw и выберите Диаграмма ER. Создайте 8 сущностей.

Далее необходимо установить связи между сущностями. Сначала составим описание предметной области на естественном языке. Любой студент должен выполнить одну или несколько курсовых работ. Каждая курсовая работа должна выполняться одним студентом (в идеале). Каждая курсовая работа выполняется в соответствии с методическими указаниями и положением о курсовом проектировании. Курсовая работа сдается по графику. Курсовая работа оформляется в виде пояснительной записки. Преподаватель проводит консультации, проверяет и ставит оценку за курсовую работу.

Таким образом, сформулируем имена связей:

СТУДЕНТ выполняет КУРСОВУЮ РАБОТУ.  
ПРЕПОДАВАТЕЛЬ проверяет КУРСОВУЮ РАБОТУ.  
КУРСОВАЯ РАБОТА выполняется в соответствии с ЗАДАНИЕМ.  
КУРСОВАЯ РАБОТА оформляется в виде ПОЯСНИТЕЛЬНОЙ ЗАПИСКИ.  
МЕТОДИЧЕСКИЕ УКАЗАНИЯ определяют требования к КУРСОВОЙ РАБОТЕ.  
КУРСОВАЯ РАБОТА организуется согласно ПОЛОЖЕНИЮ ПО КУРСОВОМУ ПРОЕКТИРОВАНИЮ.  
КУРСОВАЯ РАБОТА сдается по ГРАФИКУ.

Во всех случаях сущность Курсовая работа является дочерней, за исключением связи с сущностью Пояснительная записка. Определим типы связей и построим модель (рисунок. 1). В дальнейшем можно будет подкорректировать связи между сущностями.

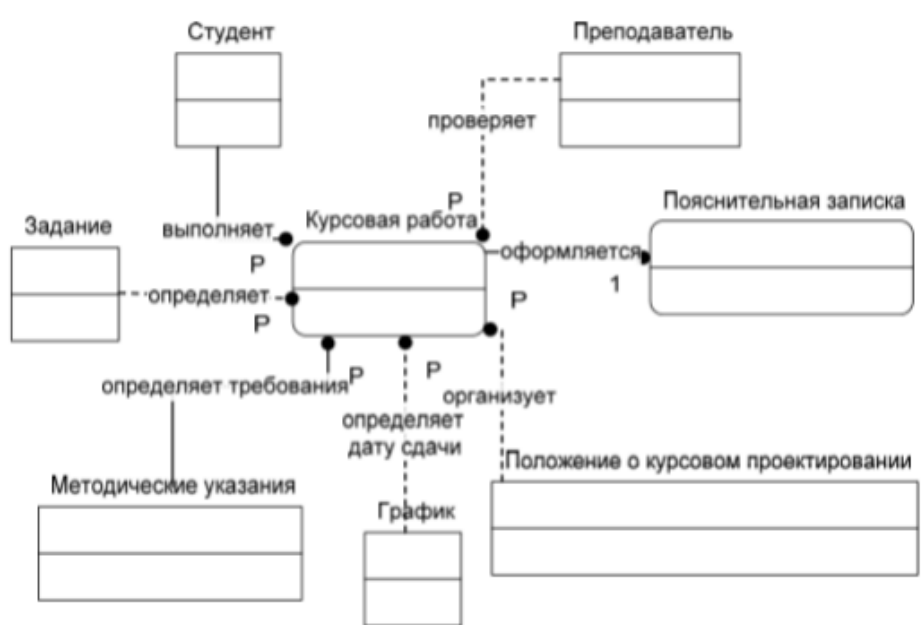


Рисунок 1. Информационная модель уровня «сущность-связь»

Далее необходимо определить ключевые атрибуты для каждой сущности, обращая внимание на то, что дочерние сущности наследуют ключевые атрибуты от родительских (рисунок 2).

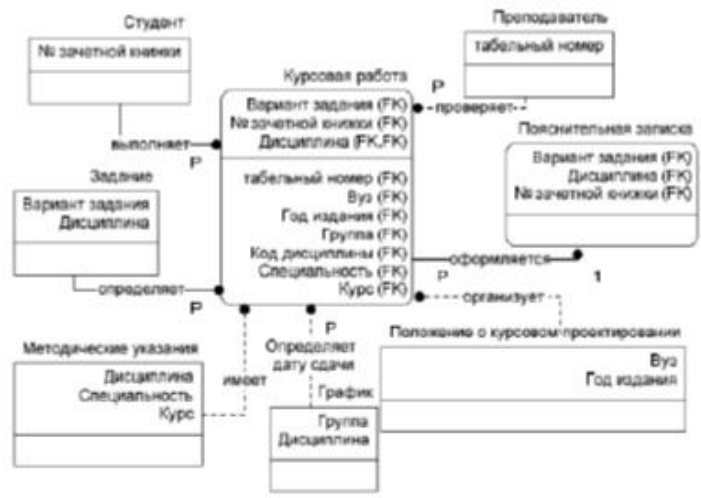


Рисунок 2. Информационная модель с ключевыми атрибутами

Как видно из рисунка 2 по сравнению с информационной моделью уровня «сущность-связь», был изменен тип связи между сущностями Методические указания и Курсовая работа, поскольку ключевые атрибуты сущности Методические указания для сущности Курсовая работа будут являться

избыточными (зная номер зачетной книжки, можно узнать специальность и курс, на котором учится студент).

Кроме того, отметим, что три сущности (Задание, График, Методические указания) содержат одинаковые атрибуты Дисциплина. Это является некорректным. Чтобы устранить данную ошибку, выделим одноименную сущность и свяжем ее идентифицирующими связями с вышеуказанными сущностями (рисунок 3).



Рисунок 3. Скорректированная модель, основанная на ключах

**Задание 2.** В соответствии с индивидуальным вариантом создайте концептуальную схему базы данных.

**Варианты индивидуальных заданий:**

- 1. Отделение коммерческого банка
- 2. Поликлиника
- 3. Колледж
- 4. Отделение полиции
- 5. Дизайнерская фирма
- 6. Офис интернет-провайдера
- 7. Агентство недвижимости
- 8. Туристическое агентство
- 9. Офис благотворительного фонда
- 10. Издательство
- 11. Рекламное агентство
- 12. Отделение налоговой службы
- 13. Редакция газеты
- 14. Гостиница
- 15. Праздничное агентство
- 16. Городской архив
- 17. Диспетчерская служба такси
- 18. Железнодорожная касса

## **ПРАКТИЧЕСКАЯ РАБОТА № 4.**

### **Тема: Создание концептуальной модели БД**

**Цель работы:** научиться создавать концептуальную схему базы данных для решения конкретной экономической задачи в соответствии с индивидуальным вариантом.

**Оборудование:** ПК, интернет, программное обеспечение – программа для онлайн-диаграмм, MS Word, инструкции по выполнению работы.

#### **Содержание работы:**

**Задание 1.** В соответствии с индивидуальным вариантом создайте концептуальную схему базы данных.

#### **Варианты индивидуальных заданий:**

1. Рекламное агентство
2. Отделение налоговой службы
3. Редакция газеты
4. Гостиница
5. Праздничное агентство
6. Городской архив
7. Диспетчерская служба такси
8. Железнодорожная касса

**Задание 2.** В соответствии с описанием предметной области из Практической работы № 2 создайте концептуальную схему базы данных.

## **ПРАКТИЧЕСКАЯ РАБОТА № 5.**

### **Тема: Построение логической схемы БД**

**Цель работы:** научиться проектировать логическую схему БД

**Оборудование:** ПК, интернет, программное обеспечение – программа для онлайн-диаграмм, MS Word, инструкции по выполнению работы.

#### **Справочный материал:**

Цель логического этапа проектирования - организация данных, выделенных на этапе инфологического проектирования в форму, принятую в выбранной СУБД. Задачей логического этапа проектирования является отображение объектов предметной области в объекты используемой модели данных, чтобы это отображение не противоречило семантике предметной области и было по возможности наилучшим (эффективным, удобным и т.д.). С точки зрения выбранной СУБД задача логического проектирования реляционной базы данных состоит в обоснованном принятии решений о том:

- из каких отношений должна состоять база данных;
- какие атрибуты должны быть у этих отношений;
- какие ограничения должны быть наложены на атрибуты и отношения базы данных, чтобы обеспечить ее целостность.

Требования к выбранному набору отношений и составу их атрибутов должны удовлетворять следующим условиям:

- отношения должны отличаться минимальной избыточностью атрибутов;
- выбранные для отношения первичные ключи должны быть минимальными;
- между атрибутами не должно быть нежелательных функциональных зависимостей;
- выбор отношений и атрибутов должен обеспечивать минимальное дублирование данных;
- не должно быть трудностей при выполнении операций включения, удаления и модификации данных;
- время выполнения запросов на выборку данных должно удовлетворять предъявляемым требованиям;
- перестройка набора отношений при введении новых типов должна быть минимальной.

#### **Содержание работы:**

**Задание 1.** Рассмотрим БД Факультет, состоящую из трех отношений (таблиц):

СТУДЕНТ (Номер, Фамилия И.О., Группа)

СЕССИЯ (Номер, Оценка1, Оценка2, Оценка3, Средний балл)

СТИПЕНДИЯ (Средний балл, процент стипендии)

Таблицы в БД связываются между собой по ключу. У таблиц СТУДЕНТ и СТИПЕНДИЯ ключи простые: таблица СТУДЕНТ имеет ключ Номер, а таблица СТИПЕНДИЯ – Средний балл. У таблицы СЕССИЯ ключ составной: Номер (для связи с таблицей СТУДЕНТ) и Средний балл (для связи с таблицей СТИПЕНДИЯ). Информационно-логическая модель БД Факультет представлена на рисунке.





Все информационные объекты предметной области должны быть связаны между собой. Различаются связи нескольких типов, для которых введены следующие обозначения:

- один к одному (1:1);
- один ко многим (1:M) или (1:∞);
- многие ко многим (M:M) или (∞:∞).

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. Такой тип связи существует между отношениями **СТУДЕНТ** и **СЕССИЯ** - каждый студент имеет один набор оценок за сессию.

При связи один ко многим (1:M) одному экземпляру информационного объект А соответствует 0,1 или больше экземпляров объекта В, но каждый экземпляр объекта В связан не более чем с одним экземпляром объекта А. Данный тип связи существует между таблицами **СТИПЕНДИЯ** и **СЕССИЯ**. В таблице **СЕССИЯ** хранится множество наборов оценок для всех студентов факультета, а число градаций стипендии ограничено.

Связь многие со многими (M:M) предполагает, что в каждый момент времени одному экземпляру информационного объекта А соответствует 0,1 или более экземпляров объекта В и наоборот. Такой тип связи может быть установлен, например, между таблицами **СТУДЕНТ** и **ПРЕПОДАВАТЕЛЬ**. Каждый преподаватель работает со многими студентами и каждый студент учится у многих преподавателей.

**Задание 2.** В соответствии с построенной концептуальной модели в Практической работе № 3 построить логическую модель схемы БД.

## **ПРАКТИЧЕСКАЯ РАБОТА № 6.**

### **Тема: Построение логической схемы БД**

**Цель работы:** научиться проектировать логическую схему БД

**Оборудование:** ПК, интернет, программное обеспечение – программа для онлайн-диаграмм, MS Word, инструкции по выполнению работы.

### **Содержание работы:**

**Задание 1.** В соответствии с построенными концептуальными моделями в Практической работе № 4 построить логические модели схемы БД.

## **ПРАКТИЧЕСКАЯ РАБОТА № 7.**

### **Тема: Приведение БД к нормальной форме 3НФ**

**Цель работы:** изучить нормальные формы отношений, научиться приводить отношения в соответствии с ними.

**Оборудование:** ПК, интернет, программное обеспечение – MS Word, инструкции по выполнению работы

#### **Справочный материал:**

Сущность (отношение) – это класс однотипных объектов, информация о которых должна быть учтена в модели. Каждая сущность должна иметь наименование.

Экземпляр сущности – конкретный представитель данной сущности.

Атрибут сущности – это именованная характеристика, являющаяся некоторым свойством сущности.

Первичный ключ – минимальный набор атрибутов, по значениям которых можно однозначно найти требуемый экземпляр сущности (строку таблицы).

Внешний ключ – атрибут зависимой таблицы, с помощью которого производится ссылка на первичный ключ другой таблицы, т.е. производится связывание таблиц.

Нормализация – разбиение исходного отношения на два или несколько, обладающих лучшими свойствами при включении, изменении и удалении данных. Целью нормализации является получение БД, где каждый факт появляется только один раз, т.е. исключена избыточность, причем к этому стремятся для исключения возможной противоречивости хранимых данных.

Нормализация выражается в приведении отношений в соответствии с некоторыми требованиями, называемыми нормальными формами: 1) первая нормальная форма (1НФ), 2) вторая нормальная форма (2НФ), 3) третья нормальная форма (3НФ), 4) нормальная форма Бойса-Кодда (НФБК).

#### **Порядок выполнения работы:**

**Задание 1:** Требуется спроектировать БД для сети продуктовых магазинов, в которой будут храниться данные о поступлении товаров в магазины и их продажах.

##### Исходные данные:

1.Товар производится производителем. При этом: у каждого товара обязательно есть производитель, и каждый производитель производит хотя бы один товар.

2.У товара есть тип. Каждый товар обязательно относится к какому-либо типу, причем, только к одному. В магазине могут быть в наличии товары не всех типов. Товаров одного типа может быть несколько.

3.Экземпляр товара находится в магазине или продан. Экземпляр товара обязательно находится в каком-то магазине, и в каждом магазине обязательно есть товары. Каждый экземпляр товара может быть только в одном магазине, при этом в каждом магазине может быть множество экземпляров товаров.

Составим универсальное отношение (рис.1а) – отношение, включающее все атрибуты и содержащее все данные БД. Таблица будет иметь 5 полей: «Производитель» со всеми данными о производителе, «Магазин» со всеми данными о магазине, «Тип товара», «Товар», «Экземпляр товара».

С учетом полученных сведений можно заключить, что будет иметь место дублирование данных (рис.1б), равно как и отсутствие значений некоторых полей в записях.

Универсальное отношение	
Производитель	
Магазин	
Тип Товара	
Товар	
Экземпляр товара	

а – универсальное отношение

Производитель	Магазин	Тип Товара	Товар	Экземпляр товара
Kraft Foods	Eurospar	Конфеты	Alpen Gold Composition	Alpen Gold Composition ассорти из тем
Kraft Foods	Eurospar	Конфеты	Alpen Gold Composition	Alpen Gold Composition ассорти из мо
Kraft Foods	Eurospar	Конфеты	Alpen Gold Composition	Alpen Gold Composition С ликером Ам
Kraft Foods	Eurospar	Конфеты	Alpen Gold Composition	Alpen Gold Composition Вишня в конь
Kraft Foods	Eurospar	Конфеты	Alpen Gold Composition	Alpen Gold Composition конфеты с кон
*				

б – дублирование данных

Рис.1. Дублирование данных в универсальном отношении

Рассмотрим процесс нормализации отношений и нормальные формы.

*Первая нормальная форма.*

Все атрибуты отношения простые, т.е. их значения неделимы. В данном случае все атрибуты не являются простыми. Так, например, данные о производителе можно разделить на название, телефон, адрес и т.д. 1НФ содержит правило об исключении повторяющихся групп, т.е. атрибут некоторого экземпляра сущности должен содержать одно значение, а не список значений. Например, атрибут «Цена» сущности «Товар» должен содержать только одно значение цены для каждого экземпляра этой сущности.

После приведения отношения к 1НФ можно получить, например, такое отношение:

Первая НФ	
Название производителя	
Адрес производителя	
Телефон производителя	
Название магазина	
Адрес магазина	
Телефон магазина	
Тип товара	
Название товара	
Количество товара	
Цена товара	
Дата поступления	
Продано	

Рис.2. Первая НФ

*Вторая нормальная форма.*

Каждое неключевое поле связано полной функциональной зависимостью с первичным ключом. Полной функциональной зависимостью называется ситуация, при которой значение данного атрибута определяется значением некоторого составного атрибута и не определяется значением любой его части.

Ключ может быть составным, т.е. состоять из нескольких полей. Поэтому 2НФ фактически добавляет требование о том, чтобы значение любого неключевого поля зависело от значения всего ключа и не зависело от значения его части.

В примере таким ключом могла бы быть совокупность полей: «Название производителя», «Название магазина», «Тип товара», но например, значение поля «Адрес производителя» зависит от части ключа «Название производителя».

Решением проблемы является введение дополнительного атрибута «Код», являющегося первичным ключом. После этого отношение будет выглядеть так, как показано на рисунке 3.

Вторая НФ	
Код	
Название производителя	
Адрес производителя	
Телефон производителя	
Название магазина	
Адрес магазина	
Телефон магазина	
Тип товара	
Название товара	
Количество товара	
Цена товара	
Дата поступления	
Продано	

Рис.3. Вторая НФ

*Третья нормальная форма.*

Любой неключевой атрибут нетранзитивно зависит от ключа. Транзитивная зависимость – ситуация, при которой в некоторой тройке атрибутов первый зависит от второго, второй от третьего, а третий от первого. В рассмотренном отношении транзитивные зависимости присутствуют.

Исключить транзитивные зависимости можно, разбив данное отношение на пять отношений: «Производитель», «Магазин», «Тип товара», «Товар» и «Экземпляр товара». При этом в каждое из них следует добавить поле, являющееся первичным ключом отношения, например, поле с именем «Код».

Кроме того, чтобы данные не потеряли связи между собой, нужно добавить в таблицы вторичные ключи. Этими ключами будут: в таблице «Товар» - «Производитель», «Тип»; в таблице «Экземпляр товара» - «Товар». В результате получим схему данных, изображенную на рисунке 4:

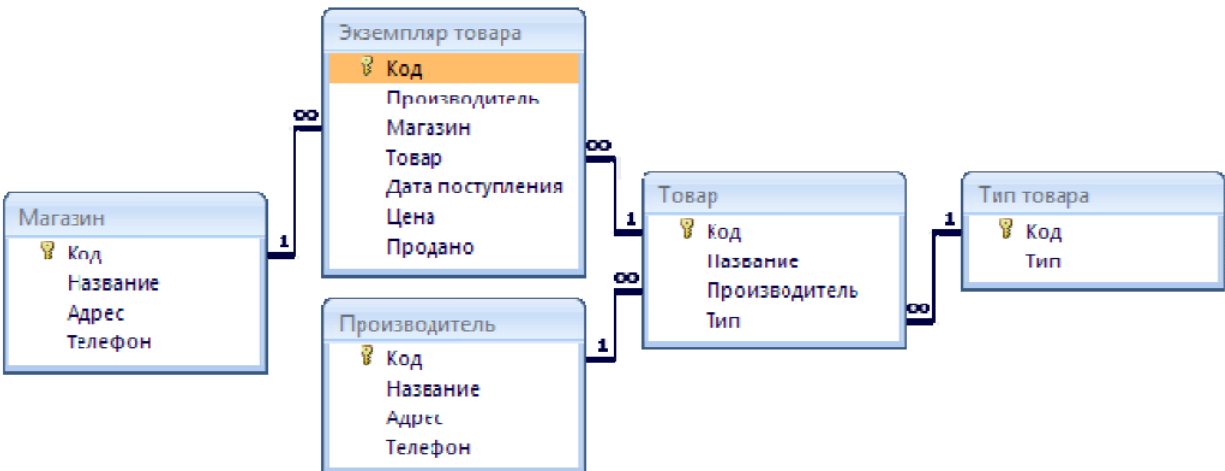


Рис.4. Третья НФ

3НФ содержит правило о том, что каждый атрибут, не являющийся ключевым, должен зависеть только от ключа сущности и не зависеть от других

атрибутов. Требование о независимости атрибутов друг от друга означает, что изменение одного атрибутов, не входящего в ключ сущности, никак не отразится на другом.

*Нормальная форма Бойса-Кодда.*

Каждое поле, по значению которого можно однозначно идентифицировать запись, по требованиям соответствует первичному ключу, хотя может им и не являться. Требования НФБК отношения не нарушают.

Каждая последующая нормальная форма является недостижимой, пока не будут достигнуты все предыдущие.

**Задание 2.** Спроектировать БД для учебного центра, в которой будут храниться данные о преподавателях, курсах и графике учебного процесса.

## **ПРАКТИЧЕСКАЯ РАБОТА № 8.**

### **Тема: Приведение БД к нормальной форме 3НФ**

**Цель работы:** изучить нормальные формы отношений, научиться приводить отношения в соответствии с ними.

**Оборудование:** ПК, интернет, программное обеспечение – MS Word, инструкции по выполнению работы

#### **Порядок выполнения работы:**

**Задание 1.** Спроектировать БД для торговой фирмы, в которой будут храниться данные о товарах и распределение их различным партнерам.

#### **Контрольные вопросы:**

1. Понятие первичного и вторичного ключей.
2. Что такое нормализация? Каковы ее цели?
3. Что такое нормальная форма? По какому принципу они строятся?
4. Какие требования предъявляют к каждой нормальной форме?
5. Недостатки моделирования структуры БД с помощью алгоритма нормализации.

## ПРАКТИЧЕСКАЯ РАБОТА № 9.

**Тема: Проектирование реляционной схемы базы данных в среде СУБД.**

**Цель работы:** научить проектировать реляционные схемы БД

**Оборудование:** ПК, интернет, программное обеспечение – SmartDraw, MS Word, инструкции по выполнению работы.

**Содержание работы:**

**Задание 1.** Разработать базу данных «Учеба студентов»

### 1. 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 стажем 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.

### 2. 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	Экзаменационно-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 на физическом 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. Это целое число. Т.е. для идентификации каждого факультета будет применяться 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. Однако для идентификации каждой кафедры первичного ключа недостаточно, так как каждая кафедра принадлежит определенному факультету. 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 кафедре. Для этого будем использовать внешний ключ.
ВК (внешний ключ)	Код кафедры	С помощью данного внешнего ключа будет определено к какой кафедре принадлежит каждый преподаватель.
	ФИО	
	должность	Ассистент, доцент, профессор, 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	
	Расчетная	
	Форма контроля	

*Список атрибутов таблицы «Ведомости»*

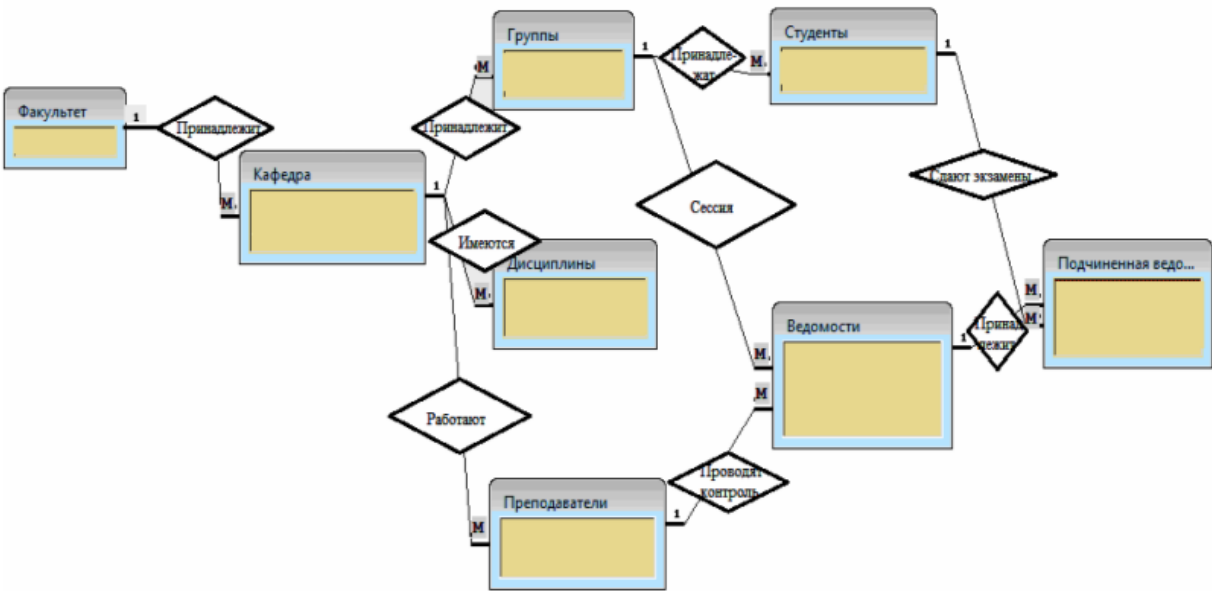
Ключевое поле	Название	Назначение
ПК (первичный ключ)	Код ведомости	Ключевое поле. Представляет собой первичный ключ. Это уникальное значение, соответствующее каждой учебной ведомости. Однако для идентификации каждой ведомости первичного ключа недостаточно, так как каждая ведомость выписывается для определенной учебной группы 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	

3. Построение инфологической модели

Инфологическую модель лучше представить графически, где будут изображены все таблицы и связи между ними. В нашем случае схема связей представлена на рисунке



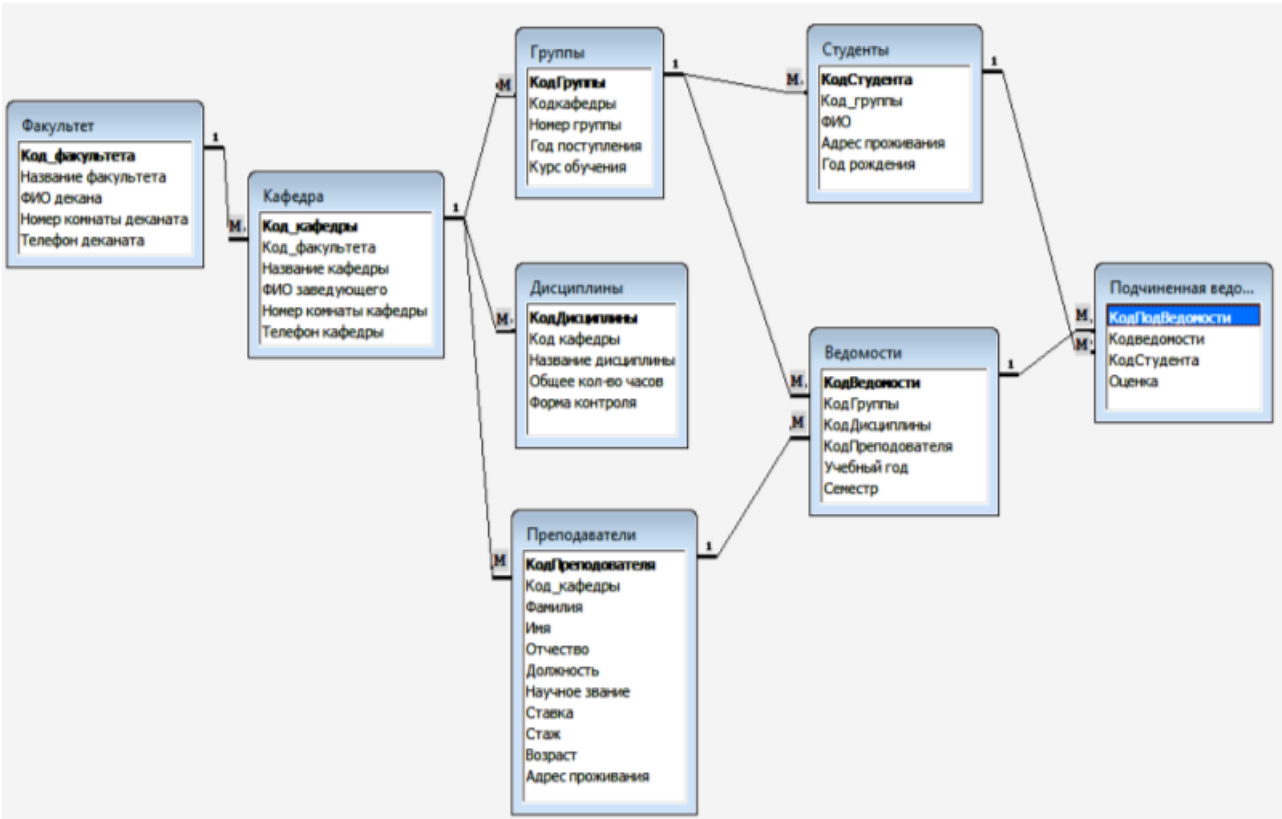
Для выявленных связей заполним таблицу

## 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

### 4. Построение даталогической модели БД.

Даталогическая модель отражается графически в виде схемы базы данных, где указываются 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 заведующего	FIO	Текстовый	нет	
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

**Задание 2.** Спроектировать БД по выбранному индивидуальному заданию.

№ варианта	Условие
1	<p>На основании выбранного варианта выполнить следующее:</p> <ol style="list-style-type: none"> <li>1. Выполнить анализ предметной области исследуемой организации;</li> <li>2. Описать основные сущности предметной области;</li> <li>3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями;</li> <li>4. Построить инфологическую модель базы данных организации;</li> <li>5. Построить даталогическую модель базы данных организации.</li> </ol> <p><b>БД – успеваемость студентов ВУЗА.</b></p> <p>БД состоит из следующих таблиц: факультеты, кафедры, учебные группы, студенты, ведомости успеваемости.</p> <p>Таблица факультеты имеет следующие атрибуты: название факультета, ФИО декана, номер комнаты, номер корпуса, телефон.</p> <p>Таблица кафедры имеет следующие атрибуты: название кафедры, факультет, ФИО заведующего, номер комнаты, номер корпуса, телефон, кол-во преподавателей.</p> <p>Таблица учебные группы имеет следующие атрибуты: название группы, год поступления, курс обучения, кол-во студентов в группе.</p> <p>Таблица студенты имеет следующие атрибуты: студента, фамилия, имя, отчество, группа, год рождения, пол, адрес, город, телефон.</p> <p>Таблица ведомости успеваемости имеет следующие атрибуты: группа, студент, предмет, оценка.</p>
2	<p>На основании выбранного варианта выполнить следующее:</p> <ol style="list-style-type: none"> <li>1. Выполнить анализ предметной области исследуемой организации;</li> <li>2. Описать основные сущности предметной области;</li> <li>3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями;</li> <li>4. Построить инфологическую модель базы данных организации;</li> <li>5. Построить даталогическую модель базы данных организации.</li> </ol> <p><b>БД – информационная система супермаркета.</b> БД состоит из следующих таблиц: отделы, сотрудники, товары, продажа товаров, должности.</p> <p>Таблица отделы имеет следующие атрибуты: название отдела, кол-во прилавков, кол-во продавцов, номер зала.</p> <p>Таблица сотрудники имеет следующие атрибуты: фамилия, имя, отчество, отдел, год рождения, год поступления на работу, стаж, должность, пол, адрес, город, телефон.</p> <p>Таблица должности имеет следующие атрибуты: название должности, сумма ставки.</p> <p>Таблица товары имеет следующие атрибуты: название товара,</p>

	<p>отдел, страна производитель, условия хранения, сроки хранения.</p> <p>Таблица продажа товаров имеет следующие атрибуты: сотрудника являющегося продавцом, товара дата, время, кол-во, цена, сумма.</p>
3	<p>На основании выбранного варианта выполнить следующее:</p> <ol style="list-style-type: none"> <li>1. Выполнить анализ предметной области исследуемой организации;</li> <li>2. Описать основные сущности предметной области;</li> <li>3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями;</li> <li>4. Построить инфологическую модель базы данных организации;</li> <li>5. Построить даталогическую модель базы данных организации.</li> </ol> <p><b>БД – информационная система библиотеки.</b> БД состоит из следующих таблиц: библиотеки, фонд библиотеки, тип литературы, сотрудники, пополнение фонда.</p> <p>Таблица библиотеки имеет следующие атрибуты: название, адрес, город.</p> <p>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.</p> <p>Таблица тип литературы имеет следующие атрибуты: название типа.</p> <p>Таблица сотрудники имеет следующие атрибуты: фамилия сотрудника, библиотека, должность, год рождения, год поступления на работу, образование, зарплата.</p> <p>Таблица пополнение фонда имеет следующие атрибуты: фонд, сотрудник, дата, название источника литературы, тип литературы, издательство, дата издания, кол-во экземпляров.</p>
4	<p>На основании выбранного варианта выполнить следующее:</p> <ol style="list-style-type: none"> <li>1. Выполнить анализ предметной области исследуемой организации;</li> <li>2. Описать основные сущности предметной области;</li> <li>3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями;</li> <li>4. Построить инфологическую модель базы данных организации;</li> <li>5. Построить даталогическую модель базы данных организации.</li> </ol> <p><b>БД – информационная система туристического агентства.</b> БД состоит из следующих таблиц: пансионаты, туры, клиенты, путевки, вид жилья.</p> <p>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.</p> <p>Таблица вид жилья имеет следующие атрибуты: название (дом, бунгало, квартира, 1-я комната, 2-я комната и т.д.), категория жилья (люкс, полуплюкс, и т.д.), пансионат, описание условий</p>



	<p>проживания, цена за номер в сутки.</p> <p>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.</p> <p>Таблица клиенты имеет следующие атрибуты: фамилия, имя, отчество, паспортные данные, дата рождения, адрес, город, телефон.</p> <p>Таблица путевки имеет следующие атрибуты: клиент, пансионата, вид жилья, дата заезда, дата отъезда, наличие детей, наличие мед. страховки, кол-во человек, цена, сумма</p>
5	<p>На основании выбранного варианта выполнить следующее:</p> <ol style="list-style-type: none"> <li>1. Выполнить анализ предметной области исследуемой организации;</li> <li>2. Описать основные сущности предметной области;</li> <li>3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями;</li> <li>4. Построить инфологическую модель базы данных организации;</li> <li>5. Построить даталогическую модель базы данных организации.</li> </ol> <p><b>БД – информационная система поликлиники.</b> БД состоит из следующих таблиц: врачи, пациенты, история болезней, отделения, обслуживающий персонал.</p> <p>Таблица отделения имеет следующие атрибуты: название отделения (хирургия, терапия, неврология и т.д.), этаж, номера комнат, ФИО заведующего.</p> <p>Таблица врачи имеет следующие атрибуты: фамилия, имя, отчество, должность, стаж работы, научное звание, адрес, номер отделения, в котором он работает.</p> <p>Таблица пациенты имеет следующие атрибуты: фамилия, имя, отчество, адрес, город, возраст, пол.</p> <p>Таблица диагнозы имеет следующие атрибуты: название диагноза, признаки болезни, период лечения, назначения.</p> <p>Таблица история болезни имеет следующие атрибуты: пациент, врач, диагноз, лечение, дата заболевания, дата вылечивания, вид лечения (амбулаторное, стационарное)</p>
6	<p>На основании выбранного варианта выполнить следующее:</p> <ol style="list-style-type: none"> <li>1. Выполнить анализ предметной области исследуемой организации;</li> <li>2. Описать основные сущности предметной области;</li> <li>3. Расставить существующие связи между сущностями;</li> <li>4. Построить инфологическую модель базы данных организации;</li> <li>5. Построить даталогическую модель базы данных организации.</li> </ol> <p><b>БД – информационная система библиотек города.</b> БД состоит из следующих таблиц: библиотеки, читальные залы, литература, читатели, выдача литературы.</p> <p>Таблица библиотеки имеет следующие атрибуты: название, адрес, город.</p> <p>Таблица читальные залы имеет следующие атрибуты: название</p>

	<p>читального зала, библиотека, кол-во единиц литературы, кол-во посадочных мест, время работы, этаж, кол-во сотрудников.</p> <p>Таблица читатели имеет следующие атрибуты: фамилия, имя, отчество, категория читателя, место работы или обучения, возраст, дата регистрации в библиотеке.</p> <p>Таблица литература имеет следующие атрибуты: название, категория литературы, авторы, издательство, год издательства, кол-во страниц, читальный зал.</p> <p>Таблица выдача литературы имеет следующие атрибуты: читатель, литература, дата выдачи, срок выдачи, вид выдачи, наличие залога.</p>
7	<p>На основании выбранного варианта выполнить следующее:</p> <ol style="list-style-type: none"> <li>1. Выполнить анализ предметной области исследуемой организации;</li> <li>2. Описать основные сущности предметной области;</li> <li>3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями;</li> <li>4. Построить инфологическую модель базы данных организации;</li> <li>5. Построить даталогическую модель базы данных организации.</li> </ol> <p><b>БД – информационная система автосалона.</b> БД состоит из следующих таблиц: автомобили, марка автомобиля, сотрудники, продажа автомобилей, покупатели.</p> <p>Таблица марка автомобиля имеет следующие атрибуты: название марки, страна производитель, завод производитель, адрес.</p> <p>Таблица автомобиля имеет следующие атрибуты: название автомобиля, марка, год производства, цвет, категория, цена.</p> <p>Таблица покупатели имеет следующие атрибуты: фамилия, имя, отчество, паспортные данные, адрес, город, возраст, пол.</p> <p>Таблица сотрудника имеет следующие атрибуты: фамилия, имя, отчество, стаж, зарплата.</p> <p>Таблица продажа автомобилей имеет следующие атрибуты: дата, сотрудник, автомобиль, покупатель.</p>
8	<p>На основании выбранного варианта выполнить следующее:</p> <ol style="list-style-type: none"> <li>1. Выполнить анализ предметной области исследуемой организации;</li> <li>2. Описать основные сущности предметной области;</li> <li>3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями;</li> <li>4. Построить инфологическую модель базы данных организации;</li> <li>5. Построить даталогическую модель базы данных организации.</li> </ol> <p><b>БД – торговая организация.</b> БД состоит из следующих таблиц: торговая организация, торговая точка, продавцы, поставщики, заказы поставщикам.</p> <p>Таблица торговая организация имеет следующие атрибуты: название торговой организации, адрес, ФИО директора, налоговый номер.</p>

	<p>Таблица торговая точка имеет следующие атрибуты: название торговой точки, тип (универмаги, магазины, киоски, лотки и т.д.), торговая организация, адрес, ФИО заведующего.</p> <p>Таблица продавцы имеет следующие атрибуты: фамилия, имя, отчество, торговая точка, должность, год рождения, пол, адрес проживания, город.</p> <p>Таблица поставщики имеет следующие атрибуты: название поставщика, тип деятельности, страна, город, адрес.</p> <p>Таблица заказы поставщикам имеет следующие атрибуты: дата заказа, торговая точка, поставщик, название товара, кол-во, цена.</p>
9	<p>На основании выбранного варианта выполнить следующее:</p> <ol style="list-style-type: none"> <li>1. Выполнить анализ предметной области исследуемой организации;</li> <li>2. Описать основные сущности предметной области;</li> <li>3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями;</li> <li>4. Построить инфологическую модель базы данных организации;</li> <li>5. Построить даталогическую модель базы данных организации.</li> </ol> <p><b>БД – проектная организация.</b> БД состоит из следующих таблиц: отделы, сотрудники, организации, договора, проектные работы.</p> <p>Таблица отделы имеет следующие атрибуты: название отдела, этаж, телефон, начальник отдела.</p> <p>Таблица сотрудники имеет следующие атрибуты: ФИО, должность (конструкторы, инженеры, техники, лаборанты, прочий обслуживающий персонал), номер отдела, в котором работает, пол, адрес, дата рождения.</p> <p>Таблица организации имеет следующие атрибуты: название организации, тип деятельности, страна, город, адрес, ФИО директора.</p> <p>Таблица договора имеет следующие атрибуты: номер договора, дата заключения договора, организация, стоимость договора.</p> <p>Таблица проектные работы имеет следующие атрибуты: дата начала проектной работы, дата завершения проектной работы, номер договора, отдел, осуществляющий разработку.</p>
10	<p>На основании выбранного варианта выполнить следующее:</p> <ol style="list-style-type: none"> <li>1. Выполнить анализ предметной области исследуемой организации;</li> <li>2. Описать основные сущности предметной области;</li> <li>3. Расставить существующие связи между сущностями: самостоятельно добавить в каждую сущность первичные ключи и установить внешние ключи между сущностями;</li> <li>4. Построить инфологическую модель базы данных организации;</li> <li>5. Построить даталогическую модель базы данных организации.</li> </ol> <p><b>БД – цирк.</b> БД состоит из следующих таблиц: работники цирка, представления, расписание гастролей, труппа цирка, программа цирка.</p> <p>Таблица работники цирка имеет следующие атрибуты: фамилия,</p>

	<p>имя, отчество, год рождения, год поступления на работу, стаж, должность (акробат, клоун, гимнаст, музыкант, постановщик, служащий и т.д.), пол, адрес, город, телефон.</p> <p>Таблица представления имеет следующие атрибуты: название, режиссер-постановщик, художник-постановщик, дирижер-постановщик, автор, жанр, тип.</p> <p>Таблица расписание гастролей имеет следующие атрибуты: представление, дата начала, дата окончания, места проведения гастрولي.</p> <p>Таблица труппа представления цирка имеет следующие атрибуты: представление, актер цирка, название роли.</p> <p>Таблица программа цирка имеет следующие атрибуты: представление, дата премьеры, период проведения, дни и время, цена билета.</p>
--	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

## **ПРАКТИЧЕСКАЯ РАБОТА № 10.**

**Тема: Проектирование реляционной схемы базы данных в среде СУБД.**

**Цель работы:** научить проектировать реляционные схемы БД

**Оборудование:** ПК, интернет, программное обеспечение – SmartDraw, MS Word, инструкции по выполнению работы.

**Содержание работы:**

**Задание 1.** Спроектировать БД «Спортивный комплекс»

*Сущность задачи:* Спортивный комплекс предоставляет услуги по проведению спортивных тренировок. Тренировки, относящиеся к одному виду спорта, объединяются в спортивные секции. Клиент обращается в спортивный комплекс, где получает абонемент на посещение спортивной секции. На основе купленных абонементов составляется расписание тренировок на следующий месяц. Также, в зависимости от загруженности спортивного комплекса, распределяются тренеры спортивных секций. По результатам своей деятельности спортивный комплекс производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики.

**Задание 2.** Спроектировать БД «Страховая медицинская компания».

*Сущность задачи:* Страховая медицинская компания (СМК) заключает договора добровольного медицинского страхования с населением и договора с лечебными учреждениями на лечение застрахованных клиентов. При возникновении страхового случая клиент подает заявку на оказание медицинских услуг по условиям договора инспектору, который работает с данным клиентом. Инспектор направляет данного клиента в лечебное учреждение. Отчеты о своей деятельности инспектор предоставляет в бухгалтерию. Бухгалтерия проверяет оплату договоров, перечисляет денежные средства за оказанные услуги лечебным учреждениям, производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики. СМК не только оплачивает лечение застрахованного лица при возникновении с ним страхового случая, но и, при возникновении каких-либо осложнений после лечения, оплачивает лечение этих осложнений.

**Задание 3.** Спроектировать БД «Компания по разработке программных продуктов»

*Сущность задачи:* Компания заключает договор с клиентом на разработку программного продукта согласно техническому заданию. После утверждения технического задания определяется состав и объем работ, составляется предварительная смета. На каждый проект назначается ответственный за его выполнение – куратор проекта, который распределяет нагрузку между программистами и следит за выполнением технического задания. Когда программный продукт готов, то его внедряют, производят обучение клиента и осуществляют дальнейшее сопровождение. По результатам своей деятельности компания производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики.

**Задание 4.** Спроектировать БД «Агентство недвижимости»

*Сущность задачи:* Агентство недвижимости занимается покупкой, продажей, сдачей в аренду объектов недвижимости по договорам с их собственниками. Агентство управляет объектами недвижимости как физических, так и юридических лиц. Собственник может иметь несколько объектов. В случае покупки или аренды клиент может произвести осмотр объекта. В качестве одной

из услуг, предлагаемых агентством, является проведение инспектирования текущего состояния объекта для адекватного определения его рыночной цены. По результатам своей деятельности агентство производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики.

**Задание 5.** Спроектировать БД «Компьютерная компания»

*Сущность задачи:* Компьютерная компания занимается продажей, ремонтом, сборкой, тестированием компьютерной техники. Также, специалисты компании предоставляют услуги по разработке и монтажу локальных вычислительных сетей. Вся техника и комплектующие закупаются оптом у дилеров и хранятся на складе. Клиент, который хочет приобрести товар, оформляет заказ в торговом зале, а забирает технику со склада или оставляет заявку на ее доставку. Клиент, который хочет отремонтировать технику, приносит ее в сервисный отдел, откуда, по прошествии некоторого времени, забирает как от-ремонтированную или как технику, не подлежащую ремонту. По желанию клиента, специалисты компании могут выехать к клиенту для общей диагностики возникшей проблемы с техникой. По результатам своей деятельности компьютерная компания производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики.

**Задание 6.** Спроектировать БД «Строительная организация»

*Сущность задачи:* Строительная организация занимается строительством объектов по заказам клиентов. Сначала заказ проходит предварительную стадию: сбор различных разрешений на строительство, составление эскиза объекта, расчет объема и закупка строительных материалов. Сами строительные материалы доставляются на объект партиями. По мере поступления очередной партии стройматериалов закладывается фундамент объекта, строится каркас здания. По результатам данной работы происходит согласование с заказчиком, после чего утепляется контур, вставляются окна, устанавливается крыша. Далее идет обсуждение с клиентом внутренней отделки здания, закупаются отделочные материалы. После того, как объект проходит технический контроль, он передается заказчику. В дополнительные услуги строительной организации входят: услуги дизайнера по интерьеру, закупка и доставка мебели, сотрудничество с охранным предприятием по установке сигнализации. По результатам своей деятельности строительная организация производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики.

## **ПРАКТИЧЕСКАЯ РАБОТА № 11.**

### **Тема: Модификация отношений БД**

**Цель работы:** получить практический опыт выполнения модификации базы данных.

**Оборудование:** ПК, интернет, программное обеспечение – SmartDraw, MS Word, инструкции по выполнению работы.

#### **Содержание работы:**

**Задание 1.** Руководство супермаркета желает разработать БД для хранения информации о счетах покупателей. Нужно создать базу, в которой для каждого покупателя будут храниться номер счёта, фамилия, адрес, номер телефона, кредитоспособность (высокая, средняя, низкая, слабая) и сведения об уплате налогов.

#### **Необходимо:**

1. Изобразить диаграмму функциональных зависимостей, используя соответствующие атрибуты и выделяя сделанные предположения.
2. Получить для БД отношения в нормальных формах.

**Задание 2.** Секретарь государственного учреждения проектирует БД для хранения информации обо всех автомобилях, зарегистрированных в штате. Нужно разработать диаграммы функциональных зависимостей для соответствующих атрибутов.

#### **Необходимо:**

1. Сформулировать назначение проектируемой БД и описать предметную область.
2. Определить информационные потребности конечных пользователей (в виде форм выходных документов для типовых запросов к БД).
3. Разработать инфологическую модель предметной области (в виде диаграммы «сущность-связь»).
4. Сформировать схему БД (путём преобразования диаграммы «сущность-связь»).
5. Рассчитать требуемый объём памяти для БД.

## **ПРАКТИЧЕСКАЯ РАБОТА № 12.**

**Тема: Работа с первичными, вторичными ключами отношений БД**

**Цель работы:** получить практический опыт в работе с первичными и вторичными ключами между отношениями БД.

**Оборудование:** ПК, интернет, программное обеспечение – SmartDraw, MS Word, инструкции по выполнению работы.

**Справочный материал:**

**Первичный ключ** (primary key) — это столбец или набор столбцов в таблице, который однозначно идентифицирует каждую строку.

**Вторичный ключ** (foreign key) — это столбец или набор столбцов в одной таблице, которые ссылаются на столбцы первичного ключа другой таблицы.

**Первичные ключи**

**Назначение:** обеспечивают уникальный поиск, обновление или удаление каждой записи для обеспечения целостности объекта.

**Свойства:**

- Значение ключа всегда уникально и не может быть пустым (Null).
- Может быть естественным (например, паспортные данные, фамилии и имена) или суррогатным (порядковый номер).
- Для каждой таблицы можно создавать только один первичный ключ.

**Вторичные ключи**

**Назначение:** создают связи между таблицами и обеспечивают целостность данных.

**Свойства:**

- Вторичный ключ не всегда уникален (в таблице может быть несколько записей с одинаковыми значениями вторичного ключа).
- Значения могут быть пустыми (NULL), например, если связанные записи отсутствуют.
- Определить вторичный ключ можно с помощью ограничений (в SQL — FOREIGN KEY), что обеспечит правильное сопоставление вторичного ключа с первичным.

**Содержание работы:**

**Задание 1.** В спроектированных БД (Практическая работа № 11) необходимо установить первичные и вторичные ключи.



## ПРАКТИЧЕСКАЯ РАБОТА № 13

### Тема: Установка и настройка SQL-сервера

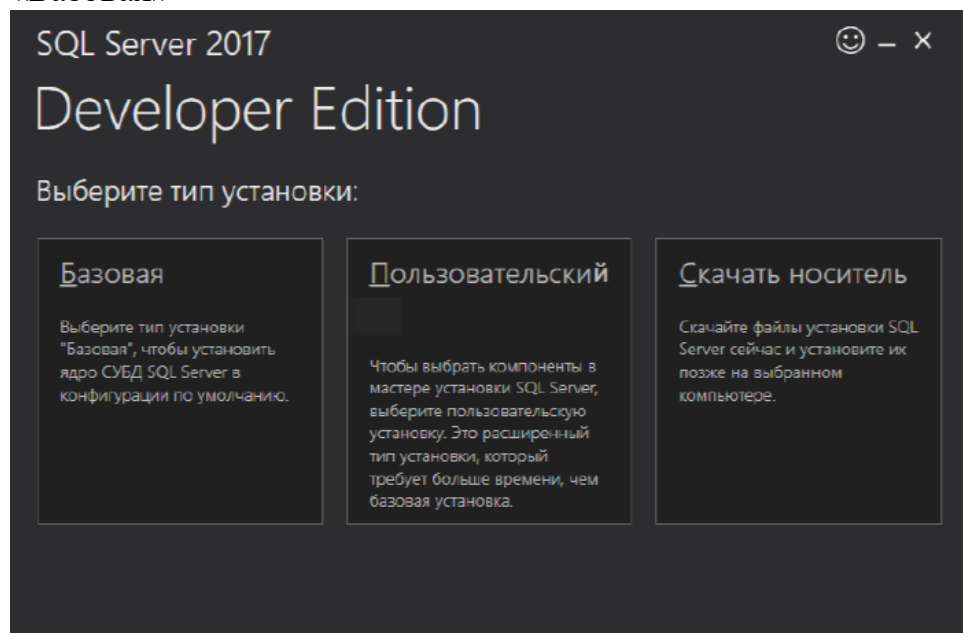
**Цель работы:** научиться устанавливать Microsoft SQL Server 2019 и настраивать его для создания БД.

**Оборудование:** ПК, интернет, программное обеспечение – Microsoft SQL Server 2019, MS SQL Server Management Studio, инструкции по выполнению работы

#### Содержание работы:

**Задание 1.** Установить MS SQL Server и MS SQL Server Management Studio

1. Для установки SQL Server с сайта <https://www.microsoft.com/ru-ru/sql-server/sql-server-downloads> выберите Developer Edition и нажмите кнопку «Скачать». После скачивания автоматически запускается установщик. Выберите тип установки – «Базовая»

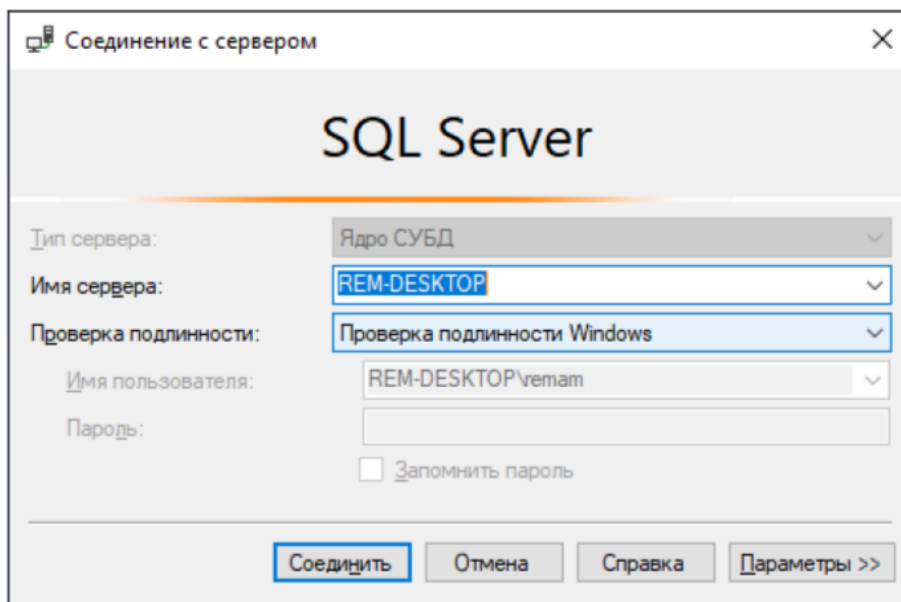


2. После принятия условия соглашения, укажите целевое расположение SQL Server

3. SQL Server Management Studio (SSMS) – это интегрированная среда для управления любой инфраструктурой SQL, от SQL Server до баз данных SQL Azure. SSMS предоставляет средства для настройки, наблюдения и администрирования экземпляров SQL Server и баз данных. С помощью SSMS можно развертывать, отслеживать и обновлять компоненты уровня данных, используемые вашими приложениями, а также создавать запросы и скрипты.

Скачайте SQL Server Management Studio (SSMS) с сайта: <https://docs.microsoft.com/ru-ru/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15>

4. Запускайте SQL Server Management Studio. В диалоговом окне Соединение с сервером подтвердите заданные по умолчанию параметры и нажмите кнопку Подключиться. Для соединения необходимо, чтобы поле Имя сервера содержало имя компьютера, на котором установлен SQL Server. Если компонент Database Engine представляет собой именованный экземпляр, то поле «Имя сервера» должно также содержать имя экземпляра в формате <имя\_компьютера>\<имя\_экземпляра>. Имя компьютера у каждого свое

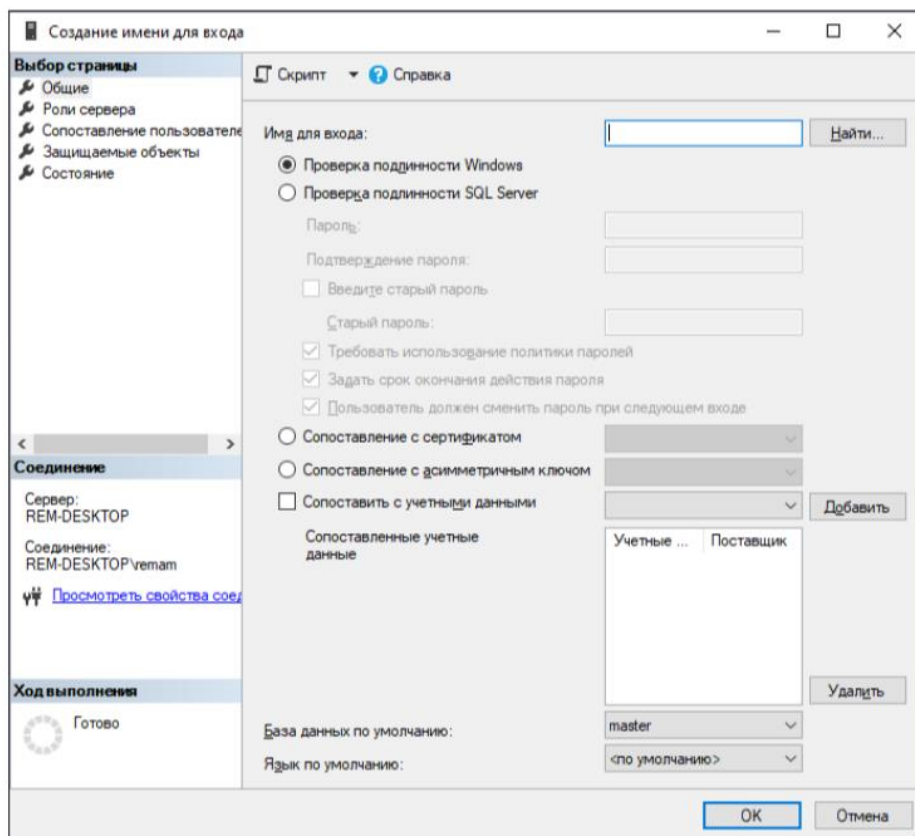


5. MS SQL Server использует два режима аутентификации: аутентификацию Windows и аутентификацию MS SQL Server (SQL Server authentication). Для аутентификации по умолчанию используется аутентификация Windows. Windows Authentication mode (Проверка подлинности Windows) – это режим, который использует для подключения к серверу только логины Windows. В этом случае пользователям нет необходимости вводить какие-то пароли при подключении к SQL Server, если они уже вошли в сеть Windows.

Для использования аутентификации SQL Server вначале необходимо создать учетную запись в самом SQL Server. В отличие от логинов Windows, логины SQL Server – это самостоятельные учетные записи со своими именами и паролями, информация о которых хранится в системной базе данных SQL Server. При подключении к серверу при помощи логина SQL Server вам придется указать имя логина и пароль.

6. Создайте новую учетную запись с именем STUDENT. В Management Studio список учетных записей, сконфигурированных на сервере, содержится в папке «\Безопасность\Имена для входа». Чтобы добавить новую учетную запись, необходимо выделить узел «Имена для входа» в контекстном меню и выбрать пункт «Создать имя для входа...».

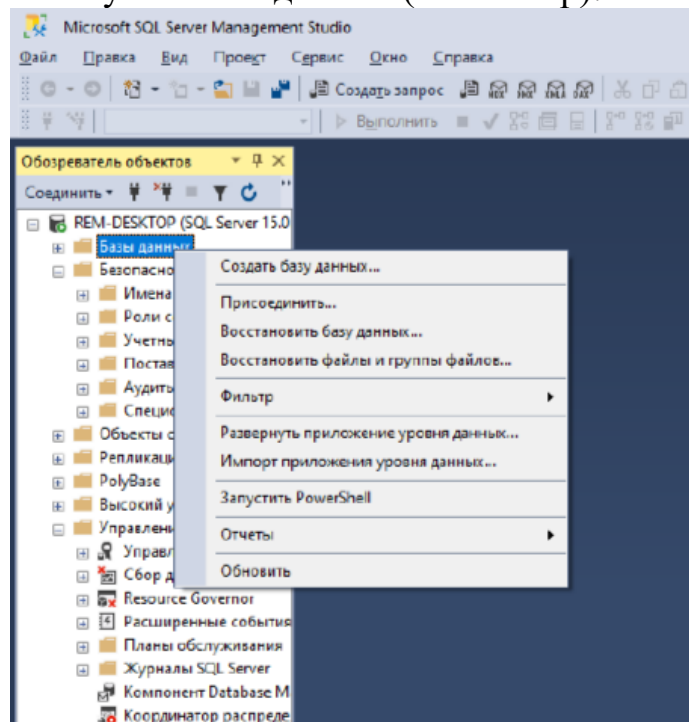
7. В открывшемся окне в поле «Имя для входа» введите *STUDENT*. Далее выберите переключатель «Проверка подлинности SQL Server» и в поле «Пароль» наберите пароль. Снимите флажок «Пользователь должен сменить пароль при следующем входе». Остальные поля оставьте без изменений.

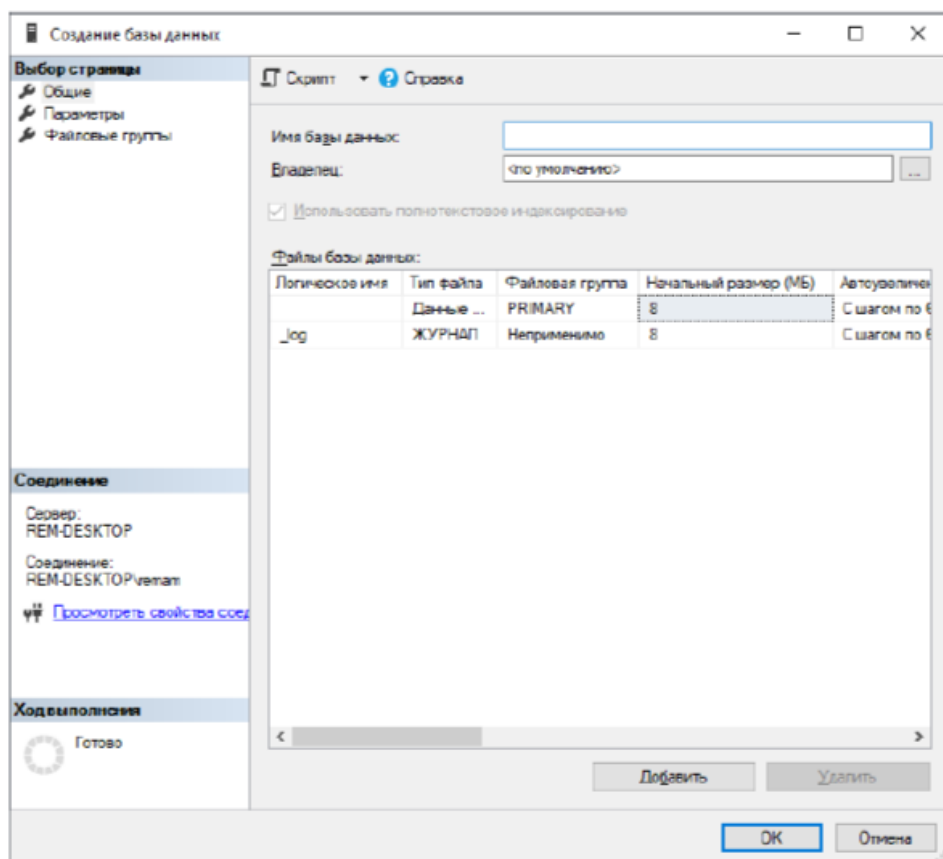


## Задание 2. Создать БД

Базу данных может создать только пользователь с правами администратора. Выберите в контекстном меню папки «Базы данных» команду «Создать базу данных...».

В поле «Имя базы данных» введите имя создаваемой базы данных (БД) – «Учебная». Здесь также можно изменить путь сохранения созданной БД. Обратите внимание, что создаётся пустая база данных (контейнер).





После создания БД окно «Обозреватель объектов» обновится. В ветке «Базы данных» появится новая база данных «Учебная». Эта база данных принадлежит пользователю по имени sysadmin (системный администратор).

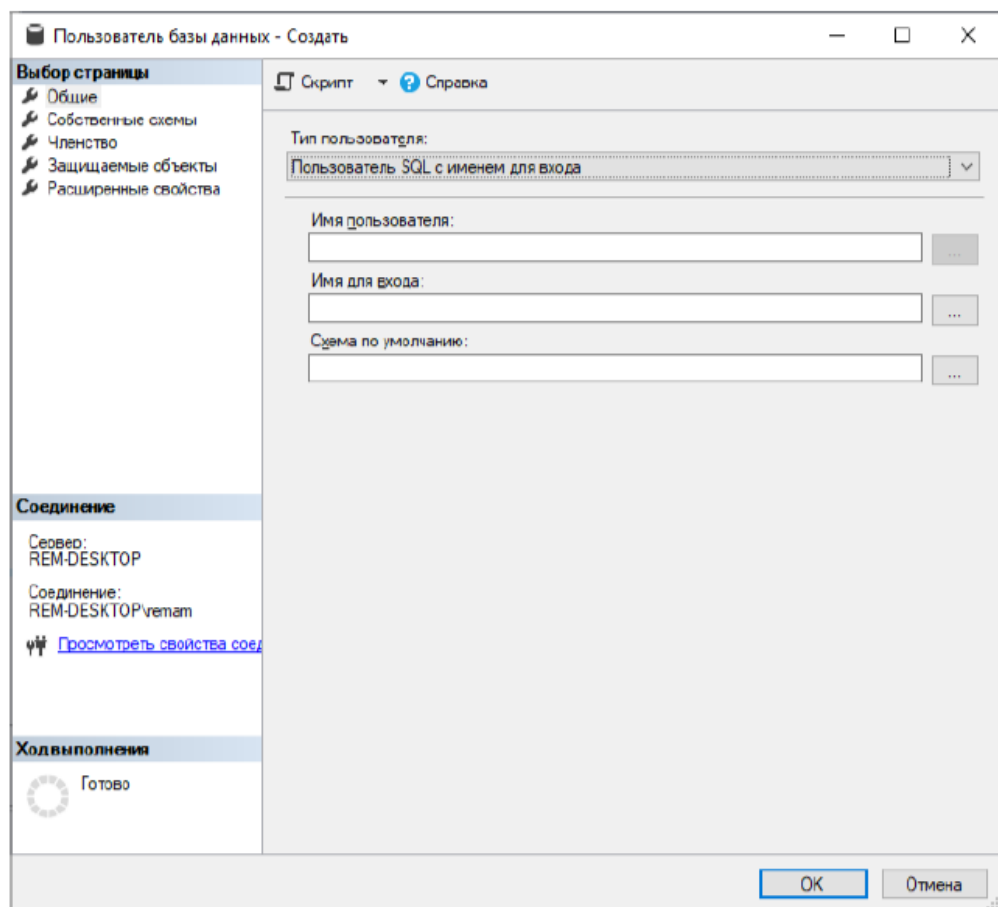
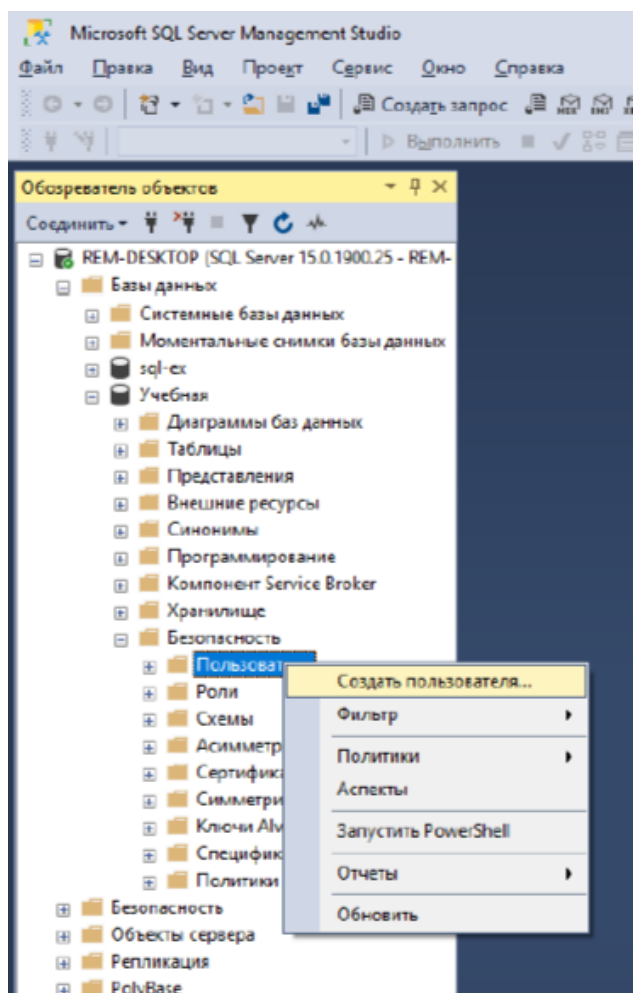
### Задание 3. Создать нового пользователя

Создание пользователя в SQL Server осуществляется посредством создания учетной записи. Пользователи БД – это специальные объекты, которые создаются на уровне базы данных и используются для предоставления разрешений в базе данных (на таблицы, представления, хранимые процедуры). Логин и пользователи БД – это совершенно разные объекты.

Чтобы добавить в базу данных «Учебная» нового пользователя, надо:

1. выделить узел «\Базы данных\Учебная\Безопасность\Пользователи» и в контекстном меню выбрать пункт «Создать пользователя...»
2. В открывшемся окне в поля «Имя пользователя» и «Имя для входа» ввести имя – Student.
3. На странице «Собственные схемы» в списке «Схемы, принадлежащие данному пользователю:» установить флажок db\_owner→ОК.

Таким образом, на основе учетной записи *Student* будет создан новый пользователь для БД «Учебная» с правами владельца этой базы данных.



## ПРАКТИЧЕСКАЯ РАБОТА № 14

### Тема: Создание базы данных в среде разработки на языке SQL

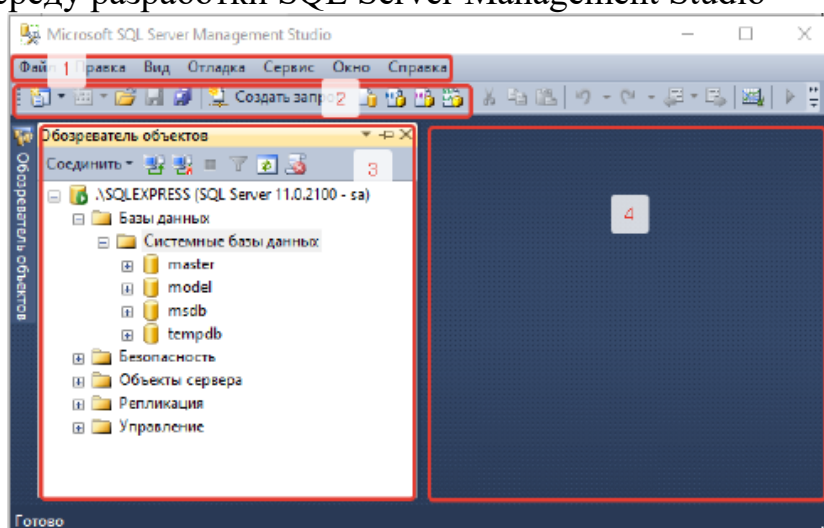
**Цели работы:** научить создавать БД в среде разработки MS SQL Server Management Studio.

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, инструкции по выполнению работы.

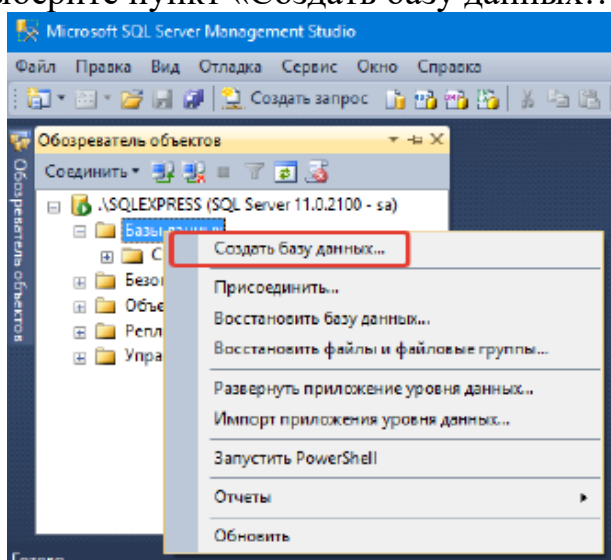
#### Содержание работы:

**Задание 1.** Используя инструмент Microsoft SQL Management Studio создать базу данных.


1. Запустить MS SQL Server Management Studio. После запуска среды разработки появится окно подключения к серверу Соединение с сервером. В этом окне необходимо нажать кнопку Соединить. После нажатия кнопки «Соединить» появится окно среду разработки SQL Server Management Studio



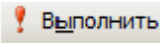
2. Теперь перейдём непосредственно к созданию файла данных. Для этого в обозревателе объектов щёлкните правой кнопкой мыши на папке Базы данных и в появившемся меню выберите пункт «Создать базу данных...».



3. Появится окно настроек параметров файла данных новой базы данных Создание базы данных. В левой части окна настроек имеется список «Выбор страницы». Этот список позволяет переключаться между группами настроек.

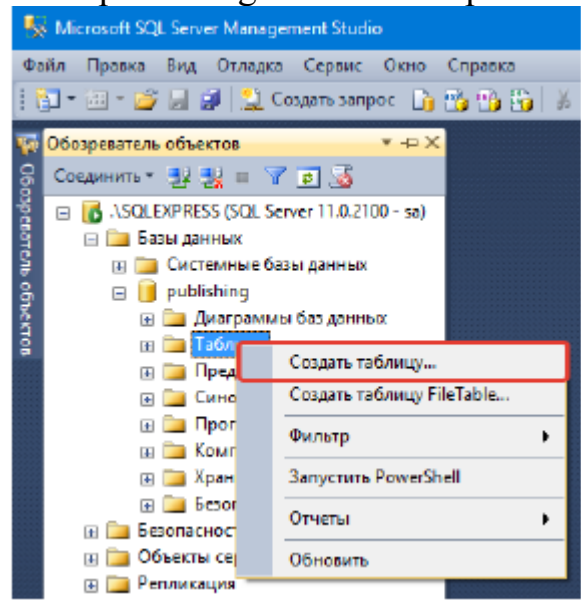
4. Новую БД можно создать, используя стандартные команды языка T-SQL. Все команды языка T-SQL набираются на вкладке нового запроса (SQLQuery). Для того чтобы создать новый запрос на панели инструментов необходимо нажать кнопку  Создать запрос. Для выполнения команд языка T-SQL на панели



инструментов необходимо нажать кнопку  или на вкладке нового запроса набрать команду GO. Для создания нового файла данных используется команда CREATE DATABASE, которая имеет следующий синтаксис:

```
CREATE DATABASE [Имя БД] ON PRIMARY
(
    NAME = <Логическое имя>,
    FILENAME = <Имя файла>,
    SIZE = <Нач.размер>,
    MAXSIZE = <Макс.размер>,
    FILEGROWTH = <Шаг> )
LOG ON
(
    NAME = <Логическое имя>,
    FILENAME = <Имя файла>,
    SIZE = <Нач.размер>,
    MAXSIZE = <Макс.размер>,
    FILEGROWTH = <Шаг> )
)
```

5. Перейдём теперь к созданию таблиц. Все таблицы нашей БД находятся в подпапке «Таблицы» папки «publishing» в окне обозревателя объектов.



6.Создадим таблицу «Сотрудники». Для этого необходимо кликнуть правой кнопкой мыши по папке «Таблицы» и в появившемся меню выбрать пункт «Создать таблицу». Появится окно создания новой таблицы

Имя столбца	Тип данных	Разрешит...
▶		<input type="checkbox"/>

В правой части окна расположена таблица определения полей новой таблицы. Данная таблица имеет следующие столбцы:

- Имя столбца должно всегда начинаться с буквы и не должно содержать различных специальных символов и знаков препинания. Если имя поля содержит пробелы, то оно автоматически заключается в квадратные скобки.
- Тип данных столбца.
- Разрешить значения Null. Если эта опция поля включена, то в случае не заполнения поля в него будет автоматически подставлено значение Null. То есть, поле необязательно для заполнения.

Под таблицей определения полей располагается таблица свойств выделенного поля «Свойства столбца». В данной таблице настраиваются свойства выделенного поля.

7.Перейдём к созданию полей и настройке их свойств. В таблице определения полей задайте значения столбцов «Имя столбца», «Тип данных» и «Разрешить значения Null»


	Имя столбца	Тип данных	Разрешить значения NULL
	E_ID	int	<input type="checkbox"/>
	E_SURNAME	nvarchar(20)	<input type="checkbox"/>
	E_NAME	nvarchar(20)	<input type="checkbox"/>
	E_FNAME	nvarchar(30)	<input checked="" type="checkbox"/>
	E_BIRTHDAY	date	<input checked="" type="checkbox"/>
	E_SEX	bit	<input checked="" type="checkbox"/>
	E_POST	int	<input type="checkbox"/>
	E_ROOM	int	<input checked="" type="checkbox"/>
	E_PHONE	nvarchar(10)	<input checked="" type="checkbox"/>
	E_INN	nvarchar(12)	<input type="checkbox"/>
	E_PASSP_NUMBER	nvarchar(12)	<input type="checkbox"/>
	E_PASSP_ORG	nvarchar(50)	<input type="checkbox"/>
	E_PASSP_DATE	date	<input type="checkbox"/>
	E_ADDRESS	nvarchar(80)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Таблица Сотрудники (Employees) имеет следующий набор столбцов:



- E\_ID –идентификатор сотрудника (первичный ключ, уникальный);
- E\_SURNAME – фамилия сотрудника (обязательное, строка 20 символов);
- E\_NAME – имя сотрудника (обязательное, строка 20 символов);
- E\_FNAME – отчество (не обязательно, строка 30 символов);
- E\_BIRTHDAY – дата рождения (дата);
- E\_SEX – пол (логический тип данных);
- E\_POST – идентификатор должности, служит для связи с таблицей «должности» (posts), в которой хранятся все должности;
- E\_ROOM – номер комнаты (составной внешний ключ, служит для связи с таблицей «Комнаты»;
- E\_TEL – номер телефона (составной внешний ключ, служит для связи с таблицей «Комнаты»;
- E\_INN – идентификационный номер налогоплательщика;
- E\_PASSP\_NUMBER – номер паспорта (строка 12 символов, обязательное);
- E\_PASSP\_ORG – кем выдан паспорт(строка, 50 символов, обязательной);
- E\_PASSP\_DATE – дата выдачи паспорта (дата, обязательное);
- E\_ADDRESS – адрес проживания (строка 80 символов).

Так как, поле E\_ID будет являться первичным полем связи, то мы должны сделать его числовым счётчиком. То есть данное поле должно автоматически заполняться числовыми значениями. Более того, оно должно быть ключевым. Для этого выделите поле, просто щёлкнув по нему мышкой в таблице определения полей. В таблице свойств столбца отобразятся свойства поля E\_ID. Разверните группу свойств Спецификация идентификатора Свойство (Идентификатор) установите в значение Да. Задайте свойства Начальное значение идентификатора и Шаг приращения идентификатора равными 1





Сжатый тип данных	bigint
<input checked="" type="checkbox"/> Спецификация вычисляемого столбца	
<input checked="" type="checkbox"/> Спецификация идентификатора	Да
(Идентификатор)	Да
Начальное значение идентификатора	1
Шаг приращения идентификатора	1
<input checked="" type="checkbox"/> Спецификация полнотекстового столбца	Нет

Эти настройки показывают, что значение поля E\_ID у первой записи в таблице будет равным 1, у второй – 2, у третьей 3 и т.д.

8.Теперь сделаем поле E\_ID ключевым полем. Выделите поле, а затем на панели инструментов нажмите кнопку с изображением ключа . В таблице определения полей, рядом с полем E\_ID появиться изображение ключа  SID, говорящее о том, что поле ключевое.

На этом настройку таблицы «Сотрудники» можно считать да данном этапе завершённой.

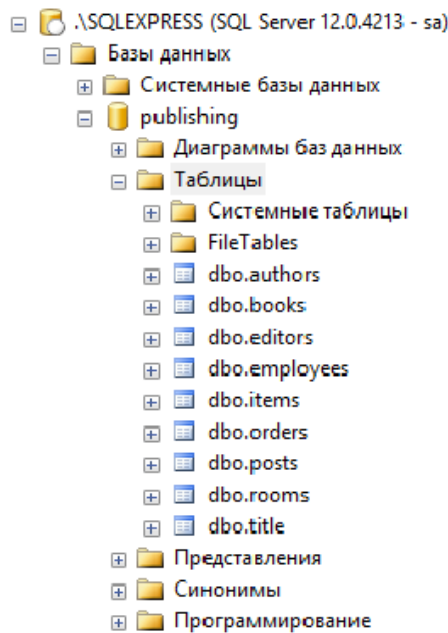
9.Закройте окно создания новой таблицы, нажав кнопку закрытия  в верхнем правом углу окна, над таблицей определения полей.

IDEA-PC\SQLEXPRES...ing - dbo.Table_1* 	
Имя столбца	Тип
E_SURNAME	nvarchar(20)
E_NAME	nvarchar(20)

Появиться окно с запросом о сохранении таблицы. В этом окне необходимо нажать Да. Появиться окно Выбор имени, предназначенное для определения имени новой таблицы. В этом окне задайте имя новой таблицы как «employees» и нажмите кнопку ОК. Таблица «employees» отобразиться в обозревателе объектов в папке Таблицы базы данных «publishing».

Выбор имени
<p>Введите имя таблицы:</p> <input type="text" value="employees"/>
<div> <div>OK</div> <div>Отмена</div> </div>

10. По аналогии создадим все таблицы



## Задание 2. Заполните информацией созданную базу данных

1. Новые данные добавляются оператором INSERT. Наименьшей единицей информации, которую можно добавить в реляционную базу данных, является одна строка таблицы. Немного упрощенный синтаксис оператора INSERT имеет вид:

```
INSERT INTO Имя_Таблицы [(Колонка [, Колонка ...])]
{VALUES(<величина> [, <величина> ...]) | <оператор SELECT>};
<величина>= {;Переменная |<константа>| <выражение>
|<функция> | udf([<величина> [, <величина>...]])
| NULL | USER}
<константа>= Число | 'Строка'
<функция>= CAST(<величина> AS <тип данных>)
UPPER(<величина>)
GEN_ID(Имя_Генератора, <величина>)
<выражение>= SQL выражение, возвращающее единичное значение
Например, I
INSERT INTO Person(Pr_ID, Pr_LastName, Pr_FirstName)
VALUES(150, 'Иванов', 'Петр');
```

2. Для удаления строк из таблицы используется оператор DELETE. Вот его упрощенный синтаксис:

```
DELETE FROM Имя_Таблицы
[WHERE <условие поиска>];
Например,
Удаление всех служащих: DELETE FROM Employee;
Удаление всех людей с номерами 150 и больше:
DELETE FROM Person WHERE Pr_ID >= 150;
```

3. Оператор UPDATE обновляет значения одного или нескольких столбцов в выбранных строках одной таблицы. Строки для обновления указываются в предложении WHERE. Если пропустить предложение WHERE, то изменятся все строки таблицы.

```
UPDATE Имя_Таблицы
SET Колонка = <величина>[, Колонка =<величина> ...]
[WHERE <условие поиска>]
```

<величина>= { Колонка | :Переменная | <константа>  
| <выражение> | <функция>  
| udf([ <величина>[, <величина>...]]) | NULL | USER }  
<выражение>= SQL выражение, возвращающее единичное значение  
<условие поиска>= как в операторе SELECT

Например,

Увеличить зарплату всем служащим на 10%:

```
UPDATE Employee SET Salary = 1.1*Salary;
```

Увеличить зарплату всем служащим, которые имеют зарплату меньше 10000 на 15%:

```
UPDATE Employee SET Salary = 1.15*Salary;
```

```
WHERE Salary <= 10000;
```

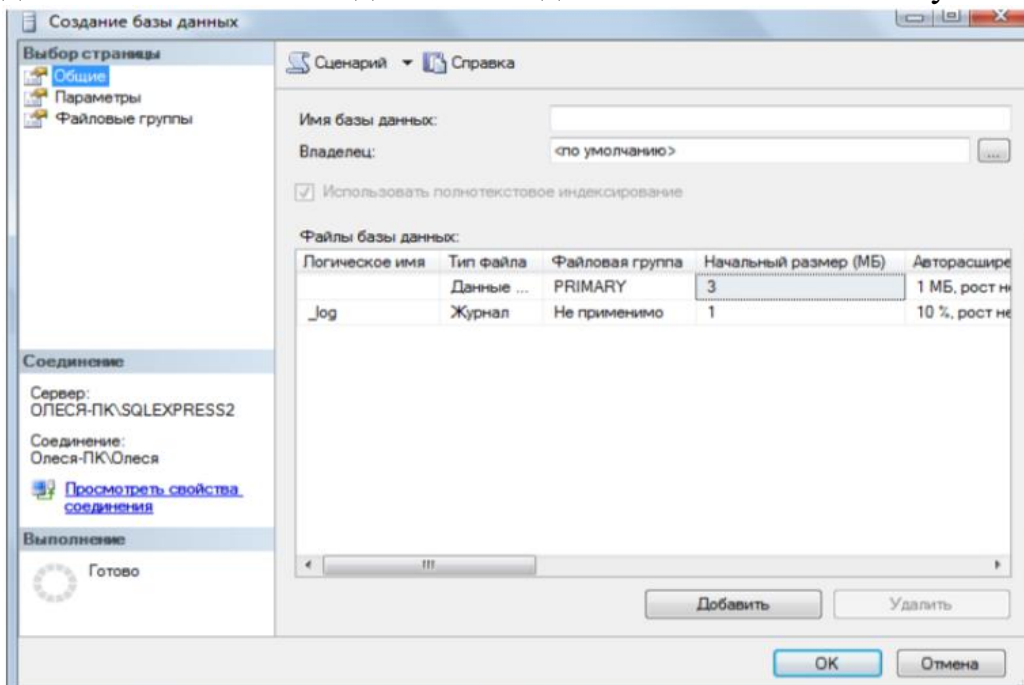
4. Вставка отдельных строк. Предложение INSERT... VALUES выполняет вставку в таблицу одной строки. Его удобно использовать для небольших операций, когда в таблицу нужно вставить несколько строк. Синтаксис этого предложения следующий:

```
INSERT INTO имя_таблицы [<имя столбца> [,имя_столбца]...]
```

```
VALUES (значение [,значение]...);
```

**Задание 3.** Создать базу данных Университет (University.mdf).

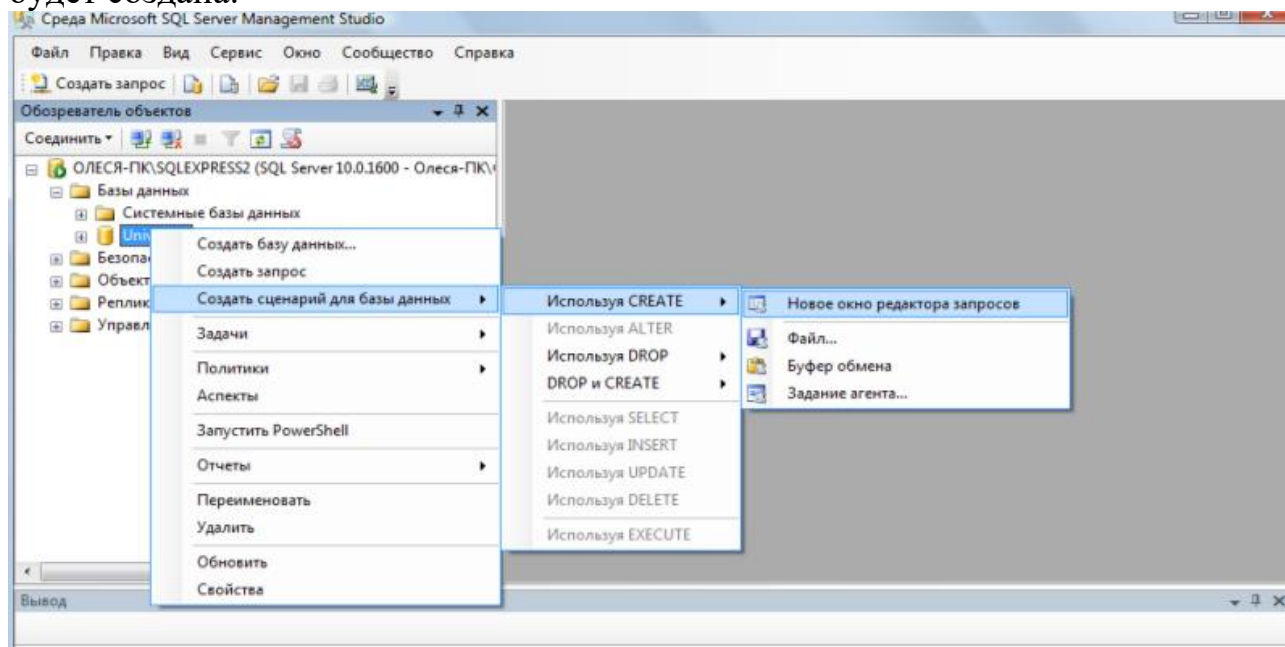
1. Выполнить команду "База данных/Создать базу данных..." в программе SQL Server Management Studio, ввести параметры создаваемой базы данных в диалоговом окне "Создание базы данных" и нажать кнопку ОК.



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.

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 рисунке (нажмите на имени базы данных University правой клавишей и из контекстного меню выберите

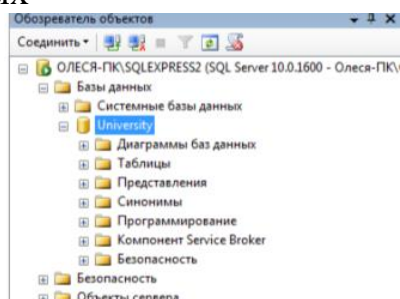
Создать скрипт как.. CREATE). Если параметры введены правильно, база данных будет создана.



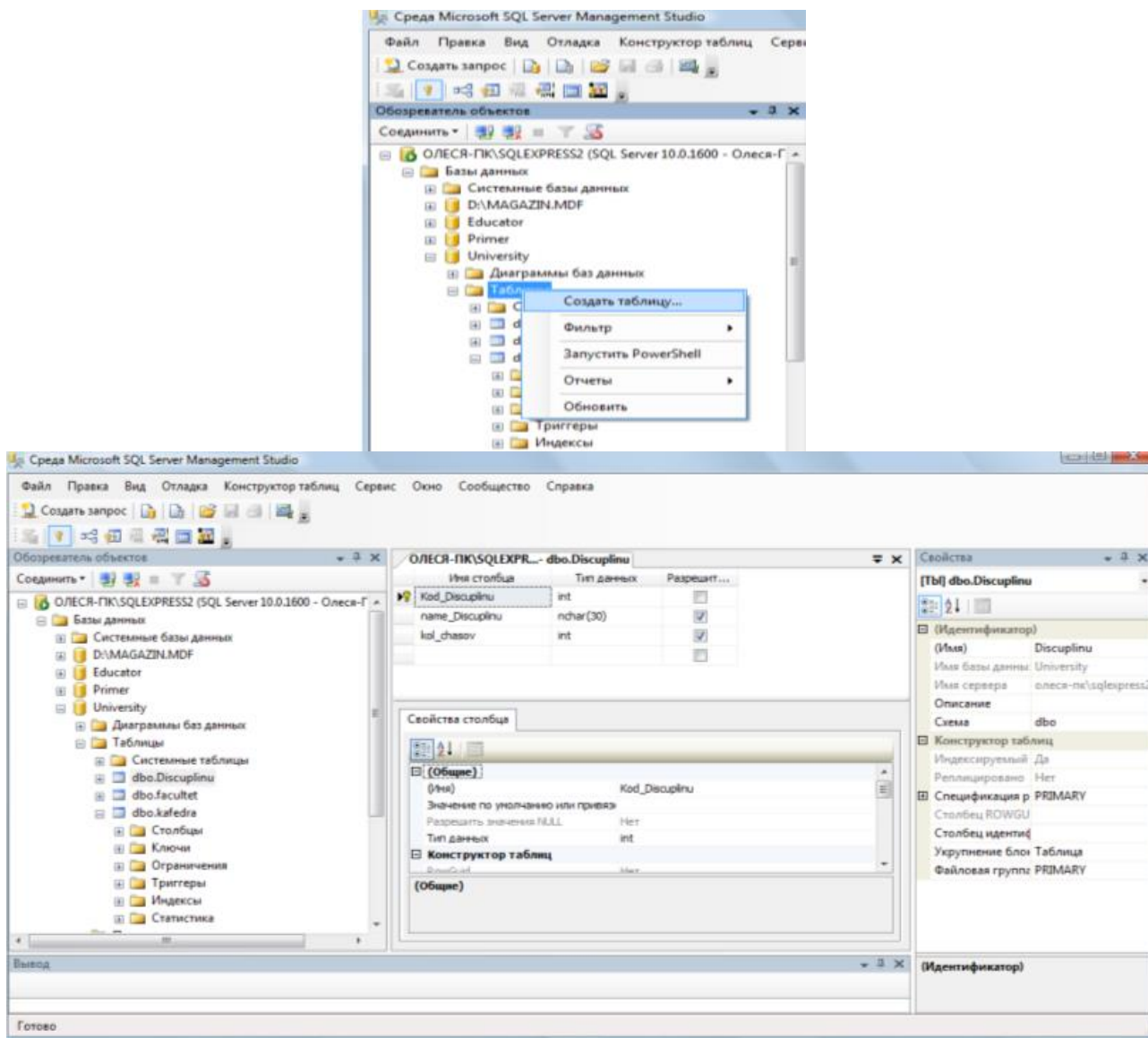
При создании базы данных возможны следующие типичные ошибки:

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 окне редактора в дерево на вкладке "Проводник" добавляется узел с зарегистрированной базой данных



2. Для создания таблицы в диалоговом режиме, нажмите в окне "Обозревателя объектов" правую клавишу мыши на узле "Tables" (Таблицы) или на одной из имеющихся таблиц и в открывшемся меню выберите команду "New Table...(Создать таблицу)". В результате откроется окно создания таблицы



Сетка в средней части окна содержит сведения о полях таблицы.

Чтобы добавить поле в таблицу, следует нажать правую клавишу и выбрать из контекстного меню Вставить столбец.

В колонке "Имя столбца" вводится имя создаваемого поля, в колонке "Тип данных" выбирается тип данных. Чтобы задать полю ограничение "NOT NULL" достаточно установить флажок в колонке "Разрешить значения NULL (пустые значения)".

Чтобы присвоить полю статус первичного ключа необходимо в колонке ПК щелкнуть правой клавишей мыши и выбрать из контекстного меню Создать первичный ключ.

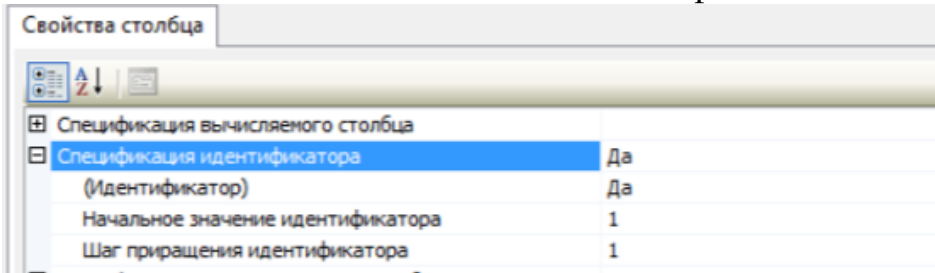
Дополнительные свойства можно настраивать с помощью окна Свойства столбца, которое находится внизу рабочей области.

3. Создайте новую таблицу, хранящую информацию о всех факультетах в университете. Присвойте ей имя – Facultet. Добавьте поля Kod\_faculteta (уникальный номер факультета, тип – целый int, первичный ключ, автоинкремент), Name\_faculteta (название факультета, тип текстовый - varchar, длина 255 символов, не допускается пустых значений), Fio\_Decana (ФИО декана факультета, тип – текстовый varchar), Nomer\_komnatu (номер комнаты деканата, тип – символьный varchar(допускается запись 134-2, где 134 – номер комнаты, а 2 – номер корпуса)), Tel\_decanata (телефон деканата, тип – длинное целое число bigint с точностью в 10

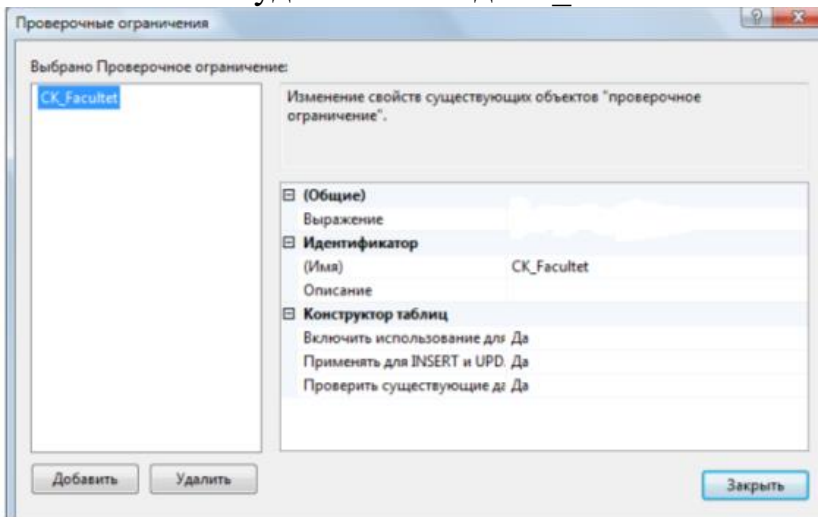
символов, значение по умолчанию ‘999999’,ограничение на значение меньше ‘1 000 000’).

*Примечание.*

1) Для того, чтобы сделать автоприращение ключевого поля kod\_faculteta измените в дополнительных свойствах столбца свойство – Спецификация идентификатора:



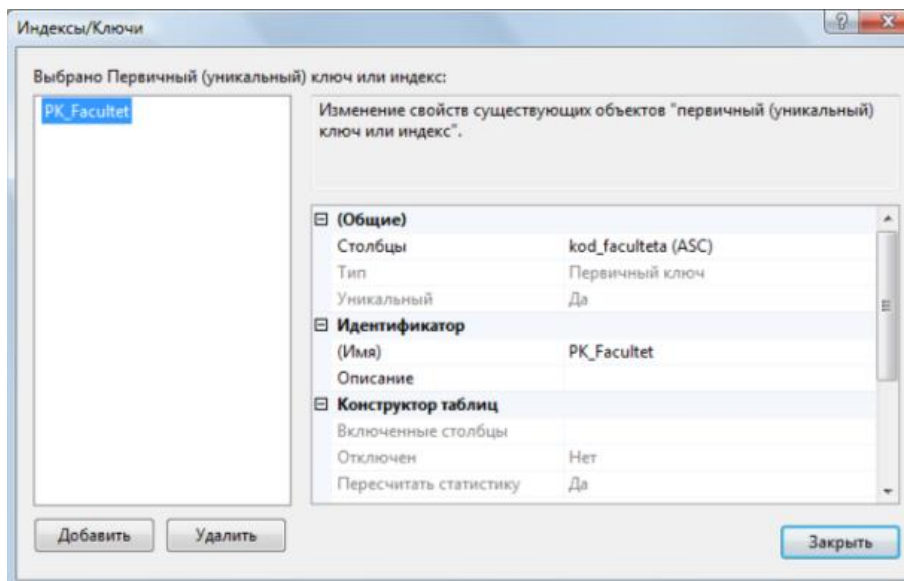
2) Для того, чтобы добавить проверочное ограничение на столбец Tel\_decanata, выделите его и правой клавишей из меню выберите Проверочные ограничения. В появившемся окне нажмите кнопку Добавить. Будет создано ограничение под именем по умолчанию, для которого необходимо создать выражение вычисления ограничения на вводимые значения выбранного поля. В нашем случае ограничение на значение меньше ‘1000000’ будет иметь вид Tel\_decanata< 1 000 000.




После введения данных о всех полях таблицы следует нажать кнопку Сохранить на панели инструментов.

4. Создадим для данной таблицы индексы (для других таблиц создавать индексы не нужно!!). Для этого выберите из контекстного меню "Индексы и ключи". В нашем случае для первичного ключа автоматически присваивается уникальный индекс по возрастанию.





Для создания индекса выполните следующие действия:



1. Нажмите в этой сетке клавишу Добавить. В результате будет вставлена новая строка.
  2. Задайте в колонке "имя" имя индекса – My\_Index\_Facultet.
  3. Нажмите кнопку  в колонке "Столбцы". В результате откроется окно с для выбора столбцов и порядка сортировки. В левом списке "Имя столбца" будут находиться поля, которые можно добавить к индексу, в правом списке "Сортировка" находится список сортировки. Для формирования перечня полей, которые будут входить в индекс, добавьте нужные поля.
  4. Если создается уникальный индекс, то изменяется свойство в колонке "Уникальный".
  5. Чтобы создать индекс нажмите кнопку сохранить.
5. Для заполнения таблиц данными вначале выделите таблицу в окне проводника и из контекстного меню выберите Изменить первые 200 строк. Вид окна представлен на рисунке

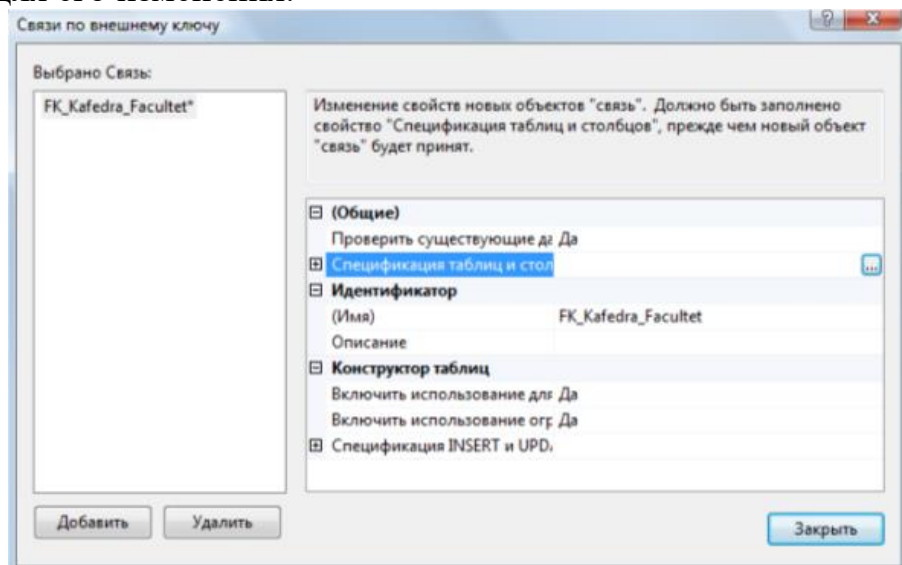
I-ПК\SQLEXP... - dbo.Facultet*		I-ПК\SQLEXP...y - dbo.Facultet			
	kod_faculteta	Name_faculteta	Fio_Decana	Nomer_komnatu	Tel_decan...
	1	Математики и информатики	... Статывка Юрий Иванович ...	31/a	417499
	2	Компьютерных систем и технологий ...	Губачева Лариса Александровна...	204/12	477051
	3	Международный	... Харченко Евгений Иванович ...	310/9	500830
►*	NULL	NULL	NULL	NULL	NULL

Введите самостоятельно три записи. Для добавления новой записи воспользуйтесь кнопкой перехода .

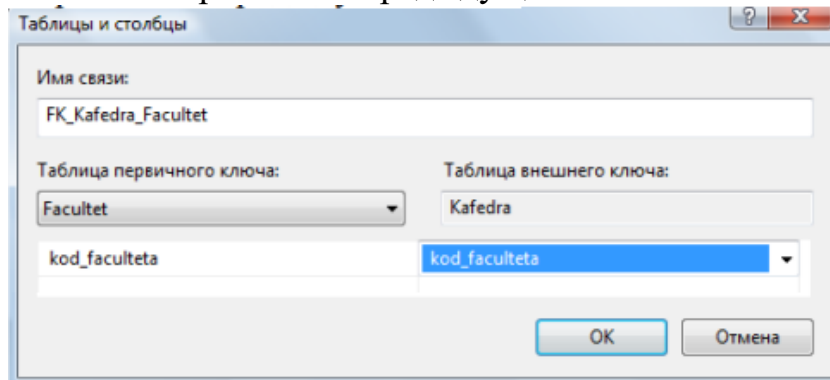
6. Создайте новую таблицу, присвойте ей имя Kafedra. Добавьте следующие поля в таблицу: Kod\_kafedru (уникальный номер кафедры, тип – целый, первичный ключ, автоинкремент), kod\_faculteta (номер факультета, к которому принадлежит кафедра, тип – целый, не допускается пустых значений), Name\_kafedru (название кафедры, тип текстовый, длина 50 символов, не допускается пустых значений), Fio\_zavkaf (ФИО заведующего кафедрой, тип – текстовый), Nomer\_komnatu (номер комнаты кафедры, тип – числовой, num\_korpusa (номер корпуса, тип – числовой, с ограничением – номер корпуса может быть, только от 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12), Tel\_kafedru (телефон кафедры, тип – текстовый).

*Примечание.* Для создания ограничений для столбца num\_korpusa используйте в выражении функцию IN ('знач1', 'знач2', ..., 'значn')

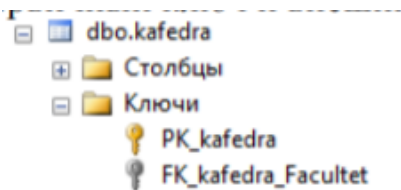
7. Создадим связь между двумя таблицами Факультет и Кафедра. Между данными таблицами можно установить связь «один-ко-многим», так как на одном факультете может быть несколько кафедр. Например, к факультету Математики и информатики относятся кафедры Информатики, Компьютерные сети и системы, Прикладной математики и Математического анализа. Чтобы создать связь «один-ко-многим» необходимо вызвать в окне редактирования таблицы Кафедра каскадное меню и выбрать Отношения  **Отношения...**. Внешний ключ будет создан к полю kod\_faculteta и оно будет связано с полем первичным ключом kod\_faculteta внешней таблицы Факультет. Для этого в появившемся окне Связи по внешнему ключу по внешнему перейдите на поле Спецификация таблиц и столбцов и нажмите кнопку напротив  для его изменения.




Вам будет открыто диалоговое окно создания Внешних ключей, где имя связи по умолчанию присвоенное внешнему ключу можете оставить без изменения. Выберите в качестве таблицы первичного ключа – Facultet и поле kod\_faculteta. Выберите в качестве таблицы внешнего ключа – Kafedra и поле kod\_faculteta. Нажмите Ок и вернитесь в предыдущее окно.




Установите правила удаления и обновления данных. Для этого перейдите на поле Спецификации INSERT и UPDATE и выберите правило каскадного удаления и обновления. Сохраните таблицу. После подтверждения операции у вас в таблице Кафедра появится внешний ключ по полю kod\_faculteta. Для этого в окне проводника выберите в таблице Кафедра элемент Ключи и вы увидите первичный ключ и внешний ключ.



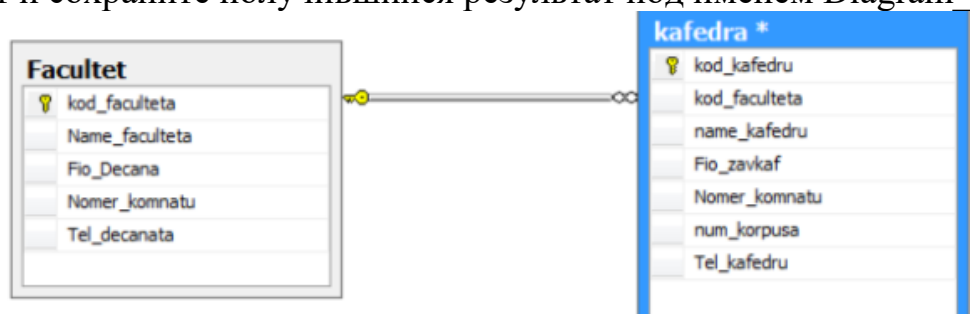


Введите самостоятельно название кафедр и их реквизиты на все созданные факультеты. Для добавления новой записи воспользуйтесь кнопкой перехода .

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	Кочевской	205	1	425262
3		1	Математического анализа	Арлинской	61	1	627272
4		1	Информатики	Пождаев	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

После ввода данных нажмите на панели кнопку о подтверждении .

Для визуального отображения связей между таблицами воспользуйтесь средством Диаграммы базы данных, добавьте две созданных вами таблицы в проект и сохраните получившийся результат под именем Diagram BD.



8. Новую таблицу под именем STUDENT (Студент) с помощью sql-операторов с полями:

STUDENT\_ID – целого типа для уникальной идентификации записей в таблице первичный ключ тип счетчик,

SUTNAME – текстового типа для обозначения имени студента,

SUTFNAME - текстового типа для обозначения фамилии,

STIPEND – действительного типа для обозначения стипендии. При этом на это поле наложено ограничение числом – величина размера стипендии должна быть меньше 500р.

KURS - целого типа для обозначения курса. При этом на это поле наложено ограничение – курс на котором может учиться студент может принимать значение от 1 до 5 ,

CITY - текстового типа для обозначения города,

BIRTHDAY – типа даты/времени для обозначения день рождения,

GROUP - текстового типа для обозначения студенческой группы,

KOD\_KAFEDRU – целого типа для обозначения названия кафедры, на которой учится студент. Поле KOD\_KAFEDRU из таблицы STUDENT и поле KOD\_KAFEDRU из таблицы KAFEDRA связаны тем, что описывают одни и те же данные, т.е. содержат идентификаторы кафедр, информация о которых содержит база данных. Более того, значение идентификаторов кафедр, которые допустимы в таблице STUDENT, должны выбираться только из списка значений поля KOD\_KAFEDRU, т.е. принадлежащих реально описанных в базе данных кафедрам. Т.е. между этими полями имеется прямая связь. Таким образом, поле KOD\_KAFEDRU из таблицы STUDENT будет являться внешним ключом.

Кроме того, при определении таблицы STUDENT запрещено использовать значение – значений для столбцов STUDENT\_ID, SUTNAME, SUTFNAME.

В качестве первичного ключа принято значение столбца STUDENT\_ID. Выполните sql-код. Обновите базу данных и просмотрите созданную таблицу. Сохраните sql-запрос под именем Студент.sql.

9. Новую таблицу под именем TEACHER (Преподаватели) с помощью sql-операторов. Эта таблица содержит информацию о преподавателях вуза. Каждый преподаватель может работать на одной кафедре, иметь множество лекционных занятий и быть куратором более чем одной группы.

Описание столбцов таблицы TEACHER

KOD\_TEACHER Уникальный идентификатор преподавателя. Является первичным ключом

KOD\_KAFEDRU Уникальный идентификатор кафедры, на которой работает преподаватель. Является внешним ключом

NAME\_TEACHER Фамилия преподавателя

INDEF\_KOD Идентификационный код. Является уникальным для преподавателя

DOLGNOST Должность, может принимать только определенные значения из списка, такие как 'профессор', 'доцент', 'старший преподаватель', 'ассистент'. Значение по умолчанию 'ассистент'.

ZVANIE Научное звание, может принимать только определенные значения из списка, такие как 'к.т.н', 'к.г.у', 'к.с.н', 'к.ф.м.н.', 'д.т.н', 'д.г.у', 'д.с.н', 'д.ф.м.н', 'нет'. Значение по умолчанию 'нет'.

SALARY ставка зарплаты . Значение по умолчанию 1000р. Зарплата должна быть больше нуля.

RISE надбавка к зарплате. Ее значение по умолчанию =0 и не может быть отрицательным числом.

DATA\_HIRE дата приема на работу. По умолчанию текущая дата.

BIRTHDAY день рождения POL пол, может принимать только определенные значения из списка, 'ж', 'Ж', 'м', 'М'

TEL\_TEACHER Телефон. Может принимать значения только в виде '[1-9][0-9]- [0-9][0-9]-[0-9][0-9]'.

В качестве первичного ключа принято значение столбца KOD\_TEACHER.

Поле KOD\_KAFEDRU из таблицы TEACHER и поле KOD\_KAFEDRU из таблицы KAFEDRA связаны тем, что описывают одни и те же данные, т.е. содержат идентификаторы кафедр, информация о которых содержит база данных. Более того, значение идентификаторов кафедр, которые допустимы в таблице TEACHER, должны выбираться только из списка значений поля KOD\_KAFEDRU, т.е. принадлежащих реально описанных в базе данных кафедрам. Т.е. между этими полями имеется прямая связь. Таким образом, поле KOD\_KAFEDRU из таблицы TEACHER будет являться внешним ключом.

Выполните sql-код. Обновите базу данных и просмотрите созданную таблицу. Сохраните sql-запрос под именем Преподаватель.sql

10. Самостоятельно заполните вручную данными таблицы Студент и Преподаватель согласно рисункам, приведенным ниже

I-ПК\SQLXP... - dbo.Facultet*						I-ПК\SQLXP...y - dbo.Facultet	
	kod_faculteta	Name_faculteta	Fio_Decana	Nomer_komnatu	Tel_decan...		
1		Математики и информатики	Статька Юрий Иванович	31/а	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		

I\COMP\SQLXP...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	Кочевской	205	1	425262
	3	1	Математического анализа	Арлюнкой	61	1	627272
	4	1	Информатики	Пождаев	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	Ульшан	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

Запрос 1. Выполнить вставку одной строки в таблицу FACULTET

Выполнить	SQLQuery18.sql - ОЛЕСЯ-ПК\...\O...))	SQLQuery17.sql - ОЛЕСЯ-ПК\...\O...))
<pre> INSERT INTO Facultet (Name_faculteta, Fio_Decana, Nomer_komnatu, Tel_decanata) VALUES ('Юридический факультет', 'Лазор Лариса Ивановна', '123/9', '658901' ); </pre>		
Сообщения		
(строк обработано: 1)		

Запрос 2. Выполнить вставку одной строки в таблицу KAFEDRA для столбцов name\_kafedru, fio\_zavkaf, kod\_faculteta с данными 'Психологии', 'Иванова', 5.

Запрос 3. Ввести в таблицу TEACHER данные (50, 10, ' Капуста Леонид Владимирович', 1271, 1271/3, 'доцент', GETDATE()-1)

Запрос 4. Например, в следующем предложении в качестве фамилии заведующего вновь вставляемой кафедры выбирается фамилия декана факультета «Компьютерных наук и технологий» .

```

INSERT INTO KAFEDRA (name_kafedru, kod_faculteta, fio_zavkaf)
VALUES ( 'Философии', 5, (SELECT fio_decana FROM FACULTET
WHERE LOWER(Name_faculteta) = 'международный'));

```

Запрос 5. Применяя SQL-команды для внесения данных в таблицу, заполните таблицу STUDENT, используя следующую таблицу

-ПК\SQLXP...ty - dbo.kafedra									
ПК\SQLEX...y - dbo.STUDENT									
добавление.sql - (									
I-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	MT-401	
3	Борис	Котовский	400	2	Ростов-на-Дону	1989-05-27	1	MT-401	
4	Петр	Комаров	300	2	Ростов-на-Дону	1989-11-18	1	MT-401	
5	Марина	... Погребняк	353	2	Ростов-на-Дону	1989-10-21	1	MT-402	
6	Иванна	... Смирнова	400	2	Ростов-на-Дону	1989-03-11	1	MT-402	
7	Роман	Сергеенко	0	2	Ростов-на-Дону	1989-07-04	1	MT-402	
8	Владимир	... Невечеров	100	3	Ростов-на-Дону	1989-03-24	2	MT-231	
9	Мая	Солонка	450	3	Ростов-на-Дону	1988-01-12	2	MT-231	
10	Герман	... Степанюга	300	3	Ростов-на-Дону	1988-11-18	2	MT-231	
11	Филипп	... Мозговой	0	1	Ростов-на-Дону	1987-01-08	3	MT-521	
12	Федор	... Филоненко	0	1	Донецк	1988-12-22	3	MT-521	
13	Наталья	... Николенко	100	1	Москва	1989-12-02	3	MT-521	
14	Егор	Осадчий	100	5	Москва	1989-12-02	4	MT-341	
15	Анастасия	... Петрова	489	5	Донецк	1989-07-22	4	MT-341	
16	Марьян	... Шичанина	360	5	Донецк	1989-05-09	4	MT-341	
17	Роман	Тараненко	300	5	Донецк	1988-02-29	4	MT-341	
18	Виктория	... Изотова	350	1	Донецк	1988-03-23	4	KT-121	
19	Валерий	... Киреев	150	1	Донецк	1987-06-08	4	KT-121	
20	Наталья	... Черткова	450	3	Ростов-на-Дону	1988-09-18	5	KT-241	
21	Андрей	... Романенко	250	3	Ростов-на-Дону	1986-11-24	5	KT-241	
22	Владимир	... Черней	480	3	Ростов-на-Дону	1986-11-11	5	KT-241	
23	Петр	Молчанов	450	3	Ростов-на-Дону	1986-11-11	6	KT-341	
24	Анастасия	... Коваленко	350	3	Ростов-на-Дону	1986-11-11	6	KT-341	
25	Андрей	... Жиркевич	250	3	Ростов-на-Дону	1986-07-08	6	KT-341	
26	Филипп	... Шевцов	450	1	Донецк	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.

Новые данные в таблицу TEACHER ввести вручную.



добавление.sql - ОЛЕСЯ-ПК\SQLXP...ty - dbo.TEACHER												
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	Игнатъева 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	Деттарева 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	22-23-96	
20	1	Ратов Денис Николаевич	1110786	ассистент	нет	1500.00	250.00	2008-05-01	1982-10-22	м	NULL	
22	2	Мальев Вячеслав Вадимович	1210786	доцент	к.т.н	3000.00	300.00	1980-05-01	1952-10-22	м	35-76-01	
23	2	Куча Владимир Иванович	12107869	доцент	к.т.н	3000.00	300.00	1980-07-21	1956-10-02	м	44-76-11	
24	2	Кочевской Андрей Александрович	12107860	доцент	к.т.н	3000.00	500.00	1999-04-11	1976-08-22	м	44-76-12	
27	3	Арлиной Юрий Моисеевич	1363528	профессор	д.ф.н	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	Статьева 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 Николаевна	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	68484900	профессор	д.т.н	6000.00	100.00	1976-01-01	1957-03-22	ж	40-03-11	
38	6	Бобровской Геннадий Александрович	61038373	доцент	к.т.н	3000.00	100.00	1988-01-01	1971-04-12	м	42-03-11	
39	6	Жученко Наталья Максимовна	110238373	ассистент	нет	1200.00	200.00	1989-01-01	1971-07-09	ж	42-03-12	
40	7	Ульшин 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	Смирнов 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	

## ПРАКТИЧЕСКАЯ РАБОТА № 15

### Тема: Взаимосвязи между отношениями БД

**Цели работы:** научить создавать взаимосвязи между отношениями БД в среде разработки MS SQL Server Management Studio.

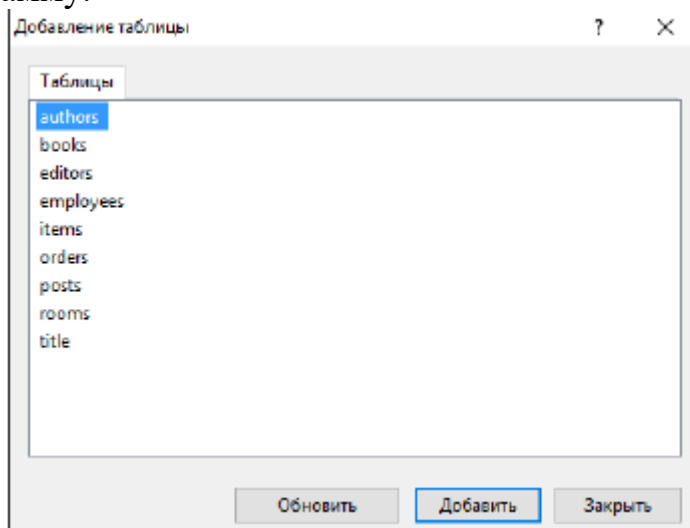
**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, инструкции по выполнению работы.

#### Содержание работы:

##### Задание 1. Связывание таблиц

- 1.Откроем базу данных, созданную в Практической работе № 13 (задание 1)
- 2.Создадим диаграмму, обеспечивающую целостность данных нашей БД «publishing».

Для создания новой диаграммы в БД «publishing» щёлкните правой кнопкой мыши по папке Диаграммы базы данных и в появившемся меню выберем пункт Новая диаграмма базы данных. 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. Обратите внимание, что на данном этапе таблицы являются не связанными, т.е. отсутствует ограничение целостности по внешнему ключу



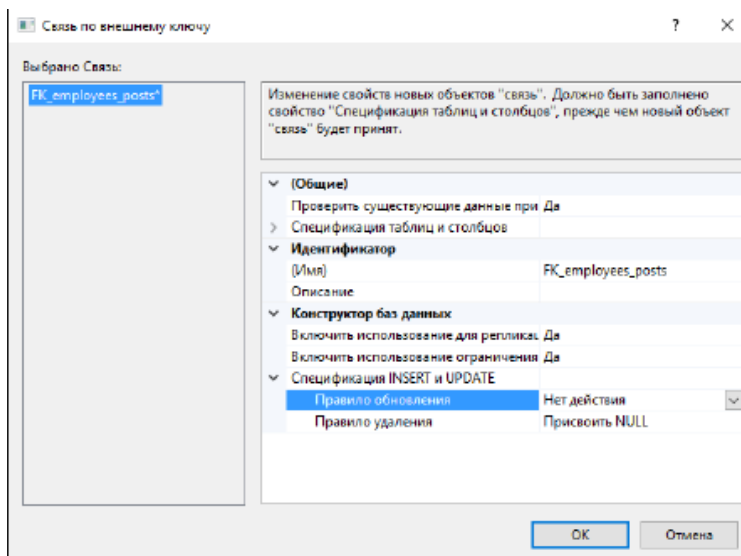
3. Теперь необходимо определить связи между таблицами. Давайте организуем связь между сотрудником и должностью. Перед этим уточним, что в случае если мы захотим удалить из БД должность, то необходимо обнулить должность у сотрудника. Для этого необходимо перетащить поле P\_ID из таблицы Posts на такое же поле в таблице Employees. Появится окно создания связи между таблицами «Таблицы и столбцы»

The screenshot shows the 'Таблицы и столбцы' (Tables and Columns) dialog box. It is configured to create a foreign key relationship between the 'posts' table and the 'employees' table. The primary key is 'P\_ID' in the 'posts' table, and the foreign key is 'E\_POST' in the 'employees' table. The relationship name is 'FK\_employees\_posts'.

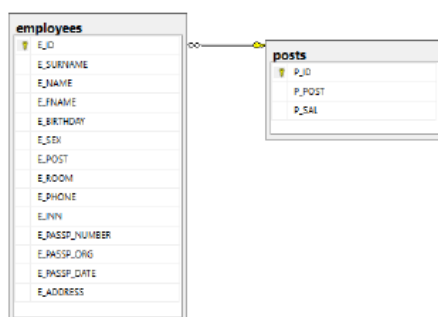
Таблица первичного ключа:	Таблица внешнего ключа:
posts	employees
P_ID	E_POST

Buttons: OK, Отмена

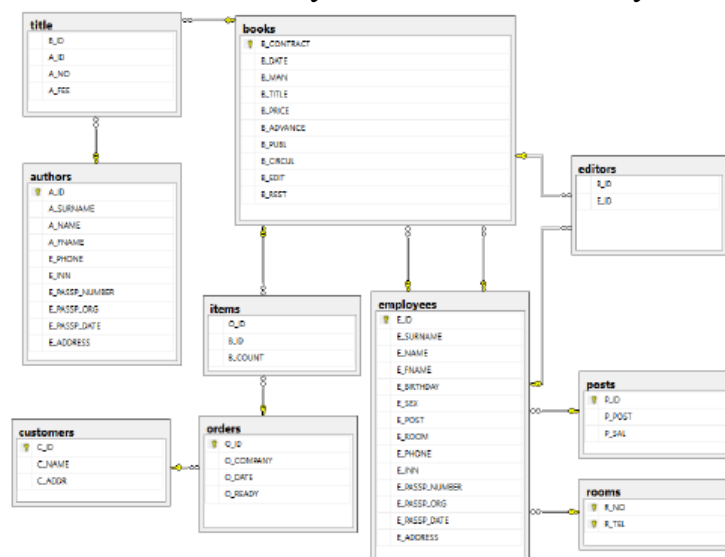
В окне создания связи нажмите кнопку ОК. Появится окно настройки свойств связи Связь по внешнему ключу



В диаграмме между таблицами employees и posts появится связь в виде линии.



4. Аналогично для всех остальных таблиц необходимо проделать те же самые действия. В итоге схема отношений будет выглядеть следующим образом.



5. Добавление данных в таблицу. Заполнение таблиц производится при помощи следующей команды:

INSERT <Имя таблицы> (<Список полей>)  
VALUES (<Значения полей>)

**Задание 2.** Создать схему взаимосвязи между таблицами к базе данных Университет, созданной в Практической работе № 14



## ПРАКТИЧЕСКАЯ РАБОТА № 16

### Тема: Организация локальной сети. Настройка локальной сети

**Цель работы:** ознакомиться с вариантами организации локальных компьютерных сетей; научиться настраивать локальную сеть.

**Оборудование:** ПК, интернет, программное обеспечение – операционная система Windows 10, MS Word, инструкции по выполнению работы

#### Справочный материал:

**Локальные компьютерные сети** – это сеть, которая объединяет компьютеры, установленные в одном помещении или в одном здании.

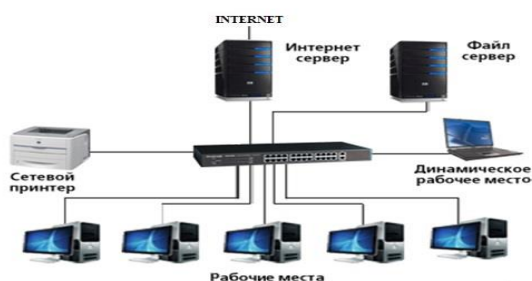
**Сетевая плата** (также известная как сетевая карта, сетевой адаптер, Ethernet-адаптер) – периферийное устройство, позволяющее компьютеру взаимодействовать с другими устройствами сети.

**Диспетчер устройств** отображает установленное на компьютере оборудование в графическом представлении. С помощью диспетчера устройств можно устанавливать и обновлять драйвера аппаратных устройств, изменять параметры этих устройств и устранять неполадки в их работе.

Существует разнообразные варианты организации локальных компьютерных сетей.

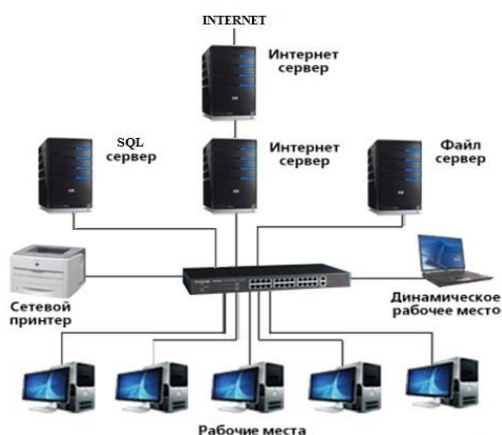
**Вариант 1.** Сеть без сервера с организацией выхода в Интернет для малых организаций от трех рабочих мест.

**Вариант 2.** Сеть с выделенными файловым и интернет-серверами для компаний среднего масштаба с числом компьютеров от 10 и более. Данная конфигурация подходит для организации сетевых ресурсов на выделенном сервере в сети, а также для организации безопасного выхода в Интернет.



**Вариант 3.** Сеть с выделенным сервером для организаций среднего масштаба с числом компьютеров от 10 и более. Данная конфигурация (рис. 3) подходит для организации сетевых ресурсов (общих документов, принтеров, баз данных) на выделенном сервере в сети.

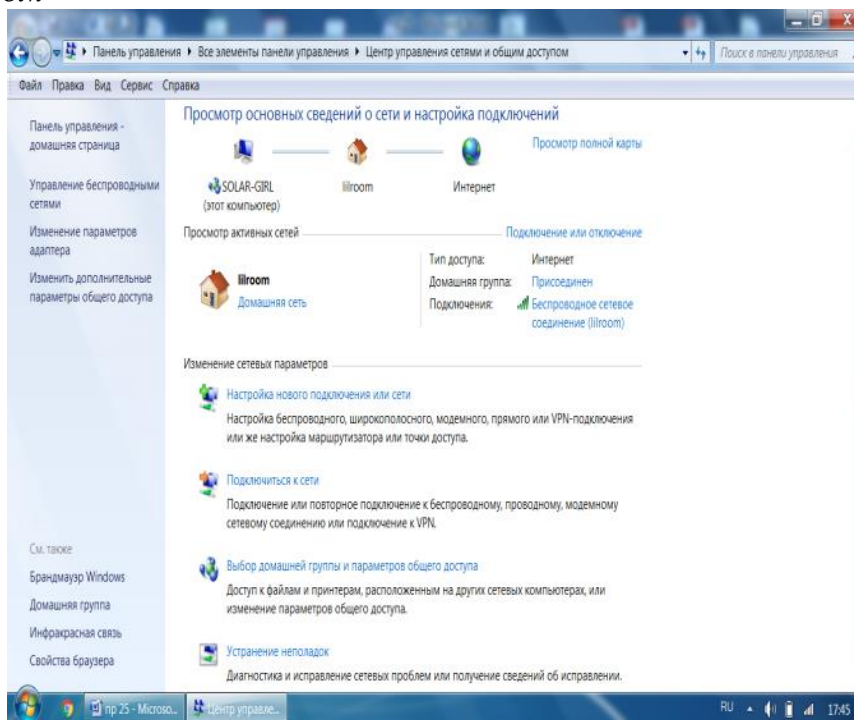
**Вариант 4.** Сеть с выделенными файловым, интернет- и SQL-серверами для средних и крупных компаний с числом компьютеров от 20 и более. Данная конфигурация (рис. 4) подходит для организации сетевых ресурсов на выделенном сервере в сети. Имеется возможность «жесткого» контроля доступа к интернет-ресурсам. SQL-сервер позволяет ускорить работу объемных баз данных (например, 1С).



#### Порядок выполнения работы:

**Задание 1.** Определите вариант организации локальной сети в компьютерном классе.

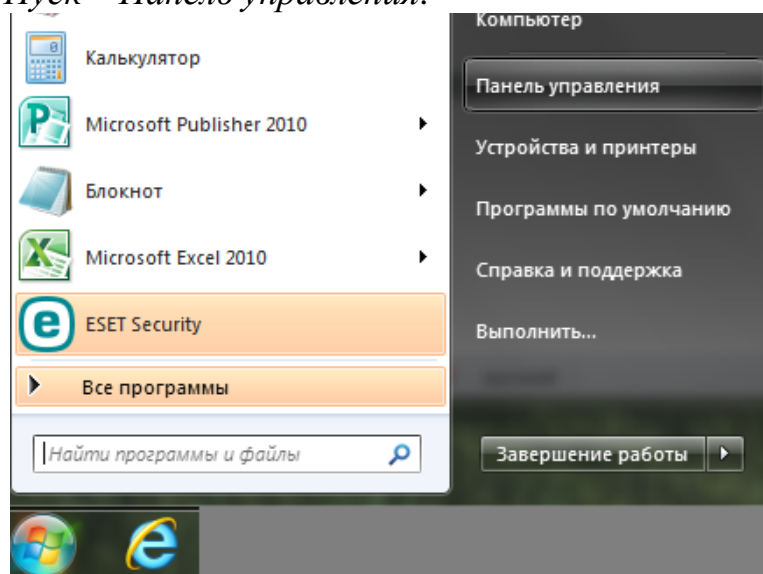
Выполните команду *Пуск - Панель управления - Центр управления сетями и общим доступом*



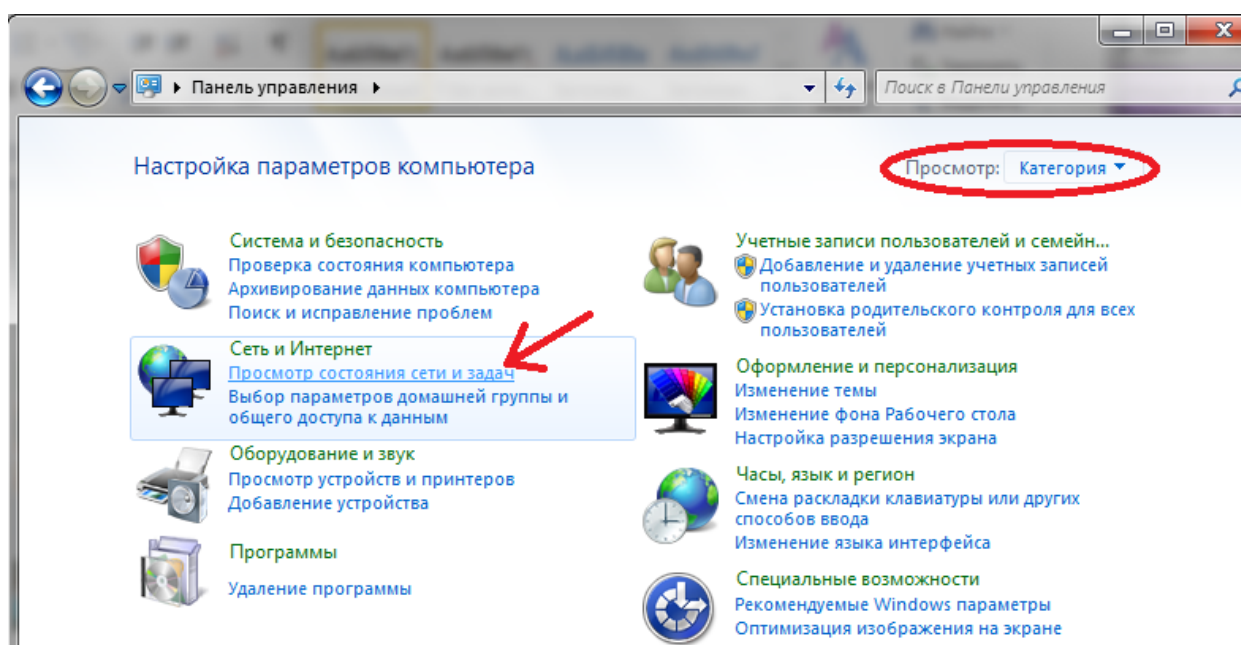
Продемонстрируйте преподавателю результат работы.

## Задание 2. Настроить локальную сеть.

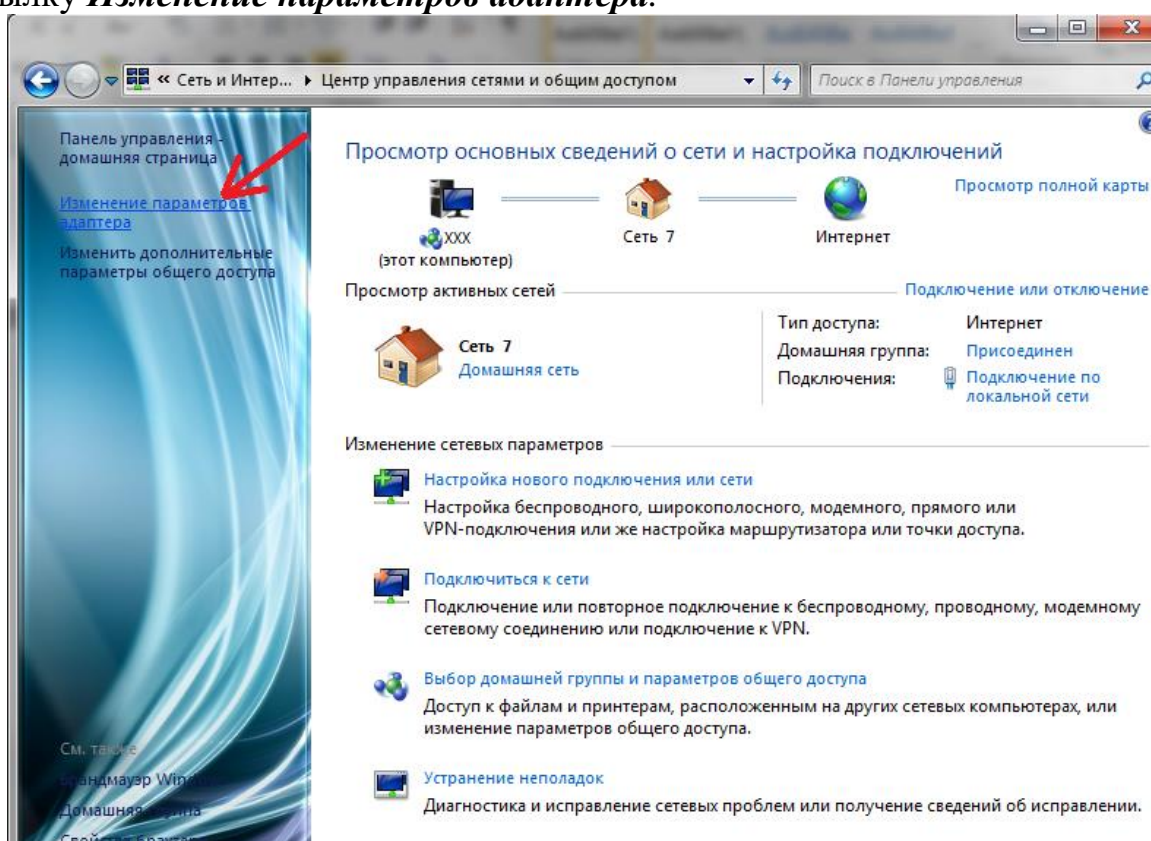
Для настройки статического IP-адреса на компьютерах необходимо выполнить команду *Пуск – Панель управления*.



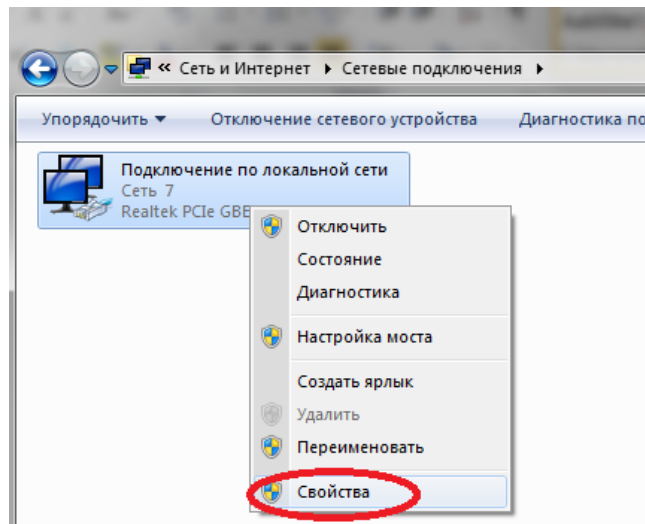
В разделе *Сеть и Интернет* нажать на ссылку *Просмотр состояния сети и задач*. **Примечание.** Если в Панели управления отображается список значков, нажмите на раскрывающееся меню *Просмотр:* и выберите параметр *Категория*.



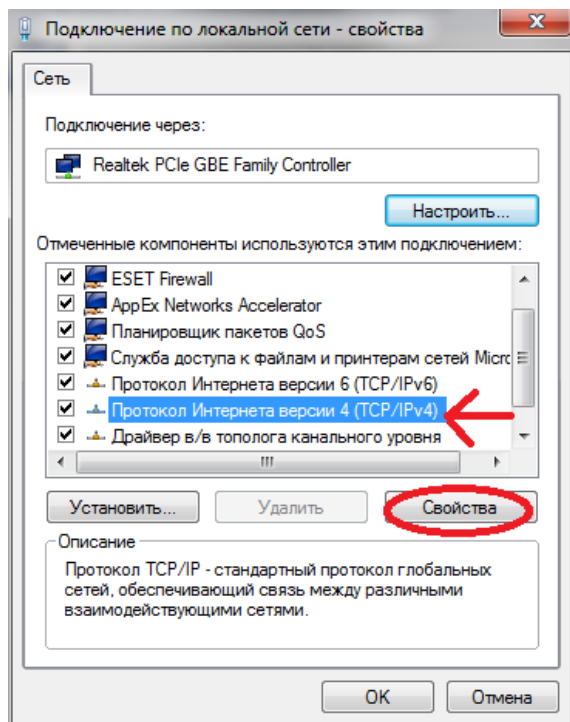
В левой части окна «Центр управления сетями и общим доступом» нажмите на ссылку ***Изменение параметров адаптера***.



В окне «Сетевые подключения» отображаются доступные интерфейсы ПК. Нажмите правой кнопкой мыши на значок *Подключение по локальной сети* и выберите пункт *Свойства*.

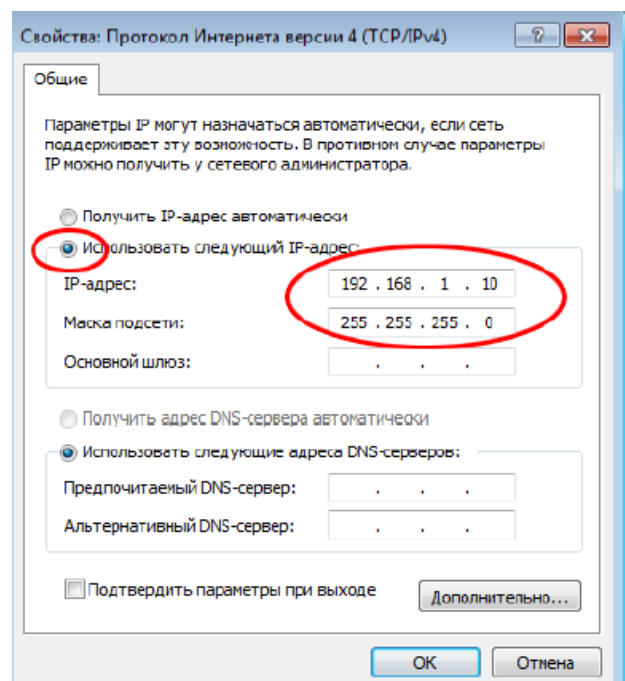


Выберите опцию *Протокол Интернета версии 4 (TCP/IPv4)* и нажмите кнопку *Свойства*



Чтобы настроить IP-адрес, маску подсети и шлюз по умолчанию вручную, установите переключатель *Использовать следующий IP-адрес*.

Указав все данные IP, нажмите кнопку *ОК*. Нажмите кнопку *ОК* в окне «Свойства подключения по локальной сети», чтобы присвоить IP-адрес адаптеру локальной сети.



### **Задание 3. Работа в локальной сети с БД.**

Выполнив команду *Пуск – Компьютер – Сеть*, можно работать с доступными дисками других ПК в локальной сети.



Так отображаются общедоступные папки. Чтобы предоставить доступ к папке для использования в локальной сети, нужно выбрать ее, правой кнопкой мыши открыть контекстное меню и выбрать *Свойства*, затем *Безопасность* и определить доступ.

Зайдите на ПК учителя и откройте свои созданные БД.

#### **Контрольные вопросы:**

- 1.Что такое локальная компьютерная сеть?
- 2.Для чего предназначен диспетчер устройств?
- 3.Какие варианты организации локальных компьютерных сетей вы знаете?

Опишите их.

- 4.Как можно диагностировать и получить сведения о работоспособности сети?
- 5.Как организовать общий и ограниченный доступ к БД?

## ПРАКТИЧЕСКАЯ РАБОТА № 17

### Тема: Обработка данных БД в модели «Клиент-Сервер» с использованием простых SQL запросов

**Цель работы:** изучить основы работы с БД в архитектуре «клиент-сервер» с использованием SQL, научиться использовать запросы для работы с данными.

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, инструкции по выполнению работы.

#### Справочный материал:

подавляющее большинство пользователей используют SQL для организации выборки данных. Для выборки данных из БД используется запрос SELECT. Он позволяет фильтровать выбранные данные и преобразовать их к нужному виду. Результатом выполнения запроса SELECT является другая таблица, к которой снова может быть применен запрос SELECT.

Полный синтаксис инструкции SELECT сложен, однако основные предложения можно вкратце описать следующим образом:

```
[ WITH { [ XMLNAMESPACES , ] [ <common_table_expression> ] } ]  
SELECT select_list [ INTO new_table ]  
[ FROM table_source ]  
[ WHERE search_condition ]  
[ GROUP BY group_by_expression ]  
[ HAVING search_condition ]  
[ ORDER BY order_expression [ ASC | DESC ] ]
```

Обработка элементов запроса SELECT выполняется в следующей последовательности:

1. FROM – определяет имена используемых таблиц;
2. WHERE – фильтрует строки таблицы в соответствии с заданными условиями;
3. GROUP BY – группирует строки, имеющие одинаковые значения в указанном столбце;
4. HAVING – фильтрует группы строк в соответствии с указанным условием;
5. SELECT – форматирует выходные данные;
6. ORDER BY – сортирует результаты выполнения запроса.

Порядок предложений в запросе SELECT не может быть изменен. Предложения SELECT и FROM являются обязательными, присутствие остальных зависит от контекста.

В предложении SELECT указывается список столбцов, которые должны быть возвращены запросом. Можно указать исходные элементы или вычисляемые поля во время выполнения запроса.

Конструкция DISTINCT | ALL исключает / разрешает вывод повторяющихся строк. Конструкция ALL используется по умолчанию.

\* означает вывод всех столбцов указанной таблицы. В случае, если выборка производится из нескольких таблиц, перед символом звездочки может указываться имя таблицы.

SQL-запрос может содержать вычисляемые столбцы, значения которых могут определяться на основе значений данных, хранящихся в БД конструкции. Вычисляемому столбцу следует давать название с помощью ключевого слова AS.

Вычисляемый столбец можно создать как: <Новое поле> = <выражение>



Если название столбца состоит из нескольких слов, разделенных пробелами, следует их записать в квадратных скобках: [].

Сортировка данных выполняется с помощью команды ORDER BY, которая добавляется в конец запроса, после чего перечисляется список столбцов. Для каждого столбца указывается тип сортировки ASC | DESC (ascending – по возрастанию | descending – по убыванию). ASC – по умолчанию, можно не указывать.

Конструкция TOP <N> позволяет выбрать определенное количество строк из таблицы. Дополнительный оператор PERCENT позволяет выбрать процентное количество строк из таблицы. Дополнительный оператор WITH TIES позволяет выбрать все строки с такими же свойствами.

Конструкция OFFSET <N> ROWS указывает число строк, которые необходимо пропустить, прежде чем будет начат возврат строк из выражения запроса.

Конструкция FETCH NEXT <N> ROWS ONLY указывает число строк, возвращаемых после обработки предложения OFFSET.

**Содержание работы:**

**Задание 1.** Дана таблица Академики

ФИО	Дата_рождения	Специализация	Год_присвоения_звания
Аничков Николай Николаевич	1885-11-03	медицина	1939
Бартольд Василий Владимирович	1869-11-15	историк	1913
Белопольский Аристарх Аполлонович	1854-07-13	астрофизик	1903
Бородин Иван Парфеньевич	1847-01-30	ботаник	1902
Вальден Павел Иванович	1863-07-26	химик-технолог	1910
Вернадский Владимир Иванович	1863-03-12	геохимик	1908
Виноградов Павел Гаврилович	1854-11-30	историк	1914
Ипатьев Владимир Николаевич	1867-11-21	химик	1916
Истрин Василий Михайлович	1865-02-22	филолог	1907
Карпинский Александр Петрович	1847-01-07	геолог	1889
Коковцов Павел Константинович	1861-07-01	историк	1906
Курнаков Николай Семёнович	1860-12-06	химик	1913
Марр Николай Яковлевич	1865-01-06	лингвист	1912
Насонов Николай Викторович	1855-02-26	зоолог	1906
Ольденбург Сергей Фёдорович	1863-09-26	историк	1903
Павлов Иван Петрович	1849-09-26	физиолог	1907
Перетц Владимир Николаевич	1870-01-31	филолог	1914
Соболевский Алексей Иванович	1857-01-07	лингвист	1900
Стеклов Владимир Андреевич	1864-01-09	математик	1912

1. Вывести список академиков:  
SELECT \* FROM Академики
2. Вывести ФИО и дату рождения всех академиков:  
SELECT ФИО, Дата\_рождения FROM Академики
3. Создайте вычисляемое поле «Информация», содержащее информацию об академике в таком виде: «Академик Петров Петр Петрович, специализация: математика»:  
SELECT 'Академик ' + ФИО + ', специализация: ' + Специализация AS Информация

FROM Академики

4. Вывести ФИО академиков и номер следующего года после присвоения звания:

SELECT ФИО , [Через год] = Год\_присвоения\_звания + 1 FROM Академики

5. Выведите список специализаций, убрав дубликаты:

SELECT DISTINCT Специализация FROM Академики

6. Вывести список академиков, отсортированный по возрастанию года присвоения звания:

SELECT \* FROM Академики ORDER BY Год\_присвоения\_звания

7. Вывести список академиков, отсортированный в обратном алфавитном порядке по полю «Специализация» и в алфавитном порядке по полю «ФИО»:

SELECT \* FROM Академики ORDER BY Специализация DESC ,ФИО ASC

8. Вывести первые две строки из списка академиков, отсортированного в алфавитном порядке по полю «ФИО»:

SELECT TOP 2 \* FROM Академики ORDER BY ФИО ASC

9. Вывести первые 30% строк из списка академиков, отсортированного по возрастанию года присвоения звания:

SELECT TOP 30 PERCENT \* FROM Академики ORDER BY Год\_присвоения\_звания

10. Вывести из таблицы «Академики», отсортированной по возрастанию года присвоения звания, список академиков, у которых год присвоения звания – один из первых четырех в отсортированной таблице:

SELECT TOP 4 WITH TIES \* FROM Академики ORDER BY

Год\_присвоения\_звания

11. Вывести, начиная с третьего, список академиков, отсортированный в алфавитном порядке ФИО:

SELECT \* FROM Академики ORDER BY ФИО OFFSET 2 ROWS

12. Вывести, начиная с третьего и до десятого, список академиков, отсортированный в алфавитном порядке ФИО:

SELECT \* FROM Академики ORDER BY ФИО OFFSET 2 ROWS FETCH NEXT 8 ROWS ONLY

## **Задание 2. Выполнить задания**

1. Вывести ФИО, специализацию и дату рождения всех академиков.

2. Создать вычисляемое поле «О присвоении звания», которое содержит информацию об академике в виде: «Петров Петр Петрович получил звание в 1974».

3. Вывести ФИО академиков и вычисляемое поле «Через 5 лет после присвоения звания».

4. Вывести список годов присвоения званий, убрав дубликаты.

5. Вывести список академиков, отсортированный по убыванию даты рождения.

6. Вывести список академиков, отсортированный в обратном алфавитном порядке специализаций, по убыванию года присвоения звания, и в алфавитном порядке ФИО.

7. Вывести первую строку из списка академиков, отсортированного в обратном алфавитном порядке ФИО.

8. Вывести фамилию академика, который раньше всех получил звание.

9. Вывести первые 10% строк из списка академиков, отсортированного в алфавитном порядке ФИО.



10. Вывести из таблицы «Академики», отсортированной по возрастанию года присвоения звания, список академиков, у которых год присвоения звания – один из первых пяти в отсортированной таблице.
11. Вывести, начиная с десятого, список академиков, отсортированный по возрастанию даты рождения.
12. Вывести девятую и десятую строку из списка академиков, отсортированного в алфавитном порядке ФИО.

## ПРАКТИЧЕСКАЯ РАБОТА № 18

### Тема: Обработка данных БД в модели «Клиент-Сервер» с использованием простых SQL запросов

**Цель работы:** изучить основы работы с БД в архитектуре «клиент-сервер» с использованием SQL, научиться использовать запросы для работы с данными.

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, инструкции по выполнению работы.

#### Справочный материал:

Ниже приведен почти полный синтаксис оператора SELECT

SELECT [DISTINCT | ALL]

{\* | <величина> [, <величина> ...]}

[INTO :Переменная [, :Переменная ...]]

FROM <tableref> [, <tabkeref>...]

[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>...]

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

### Содержание работы:

#### Задание 1. Выполнить запросы к БД Университет

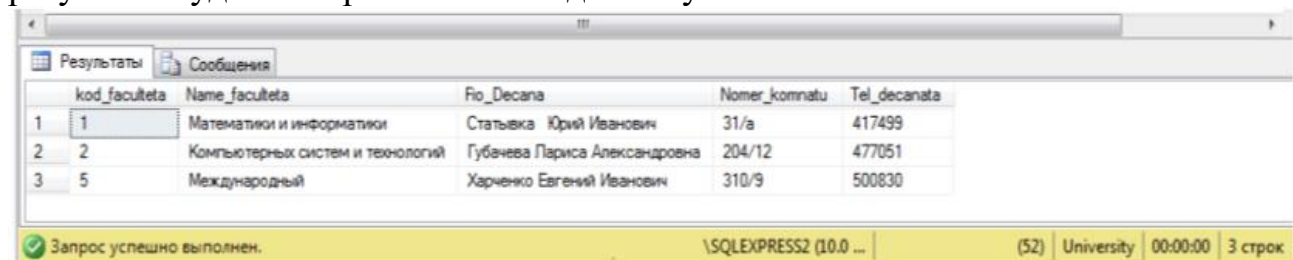
Для выполнения запросов SELECT в программе SQL Server Managment Studio необходимо выполнить следующие действия:

1. Подключиться к базе данных и выполнить команду Создать запрос. В результате откроется окно Конструктора запросов. Ввести текст запроса:



2. Нажать на панели инструментов кнопку [Выполнить] .

3. Если запрос правильный, то в результате произойдет его выполнение и результат будет отображен на вкладке Результаты



4. Количество извлеченных в результате выполнения запроса строк отображается над сеткой с данными справа. На рисунке там содержится строка «3 строк». В данном 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».

## Задание 2. Создание 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).

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 и форматируется 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.


Создайте новый запрос, введите sql-запрос, выполните его, сохраните его в рабочую папку под именем 1.sql.

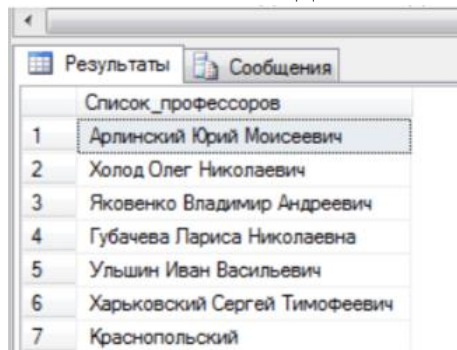
*Запрос 1.* Вывести фамилии профессоров.

```
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,



Список_профессоров	
1	Арлинский Юрий Моисеевич
2	Холод Олег Николаевич
3	Яковенко Владимир Андреевич
4	Губачева Париса Николаевна
5	Ульшин Иван Васильевич
6	Харьковский Сергей Тимофеевич
7	Краснопольский

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 его как Список профессоров.

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'
```

*Запрос 2.* Найти всех студентов с стипендией, превышающим 300 р.

В sql-редакторе создайте новый запрос и введите следующий код:

```
SELECT SUTNAME, SUTFNAME  
FROM STUDENT  
WHERE STIPEND > 300;
```

Выполните его.

*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, DOLGNOST  
FROM TEACHER  
WHERE UPPER(NAME_TEACHER) > 'Сычева';
```

*Запрос 5.* Вывести фамилии преподавателей, у которых надбавка меньше ставки в 2,5 и более раз.

**Задание 3.** Создать запросы, используя логические операторы. Стандартными логическими операторами являются AND, OR и NOT.

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

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

*Запрос 7.* Вывести фамилии преподавателей, которые являются профессорами и ставка которых превышает 4500.

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

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

```
SELECT NAME_KAFEDRU, NUM_KORPUSA  
FROM KAFEDRA  
WHERE NUM_KORPUSA = 1 OR NUM_KORPUSA = 8;
```

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

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

**Задание 4.** Создать запросы, используя комбинированные логические операторы.

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.

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

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

*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_TEACHER
FROM TEACHER
WHERE (Salary + Rise) / 2 > 5 * Rise;
```

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

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

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

*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);
```

*Запрос 17.* Вывести фамилии преподавателей со ставкой в диапазоне 1000-2000.

*Запрос 18.* Вывести фамилии преподавателей, начинающиеся на буквы от 'З' до 'Л'.

*Запрос 19.* Показать фамилии преподавателей, принятых на работу между 01.01.2000 и 12.12.2001.

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

### **Задание 5.** Создание запроса с использованием 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' (но не 'лес', 'листок', 'плес').

*Запрос 20.* Найти фамилии преподавателей на букву 'М'.

```
SELECT NAME_TEACHER  
FROM TEACHER  
WHERE UPPER(NAME_TEACHER) LIKE 'M%';
```

*Запрос 21.* Указать преподавателей, в фамилиях которых первой буквой является 'М', а четвертой — 'ы'.

## ПРАКТИЧЕСКАЯ РАБОТА № 19

### Тема: Обработка данных БД в модели «Клиент-Сервер» с использованием многотабличных SQL запросов

**Цель работы:** изучить основы работы с БД в архитектуре «клиент-сервер» с использованием SQL, научиться использовать многотабличные запросы для работы с данными.

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, инструкции по выполнению работы.

#### **Справочный материал:**

Одна из наиболее важных особенностей предложения SELECT — это способность использования связей между различными таблицами, а также вывода содержащейся в них информации. Операция, которая приводит к соединению из двух таблиц всех пар строк, для которых выполняется заданное условие, называется соединением таблиц. Для того чтобы указать соединяемые таблицы, их следует перечислить через запятую во фразе FROM.

Соединение таблиц - это частный случай операции декартового произведения (или просто произведения). Декартово произведение двух таблиц — это таблица, состоящая из всех возможных пар строк обеих таблиц. Это определение можно естественным образом расширить на любое количество таблиц. В SQL декартово произведение выражается указанием имен перемножаемых таблиц во фразе FROM и указанием всех их столбцов во фразе SELECT.

Так, произведение таблиц FACULTET и KAFEDRA выражается следующим образом:

```
SELECT *  
FROM FACULTET, KAFEDRA
```

В произведении может участвовать много таблиц. Например, произведение таблиц факультетов, кафедр и преподавателей записывается следующим образом:

```
SELECT * FROM FACULTET, KAFEDRA, TEACHER
```

Соединение таблиц может быть указано во фразе WHERE или во фразе FROM. Сначала рассмотрим первый вариант. Большинство запросов, имеющих несколько таблиц во фразе FROM, содержат фразу WHERE, в которой указаны условия, попарно сравнивающие столбцы из различных таблиц. Такое условие называется условием соединения. В этом случае SQL предполагает сцепление только тех пар строк из разных таблиц, для которых условие соединения принимает истинное значение. Теоретически при соединении сначала выполняется декартово произведение указанных таблиц в одну, а затем из нее отбираются строки согласно условию соединения. Естественно, ни одна СУБД не работает таким образом.

Фраза WHERE помимо условия соединения может также содержать другие условия, каждое из которых ссылается на столбцы соединенной таблицы. Эти условия производят отбор строк соединенной таблицы.

Соединения можно разделить на следующие категории.

- Внутренние соединения (типичные операции соединения, использующие такие операторы сравнения, как = или <>). Они включают эквивалентные соединения и естественные соединения. Внутренние соединения используют оператор сравнения для установки соответствия строк из двух таблиц на основе значений общих столбцов в каждой таблице. Примером может быть получение



всех строк, в которых идентификационный номер студента одинаковый как в таблице students, так и в таблице courses.

- **Внешние соединения.** Внешние соединения бывают левыми, правыми и полными. Если внешние соединения задаются в предложении FROM, они указываются с одним из следующих наборов ключевых слов.

- **LEFT JOIN или LEFT OUTER JOIN** Результирующий набор левого внешнего соединения включает все строки из левой таблицы, заданной в предложении LEFT OUTER, а не только те, в которых соединяемые столбцы соответствуют друг другу. Если строка в левой таблице не имеет совпадающей строки в правой таблице, результирующий набор строк содержит значения NULL для всех столбцов списка выбора из правой таблицы.

- **RIGHT JOIN или RIGHT OUTER JOIN** Правое внешнее соединение является обратным для левого внешнего соединения. Возвращаются все строки правой таблицы. Для левой таблицы возвращаются значения NULL каждый раз, когда строка правой таблицы не имеет совпадающей строки в левой таблице.

- **FULL JOIN или FULL OUTER JOIN** Полное внешнее соединение возвращает все строки из правой и левой таблицы. Каждый раз, когда строка не имеет соответствия в другой таблице, столбцы списка выбора другой таблицы содержат значения NULL. Если между таблицами имеется соответствие, вся строка результирующего набора содержит значения данных из базовых таблиц.

- **Перекрестные с соединения** Перекрестное соединение возвращает все строки из левой таблицы. Каждая строка из левой таблицы соединяется со всеми строками из правой таблицы. Перекрестные соединения называются также декартовым произведением. Таблицы или представления в предложении FROM могут указываться в любом порядке с внутренним соединением или полным внешним соединением. Однако важен порядок таблиц или представлений, заданных при использовании левого или правого внешнего соединения.

Если таблицы соединяются по равенству значений пары столбцов (группы столбцов) из различных таблиц, такая операция называется соединением таблиц по равенству. Соединение по равенству, в отличие от декартового произведения, позволяет соединить только те пары строк, которые действительно взаимосвязаны друг с другом. Так, например, мы можем соединить таблицы факультетов и кафедр по условию `FACULTET.Kod_faculteta = KAFEDRA.Kod_faculteta`. В таком варианте мы соединяем таблицы осмысленно, так как каждая строка таблицы FACULTET соединяется только со строками соответствующих кафедр. На базе таблиц FACULTET и KAFEDRA мы получаем таблицу со столбцами из обеих таблиц, имеющую строки с понятным смыслом. Можно также сказать, что в таблицу KAFEDRA вместо столбца Kod\_faculteta мы вставляем все характеристики (столбцы) соответствующего факультета из таблицы FACULTET.

### **Содержание работы:**

**Задание 1.** Создать запрос на соединение таблиц в БД Университет

```
SELECT FACULTET.Name_faculteta, FACULTET. Kod_faculteta, KAFEDRA.  
Kod_faculteta, KAFEDRA.Name_Kafedru  
FROM FACULTET, KAFEDRA;
```

Каждая строка таблицы факультетов оказалась соединенной с каждой строкой таблицы кафедр, в результате получилось 27 строк (3 факультета x 9 кафедр = 27 комбинаций).

**Задание 2.** Вывести названия кафедр и номера их групп.

```
SELECT Name_Kafedru, [Group]
FROM KAFEDRA, STUDENT
WHERE KAFEDRA.kod_kafedru = STUDENT.kod_kafedru;
```

или

```
SELECT Name_Kafedru, student.[GROUP]
FROM KAFEDRA, STUDENT
WHERE KAFEDRA.kod_kafedru = STUDENT.kod_kafedru;
```

**Задание 3.** Вывести названия факультетов и их кафедр.

```
SELECT FACULTET.NAME_FACULTETA, KAFEDRA.Name_Kafedru
FROM FACULTET, KAFEDRA
WHERE FACULTET.Kod_faculteta = KAFEDRA.Kod_faculteta;
```

**Задание 4.** Вывести названия кафедр факультета Математики и информатики.

```
SELECT KAFEDRA.Name_Kafedru AS 'Кафедры факультета математики и
информатики'
FROM FACULTET, KAFEDRA
WHERE FACULTET. Kod_faculteta = KAFEDRA. Kod_faculteta AND
LOWER(FACULTET.NAME_FACULTETA) = 'математики и информатики';
```

**Задание 5.** Вывести фамилии доцентов кафедры информатики.

```
SELECT TEACHER.NAME_TEACHER AS 'Доценты кафедры информатики'
FROM KAFEDRA, TEACHER
WHERE KAFEDRA.kod_kafedru = TEACHER. kod_kafedru AND
LOWER(KAFEDRA.Name_Kafedru) = 'информатики' AND
LOWER(TEACHER.DOLGNOST) = 'доцент';
```

**Задание 6.** Вывести названия кафедр, на которых имеются студенты со 124 стипендией >200 р

## ПРАКТИЧЕСКАЯ РАБОТА № 20

### Тема: Обработка данных БД в модели «Клиент-Сервер» с использованием многотабличных SQL запросов

**Цель работы:** изучить основы работы с БД в архитектуре «клиент-сервер» с использованием SQL, научиться использовать многотабличные запросы для работы с данными.

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, инструкции по выполнению работы.

#### **Справочный материал:**

Общая процедура составления многотабличного запроса

1. Определить множество таблиц, необходимых для ответа на запрос. В это множество должны входить таблицы, на столбцах которых сформулированы условия, а также те, столбцы которых необходимо вывести. Это так называемые базовые таблицы запроса.

2. В структуре взаимосвязанных таблиц найти путь, соединяющий базовые таблицы. Это так называемый путь вычисления запроса. В результате вы получите перечень таблиц, необходимых для формулировки запроса. Это так называемые таблицы запроса.

3. Во фразе FROM перечислить необходимые таблицы.

4. Во фразе WHERE соединить таблицы запроса и при необходимости задать условия отбора строк в базовых таблицах запроса.

5. Во фразе SELECT перечислить выводимые столбцы.

#### **Содержание работы:**

**Задание 1.** Создать запросы по трем и более полей.

Вывести названия тех кафедр факультета математики и информатики, на которых работают профессора.

```
SELECT DISTINCT KAFEDRA.Name_Kafedru  
FROM FACULTET, KAFEDRA, TEACHER  
WHERE FACULTET.Kod_faculteta = KAFEDRA.Kod_faculteta AND  
KAFEDRA.Kod_kafedru = TEACHER.Kod_kafedru AND  
FACULTET.Name_faculteta = 'Математики и информатики' AND  
TEACHER.DOLGNOST = 'профессор';
```

Для ответа на запрос необходимы три таблицы: на таблицах факультетов и преподавателей заданы условия отбора, а из таблицы кафедр следует вывести столбец названий. Поэтому три необходимые таблицы указываются во фразе FROM, а во фразе WHERE производится их соединение по условию равенства первичного и внешнего ключей:

FACULTET. Kod\_faculteta = KAFEDRA. Kod\_faculteta - соединение таблиц факультетов и кафедр

KAFEDRA. Kod\_kafedru = TEACHER. Kod\_kafedru - соединение таблиц кафедр и преподавателей

Таблица, образуемая в результате соединений, будет иметь столько же строк, сколько имеется в таблице преподавателей (если все преподаватели работают на кафедрах). Выясним, почему это так, но сначала заметим, что результат соединения таблиц не зависит от порядка соединения. Поэтому рассмотрим случай, когда сначала мы соединяем таблицы кафедр и преподавателей, а затем результат соединяем с таблицей факультетов.

Так как между таблицами кафедр и преподавателей существует связь типа один-ко-многим, их соединение фактически означает приписывание к строке каждого преподавателя данных о его кафедрах. Количество строк этого соединения будет равным количеству преподавателей. Связь между таблицами факультетов и кафедр также имеет тип один-ко-многим, поэтому второе соединение означает, что к каждой строке таблицы, полученной после первого соединения, приписываются данные о факультете кафедры. Таким образом, количество строк останется равным числу преподавателей.

Последние два условия фразы WHERE отбирают строки из соединенной таблицы, а во фразе SELECT указан выводимый столбец. Ключевое слово DISTINCT указано в нем потому, что названия кафедр в соединенной таблице могут повторяться

**Задание 2.** Вывести фамилии ассистентов факультета математики и информатики.

```
SELECT TEACHER.NAME_TEACHER AS 'Ассистенты ф-та математики и  
125 информатики'
```

```
FROM FACULTET, KAFEDRA, TEACHER  
WHERE FACULTET.Kod_faculteta = KAFEDRA.Kod_faculteta AND  
KAFEDRA.Kod_kafedru = TEACHER.Kod_kafedru AND  
FACULTET.Name_faculteta = 'Математики и информатики' AND  
TEACHER.DOLGNOST = 'ассистент';
```

В этом случае для ответа нужны две таблицы — факультетов и преподавателей. Однако они связаны между собой опосредованно, через таблицу кафедр. Поэтому для соединения таблиц факультетов и преподавателей следует использовать таблицу кафедр.

**Задание 3.** Вывод всех столбцов соединяемой таблицы.

В многотабличном запросе конструкция SELECT \* означает выбор всех столбцов соединенной таблицы. Например, результирующая таблица следующего запроса состоит из 21 столбца: 5 столбцов таблицы факультетов, 6 столбцов таблицы кафедр и 10 столбцов таблицы преподавателей.

```
SELECT *  
FROM FACULTET f, KAFEDRA k, TEACHER t  
WHERE f.Kod_faculteta = k.Kod_faculteta AND k.Kod_kafedru =  
t.Kod_kafedru;
```

**Задание 4.** Если фамилия заведующего кафедры совпадает с фамилией декана какого-нибудь из факультетов, вывести название этой кафедры вместе с названием соответствующего факультета.

**Задание 5.** Вывести пары названий кафедр и фамилий преподавателей, у которых совпадают первичные ключи.

**Задание 6.** Вывести фамилии преподавателей, зарплата которых больше, чем у преподавателя Сидорова.

```
SELECT needed.NAME_TEACHER  
FROM TEACHER needed, TEACHER given  
WHERE needed.Salary + needed.Rise > given.Salary + given.Rise AND
```

given.NAME\_TEACHER = 'Игнатьева Олеся Владимировна';

**Задание 7.** Вывести названия кафедр, располагающихся в том же корпусе, что и кафедра информатики.

**Задание 8.** Указать преподавателей-однофамильцев, которые занимают различные должности

**Задание 9.** Создать запросы с внешним соединением таблиц

Предположим, необходимо вывести список факультетов и их кафедр. Это достигается соединением таблиц FACULTET и KAFEDRA по равенству значений первичного и внешнего ключей и выбором столбцов с названиями факультетов и кафедр. Но в таком случае, если на факультете кафедр нет, он не будет включен в результат.

Для того чтобы в списке присутствовали все факультеты, даже без кафедр, необходимо использовать внешнее соединение, которое расширяет возможности обычного соединения. Внешнее соединение возвращает строки, которые удовлетворяют условию соединения, а также те строки одной из таблиц, для которых в другой не нашлось удовлетворяющих условию соединения строк.

Внутренние соединения возвращают результат, когда в обеих таблицах есть хотя бы одна строка, соответствующая условиям соединения. Внутренние соединения исключают строки, не соответствующие ни одной строке в другой таблице. Однако внешние соединения возвращают все строки хотя бы из одной таблицы или представления, упомянутых в предложении FROM, если они удовлетворяют условиям поиска WHERE или HAVING.

Все строки, получаемые из левой таблицы, образуют левое внешнее соединение, а строки, получаемые из правой таблицы, — правое внешнее соединение. Все строки их обеих таблиц возвращаются в полном внешнем соединении.

Для внешних соединений в предложении FROM SQL Server использует ключевые слова ISO:

- LEFT OUTER JOIN или LEFT JOIN;
- RIGHT OUTER JOIN или RIGHT JOIN;
- FULL OUTER JOIN или FULL JOIN.

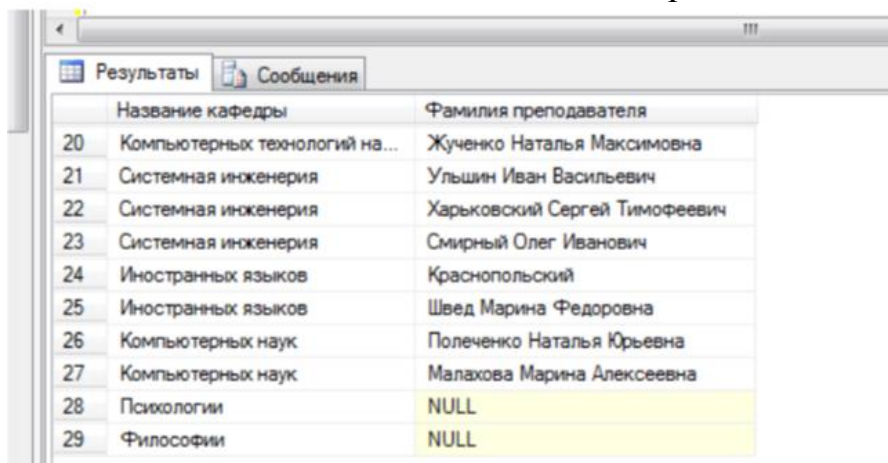
*Работа с левыми внешними соединениями.* Рассмотрим соединение таблиц KAFEDRA и TEACHER по столбцам kod\_kafedru. В результате будут выведены только те кафедры, для которых были написаны преподаватели. Чтобы включить в результаты все кафедры, независимо от того, были ли написаны их преподаватели, используйте левое внешнее соединение ISO.

*Запрос.* Вывести фамилии всех преподавателей с указанием их кафедр, если она есть.

```
SELECT KAFEDRA.Name_Kafedru AS 'название кафедры',  
TEACHER.NAME_TEACHER AS 'фамилия преподавателя'  
FROM KAFEDRA LEFT OUTER JOIN TEACHER  
ON KAFEDRA.kod_kafedru = TEACHER.kod_kafedru;
```

Ключевые слова LEFT OUTER JOIN включают в вывод все строки таблицы KAFEDRA независимо от того, есть ли для них соответствующие значения в столбце kod\_kafedru таблицы TEACHER. Обратите внимание на то, что в

результатах, где для кафедры нет соответствующего преподавателя, строки содержат значение NULL в столбце Фамилия преподавателя.



	Название кафедры	Фамилия преподавателя
20	Компьютерных технологий на...	Жученко Наталья Максимовна
21	Системная инженерия	Ульшин Иван Васильевич
22	Системная инженерия	Харьковский Сергей Тимофеевич
23	Системная инженерия	Смирный Олег Иванович
24	Иностранных языков	Краснопольский
25	Иностранных языков	Швед Марина Федоровна
26	Компьютерных наук	Полченко Наталья Юрьевна
27	Компьютерных наук	Малахова Марина Алексеевна
28	Психологии	NULL
29	Философии	NULL

Работа с правыми внешними соединениями. Рассмотрим соединение таблиц KAFEDRA и TEACHER по столбцам kod\_kafedru. Оператор правого внешнего соединения ISO, RIGHT OUTER JOIN, включает в результаты все строки второй таблицы независимо от того, есть ли для них совпадающие данные в первой таблице. Чтобы включить в результаты всех преподавателей независимо от того, есть ли связанные с ними кафедры, используйте правое внешнее соединение ISO.

*Запрос.* Вывести названия всех кафедр с указанием фамилий преподавателей, если они есть.

```
SELECT KAFEDRA.Name_Kafedru AS 'название кафедры',  
TEACHER.NAME_TEACHER AS 'фамилия преподавателя'  
FROM KAFEDRA RIGHT OUTER JOIN TEACHER  
ON KAFEDRA.kod_kafedru = TEACHER.kod_kafedru;
```

**Задание 10.** Вывести названия всех кафедр корпуса 1 с указанием их преподавателей, если они есть.

**Задание 11.** Вывести названия всех кафедр с указанием их преподавателей, если они есть, ставка которых больше 3000.

## ПРАКТИЧЕСКАЯ РАБОТА № 21

### Тема: Создание запросов на группировку и Сортировку данных. Запросы на изменение. Использование встроенных функций

**Цель работы:** Изучить используемый в реляционных СУБД оператор извлечения данных из таблиц SELECT и выполнение группировки и сортировки данных. Изучить синтаксис языка модификации данных. Научится использовать встроенные функции в запросах.

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, инструкции по выполнению работы.

#### Справочный материал:

Функции SQL подобны любым другим операторам языка в том смысле, что они производят действия с данными и возвращают результат в качестве своего значения. Функции имеют тип, который определяется типом возвращаемого значения, поэтому можно говорить о числовых, строковых, временных функциях и т. д. От обычных операторов функции отличаются форматом представления: имя\_функции[(аргумент[, аргумент]...)]

Этот формат допускает, что функции могут иметь ноль, один или более аргументов, причем при отсутствии аргументов круглые скобки не используются.

Имеется два основных класса функций SQL: *встроенные* и *определяемые пользователем*.

Встроенными являются функции, предопределенные в SQL. Ко второму классу относятся функции, которые пишутся пользователями на специальном языке, обеспечивающем использование всех возможностей SQL. Каждая СУБД использует для этого свой собственный язык.

SQL Server содержит множество встроенных функций, а также поддерживает создание определяемых пользователем функций.

В SQL определено множество встроенных функций различных категорий. На этом уроке мы рассмотрим:

- *агрегатные* (или *групповые*) функции, оперирующие значениями столбцов множества строк и возвращающие одно значение;
- *функции одной строки*, использующие в качестве аргументов значения столбцов одной строки и возвращающие одно значение.

#### Агрегатные функции

Аргументами агрегатных функций могут быть как столбцы таблиц, так и результаты выражений над ними. Агрегатные функции и сами могут включаться в другие арифметические выражения. В стандарте SQL определены следующие виды агрегатных функций: унарные, бинарные, инверсного распределения, гипотетические функции множеств.

AVG - среднее

MIN - минимум

CHECKSUM\_AGG - Возвращает контрольную сумму значений в группе. Значения NULL не учитываются.

SUM - сумма

COUNT - количество

STDEV – среднее квадратическое отклонение

COUNT\_BIG - Возвращает количество элементов в группе.

STDEVP - Возвращает статистическое стандартное отклонение всех значений в указанном выражении.

GROUPING - Указывает, является ли указанное выражение столбца в списке GROUP BY статистическим или нет. В результирующем наборе функция GROUPING возвращает 1 (статистическое выражение) или ноль (нестатистическое выражение).

VAR - дисперсия

GROUPING\_ID - Представляет собой функцию, которая вычисляет уровень группирования.

VARP - Возвращает статистическую дисперсию для заполнения всех значений в указанном выражении.

MAX - максимум

Общий формат унарной агрегатной функции следующий:

имя\_функции([ALL | DISTINCT] выражение) [FILTER (WHERE условие)]

где DISTINCT указывает, что функция должна рассматривать только различные значения аргумента, а ALL — все значения, включая повторяющиеся (этот вариант используется по умолчанию). Фраза FILTER позволяет дополнительно отобрать строки таблицы, столбец которой используется в качестве аргумента функции.

*Агрегатные функции* применяются во фразах SELECT и HAVING. Здесь мы рассмотрим их использование во фразе SELECT. В этом случае выражение в аргументе функции применяется ко всем строкам входной таблицы фразы SELECT. Кроме того, во фразе SELECT нельзя использовать и агрегатные функции, и столбцы таблицы (или выражения с ними) при отсутствии фразы GROUP BY.

Функция COUNT. Функция COUNT имеет два формата. В первом случае возвращается количество строк входной таблицы, во втором случае — количество значений аргумента во входной таблице:

COUNT(\*)

COUNT([DISTINCT | ALL] выражение)

Простейший способ использования этой функции - подсчет количества строк в таблице (всех или удовлетворяющих указанному условию). Для этого используется первый вариант синтаксиса

Функция SUM. Эта агрегатная функция подсчитывает сумму значений аргумента для всех строк входной таблицы. Аргумент должен иметь числовой тип или быть временным промежутком. В качестве аргумента может выступать имя столбца или выражение над столбцами входной таблицы. В этой функции также допускается использовать ключевые слова DISTINCT и ALL.

Функция AVG. Агрегатная функция AVG подсчитывает среднее значение аргумента для всех строк входной таблицы. Аргумент должен иметь числовой тип или быть временным промежутком. В качестве аргумента может выступать имя столбца или выражение над столбцами входной таблицы. Допускается использовать ключевые слова DISTINCT и ALL.

Функции MIN и MAX. Эти функции позволяют находить максимальное (MAX) и минимальное (MIN) значения аргумента для всех строк входной таблицы. Хотя и в этом допускается использование ключевых слов DISTINCT и ALL, они не оказывают влияния на результат. Аргумент этих функций может быть любого типа, для которого определено упорядочение, то есть числовой, строковый и временной.

*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.

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

WIDTH\_BUCKET, с помощью которой можно легко строить гистограммы:

WIDTH\_BUCKET(число, минимум, максимум, количество)

*Функции, получающие компоненты даты и времени.*

Функция извлекает из операнда указанный компонент и возвращает его в виде числа.

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.

*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.

*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, 139 MONTH, DAY, HOUR, MINUTE, SECOND.

Например, DATEADD(month, 1, '2006-08-30')

*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.

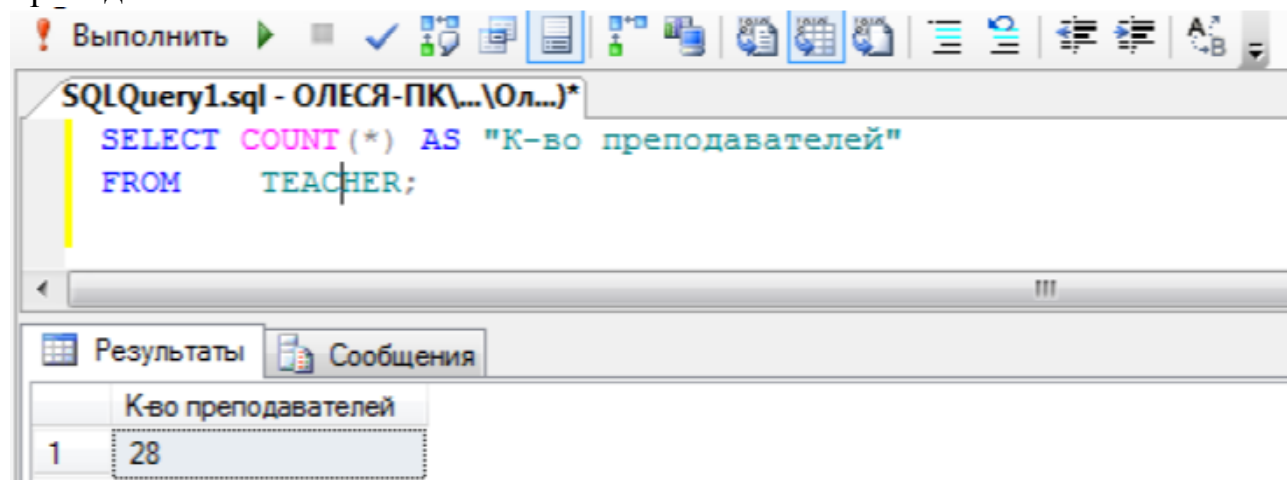
**Содержание работы:**

**Задание 1.** Выполнение запросов к базе данных Университет

*Запрос 1.* Информация о скольких преподавателях имеется в базе данных?

```
SELECT COUNT(*) AS "К-во преподавателей"  
FROM TEACHER;
```

Чтобы выполнить sql-команду нажмите на панели редактора кнопку Выполнить. В результате выполнения данного кода будет подсчитано кол-во всех преподавателей.



*Запрос 2.* Создать запрос на нахождение количества ассистентов, не имеющих телефонов.

*Запрос 3.* Создать запрос на нахождение количества кафедр на факультете математики и информатики.

*Запрос 4.* На скольких различных должностях работают преподаватели кафедры «Компьютерные системы и сети»?

```
SELECT COUNT(DISTINCT DOLGNOST)
FROM KAFEDRA d, TEACHER t
WHERE d.KOD_KAFEDRU = t.KOD_KAFEDRU AND
LOWER(d.NAME_KAFEDRU) = 'Компьютерные системы и сети';
```

*Запрос 5.* Какая суммарная ставка всех ассистентов?

```
SELECT SUM(Salary)
FROM TEACHER
WHERE LOWER(DOLGNOST) = 'ассистент';
```

*Запрос 6.* Создать запрос, определяющий среднюю ставку среди всех преподавателей.

*Запрос 7.* Какова максимальная зарплата преподавателя?

```
SELECT MAX(Salary + Rise)
FROM TEACHER;
```

*Запрос 8.* Вывести процентное соотношение суммарной ставки к суммарной зарплате и наоборот.

```
SELECT SUM(Salary)*100/SUM(Rise) AS "Процент зарплаты к зарплате",
SUM(Rise)*100/SUM(Salary) AS "Процент зарплаты к ставке"
FROM TEACHER;
```

*Запрос 9.* Вывести фамилии всех преподавателей прописными буквами.

```
SELECT UPPER(NAME_TEACHER) AS "Все прописные"
FROM TEACHER;
```

*Запрос 10.* Вывести фамилии всех преподавателей родившихся в 1979 году.

```
SELECT Name_teacher, BIRTHDAY
FROM TEACHER
WHERE DATENAME(YEAR, BIRTHDAY)=1979;
```

*Запрос 11.* 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 'С%';
```

*Запрос 12.* Например, если вы хотите узнать, сколько месяцев уже проработал Иванов, можно выполнить такой запрос:

```
SELECT 'Иванов проработал ' ||
ROUND(DATEDIFF(MONTH, GETDATE(), DATA_HIRE), 1) || ' месяцев' AS
"Стаж Иванова"
FROM TEACHER
WHERE NAME_TEACHER LIKE 'Иван%';
```

## ПРАКТИЧЕСКАЯ РАБОТА № 22

**Тема: Создание запросов на группировку и Сортировку данных. Запросы на изменение. Использование встроенных функций**

**Цель работы:** Изучить используемый в реляционных СУБД оператор извлечения данных из таблиц SELECT и выполнение группировки и сортировки данных. Изучить синтаксис языка модификации данных. Научится использовать встроенные функции в запросах.

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, инструкции по выполнению работы.

**Справочный материал:**

*HAVING, GROUP BY и ORDER BY.*

Первая из них позволяет группировать строки таблицы и применять к созданным группам агрегатные функции.

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.

*Группировка по одному столбцу*

Группировка по значениям одного столбца является самым простым вариантом использования фразы GROUP BY.

*Группировка по нескольким столбцам*

SQL позволяет группировать строки таблицы и по нескольким столбцам. В этом случае имена столбцов перечисляются во фразе GROUP BY через запятую.

*Вложение агрегатных функций*

Если фраза GROUP BY в запросе отсутствует, то во фразе SELECT нельзя вкладывать агрегатные функции друг в друга. Например, следующий запрос приведет к ошибке:

```
SELECT AVG(MIN(Salary))  
FROM TEACHER;
```

Однако при наличии фразы GROUP BY такое вложение допускается. Оно интерпретируется следующим образом: сначала для каждой группы выполняется вложенная агрегатная функция, затем к полученной таким образом промежуточной таблице применяется внешняя агрегатная функция. Двойное вложение, например, MAX(AVG(MIN(Salary))), недопустимо.

#### *Условие отбора групп*

Для отбора строк среди полученных групп следует применять фразу HAVING. Она играет такую же роль для групп, что и фраза WHERE для исходных таблиц, и может использоваться лишь при наличии фразы GROUP BY. В предложении SELECT фразы WHERE, GROUP BY и HAVING обрабатываются в следующем порядке.

1. Фразой WHERE отбираются строки, удовлетворяющие указанному в ней условию.
2. Фраза GROUP BY группирует отобранные строки.
3. Фразой HAVING отбираются группы, удовлетворяющие указанному в ней условию.

Фраза HAVING может использоваться лишь при наличии фразы GROUP BY. Из этого правила синтаксис SQL допускает только одно исключение: когда вся таблица интерпретируется как одна группа. В этом случае в списке фразы SELECT можно использовать только константы, агрегатные функции и выражения над ними

#### *Сортировка результирующих строк*

Строки в таблицах базы данных неупорядочены. Также неупорядочены и строки результирующей таблицы запроса, однако для их упорядочения в предложении SELECT можно воспользоваться фразой ORDER BY. Она сортирует по значению указанных в ней столбцов (и выражений над столбцами) строки результирующей таблицы запроса. Синтаксис этой фразы следующий:

ORDER BY спецификация\_сортировки[. спецификация\_сортировки]... 144

где спецификация\_сортировки имеет такой синтаксис:

выражение\_сортировки [направление\_сортировки] [положение\_NULL]

Сортировать можно по столбцам (выражениям) тех типов, для которых определены операции сравнения. Это относится, в частности, к символьным строкам, числам и временным значениям. Можно указывать направление сортировки и место расположения строк, имеющих значение NULL для выражений сортировки.

*Сортировка по столбцу или выражению.* Сортировать строки результирующей таблицы запроса можно по отдельным столбцам, совокупности столбцов, а также по одному или нескольким выражениям над столбцами.

*Сортировка по столбцу.* Простейший вариант сортировки - это сортировка по одному из столбцов результирующей таблицы.

*Направление сортировки.* В SQL порядок сортировки по возрастанию определен по умолчанию. Однако есть возможность и явно указать направление сортировки с помощью ключевых слов ASC (по возрастанию) и DESC (по

убыванию), которые следует располагать после имени сортируемого столбца (выражается).

### **Содержание работы:**

**Задание 1.** Создать запросы к базе данных Университет

*Запрос 1.* Для каждого корпуса подсчитать количество находящихся в нем кафедр.

```
SELECT NUM_KORPUSA AS "Корпус",  
COUNT(*) AS "К-во кафедр"  
FROM KAFEDRA GROUP BY NUM_KORPUSA;
```

*Запрос 2.* Для каждой из должностей указать суммарный фонд заработной платы.

*Запрос 3.* Для каждого факультета, расположенного в корпусе 1, вывести количество групп и общее количество студентов по каждой кафедре.

*Запрос 4.* Для каждого факультета, расположенного в корпусе 1, вывести сколько учится студентов по каждой группе.

```
SELECT f.Name_faculteta, s."GROUP", count(s."GROUP") AS "Кол-во  
студентов в группе"
```

```
FROM FACULTET f, KAFEDRA d, STUDENT s  
WHERE f.KOD_FACULTETA = d.KOD_FACULTETA  
AND d.KOD_kafedru = s.KOD_kafedru AND d.NUM_KORPUSA = '1'  
GROUP BY f.Name_faculteta,s."GROUP";
```

*Запрос 5.* Для каждой кафедры и должности вывести суммарную и среднюю зарплату преподавателей

*Запрос 6.* Для каждого значения зарплаты, не превышающего 1500, вывести это значение и количество преподавателей, такую зарплату получающих.

```
SELECT Salary + Rise, COUNT(*)  
FROM TEACHER  
WHERE Salary + Rise <= 1500 GROUP BY Salary + Rise;
```

*Запрос 7.* Вывести среднее значение среди минимальных и максимальных ставок для каждой группы преподавателей, занимающих одну должность, а также минимальное и максимальное значения среди средних ставок.

```
SELECT AVG(MIN(Salary)) AS AVG_MIN,  
AVG(MAX(Salary)) AS AVG_MAX,  
MIN(AVG(Salary)) AS MIN_AVG,  
MAX(AVG(Salary)) AS MAX_AVG  
FROM TEACHER GROUP BY Dolgnost;
```

*Запрос 8.* Вывести номера кафедр, у которых суммарное количество работающих профессоров более 1.

```
SELECT KOD_kafedru as "Номер кафедры" ,Count(*) as "Кол-во профессоров  
на кафедре"  
FROM TEACHER
```

```
WHERE dolgnost='профессор'  
GROUP BY KOD_kafedru 143  
HAVING count(dolgnost) > 1 ;
```

*Запрос 9.* Вывести названия кафедр факультета математики и информатики, на которых работают один и более профессоров. Указать также количество профессоров и их суммарную зарплату.

```
SELECT d.Name_kafedru, Count(*), SUM(t.salary + t.Rise)  
FROM FACULTET f, KAFEDRA d, TEACHER t  
WHERE f.KOD_FACULTETA = d.KOD_FACULTETA  
AND d.KOD_kafedru = t.KOD_kafedru  
AND LOWER(f.Name_faculteta) = 'математики и информатики'  
AND LOWER(t.Dolgnost) = 'профессор'  
GROUP BY d.Name_kafedru  
HAVING COUNT(*) > 0;
```

*Запрос 10.* Если суммарная зарплата всех преподавателей превышает 15 000, вывести их минимальную ставку, максимальную надбавку и суммарную зарплату.

```
SELECT MIN(Salary), MAX(Rise), SUM(Salary + Rise)  
FROM TEACHER  
HAVING SUM(Salary + Rise) > 15000;
```

*Запрос 11.* Если суммарная зарплата всех ассистентов превышает 2500, вывести их среднюю ставку, среднюю надбавку и суммарную зарплату.

```
SELECT AVG(Salary), AVG(Rise), SUM(Salary + Rise)  
FROM TEACHER  
WHERE LOWER(Dolgnost) = 'ассистент'  
HAVING SUM(Salary + Rise) > 2500;
```

*Запрос 12.* Вывести алфавитный список фамилий профессоров и доцентов.

```
SELECT NAME_TEACHER  
FROM TEACHER  
WHERE LOWER(Dolgnost) = 'профессор' OR LOWER(Dolgnost) = 'доцент'  
ORDER BY NAME_TEACHER;
```

*Запрос 13.* Вывести фамилии ассистентов и их зарплату по ее возрастанию.

```
SELECT Name_teacher, Salary + Rise  
FROM TEACHER  
WHERE LOWER(Dolgnost) = 'ассистент'  
ORDER BY Salary + Rise;
```

*Запрос 14.* Вывести фамилии ассистентов и дату их приема на работу по возрастанию даты.

```
SELECT Name_teacher, Data_hire  
FROM TEACHER  
WHERE LOWER(Dolgnost) = 'ассистент'  
ORDER BY Data_hire ASC;
```

*Запрос 15.* Вывести фамилии доцентов в обратном алфавитном порядке и их зарплату.



## ПРАКТИЧЕСКАЯ РАБОТА № 23

### Тема: Создание базы данных в среде разработки на языке SQL

**Цели работы:** научить создавать БД в среде разработки MS SQL Server Management Studio.

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, инструкции по выполнению работы.

#### **Содержание работы:**

**Задание 1.** Создать БД проектной организации на языке SQL. Основным видом деятельности такой организации – выполнение проектов по договорам с заказчиками.

#### *Создание таблиц*

1. Отношение Departs (отделы):

```
create table departs (  
  d_id varchar(12) primary key,  
  d_name varchar(100) not null);
```

2. Отношение Rooms (комнаты):

```
create table rooms (  
  d_depart varchar(12) references departs(d_id),  
  r_room numeric(4) not null,  
  r_phone varchar(20),  
  unique(r_room, r_phone));
```

3. Отношение Posts (должности):

```
create table posts (  
  p_post varchar(30) primary key,  
  p_salary numeric(8,2) not null check(p_salary>=4500));
```

4. Отношение Employees (сотрудники):

```
create table employees (  
  e_id numeric(4) primary key,  
  e_fname varchar(25) not null,  
  e_lname varchar(30) not null,  
  e_born date not null,  
  e_sex char(1) check(e_sex in ('ж','м')),  
  e_pasp char(10) not null unique,  
  e_date date not null,  
  e_given varchar(50) not null,  
  e_inn char(12) not null unique,  
  e_pens char(14) not null unique,  
  e_depart varchar(12) references departs,  
  e_post varchar(30) references posts,  
  e_room numeric(4) not null,  
  e_phone varchar(20) not null,  
  e_login varchar(30),  
  foreign key(e_room,e_phone)  
  references rooms(r_room,r_phone));
```

(Если внешний ключ ссылается на первичный ключ отношения, его можно не указывать, как в случае ссылок на Departs и Posts).

5. Отношение Edu (образование):

```
create table edu (  
  e_id numeric(4) primary key,
```

```
u_id numeric(4) references employees,  
u_type varchar(20) not null,  
u_spec varchar(40),  
u_diplom varchar(15),  
u_year number(4) not null,  
check(u_spec in ('начальное', 'среднее', 'высшее', 'средне-специальное')));
```

6. Отношение AdrTel (адреса-телефоны):

```
create table adrtel (  
a_id numeric(4) references employees,  
a_adr varchar(50),  
a_phone varchar(30));
```

7. Отношение Clients (заказчики):

```
create table clients (  
c_id numeric(4) primary key,  
c_company varchar(40) not null,  
c_adr varchar(50) not null,  
c_person varchar(50) not null,  
c_phone varchar(30));
```

8. Отношение Projects (проекты):

```
create table projects (  
p_id numeric(6) not null unique,  
p_title varchar(100) not null,  
p_abbr char(10) primary key,  
p_depart varchar(12) references departs,  
p_company numeric(4) references clients,  
p_chief numeric(4) references employees,  
p_begin date not null,  
p_end date not null,  
p_finish date,  
p_cost numeric(10) not null check(p_cost>0),  
check (p_end>p_begin),  
check (p_finish is null or p_finish>p_begin));
```

9. Отношение Stages (этапы проектов):

```
create table stages (  
s_pro char(10) references projects,  
s_num numeric(2) not null,  
s_title varchar(200) not null,  
s_begin date not null,  
s_end date not null,  
s_finish date,  
s_cost numeric(10) not null,  
s_sum numeric(10) not null,  
s_form varchar(100) not null,  
check (s_cost>0),  
check (s_end>s_begin),  
check (s_finish is null or s_finish>s_begin));
```

10. Отношение Job (участие):

```
create table job (  

```

```
j_pro char(10) references projects,
j_emp numeric(2) references employees,
j_role varchar(20) not null,
j_bonus numeric(2) not null,
check(j_bonus>0),
check (j_role in ('исполнитель', 'консультант')));
```

*Создание представлений (готовых запросов)*

Приведём примеры нескольких готовых запросов (представлений):

1. Список всех текущих проектов (sysdate – функция, возвращающая текущую дату, определена в СУБД Oracle; в других системах аналогичная функция может называться по-другому, например, getdate() в Transact-SQL, now() в MS Access, currrdate() в MySQL и т.д.):

```
create view curr_projects as
select *
from projects
where p_begin<=sysdate and sysdate<=p_end;
```

2. Определение суммы по текущим проектам, полученной на текущую дату:

```
create or replace view summ (title, cost, total) as
select p_title, p_cost, sum(s_sum)
from curr_projects, stages
where p_abbr=s_pro
group by p_title, p_cost;
```

3. Список сотрудников, участвующих в текущих проектах:

```
create view participants (project, name, role) as
select p_abbr, e_fname||' '||e_lname, 'руководитель'
from curr_projects, employees
where p_chief=e_id
union all
select p_abbr, e_fname||' '||e_lname, j_role
from curr_projects, employees, job
where p_abbr=j_pro and e_id=j_emp
order by 1, 3 desc;
```

4. Список рабочих телефонов сотрудников:

```
create or replace view worktel (name, room, phone) as
select e_fname||' '||e_lname, e_room, e_phone
from employees
order by 1;
```

5. Форма отчётности и сроки выполнения этапов по текущим проектам:

```
create or replace view reports as
select s_pro, s_num, s_title, s_begin, s_end, s_form
from stages
order by 1, 2;
```

6. Данные о проектах для руководителя проектов:

```
create or replace view my_projects as
select *
from projects p
where exists (select * from employees e
where e.e_id=p.p_chief and e.e_login=user);
```

Функция user возвращает имя пользователя, выполняющего текущий запрос. Таким образом, каждый пользователь получит данные только о тех проектах, руководителем которых является. Используя аналогичный способ, можно ограничить участника проекта данными только о сотрудниках тех проектов, в которых он сам участвует.

7. Данные об этапах проектов для руководителя проектов:

```
create or replace view my_stages as
select s.*
from stages s
where exists (select *
from employees e, projects p
where e.e_id=p.p_chief and e.e_login=user
and s.s_pro=p.p_abbr);
```

8. Данные об участниках проектов для руководителя проектов:

```
create or replace view my_staff as
select j.*
from job j
where exists (select *
from employees e, projects p
where e.e_id=p.p_chief and e.e_login=user
and j.j_pro=p.p_abbr);
```

9. Данные о других участниках проекта:

```
create or replace view my_emps as
select je.j_pro, e.e_fname||' '||e.e_lname e_name,
e_depart, e_post, e_phone, e_room
from employees e, job je
where e.e_id=je.j_emp and exists (select *
from job jm, employees m
where m.e_id=jm.j_emp and
m.e_login=user and je.j_pro=jm.j_pro);
```

Для того чтобы можно было работать с этими представлениями, соответствующим пользователям нужно назначить права доступа к представлениям.

Представления	Права доступа к представлениям		
	Группы пользователей (роли)		
	Руководители организации	Руководители проектов	Участники проектов
Текущие проекты (curr_projects)	S	S	
Сумма по текущим проектам (summ)	S	S	
Рабочие телефоны (worktel)	S	S	S
Участники проектов (participants)	S	S	S
Отчетность (reports)	S	S	S
Проекты для руководителя (my_projects)		SIUD	
Стадии проектов (my_stages)		SIUD	
Участники проектов для руководителей (my_staff)		SIUD	
Участники проектов (my_emps)			S

Заполните созданную БД данными, используя команды SQL

## ПРАКТИЧЕСКАЯ РАБОТА № 24

### Тема: Создание БД в среде разработки

**Цели работы:** научить создавать БД в среде разработки MS SQL Server Management Studio.

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, инструкции по выполнению работы.

#### **Содержание работы:**

**Задание.** Создать и заполнить таблицы, представляющие собой фрагмент базы данных торговой фирмы, используя язык SQL. Создать схему данных.

Таблица «Партнер» с полями: «Код партнера» - числовое поле, размер поля – целое, ключевое поле; «ИНН» - числовое поле, размер поля – целое, обеспечить проверку на уникальность поля; «КПП» - числовое поле, размер поля – целое, обеспечить проверку на уникальность поля; «Юридический статус» - текстовое поле, размер поля – 50; «Наименование» - текстовое поле, размер поля – 150; «Юридический адрес» - текстовое поле, размер поля – 150; «Адрес e-mail» - текстовое поле, размер поля – 20.

Таблица «Телефоны партнеров»: «Код партнера» - числовое поле, размер поля – целое; «Телефон» - текстовое поле, размер поля – 15. Создать составной ключ, включающий оба эти поля.

Таблица «Товары»: «Код товара» - числовое поле, размер поля – целое, ключевое поле; «Наименование товара» - текстовое поле, размер поля – 100; «Цена» - денежный тип данных; «Наличие на складе» - числовое поле, размер поля – целое.

Таблица «Сотрудник»: «Код сотрудника» - тип данных «Счетчик», поле создать автоматически при завершении описания структуры таблицы, после чего переименовать его; «ФИО» - текстовое поле, размер поля – 30.

Таблица «Счета»: «Номер счета» - тип данных «Счетчик», поле создать автоматически как ключ таблицы; «Код партнера» - числовое поле, размер поля – целое; «Дата» - поле типа «Дата/время»; «Код менеджера» - числовое поле, размер поля – длинное целое.

Таблица «Позиция счета»: «Номер счета» - числовое поле, размер поля – длинное целое; «Код товара» - числовое поле, размер поля – целое; «Количество» - числовое поле, размер поля – целое. Создать составной ключ, включающий поля «Номер счета» и «Код товара».

## ПРАКТИЧЕСКАЯ РАБОТА № 25

### Тема: Создание БД в среде разработки

**Цели работы:** научить создавать БД в среде разработки MS SQL Server Management Studio.

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, инструкции по выполнению работы.

#### **Содержание работы:**

**Задание 1.** Создать и заполнить таблицы, представляющие собой фрагмент базы данных учебного центра, используя язык SQL. Создать схему данных.

Таблица «Курс» с полями: «Код курса» - поле типа «Счетчик», создать автоматически как ключ при завершении описания таблицы; «Наименование курса» - текстовое поле длиной 120 символов; «Продолжительность» - числовое поле, размер поля – целое; «Стоимость обучения» - поле денежного типа.

Таблица «Преподаватель» с полями: «Код преподавателя» - поле типа «Счетчик», ключевое поле; «ФИО преподавателя» - текстовое поле, 50 символов; «Дата рождения» - поле типа «Дата/время», «Должность» - текстовое поле, 25 символов, «Научно-педагогический стаж» - числовое; «Общий стаж работы» - числовое; «Контактный телефон» - текстовое поле, 15 символов.

Таблица «Владение предметами» с полями: «Код преподавателя», «Код курса» - числовое поле, размер поля – длинное целое. Создать составной ключ, включающий оба эти поля.

Таблица «График учебного процесса» с полями: «Код потока» - поле типа «Счетчик», создать автоматически как ключ при завершении описания таблицы; «Код курса» - числовое поле, размер поля – длинное целое; «Дата начала» - поле типа «Дата/время»; «Дата завершения» - поле типа «Дата/время»; «Время начала» - поле типа «Дата/время»; «Время завершения» - поле типа «Дата/время».

Таблица «Слушатель» с полями: «Код слушателя» - числовое поле, размер поля – длинное целое, поле ключа; «ФИО слушателя» - текстовое поле, 50 символов; «Контактный телефон» - текстовое поле, 10 символов.

Таблица «Запись на курс» с полями: «Код потока» - числовое поле, размер поля – длинное целое; «Код слушателя» - числовое поле, размер поля – длинное целое. Создать составной ключ, включающий оба эти поля.

## **ПРАКТИЧЕСКАЯ РАБОТА № 26**

### **Тема: Создание БД в среде разработки**

**Цели работы:** научить создавать БД в среде разработки MS SQL Server Management Studio.

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, инструкции по выполнению работы.

#### **Содержание работы:**

**Задание 1.** Создать БД «Компания по разработке программных продуктов», на основании описания предметной области:

*Сущность задачи:* Компания заключает договор с клиентом на разработку программного продукта согласно техническому заданию. После утверждения технического задания определяется состав и объем работ, составляется предварительная смета. На каждый проект назначается ответственный за его выполнение – куратор проекта, который распределяет нагрузку между программистами и следит за выполнением технического задания. Когда программный продукт готов, то его внедряют, производят обучение клиента и осуществляют дальнейшее сопровождение. По результатам своей деятельности компания производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики.

Создать и заполнить таблицы, представляющие собой фрагмент базы данных компании по разработке программных продуктов, используя язык SQL. Создать схему данных.



## ПРАКТИЧЕСКАЯ РАБОТА № 27

### Тема: Программирование баз данных на языке C#. Технология ADO.NET.

#### Соединение с базой данных

**Цель работы:** научиться разрабатывать программы на языке C# с применением технологии ADO.NET для программирования баз данных.

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, MS Visual Studio, инструкции по выполнению работы.

#### Содержание работы:

**Задание 1.** Создать простое приложение баз данных, которое выводит на экранную форму информацию из таблиц разработанной базы данных Microsoft SQL Server University.mdf.

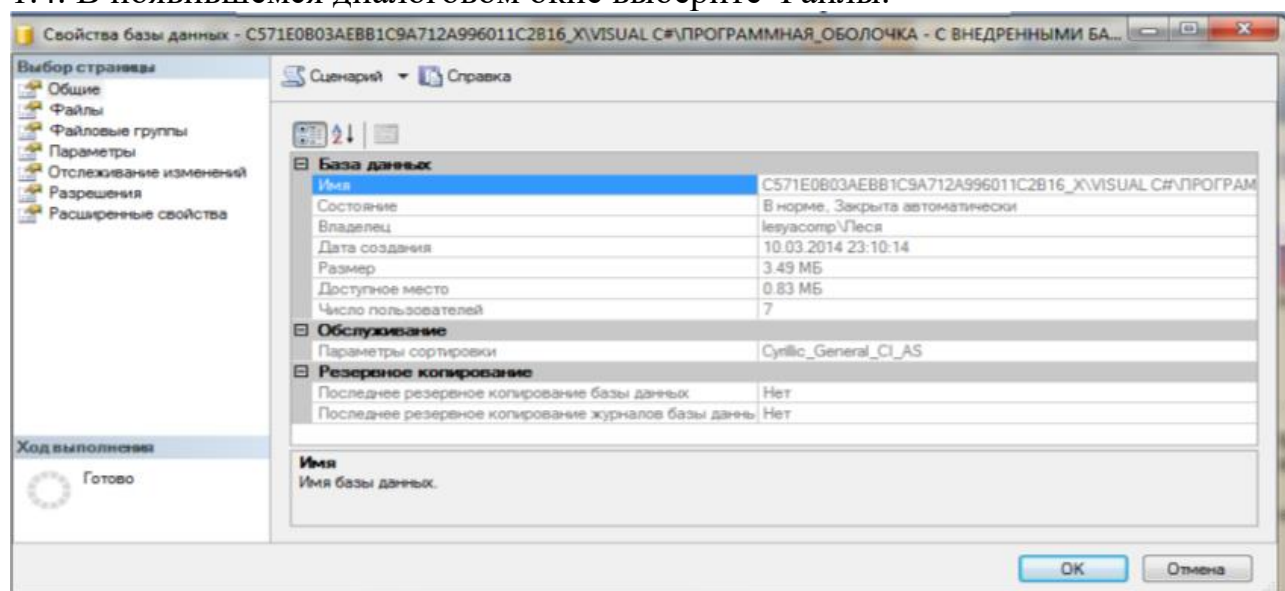
1. Создадим копию базы данных. Для этого выполните следующие действия:

1.1. Откройте среду SQL Server Managemant Studio.

1.2. Соединитесь с сервером.

1.3. Выберите в обозревателе объектов базу данных University.mdf, нажмите правой клавишей и из контекстного меню выберите Свойства.

1.4. В появившемся диалоговом окне выберите Файлы.

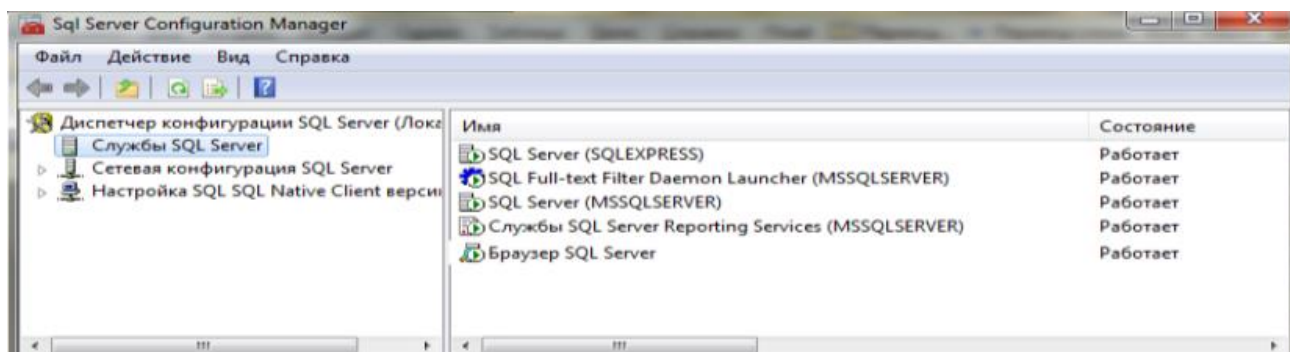


1.5. В появившемся диалоговом окне вы увидите свойства 2-х основных файлов базы данных University.mdf и University\_log.ldf. Прокрутите свойства и прочитайте Путь к файлу.

1.6. Теперь запомнив путь, полностью закрываем среду SQL Server Managemant Studio.

1.7. Далее для того, чтобы сделать копию базы данных и перенести эту копию в другое место необходимо остановить сервер. Для этого откройте Диспетчер конфигурации SQL Server (Пуск/Программы/Microsoft SQL Server 2022 R2/Средства настройки/ Диспетчер конфигурации SQL Server)

1.8. В появившемся окне диспетчера сервера выберите SQL Server (SQLEXPRESS) и остановите его.



1.9. Далее найдите файлы базы данных University.mdf и University\_log.ldf и скопируйте их.

1.10. В вашей папке Базы данных создайте папку VS# и перенесите копию базы данных в нее.

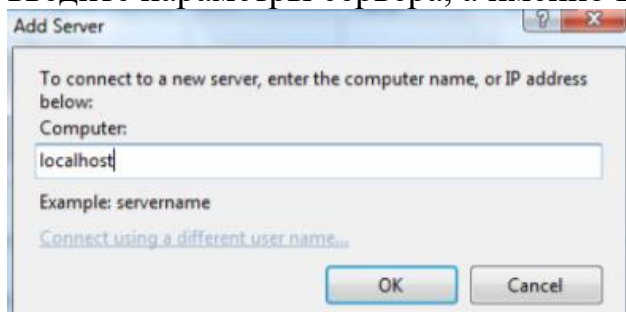
11. Далее, запустите обратно сервер с помощью Диспетчера конфигурации SQL Server.

2. В качестве среды разработки клиентского приложения будем использовать среду программирования MS Visual Studio 2022.

2.1. Создадим пустое Windows-приложение. Команда Создать/Проект/ (File/Project..). Далее выберите в этом диалоговом окне слева язык программирования – Visual C# и затем выберите в выпадающем списке Windows. Далее в появившемся списке выберите Windows Forms Application. Укажите внизу имя и расположение приложения. Имя можно оставить без изменения, а путь укажите созданной в п.1.10 папке.

2.2. Шаг 1. Соединение с сервером и с базой данных в визуальном режиме Server Explorer. В состав Visual Studio входит замечательный инструмент управления и обзора подключениями к базам данных – Обзорщик баз данных (Server Explorer). Выполним соединение с базой данных University. Для этого:

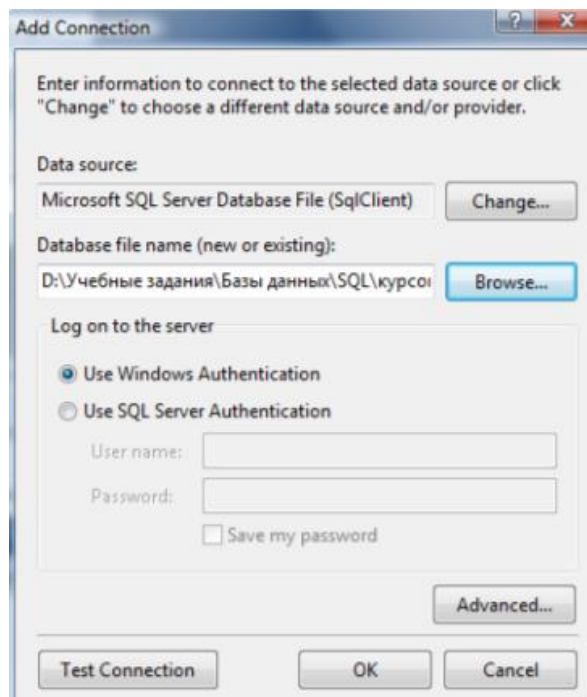
2.2.1. Выберите меню Tools/Connect to Server. В появившемся диалоговом окне введите параметры сервера, а именно Localhost



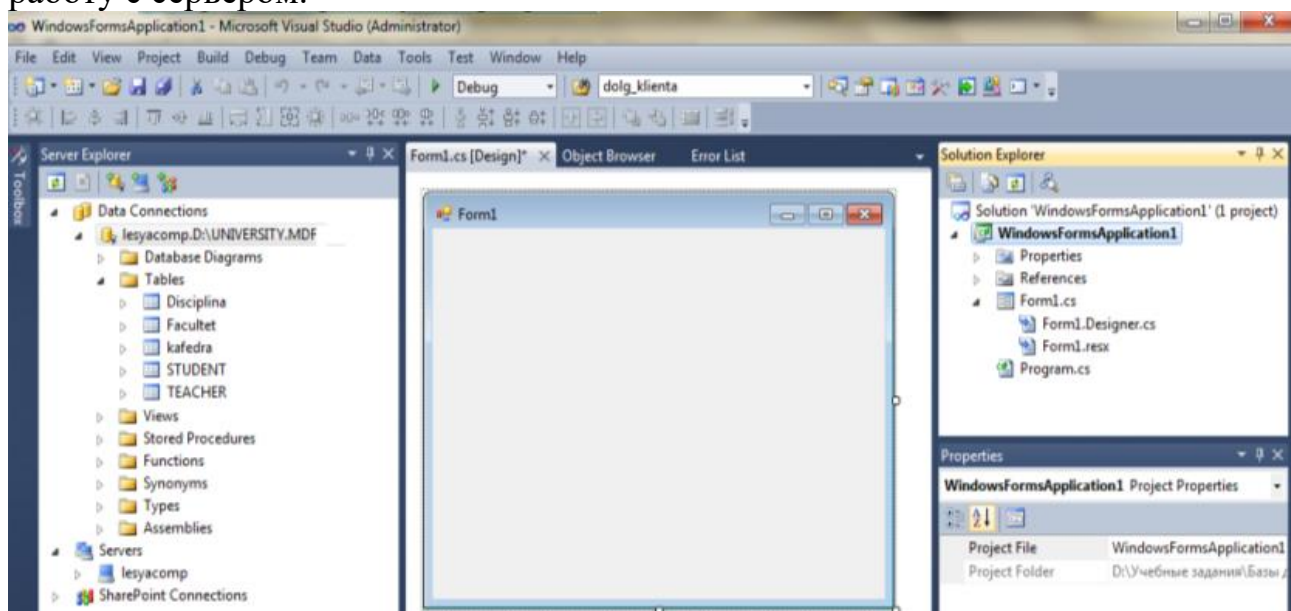
2.2.2. Для соединения с базой данных выберите в меню Tools/Connect to database/ .

2.2.3. В появившемся окне Add Connection - выбрать адаптер Microsoft SQL Server Database File (SQLClient) в поле Data source с помощью кнопки Change.. и выберите базу данных с помощью кнопки Browse...

2.2.4. Проверьте подключение, нажав кнопку Test Connection. Если соединение имеется, то выполните его, нажав кнопку Ok.

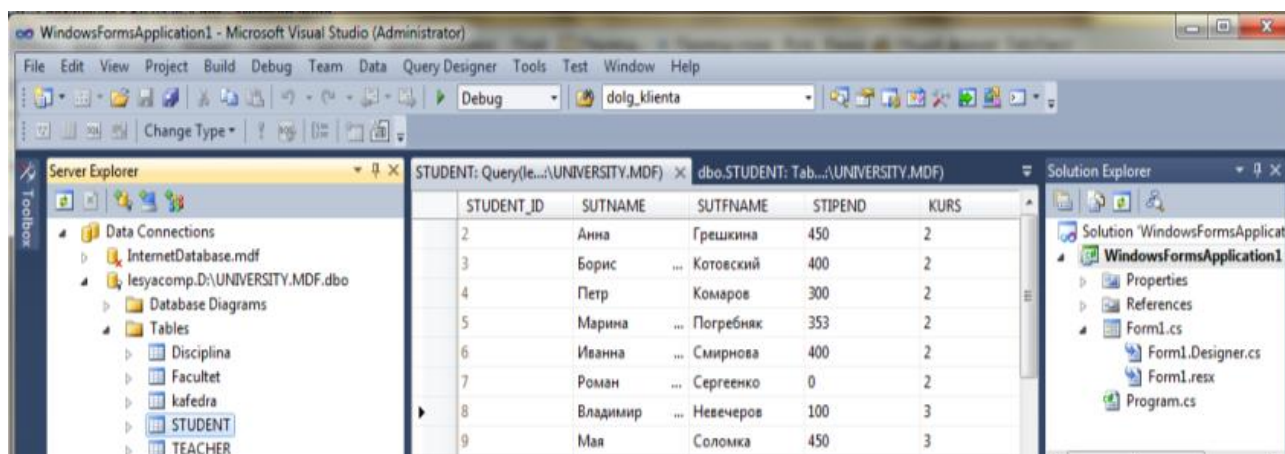


2.2.5. После успешного соединения у вас появится панель Server Explorer, с помощью которой вы можете просмотреть содержимое базы данных и настраивать работу с сервером.



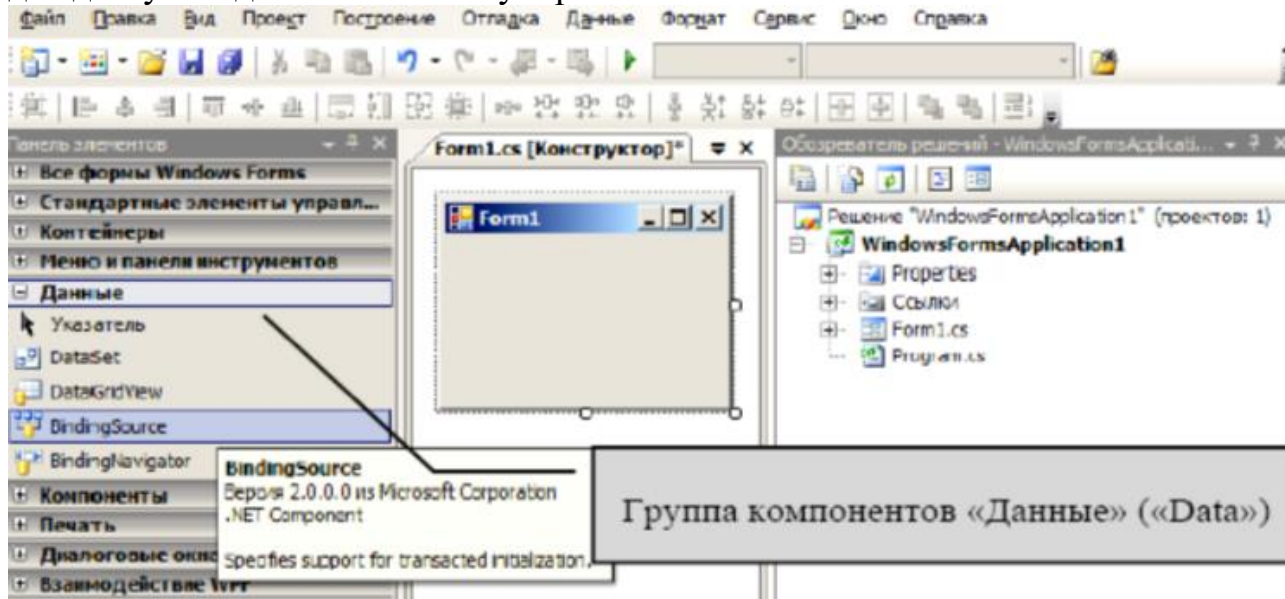
2.2.6. Например, откройте таблицы базы данных University.mdf, выберите таблицу Facultet. Двойным щелчком автоматически добавиться окно с данными таблицы. Этот режим напоминает режим конструктора таблиц.

2.2.7. Также можно использовать окно Обозреватель баз данных для быстрого просмотра содержимого баз данных и – если подключение было создано с правами администратора – изменения их. Например, откроем нужную таблицу – Student и в выпадающем меню выберем пункт Отобразить данные таблицы (Show Table Data).

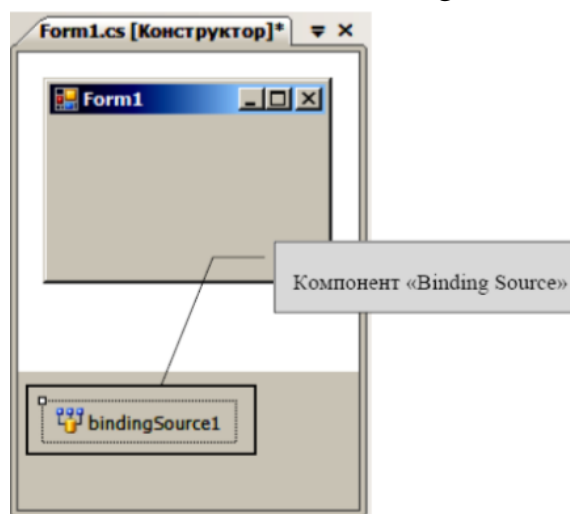


2.3. Шаг 2. Создание простейшего приложения с использованием визуальной среды. В качестве источника данных будем использовать базу данных University. Разработаем диалоговую форму для отображения двух таблиц – Кафедры и Студенты.

2.3.1. В рабочей области закройте две открытые таблицы и вернитесь к пустой форме. Слева расположена Панель элементов (ToolBox). Все элементы сгруппированы. Выберите группу Данные (Data), которая содержит компоненты для доступа к данным и манипулирования ими.



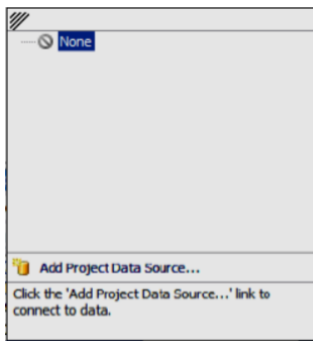
2.3.2. Привязку данных БД к форме осуществляет компонент Binding Source». Перенесем его на форму. После размещения его на форме среда разработки примет следующий вид:



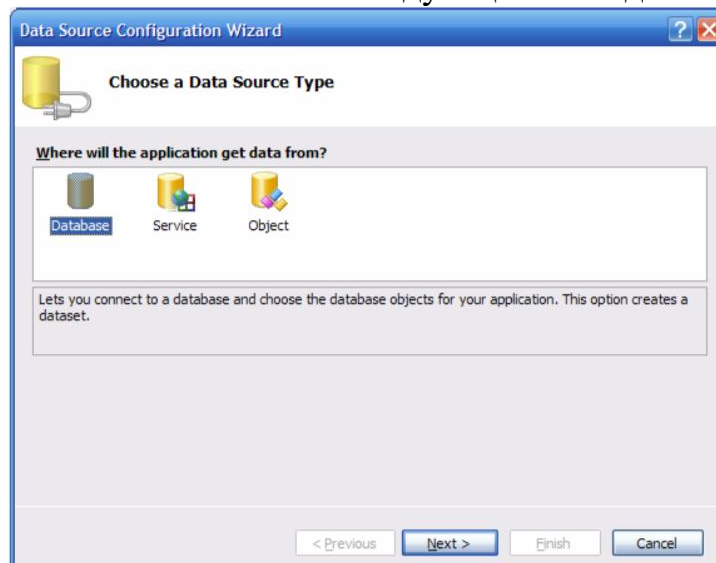
Компонент является не визуальным, поэтому он отображается на дополнительной панели. Основным свойством компонента является свойство DataSource, указывающее на источник данных. По умолчанию свойство является пустым, поэтому необходимо сформировать его



значение. При выборе данного свойства в окне свойств появляется следующее окно Список источников данных:

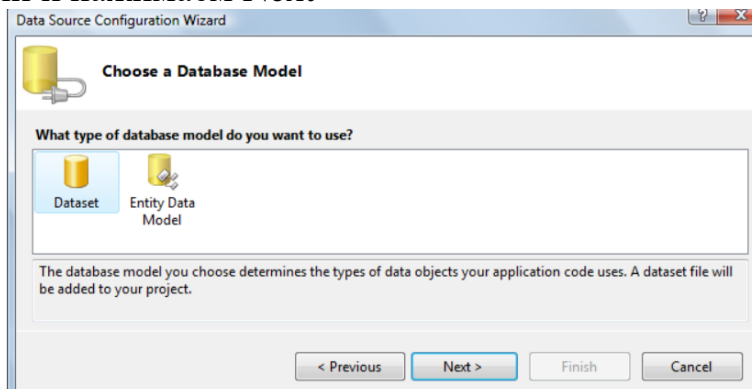


2.3.3. В данный момент список пуст, поэтому необходимо создать новый источник данных, выбрав команду Add Project Data Source для создания нового источника данных и соединения с ним. Появляется следующее окно диалога

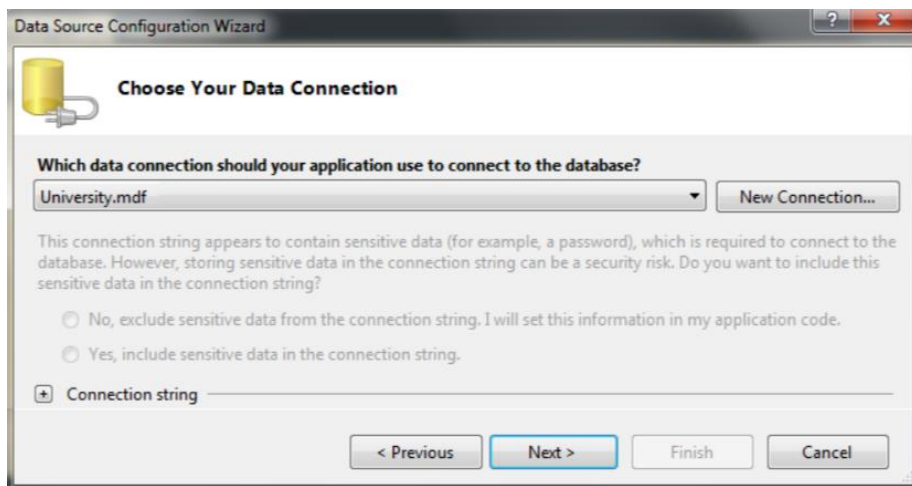


Данный диалог предоставляет следующий выбор источников данных: Database – база данных; Service –это некоторый сервис, предоставляющий данные, чаще всего это Web-сервис; Object –для выбора объекта, который будет генерировать данные и объекты для работы с ними.

2.3.4. Необходимо выбрать пункт Database (База данных). Появляется окно выбора соединения с данными и нажимаем Next



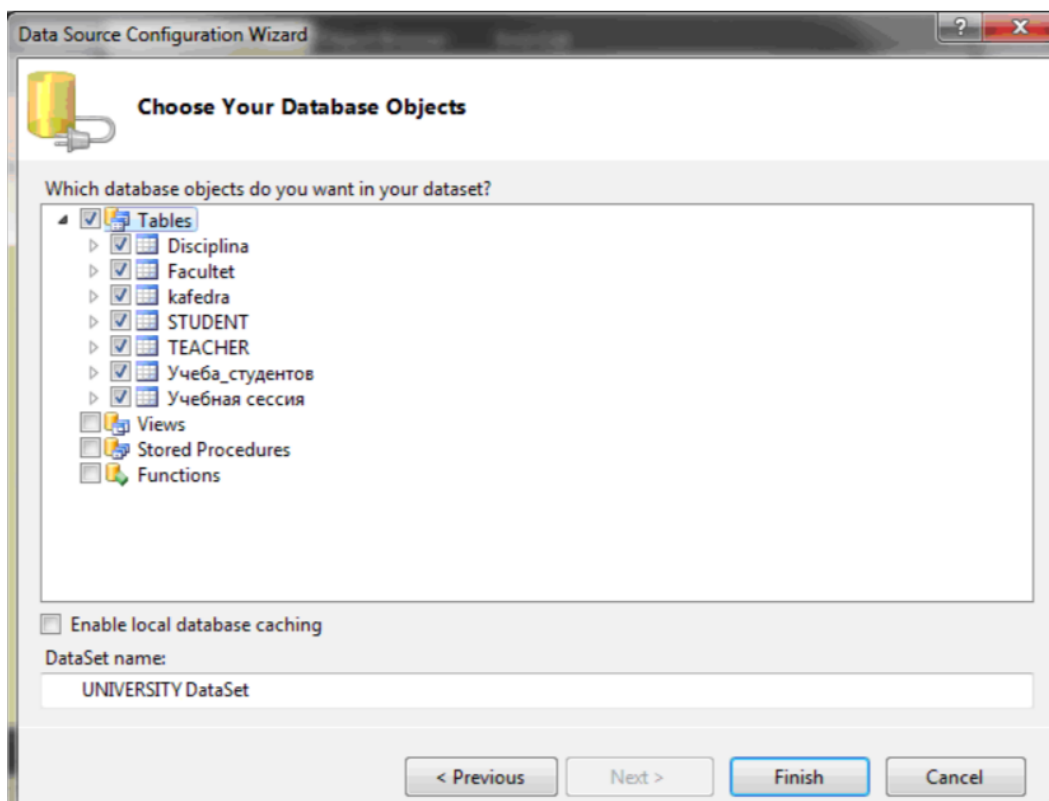
Выберите готовое уже созданное ранее подключение – University.



Целью данного диалога является создание строки соединения, в которой будут описаны параметры соединения для механизма ADO, такие как тип БД, ее местонахождение, имена пользователей, средства безопасности и пр.

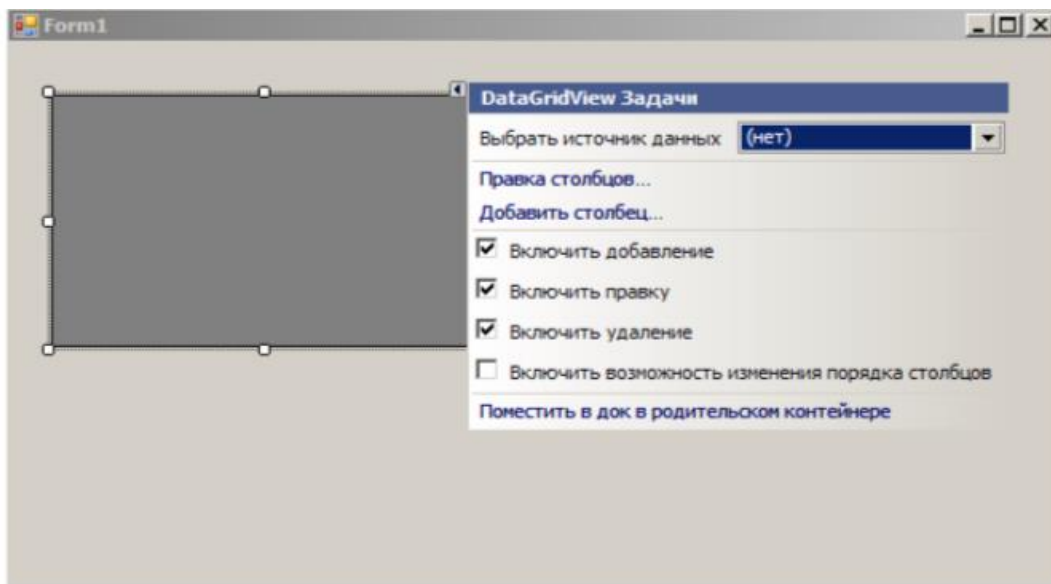
В выпадающем списке диалога находятся все создаваемые ранее соединения. Если необходимого соединения в списке нет, то следует использовать кнопку Создать подключение (New connection).

2.3.5. Если подключение выбрано, то нажать кнопку Далее (Next). В следующем окне выбираем, что будет отображаться в БД. Установите галочку на элемент Tables.



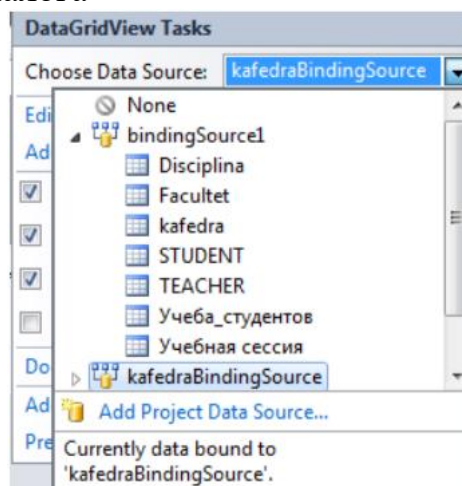
На этом создание источника данных завершено. После нажатия кнопки Готово (Finish) рядом с компонентом BindingSource на форме появляется компонент DataSet.

2.3.6. Теперь данные, подключенные выше, необходимо отобразить на форме. Простейшим способом отображения данных является использование компонента DataGridView из группы компонентов Data. Компонент является визуальным и на форме выглядит следующим образом.

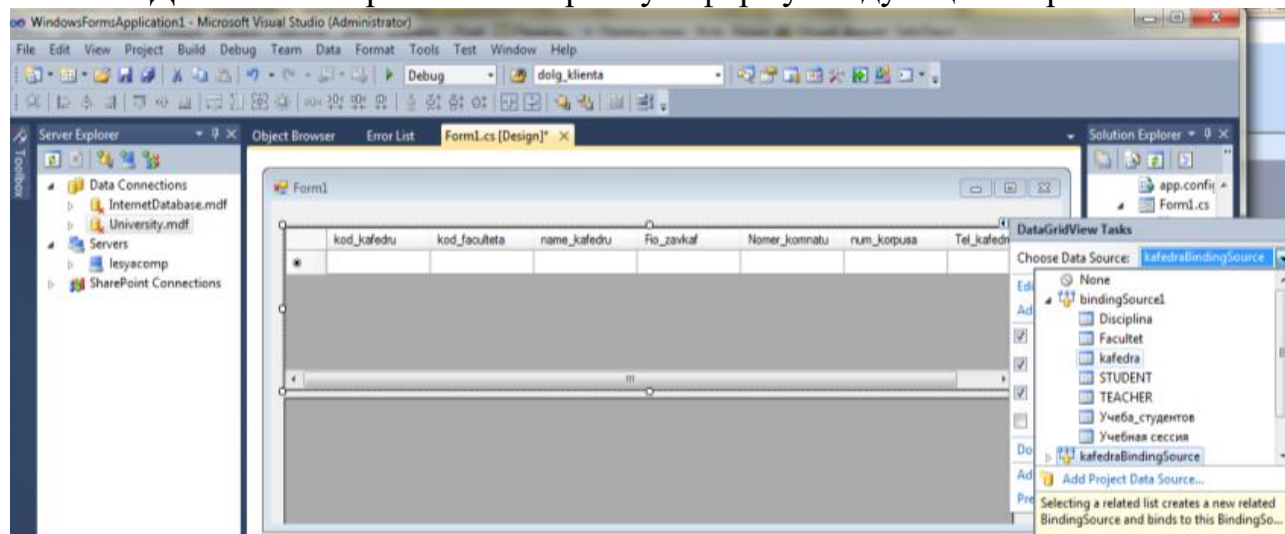


Сразу же возникает окно настройки компонента, которое определяет его возможности по редактированию данных: Включить редактирование (Enable Adding), Включить правку (Enable Editing), Включить удаление (Enable Deleting); возможность изменения последовательности столбцов: Включить возможность изменения порядка столбцов (Enable Column Reordering); а также возможность закрепления в контейнере-родителе.

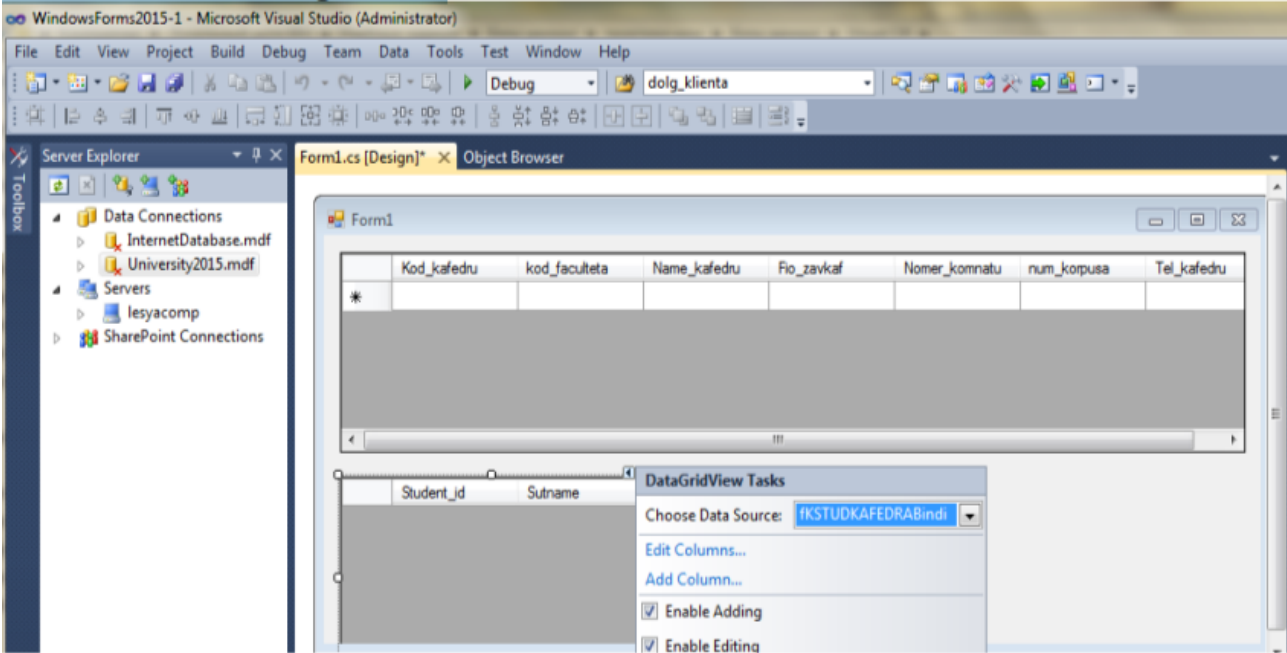
2.3.7. Для того чтобы компонент мог отображать данные, необходимо выбрать источник данных в выпадающем списке. Выбор выпадающего списка приводит к появлению следующего диалога



Выберите источник BindingSource1, откройте источник, выберите таблица Kafedra. Данный выбор заменит экранную форму следующим образом.



На рисунке видно, что появился еще один компоненты BindingSource и компонент TableAdapter, работающий с таблицей Кафедры. Обратите внимание, что в design-time или в процессе разработки данные из таблицы не отображаются! 2.3.8. Самостоятельно отобразите данные из связанной таблицы Студенты. Для этого добавьте ниже еще один компонент DataGridView и в качестве источника выберите соответствующую таблицу – Студенты. Однако, после выбора таблицы откройте ее список и выберите внешнюю связь fkStudKafedraBuildingSource.



Здесь в качестве источника данных выступает таблица Студенты, связанная с таблицей Кафедры. Такой выбор гарантирует выбор из таблицы Студенты только тех строк, которые связаны с текущей строкой в таблицы Кафедры. Также такой выбор гарантирует правильность обновления и удаления связанных данных. Например, на 1 кафедре учатся 4 студента, а на 11 – не одного.





Kod_kafedru	kod_faculteta	Name_kafedru	Fio_zavkaf	Nomer_komnatu	num_korpusa	Tel_kafedn
6	2	Компьютерных ...	Губачева	342	12	373728
7	2	Системная инж...	Ульшин	234	12	727287
10	3	Иностранных яз...	Краснопольский	123	2	762728
11	3	Компьютерных ...	Дядечев	703	9	563272

Student_id	Sutname	Surname	Stipend	Kurs	City	Birthday
*						

2.3.8. Перемещение по данным при помощи стрелочных клавиш является неудобным. Для упрощения навигации по данным существует компонент BindingNavigator. Поместим его на форме.

Данный компонент позволяет осуществлять навигацию между записями таблицы, добавлять и удалять строки таблицы. Возможности и внешний вид компонента можно настраивать, так как он представляет собой полосу меню ToolStripContainer. Свойством, определяющим таблицу, по которой производится навигация, является свойство BindingSource. В работе компонент выглядит следующим образом

2.3.9. Самостоятельно добавьте панель навигации для таблицы Студенты, с указанием в качестве источника внешний ключ fkStudKafedraBuildingSource. Запустите приложение и воспользуйтесь панелью навигации для перехода по строкам. Добавьте новую запись о студенте.

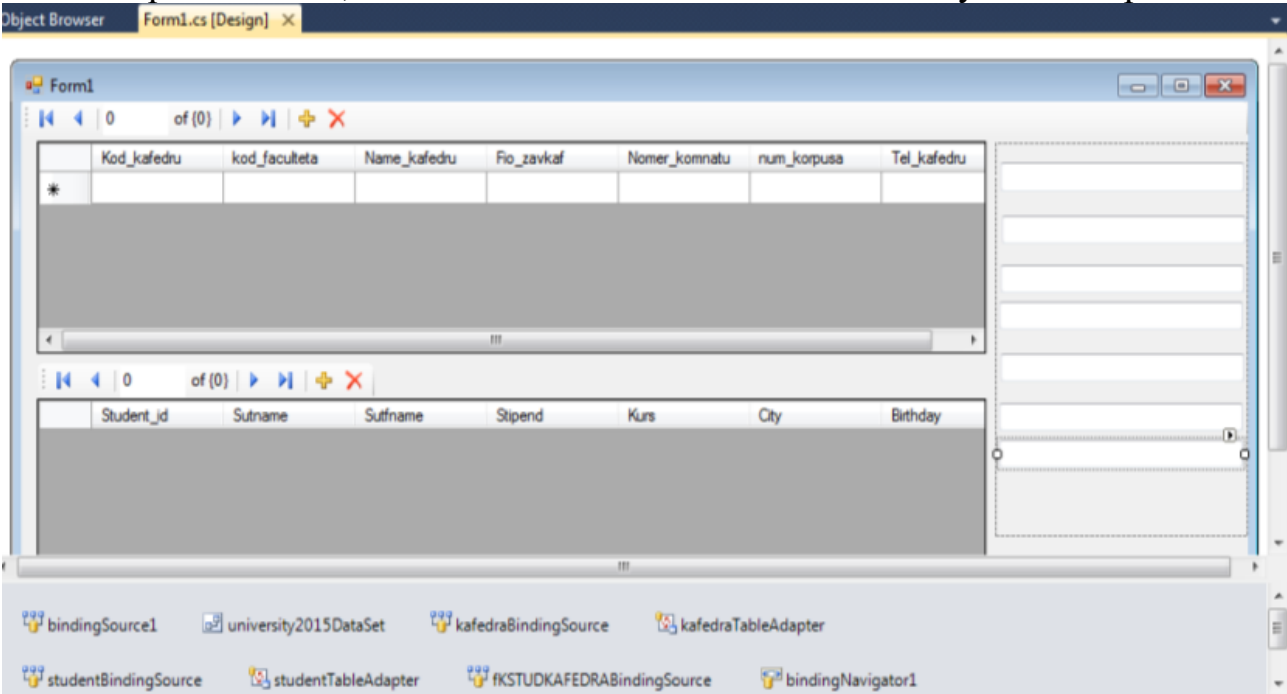
Kod_kafedru	kod_faculteta	Name_kafedru	Fio_zavkaf	Nomer_komnatu	num_korpusa	Tel_kafedn
1	1	Компьютерные ...	Соловьев	414	1	899028
2	1	Прикладная ма...	Кочевский	205	1	425262
3	1	Математическо...	Арлинский	61	1	627272
4	1	Информатики	Пождаев	404	1	738328
5	2	Автоматизации ...	Малахов	123	12	637327

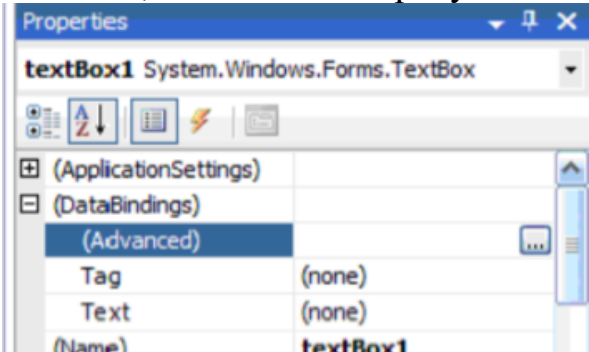
Student_id	Sutname	Surname	Stipend	Kurs	City	Birthday
13	Анна	Грешкина	450	2	Ростов-на-Дону	01.01.1989
15	Борис	Котовский	400	2	Ростов-на-Дону	27.05.1989
16	Петр	Комаров	300	2	Ростов-на-Дону	18.11.1989
17	Марина	Погребняк	353	2	Ростов-на-Дону	21.10.1989

2.3.10. Редактирование данных в ячейках компонента DataGridView при соответствующих настройках возможно, но неудобно и не рационально. В

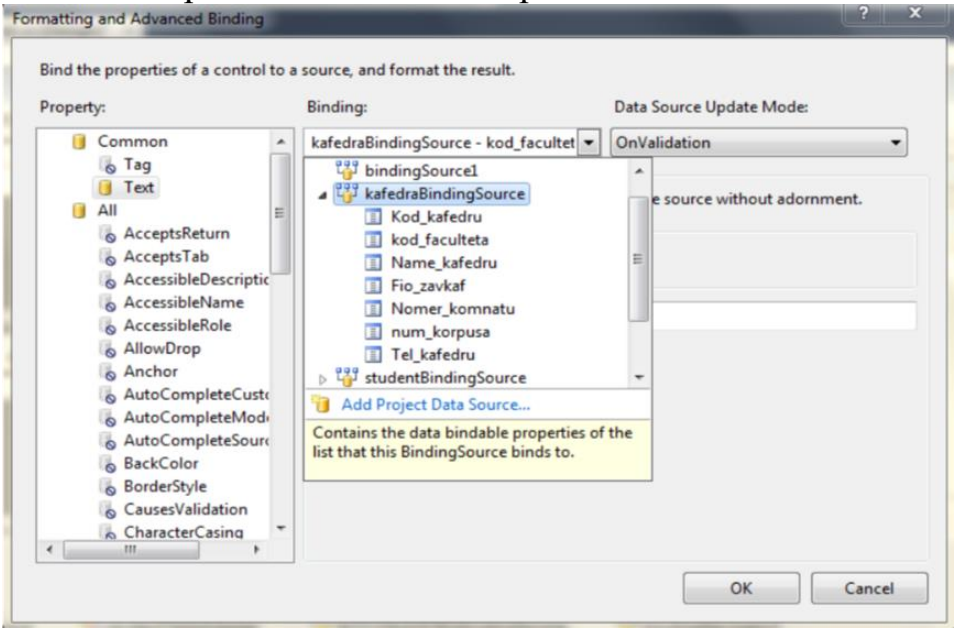
частности, трудно проверять введенные значения на ошибки. Поэтому для таблицы Кафедры сделаем экранную форму, позволяющую отображать данные в компонентах TextBox и редактировать их. Для этого разместим на форме контейнер типа Panel, а на нем 7 компонентов TextBox следующим образом.



2.3.11. Теперь необходимо осуществить привязку компонентов TextBox к соответствующим полям таблицы Кафедра. Для этого используем свойство из группы DataBindings - Advanced, показанное на рисунке.



Выбор данного свойства приводит к появлению диалога:



Данный диалог позволяет осуществить не только привязку данных, но также задать событие, в рамках которого будет проводиться обновление данных, а также форматирование данных при их выводе. Для верхнего компонента TextBox в выпадающем списке Binding выберем источником данных `kafedraBindingSource` и поле источника – `kod_kafedru`. Для среднего и нижнего компонента TextBox выберем тот же источник данных и поля `kod_faculteta`, `name_kafedru`, `fio_zafkaf`, `nomer_komnatu`, `num_korpusa`, `tel_kafedru` соответственно.

Разработанное приложение в итоге выглядит следующим образом:

2.3.12. Однако при внесении изменений все новые данные остаются только на форме. В БД они не сохраняются, и при повторном вызове приложения будут отсутствовать. Это происходит потому, что данные были загружены в объект `DataSet`, который представляет собой копию таблицы в памяти. Все действия выполняются с этой копией. Для того чтобы изменения отобразились в БД, необходимо выполнить метод `Update` класса `TableAdapter`. Таким образом, в разрабатываемом приложении необходимо разместить кнопку Обновить – элемент `Button` с панели `ToolBox` и записать обработчик события `Click` следующий программный код:

```
private void button1_Click(object sender, EventArgs e)
{
    kafedraTableAdapter.Update(university2015DataSet);
    studentTableAdapter.Update(university2015DataSet);
}
```

Данный код обновляет информацию в таблицах Кафедры и Студенты, предоставляемых источником данных. Данный метод является перегруженным, и его варианты позволяют обновлять как отдельную строку таблицы, так и группу строк.

*Примечание.* Чтобы запустить обработчик события `Click` достаточно два раза щелкнуть мышью на кнопке `Button`. В результате будет открыто окно кода – `Form1.cs` в области процедуры. Чтобы переименовать кнопку – измените свойство `Text`.

Запустите приложение введите новую запись о кафедре и нажмите кнопку Обновить. Закройте форму и вновь откройте ее. Ваши новые данные должны появиться в общем списке и никуда не исчезнуть.

## **ПРАКТИЧЕСКАЯ РАБОТА № 28**

**Тема: Программирование баз данных на языке C#. Технология ADO.NET.**

### **Соединение с базой данных**

**Цель работы:** научиться разрабатывать программы на языке C# с применением технологии ADO.NET для программирования баз данных.

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, MS Visual Studio, инструкции по выполнению работы.

### **Содержание работы:**

**Задание 1.** Для созданной базы данных Проектная организация самостоятельно создать клиентское приложение в среде Visual Studio C#. Установить соединение с разработанной базой данных через Server Explorer. Клиентское приложение должно содержать 1 диалоговую форму, в которой отобразить две связанные таблицы. На форму добавить текстовые поля для редактирования данных одной из таблиц. Клиентское приложение сохранить в вашей папке.

## **ПРАКТИЧЕСКАЯ РАБОТА № 29**

**Тема: Программирование баз данных на языке С#. Технология ADO.NET.**

### **Соединение с базой данных**

**Цель работы:** научиться разрабатывать программы на языке С# с применением технологии ADO.NET для программирования баз данных.

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, MS Visual Studio, инструкции по выполнению работы.

### **Содержание работы:**

**Задание 1.** Для созданной базы данных Компания по разработке программных продуктов самостоятельно создать клиентское приложение в среде Visual Studio С#. Установить соединение с разработанной базой данных через Server Explorer. Клиентское приложение должно содержать 1 диалоговую форму, в которой отобразить две связанные таблицы. На форму добавить текстовые поля для редактирования данных одной из таблиц. Клиентское приложение сохранить в вашей папке.

## ПРАКТИЧЕСКАЯ РАБОТА № 30

### Тема: Программное подключение к БД в Visual Studio с#. Создание запросов

**Цель работы:** научиться разрабатывать программы на языке С# и технологии ADO.NET для программирования баз данных. Соединение с базами данных, создание запросов

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, MS Visual Studio, инструкции по выполнению работы.

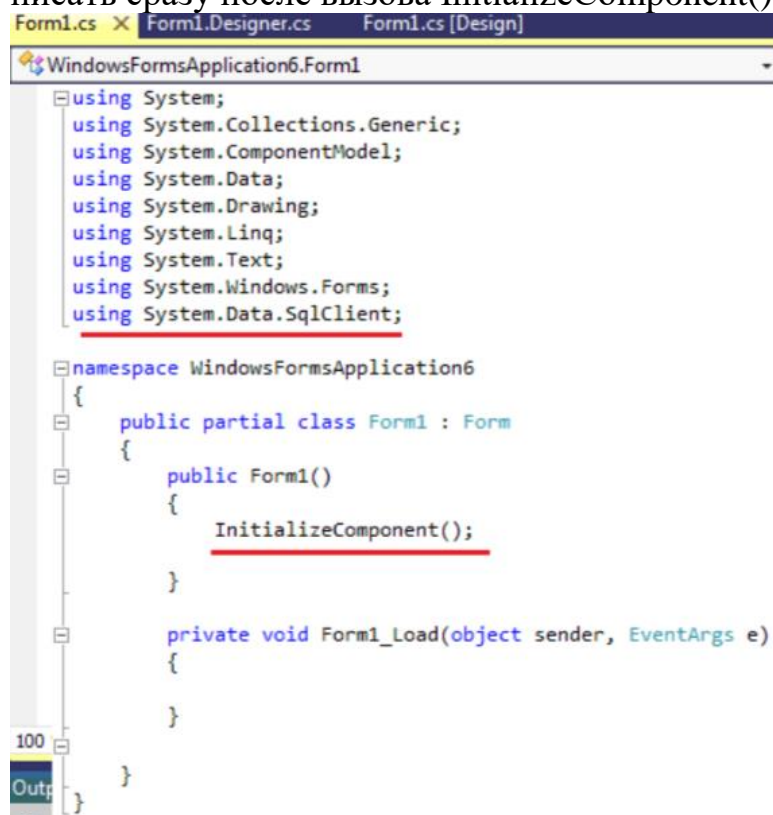
#### Содержание работы:

**Задание 1.** Создадим приложение для работы с БД

1. Создать БД Magazin.mdf.
2. Запустить Visual Studio 2022 и создать новый проект Windows Forms Application с использованием языка С#. Выполните в Visual Studio соединение с базой данных Magazin.mdf с помощью Server Explorer.
3. Размещаем на форме два элемента управления DataGridView (выберите на панели ToolBox вкладку Data или AllWindowsForm). Увеличиваем их в длину и размещаем в области формы.

Перейдите в код формы (для этого нажмите на пустую форму, нажмите правую клавишу и выберите View Code) и подключите пространство имен для MS SQL – using System.Data.SqlClient (просто напишите название этой библиотеки).

4. В конструкторе формы будем создавать подключение к базе данных. Для этого перейдите в область конструктора формы – область public Form1(). Код будем писать сразу после вызова InitializeComponent().



5. Строка соединения ConnectionString определяет параметры, необходимые для установления соединения с источником данных. Строка соединений при использовании мастеров генерируется средой, но можно писать эту строку вручную (желательно – во избежание неточностей и ошибок).

В конструкторе Form1 после InitializeComponent() создадим объект Connection:

```
string connetionString = null; SqlConnection sqlCnn;
```

В качестве строки соединения необходимо использовать строку:

```
connetionString = @"Data Source=.\SQLEXPRESS;  
AttachDbFilename=D:\Magazin.mdf;Integrated Security=True";
```

Затем создадим соединение:

```
sqlCnn = new SqlConnection(connetionString);
```

Теперь можно установить соединение, вызывая метод Open для объекта Connection: sqlCnn.Open();

Теперь создаем SqlDataAdapter для обращения к данным и выборки данных из БД:

```
SqlDataAdapter adapter = new SqlDataAdapter();
```

В качестве параметров объекта adapter необходимо передать два параметра CommandText и ConnectionString. ConnectionString – строка подключения, которую мы уже определили выше через переменную sqlCnn. Определим параметр CommandText. Этот параметр представляет собой обычный SQL-запрос на выборку.

Поместите код с объявлением строки запроса в область ниже создания объекта Connection:

```
SqlCommand sqlCmd; string sql = null;
```

Затем добавьте следующие строки после открытия соединения:

```
sql = "Select * From ТаблицаПоставщики";
```

```
sqlCmd = new SqlCommand(sql, sqlCnn);
```

```
adapter.SelectCommand = sqlCmd;
```

Последняя строка кода вызывает выполнение запроса на выборку. В нашем случае будут выбираться все поля из таблицы под именем «ТаблицаПоставщики» базы данных Magazin.mdf. Однако можно использовать и другие виды запросов.

6. Продолжим создание программы. Создаем объект DataSet:

```
DataSet ds = new DataSet();
```

Заполняем таблицу объекта ds данными из базы данных:

```
adapter.Fill(ds, "Поставщики");
```

таблицы под именем "Поставщики" нет . однако, при вызове метода adapter.Fill таблица может быть названа как угодно – ее содержимое будет представлять собой извлекаемую таблицу базы данных.

А теперь, связываем источник данных объекта DataGridView1 с таблицей под именем, который мы присвоили "Поставщики":

```
this.dataGridView1.DataSource = ds.Tables["Поставщики"].DefaultView;
```

При указании источника DataSource для объекта dataGridView1 мы ссылаемся именно на таблицу "Поставщики", которая была создана при вызове метода Fill. Таким образом, при заполнении таблиц их можно называть произвольным образом. Однако, для реальных проектов рекомендуется использовать настоящие названия таблиц, чтобы избежать путаницы и трудностей в сопровождении.

В конце с целью освобождения памяти и закрытия соединения с БД выполните следующие команды:

```
adapter.Dispose();
```

```
sqlCmd.Dispose();
```

```
sqlCnn.Close();
```

7.Теперь запустим созданное приложение. Если все сделано правильно, то на экранной форме отобразится содержимое таблицы «ТаблицаПоставщики».



8. Самостоятельно отобразить в объекте DataGridView2 таблицу «ТаблицаТовар» с выводом произвольных трех столбцов.

9.Создание запросов. В нашем примере это соответствующая строка:

```
adapter.SelectCommand = sqlCommand;
```

Рассматривая далее свойства SqlDataAdapter, отметим, что он может работать с командами Insert, Delete, Update и Select, то есть - этот элемент для многофункциональной работы с данными. Для каждого типа команд можно задать и соответствующий CommandType и исполняемый SQL-оператор.

Для выполнения выборки данных используется метод SelectCommand

Для добавления данных в БД используется метод InsertCommand

Для удаления данных из БД используется метод DeleteCommand

Для обновления данных в БД используется метод UpdateCommand

Введите в нашу программу перед строками закрытия соединения пример добавления записей в таблицу базы данных:

```
//добавление
```

```
sql = "INSERT INTO ТаблицаПоставщики (НомерЛицензии, Город, Дом, Улица, БанковскиеРеквизиты, ИНН, Телефон) "
```

```
+ "VALUES ('400000', 'Ростов-на-Дону', '44', 'московская','47484848','5858', '56-86-90')";
```

```
sqlCmd = new SqlCommand(sql, sqlCnn);
```

```
adapter.InsertCommand = sqlCmd;
```

10.Запустите программу на выполнение и вы не увидите добавленных данных. Для того чтобы все заработало необходимо применить метод ExecuteNonQuery().

Метод ExecuteNonQuery можно использовать для выполнения различных операций, в том числе для операций с каталогами, изменения данных в базе данных (выполнения предложений UPDATE, INSERT или DELETE...).

Для предложений UPDATE, INSERT и DELETE возвращаемым значением является количество измененных строк, для всех остальных предложений возвращается значение -1. Возможное исключение - InvalidOperationException - подключение не существует или подключение не открыто.

Поэтому добавьте следующую строку кода:

```
sqlCmd.ExecuteNonQuery();
```

Для сохранения измененных данных используется метод sqlDataAdapter Update(). В период создания DataAdapter при применении настраиваемых свойств InsertCommand, UpdateCommand и DeleteCommand. При вызове метода Update мы выполняем по существу три эти SQL предложения. Метод вызывает соответствующие инструкции INSERT, UPDATE или DELETE для каждой вставленной, обновленной или удаленной строки в DataSet.

Поэтому добавьте следующие строки кода:

```
adapter.Update(ds, "Поставщики");
```

```
this.dataGridView1.Update();
```

В последней строке мы обновили также таблицу dataGridView1. Выполните программный код и убедитесь, что все заработало.

*Самостоятельно:*

1. Выполните sql-запрос на удаление всех строк таблицы Поставщики проживающих в городе Шахты. Разместите данный запрос перед запросом на добавление строк.



2. Выполните sql-запрос на добавление новой строки данных таблицы Поставщики. Разместите его ниже первого запроса на добавление.
3. Выполните sql-запрос на удаление всех строк таблицы Товары с названием «KLEO S60».
4. Выполните sql-запрос на добавление строки таблицы Товары с названием «KLEO S60» с произвольным номером артикля.

11. Выполним еще один запрос, который будет создавать в нашей базе данных Magazin.mdf новую таблицу ТаблицаНакладная:

//создание новой таблицы

```
sql = "CREATE TABLE ТаблицаНакладная (НомерНакладной INT NOT NULL,
ДатаВыдачи datetime, НомерТовара INT, "+ "НомерлицензииПоставщика INT,
Цена money, Количество INT) ";
```

```
sqlCmd = new SqlCommand(sql, sqlCnn);
```

```
int Uspeshnoelzmenenie= sqlCmd.ExecuteNonQuery();
```

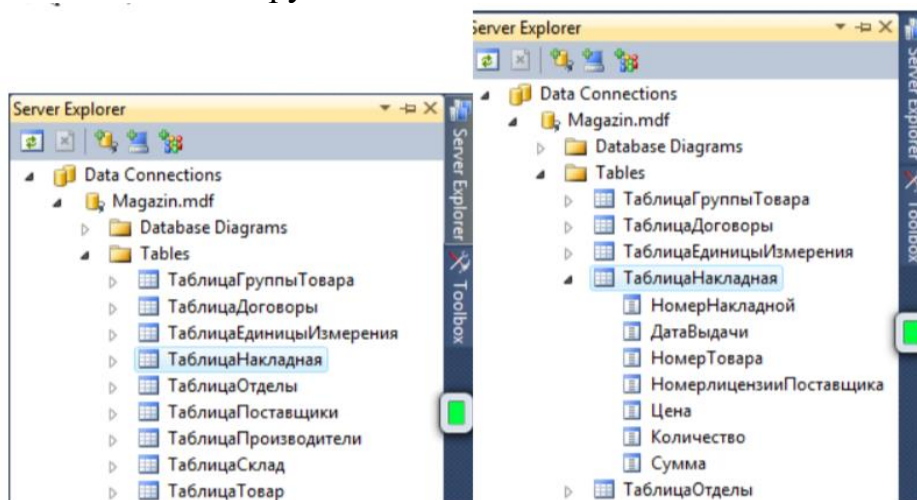
```
if (Uspeshnoelzmenenie != 0) {
```

```
    MessageBox.Show ("Изменения внесены в БД в виде создания таблицы"); }
else {
```

```
    MessageBox.Show("Не удалось внести изменения в БД связанных с созданием
таблицы"); }
```

12. Запускаем приложение, затем переходим в Server Explorer, нажимаем кнопку

Обновить  на панели инструментов – в базе появится новая таблица



13. Для добавления нового столбца «Сумма» строка sql должна иметь следующий вид:

// Добавление столбцов

```
sql = "ALTER TABLE ТаблицаНакладная ADD Сумма money";
```

```
sqlCmd = new SqlCommand(sql, sqlCnn);
```

```
sqlCmd.ExecuteNonQuery();
```

Запускаем приложение, затем переходим в Server Explorer, нажимаем кнопку

Обновить .

14. Для удаления таблицы «ТаблицаНакладная» запускаем приложение, содержащее следующую строку:

// удаление таблицы

```
sql = "DROP TABLE ТаблицаНакладная";
```

```
sqlCmd = new SqlCommand(sql, sqlCnn);
```

```
sqlCmd.ExecuteNonQuery();
```

В Server Explorer видно, что таблица полностью исчезла из БД.

15. // подсчет количества Поставщиков проживающих в городе Ростове

```
sql = "SELECT COUNT(*) FROM ТаблицаПоставщики
```

```
WHERE Город='Ростов-на-Дону';
```

```
sqlCmd = new SqlCommand(sql, sqlCnn);
```

```
string KolvoPostav=Convert.ToString(sqlCmd.ExecuteNonQuery());
```

```
MessageBox.Show("Количество поставщиков="+ KolvoPostav);
```

*Самостоятельно* посчитать количество товара, максимальную, минимальную и среднюю цену товара. Вывести результаты в текстовые поля.

## **ПРАКТИЧЕСКАЯ РАБОТА № 31**

**Тема: Программное подключение к БД в Visual Studio c#. Создание запросов**

**Цель работы:** научиться разрабатывать программы на языке C# и технологии ADO.NET для программирования баз данных. Соединение с базами данных, создание запросов

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, MS Visual Studio, инструкции по выполнению работы.

**Содержание работы:**

**Задание 1.** Для созданной базы данных Торговая фирма самостоятельно создать новое клиентское приложение в среде Visual Studio C#. Установить соединение с разработанной базой данных через Server Explorer. Клиентское приложение должно содержать 1 диалоговую форму. Программно выполнить соединение с базой данных и вывести данные трех таблиц. Создать запросы SELECT, INSERT, UPDATE, DELETE, CREATE TABLE, DROP TABLE, SELECT COUNT. Клиентское приложение сохранить в папке

## **ПРАКТИЧЕСКАЯ РАБОТА № 32**

**Тема: Программное подключение к БД в Visual Studio c#. Создание запросов**

**Цель работы:** научиться разрабатывать программы на языке C# и технологии ADO.NET для программирования баз данных. Соединение с базами данных, создание запросов

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, MS Visual Studio, инструкции по выполнению работы.

**Содержание работы:**

**Задание 1.** Для созданной базы данных Учебный центр самостоятельно создать новое клиентское приложение в среде Visual Studio C#. Установить соединение с разработанной базой данных через Server Explorer. Клиентское приложение должно содержать 1 диалоговую форму. Программно выполнить соединение с базой данных и вывести данные трех таблиц. Создать запросы SELECT, INSERT, UPDATE, DELETE, CREATE TABLE, DROP TABLE, SELECT COUNT. Клиентское приложение сохранить в папке

## ПРАКТИЧЕСКАЯ РАБОТА № 33

### Тема: Управление событиями в программировании БД

**Цель работы:** научиться программно подключаться к базе данных с использованием событий Connection ADO.NET.

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, MS Visual Studio, инструкции по выполнению работы.

#### Содержание работы:

**Задание 1.** 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 БД BDTur\_firmSQL.mdf. На форме разместим объект dataGridView со свойством Dock=Top, кнопку 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.

Подключаем пространство имен для работы с базой в файле Form.cs:

using System.Data.SqlClient;

В классе формы создадим строки connectionString и commandText:

```
string connectionString = @"Data Source=.\SQLEXPRESS; AttachDbFilename=" +  
    @"D:\BMM\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);  
    dataGridView1.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. В методе conn\_Disposed просто будем выводить текстовое сообщение в надпись Label2:

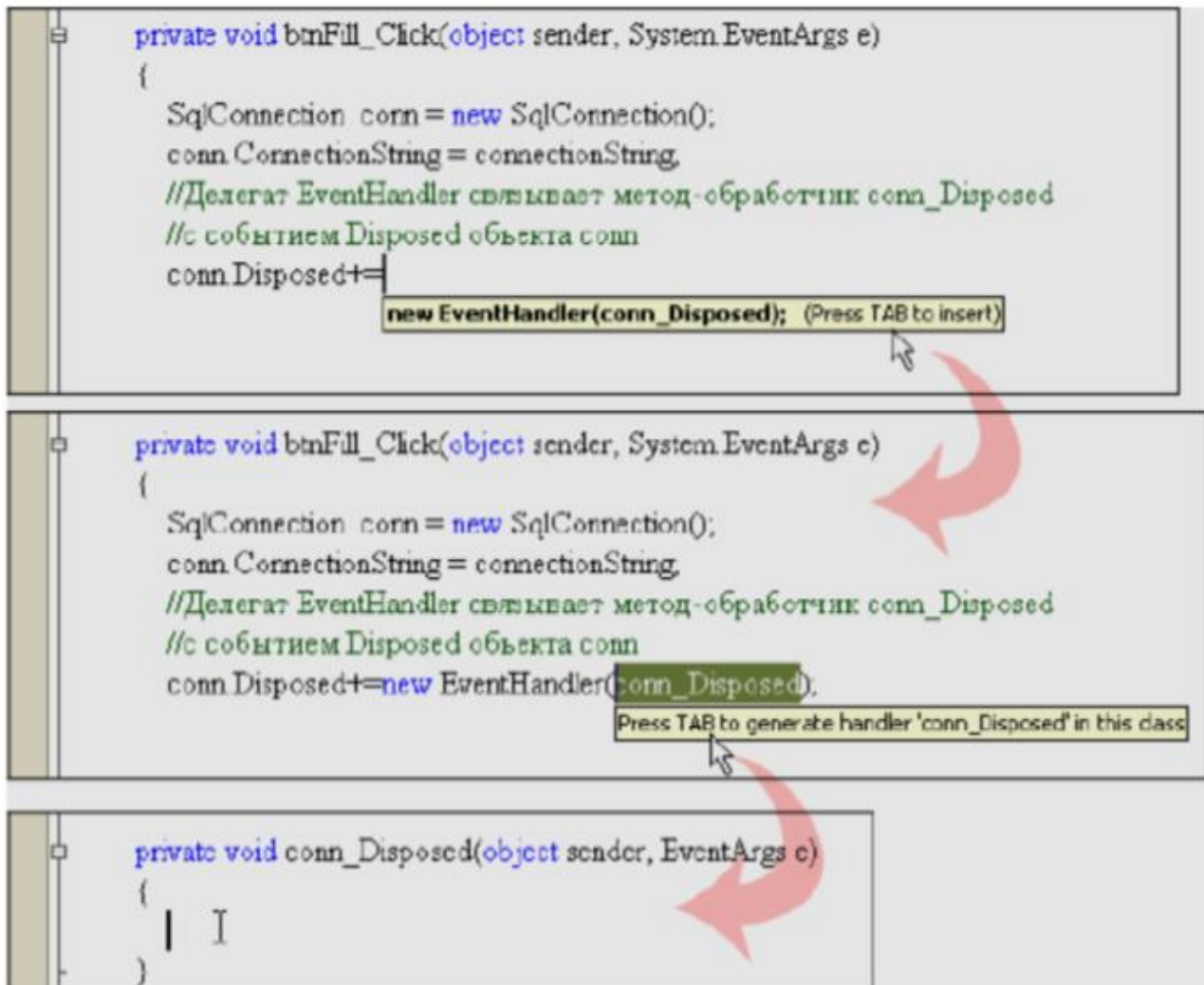
```
private void conn_Disposed(object sender, EventArgs e)  
{  
    label2.Text+="Событие Dispose";  
}
```

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 состояния объекта `conn` было закрытым (Closed). В момент открытия текущим состоянием становится Open, а предыдущем – Closed. Этому соответствуют первые две строки, выведенные в надпись. После закрытия соединения (вызов метода `Dispose`) текущим состоянием становится закрытое (Closed), а предыдущим – открытое (Open). Этому соответствуют последние две строки, выведенные в надпись.



## Задание 2. Самостоятельно выполнить:

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 скрытые 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).

**Задание 3.** Для созданной базы данных Проектная организация самостоятельно создать новое клиентское приложение в среде Visual Studio C#.

В начале должна появляться 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 скрытые наборы вкладок формы. В каждой отобразить объекты DataGridView, в которых отобразить все таблицы базы данных.

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).

Клиентское приложение сохранить в папке

## ПРАКТИЧЕСКАЯ РАБОТА № 34

### Тема: Управление событиями в программировании БД

**Цель работы:** научиться программно подключаться к базе данных с использованием событий Connection ADO.NET.

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, MS Visual Studio, инструкции по выполнению работы.

#### Содержание работы:

**Задание 1.** Для созданной базы данных Университет самостоятельно создать новое клиентское приложение в среде Visual Studio C#.

В начале должна появляться 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 скрытые наборы вкладок формы. В каждой отобразить объекты DataGridView, в которых отобразить все таблицы базы данных.

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).

Клиентское приложение сохранить в папке

**Задание 2.** Для созданной базы данных Торговая фирма самостоятельно создать новое клиентское приложение в среде Visual Studio C#.

В начале должна появляться 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 скрытые наборы вкладок формы. В каждой отобразить объекты DataGridView, в которых отобразить все таблицы базы данных.

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).

Клиентское приложение сохранить в папке



## ПРАКТИЧЕСКАЯ РАБОТА № 35

### Тема: Определение класса для подключения к базе данных. Создание многооконного интерфейса

**Цель работы:** научиться создавать классы для подключения к базе данных, использовать классы в нескольких диалоговых окнах для отображения данных из разных объектов базы данных.

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, MS Visual Studio, инструкции по выполнению работы.

#### Содержание работы:

**Задание 1.** К базе данных Университет создать многооконный интерфейс.

1. Откройте Visual Studio и создайте новое Windows-приложение с использованием Visual C# (выполните меню File/ New / Project/ выберите Visual C# , затем выберите приложение Windows – WindowsForms).

2. Затем скопируйте созданную ранее базу данных University в текущую директорию вашего приложения. Для этого:

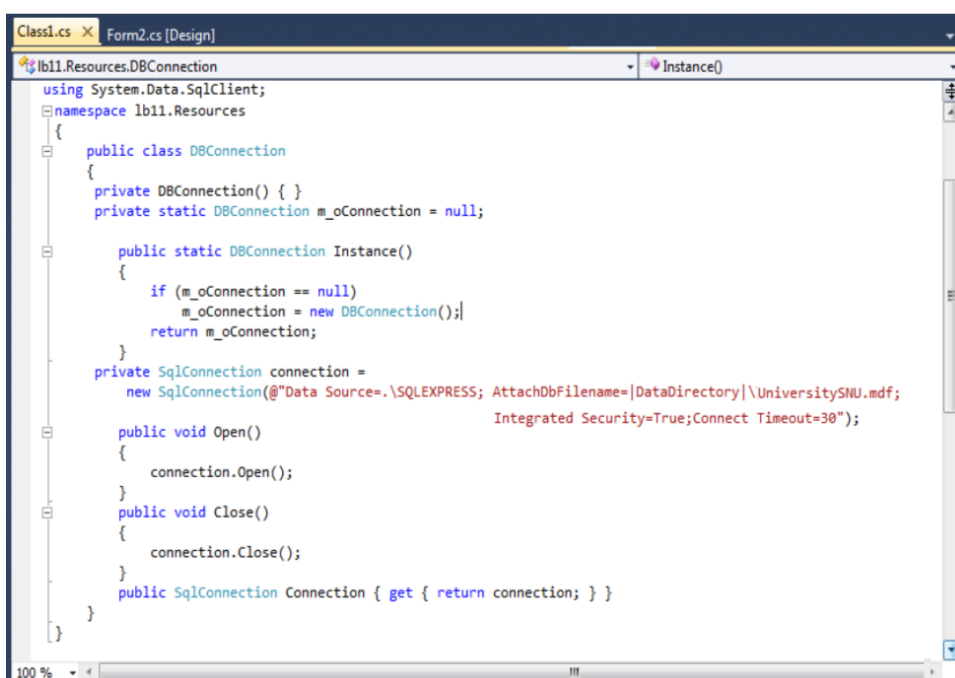
- остановите MS SQL Server (для этого выполните Пуск/Программы/Microsoft SQL Server 2022/ Средства настройки / Диспетчер конфигурации SQL Server, затем выберите сервер SQL SERVER AGENT (SQLEXPRESS) и остановите его).

- скопируйте базу данных в папку приложения (для этого зайдите в папку, где храниться ваша база данных, например в системном каталоге C:/Program Files/Microsoft SQL Server/MSSQL10.SQLEXPRESS/MSSQL/DATA, затем скопируйте два файла University.mdf и University.ldf и поместите их в папку с вашим приложением приложения в вложенную папку /bin/Debug).

- запустите MS SQL Server (для этого в Диспетчере конфигурации SQL Server выберите сервер SQL SERVER AGENT (SQLEXPRESS) и запустите его).

3. Затем создадим универсальное правило подключения к базе данных. Для этого добавьте в ваш проект новый класс. Для этого выполните команду меню Project/ Add Class.

4. Объявим класс DBConnection, с помощью которого будем создавать новое подключение к базе данных. В классе создадим два метода – открытие соединения и закрытие соединения. Листинг кода класса:



```
using System.Data.SqlClient;

namespace lb11.Resources
{
    public class DBConnection
    {
        private DBConnection() { }
        private static DBConnection m_oConnection = null;

        public static DBConnection Instance()
        {
            if (m_oConnection == null)
                m_oConnection = new DBConnection();
            return m_oConnection;
        }

        private SqlConnection connection =
            new SqlConnection(@"Data Source=.\SQLEXPRESS; AttachDbFilename=|DataDirectory|\UniversitySNU.mdf;
                               Integrated Security=True;Connect Timeout=30");

        public void Open()
        {
            connection.Open();
        }

        public void Close()
        {
            connection.Close();
        }

        public SqlConnection Connection { get { return connection; } }
    }
}
```

5. Далее из вашей формы сделаем форму приветствия и

разместим на нее кнопки, при нажатии на которых будут открываться другие формы для отображения на экран содержимого каждой таблицы в базе данных Университет, а именно – Факультеты, Кафедры, Студенты и Преподаватели.

6. В первой форме измените заголовок формы с помощью панели свойств на – «Приложение для работы с базой данных Университет». Затем добавьте 5 кнопок Butoon на форму с помощью панели Toolbox. Измените подписи кнопок на - Факультеты, Кафедры, Студенты, Преподаватели и Выход.

7. Откройте листинг формы Form1.cs. Добавьте библиотеку для работы с базами данных Microsoft SQL Server, а именно напишите в разделе описаний:

```
using System.Data.SqlClient;
```

8. Добавьте ссылку на наш созданный класс. Класс находится в ссылках проекта, поэтому необходимо написать в разделе описаний:

```
using Ваше_имя_проекта.Resources;
```

9. Далее организуем, что бы при загрузке нашей первой формы выполнялось подключение к базе данных. Для этого впишите после InitializeComponent() формы следующее:

```
public Form1() {  
    InitializeComponent();  
    DBConnection.Instance().Open();    }
```

10. Далее организуем, что бы при выходе из нашей первой формы выполнялось отключение от базы данных. Для этого создайте процедуру на нажатие кнопки Выход. Внутри процедуры впишите следующее:

```
DBConnection.Instance().Close();  
Close();
```

11. Сохранитесь и запустите проект. У вас не должно появляться ошибок. Первая форма с 5 кнопками должна появиться на экране, и это значит, что соединение с базой данных произошло успешно!!

12. Самостоятельно добавьте в проект еще 4 пустых формы (для этого выполните команды Project/ Add Windows Form). Измените для каждой формы заголовки на Факультеты, Кафедры, Студенты и Преподаватели.

13. Добавим обработчики на нажатие каждой кнопки на первой форме. Например, при нажатии кнопки Факультеты будем выводить форму Факультеты. Для этого перейдите на первую форму, два раза щелкните мышью на кнопке Факультеты и впишите туда следующие команды:

```
private void button1_Click(object sender, EventArgs e)  
{  
    Form2 f2 = new Form2();  
    f2.Show();  
}
```

14. Самостоятельно измените три других кнопки.

15. На каждую форму разместите объекты DataGridView и BindingNavigator.

16. Самостоятельно в листингах всех форм добавьте библиотеку Microsoft SQL Server и ссылку на класс проекта.

17. Теперь, с помощью использования класса мы выполним, например, отображение данных таблицы Факультеты в форме Факультеты.

18. Перейдите в листинг формы Forms2.cs. В разделе описания формы зададим три объекта – это адаптер SqlDataAdapter, набор данных DataSet и источник данных BindingSource.

```
public partial class Form2 : Form
{
    SqlDataAdapter dataAdapter1 = new SqlDataAdapter();
    DataSet ds1 = new DataSet();
    BindingSource bs = new BindingSource();
}
```

19. Затем после инициализации формы впишем программный код, для отображения данных таблицы Факультеты в объект **DataGridView**.

```
public Form2()
{
    InitializeComponent();
    dataAdapter1.SelectCommand = new SqlCommand("Select * from Facultet", DBConnection.Instance().Connection);
    dataAdapter1.Fill(ds1, "Facultet");
    bs.DataSource = ds1;
    bs.DataMember = "Facultet";
    dataGridView1.DataSource = bs;

    bindingNavigator1.BindingSource = bs;
}
```

20. Сохраните проект и запустите его на выполнение. Нажмите кнопку факультеты и у вас должны быть отображены все данные таблицы Факультеты. При чем по записям таблицы можно двигаться с помощью панели навигации.

21. Аналогично отобразите на формах Кафедры, Студенты и Преподаватели соответствующие таблицы с данными.

22. Сохраните проект и отобразите результаты его работы преподавателю.

## **ПРАКТИЧЕСКАЯ РАБОТА № 36**

### **Тема: Определение класса для подключения к базе данных. Создание многооконного интерфейса**

**Цель работы:** научиться создавать классы для подключения к базе данных, использовать классы в нескольких диалоговых окнах для отображения данных из разных объектов базы данных.

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, MS Visual Studio, инструкции по выполнению работы.

#### **Содержание работы:**

**Задание 1.** Для созданной базы данных Учебный центр самостоятельно создать новое клиентское приложение в среде Visual Studio C#. Клиентское приложение сохранить в своей папке. Разместите созданную ранее базу данных в текущую директорию вашего приложения. Программно установите подключение к базе данных с помощью классов. Приложение должно содержать столько форм, сколько таблиц в базе данных. Создать одну главную диалоговую форму, на которую разместить кнопки с названиями таблиц. При нажатии на кнопки должны открываться соответствующие диалоговые формы. Для каждой таблицы добавить панель навигации.

**Задание 2.** Для созданной базы данных Торговая фирма самостоятельно создать новое клиентское приложение в среде Visual Studio C#. Клиентское приложение сохранить в своей папке. Разместите созданную ранее базу данных в текущую директорию вашего приложения. Программно установите подключение к базе данных с помощью классов. Приложение должно содержать столько форм, сколько таблиц в базе данных. Создать одну главную диалоговую форму, на которую разместить кнопки с названиями таблиц. При нажатии на кнопки должны открываться соответствующие диалоговые формы. Для каждой таблицы добавить панель навигации.

## **ПРАКТИЧЕСКАЯ РАБОТА № 37**

### **Тема: Определение класса для подключения к базе данных. Создание многооконного интерфейса**

**Цель работы:** научиться создавать классы для подключения к базе данных, использовать классы в нескольких диалоговых окнах для отображения данных из разных объектов базы данных.

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, MS Visual Studio, инструкции по выполнению работы.

#### **Содержание работы:**

**Задание 1.** Для созданной базы данных Проектная организация самостоятельно создать новое клиентское приложение в среде Visual Studio C#. Клиентское приложение сохранить в своей папке. Разместите созданную ранее базу данных в текущую директорию вашего приложения. Программно установите подключение к базе данных с помощью классов. Приложение должно содержать столько форм, сколько таблиц в базе данных. Создать одну главную диалоговую форму, на которую разместить кнопки с названиями таблиц. При нажатии на кнопки должны открываться соответствующие диалоговые формы. Для каждой таблицы добавить панель навигации.

**Задание 2.** Для созданной базы данных Компания по разработке программных продуктов самостоятельно создать новое клиентское приложение в среде Visual Studio C#. Клиентское приложение сохранить в своей папке. Разместите созданную ранее базу данных в текущую директорию вашего приложения. Программно установите подключение к базе данных с помощью классов. Приложение должно содержать столько форм, сколько таблиц в базе данных. Создать одну главную диалоговую форму, на которую разместить кнопки с названиями таблиц. При нажатии на кнопки должны открываться соответствующие диалоговые формы. Для каждой таблицы добавить панель навигации.

## **ПРАКТИЧЕСКАЯ РАБОТА № 38**

### **Тема: Параметризованные запросы на C# ADO.NET**

**Цель работы:** научиться разрабатывать программы языке C# с использованием параметризованными запросами.

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, MS Visual Studio, инструкции по выполнению работы.

#### **Справочный материал:**

Данная практическая работа выполняется в проекте, выполненном в Практической работе №35. В этом проекте был использован специальный класс для соединения с базой данных Univesity.mdf.

Откройте свой проект, основанный на классе. В этом проекте было выполнено соединение с базой данных и созданы 5 диалоговых форм. Первая форма – форма приветствия, а остальные формы – форма Факультета, Кафедры, Преподавателей и Студентов. В каждой форме данные отображаются в объектах DataGridView.

Однако, чаще на практике данные из базы данных не просто нужно выводить и просматривать в режиме таблиц (так как было у нас выполнено), а данные выводятся в полях или других элементах управления, возможно выполняется проверка на вводимые данные, определенные вычисления и т.д. Затем для таких данных создают кнопки управления для добавления данных в базу данных, удаления, изменения, создания новой записи, переходы по записям.

Выполним в этой работе именно эту задачу, путем изучения новой темы – параметризованных запросов

#### **Содержание работы:**

##### **Задание 1.** Создание диалоговых форм.

1. Создание диалоговой формы Факультеты.

1.1. Откройте Visual Studio и откройте приложение созданное в Практической работе № 35 сохраненное в вашей папке.

1.2. Добавьте кнопку Button на главную форму с помощью панели Toolbox и измените ее подпись на «Ввод данных в таблицу Факультет».

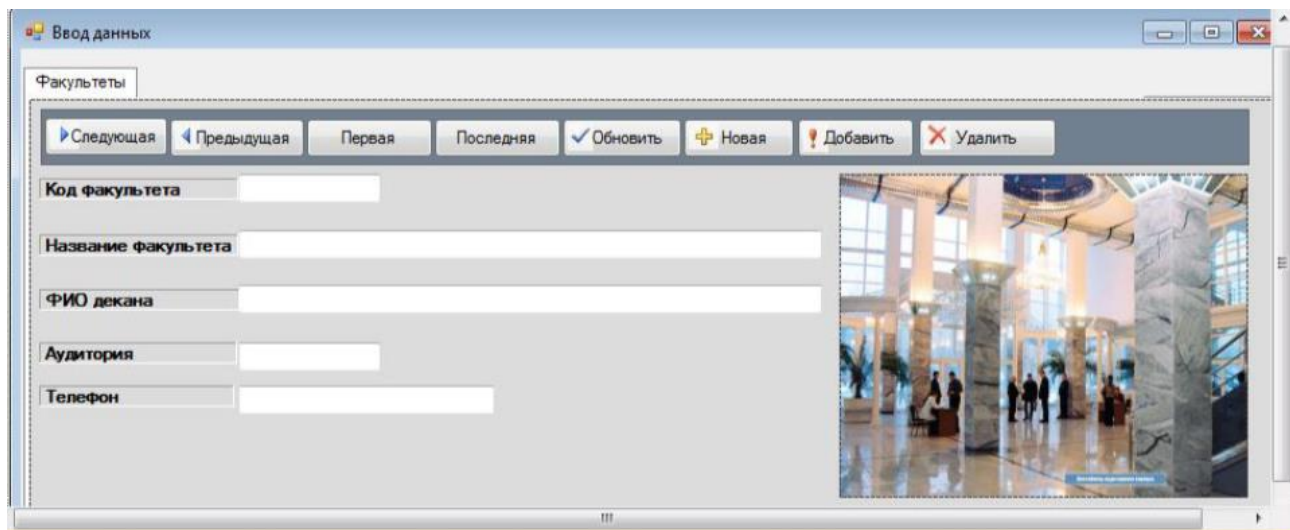
1.3. Добавьте в проект еще одну форму (у вас она будет уже шестая). В этой форме будем отображать данные таблицы факультет.

1.4. Вернитесь в главную форму и создайте обработчик на нажатие кнопки «Ввод данных в таблицу Факультет», с помощью которого будет открываться шестая форма.

1.5. На новую пустую форму последовательно добавьте 5 текстовых полей TextBox и 5 подписей Label.

1.6. Измените подписи Label на следующие: Код факультета, Название факультета, ФИО декана, Аудитория и Телефон деканата.

1.7. Текстовые поля разместите напротив каждой надписи, так как показано на рис. 1.



1.8. Далее добавьте на форму 8 новых кнопок Button, измените их подписи на Следующая, Предыдущая, Первая, Последняя, Обновить, Новая, Добавить и Удалить.

1.9. Далее, выполним соединение данной формы с таблицей Факультет. Для этого добавьте в проект этой формы две библиотеки  
using System.Data.SqlClient и using lb11.Resources.

1.10. Добавьте адаптер SqlDataAdapter и набор данных DataSet:

```
public partial class Form6 : Form
{
    SqlDataAdapter dataAdapter1 = new SqlDataAdapter();
    DataSet ds1 = new DataSet();
}
```

1.11. Добавьте переменные для выполнения запросов:

```
string Sql;
SqlCommand myCommand;
int UspeshnoeIzmenenie;
```

1.12. Далее после инициализации формы впишем программный код, для отображения данных таблицы Факультеты в текстовые поля:

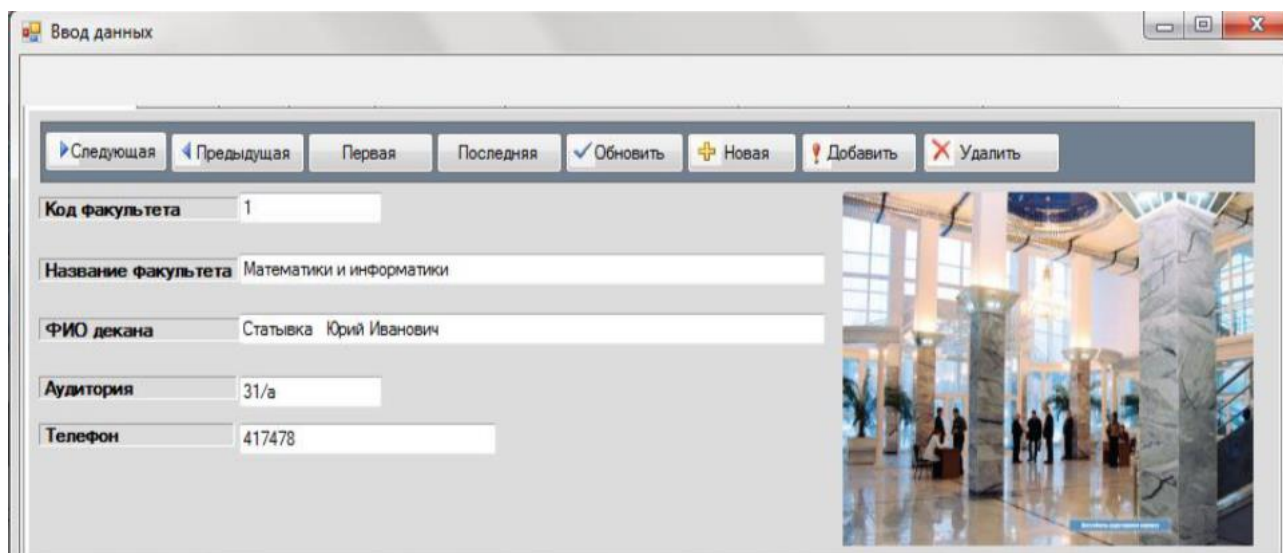
```
public Form6()
{
    InitializeComponent();
    dataAdapter1.SelectCommand = new SqlCommand("Select * from Facultet", DBConnection.Instance().Connection);
    dataAdapter1.Fill(ds1, "Facultet");

    // заполнение текстовых полей данными таблицы Факультет
    textBox1.DataBindings.Add("Text", ds1, "Facultet.Kod_faculteta");
}
```

1.13. Аналогично, самостоятельно заполните все остальные поля формы.

1.14. Форма практически готова. Сохранитесь и запустите на выполнение проект. На главной форме нажмите кнопку «Ввод данных в таблицу Факультет», вам должна открыться наша новая форма и в текстовых полях будет отображаться первая запись из таблицы Факультет.





1.15. Итак, первая часть работы выполнена. Однако, как вы видите, мы не можем перейти на следующую запись и на предыдущую, не можем ничего изменять и сохранять эти изменения в базе данных. Для выполнения этих действий, мы будем использовать параметризованные запросы.

## 2. Создание кнопок перехода по записям

Итак, выполним простейшие действия для перехода по записям таблицы Факультет и отображение соответствующих данных в полях нашей формы. Управление просмотром записей и обновлением элементов управления, связанных с данными, осуществляется на уровне источника данных при помощи объекта Currency Manager.

В этом случае источник данных ссылается на одно- или двухмерное хранилище данных, такое, как объект DataTable, DataView, массив или набор.

Объект DataSet, содержащий несколько объектов DataTable, способен поддерживать несколько источников данных.

Объект Currency Manager «следит» за положением текущей записи в источнике данных.

Приложение может использовать несколько источников данных одновременно, при этом каждый из них поддерживает собственный объект Currency Manager.

Поскольку форма способна отображать содержимое нескольких источников данных одновременно, любая форма управляет связанными с ними объектами Currency Manager посредством главного объекта — BindingContext.

Этот объект упорядочивает объекты Currency Manager, связанные со всеми источниками данных, и предоставляет к ним доступ.

Так, свойство BindingContext формы позволяет управлять положением текущей записи в любом источнике данных.

Чтобы получить доступ к объекту Currency Manager для некоторого источника данных, вызовите свойство BindingContext, передав ссылку на объект этого источника данных, например:

```
this.BindingContext[DataSet1.Customers]
```

Этот пример предполагает наличие таблицы Customers в объекте DataSet с именем DataSet1, содержащимся в текущей форме.

Для перемещения между записями устанавливают свойство Position соответствующего объекта BindingContext, как показано ниже:

```
//Перемещение по записям источника данных, связанного с формой
```



```
// Сделать первую запись источника данных текущей,
this.BindingContext[DataSet1, "Customers"].Position = 0;
// Переместить текущую запись на одну позицию вперед,
this.BindingContext[DataSet1, "Customers"].Position ++;
// Переместить текущую запись на одну позицию назад,
this.BindingContext[DataSet1, "Customers"].Position --;
// Сделать пятую запись источника данных текущей,
this.BindingContext[DataSet1, "Customers"].Position - 4;
// Перейти к последней записи,
this.BindingContext[DataSet1, "Customers"].Position =
DataSet1.Tables["Customers"].Rows.Count - 1;
```

Поскольку в .NET Framework значение свойства Position не может быть меньше нуля или больше верхней границы набора, при попытке выбрать запись за пределами источника данных ошибка не возникает — такая команда попросту игнорируется. Тем не менее, следует добавить к приложению код, уведомляющий пользователей (возможно, посредством сообщений) о достижении границ источника данных.

Итак, самостоятельно создайте обработчики для 4-х кнопок «Следующая», «Предыдущая», «Первая» и «Последняя», аналогично приведенным примерам выше, но с учетом наших обозначений имен объектов и с учетом, что данные необходимо отобразить из таблицы Facultet.

Сохраните проект и запустите программу на выполнение. Откройте форму Ввод данных Факультета и понажимайте на все кнопки перехода.

У вас все кнопки должны верно работать, иначе исправьте ошибки и вновь проверьте на правильность работы приложения.

### 3. Использование параметризованных запросов для создания управляющих кнопок

Следовательно, необходимо привязывать вводимые значения к элементам пользовательского интерфейса, например, к текстовым полям. Но это означает, что параметры строки запроса будут неизвестны до тех пор, пока пользователь не введет соответствующие значения. Например, для метода ExecuteNonQuery строка commandText имела следующий вид:

```
myCommand.CommandText = "UPDATE Туристы SET Фамилия = 'Сергеева'
WHERE Кодтуриста = 3";
```

Если создадим приложение, где пользователь будет вводить фамилию и код туриста, то мы не можем заранее указать, какие это будут значения.

Логически запрос можно представить примерно так:

```
myCommand.CommandText = " UPDATE Туристы SET
Фамилия = 'Какая-то_фамилия,_которую_введет_пользователь'
WHERE Кодтуриста = 'Какой-то_код,_который_введет_пользователь' ";
```

Для решения таких задач, которые возникли еще в самом начале разработки языка SQL, были придуманы параметризованные запросы. В них неизвестные значения заменяются параметрами следующим образом:

```
myCommand.CommandText = "UPDATE Туристы SET Фамилия = @Family
WHERE Кодтуриста = @TouristID";
```

Здесь @Family – параметр для неизвестного значения фамилии, @TouristID – параметр для неизвестного значения кода туриста.

Отметим, что параметры пишутся без кавычек с использованием символа @.

Теперь можно выполнить привязку параметров к тексту, вводимому пользователем. Для демонстрации привязки параметров создадим в нашем Windows-приложении специальные обработчики с использованием метода ExecuteNonQuery.

4. Создание обработчика на Изменение данных в форме.

4.1. Вначале объявим параметры, это будут переменные, в которые мы будем сохранять значения наших текстовых полей. Самостоятельно в форме в области объявления переменных объявите следующие переменные:

```
int kod_fac;  
long tel_d;  
string name_fac, fio_d, nom_au;
```

4.2. В предыдущих работах уже создавали запрос на изменение Аналогично, будем также составлять программный код.

4.3. Создайте обработчик на нажатие кнопки «Изменить».

4.4. Вначале необходимо передать значения полей в наши параметры, с учетом типов данных:

```
private void button6_Click(object sender, EventArgs e)  
{  
    kod_fac = int.Parse(this.textBox1.Text);  
    name_fac = Convert.ToString(this.textBox2.Text);
```

Самостоятельно, передайте все остальные значения.

4.5. Далее, выполняем запрос на изменение UPDATE . Этот запрос составляем аналогично, как и в примере выше, но будем менять запись таблицы Facultet с параметрами @name\_fac, @fio\_d , @nom\_au, @tel\_d при условии, что код факультета совпадает с параметром @kod\_fac. Согласно нашим обозначениям напишите ниже:

```
SqlCommand myCommand = DBConnection.Instance().Connection.CreateCommand();  
myCommand.CommandText = "UPDATE Facultet SET
```

Самостоятельно закончите данный запрос на изменение, см. пример выше.

4.6. Значения, введенные пользователем в текстовые поля помещаются в переменные @kod\_fac, @name\_fac, @fio\_d , @nom\_au, @tel\_d. Они должны добавляться в коллекцию объекта Command с помощью метода Add свойства Parameters, а затем уже значения параметров устанавливаются равными переменным kod\_fac, name\_fac, fio\_d , nom\_au, tel\_d. Конструкция метода Add, свойства Parameters объекта Command для поставщика данных OLE DB имеет в точности такую же структуру.

Таблица 1. Свойства метода Add

Свойство	Описание
parameterName	Название параметра
sqlDbType	Тип данных передаваемого параметра
size	Размер параметра
sourceColumn	Название имени столбца объекта DataSet, на который ссылается данный параметр

4.7. Поэтому для добавления значений, введенных пользователем в текстовые поля в переменные @kod\_fac, @name\_fac, @fio\_d , @nom\_au, @tel\_d напишите ниже сразу после запроса следующий код:

```

myCommand.Parameters.Add("@kod_fac", SqlDbType.Int, 4);
myCommand.Parameters["@kod_fac"].Value = kod_fac;
myCommand.Parameters.Add("@name_fac", SqlDbType.NVarChar, 255);
myCommand.Parameters["@name_fac"].Value = name_fac;
myCommand.Parameters.Add("@fio_d", SqlDbType.NVarChar, 50);
myCommand.Parameters["@fio_d"].Value = fio_d;
myCommand.Parameters.Add("@nom_au", SqlDbType.NVarChar, 10);
myCommand.Parameters["@nom_au"].Value = nom_au;
myCommand.Parameters.Add("@tel_d", SqlDbType.BigInt, 8);
myCommand.Parameters["@tel_d"].Value = tel_d;

```

Где значения передаваемых параметров, должны совпасть точно с типами данных и их количествами символов, которые были объявлены в таблице Facultet в базе данных Microsoft SQL Server.

4.8. Далее заканчиваем написание нашего обработчика, программным кодом для выполнения запроса и проверкой его выполнения. Для этого ниже напишите:

```

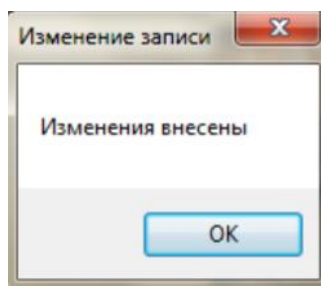
UspeshnoeIzmenenie = myCommand.ExecuteNonQuery();
if (UspeshnoeIzmenenie != 0)
{
    MessageBox.Show("Изменения внесены", "Изменение записи");
}
else
{
    MessageBox.Show("Не удалось внести изменения", "Изменение записи");
}

```

4.9. Сохранитесь и запустите ваше приложение на выполнение. Откройте нашу форму для Ввода данных.

4.10. Изменим, например информацию второй записи в таблице Факультеты. Для этого нажмите на кнопке вначале Следующая и перейдем на вторую запись. Изменим вручную старое название факультета – КОМПЬЮТЕРНЫХ СИСТЕМ И ТЕХНОЛОГИЙ на новое – ТРАНСПОРТА И ЛОГИСТИКИ.

4.11. Далее нажимаем на кнопку Обновить. У вас, если вы не допустили ошибок в коде, должно появиться следующее сообщение:



4.12. Чтобы убедиться, что данные действительно были изменены в базе данных, мы можем не открывать СУБД, а можем, не выходя из приложения, вернуться на главную форму и нажать на кнопку Факультет. В этой форме, данные таблицы Факультет отображаются в виде таблицы. И вы сразу увидите результат своей работы – там должен появиться факультет с измененным названием.

Факультет						
...    1 of 4 ...						
	kod_faculteta	Name_faculteta	Fio_Decana	Nomer_komnatu	Tel_decanata	
▶	1	Математики и информатики	Статьева Юрий Иванович	31/a	417478	
	2	Транспорта и логистики	Губачева Париса Александровна	204/12	477099	
	5	Международный	Харченко Евгений Иванович	310/9	500830	
	7	Юридический факультет	Лазор Париса Ивановна	123/9	658901	
*						

## 5. Создание обработчика на кнопку Новая в форме

Смысл данной кнопки в том, чтобы все поля приняли пустые значения.

5.1. Для этого создайте обработчик события на нажатие этой кнопки.

5.2. Введите программный код, в котором все текстовые поля примут значения пустой строки, например, так:

```
private void button3_Click(object sender, EventArgs e)
{
    textBox1.Text = "";
}
```

5.3. Аналогично, заполняются все остальные поля.

5.4. Но лучше, чтобы первое поле с кодом факультета было не пустым, а приняло значение со следующим по порядку номером факультета. Для этого, мы можем извлечь из базы данных последнее значение с номером факультета и присвоить его полю плюс 1. Тогда лучше изменить присвоение для первого текстового поля на следующее:

```
textBox1.Text = Convert.ToString(int.Parse(ds1.Tables["Facultet"].Rows[ds1.Tables["Facultet"].Rows.Count - 1][0].ToString())
```

5.5. Сохранитесь и запустите приложение на выполнение.

5.6. При нажатии на кнопку Новая у вас все поля в форме должны принять пустые значения, а первое поле примет новое значение, равное последнему номеру факультета +1 (в примере, последнее значение было равно 7, поэтому новый факультет будет равен 8).

## 6. Создание обработчика на Добавление данных в форме

Смысл данной кнопки в том, чтобы все поля, которые заполнили после нажатия кнопки Новая, необходимо добавить в базу данных. Команда добавления осуществляется методом INSERT. Добавление данных должно быть произойти после нажатия на кнопку Добавить.

6.1. Для этого создайте обработчик события на нажатие этой кнопки.

6.2. В предыдущих работах создавали запрос на добавление. Аналогично, будем также составлять программный код.

6.3. Вначале необходимо передать значения полей в наши параметры, с учетом типов данных. Самостоятельно выполните это, согласно обработчику UPDATE.

6.4. Далее, выполняем запрос на добавление INSERT .

6.5. Этот запрос составляем согласно синтаксису SQL:

```
"INSERT INTO Facultet (все поля таблицы, кроме первичного, так как его номер должен у вас автоматически считаться, если это не так, то все поля таблицы перечисляете через запятую)
VALUES (все параметры)";
```

6.6. Значения, введенные пользователем в текстовые поля помещаются в переменные @kod\_fac, @name\_fac, @fio\_d , @nom\_au, @tel\_d. Они должны добавляться в коллекцию объекта Command с помощью метода Add свойства Parameters, а затем уже значения параметров устанавливаются равными переменным kod\_fac, name\_fac, fio\_d , nom\_au, tel\_d. Поэтому выполните добавление значений, введенных пользователем в текстовые поля в переменные @kod\_fac, @name\_fac, @fio\_d , @nom\_au, @tel\_d.

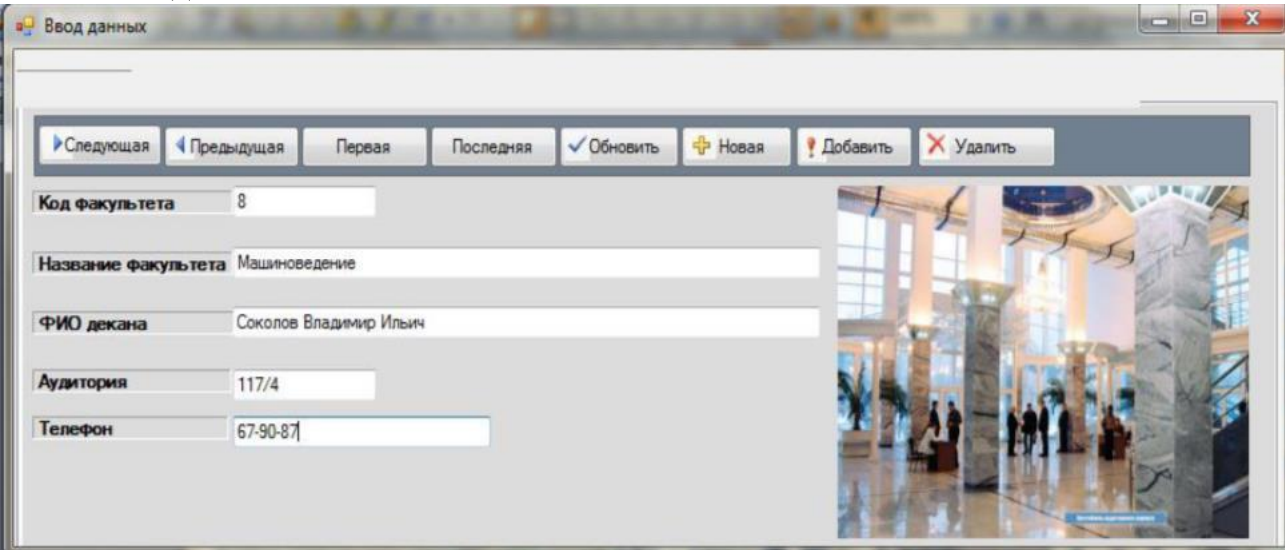
Самостоятельно выполните это, согласно обработчику UPDATE Где значения передаваемых параметров, должны совпасть точно с типами данных и их количествами символов, которые были объявлены в таблице Facultet в базе данных Microsoft SQL Server.

6.7. Далее заканчиваем написание нашего обработчика, программным кодом для выполнения запроса и проверкой его выполнения.

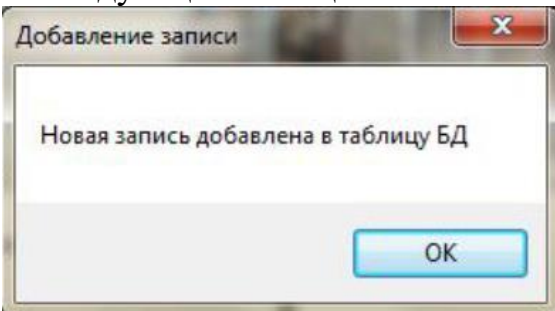


```
UspeshnoeIzmenenie = myCommand.ExecuteNonQuery();
if (UspeshnoeIzmenenie !=0)
{
    MessageBox.Show("Новая запись добавлена в таблицу БД", "Добавление записи");
}
else
{
    MessageBox.Show("Не удалось добавить запись в таблицу БД", "Добавление записи");
}
```

- 6.8. Сохранитесь и запустите приложение на выполнение. Откройте форму для Ввода данных.
- 6.9. Сначала нажимаем на кнопку Новая, вводим новый факультет, например Машиноведения.



- 6.10. Далее нажимаем на кнопку Добавить. У вас, если вы не допустили ошибок в коде, то должно появиться следующее сообщение:



- 6.11. Чтобы убедиться, что данные действительно были добавлены в базу данных, мы можем не открывать СУБД, а можем, не выходя из приложения, вернуться на главную форму и нажать на кнопку Факультет. В этой форме, данные таблицы Факультет отображаются в виде таблицы. И вы сразу увидите результат своей работы – там должен появиться новый факультет Машиноведения.

Факультет					
1 of 5					
	kod_faculteta	Name_faculteta	Fio_Decana	Nomer_komnatu	Tel_decanata
▶	1	Математики и информатики	Статьева Юрий Иванович	31/а	417478
	2	Транспорта и логистики	Губачева Париса Александровна	204/12	477099
	5	Международный	Харченко Евгений Иванович	310/9	500830
	7	Юридический факультет	Лазор Париса Ивановна	123/9	658901
	10	Машиноведение	Соколов Владимир Ильич	117/4	658901
*					

7. Создание обработчика на Удаление данных в форме

Смысл данной кнопки в том, чтобы выбранную запись удалить из базы данных. Команда удаления осуществляется методом DELETE. Удаление данных должно быть произойти после нажатия на кнопку Удалить.

7.1. Для этого создайте обработчик события на нажатие этой кнопки.

7.2. В предыдущих работах создавали запрос на удаление. Аналогично, будем также составлять программный код.

7.3. В начале необходимо передать значения полей в наши параметры, с учетом типов данных. Самостоятельно выполните это, согласно обработчику UPDATE.

7.4. Далее, выполняем запрос на удаление DELETE .

7.5. Этот запрос составляем согласно синтаксису SQL:

"DELETE FROM Facultet where условие, что код факультета совпадает с передаваемым параметром";

7.6. В отличие, от предыдущих запросов, нас интересует для выполнения данного запроса только один параметр, а именно - @kod\_fac. Он должен добавиться в коллекцию объекта Command с помощью метода Add свойства Parameters, а затем уже значение параметра устанавливается равным переменной kod\_fac. Поэтому выполните добавление значения, введенных пользователем в текстовое поле в переменную @kod\_fac.

Самостоятельно выполните это, согласно обработчику UPDATE. Где значение передаваемого параметра, должно совпасть точно с типом данных и их количеством символов, которое было объявлено в таблице Facultet в базе данных Microsoft SQL Server.

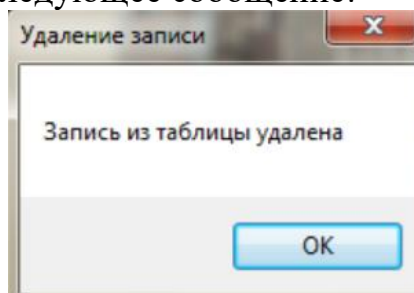
7.7. Далее заканчиваем написание нашего обработчика, программным кодом для выполнения запроса и проверкой его выполнения.

```
UspeshnoeIzmenenie = myCommand.ExecuteNonQuery();
if (UspeshnoeIzmenenie != 0)
{
    MessageBox.Show("Запись из таблицы удалена", "Удаление записи");
    textBox1.Text = "";
    textBox2.Text = "";
    textBox3.Text = "";
    textBox4.Text = "";
    textBox5.Text = "";
}
else
{
    MessageBox.Show("Не удалось удалить запись из таблицы", "Удаление записи");
}
```

7.8. Сохранитесь и запустите приложение на выполнение. Откройте форму для Ввода данных.

7.9. Выполните переход на последнюю запись, нажав на кнопку Последняя. Это будет факультет Машиноведения.

7.10. Далее нажимаем на кнопку Удалить. У вас, если вы не допустили ошибок в коде, то должно появиться следующее сообщение:



7.11. Чтобы убедиться, что данные действительно были удалены из базы данных, мы можем не открывать СУБД, а можем, не выходя из приложения, вернуться на

главную форму и нажать на кнопку Факультет. В этой форме, данные таблицы Факультет отображаются в виде таблицы. И вы сразу увидите результат своей работы – там должен исчезнуть новый факультет машиноведения.

Факультеты и Кафедры					
...  <  > 1 of 4  <  >  <  >  <  >					
	kod_faculteta	Name_faculteta	Fio_Decana	Nomer_komnatu	Tel_decanata
▶	1	Математики и информатики ...	Статьева Юрий Иванович ...	31/а	417478
	2	Транспорта и логистики ...	Губачева Лариса Александровна ...	204/12	477099
	5	Международный ...	Харченко Евгений Иванович ...	310/9	500830
	7	Юридический факультет ...	Лазор Лариса Ивановна ...	123/9	658901
*					



## **ПРАКТИЧЕСКАЯ РАБОТА № 39**

### **Тема: Параметризованные запросы на C# ADO.NET**

**Цель работы:** научиться разрабатывать программы языке C# с использованием параметризованными запросами.

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, MS Visual Studio, инструкции по выполнению работы.

#### **Содержание работы:**

**Задание 1.** Для созданной базы данных Учебный центр продолжить разрабатывать клиентское приложение в среде Visual Studio C#, сделанного в Практической работе № 36. Создать новую диалоговую форму, с помощью которой осуществить ввод данных, редактирование и навигацию в одной из таблиц базы данных. Данные должны вводиться с помощью текстовых полей (или других элементов управления). Создать кнопки управления для добавления данных в базу данных, удаления, изменения, создания новой записи, переходы по записям. Клиентское приложение сохранить в папке

**Задание 2.** Для созданной базы данных Торговая фирма продолжить разрабатывать клиентское приложение в среде Visual Studio C#, сделанного в Практической работе № 36. Создать новую диалоговую форму, с помощью которой осуществить ввод данных, редактирование и навигацию в одной из таблиц базы данных. Данные должны вводиться с помощью текстовых полей (или других элементов управления). Создать кнопки управления для добавления данных в базу данных, удаления, изменения, создания новой записи, переходы по записям. Клиентское приложение сохранить в папке

## **ПРАКТИЧЕСКАЯ РАБОТА № 40**

### **Тема: Параметризованные запросы на C# ADO.NET**

**Цель работы:** научиться разрабатывать программы языке C# с использованием параметризованными запросами.

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, MS Visual Studio, инструкции по выполнению работы.

#### **Содержание работы:**

**Задание 1.** Для созданной базы данных Проектная организация продолжить разрабатывать клиентское приложение в среде Visual Studio C#, сделанного в Практической работе № 37. Создать новую диалоговую форму, с помощью которой осуществить ввод данных, редактирование и навигацию в одной из таблиц базы данных. Данные должны вводиться с помощью текстовых полей (или других элементов управления). Создать кнопки управления для добавления данных в базу данных, удаления, изменения, создания новой записи, переходы по записям. Клиентское приложение сохранить в папке

**Задание 1.** Для созданной базы данных Компания по разработке программных продуктов продолжить разрабатывать клиентское приложение в среде Visual Studio C#, сделанного в Практической работе № 37. Создать новую диалоговую форму, с помощью которой осуществить ввод данных, редактирование и навигацию в одной из таблиц базы данных. Данные должны вводиться с помощью текстовых полей (или других элементов управления). Создать кнопки управления для добавления данных в базу данных, удаления, изменения, создания новой записи, переходы по записям. Клиентское приложение сохранить в папке

## ПРАКТИЧЕСКАЯ РАБОТА № 41

### Тема: Поиск и фильтрация данных в базе данных

**Цель работы:** научиться разрабатывать программы языке C# для решение задач поиска и фильтрации данных в базе данных

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, MS Visual Studio, инструкции по выполнению работы.

#### Справочный материал:

В этой практической работе продолжим создание клиентского приложения к базе данных Университет, созданных в Практических работах № 35, 38, путем изучения темы поиска данных и создания пользовательских фильтров.

**Поиск** — это нахождение записи, удовлетворяющей определенным условиям, и возврат значений ее полей с возможным переходом на найденную запись. Отметим, что поиск можно вести по одним полям, а возвращать значения других полей. Составы полей для поиска и для возврата значений в общем случае не совпадают.

При организации поиска записей важное значение имеет наличие индекса для полей, по которым ведется поиск. Индексирование значительно повышает скорость обработки данных, кроме того, ряд методов может работать только с индексированными полями.

**Фильтрация** — это задание ограничений для записей, отбираемых в набор данных. Напомним, что набор данных представляет собой записи, выбранные из одной или нескольких таблиц. Состав записей в наборе данных в данный момент времени зависит от установленных ограничений, в том числе и от фильтров.

#### Содержание работы:

**Задание 1.** Создать диалоговую форму, в которой первоначально будут отображаться в таблице все записи преподавателей университета. В форме предусмотреть поля для ввода критериев поиска информации. Осуществить автоматическое выполнение фильтрации информации по введенным сложным критериям отбора информации и отображение результатов на экране.

1. Откройте Visual Studio и откройте приложение, сохраненное в вашей папке (созданное в Практической работе № 38).

2. Уже была создана ранее диалоговая форма, в которой отображаются данные о всех преподавателях. Поэтому для решения поставленной задачи не будем создавать новую форму, а воспользуемся ранее созданной.

Данные в этой форме отображаются в виде обычной таблицы:

	KOD_TE...	KOD_KAF...	NAME_TEACHER	INDEF_KOD	DOLGHO...	Zvanie	SALARY	RISE	DATA_HIRE	BIRTHDAY	POL	TEL_TEACHER
1	9		Соловьев Виктор Иванович	00002292	доцент	К.Т.Н	3000.00	500.00	1990-03-22...	1965-03-22...	м	12-09-98
3	9		Ипчатьева Олеся Владимировна	111019182	доцент	К.Т.Н	3000.00	50.00	2001-08-22...	1979-03-22...	ж	44-03-98
4	9		Полупан Юлия Викторовна	111019183	доцент	К.Т.Н	3000.00	50.00	2001-08-22...	1979-05-18...	ж	11-08-98
5	9		Детярева Лариса Николаевна	111078654	доцент	К.Т.Н	3000.00	150.00	1990-05-01...	1969-03-21...	ж	13-22-98
6	9		Белозеров Евгений Владимирович	111078659	доцент	К.Т.Н	3000.00	250.00	1992-05-01...	1969-11-18...	м	22-23-96
20	9		Ратков Денис Николаевич	1110786	асистент	нет	1500.00	250.00	2008-05-01...	1982-10-22...	м	NULL
22	10		Малый Вячеслав Вадимович	1210786	доцент	К.Т.Н	3000.00	300.00	1980-05-01...	1952-10-22...	м	35-76-01
23	10		Куча Владимир Иванович	12107869	доцент	К.Т.Н	3000.00	300.00	1980-07-21...	1956-10-02...	м	44-76-11
24	10		Кочевский Андрей Александрович	12107860	доцент	К.Т.Н	3000.00	500.00	1999-04-11...	1975-08-22...	м	44-76-12
27	11		Арляноый Юрий Моисеевич	1363528	профессор	д.ф.н.	5000.00	500.00	1976-04-11...	1954-08-22...	м	41-73-12
28	11		Дмитрук Евгения Викторовна	13635287	доцент	К.Т.Н	3000.00	300.00	2000-08-29...	1978-02-01...	ж	41-41-41
29	11		Владыкина Нина Довыдовна	13635288	асистент	нет	1500.00	300.00	1980-08-18...	1956-02-01...	ж	41-42-41
30	12		Статькина Юрий Иванович	172625238	доцент	К.Т.Н	3500.00	500.00	1978-08-21...	1966-02-11...	м	41-46-51
31	12		Пархоненко Виталий Павлович	172625000	доцент	К.Т.Н	3000.00	100.00	2001-09-11...	1978-02-11...	м	41-46-87
33	12		Тарасенко Андрей Владимирович	17262501	старший препод...	нет	2000.00	100.00	2000-11-15...	1976-12-11...	м	41-52-87
34	14		Холод Олег Николаевич	584849388	профессор	д.т.н.	5000.00	100.00	2000-11-15...	1956-12-01...	м	41-00-07
35	14		Швец Светлана Николаевна	584849865	старший препод...	К.Т.Н	2000.00	400.00	2001-10-22...	1975-07-01...	ж	40-00-00
36	14		Яковенко Владимир Андреевич	584849865	профессор	д.т.н.	4000.00	200.00	1980-10-01...	1951-11-18...	м	40-03-09
37	15		Губачева Лариса Николаевна	58484900	профессор	д.т.н.	6000.00	100.00	1976-01-01...	1957-03-22...	ж	40-03-11
38	15		Бобровский Геннадий Александрович	51038373	доцент	К.Т.Н	3000.00	100.00	1988-01-01...	1971-04-12...	м	42-03-11
39	15		Жученко Наталья Макаровна	110208373	асистент	нет	1200.00	200.00	1989-01-01...	1971-07-09...	ж	42-03-12
40	16		Ульянов Иван Васильевич	11020980	профессор	д.т.н.	4000.00	500.00	1977-01-01...	1945-12-19...	м	49-03-11

3. Перейдите в режим дизайна формы Преподаватели и растяните ее. Добавьте справа элементы управления: Label, Textbox, ComboBox, Button, так как показано на рисунке

**ВВЕДИТЕ КРИТЕРИЙ ПОИСКА**

**ФАМИЛИЯ ИМЯ ОТЧЕСТВО:**

**МЕСТО РАБОТЫ:**

**ДОЛЖНОСТЬ:**

**НАУЧНОЕ ЗВАНИЕ:**

**ВОЗРАСТ:**

**СТАЖ:**

**СТАВКА:**

4. Поиск будем осуществлять по следующим критериям – ФИО, Кафедра, Должность, Научное звание, Возраст, Стаж и Ставка. При чем, будем выполнять поиск по введенному значению целиком или его частью, по одному значению или по разным комбинациям разных полей поиска. Результат выполнения поиска будем отображать в таблице. Также предусмотрена отмена выполнения фильтра и возврат всех данных исходной таблицы.

5. Перейдите в код данной формы и объявим необходимые переменные. В области описания переменных добавьте следующие переменные:

```

public partial class Form5 : Form
{
    SqlDataAdapter dataAdapter1 = new SqlDataAdapter();
    SqlDataAdapter dataAdapter2 = new SqlDataAdapter();
    DataSet ds1 = new DataSet();
    SqlCommand myCommand;
    int UspeshnoeIzmenenie;
    string Sqlmy;
    // объявление параметров для выполнения запроса-фильтра

```

```
int kod_kafedru_sotrud;  
string fio_sotrud, dolgn_sotrud, vozrast_sotrud, nauch_zvanie_sotrud,  
stagem_sotrud, stavka_sotrud;  
int l1, l2, m1, m2, k1, k2;
```

6. Согласно исходным данным пользователь будет вводить ФИО в текстовое поле, а вот все остальные критерии организуем в виде выбора из раскрывающихся списков. Для этого необходимо заполнить эти списки.

Первый раскрывающийся список – Место работы. Здесь необходимо отобразить все кафедры, которые имеются в нашем университете. Вписывать кафедры вручную мы не будем, так как в нашей базе данных University уже имеется таблица KAFEDRA. Поэтому лучше воспользоваться созданием нового набора данных и вывести его в элементе Combobox1.

Поэтому после инициализации элементов формы:

```
public Form2() {  
    InitializeComponent();  
    // ниже после объявления dataAdapter1 объявляем dataAdapter2  
    dataAdapter2.SelectCommand = new SqlCommand("Select * from Kafedra",  
        DBConnection.Instance().Connection);  
    dataAdapter2.Fill(ds1, "Kafedra");  
    // Далее перемещаем данные таблицы Кафедры в раскрывающийся список  
    comboBox1.DataSource = ds1.Tables["Kafedra"];  
    comboBox1.DisplayMember = "name_kafedru"; // столбец для отображения  
    comboBox1.ValueMember = "kod_kafedru"; // столбец с id  
    comboBox1.SelectedIndex = -1;
```

7. Далее, наполним раскрывающиеся списки для выбора должности, научного звания, возраста, стажа и ставки. Эти данные мы не можем взять, как это было выше, из каких-то ранее определенных таблиц, например у нас в нашей базе данных нет таблиц Должности, Ставки и т.д. Поэтому эти списки определим самостоятельно.

Добавьте ниже следующий код:

```
// Должности  
this.comboBox2.Text = "";  
this.comboBox2.Items.Add("Профессор");  
this.comboBox2.Items.Add("Доцент");  
this.comboBox2.Items.Add("Старший преподаватель");  
this.comboBox2.Items.Add("Преподаватель-стажер");  
this.comboBox2.Items.Add("Ассистент");  
this.comboBox2.Items.Add("Заведующий лабораторией");  
this.comboBox2.Items.Add("Лаборант");  
this.comboBox2.Items.Add("Инженер");  
this.comboBox2.Items.Add("Секретарь");  
this.comboBox2.Items.Add("Заведующий кафедрой");  
// Научные звания  
this.comboBox3.Text = "";  
this.comboBox3.Items.Add("нет");  
this.comboBox3.Items.Add("к.т.н");  
this.comboBox3.Items.Add("к.г.н");  
this.comboBox3.Items.Add("к.г.у");
```

```
this.comboBox3.Items.Add("к.м.н");
this.comboBox3.Items.Add("к.п.н");
this.comboBox3.Items.Add("к.с.н");
this.comboBox3.Items.Add("к.б.н");
this.comboBox3.Items.Add("к.в.н");
this.comboBox3.Items.Add("к.д.н");
this.comboBox3.Items.Add("к.э.н");
this.comboBox3.Items.Add("к.и.н");
this.comboBox3.Items.Add("к.ф.-м.н");
this.comboBox3.Items.Add("д.т.н");
this.comboBox3.Items.Add("д.г.н");
this.comboBox3.Items.Add("д.г.у");
this.comboBox3.Items.Add("д.м.н");
this.comboBox3.Items.Add("д.п.н");
this.comboBox3.Items.Add("д.с.н");
this.comboBox3.Items.Add("д.б.н");
this.comboBox3.Items.Add("д.в.н");
this.comboBox3.Items.Add("д.д.н");
this.comboBox3.Items.Add("д.э.н");
this.comboBox3.Items.Add("д.и.н");
this.comboBox3.Items.Add("д.ф.-м.н");
// Возраст
this.comboBox4.Text = "";
this.comboBox4.Items.Add("от 20 до 30 лет");
this.comboBox4.Items.Add("от 30 до 40 лет");
this.comboBox4.Items.Add("от 40 до 50 лет");
this.comboBox4.Items.Add("от 50 до 60 лет");
this.comboBox4.Items.Add("от 60 до 70 лет");
this.comboBox4.Items.Add("старше 70 лет");
// Стаж
this.comboBox5.Text = "";
this.comboBox5.Items.Add("меньше 5 лет");
this.comboBox5.Items.Add("от 5 до 10 лет");
this.comboBox5.Items.Add("от 10 до 15 лет");
this.comboBox5.Items.Add("от 15 до 20 лет");
this.comboBox5.Items.Add("от 20 до 25 лет");
this.comboBox5.Items.Add("от 25 до 30 лет");
this.comboBox5.Items.Add("от 30 до 35 лет");
this.comboBox5.Items.Add("от 35 до 40 лет");
this.comboBox5.Items.Add("от 40 до 45 лет");
this.comboBox5.Items.Add("от 45 до 50 лет");
this.comboBox5.Items.Add("свыше 50 лет");
// Ставка
this.comboBox6.Text = "";
this.comboBox6.Items.Add("меньше 1000 руб");
this.comboBox6.Items.Add("от 1000 до 2000 руб");
this.comboBox6.Items.Add("от 2000 до 3000 руб");
this.comboBox6.Items.Add("от 3000 до 4000 руб");
```

```
this.comboBox6.Items.Add("от 4000 до 5000 руб");
```

```
this.comboBox6.Items.Add("свыше 5000 руб");
```

8. Сохраните программный проект и запустите его на выполнение. У вас не должно быть ошибок, если они есть, то исправьте их. На экране должна появиться форма приветствия, нажимаем на кнопку Преподаватели. Данные всех преподавателей должны быть отображены в таблице и рядом пока пустые поля. Пооткрывайте списки – в них должны быть варианты выбора.

Возвращаемся далее к написанию приложения.

9. Добавьте обработчик события на нажатие кнопки Применить. При нажатии на эту кнопку будем вначале передавать в параметры значения выбранных критериев поиска информации.

```
// поиск информации
```

```
dataAdapter1.Dispose(); // освобождение набора данных
```

```
// считывание ФИО из текстового поля
```

```
fio_sotrud = this.textBox1.Text;
```

```
fio_sotrud = "%" + fio_sotrud + "%"; // добавляем к строке подстановочные символы
```

```
fio_sotrud = fio_sotrud.ToLower(); // переводим все символы в нижний регистр
```

```
// считывание кафедры из раскрывающегося списка, но нам не нужно само
```

```
//название, а будем извлекать код кафедры из раскрывающегося списка
```

```
kod_kafedru_sotrud = this.comboBox1.SelectedIndex + 1;
```

```
// считывание должности из раскрывающегося списка
```

```
dolgn_sotrud = this.comboBox2.Text;
```

```
dolgn_sotrud = "%" + dolgn_sotrud + "%";
```

```
dolgn_sotrud = dolgn_sotrud.ToLower();
```

```
// считывание научного звания из раскрывающегося списка
```

```
nauch_zvanie_sotrud = this.comboBox3.Text;
```

```
nauch_zvanie_sotrud = "%" + науч_zvanie_sotrud + "%";
```

```
nauch_zvanie_sotrud = науч_zvanie_sotrud.ToLower();
```

```
// считывание возраста из раскрывающегося списка
```

```
vozrast_sotrud = this.comboBox4.Text;
```

```
if (vozrast_sotrud == "от 20 до 30 лет") { l1 = 20; l2 = 30; }
```

```
if (vozrast_sotrud == "от 30 до 40 лет") { l1 = 30; l2 = 40; }
```

```
if (vozrast_sotrud == "от 40 до 50 лет") { l1 = 40; l2 = 50; }
```

```
if (vozrast_sotrud == "от 50 до 60 лет") { l1 = 50; l2 = 60; }
```

```
if (vozrast_sotrud == "от 60 до 70 лет") { l1 = 60; l2 = 70; }
```

```
if (vozrast_sotrud == "старше 70 лет") { l1 = 70; l2 = 500; }
```

```
// считывание стажа из раскрывающегося списка
```

```
stagem_sotrud = this.comboBox5.Text;
```

```
// считывание ставки из раскрывающегося списка
```

```
stavka_sotrud = this.comboBox6.Text;
```

10. Самостоятельно присвойте значения переменным m1, m2, k1, k2 – как левые и правые границы диапазонов для стажа и ставки преподавателя.

11. Далее ниже выполним присвоение строки запроса, но пока выполнять его не будем:

```
SqlCommand myCommand = DBConnection.Instance().Connection.CreateCommand();
```

12. А теперь после того как все параметры были переданы, будем выполнять проверку, если ничего не выбрано и не введено не в одно поле, то выполнить

запрос на вывод всей таблицы Преподавателей, а иначе выполнять разные проверки в зависимости какие поля были выбраны или их комбинации.

// если ничего не выбрано

```
if ((fio_sotrud == "%%") & (kod_kafedru_sotrud == 0) & (dolgn_sotrud == "%%") &
(nauch_zvanie_sotrud == "%%") & (vozrast_sotrud == "") & (stagem_sotrud == "")&
(stavka_sotrud == "")) {
```

```
Sqlmy = "select * from Teacher ";
```

```
myCommand.CommandText = Sqlmy;
```

```
myCommand.ExecuteScalar(); // выполняем запрос
```

```
// очищаем таблицу dataGridView1 от старых данных
```

```
while (dataGridView1.Rows.Count > 1)
```

```
for (int i = 0; i < dataGridView1.Rows.Count - 1; i++)
```

```
dataGridView1.Rows.Remove(dataGridView1.Rows[i]);
```

```
// Передача новых данных
```

```
dataAdapter1.SelectCommand = myCommand;
```

```
dataAdapter1.Fill(ds1, "Преподаватели");
```

```
this.dataGridView1.DataSource = ds1.Tables["Преподаватели"].DefaultView; }
```

```
// иначе, если пользователь что-то выбрал
```

```
else {
```

```
// первая проверка, например, пользователь ввел что-то в поле ФИО
```

```
if ((fam_sotrud != "%%") & (kod_kafedru_sotrud == 0) & (dolgn_sotrud == "%%") &
(nauch_zvanie_sotrud == "%%") & (vozrast_sotrud == "") &
(stagem_sotrud == "")& (stavka_sotrud == "")) {
```

```
Sqlmy = "select * " + "from Teacher " + "where Lower(Teacher.Name_teacher) Like
@fio_sotrud ";
```

```
myCommand.CommandText = Sqlmy;
```

```
// передача параметров, в нашем случае пока только одного
```

```
myCommand.Parameters.Add("@fio_sotrud", SqlDbType.VarChar, 140);
```

```
myCommand.Parameters["@fio_sotrud"].Value = fio_sotrud;
```

```
// выполняем запрос
```

```
myCommand.ExecuteScalar();
```

```
// очистка таблицы dataGridView1 от старых данных
```

```
while (dataGridView1.Rows.Count > 1)
```

```
for (int i = 0; i < dataGridView1.Rows.Count - 1; i++)
```

```
dataGridView1.Rows.Remove(dataGridView1.Rows[i]);
```

```
// заполнение таблицы dataGridView1 новыми данными
```

```
dataAdapter1.SelectCommand = myCommand;
```

```
dataAdapter1.Fill(ds1, "Преподаватели");
```

```
this.dataGridView1.DataSource = ds1.Tables["Преподаватели"].DefaultView; }
```

```
// вторая проверка, например пользователь выбрал кафедру
```

```
if ((fam_sotrud == "%%") & (kod_kafedru_sotrud != 0) & (dolgn_sotrud == "%%") &
(nauch_zvanie_sotrud == "%%") & (vozrast_sotrud == "") & (stagem_sotrud == "")&
(stavka_sotrud == "")) {
```

// Самостоятельно выполнить запрос

// причем не забудьте, что параметр kod\_kafedru\_sotrud – это целое число



```
// третья проверка, например, пользователь выбрал кафедру и ФИО
if ((fam_sotrud != "%%") & (kod_kafedru_sotrud!= 0) & (dolgn_sotrud == "%%") &
(nauch_zvanie_sotrud == "%%") & (vozrast_sotrud == "") & (stagem_sotrud ==
"")& (stavka_sotrud == "")) {
// Самостоятельно выполнить запрос с двумя условиями
}
// САМОСТОЯТЕЛЬНО ПЕРЕБРАТЬ ВСЕ ВОЗМОЖНЫЕ КОМБИНАЦИИ
// последняя проверка, например, пользователь выбрал все поля
if ((fam_sotrud != "%%") & (kod_kafedru_sotrud!= 0) & (dolgn_sotrud != "%%") &
(nauch_zvanie_sotrud != "%%") & (vozrast_sotrud != "") & (stagem_sotrud != "")&
(stavka_sotrud != "")) {
// Самостоятельно выполнить запрос с семью условиями
// При чем, возраста и стажа в таблице Преподавателей нет, их нужно
вычислить
// как разность года текущей даты с годом рождения или годом поступления на
работу
// (YEAR(GETDATE())-YEAR(BIRTHDAY))) } }
```

13. Проверьте, чтобы работали запросы.

14. Сохраните проект и запустите его на выполнение.

1) Введете сначала в текстовое поле ФИО полное имя какого-нибудь преподавателя, который у вас имеется в вашей таблице, например, СОЛОВЬЕВ ВИКТОР ИВАНОВИЧ

И нажмите кнопку Применить. В результате у вас на экране должны быть стерты все старые данные и показана только одна запись с нашим выбранным преподавателем.

2) Затем введите в поле только часть поискового слова, например, ОЛЕГ

И нажмите кнопку Применить. В результате у вас на экране должны быть стерты все старые данные и показаны две записи с нашими выбранными преподавателями с такими именами.

3) Затем введите в поле только часть поискового слова, например, ЮЛИЯ и выберите из первого списка кафедру, например, КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ

И нажмите кнопку Применить. В результате у вас на экране должны быть стерты все старые данные и показана одна запись с нашим выбранным преподавателем с такими именем и такой кафедрой.

15. Продолжить работу с фильтрами и выполнить, чтобы работали все возможные комбинации поиска информации по всем полям.

16. В конце необходимо создать еще один обработчик для Очистки всех фильтров и возвращению к исходной таблице Преподаватели.

Для этого создайте обработчик события на нажатие кнопки Очистить и введите следующий код:

```
// возвращение исходной всей таблицы Преподавателей
dataAdapter1.Dispose();
Sqlmy = "select * from Teacher ";
```

```
SqlCommand myCommand =  
DBConnection.Instance().Connection.CreateCommand();  
myCommand.CommandText = Sqlmy;  
myCommand.ExecuteScalar();  
while (dataGridView1.Rows.Count > 1)  
for (int i = 0; i < dataGridView1.Rows.Count - 1; i++)  
dataGridView1.Rows.Remove(dataGridView1.Rows[i]);  
dataAdapter1.SelectCommand = myCommand;  
dataAdapter1.Fill(ds1, "Преподаватели");  
this.dataGridView1.DataSource = ds1.Tables["Преподаватели"].DefaultView; }  
17. Сохраните программный проект и продемонстрируйте преподавателю  
разные комбинации выполнения поиска преподавателей по разным полям.
```

## **ПРАКТИЧЕСКАЯ РАБОТА № 42**

### **Тема: Поиск и фильтрация данных в базе данных**

**Цель работы:** научиться разрабатывать программы языке C# для решение задач поиска и фильтрации данных в базе данных

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, MS Visual Studio, инструкции по выполнению работы.

#### **Содержание работы:**

**Задание 1.** Для созданной базы данных Учебный центр продолжить разрабатывать клиентское приложение в среде Visual Studio C#, сделанного в Практических работах № 36, 39. Измените одну из диалоговых форм, с помощью которой, осуществить поиск данных в зависимости от разных критериев. В данной форме первоначально будут отображаться все записи одной из таблиц. В форме предусмотреть поля для ввода критериев поиска информации. Осуществить автоматическое выполнение фильтрации информации по введенным сложным критериям отбора информации и отображение результатов на экране. На форме должны быть две кнопки Применить и Отменить, для применения фильтра и возврата данных в исходное положение. Клиентское приложение сохранить в папке

**Задание 2.** Для созданной базы данных Торговая фирма продолжить разрабатывать клиентское приложение в среде Visual Studio C#, сделанного в Практических работах № 36, 39. Измените одну из диалоговых форм, с помощью которой, осуществить поиск данных в зависимости от разных критериев. В данной форме первоначально будут отображаться все записи одной из таблиц. В форме предусмотреть поля для ввода критериев поиска информации. Осуществить автоматическое выполнение фильтрации информации по введенным сложным критериям отбора информации и отображение результатов на экране. На форме должны быть две кнопки Применить и Отменить, для применения фильтра и возврата данных в исходное положение. Клиентское приложение сохранить в папке

## **ПРАКТИЧЕСКАЯ РАБОТА № 43**

### **Тема: Поиск и фильтрация данных в базе данных**

**Цель работы:** научиться разрабатывать программы языке C# для решение задач поиска и фильтрации данных в базе данных

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, MS Visual Studio, инструкции по выполнению работы.

#### **Содержание работы:**

**Задание 1.** Для созданной базы данных Проектная организация продолжить разрабатывать клиентское приложение в среде Visual Studio C#, сделанного в Практических работах № 37, 40. Измените одну из диалоговых форм, с помощью которой, осуществить поиск данных в зависимости от разных критериев. В данной форме первоначально будут отображаться все записи одной из таблиц. В форме предусмотреть поля для ввода критериев поиска информации. Осуществить автоматическое выполнение фильтрации информации по введенным сложным критериям отбора информации и отображение результатов на экране. На форме должны быть две кнопки Применить и Отменить, для применения фильтра и возврата данных в исходное положение. Клиентское приложение сохранить в папке

**Задание 2.** Для созданной базы данных Компания по разработке программных продуктов продолжить разрабатывать клиентское приложение в среде Visual Studio C#, сделанного в Практических работах № 37, 40. Измените одну из диалоговых форм, с помощью которой, осуществить поиск данных в зависимости от разных критериев. В данной форме первоначально будут отображаться все записи одной из таблиц. В форме предусмотреть поля для ввода критериев поиска информации. Осуществить автоматическое выполнение фильтрации информации по введенным сложным критериям отбора информации и отображение результатов на экране. На форме должны быть две кнопки Применить и Отменить, для применения фильтра и возврата данных в исходное положение. Клиентское приложение сохранить в папке

## ПРАКТИЧЕСКАЯ РАБОТА № 44

**Тема:** Создание отчетных форм в клиентских приложениях

**Цель работы:** научиться создавать формы отчетных документов по данным БД

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, MS Visual Studio, инструкции по выполнению работы.

**Справочный материал:**

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 для поддержки разработки обычных отчетных приложений.

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 2022 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-контроля ASP.NET ReportViewer или Windows Forms-контроля ReportViewer. Это позволяет встраивать отчеты прямо в веб-страницы или .NET-приложения.

Службы MicrosoftSQL Server 2022 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.

## Содержание работы:

**Задание 1.** Создание простого табличного отчета (на основе базы данных) с помощью конструктора отчетов служб Microsoft SQL Server 2022 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.

Для создания отчетов используем пример базы данных с названием University.mdf, которая была создана ранее в MS SQL Server.

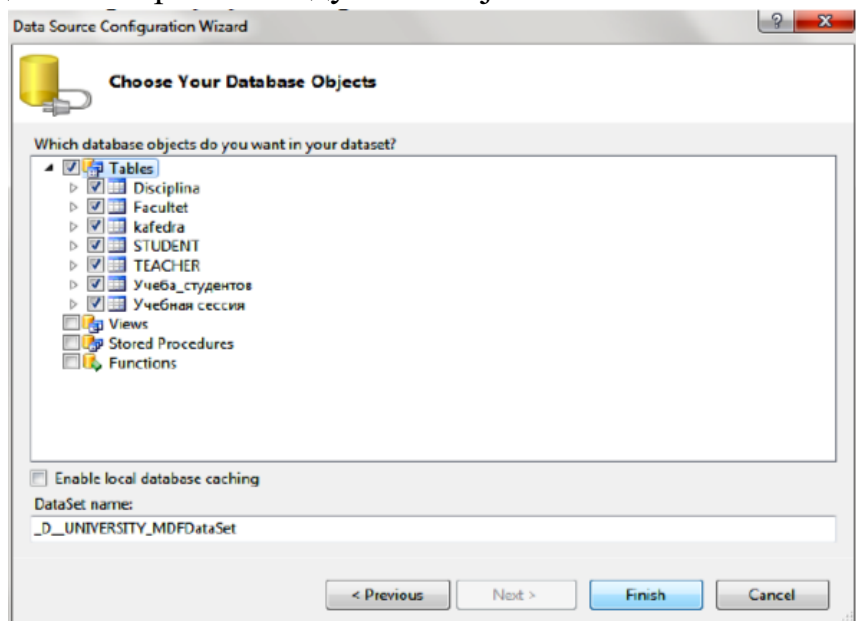
При выполнении примеров и заданий обращайтесь внимание на соответствие названий БД, таблиц и других объектов проекта.

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.

В Visual Studio C# 2022 есть несколько способов создания отчетов. Один из способов создание отчетов это использование встроенного генератора отчета *Reporting Services*.

1. Откройте проект БД Университет, созданный в Практической работе № 27. В этом проекте 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 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 2022 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.

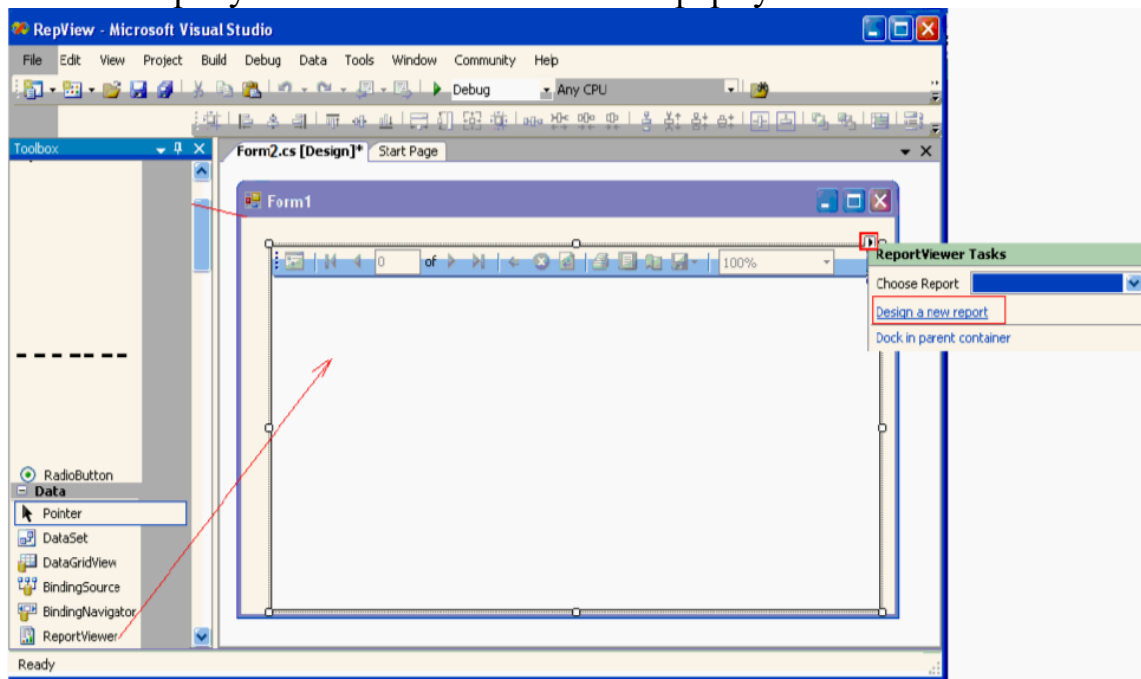
3. Размещать отчет мы будем на новой диалоговой форме. Поэтому в нашу первую форму добавим внизу формы кнопку Button с панели инструментов Toolbox. Переименуем подпись кнопки на «Отчет по преподавателям».

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.

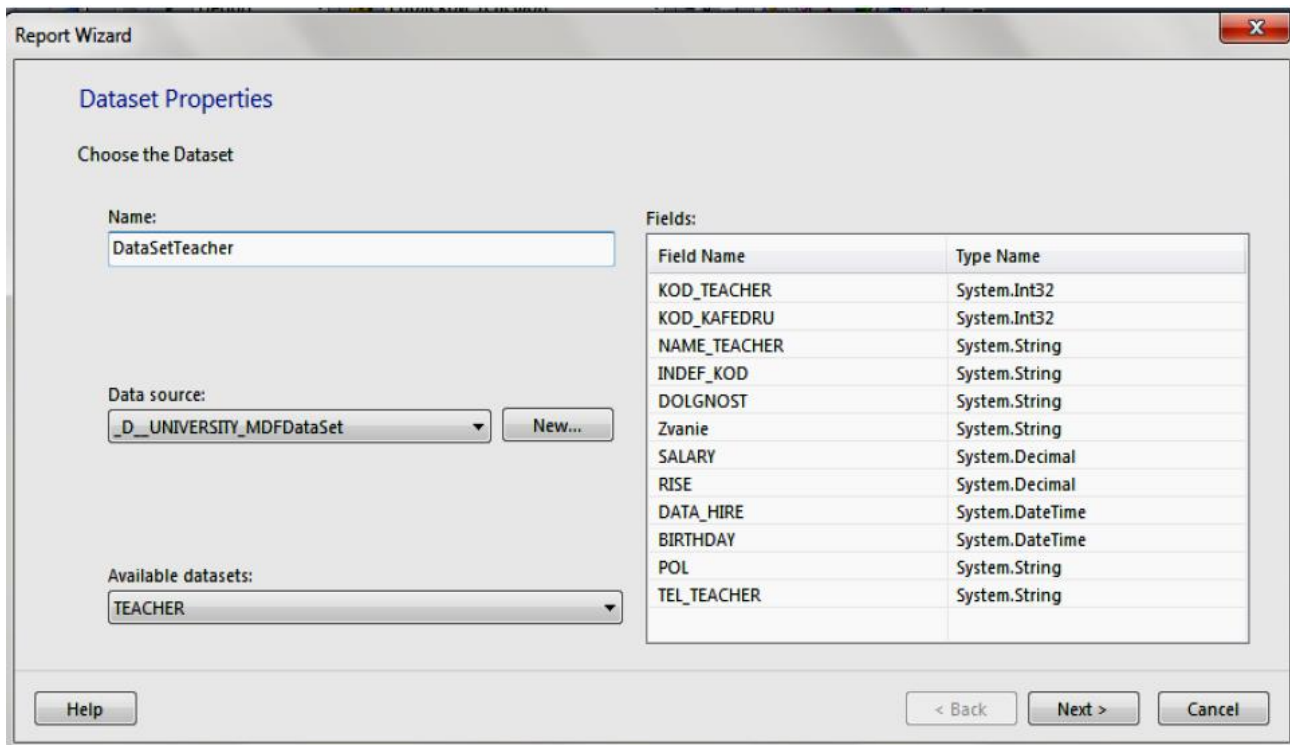


5. Настроим источник данных для отображения. Нажимаем в правом верхнем углу контрола ReportViewer треугольничек и выбираем пункт Design a new

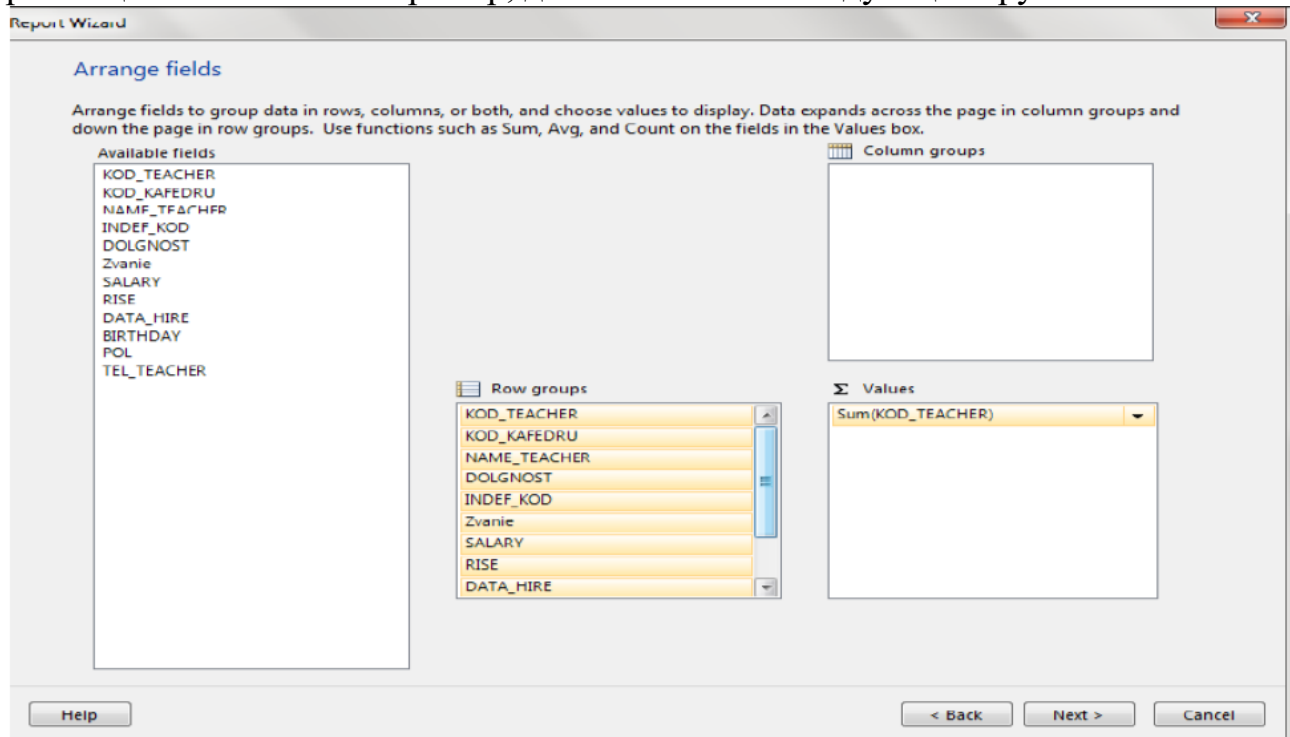


Report. В появившемся окне редактора ReportViewer, нажимаем кнопку "Add New Source.."

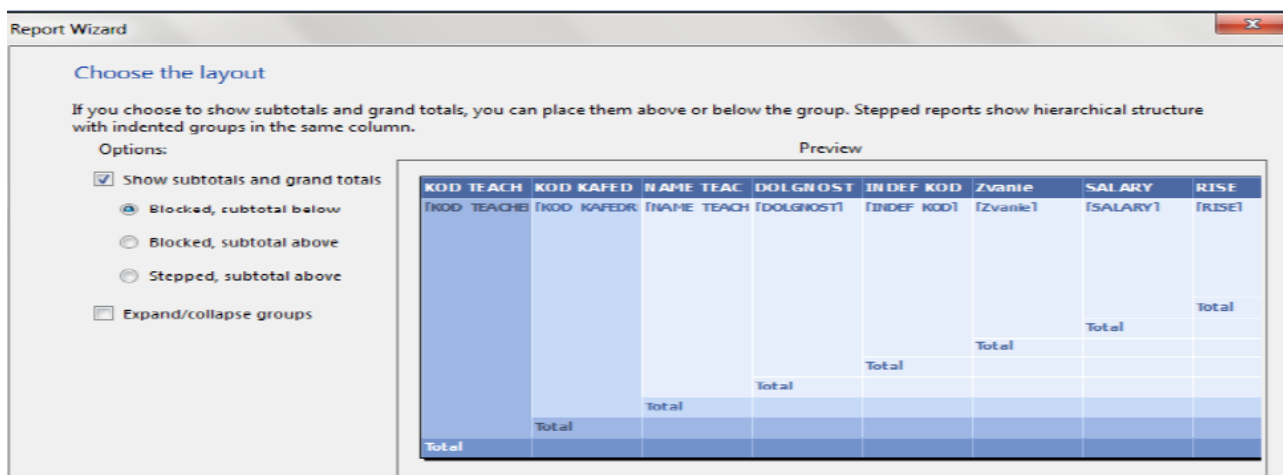
В появившемся окне выберем уже созданный нами ранее общий источник данных – UniversityDataSet, таблица – Teacher, переименуем имя на DataSetTeacher.



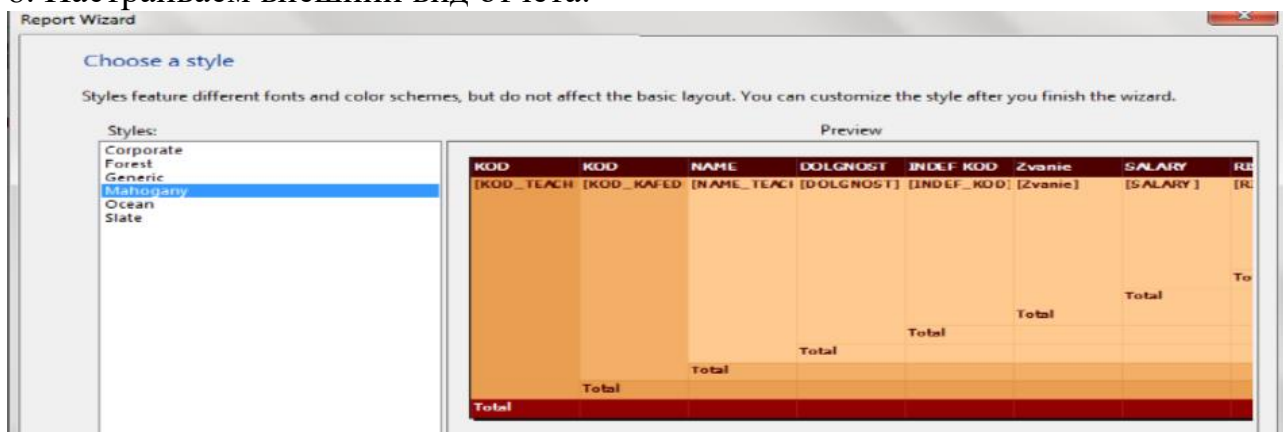
6. Нажимаем кнопку NEXT и переходим на второе диалоговое окно. В этом диалоговом окне, с помощью мастера можно настроить, как данные будут размещены в отчете. Например, добавим в отчет следующие группы:



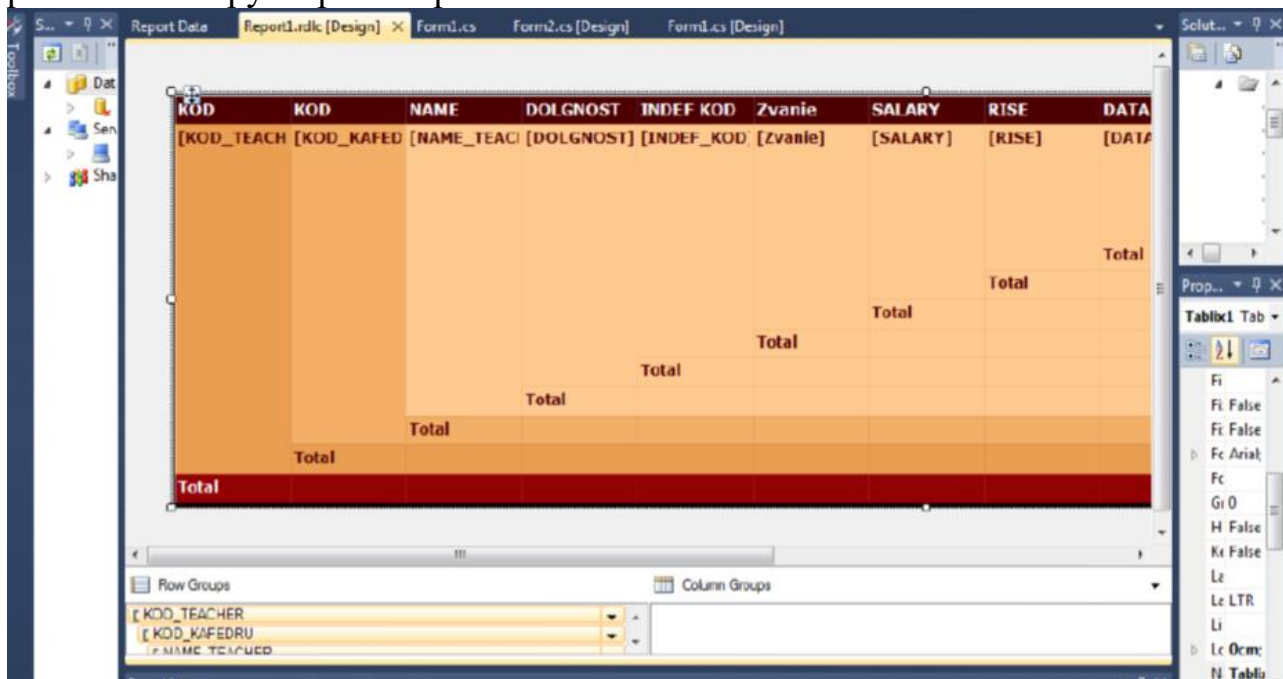
7. Настраиваем, как будут отображаться группы в отчете:



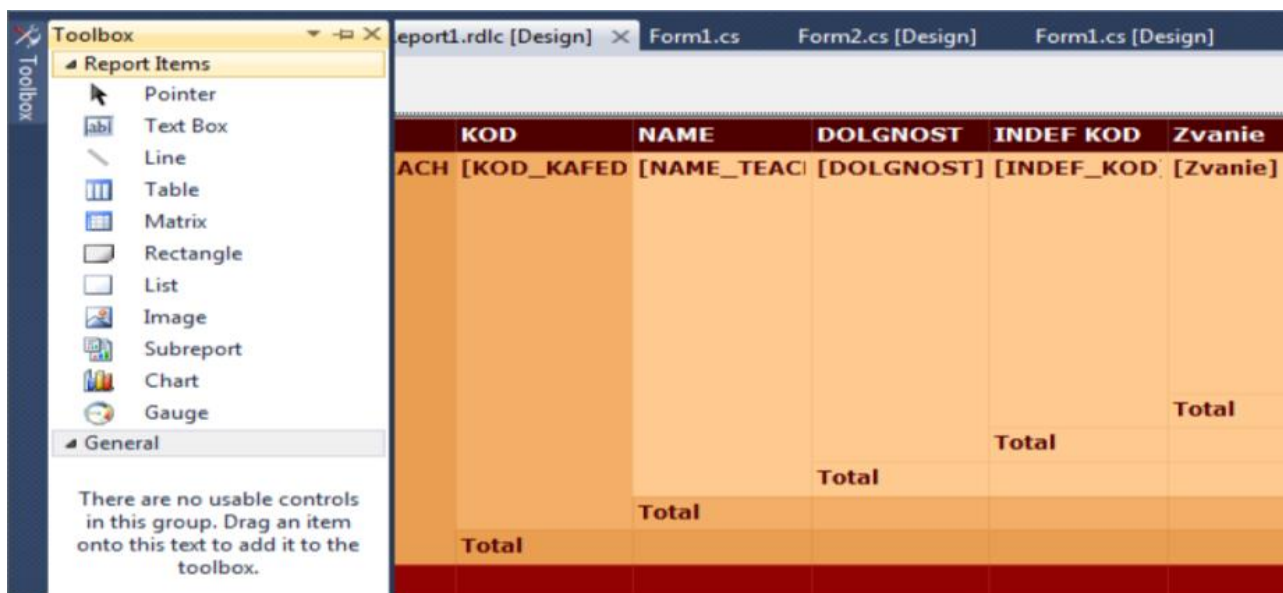
## 8. Настраиваем внешний вид отчета.



9. На последнем шаге мастера нажимаем кнопку FINISH. Итак, в результате у вас появиться новый объект проекта – Report1.rdl, который будет вам открыт в режиме конструктора. Сохранитесь.

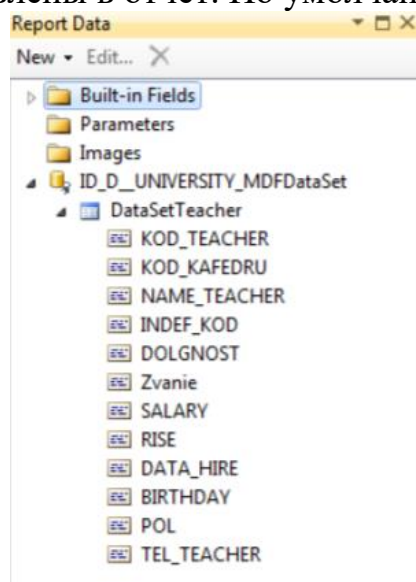


10. Отображение данных. Кликнем мышкой в поле дизайнера Report1.rdlc и в меню View выберем ToolBox. В результате в проекте отобразится ToolBox, с контролами, которые доступны для использования в ReportViewer. Здесь мы видим все доступные для отображения в ReportViever контролы.



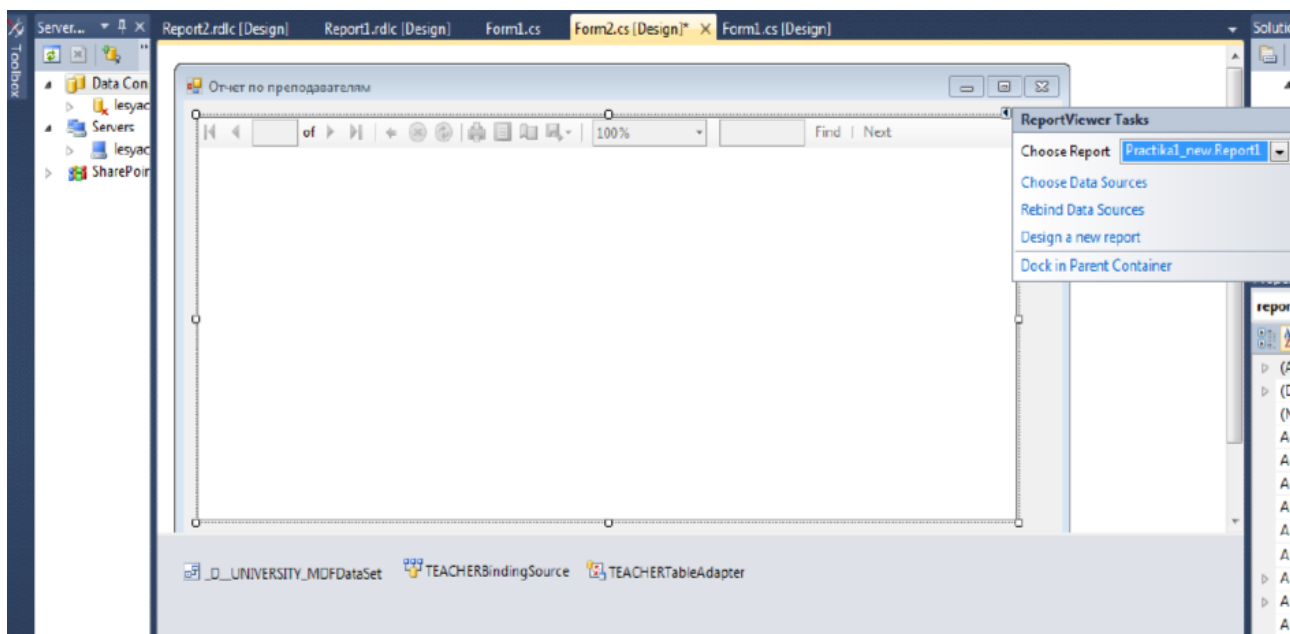
В наш отчет уже добавлена таблица для отображения. Поэтому в этот отчет мы не будем добавлять элемент таблицы.

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, TeacherBindingSource, TeacherTableAdapter.



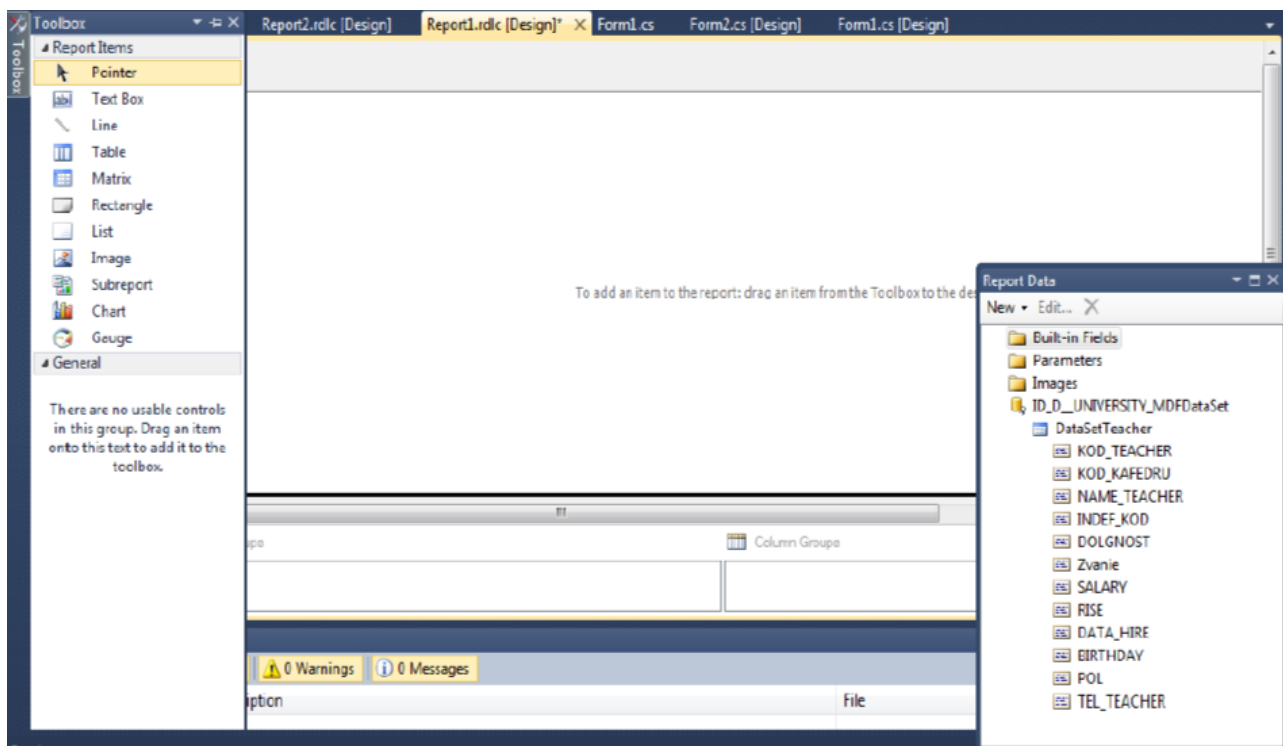
12. Запускаем проект на выполнение и нажимаем на первой форме на кнопку. После чего у вас на экране появится следующий отчет.

Отчет по преподавателям

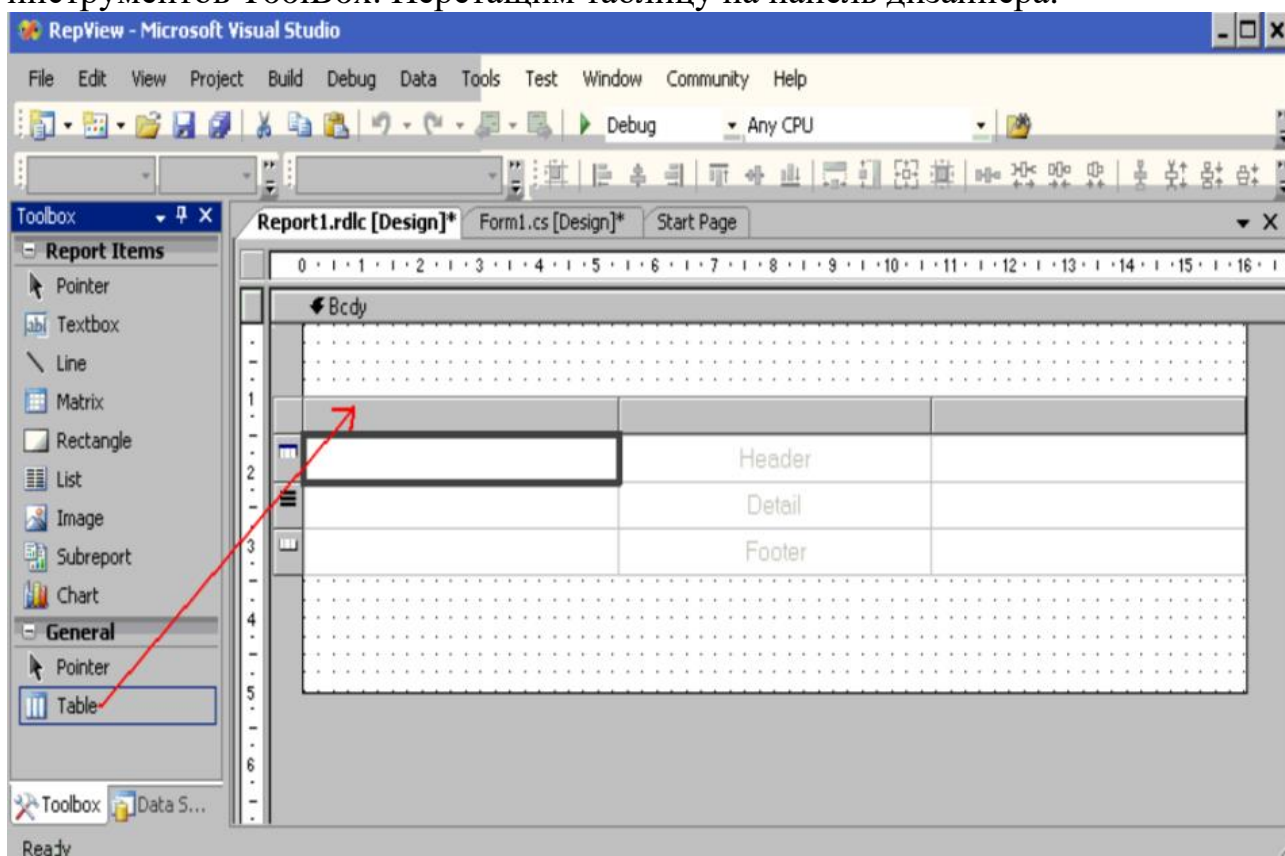
1 of 2 ? 100% Find | Next

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:



Теперь последовательно добавляем столбцы таблицы из панели ReportDate. У вас должно получиться следующее:



KOD	KOD	NAME	INDEF_KOD	DOLGNOST	Zvanie	SALARY	RISE	DATA_HIRE	BIRTHDAY
[KOD_TEACHE]	[KOD_KAFEDF]	[NAME_TEACH]	[INDEF_KOD]	[DOLGNOST]	[Zvanie]	[SALARY]	[RISE]	[DATA_HIRE]	[BIRTHDAY]

Затем, изменим подписи к столбцам, перепишем их по-русски, сделаем заливку строк, добавим границу всех ячеек таблицы.

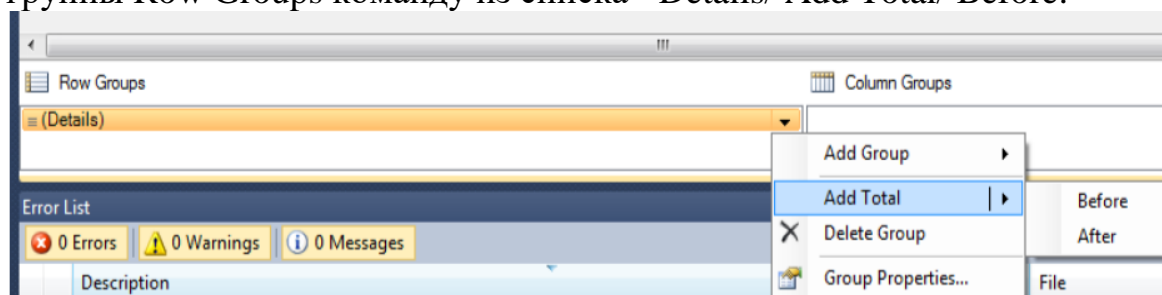
Сместите таблицу немного вниз и добавим заголовок к таблице. Для этого добавьте с панели инструментов ToolBox элемент TextBox и введите туда текст «Университет, список преподавателей». Оформите заголовок.

Университет Список преподавателей										
Код преподавателя	Код кафедры	ФИО	Индекс код	Должность	Научное звание	Ставка	Надбавка	Дата работы	Дата рождения	Пол
[KOD_TEACHER]	[KOD_KAFEDF]	[NAME_TEACH]	[INDEF_KOD]	[DOLGNOST]	[Zvanie]	[SALARY]	[RISE]	[DATA_HIRE]	[BIRTHDAY]	[POL]

Сохранимся. И запустим на выполнение проект. Просмотрите новый вид отчета.

Университет Список преподавателей										
Код преподавателя	Код кафедры	ФИО	Индекс код	Должность	Научное звание	Ставка	Надбавка	Дата работы	Дата рождения	Пол
2	9	Соловьев Виктор Иванович	00002292	доцент	к.т.н	3360.00	540.00	22.03.0		
3	9	Игнатова Олеся Владимировна	111019182	доцент	к.т.н	3360.00	54.00	22.08.0		

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.

Report1.rdlc [Design] \* Form1.cs Form2.cs [Design] Form1.cs [Design]

университет										
Код преподавателя	Код кафедры	ФИО	Идент код	Должность	Научное звание	Ставка	Надбавка	Дата работы	Дата рождения	Пол
[KOD_TEACHER]	[KOD_KAFEDR]	[NAME_TEACHER]	[INDEF_KOD]	[DOLGNOST]	[Zvanie]	[SALARY]	[RISE]	[DATA_HIRE]	[BIRTHDAY]	[POL]
[Sum(KOD_TEACHER)]	[Sum(KOD_KAFEDR)]					[Sum(SALARY)]	[Sum(RISE)]			

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).

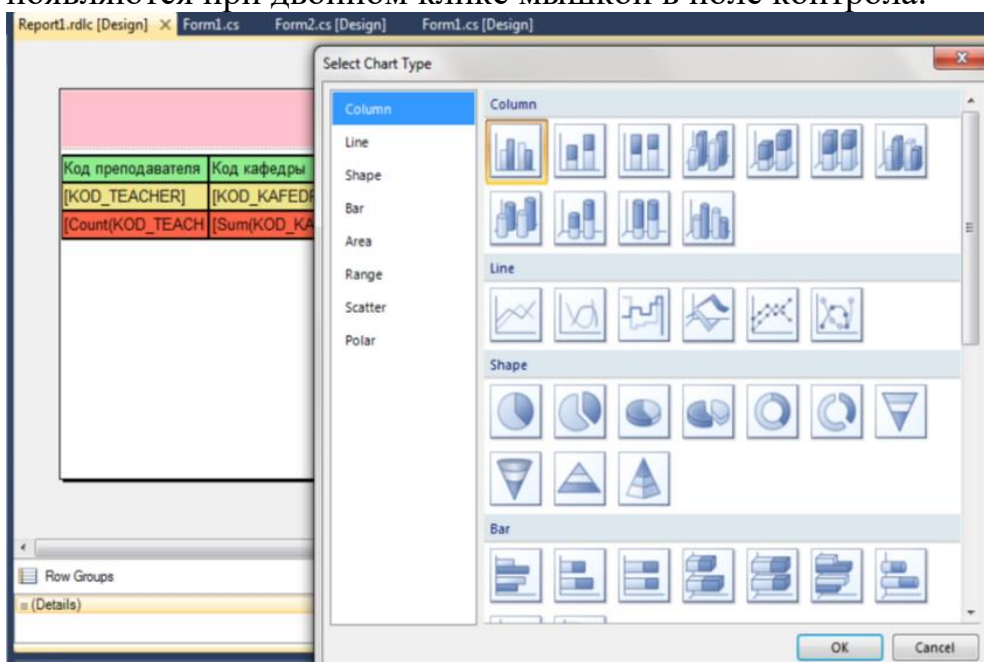
Сохранитесь и просмотрите результат. Прокрутите отчет до самого конца и в последней строке будут выведены итоги.

Отчет по преподавателям

1 of 1 100% Find | Next

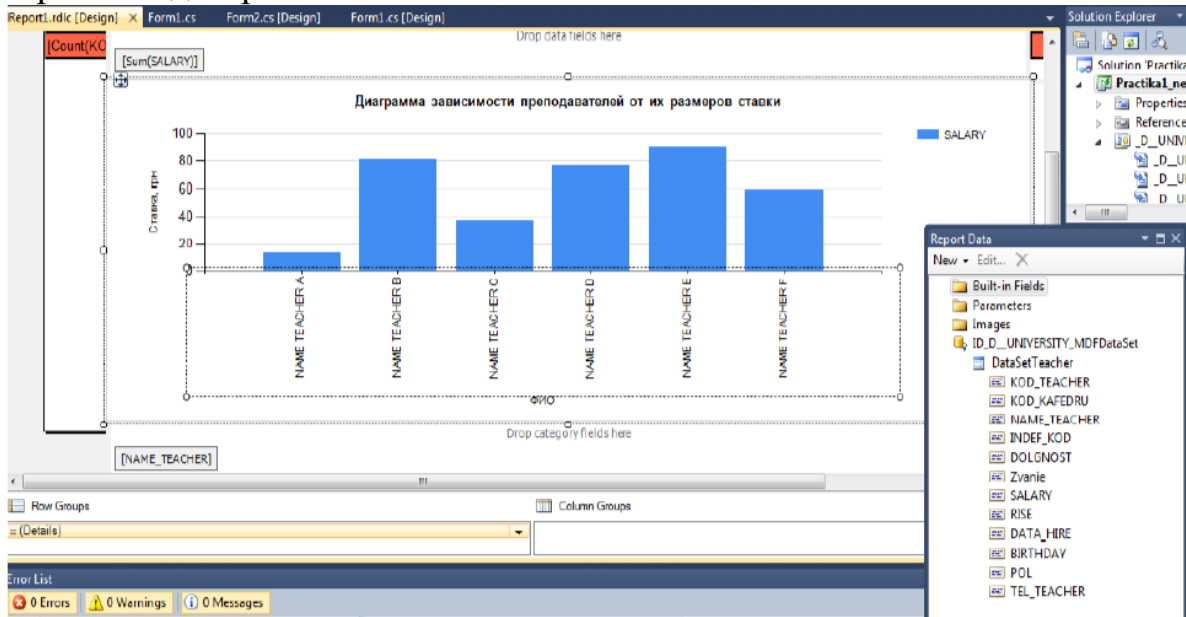
49	10	Капуста Леонид Владимирович		доцент	нет	2135.28	456.84	20.11 23
27	347					95509.68	7402.02	

15. Теперь добавим контрол Chart с вкладки Tools ReportViewer. И, как показано на рисунке, перетащим поля в своеобразные ушки контрола, которые появляются при двойном клике мышкой в поле контрола.





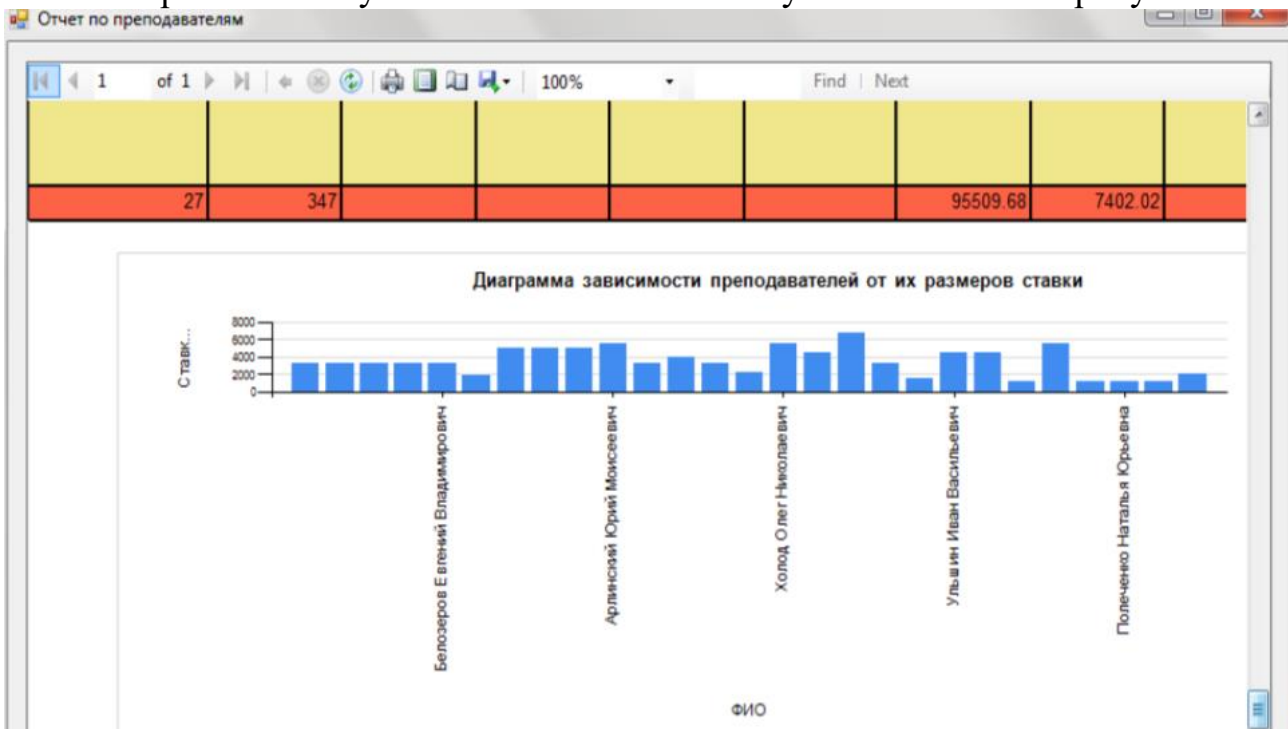
Выберите первую гистограмму и нажимаем на кнопку Ок. Диаграмма добавиться под таблицей. Но на ней пока не добавлены данные, по каким будет строиться диаграмма.



В качестве данных по оси ОХ перетащите из панели ReportDate поле NameTeacher и разместите в нижнюю область под осью Х.

В качестве данных оси ОУ 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 рисунке.



## ПРАКТИЧЕСКАЯ РАБОТА № 45

**Тема:** Создание отчетных форм в клиентских приложениях

**Цель работы:** научиться создавать формы отчетных документов по данным БД

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, MS Visual Studio, инструкции по выполнению работы.

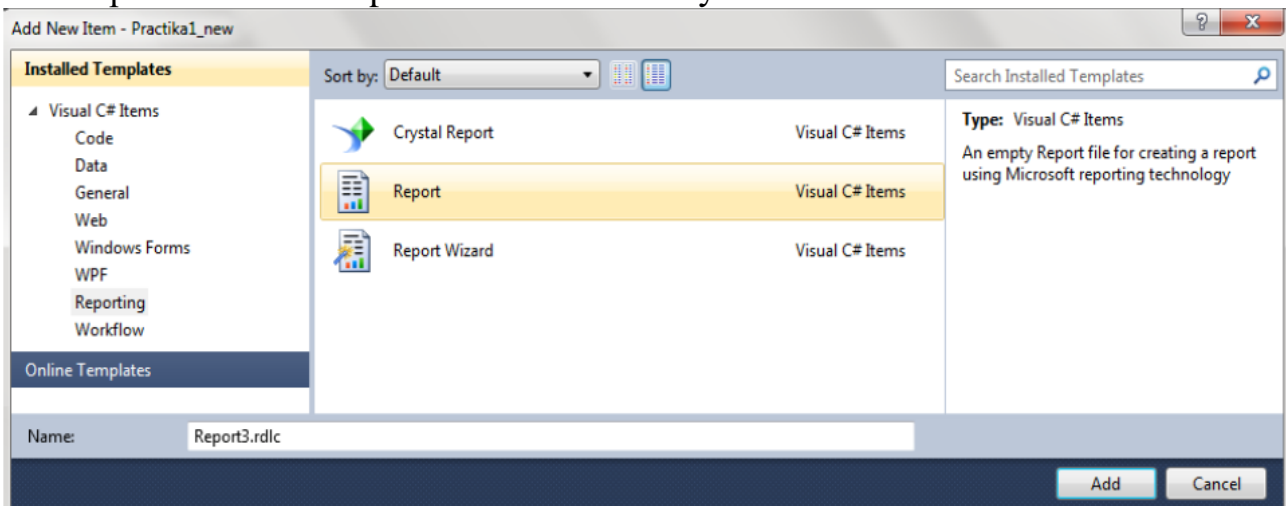
**Содержание работы:**

**Задание 1.** Определение запроса Transact-SQL для данных отчета по БД Университет

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 имеющих на них кафедрах.

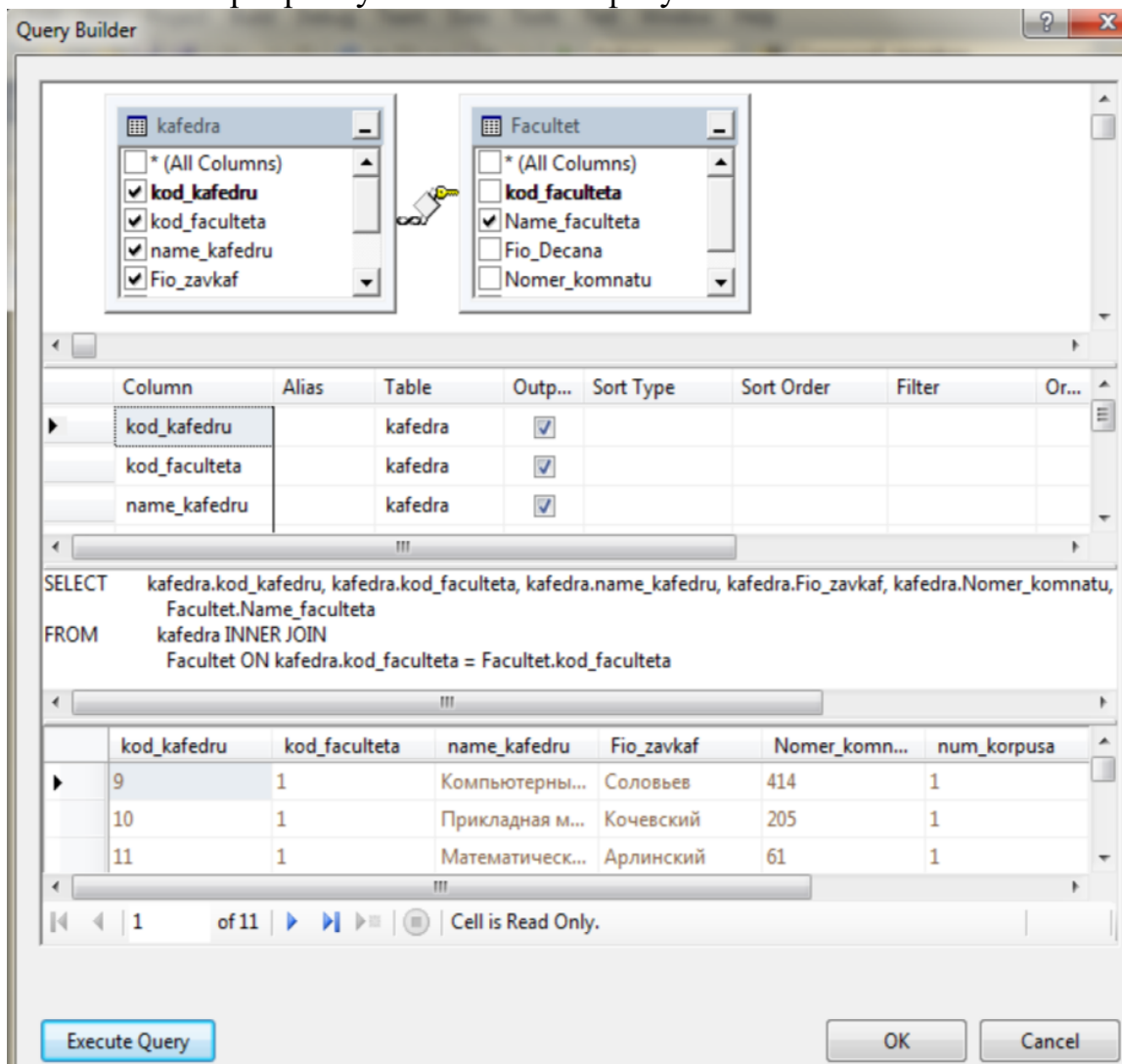
- 1) Предварительно создадим новую диалоговую форму и изменим подпись ее на «Факультеты и кафедры».
- 2) Добавьте еще одну кнопку на первую форму для открытия третьей формы.
- 3) Добавляем на новую форму элемент ReportViewer и пока не будем ничего настраивать.
- 4) Создадим пустой бланк отчета, для этого выполним команду меню Project/Add windows Form и в окне проекта выберем слева категорию Reporting и выберите элемент Report. Нажмите кнопку ADD.



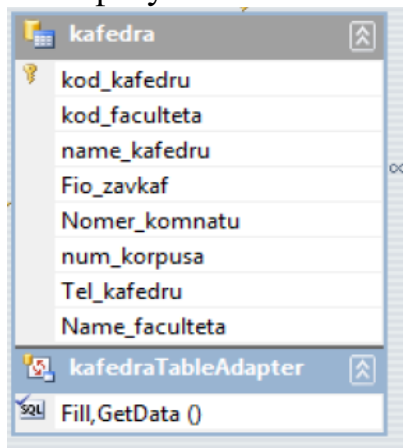
- 5) У вас создается пустой бланк нового отчета. Нам необходимо теперь создать источник данных для этого отчета.
- 6) Чтобы создать запрос для отчета перейдем в область Solution Explorer. В проводнике выберите элемент University\_DataSet. Нажмите правой кнопкой мыши на нем и выберите из меню команду Open.



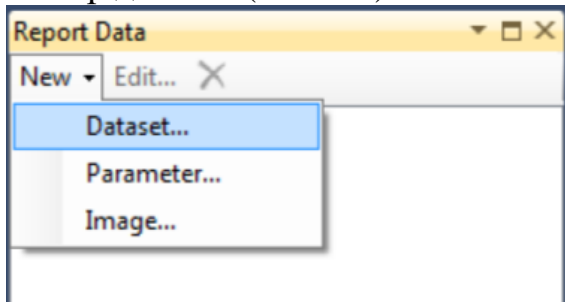
Из таблицы факультетов нас интересует только поле с названиями факультетов, поэтому нажмите галочкой на поле Name\_faculteta. Можно выполнить и сортировку по названиям факультетов.



Можно посмотреть 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.

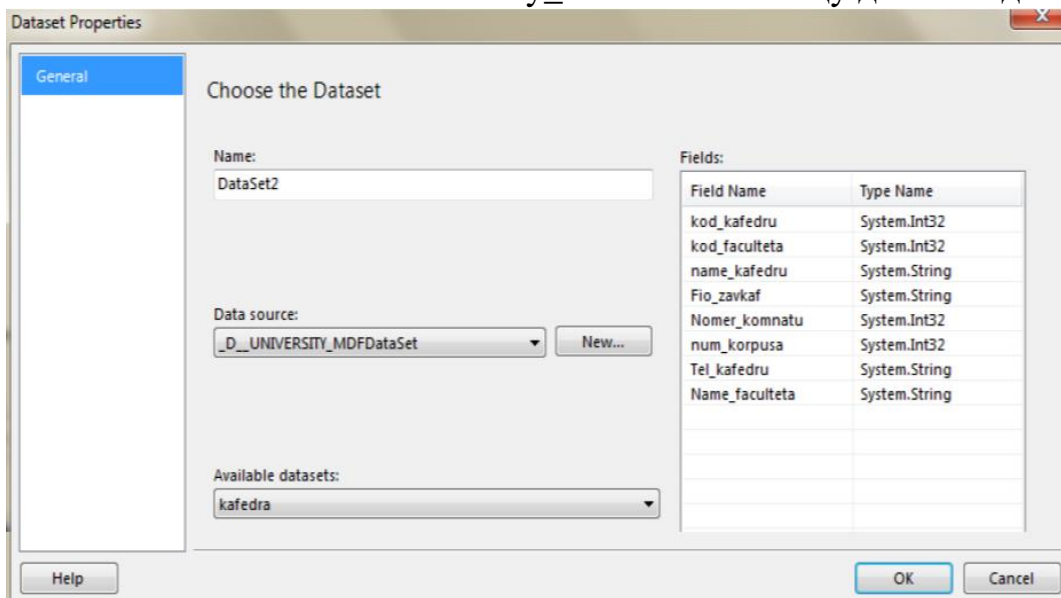


10) Возвращаемся в режим конструктора нашего второго отчета Report2.rdlrs . В области Данные отчета (Report Date) нажмите кнопку Создать (Add) и выберите Набор данных (DataSet)

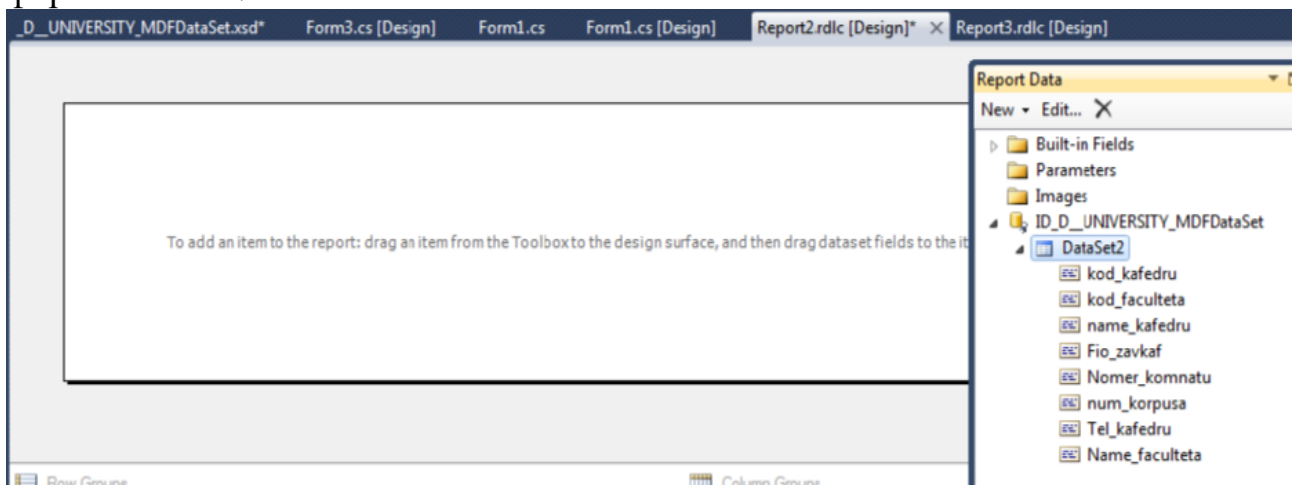


Откроется диалоговое окно Свойства набора данных.

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.



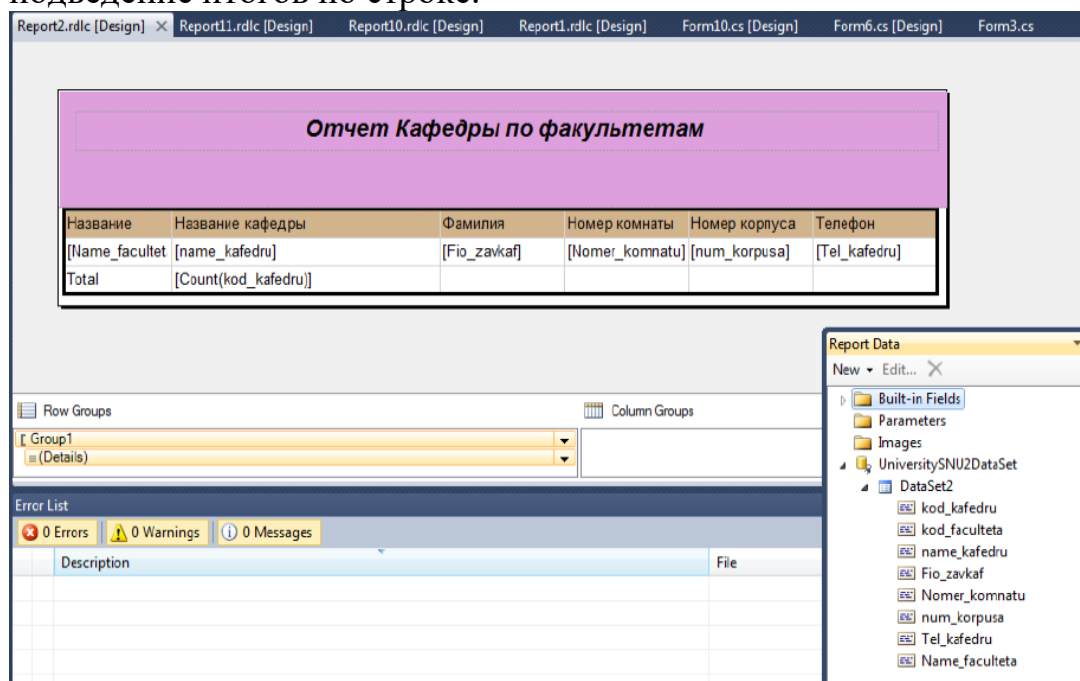
12) Добавление табличной области данных и полей в макет отчета. После определения набора данных можно приступать к определению макета отчета. Макет отчета создается путем перетаскивания в область конструктора областей

данных, текстовых полей, изображений и других элементов, которые необходимо включить в отчет.

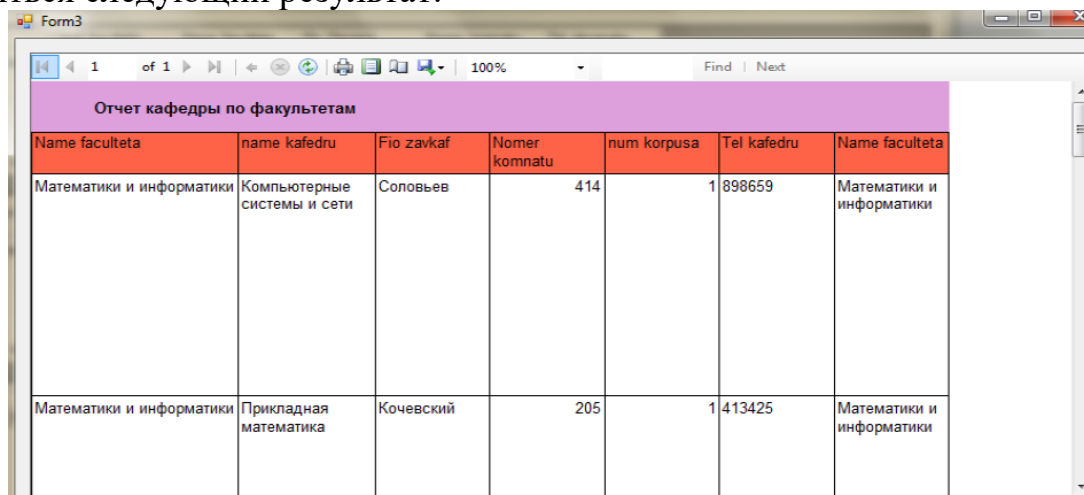
Элементы, содержащие повторяющиеся строки данных из базовых наборов данных, называются областями данных. Обычно отчеты имеют только одну область данных, но можно добавить больше, например, если требуется добавить диаграмму в табличный отчет. После добавления области данных можно добавлять в нее поля.

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 рисунке. Добавьте подведение итогов по строке.



Далее, вернитесь в третью форму, выделите объект ReportView и из треугольника выберите Report2. Сохранитесь и запустите проект на выполнение. В главной форме нажмите вторую кнопку и на экране должен появиться следующий результат:



## **ПРАКТИЧЕСКАЯ РАБОТА № 46**

**Тема:** Создание отчетных форм в клиентских приложениях

**Цель работы:** научиться создавать формы отчетных документов по данным БД

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, MS Visual Studio, инструкции по выполнению работы.

**Содержание работы:**

**Задание 1.** Для созданной базы данных Проектная организация продолжить разрабатывать клиентское приложение в среде Visual Studio C#, сделанного в Практической работе № 28. 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 2022 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

**Задание 2.** Для созданной базы данных Компания по разработке программных продуктов продолжить разрабатывать клиентское приложение в среде Visual Studio C#, сделанного в Практической работе № 29. 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 2022 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



## ПРАКТИЧЕСКАЯ РАБОТА № 47.

**Тема: Экспорт данных базы в документы пользователя**

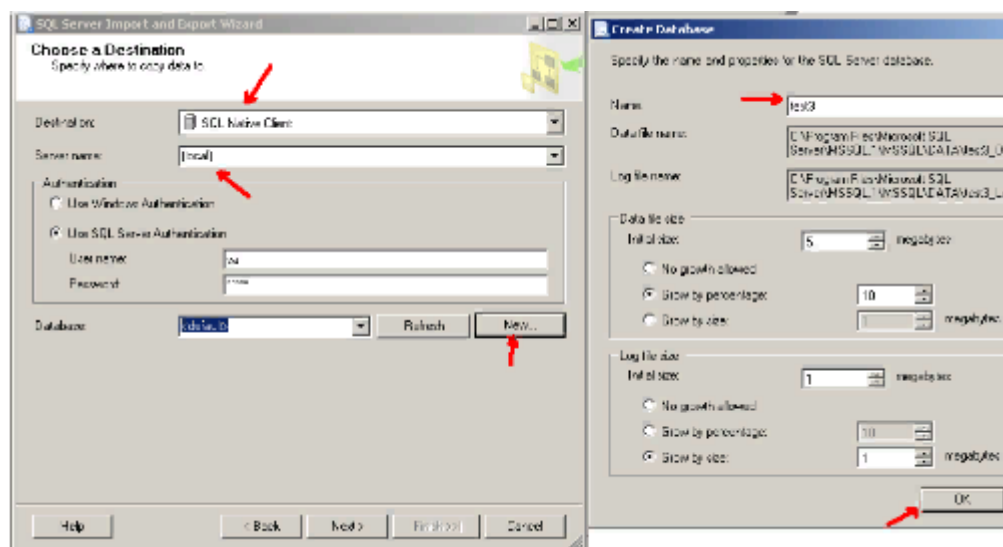
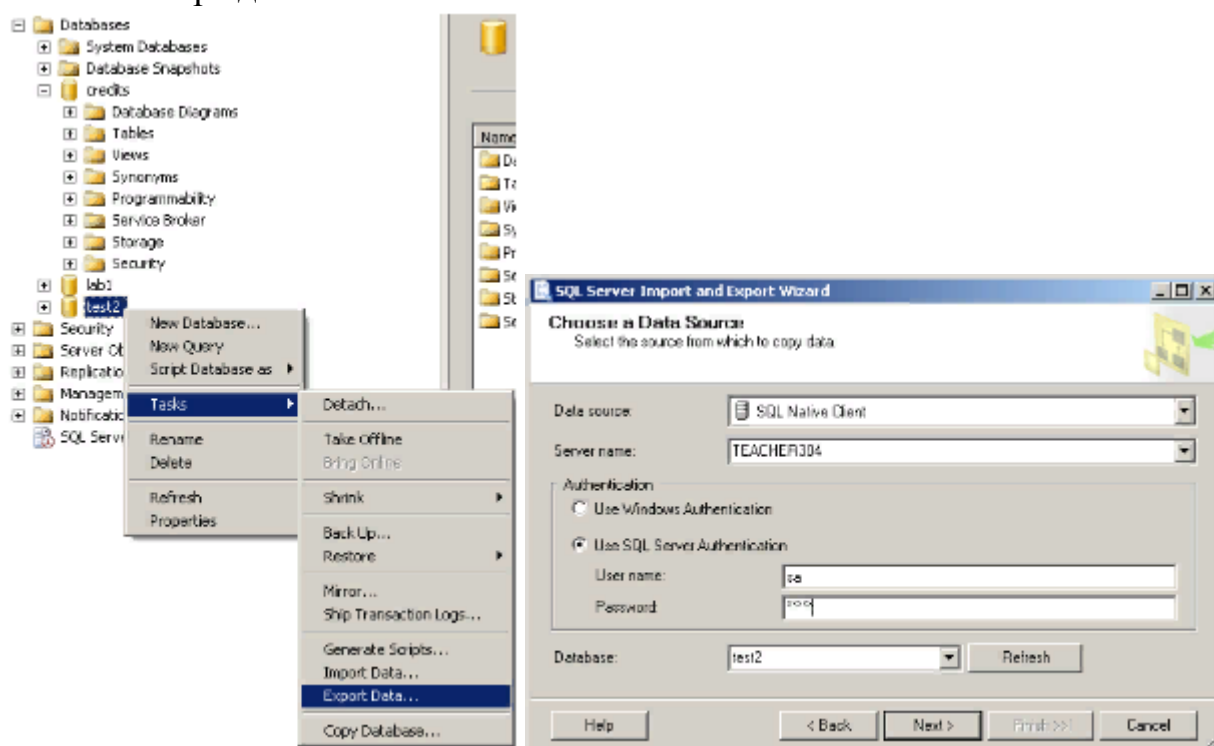
**Цель работы:** Научить создавать компоненты БД в программе MS SQL Server Management Studio, обменивать информацию между программами MS Office 2010.

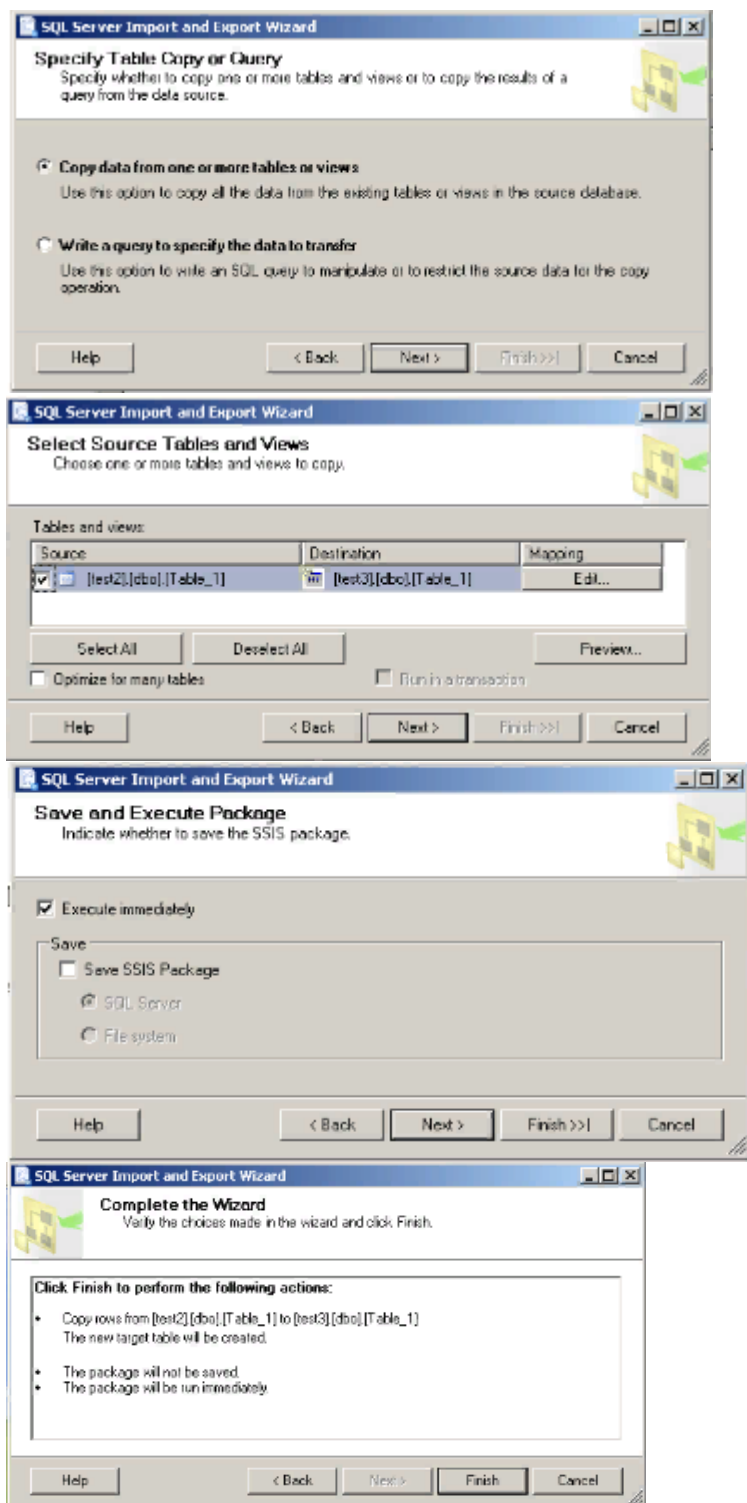
**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, MS Office, инструкции по выполнению работы.

**Содержание работы:**

**Задание 1.** Экспорт данных, созданную в Практической работе № 27 БД Университет в другую СУБД

Экспорт данных выполняется по шагам:





**Задание 2.** Выполним экспорт данных из таблицы Сотрудники (проекты) в текстовый документ в формате RTF. Для этого выполните следующие простые действия:

1. Откройте базу данных Университет, созданную в Практической работе № 27.
2. Перейдите на закладку **Таблицы**, выделите таблицу Сотрудники.
3. Выполните меню **Файл/Экспорт**.
4. Укажите тип файла – формат RTF, имя файла оставьте без изменения, укажите свою папку и экспортируйте туда файл.

Процесс экспорта закончен. Чтобы просмотреть результат откройте свою папку и откройте тестовый файл Сотрудники.rtf.

**Задание 3.** Выполнить экспорт данных из БД «Проектная организация», созданной в Практической работе № 28, в документы формата .docx.

**Задание 4.** Произведите экспорт данных из БД «Компания по разработке программных продуктов», созданную в Практической работе № 29, в документы формата .docx.

## ПРАКТИЧЕСКАЯ РАБОТА № 48.

### Тема: Импорт данных пользователя в базу данных

**Цель работы:** Научить создавать компоненты БД в программе MS SQL Server Management Studio, обменивать информацию между программами MS Office 2010.

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, MS Office, инструкции по выполнению работы.

#### Содержание работы:

**Задание 1.** В Excel хранятся данные о поставщиках (файл ПоставщикиExcel.xls), клиентах (КлиентыExcel.xls), товарах (ТоварыExcel.xls). Откройте эти файлы на своих компьютерах (D:/Studentu/Access/) и просмотрите их.

Чтобы эти данные не вводить в базу данных можно воспользоваться мастером **Импорта данных**. Для этого выполните следующие действия:

1. Откройте базу данных.
2. Перейдите на вкладку **Таблицы**.
3. Откройте меню **Файл/Внешние данные/Импорт**.
4. Укажите путь к файлу **ПоставщикиExcel.xls** и тип файлов – Microsoft Excel (\*.xls) и нажмите кнопку **Импорт**.
5. У вас появится диалоговое окно мастера Импорта данных.
6. В первом шаге мастера установите флаг «**Первая строка содержит заголовки столбцов**». Нажмите **Далее**.
7. Во втором шаге мастер будет спрашивать, куда сохранять данные - либо в новую таблицу, либо в уже созданную. Выберите **созданную** таблицу Поставщики. Нажмите **Далее**.
8. В следующем шаге мастер ждет подтверждения импорта данных в выбранную таблицу. Нажмите **Готово**.

Если в вашей таблице Поставщики все поля были созданы верно и имена совпадают с именами полей добавляемой таблицы ПоставщикиExcel.xls и порядок следования имен полей тоже совпадают, то в результате импорта будут добавлены данные о Поставщиках. Откройте таблицу и просмотрите ее.

**Самостоятельно** аналогично добавьте данные в таблицы **Клиенты** и **Товары**.

Однако добавление данных этих трех таблиц недостаточно, так как мы не ввели главные данные, с которыми потом в дальнейшем будем работать, а именно это данные наших приходных и расходных накладных.

Необходимо добавить данные из таблиц ПриходExcel.xls, Ввод\_приходаExcel.xls и РасходExcel.xls, Ввод\_расходаExcel.xls.

И тут возникает проблема. Дело в том, чтобы при осуществлении операции импорта данных возможно однозначное добавление данных в простые таблицы. При этом данные добавляются в список, где каждой новой записи соответствует номер из добавленной таблицы.

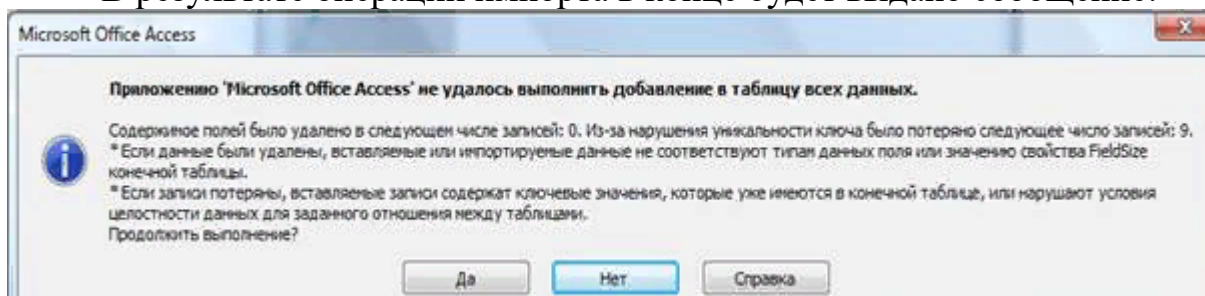
Например, если таблица **Товары** содержала бы следующие данные:

Код_товара	Название_товара	Цена_приходная	Цена_расходная	Примечание	Единица_измерения
	Телевизор Sony	\$2,00	\$4,00		шт
	Телевизор Nokia	\$3,00	\$3,50		шт
	Флеш-носитель 1Гб	\$2,00	\$3,00		шт
	Флеш-носитель 2Гб	\$3,00	\$3,50		шт
	Флеш-носитель 4Гб	\$4,00	\$4,50		шт
	Флеш-носитель 8Гб	\$5,00	\$5,80		шт
	Фото-принтер	\$80,00	\$90,00		шт
	Лазерный принтер	\$51,00	\$15,00		шт
	Струнный принтер	\$120,00	\$170,00		шт

А необходимо добавить данные из таблицы **ТоварыExcel.xls**:

Код_товара	Название_товара	Цена_приходная	Цена_расходная	Примечание	Единица_измерения
	Каша рисовая	2,00р.	4,00р.		литр
	Каша гречневая	3,00р.	3,50р.		ящик
	Каша с курицей	2,00р.	3,00р.		литр
	Каша с грибами	3,00р.	3,50р.		ящик
	Крышка закаточная	4,00р.	4,50р.		ящик
	Паста томатная	5,00р.	5,80р.		шт
	Печенье овсяное	8,00р.	9,00р.		шт
	Печенье ороматное	4,00р.	4,30р.		шт
	Молоко	6,70р.	7,00р.		шт
	Сливки	12,10р.	12,30р.		шт
	Масло сливочное	5,00р.	8,00р.		шт
	Масло оливковое	1,00р.	2,00р.		шт
	Повидло яблочное	2,00р.	5,00р.		ящик
	Повидло сливовое	1,00р.	2,00р.		ящик
	Повидло абрикосовое	4,00р.	6,00р.		шт
	Торт Изобелла	1,00р.	2,00р.		шт
	Торт классический	6,00р.	7,00р.		шт
	Торт пикатный	6,00р.	7,00р.		шт
	Кофе Нескафе	9,80р.	10,00р.		
	Кофе Классик	9,60р.	10,00р.		шт

В результате операции импорта в конце будет выдано сообщение:



Если вы нажмете кнопку Да, то после операции импорта данных получится следующее:

Код_товара	Название_товара	Цена_приходная	Цена_расходная	Примечание	Единица_измерения
	Телевизор Sony	\$2,00	\$4,00		шт
	Телевизор Nokia	\$3,00	\$3,50		шт
	Флеш-носитель 1Гб	\$2,00	\$3,00		шт
	Флеш-носитель 2Гб	\$3,00	\$3,50		шт
	Флеш-носитель 4Гб	\$4,00	\$4,50		шт
	Флеш-носитель 8Гб	\$5,00	\$5,80		шт
	Фото-принтер	\$80,00	\$90,00		шт
	Лазерный принтер	\$51,00	\$15,00		шт
	Струнный принтер	\$120,00	\$170,00		шт
	Сливки	\$12,10	\$12,30		шт
	Масло сливочное	\$5,00	\$8,00		шт
	Масло оливковое	\$1,00	\$2,00		шт
	Повидло яблочное	\$2,00	\$5,00		ящик
	Повидло сливочное	\$1,00	\$2,00		ящик
	Повидло абрикосовое	\$4,00	\$6,00		шт
	Торт Изобелла	\$1,00	\$2,00		шт
	Торт классический	\$6,00	\$7,00		шт
	Торт пикатный	\$6,00	\$7,00		шт
	Кофе Нескафе	\$9,80	\$10,00		шт
	Кофе Классик	\$9,60	\$10,00		шт

Если проанализировать, то первые 9 записей остались без изменения, а новые данные добавились в конец списка, где номер кода товара соответствуют кодам товаров из добавляемой таблицы.

А если мы хотим добавить данные в связанные таблицы, как это в таблицах Приход и Ввод\_прихода и Расход и Ввод\_Расхода, то простой операцией импорта ничего не получится, так как мы не сможем обеспечить однозначность и целостность данных.

С чем это связано? Дело в том, что при выполнении операции импорта данных будут добавлены записи под старыми кодами. А коды в связанных таблицах должны совпадать, например код\_прихода или код\_расхода. Код\_прихода или код\_расхода являются ключевыми полями, они однозначно определяют набор записей в связанных таблицах Приход и Ввод\_приход или Расход и Ввод\_расхода.

Например, запись в таблице Приход

Код_прихода	Поставщик	Дата_накладной	Идентифик№поставщика	ФИОпоставщика
	МЧП ЛЕЛЕКА	02.10.2002		Моринский

Однозначно соответствуют записи в таблице Ввод\_прихода:

Код_ввода_прихода	Код_прихода	Товар	цена	кол-во	сумма
		Торт классический	6,00р.		12,00р.
		Масло оливковое	1,00р.		89,00р.

Проблема заключается в том, что нам нужно будет добавить данные не в одну таблицу, а сразу в две связанные. При этом вначале необходимо добавить

данные в главную таблицу, например Приход. Сразу же возникает особенность данной операции. Если бы мы выполняли добавление через импорт данных, то будут созданы записи под кодами, совпадающими с таблицей ПриходExcel.xls, при этом будут частично потеряны данные. Затем необходимо будет добавить данные в подчиненную ей таблицу Ввод\_прихода. А так как уже были потеряны данные, то и к связанной таблице они тоже не добавятся.

Как же решить данную проблему? Решим ее с помощью специального вида запросов – **запроса на добавление**. При этом мы будем добавлять все записи в конец списка и создавать новые уникальные ключи записей.

Но прежде чем переходить к этим видам запросов создадим новые таблицы для хранения внешних данных.

Для этого выполните следующие действия:

1. Перейдите на вкладку **Таблицы**.
2. Откройте меню **Файл/Внешние данные/Импорт.../**
3. Выберите Тип файлов – **Microsoft Excel (\*.xls)**
4. Найдите файл под именем **ПриходExcel.xls** и нажмите кнопку **Импорт**.
5. В открывшемся диалоговом окне мастера Импорта электронной таблицы на вопрос «Файлы электронной таблицы содержит несколько листов и диапазонов. Выберите нужный объект» установите опцию **Листы**.
6. Во втором шаге мастера выберите Первая строка содержит заголовки столбцов.
7. На третьем шаге мастера выберите Данные необходимо добавить в **новую таблицу**.
8. На четвертом шаге имеется возможность описать каждое поле импорта. По умолчанию для всех полей автоматически определяются типы данных, кроме индексированного поля. Для этого выберите из списка Имя поля– поле **Код\_прихода**, а в списке Индекс выберите – **Да (совпадения не допускаются)**.
9. На пятом шаге мастера рекомендуется создать ключевое поле либо автоматически, либо самостоятельно определить, либо не создавать ключ. Выберите Определить ключ – **Код\_прихода**.
10. На шестом шаге мастера введите имя файла – **Приход\_импорт**.

**Самостоятельно** аналогично создайте новые таблицы под именами Ввод\_прихода\_импорт, Расход\_импорт, Ввод\_расхода\_импорт используя импортированные электронные таблицы.

**Задание 2.** Произведите импорт данных из таблиц Excel в базу данных «Проектная организация», созданную в Практической работе № 28

**Задание 3.** Произведите импорт данных из таблиц Excel в базу данных «Университет», созданную в Практической работе № 27

**Задание 4.** Произведите импорт данных из таблиц Excel в базу данных «Компания по разработке программных продуктов», созданную в Практической работе № 29



## ПРАКТИЧЕСКАЯ РАБОТА № 49.

**Тема: Выполнение настроек для автоматизации обслуживания БД**

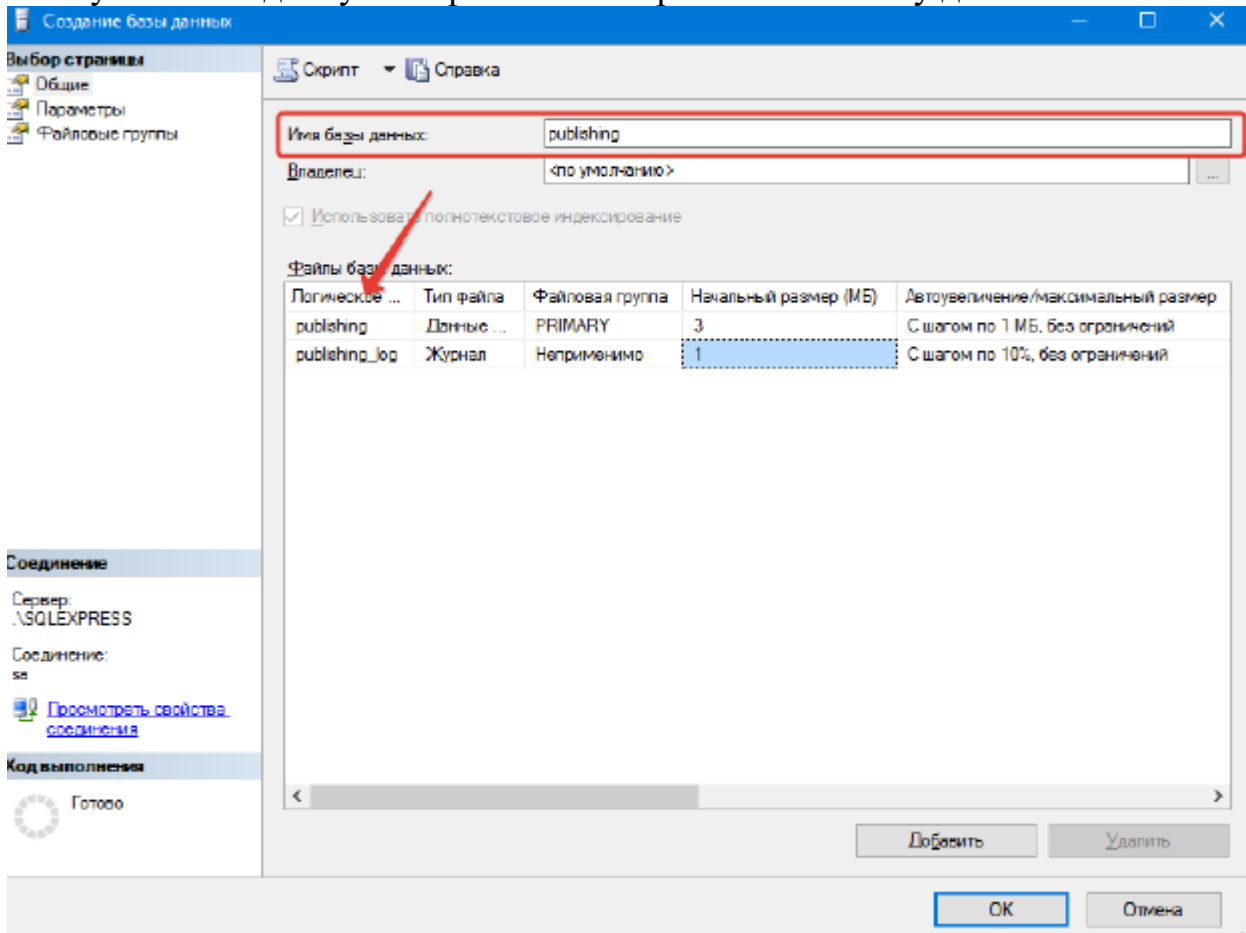
**Цель работы:** Освоение некоторых возможностей автоматизации управления базой данных. Создание и применение макросов. Создание пользовательского кнопочного и ниспадающего меню.

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, инструкции по выполнению работы.

**Содержание работы:**

**Задание 1.** Произвести настройки параметров файла данных созданной в Практической работе № 23 базы данных.

1. Запустите созданную в Практической работе № 23 базу данных



В левой части окна настроек имеется список «Выбор страницы». Этот список позволяет переключаться между группами настроек.

2. Для начала настроим основные настройки. Для выбора основных настроек нужно просто щёлкнуть мышью по пункту «Общие» в списке «Выбор страницы». В правой части окна «Создание базы данных» появятся основные настройки.

В верхней части окна расположено два параметра: «Имя базы данных» и «Владелец». Задайте параметр «Имя базы данных», таким образом, чтобы имя полностью описывало Выбранную Вами предметную область, в первой лабораторной работе был рассмотрен процесс проектирования БД издательства, поэтому название будет соответствующее, но на обязательно на английском языке – «Publishing». Параметр Владелец оставьте без изменений.

Под вышеприведёнными параметрами в виде таблицы располагаются настройки файла данных и журнала транзакций. Таблица имеет следующие столбцы:

- Логическое имя – логическое имя файла данных и журнала транзакций. По этим именам будет происходить обращение к вышеприведённым файлам в БД. Можно заметить, что файл данных имеет то же имя что и БД, а имя файла журнала транзакций составлено из имени БД и суффикса «\_log».
- Тип файла – этот параметр показывает, является ли файл файлом данных или журналом транзакций.
- Файловая группа – группа файлов, показывает к какой группе файлов относится файл. Группы файлов настраиваются в группе настроек Файловая группа.
- Начальный размер (МБ) – начальный размер файла данных и журнала транзакций в мегабайтах.
- Авторасширение – как только файл заполняется информацией его размер автоматически увеличивается на величину, указанную в параметре Авторасширение. Увеличение можно задавать как в мегабайтах так и в процентах. Здесь же можно задать максимальный размер файлов. Для изменения этого параметра надо нажать кнопку «...». В нашем случае размер файлов не ограничен. Файл данных увеличивается на 1 мегабайт, а файл журнала транзакций на 10%.
- Путь – путь к папке, где хранятся файлы. Для изменения этого параметра также надо нажать кнопку «...».
- Имена файлов – по умолчанию имена файлов аналогичны логическим именам. Однако файл данных имеет расширение .mdf, а файл журнала транзакций – расширение .ldf.

В рассматриваемом случае все основные настройки без изменений были оставлены без изменений.

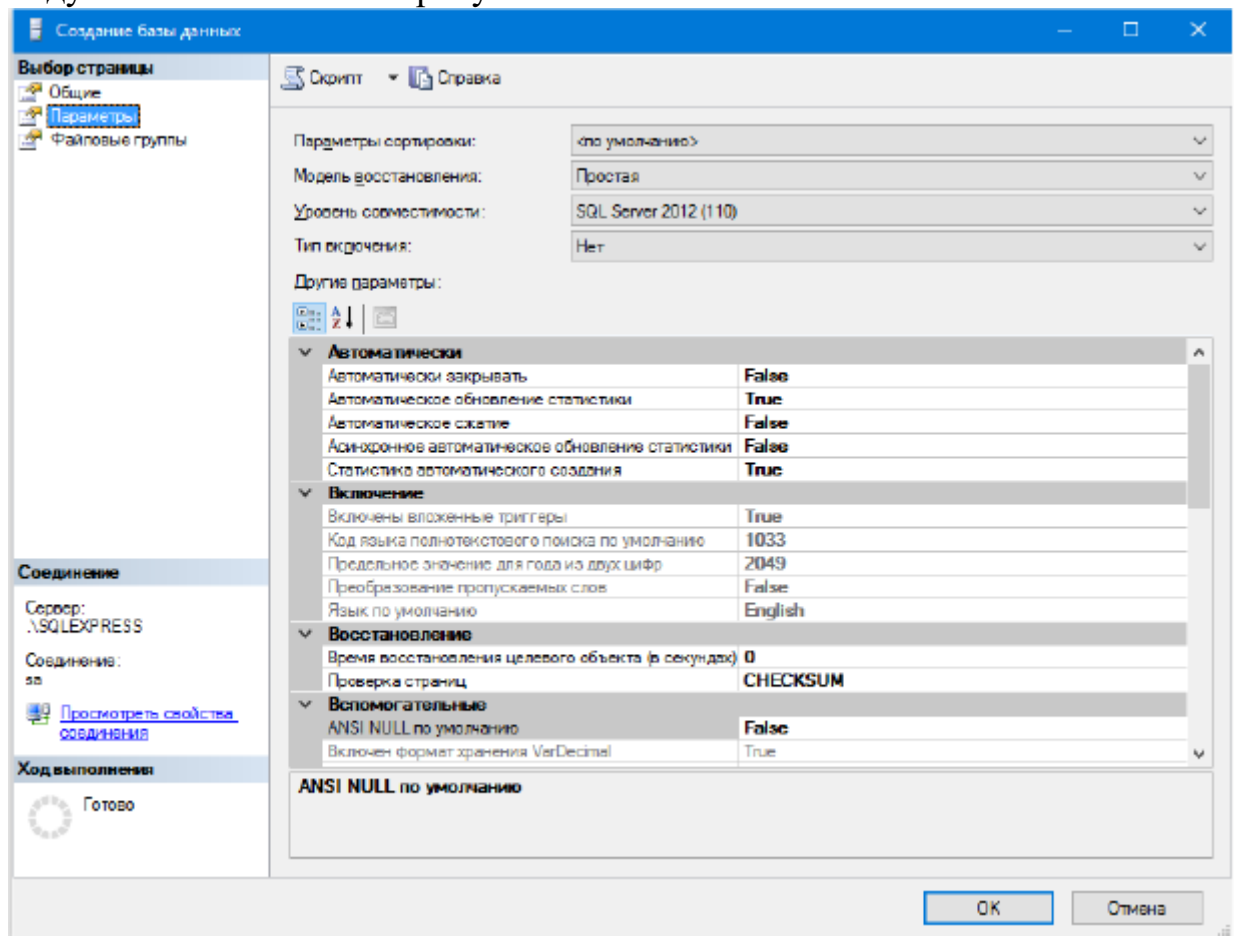
Теперь перейдём к другим второстепенным настройкам файла данных. Для доступа к этим настройкам необходимо щёлкнуть мышью по пункту Параметры в списке Выбор страницы. Появится следующее окно.

В правой части окна мы видим следующие настройки:

- Параметры сортировки – этот параметр отвечает за обработку текстовых строк, их сравнение, текстовый поиск и т.д. Рекомендуется оставить его как <по умолчанию сервера>. При этом данный параметр будет равен значению, заданному на вкладке Параметры сортировки, при установке сервера.
- Модель восстановления – данный параметр отвечает за информацию, предназначенную для восстановления БД, хранящуюся в файле транзакций. Чем полнее модель восстановления, тем больше вероятность восстановления данных при сбое системы или ошибках пользователей, но и больше размер файла журнала транзакций. При наличии места на диске, рекомендуется оставить этот параметр в значении Простая.
- Уровень совместимости – определяет совместимость файла данных с более ранними версиями сервера. Если планируется перенос данных на другую, более раннюю версию сервера, то её необходимо указать в этом параметре.

- Другие параметры – данные параметры являются необязательными для изменения.

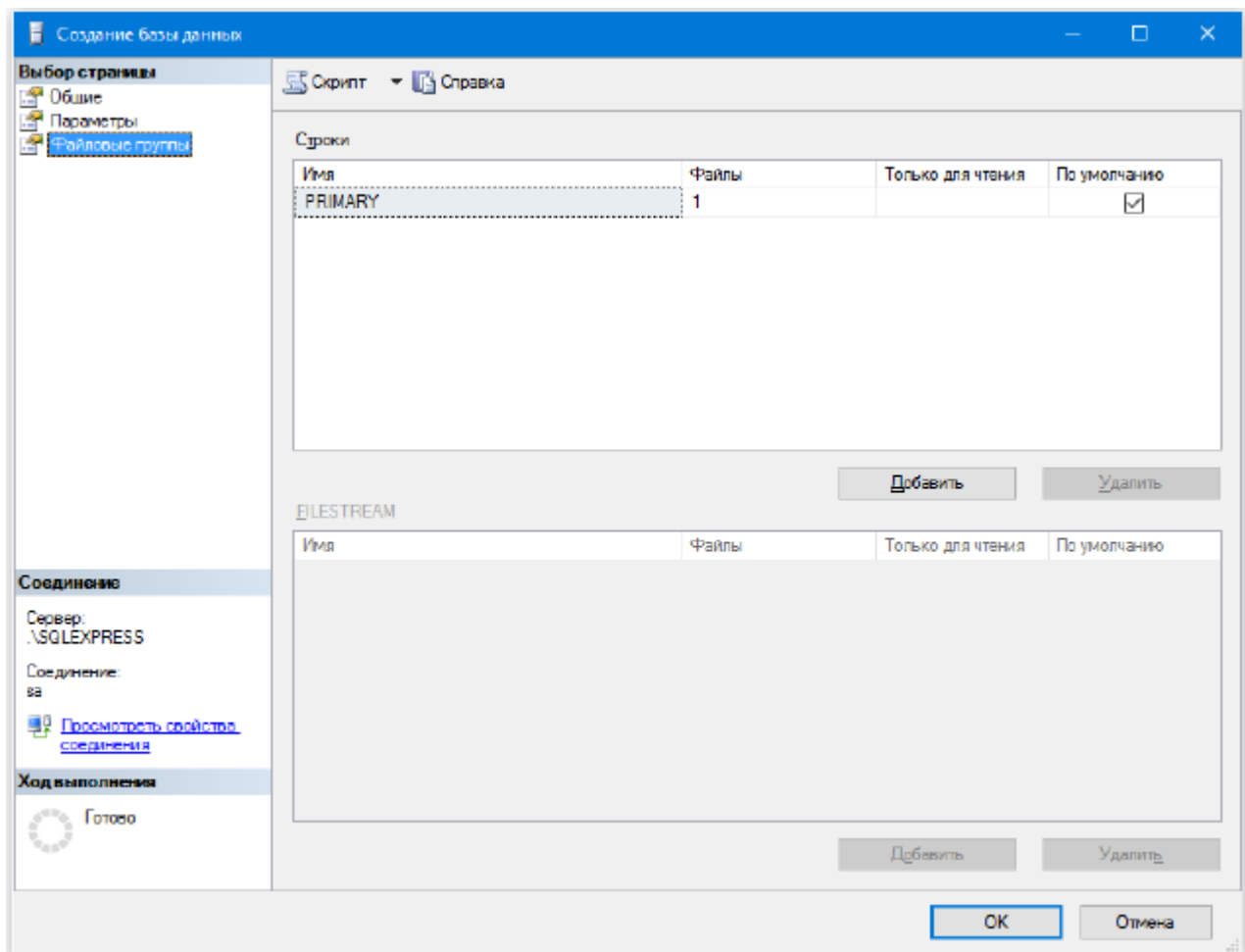
В нашем случае все параметры в разделе Другие параметры, рекомендуется оставить как на рисунке



3. Наконец рассмотрим последнюю группу настроек Файловые группы. Данная группа настроек отвечает за группы файлов. Группы файлов представлены в таблице Строки в правой части окна. Данная таблица имеет следующие столбцы:

- Имя – имя группы файлов.
- Файлы – количество файлов, входящих в группу.
- Только для чтения – файлы в группе будут только для чтения. То есть, их можно только просматривать, но нельзя изменять.
- По умолчанию – группа по умолчанию. Все новые файлы данных будут входить в эту группу.

В рассматриваемой БД нет необходимости добавлять новые группы файлов. Поэтому оставим группу настроек Файловые группы без изменений.



Для принятия всех настроек и создание фала данных и журнала транзакций нашей БД в окне «Создание базы данных» нажмём кнопку ОК. Произойдёт возврат в окно среду разработки SQL Server Management Studio.

## ПРАКТИЧЕСКАЯ РАБОТА № 50

### Тема: Выполнение резервного копирования

**Цель работы:** Освоить технологию резервного копирования

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, инструкции по выполнению работы.

#### Справочный материал:

Резервное копирование (backup) базы данных и восстановление из резервной копии (restore) – два важнейших и наиболее частых процесса, осуществляемых администраторами баз данных.

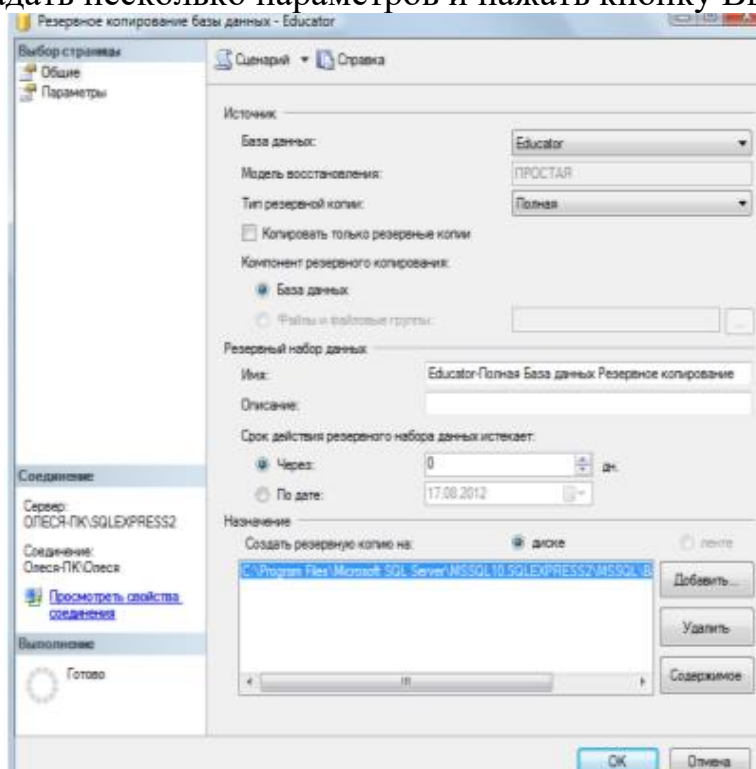
Резервное копирование базы данных – единственный надежный способ предохранить данные от потери в результате поломки диска, сбоев электропитания, действий злоумышленников и ошибок в программах. В процессе резервного копирования создается независимый от платформы "снимок" базы данных, с помощью которого можно перенести данные на другую операционную систему или даже другую платформу.

Полный цикл: резервное копирование и восстановление из резервной копии приводит к корректировке статистической информации, является средством от излишнего "разбухания" базы данных и необходимой операцией обслуживания базы данных. Кроме того, миграция от одной версии сервера к другой также происходит при помощи процесса backup/restore.

#### Содержание работы:

**Задание 1.** Создать резервную копию БД Университет

Для создания резервной копии базы данных с помощью программы "SQL Server Management Studio" необходимо подключиться к базе данных Университет, выбрать из контекстного меню базы данных Задачи/ Создать резервную копию. В открывшемся диалоговом окне "Мастер резервного копирования" задать несколько параметров и нажать кнопку Выполнить.



После выбора пути и файла для резервной копии в окне Back Up Database нажатием на ОК запускаем процесс создания резервной копии. В случае успешной работы появится сообщение.

В результате будет создан файл с резервной копией. Стандартным расширением таких файлов для " SQL Server Management Studio " является "\*.bak". Файл с резервной копией базы данных обычно на порядок меньше оригинала.

**Задание 2.** Выполнить резервное копирование баз данных «Проектная организация», «Учебный центр», «Торговая фирма», «Компания по разработке программных продуктов»

## ПРАКТИЧЕСКАЯ РАБОТА № 51

### Тема: Выполнение резервного копирования

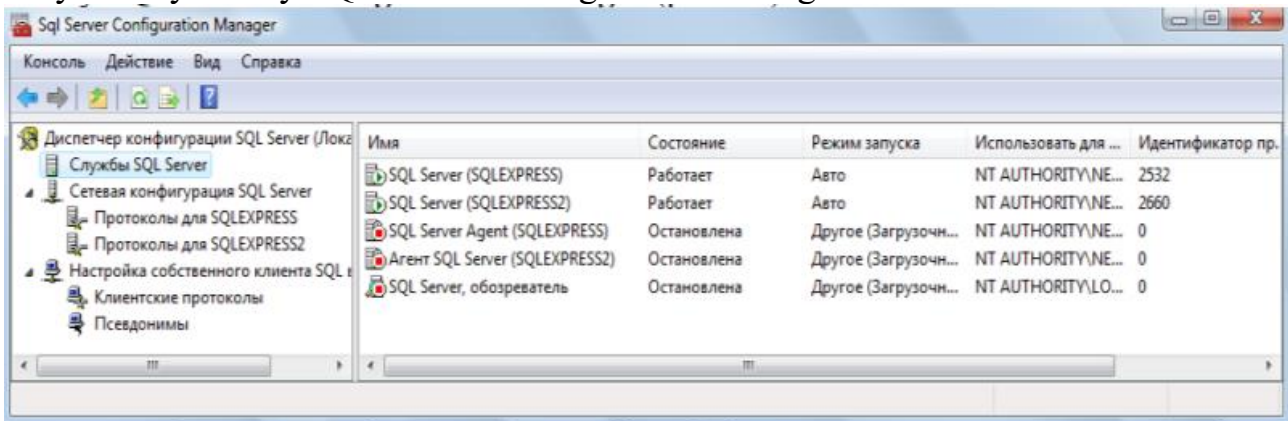
**Цель работы:** Освоить технологию резервного копирования

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, инструкции по выполнению работы.

**Содержание работы:**

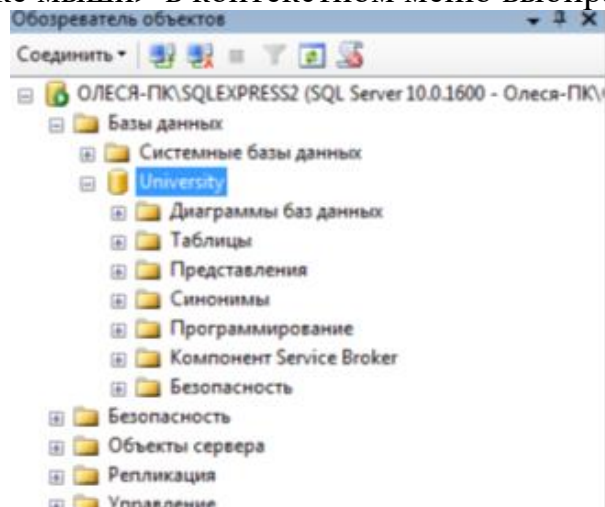
**Задание 1.** Произвести копирование и перенос базы данных на другой сервер БД

Для просмотра, запуска, остановки служб MS SQL Server необходимо запустить утилиту SQL Server Configuration Manager.



Для того чтобы скопировать БД необходимо остановить службу SQL Server (в ее контекстном меню выбрать Stop). Далее в подпапке ...\\MSSQL.1\\MSSQL\\Data\\ скопировать файлы с вашим названием БД (по умолчанию их два). Не забудьте потом снова запустить службу SQL Server (в ее контекстном меню выбрать Start).

Для того чтобы подключить скопированную БД на другом сервере, нужно предварительно скопировать ваши файлы в папку ...\\MSSQL.1\\MSSQL\\Data\\ соответствующего сервера. Далее запустить утилиту SQL Server Management Studio. В появившемся окне с названием Object Explorer Проводник объектов (его можно вызвать по <F8>) выбираем DataBases (Базы данных) и по <правой кнопке мыши> в контекстном меню выбираем Attach... (Присоединить...).



В появившемся окне Attach DataBases (Присоединение базы данных) нажать и выбрать ваш файл БД с расширением .mdf.



## ПРАКТИЧЕСКАЯ РАБОТА № 52

### Тема: Восстановление БД из резервной копии

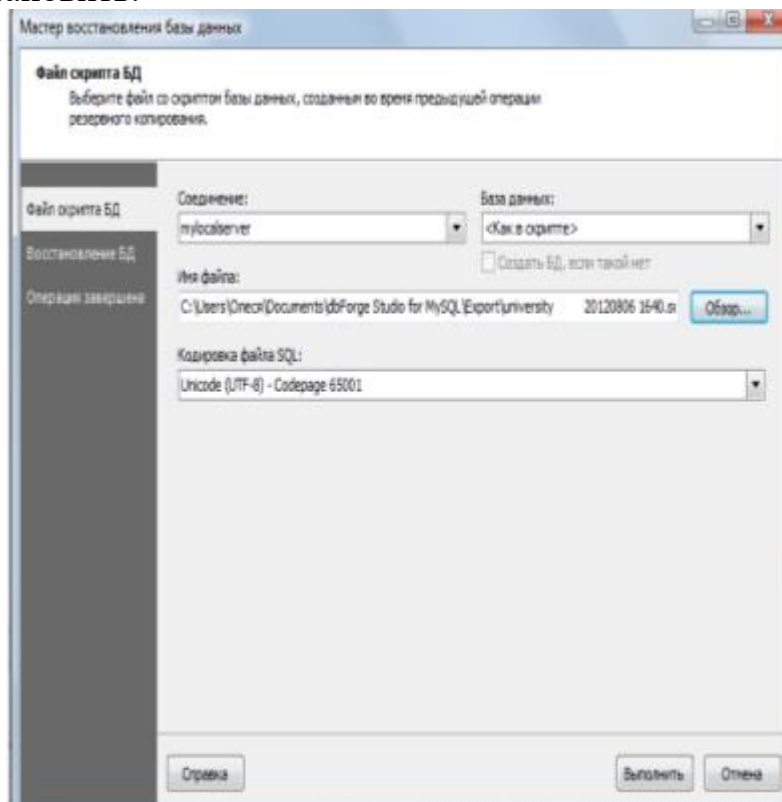
**Цель работы:** освоить технологию восстановления БД из резервной копии.

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, инструкции по выполнению работы.

#### Содержание работы:

##### Задание 1. Восстановить базу данных Университет

Для восстановления базы данных из резервной копии используется команда "База данных/ Восстановление базы данных. В результате откроется диалоговое окно "Мастер восстановления баз данных", в котором надо выбрать имя БД куда будет восстанавливаться база данных, в которую будет помещен результат, способ восстановления, файл, из которого будет восстанавливаться база данных, отмечаем выбранную резервную копию, и нажать кнопку Восстановить.



Запускаем процесс восстановления. В случае успешного выполнения получим сообщение.

**Задание 2.** Удалите свои базы данных «Проектная организация», «Учебный центр», «Торговая фирма», «Компания по разработке программных продуктов». Выполнить восстановление удаленных баз данных из резервных копий. Проверьте результат.

## **ПРАКТИЧЕСКАЯ РАБОТА № 53**

### **Тема: Восстановление БД из резервной копии**

**Цель работы:** освоить технологию восстановления БД из резервной копии.

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, инструкции по выполнению работы.

#### **Содержание работы:**

**Задание 1.** Удалите свои базы данных, созданные по индивидуальным заданиям. Выполнить восстановление удаленных баз данных из резервных копий. Проверьте результат.

## ПРАКТИЧЕСКАЯ РАБОТА № 54.

**Тема: Реализация доступа пользователей к БД. Назначение/отмена привилегий пользователя для доступа к объектам БД**

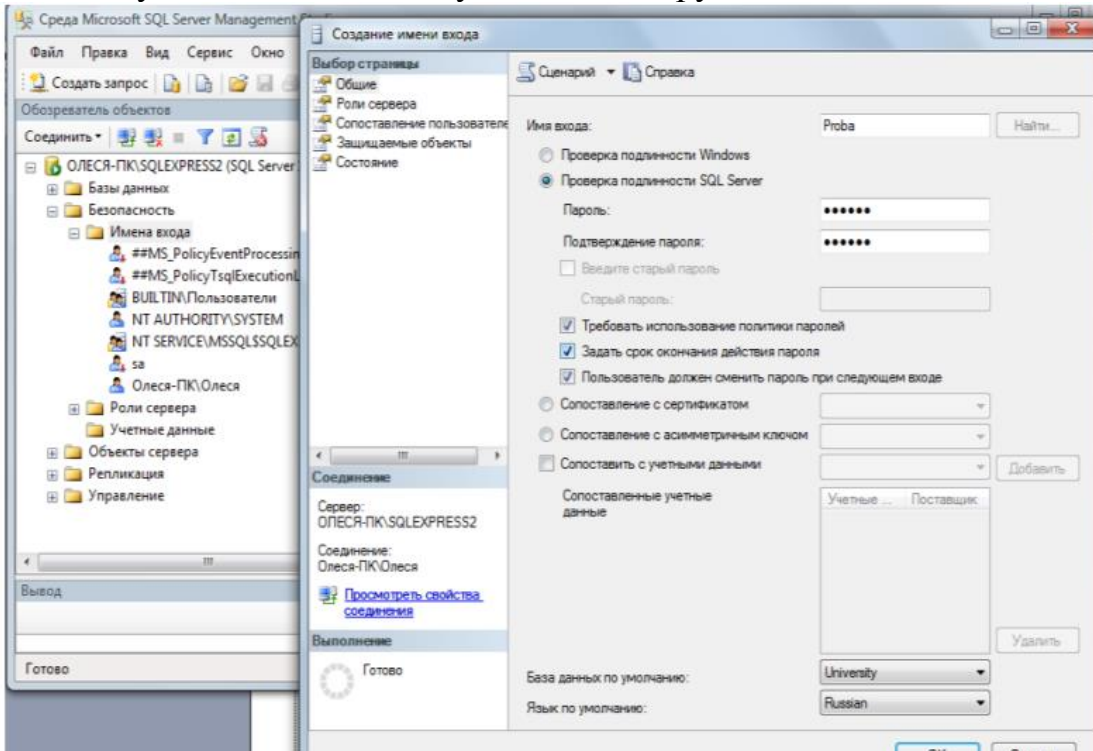
**Цель работы:** научить применять способы защиты информации в БД на примере СУБД MS SQL Server Management Studio.

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, инструкции по выполнению работы.

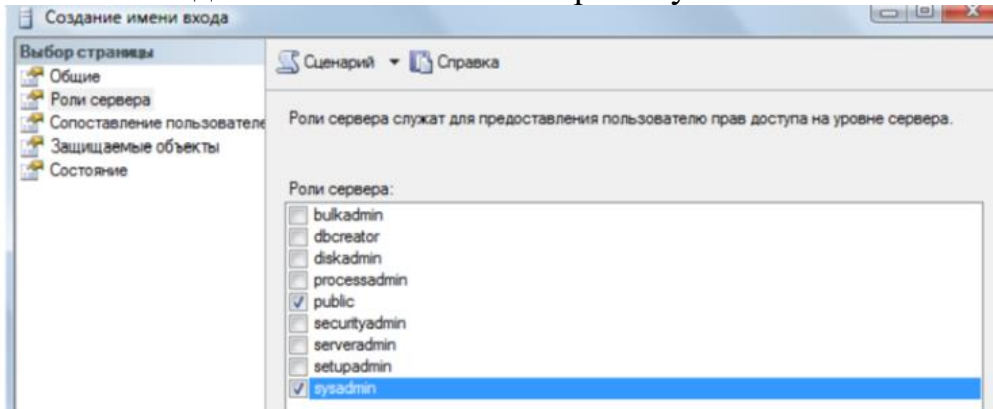
**Содержание работы:**

**Задание 1.** Создадим новую учетную запись для нашей базы данных University.

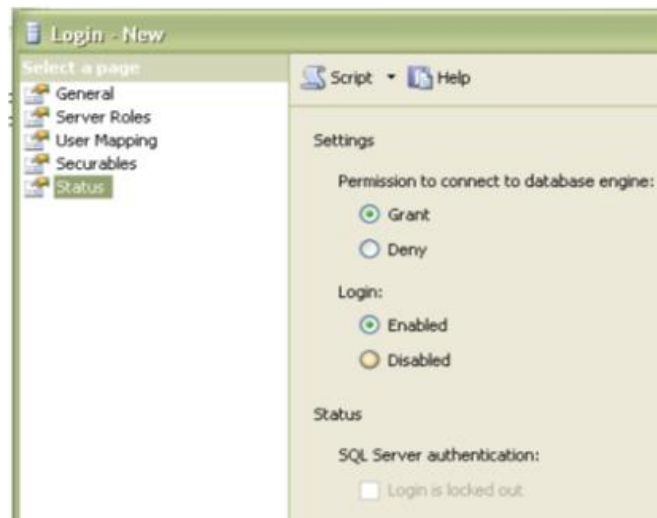
1. Для этого выберите в Обозревателе объектов раздел Безопасность/Имена входа. Добавьте новое имя входа – Proba, установите опцию Проверка подлинности SQL Server, присвойте свой пароль, примените к выбранной базе данных, установите язык по умолчанию – русский.



2. Прежде чем добавлять нового пользователя просмотрите его назначенные серверные роли. Для этого в этом же окне выберите раздел Роли сервера. Установите для пользователя Proba роль sysadmin.







После нажатия на в БД появится пользователь Proba с правами собственника БД, который может выполнять все манипуляции с БД University. Откройте в окне обозревателя объектов БД University и перейдите на вкладку Безопасность, там вы найдете только что созданного пользователя.


## Задание 2. Создание ролей программно

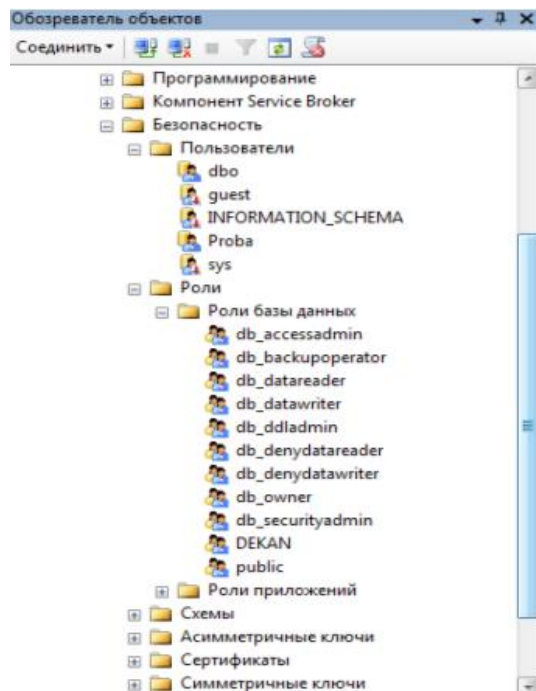
Для упрощения управления правами доступа в системе создаются роли, которые затем можно назначать группе пользователей. Создадим для нашего примера роли декана (DEKAN) и студента (STUDENT).

Пример создания роли декана:

USE University --сделать текущей БД University

EXEC sp\_addrole 'DEKAN'

Эти операторы набрать на странице, вызванной нажатием кнопки Создать запрос. Для запуска команд на выполнение нажать  **Выполнить**. Сохраните запрос.



Повторный запуск тех же команд сгенерирует ошибки типа «В БД уже существует роль DEKAN».

Чтобы просмотреть, что роль добавлена, откройте вкладку Безопасность/Роли/Роли базы данных.

Пример создания роли студента:

```
USE University --сделать текущей БД university
```

```
EXEC sp_addrole 'STUDENT'
```

Декан должен обладать правами на чтение, удаление, изменение, добавление во все таблицы БД University, а также должен иметь возможность запускать на исполнение процедуры и функции БД University. Поэтому роли декана из системных привилегий назначаем EXECUTE, а из привилегий доступа к объектам назначаем DELETE, INSERT, UPDATE, SELECT.

Студент должен обладать правами на чтение из таблиц. Поэтому роли читателя из привилегий доступа к объектам назначаем SELECT.

*Оператор представления привилегий*

Синтаксис:

```
GRANT <привилегия>, ...
```

```
ON <объект >, ...
```

```
TO <имя>
```

```
[WITH grant option];
```

Атрибут WITH GRANT OPTION дает право пользователю самому раздавать права, которые он получил.

С помощью оператора GRANT для каждого пользователя формируется список привилегий, привилегии управляют работой сервера данных с точки зрения защиты данных. Выполнению каждой транзакции предшествует проверка привилегий пользователя, сеанс которого породил транзакцию.

Например (не выполнять):

```
GRANT select, update (Sales, num) ON Sales_data TO user1
```

```
WITH GRANT OPTION
```

Пользователь, предоставивший привилегию другому, называется грантор (grantor — поставитель). Привилегия является предоставляемой, если право на нее можно предоставить другим пользователям.

PUBLIC — имя роли, которую получает пользователь при добавлении в список пользователей конкретной БД, включает в себя минимальный набор прав на чтение данных из таблиц и представлений в БД.

Для примера создадим таблицу Discuplinu. Выполните следующий sql-запрос:

```
USE University --сделать текущей БД university
```

```
create table Discuplinu (
```

```
Kod_Discuplinu int NOT NULL primary key, name_Discuplinu nchar(30)
NULL, kol_chasov int NULL );
```

Выполните код и обновите вкладку Таблицы. Вы должны увидеть созданную таблицу для сохранения данных о всех дисциплинах. Эта таблица пока пустая с тремя столбцами Kod\_Discuplinu, name\_Discuplinu, kol\_chasov.

Роль декана названа DEKAN. Операторы назначения прав доступа для этой роли представлены ниже:

```
GRANT DELETE, INSERT, UPDATE, SELECT ON Discuplinu TO DEKAN  
GRANT EXECUTE TO DEKAN
```

Роль студента названа STUDENT. Операторы назначения прав доступа для этой роли представлены ниже:

```
GRANT SELECT ON Discuplinu TO STUDENT
```

Примените роли декана и студента к созданной таблице.

*Создание пользователей с определенной ролью*

Пример создания декана Ivanov\_Dek и присвоения ему роли:

```
EXEC sp_addlogin 'Ivanov_Dek','Ivanov', 'University'  
use University
```

```
EXEC sp_adduser 'Ivanov_Dek','Ivanov_Dek'
```

```
EXEC sp_addrolemember 'DEKAN', 'Ivanov_Dek'
```

Пример создания студента Petrov\_Stud и присвоения роли:

```
EXEC sp_addlogin 'Petrov_Stud','Petrov', 'University'  
use University
```

```
EXEC sp_adduser 'Petrov_Stud','Petrov_Stud' E
```

```
EXEC sp_addrolemember 'STUDENT', 'Petrov_Stud'
```

Выполните команды. Перейдите в окне Обозреватель объектов на Роли/Роли базы данных/Dekan и просмотрите его свойства. Просмотрите назначенные общие свойства, защищаемые объекты и расширенные свойства.

*Оператор отмены привилегий*

Синтаксис отмены привилегий:

```
REVOKE [with grant option]
```

```
< привилегии >,...
```

```
ON < объект >,...
```

```
FROM <имя_пользователя>;
```

Предложение with grant option сохраняет за пользователем перечисленные привилегии, но отменяет его право передавать их кому-либо другому.

Пример:

```
REVOKE SELECT ON Discuplinu FROM STUDENT
```

Выполните команду.

*Оператор изымания роли у пользователя*

```
Revoke <список ролей> from <список пользователей>.
```

Пример:

```
use University
```

```
EXEC sp_droprolemember 'STUDENT', 'Petrov_Stud'
```

Выполните команду и просмотрите результат.

**Задание 3.** Для созданных БД «Торговая фирма», «Учебный центр», «Проектная организация», «Компания по разработке программных продуктов» выполнить следующие действия:



Определить 2-3 должностных лица, которые смогут работать с таблицами БД. Для каждого должностного лица определить набор привилегий, которыми он может пользоваться.

В утилите SQL Server Management Studio создать под каждое должностное лицо соответствующую роль, наделить эту роль определенными привилегиями. Далее создать по одному пользователю на каждую должность и присвоить им соответствующие роли.

Сохранить последовательно SQL-операторы с указанием заданий в файле.

## **ПРАКТИЧЕСКАЯ РАБОТА № 55.**

**Тема: Реализация доступа пользователей к БД. Назначение/отмена привилегий пользователя для доступа к объектам БД**

**Цель работы:** научить применять способы защиты информации в БД на примере СУБД MS SQL Server Management Studio.

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, инструкции по выполнению работы.

### **Справочный материал:**

Система безопасности БД должна обеспечивать физическую целостность БД и защиту от несанкционированного вторжения с целью чтения содержимого и изменения данных.

Защита БД производится на двух уровнях:

- на уровне пароля;
- на уровне пользователя (защита учетных записей пользователей и идентифицированных объектов).

Компонентами системы безопасности SQL Server на уровне сервера являются: система аутентификации средствами Windows и средствами SQL Server, учетные записи пользователей и встроенные роли сервера. На уровне базы данных компонентами системы безопасности являются: идентификация пользователей баз данных, фиксированные и пользовательские роли баз данных, а также роли приложений.

Фиксированными ролями сервера являются:

Sysadmin – для выполнения любых действий в сервере;

Sereradmin– для конфигурирования и выключения сервера;

Setupadmin– для управления связанными серверами и процедурами, автоматически запускающимися при старте сервера;

Securityadmin– для управления учетными записями и правами на создание базы данных, а также для контроля журнала ошибок;

Processadmin - для управления процессами, запущенными на сервере

Dbcreator – для создания и модификации баз данных;

Diskadmin– для управления файлами сервера;

Bulcadmin– для массивного копирования баз данных

При создании базы данных сервер автоматически создает для нее фиксированные роли, которые, как и фиксированные роли сервера, нельзя удалить или модифицировать:

Db\_owner – для выполнения любых действий в базе данных;

Db\_accessadmin – для добавления и удаления пользователей;

Db\_securityadmin – для управления всеми разрешениями, объектами, ролями и именами ролей;

Db\_ddladmin – для выполнения любых команд DDL, кроме GRANT, DENY и REVOKE;

Db\_backupoperator– для выполнения команд DBCC, CHECK, POINT и BACKUP;

Db\_datareader – для контроля данных во всех таблицах базы данных и и чтения;

Db\_datawriter – для модификации данных в любых таблицах базы данных;  
Db\_denydatareader – для запрета просмотра данных в любой таблице базы данных;

Db\_denydatawriter – для запрета модификации данных во всех таблицах базы данных.

Кроме этих фиксированных ролей любой базы данных есть еще одна роль public, членами которой автоматически становятся все пользователи, имеющие тот или иной доступ к базе данных. Эта роль имеет специальное назначение и обеспечивает минимальные права доступа к базе данных тем пользователям, для которых их права не определены явно. Эта роль имеется во всех базах данных, включая системные master, tempdb, msdb и model и не может быть удалена.

Если в базе данных разрешен пользователь quest, то установленный для public доступ будут иметь все пользователи, получившие доступ к SQL Server.

В отличие от сервера базы данных могут иметь пользовательские роли и роли приложения, которые создает администратор с помощью Enterprise Manager или Transact\_SQL индивидуально для групп пользователей и групп приложений, наделяя их необходимыми правами доступа к конкретной базе данных.

В любую роль базы данных можно включать:

- а) пользователей сервера;
- б) роли сервера;
- в) пользователей Windows;
- г) группы пользователей Windows.

Средствами Enterprise Manager можно включать только пользователей сервера.

### **Содержание работы:**

**Задание 1.** Создать учетную запись SQL сервера, используя графическую утилиту Enterprise Manager, выполнив следующие действия:

1. Выбрать нужный сервер.
2. Открыть папку Security этого сервера.
3. Выбрать объект Logins, щелкнув по соответствующему значку.
4. В правом окне просмотреть список учетных записей данного сервера:

Name – имя учетной записи сервера;

Type – происхождение учетной записи:

User W– пользователь Windows;

Group W– группа пользователей Windows;

Standard – пользователь SQL сервера;

Server Access – доступ к серверу SQL:

Permit – разрешен;

Deny – запрещен;

Default Database – база данных по умолчанию, к которой подключен пользователь(обязательный параметр)

User – имя пользователя базы данных;

Default Language – язык по умолчанию для данной учетной записи.

5. Для создания новой учетной записи сервера открыть контекстное меню объекта Logins, щелкнув по нему правой клавишей мыши или по значку на панели инструментов левой клавишей мыши.

клавишей мыши.

6. В появившемся диалоговом окне на вкладке General (общие) ввести имя учетной записи в поле Name.

7. Выбрать тип аутентификации: Windows Authentication или SQL Server Authentication.

8. Если выбрана аутентификация Windows, то задать в поле Domain имя домена, в котором учтен пользователь или группа Windows. Имя заданного домена добавиться впереди имени пользователя также и в поле Name (для выбора домена использовать кнопку”...”).

9. В группе Security Access (безопасный доступ) установить переключатель Grant Access (доступ разрешен). Установка переключателя Deny Access навсегда запретит регистрацию пользователя или домена Windows.

10. Если выбрана аутентификация SQL Server, то задать только пароль для учетной записи.

11. Задав параметры аутентификации Windows или SQL Server, указать в группе Defaults (умолчания) имя базы данных в поле Database, к которой пользователь будет подключаться автоматически, и язык Language (Russian). Если имя базы данных не задать, то сервер будет автоматически подключать к базе master.

12. Включить создаваемую учетную запись в требуемую встроенную роль сервера:

Sysadmin, Serveradmin, Setupadmin, Securityadmin, Processadmin, Dbcreator, Diskadmin, Bulkadmin, установив флажок против этой роли на вкладке Server Role.

13. На вкладке Database Access выбрать требуемую базу данных, установив флажок слева от ее имени, и задать имя пользователя, в которое будет отображаться рассматриваемая учетная запись, а в нижней части вкладки с помощью флажка включить

пользователя в ту или иную роль в зависимости от его работы с базой данных.

14. Щелкнув по кнопке Properties (свойства) и просмотреть список пользователей, включенных в выбранную роль рассматриваемой базы данных.

15. Щелкнув по кнопке Permissions (права) и просмотреть список прав, предоставленных выбранной роли базы данных.

16. Закрывать все окна.

17. Вновь открыть список учетных записей сервера, дважды щелкнуть по вновь созданной записи и проверить правильность введенных параметров.

18. Закрывать все окна.

19. Приступить к работе с базами данных, используя новую учетную запись.

**Задание 2.** Включить учетную запись пользователя или группы пользователей Windows в фиксированную роль сервера SQL с помощью Enterprise Manager, выполнив следующие действия:

1. Выбрать требуемый сервер в левом окне Tree.
2. Открыть объекты сервера, щелкнув по его кнопке “+”.
3. Открыть объекты Security этого сервера, щелкнув по кнопке “+” для Security.
4. Выбрать объект Logins (записи) и щелкнуть по нему, при этом в правом окне откроется список учетных записей сервера.
5. Дважды щелкнуть по требуемой учетной записи сервера.
6. В открывшемся окне на Server Login Properties открыть вкладку Server role.
7. Установить флажки возле тех ролей сервера, в которые требуется включить конфигурируемую запись.
8. Закрыть все открытые окна, щелкая по кнопкам “ОК” этих окон.
9. Повторить задания, используя следующий набор команд:
  - а) Security/Server Rolees;
  - б) Щелкнуть левой клавишей;
  - в) В правом окне выбрать требуемую фиксированную роль;
  - г) Два раза щелкнуть по выбранной роли;
  - д) В открывшемся окне Server Role Properties щелкнуть по кнопке Add вкладки General;
  - е) Добавить учетные записи в заданную роль;
  - ж) Закрыть окно со списком учетных записей;
  - з) На вкладке Permission окна Server Role Properties просмотреть предоставляемые права для рассматриваемой роли.
10. Закрыть все открытые диалоговые окна, щелкая по кнопкам ОК.
11. Проверить правильность выполненных действий.

**Задание 3.** Создать нового пользователя базы данных для учетной записи Windows с помощью Enterprise Manager, выполнив следующие действия:

1. Выбрать требуемый сервер и требуемую базу данных в левом окне Tree.
2. Открыть объекты выбранной базы данных, щелкнув по значку “+” этой базы.
3. Выбрать в раскрывшемся списке объектов рассматриваемой базы данных объект Users (пользователи).
4. Щелкнуть правой клавишей мыши и открыть контекстное меню объекта Users (пользователи).
5. В контекстном меню исполнить команду New Database User (новый пользователь базы данных).
6. В открывшемся диалоговом окне ввести:
  - а) в поле Login Name – имя учетной записи пользователя или группы пользователей Windows;
  - б) в поле User Name – имя нового пользователя рассматриваемой базы данных.
7. Включить нового пользователя в необходимые роли базы данных: public, db – owner, db – denydatareader и т.д. Для этого требуемые роли надо отметить флажками в списке фиксированных ролей базы данных, расположенном в правой части окна.

8. Щелкнуть по кнопке Properties и, просмотрев список всех пользователей базы данных, убедиться, что новый пользователь включен этот список.
9. Щелкнуть по кнопке Permission и задать права доступа пользователя к объектам базы данных: SELECT, INSERT, UPDATE, DELETE, EXEC, DRI. В окне находится полный список объектов базы данных.
10. Щелкнуть по кнопке Columns (столбцы) для выбранной базы данных и задать права доступа к конкретным столбцам таблицы: SELECT и/или UPDATE.
11. Закрыть все открытые диалоговые окна, щелкая по кнопкам ОК.
12. Проверить работу нового пользователя с рассматриваемой базой данных и его права.

**Задание 4.** Создать учетную запись SQL сервера, используя мастер Create Login Wizard, выполнив следующие действия:

1. Выполнить команду в Enterprise Manager Run an Wizard/Create Login Wizard.
2. В открывшемся окне мастера ознакомиться с этапами создания учетной записи сервера:
  - а) Select an authentication mode – выбрать режим аутентификации;
  - б) Grant access to security roles – представить доступ секретным ролям;
  - в) Grant access to databases – предоставить доступ к базам данных.
3. Щелкнуть по кнопке Next.
4. Выбрать режим аутентификации: Windows или SQL Server.
5. Щелкнуть по кнопке Next.
6. Если выбран режим аутентификации Windows, то в открывшемся окне в поле Windows account задать имя учетной записи или группы Windows и домена и указать тип доступа: Grant access to the server (предоставить доступ к серверу) или Deny access to the server (запретить доступ к серверу).
7. Если выбран режим аутентификации SQL Server, то помимо имени учетной записи, задаваемой в поле Login I указать пароль в поле Password (пароль) и в поле Confirm Password (подтвердить пароль). Этот пароль пользователь будет использовать при подключении к SQL серверу.
8. Щелкнуть по кнопке Next в том или в другом случае.
9. Включить учетную запись в требуемые фиксированные роли сервера, устанавливая против роли флажок.
10. Щелкнуть по кнопке Next.
11. Разрешить для учетной записи доступ к базам данных, отмечая их флажком.
12. Щелкнуть по кнопке Next.
13. Мастер автоматически создаст имена пользователей баз данных.
14. Проверить сделанные установки.

## ПРАКТИЧЕСКАЯ РАБОТА № 56.

### Тема: Поиск требуемой информации в БД с использованием операторов объединения таблиц

**Цель работы:** изучить явные и неявные соединения таблиц, внутреннее и внешнее соединение таблиц, научиться применять полученные знания на практике

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, инструкции по выполнению работы.

#### Справочный материал:

Данные часто хранятся в несколько связанных таблицах. Для выбора данных используются разные методы соединения таблиц. Когда выбор данных осуществляется из нескольких таблиц, в конструкции SELECT для каждого поля указывается таблица в виде <таблица>.<поле>. Если название поля уникальное, то можно его указать без таблицы, иначе это обязательно, чтобы избежать коллизий. Чтобы не повторить длинные названия таблиц, можно использовать псевдоним для таблиц. Псевдоним указывается в конструкции как FROM.

При *неявном соединении таблиц* формат конструкций FROM и WHERE имеет следующий вид:

FROM <таблица1> [псевдоним1], <таблица2> [псевдоним2]...

[WHERE <условие\_соединения> [AND <условие\_поиска>]... ]

Таким образом, если более одной таблицы присутствует в конструкции FROM, то их разделяют запятой. Если условие соединения не указывать, тогда результат будет декартовым произведением, то есть для каждой строки одной из таблиц берутся все возможные сочетания строк из других таблиц.

Для *явного соединения таблиц* используется команда JOIN. У явного соединения есть следующие разновидности:

– Внутреннее соединение – осуществляется с помощью команды INNER JOIN. Из двух таблиц берутся только связанные строки. INNER JOIN имеет следующий формат записи:

<таблица1> INNER JOIN <таблица2> ON <таблица1>.<связующее\_поле> = <таб-лица2>.<связующее\_поле>.

При внутреннем соединении слово INNER можно пропустить.

– Внешнее соединение – осуществляется с помощью команды OUTER JOIN. OUTER JOIN имеет следующий формат записи:

<таблица1> LEFT | RIGHT | FULL OUTER JOIN <таблица2> ON <таблица1>.<связующее\_поле> = <таблица2>.<связующее\_поле>.

У внешнего соединения есть три разновидности:

– Левое внешнее соединение LEFT OUTER JOIN – из таблицы, название которой является левым операндом команды JOIN, выбираются все строки, из второй таблицы – только те записи, которые имеют связь с первой таблицей.

– Правое внешнее соединение RIGHT OUTER JOIN – из таблицы, название которой является правым операндом команды JOIN, выбираются все строки, из первой таблицы только те записи, которые имеют связь со второй таблицей.



– Полное внешнее соединение FULL OUTER JOIN – из обеих таблиц выбираются все строки.

При использовании внешних соединений, слово OUTER можно пропустить.

– Перекрестное соединение CROSS JOIN – декартово произведение двух таблиц. CROSS JOIN имеет следующий формат записи:

<таблица1> CROSS JOIN <таблица2>.

При выборе данных из трех и более таблиц, с помощью явного соединения, результат зависит от порядка соединения.

Используя соединение, можно связать таблицу с собой. При таком соединении псевдоним обязателен. Набор данных, полученных из нескольких таблиц, не отличается от набора, полученного из одной таблицы. Ему тоже можно применить группировки и т.д.

### **Содержание работы:**

**Задание 1.** Даны следующие таблицы:

Таблица 1. Факультет

Аббревиатура	Название
Ен	Естественные науки
Гн	Гуманитарные науки
Ит	Информационные технологии
Фм	Физико-математический

Таблица 2. Кафедра

Шифр	Название	Факультет
вм	Высшая математика	ен
ис	Информационные системы	ит
мм	Математическое моделирование	фм
оф	Общая физика	ен
пи	Прикладная информатика	ит
эф	Экспериментальная физика	фм

Таблица 3. Специальность

Номер	Направление	Шифр
01.03.04	Прикладная математика	мм
09.03.02	Информационные системы и технологии	ис
09.03.03	Прикладная информатика	пи
14.03.02	Ядерные физика и технологии	эф
38.03.05	Бизнес-информатика	ис

Таблица 4. Сотрудник

Таб_номер	Шифр	Фамилия	Должность	Зарплата	Шеф
101	пи	Прохоров П.П.	зав. кафедрой	35 000,00 р.	101
102	пи	Семенов С.С.	преподаватель	25 000,00 р.	101
105	пи	Петров П.П.	преподаватель	25 000,00 р.	101
153	пи	Сидорова С.С.	инженер	15 000,00 р.	102
201	ис	Андреев А.А.	зав. кафедрой	35 000,00 р.	201
202	ис	Борисов Б.Б.	преподаватель	25 000,00 р.	201
241	ис	Глухов Г.Г.	инженер	20 000,00 р.	201
242	ис	Чернов Ч.Ч.	инженер	15 000,00 р.	202
301	мм	Басов Б.Б.	зав. кафедрой	35 000,00 р.	301
302	мм	Сергеева С.С.	преподаватель	25 000,00 р.	301
401	оф	Волков В.В.	зав. кафедрой	35 000,00 р.	401
402	оф	Зайцев З.З.	преподаватель	25 000,00 р.	401
403	оф	Смирнов С.С.	преподаватель	15 000,00 р.	401
435	оф	Лисин Л.Л.	инженер	20 000,00 р.	402
501	вм	Кузнецов К.К.	зав. кафедрой	35 000,00 р.	501
502	вм	Романцев Р.Р.	преподаватель	25 000,00 р.	501
503	вм	Соловьев С.С.	преподаватель	25 000,00 р.	501
601	эф	Зверев З.З.	зав. кафедрой	35 000,00 р.	601
602	эф	Сорокина С.С.	преподаватель	25 000,00 р.	601
614	эф	Григорьев Г.Г.	инженер	20 000,00 р.	602

Таблица 5. Дисциплина

Код	Объем	Название	Исполнитель
101	320	Математика	вм
102	160	Информатика	пи
103	160	Физика	оф
202	120	Базы данных	ис
204	160	Электроника	эф
205	80	Программирование	пи
209	80	Моделирование	мм

Таблица 6. Заявка

Номер		01.03.04			09.03.02					09.03.03					14.03.02				38.03.05				
Код		101	205	209	101	102	103	202	205	209	101	102	103	202	205	101	102	103	204	101	103	202	209

Таблица 7. Зав\_кафедрой

Таб_номер	101	201	301	401	501	601
Стаж	15	18	20	10	18	8

Таблица 8. Инженер

<b>Таб_номер</b>	153	241	242	435	614
<b>Специальность</b>	электроник	электроник	программист	электроник	программист

Таблица 9. Преподаватель

<b>Таб_номер</b>	<b>Звание</b>	<b>Степень</b>
101	профессор	д. т.н.
102	доцент	к. ф.-м. н.
105	доцент	к. т.н.
201	профессор	д. ф.-м. н.
202	доцент	к. ф.-м. н.
301	профессор	д. т.н.
302	доцент	к. т.н.
401	профессор	д. т.н.
402	доцент	к. т.н.
403	ассистент	–
501	профессор	д. ф.-м. н.
502	профессор	д. ф.-м. н.
503	доцент	к. ф.-м. н.
601	профессор	д. ф.-м. н.

Таблица 10. Студент

<b>Рег_номер</b>	<b>Номер</b>	<b>Фамилия</b>
10101	09.03.03	Николаева Н. Н.
10102	09.03.03	Иванов И. И.
10103	09.03.03	Крюков К. К.
20101	09.03.02	Андреев А. А.
20102	09.03.02	Федоров Ф. Ф.
30101	14.03.02	Бондаренко Б. Б.
30102	14.03.02	Цветков К. К.
30103	14.03.02	Петров П. П.
50101	01.03.04	Сергеев С. С.
50102	01.03.04	Кудрявцев К. К.
80101	38.03.05	Макаров М. М.
80102	38.03.05	Яковлев Я. Я.

Таблица 11. Экзамен

Дата	Код	Рег_номер	Таб_номер	Аудитория	Оценка
05.06.2015	102	10101	102	т505	4
05.06.2015	102	10102	102	т505	4
05.06.2015	202	20101	202	т506	4
05.06.2015	202	20102	202	т506	3
07.06.2015	102	30101	105	ф419	3
07.06.2015	102	30102	101	т506	4
07.06.2015	102	80101	102	м425	5
09.06.2015	205	80102	402	м424	4
09.06.2015	209	20101	302	ф333	3
10.06.2015	101	10101	501	т506	4
10.06.2015	101	10102	501	т506	4
10.06.2015	204	30102	601	ф349	5
10.06.2015	209	80101	301	э105	5
10.06.2015	209	80102	301	э105	4
12.06.2015	101	80101	502	с324	4
15.06.2015	101	30101	503	ф417	4
15.06.2015	101	50101	501	ф201	5
15.06.2015	101	50102	501	ф201	3
15.06.2015	103	10101	403	ф414	4
17.06.2015	102	10101	102	т505	5

1. Выбрать факультет и кафедры, используя неявное соединение. Результат отсортировать по алфавиту:

```
SELECT Ф.Название AS Факультет, К.Название AS Кафедра
FROM Факультет Ф, Кафедра К
WHERE Ф.Аббревиатура = К.Факультет
ORDER BY Факультет, Кафедра
```

2. Выбрать факультет и кафедры, используя явное соединение. Результат отсортировать по алфавиту:

```
SELECT Ф.Название AS Факультет, К.Название AS Кафедра
FROM Факультет Ф
INNER JOIN Кафедра К ON Ф.Аббревиатура = К.Факультет
ORDER BY Факультет, Кафедра
```

3. Выбрать все факультеты и их кафедры, если существуют. Результат отсортировать по алфавиту:

```
SELECT Ф.Название AS Факультет, К.Название AS Кафедра
FROM Факультет Ф
LEFT OUTER JOIN Кафедра К ON Ф.Аббревиатура = К.Факультет
ORDER BY Факультет, Кафедра
```

4. Вывести из таблиц «Кафедра», «Специальность» и «Студент» данные о студентах:

```
SELECT С.Фамилия, П.Направление, К.Название AS Кафедра
```

FROM Студент С

INNER JOIN Специальность П ON С.Номер = П.Номер

INNER JOIN Кафедра К ON П.Шифр = К.Шифр

5. Вывести для каждого сотрудника фамилию, должность, зарплату и фамилию его непосредственного руководителя:

SELECT С.Фамилия, С.Должность, С.Зарплата, П.Фамилия AS Руководитель

FROM Сотрудник С

INNER JOIN Сотрудник П ON С.Шеф = П.Таб\_номер

6. Вывести список студентов, сдавших хотя бы один экзамен. По правилам соединения, студенты, не сдававшие экзамены, в выборке представлены не будут:

SELECT С.Фамилия

FROM Студент С

INNER JOIN Экзамен Э ON С.Рег\_номер = Э.Рег\_номер

GROUP BY С.Фамилия

7. Вывести из таблиц «Студент» и «Экзамен» учетные номера и фамилии студентов, а также количество сданных экзаменов и средний балл для каждого студента:

SELECT С.Фамилия, COUNT(Э.Оценка) AS [Количество экзаменов],  
AVG(Э.Оценка) AS [Средний балл]

FROM Студент С

INNER JOIN Экзамен Э ON С.Рег\_номер = Э.Рег\_номер

GROUP BY С.Фамилия

8. Вывести список заведующих кафедрами и их зарплаты, и стаж работы:

SELECT С.Фамилия, С.Зарплата, З.Стаж

FROM Сотрудник С

INNER JOIN Зав\_кафедрой З ON С.Таб\_номер = З.Таб\_номер

9. Вывести список кандидатов и докторов физико-математических наук:

SELECT С.Фамилия, П.Степень

FROM Сотрудник С

INNER JOIN Преподаватель П ON С.Таб\_номер = П.Таб\_номер

WHERE П.Степень IN ('к.ф.-м.н.', 'д.ф.-м.н.')

10. Вывести название дисциплины, фамилию, должность и степень преподавателя, дату и место проведения экзаменов в хронологическом порядке:

SELECT DISTINCT Д.Название AS Дисциплина, С.Фамилия, С.Должность,

П.Степень, Э.Дата, Э.Аудитория

FROM Экзамен Э

INNER JOIN Дисциплина Д ON Э.Код = Д.Код

INNER JOIN Сотрудник С ON Э.Таб\_номер = С.Таб\_номер

INNER JOIN Преподаватель П ON Э.Таб\_номер = П.Таб\_номер

ORDER BY Э.Дата

11. Вывести фамилию преподавателей и количество их экзаменов:

SELECT С.Фамилия, COUNT(Э.Дата) AS [Количество экзаменов]

FROM Экзамен Э

INNER JOIN Сотрудник С ON Э.Таб\_номер = С.Таб\_номер

GROUP BY С.Фамилия

12. Вывести список студентов, не сдавших ни одного экзамена:

SELECT С.Фамилия

FROM Студент С

LEFT OUTER JOIN Экзамен Э ON С.Рег\_номер = Э.Рег\_номер

WHERE Э.Рег\_номер IS NULL

### **Задания для самостоятельного выполнения:**

1. Вывести из таблиц «Кафедра», «Специальность» и «Студент» данные о студентах, которые обучаются на данном факультете (например, «ит»).
2. Вывести из таблиц «Кафедра», «Специальность» и «Сотрудник» данные о выпускающих кафедрах (факультет, шифр, название, фамилию заведующего). Выпускающей считается та кафедра, на которую есть ссылки в таблице «Специальность».
3. Вывести в запросе для каждого сотрудника номер и фамилию его непосредственного руководителя. Для заведующих кафедрами поле руководителя оставить пустым.
4. Вывести список студентов, сдавших минимум два экзамена.
5. Вывести список инженеров с зарплатой, меньшей 20000 руб.
6. Вывести список студентов, сдавших экзамены в заданной аудитории.
7. Вывести из таблиц «Студент» и «Экзамен» учетные номера и фамилии студентов, а также количество сданных экзаменов и средний балл для каждого студента только для тех студентов, у которых средний балл не меньше заданного (например, 4).
8. Вывести список заведующих кафедрами и их зарплаты, и степень.
9. Вывести список профессоров.
10. Вывести название дисциплины, фамилию, должность и степень преподавателя, дату и место проведения экзаменов в хронологическом порядке в заданном интервале даты.
11. Вывести фамилию преподавателей, принявших более трех экзаменов.
12. Вывести список студентов, не сдавших ни одного экзамена в указанной дате.

## **ПРАКТИЧЕСКАЯ РАБОТА № 57.**

**Тема:** Поиск требуемой информации в БД с использованием операторов объединения таблиц

**Цель работы:** изучить явные и неявные соединения таблиц, внутреннее и внешнее соединение таблиц, научиться применять полученные знания на практике

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, инструкции по выполнению работы.

**Содержание работы:**

**Задание 1.** По базе данных Университет произвести поиск данных с использованием операторов объединения таблиц. За основу брать примеры из Практической работы № 56.

**Задание 1.** По базе данных Проектная организация произвести поиск данных с использованием операторов объединения таблиц. За основу брать примеры из Практической работы № 56.

## **ПРАКТИЧЕСКАЯ РАБОТА № 58.**

**Тема:** Поиск требуемой информации в БД с использованием операторов объединения таблиц

**Цель работы:** изучить явные и неявные соединения таблиц, внутреннее и внешнее соединение таблиц, научиться применять полученные знания на практике

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, инструкции по выполнению работы.

**Содержание работы:**

**Задание 1.** По базе данных Торговая фирма произвести поиск данных с использованием операторов объединения таблиц. За основу брать примеры из Практической работы № 56.

**Задание 2.** По базе данных Компания по разработке программных продуктов произвести поиск данных с использованием операторов объединения таблиц. За основу брать примеры из Практической работы № 56.



## ПРАКТИЧЕСКАЯ РАБОТА № 59.

**Тема:** Поиск требуемой информации в БД с использованием операторов лево/правостороннего объединения таблиц и хранимых процедур

**Цель работы:** научиться применять хранимые процедуры для поиска информации в базе данных.

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, инструкции по выполнению работы.

### **Справочный материал:**

При программировании в SQL Server введенный код сначала компилируется, потом за-пускается. Процесс компиляции может занимать определенное время. На языке Transact-SQL также есть возможность написанный блок кода сохранить и заранее скомпилировать. Особенно, если код многократно используется в операции базы данных, отличным решением будет произвести его инкапсуляцию в процедуры. Для этой цели используются хранимые процедуры, которые представляют собой набор инструкций, выполняющихся как единое целое. Процедуры аналогичны конструкциям в других языках программирования и выполняют следующие задачи:

- обрабатывают входные параметры и возвращают значения в виде выходных параметров;
- содержат инструкции, которые выполняют операции в базе данных, в отличие от пользовательских функций;
- возвращают сведения об успешном или неуспешном завершении.

В клиент-серверной и распределенных системах хранимые процедуры позволяют существенно сократить сетевой трафик, поскольку по сети отправляется только вызов на выполнение процедуры.

С точки зрения безопасности, хранимые процедуры выполняют очень большую роль, так как устраняют необходимость предоставлять разрешения на уровне объектов и упрощают формирование уровней безопасности. С помощью хранимых процедур можно предотвратить атаки типа «инъекция SQL».

Хранимая процедура создается с помощью команды CREATE PROCEDURE или CREATE PROC, которая имеет следующий упрощенный вид:

```
CREATE {PROC | PROCEDURE} <название>  
[(<@параметр> <тип> [= <значение по умолчанию>] [OUT | OUTPUT])]  
AS [BEGIN]  
<команды>  
[END]
```

При создании процедуры после команды CREATE указывается тип создаваемого объекта с помощью ключевого слова PROCEDURE или его сокращенного варианта PROC.

Названия процедур должны соответствовать требованиям, предъявляемым к идентификаторам, и должны быть уникальными в базе данных. При этом не следует пользоваться префиксом «sp\_». Этим префиксом в SQL Server обозначаются системные процедуры.

В хранимую процедуру можно передать до 2100 параметров. При выполнении процедуры значение каждого из объявленных параметров должно

быть указано пользователем, если для параметра не определено значение по умолчанию.

Ключевое слово OUT (можно использовать и OUTPUT) показывает, что параметр процедуры является выходным.

Для выполнения хранимой процедуры используется ключевое слово EXECUTE (или EXEC). Процедуру также можно вызывать и выполнять без ключевого слова, если она является первой инструкцией. Синтаксис команды EXECUTE имеет следующий вид:

EXECUTE [<@статус возврата>=] <название процедуры> [<@параметр>=] <значение>| <@переменная> [OUTPUT] | [DEFAULT]

В отличие от вызова функций, при вызове хранимых процедур с указанием названия параметра ([<@параметр>=] <значение>), последовательность параметров можно не соблюдать.

Для выходных параметров при вызове указывается ключевое слово OUTPUT. Если для параметра указано значение по умолчанию, можно его использовать с помощью ключевого слова DEFAULT.

Для удаления хранимых процедур используется команда DROP PROCEDURE. Упрощенный синтаксис имеет следующий вид:  
DROP PROC | PROCEDURE [IF EXISTS] <название хранимой процедуры>

Ключевые слова IF EXISTS удаляют хранимую процедуру только в том случае, если она уже существует.

### Содержание работы:

#### Задание 1. Дана таблица Страны.

Название	Столица	Площадь	Население	Континент
Австрия	Вена	83858	8741753	Европа
Азербайджан	Баку	86600	9705600	Азия
Албания	Тирана	28748	2866026	Европа
Алжир	Алжир	2381740	39813722	Африка
Ангола	Луанда	1246700	25831000	Африка
Аргентина	Буэнос-Айрес	2766890	43847000	Южная Америка
Афганистан	Кабул	647500	29822848	Азия
Бангладеш	Дакка	144000	160221000	Азия
Бахрейн	Манама	701	1397000	Азия
Белиз	Бельмопан	22966	377968	Северная Америка
Белоруссия	Минск	207595	9498400	Европа
Бельгия	Брюссель	30528	11250585	Европа
Бенин	Порто-Ново	112620	11167000	Африка
Болгария	София	110910	7153784	Европа
Боливия	Сукре	1098580	10985059	Южная Америка
Ботсвана	Габороне	600370	2209208	Африка
Бразилия	Бразилиа	8511965	206081432	Южная Америка
Буркина-Фасо	Уагадугу	274200	19034397	Африка
Бутан	Тхимпху	47000	784000	Азия

Великобритания	Лондон	244820	65341183	Европа
Венгрия	Будапешт	93030	9830485	Европа
Венесуэла	Каракас	912050	31028637	Южная Америка
Восточный Тимор	Дили	14874	1167242	Азия
Вьетнам	Ханой	329560	91713300	Азия

1. Напишите хранимую процедуру для вывода информации о сервере, о базе данных и о текущем пользователе, и вызовите ее:

```
CREATE PROC Пример1
```

```
AS
```

```
BEGIN
```

```
SELECT @@Servername AS Сервер, @@Version AS [Версия СУБД], Db_Name()
```

```
AS [База данных], User AS [Пользователь базы данных], System_User
```

```
AS [Системный пользователь]
```

```
END
```

```
EXECUTE Пример1
```

2. Напишите хранимую процедуру, которая выводит названия и столицы всех стран:

```
CREATE PROC Пример2
```

```
AS
```

```
BEGIN
```

```
SELECT Название, Столица
```

```
FROM Страны
```

```
END
```

3. Напишите хранимую процедуру, которая выводит список стран заданной части света, и вызовите ее:

```
CREATE PROC Пример3
```

```
@Конт AS VARCHAR(50)
```

```
AS
```

```
BEGIN
```

```
SELECT Название, Столица, Площадь, Население
```

```
FROM Страны
```

```
WHERE Континент = @Конт
```

```
END
```

```
EXECUTE Пример3 'Азия'
```

4. Напишите хранимую процедуру, которая выводит список стран, площадь которых находится в заданном интервале, и вызовите ее:

```
CREATE PROC Пример4
```

```
@A AS FLOAT,
```

```
@B AS FLOAT
```

```
AS
```

```
BEGIN
```

```
SELECT Название, Столица, Площадь, Население, Континент
```

```
FROM Страны
```

```
WHERE Площадь BETWEEN @A AND @B
```

END

EXECUTE Пример4 1000, 10000

5. Напишите хранимую процедуру, которая возвращает количество стран, содержащих в названии заданную букву, и вызовите ее:

CREATE PROC Пример5

@Буква AS CHAR(1),

@Количество AS INT OUTPUT

AS

BEGIN

SELECT @Количество = COUNT(\*)

FROM Страны

WHERE CHARINDEX(@Буква, Название) > 0

END

DECLARE @K AS INT

DECLARE @Б AS CHAR(1)

SET @Б = 'у'

EXECUTE Пример5 @Б, @K OUTPUT

SELECT @K AS [Количество стран]

6. Напишите хранимую процедуру для вывода трех стран с наименьшей площадью в заданной части света, и вызовите ее. Если часть света не указана, выбрать Европу:

CREATE PROC Пример6

@Конт AS VARCHAR(50) = 'Европа'

AS

BEGIN

SELECT TOP 3 Название, Столица, Площадь, Население, Континент

FROM Страны

WHERE Континент = @Конт

ORDER BY Площадь

END

EXECUTE Пример6 DEFAULT

7. Напишите хранимую процедуру, которая создает таблицу «Страны\_У», и заполняет ее странами, названия которых начинаются на букву «У»:

CREATE PROC Пример7

AS

BEGIN

SELECT Название, Столица, Площадь, Население, Континент

INTO Страны\_У

FROM Страны

WHERE LEFT(Название, 1) = 'У'

END

EXECUTE Пример7

8. Напишите хранимую процедуру, которая удаляет таблицу «Страны\_У» и возвращает количество строк:

CREATE PROC Пример8

```

AS
BEGIN
DECLARE @K AS INT
SELECT @K = COUNT(*)
FROM Страны_У
DROP TABLE Страны_У
RETURN @K
END
DECLARE @C AS INT
EXECUTE @C = Пример8
SELECT @C AS [Количество строк в удаленной таблице]
9. Напишите код, который удаляет хранимую процедуру «Пример8»:
DROP PROC Пример8

```

### **Задания для самостоятельного выполнения:**

1. Напишите хранимую процедуру для вывода информации о сервере, о базе данных, о текущем пользователе, о текущем времени, и вызовите ее.
2. Напишите хранимую процедуру, которая выводит данные всех стран.
3. Напишите хранимую процедуру, которая выводит список стран, кроме заданной части света, и вызовите ее.
4. Напишите хранимую процедуру, которая выводит список стран, население которых находится в заданном интервале, и вызовите ее.
5. Напишите хранимую процедуру, которая возвращает количество стран, у которых в названии отсутствует заданная буква, и вызовите ее.
6. Напишите хранимую процедуру для вывода пяти стран с наибольшим населением в заданной части света, и вызовите ее. Если часть света не указана, выбрать Африку.
7. Напишите хранимую процедуру, которая создает таблицу «Страны\_<первая буква вашей фамилии>», и заполняет ее странами, названия которых начинаются с первой буквой вашей фамилии.
8. Напишите хранимую процедуру, которая удаляет таблицу, которую вы создали в предыдущем задании и возвращает количество удаленных строк.
9. Напишите хранимую процедуру, принимающую число и возвращающую количество цифр в нем через параметр OUTPUT.
10. Напишите хранимую процедуру AddRightDigit, добавляющую к целому положи-тельному числу K справа цифру D (D – входной параметр целого типа, лежащий в диапазоне [0..9], K – параметр целого типа, являющийся одновременно входным и выходным).
11. Напишите хранимую процедуру InvDigit, меняющую порядок следования цифр целого положительного числа K на обратный (K – параметр целого типа, являющийся одновременно входным и выходным).
12. Напишите хранимую процедуру Swar, меняющую содержимое переменных X и Y (X и Y – вещественные параметры, являющиеся одновременно входными и выходными).

13. Напишите хранимую процедуру SortInc, меняющую содержимое переменных A, B, C, таким образом, чтобы их значения оказались упорядоченными по возрастанию (A, B, C – вещественные параметры, являющиеся одновременно входными и выходными).
14. Напишите хранимую процедуру DigitCountSum, находящую количество C цифр целого положительного числа K, а также их сумму S (K – входной, C, S – выходные параметры целого типа).
15. Напишите код, который удаляет все хранимые процедуры, вами созданные.

## **ПРАКТИЧЕСКАЯ РАБОТА № 60.**

**Тема:** Поиск требуемой информации в БД с использованием операторов лево/правостороннего объединения таблиц и хранимых процедур

**Цель работы:** научиться применять хранимые процедуры для поиска информации в базе данных.

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, инструкции по выполнению работы.

**Содержание работы:**

**Задание 1.** По базе данных Торговая фирма произвести поиск данных с использованием хранимых процедур. За основу брать примеры из Практической работы № 59.

**Задание 2.** По базе данных Проектная организация произвести поиск данных с использованием хранимых процедур. За основу брать примеры из Практической работы № 59.

## **ПРАКТИЧЕСКАЯ РАБОТА № 61.**

**Тема:** Поиск требуемой информации в БД с использованием операторов лево/правостороннего объединения таблиц и хранимых процедур

**Цель работы:** научиться применять хранимые процедуры для поиска информации в базе данных.

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, инструкции по выполнению работы.

**Содержание работы:**

**Задание 1.** По базе данных Университет произвести поиск данных с использованием хранимых процедур. За основу брать примеры из Практической работы № 59.

**Задание 2.** По базе данных Компания по разработке программных продуктов произвести поиск данных с использованием хранимых процедур. За основу брать примеры из Практической работы № 59.



## ПРАКТИЧЕСКАЯ РАБОТА № 62.

**Тема: Мониторинг безопасности работы с базами данных**

**Цель работы:** изучить технология мониторинга безопасности работы с БД.

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, инструкции по выполнению работы.

### **Справочный материал:**

Мониторинг СУБД и баз данных проводится для поддержания работоспособности и производительности СУБД, а также с целью отслеживания аварийных ситуаций и сбора статистики.

Реализуется мониторинг с помощью отдельных утилит СУБД, представляющих собой программные продукты, входящие в состав СУБД, но загружаемые отдельно от ядра СУБД, либо в виде набора прикладных интерфейсов — API (Application Program Interface). Эту утилиту или группу утилит и в операционной системе, и в СУБД часто называют монитором или системным монитором.

Для осуществления мониторинга ядро СУБД собирает информацию от приложений, работающих с базой данных, и от системных средств самой СУБД. Эта информация может использоваться администратором баз данных для следующих целей:

- обеспечение необходимого объема аппаратных ресурсов (на основе информации об их использовании);
- анализ производительности отдельных приложений или SQL-запросов;
- отслеживание интенсивности использования отношений;
- оценка эффективности используемых методов доступа;
- настройка параметров ядра СУБД в целях повышения производительности;
- оценка последствий вносимых оптимизационных изменений.

### *Сбор статистики*

Администратор системы должен следить за тем, чтобы приложения, работающие с БД, имели средства сбора или предоставления статистики. Например, каждое приложение должно учитывать общее время работы, системное время, процессорное время (total time, system time, process time).

Цель сбора статистики — настроить производительность и параметры, выяснить активность пользователей и затраты по каждому из запросов и операций.

Сбор статистики может начинаться вместе с запуском ядра СУБД или с началом сессии данного приложения. Необходимо с помощью утилит мониторинга собирать статистику по БД в целом, а именно:

- статистику открытий БД (open на базу, как говорят программисты);
- число операций ввода-вывода и время;
- статистику закрытий БД (close на базу);
- число установленных соединений в течение работы сеанса ядра СУБД;
- число взаимоблокировок при блокировании записей БД (deadlock);
- число транзакций в единицу времени;
- статистику по кодам возврата от операций с БД.

Особо отметим, что АБД должен требовать от прикладных программистов обработки кода возврата от любой операции с БД. При некоторых, определенных для каждой СУБД, кодах возврата возникают фатальные события, требующие немедленного реагирования администратора БД.

Необходимо также собирать статистику по отдельным запросам приложений, работающих с СУБД, таким как:

- стоимость процессора (сколько команд процессора, тратится на запрос);
- стоимость ввода-вывода (сколько команд ввода-вывода тратится на запрос);
- число предикатов, используемых в запросе;
- избирательность, т. е. вероятность того, что каждая найденная строка удовлетворяет предикату; обычно избирательность должна составлять около 10%);
- число занятых при запросе страниц в буферном пуле СУБД.

Еще один вид статистики, который надо собирать — это статистика по отдельным отношениям БД и по соответствующим индексным файлам. Например, какой объем памяти занят под индексы, под области переполнения, непосредственно под отношение, под рабочую область СУБД

Система безопасности SQL Server основана на концепции защищаемых объектов (securables), т.е. объектов, на которые можно назначать разрешения, и принципалов (principles), т.е. объектов, которым можно назначать разрешения. Принципами могут быть логины на уровне сервера, пользователи и роли на уровне базы данных. Роли назначаются пользователям.

Разрешения на доступ к объектам могут предоставляться как непосредственно пользователям, так и через роли. Каждый объект имеет своего владельца, и права собственности также влияют на разрешения.

SQL Server использует двухэтапную схему аутентификации. На уровне сервера пользователь распознается по своему идентификатору (LoginID), который может быть либо именем входа SQL Server, либо группой или учетной записью Windows. После входа на сервер пользователь получает те права, которые были назначены ему администратором на уровне сервера, в частности с помощью фиксированных серверных ролей. Если пользователь принадлежит роли sysadmin, то он имеет полный доступ ко всем функциям сервера, а также ко всем базам данных и объектам на нем.

Для получения доступа к базе данных логин пользователя должен быть сопоставлен с соответствующим ему идентификатором пользователя (UserID), который специфичен для каждой базы данных. Вполне возможна ситуация, когда пользователь был распознан в SQL Server, но у него нет доступа ни к одной из баз данных. Также возможно и обратное: пользователю открыт доступ к базам данных, но он не был распознан сервером. Перемещение базы данных и ее разрешений на другой сервер без параллельного перемещения имен входа сервера может привести к возникновению таких "осиротевших" пользователей.

На уровне базы данных пользователю может быть предоставлен определенный набор разрешений с помощью назначения ему фиксированных ролей базы данных. Все пользователи автоматически становятся членами

стандартной роли public, у которой по умолчанию нет никаких разрешений. Пользовательские роли - это дополнительные роли, служащие в качестве групп. Роли может быть разрешен доступ к объектам базы данных, а пользователю могут быть назначены роли.

Разрешения к объектам назначаются с помощью инструкций GRANT (предоставить), REVOKE (отозвать) и DENY (запретить). Запрет привилегии замещает собой ее предоставление, а предоставление привилегии замещает собой ее отзыв. Пользователю может быть предоставлено множество разрешений к объекту (индивидуальных, наследованных от роли, обеспеченных принадлежностью к роли public). Если какая-либо из этих привилегий запрещена, для пользователя блокируется доступ к объекту. В противном случае, если какая-либо из привилегий предоставляет разрешение, пользователь получает доступ к объекту.

Разрешения объекта достаточно детализированы. Существуют отдельные разрешения для каждого из возможных действий (SELECT, INSERT, UPDATE, RUN и т.д.) над объектом.

Выбор типа логина и настройка режима аутентификации SQL Server поддерживает два типа логинов (имен входа): логин Windows (логин для локальной учетной записи Windows, логин для доменной учетной записи Windows, логин для группы Windows); логин SQL Server.

При использовании логинов Windows в системные таблицы базы данных master записывается информация об идентификаторе учетной записи или группы Windows (но не пароль).

Аутентификация (т. е. проверка имени пользователя и пароля) производится обычными средствами Windows при входе пользователя на свой компьютер.

При использовании логина SQL Server пароль для этого логина (точнее, его хэшированное значение) хранится вместе с идентификатором логина в базе данных master. При подключении пользователя к серверу ему придется указать имя логина и пароль.

Предпочтительный вариант логина для пользователя - это логин Windows, при этом не для учетной записи, а для группы (лучше всего для локальной доменной группы). Преимуществ у такого решения множество: пользователю достаточно помнить один пароль - для входа на свой компьютер; повышается уровень защищенности SQL Server. Это происходит, по крайней мере, за счет того, что пароль не будет передаваться по сети открытым текстом, как это происходит по умолчанию при использовании команд CREATE LOGIN и ALTER LOGIN. Кроме того, хэши Windows более защищены, чем хэши логинов SQL Server; проверка при входе пользователя производится быстрее.

### **Содержание работы:**

#### **Задание 1:** Использование программы Windows SystemMonitor

1 Создайте базу данных NorthwindCopy. Для этого восстановите резервную копию базы данных Университет. Сконфигурируйте Windows System Monitor. Для этого выполните следующие действия:

- запустите программу «Системный монитор» (Мой компьютер /Администрирование /Системный монитор);
- на панели инструментов щелкните по кнопке «Добавить» (+);
- в окне диалога добавьте счетчики, используя информацию таблицу, приведенную ниже.
- в конце закройте окно кнопкой «Заккрыть».

Объект	Счетчик	Выбрать вхождение из списка
SQL Server:Access Methods (Методы доступа)	FullScans/sec(Полных сканирований в сек)	
SQL Server:Access Methods	IndexSearch/sec(Индексных поисков в сек)	
SQL Server:Buffer Manager (Диспетчер буферов)	Buffer Cashe Hit Ratio (коэффициент кэшированных операций)	
SQL Server:Databases	Active Transactions (Активные транзакции)	NorthwindCopy
SQL Server:Databases	PercentLogUsed(% использования журнала транзакций)	NorthwindCopy
SQL Server:Databases	Transactions/sec (Транзакций/сек)	NorthwindCopy
SQL Server:Memory Manager (Диспетчер памяти)	LockBlocks(Препятствующие блокировки)	
SQL Server:SQL Statistics (Статистика SQL Server )	BatchRequests/sec(Пакетных запросов в секунду)	

2. Проведите имитацию деятельности сервера. Деятельность сервера будет имитировать программа C:\МОС\2072a\Labfiles\L08\Monitor.bat, которую следует вызвать из командной строки (Пуск/Выполнить).
3. Наблюдайте окно «Просмотр диаграммы» во время выполнения командных файлов. Запишите значения счетчиков. Опишите в отчете, какие тенденции Вы отметили.
4. Отслеживание использования памяти и процессора. В окне системного монитора щелкните по кнопке «Новый набор счетчиков» и добавьте набор счетчиков в соответствии с таблицей

Объект	Счетчик	Выбрать вхождение из списка
Память	Обмен страниц/сек	

Память	Ошибок страниц/сек	
Процесс	% загрузки процессора	Sqlserver
Процесс	Ошибок страниц/сек	Sqlserver
SQL Server:Cache Manager (Диспетчер КЕШ-памяти)	CacheHitRatio(Коэффициент успешного обращения к КЕШ памяти)	Adhoc SQL Plans
SQL Server:Memory Manager	Connection Memory(KB)	
SQL Server:Memory Manager	Total Server Memory (KB)	

5. Наблюдайте за окном «Просмотр диаграммы» во время выполнения программы Monitor.bat. Какие тенденции вы наблюдаете? Кнопкой на панели инструментов перейдите в режим отображения значений счетчиков. Скопируйте это окно в отчет, сравните значения счетчиков с допустимыми, сделайте выводы.

6. Закройте все окна командной строки на панели задач. Счетчики программы Системный монитор должны отразить снижение активности на сервере.

7. Создайте журнал для записи показаний системного монитора через каждые 20 сек в течение 5-10 мин. Остановите запись и просмотрите журнал.

**Задание 2.** Осуществить мониторинг безопасности созданной БД «Компания по разработке программных продуктов».

## **ПРАКТИЧЕСКАЯ РАБОТА № 63.**

**Тема:** Мониторинг безопасности работы с базами данных

**Цель работы:** изучить технология мониторинга безопасности работы с БД.

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, инструкции по выполнению работы.

**Содержание работы:**

**Задание 1:** Осуществить мониторинг безопасности созданных БД «Учебный центр», «Проектная организация», «Торговая фирма».

### **Контрольные вопросы:**

- 1.Какова цель мониторинга безопасности БД?
- 2.С помощью чего осуществляется мониторинг?
- 3.Для чего производится сбор статистики?

## ПРАКТИЧЕСКАЯ РАБОТА № 64.

**Тема:** Резервное копирование БД, журнализация транзакций пользователя

**Цель работы:** изучите процессы резервного копирования баз данных и ведения журнала транзакций в MS SQL Server для обеспечения надежности данных, а также научитесь выполнять резервное копирование и восстанавливать базу данных при необходимости.

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, инструкции по выполнению работы.

### **Справочный материал:**

#### 1. Резервное копирование базы данных

*Резервное копирование* — это создание копии базы данных, которую можно использовать для восстановления данных в случае их потери, повреждения или сбоя системы. В MS SQL Server существуют различные типы резервных копий: полные, дифференциальные и транзакционные.

*Полное резервное копирование* — сохраняет всю базу данных.

*Дифференциальное копирование* — сохраняет только изменения, внесённые после последнего полного резервного копирования.

*Транзакционное копирование (журнал транзакций)* — сохраняет все транзакции, выполненные после последнего полного или дифференциального резервного копирования.

#### 2. Журнализация транзакций

*Журнал транзакций* — это последовательность записей о всех транзакциях, выполненных в базе данных. Он обеспечивает возможность восстановления базы данных до конкретного момента времени и поддерживает целостность данных.

В MS SQL Server журнал транзакций используется для обеспечения транзакционной надежности и восстановления данных.

Восстановление с помощью журналов позволяет вернуть базу данных к состоянию на любой момент времени.

#### 3. Восстановление базы данных

Восстановление возможно из резервных копий и журналов транзакций. В MS SQL Server используется команда RESTORE для восстановления базы данных.

### **Содержание работы:**

**Задание 1.** Создать базу данных и выполнить полное резервное копирование.

1. В MS SQL Server Management Studio (SSMS) создать новую базу данных, например, AptekaDB.

2. Выполнить резервное копирование базы данных в файл .bak.

-- Создание базы данных

CREATE DATABASE AptekaDB;

GO

-- Полное резервное копирование базы данных

BACKUP DATABASE AptekaDB

TO DISK = 'C:\Backup\ AptekaDB\_Full.bak'

WITH FORMAT, NAME = 'Full Backup of AptekaDB ';

**Задание 2.** Выполнить транзакции и сделать транзакционное резервное копирование

1. В базе данных выполнить несколько операций вставки, обновления и удаления данных.

2. Создать транзакцию, зафиксировать её.

3. Выполнить резервное копирование журнала транзакций.

-- Вставка данных

BEGIN TRANSACTION;

INSERT INTO Employees (Name, Position)

VALUES ('Иван', 'Менеджер');

COMMIT TRANSACTION;

-- Бэкап журнала транзакций

BACKUP LOG AptekaDB

TO DISK = 'C:\Backup\ AptekaDB\_Log.trn'

WITH NOFORMAT, INIT, NAME = 'Log Backup of AptekaDB';

**Задание 3.** Восстановить базу данных из резервных копий

1. Восстановить базу данных из полного бэкапа.

2. Восстановить её до конкретного момента времени, используя журнал транзакций.

-- Восстановление из полного бэкапа

RESTORE DATABASE AptekaDB

FROM DISK = 'C:\Backup\AptekaDB\_Full.bak'

WITH NORECOVERY;

-- Восстановление до определенного момента времени

RESTORE LOG AptekaDB

FROM DISK = 'C:\Backup\ AptekaDB \_Log.trn'

WITH STOPAT = '2023-10-10 15:30:00', RECOVERY;



## ПРАКТИЧЕСКАЯ РАБОТА № 65.

**Тема:** Резервное копирование БД, журнализация транзакций пользователя

**Цель работы:** изучите процессы резервного копирования баз данных и ведения журнала транзакций в MS SQL Server для обеспечения надежности данных, а также научитесь выполнять резервное копирование и восстанавливать базу данных при необходимости.

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, инструкции по выполнению работы.

**Содержание работы:**

**Задание 1.** Создать базу данных и выполнить полное резервное копирование.

1. В MS SQL Server Management Studio (SSMS) создать новую базу данных, например, AiroportDB.

2. Выполнить резервное копирование базы данных в файл .bak.

-- Создание базы данных

```
CREATE DATABASE AiroportDB;
```

```
GO
```

-- Полное резервное копирование базы данных

```
BACKUP DATABASE AiroportDB
```

```
TO DISK = 'C:\Backup\ AiroportDB _Full.bak'
```

```
WITH FORMAT, NAME = 'Full Backup of AiroportDB';
```

**Задание 2.** Выполнить транзакции и сделать транзакционное резервное копирование

1. В базе данных выполнить несколько операций вставки, обновления и удаления данных.

2. Создать транзакцию, зафиксировать её.

3. Выполнить резервное копирование журнала транзакций.

-- Вставка данных

```
BEGIN TRANSACTION;
```

```
INSERT INTO Employees (Name, Position)
```

```
VALUES ('Иван', 'Менеджер');
```

```
COMMIT TRANSACTION;
```

-- Бэкап журнала транзакций

```
BACKUP LOG AiroportDB
```

```
TO DISK = 'C:\Backup\ AiroportDB _Log.trn'
```

```
WITH NOFORMAT, INIT, NAME = 'Log Backup of AiroportDB ';
```

**Задание 3.** Восстановить базу данных из резервных копий

1. Восстановить базу данных из полного бэкапа.

2. Восстановить её до конкретного момента времени, используя журнал транзакций.

-- Восстановление из полного бэкапа

```
RESTORE DATABASE AiroportDB
```

```
FROM DISK = 'C:\Backup\AiroportDB_Full.bak'
```

```
WITH NORECOVERY;  
-- Восстановление до определенного момента времени  
RESTORE LOG AiroporDB  
FROM DISK = 'C:\Backup\ AiroporDB_Log.trn'  
WITH STOPAT = '2023-10-10 15:30:00', RECOVERY;
```

## **ПРАКТИЧЕСКАЯ РАБОТА № 66.**

### **Тема: Установка приоритетов**

**Цель работы:** изучить способы установки приоритетов

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server, инструкции по выполнению работы.

#### **Справочный материал:**

Рассмотрим настройки параметра конфигурации сервера priority boost в SQL Server с помощью среды SQL Server Management Studio или Transact-SQL. С помощью параметра priority boost задается, должен ли Microsoft SQL Server выполняться с большим приоритетом в Microsoft Windows по сравнению с остальными процессами на том же компьютере. Если установить этот параметр в значение 1, SQL Server выполняется в планировщике Windows или Windows Server R2 с базовым приоритетом 13. Значением по умолчанию является 0, что соответствует базовому значению приоритета 7.

В будущей версии Microsoft SQL Server этот компонент будет удален. Избегайте использования этого компонента в новых разработках и запланируйте изменение существующих приложений, в которых он применяется.

#### **Настройка параметра повышения приоритета с помощью:**

##### *Ограничения*

Задание слишком высокого приоритета может лишить ресурсов операционную систему и сетевые функции, что может вызвать проблемы при завершении работы SQL Server и выполнении на сервере других административных задач.

##### *Безопасность*

Разрешения на выполнение хранимой процедуры sp\_configure без параметров или только с первым параметром по умолчанию предоставляются всем пользователям. Для выполнения процедуры sp\_configure с обоими параметрами для изменения параметра конфигурации или запуска инструкции RECONFIGURE необходимо иметь разрешение ALTER SETTINGS на уровне сервера. Разрешение ALTER SETTINGS неявным образом предоставлено предопределенным ролям сервера sysadmin и serveradmin.

#### **Содержание работы:**

##### **Задание 1.** Настроить параметр повышения приоритета

1. В обозревателе объектов щелкните правой кнопкой мыши сервер и выберите пункт Свойства. Щелкните узел Процессоры.
2. В разделе Поток установите флажок Повысить приоритет SQL Server. Остановите и снова запустите SQL Server.

##### **Задание 2.** Настройка параметра повышения приоритета

1. Установите соединение с компонентом Компонент Database Engine. На панели «Стандартная» нажмите Создать запрос.
2. Скопируйте следующий пример в окно запроса и нажмите кнопку Выполнить. В этом примере описывается использование процедуры sp\_configure для задания значения параметра priority boost равным 1

```
SQL Копировать
USE AdventureWorks2012 ;
GO
EXEC sp_configure 'show advanced options', 1;
GO
RECONFIGURE ;
GO
EXEC sp_configure 'priority boost', 1 ;
GO
RECONFIGURE;
GO
```

После настройки параметра priority boost, чтобы изменения вступили в силу, необходимо перезапустить сервер.

## **ПРАКТИЧЕСКАЯ РАБОТА № 67.**

### **Тема: Установка приоритетов**

**Цель работы:** изучить механизмы определения и настройки приоритетов выполнения транзакций и ресурсов в MS SQL Server для оптимизации работы базы данных и обеспечения согласованности данных при конкурирующих операциях.

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server, инструкции по выполнению работы.

#### **Справочный материал:**

##### **1. Управление конкурентностью и приоритетами в MS SQL Server**

MS SQL Server обеспечивает многопользовательский доступ к базе данных, использующий механизмы блокировок и транзакционных уровней изоляции для предотвращения конфликтов и обеспечения согласованности данных.

##### **2. Механизмы установки приоритетов**

**Уровень изоляции транзакций:** Определяет степень видимости изменений других транзакций и влияет на приоритет выполнения. Основные уровни: Read Uncommitted, Read Committed, Repeatable Read, Serializable.

**Обработка блокировок:** SQL Server использует блокировки различных типов (Shared, Update, Exclusive). В некоторых случаях можно управлять приоритетом блокировок с помощью команд, таких как SET LOCK\_TIMEOUT.

**RESOURCE POOL и WORKLOAD CLASSIFIER** (для SQL Server Enterprise): Позволяют задавать приоритеты для ресурсов, выделяемых различным группам транзакций или подключений.

**LOCK\_TIMEOUT:** Устанавливает время ожидания для получения блокировки. Можно использовать для задания приоритета, чтобы транзакции не блокировались бесконечно.

##### **3. Приоритет выполнения транзакций**

В MS SQL Server можно управлять приоритетом выполнения транзакций посредством:

**Установка уровня изоляции.** Установка времени ожидания блокировки (LOCK\_TIMEOUT). Использование RESOURCE GOVERNOR (в Enterprise-версии) для распределения ресурсов по приоритетам.

#### **Содержание работы:**

**Задание 1.** Настроить уровень изоляции транзакций для двух транзакций с разным приоритетом

1. Создать две транзакции, которые пытаются одновременно обновить одну и ту же таблицу.

2. В первой транзакции установить уровень изоляции Read Uncommitted (низкий приоритет), во второй — Serializable (высокий приоритет).

3. Наблюдать порядок выполнения и блокировки.

-- Транзакция 1: низкий уровень изоляции

```
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;
```

```
BEGIN TRANSACTION;
```

```
UPDATE TestTable SET Value = Value + 1 WHERE ID = 1;
```

```
-- задержка, чтобы транзакция 2 могла начать
WAITFOR DELAY '00:00:10';
COMMIT;
-- Транзакция 2: высокий уровень изоляции
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
BEGIN TRANSACTION;
UPDATE TestTable SET Value = Value + 10
WHERE ID = 1; COMMIT;
```

**Задание 2.** Установить LOCK\_TIMEOUT для управления приоритетами блокировок

1. В одной транзакции выполнить блокировку строки.
2. Во второй транзакции попытаться получить такую же блокировку, но установить LOCK\_TIMEOUT в короткое время.
3. Посмотреть, как транзакция реагирует на таймаут.

```
-- Транзакция 1: захватывает блокировку
BEGIN TRANSACTION;
UPDATE TestTable SET Value = Value + 1 WHERE ID = 2;
-- Транзакция 2: установка таймаута
SET LOCK_TIMEOUT 5000; -- 5 секунд
BEGIN TRANSACTION;
BEGIN TRY
    UPDATE TestTable SET Value = Value + 5 WHERE ID = 2;
    COMMIT;
END TRY
BEGIN CATCH
    PRINT 'Таймаут ожидания блокировки. Транзакция отменена.';
    ROLLBACK; END CATCH;
```

**Задание 3.** Использовать RESOURCE GOVERNOR для задания приоритетных групп

1. В SQL Server Enterprise настроить RESOURCE POOL и WORKLOAD GROUP.
2. Назначить приоритетной группу транзакции с высоким приоритетом.
3. Выполнить несколько транзакций и наблюдать распределение ресурсов.

Эта задача требует административных привилегий и настройки через SSMS или T-SQL командой CREATE RESOURCE POOL, CREATE WORKLOAD GROUP, ALTER RESOURCE GOVERNOR и т.д., что выходит за рамки короткого ответа. Однако концептуально:

```
-- Создание пулов ресурсов и групп нагрузки (пример)
CREATE RESOURCE POOL HighPriorityPool WITH
(MAX_MEMORY_PERCENT = 50);
CREATE WORKLOAD GROUP HighPriorityGroup WITH (importance = HIGH)
USING HighPriorityPool;
-- Назначение текущего подключения к группе
ALTER SESSION SET RESOURCE_GROUP = HighPriorityGroup;
```

## **ПРАКТИЧЕСКАЯ РАБОТА № 68.**

### **Тема: Мониторинг сетевого трафика**

**Цель работы:** изучить инструменты по работе с анализаторами сетевого трафика

**Оборудование:** ПК, интернет, программное обеспечение – MS SQL Server Management Studio, инструкции по выполнению работы.

#### **Справочный материал:**

На начальном уровне перехват и анализ сетевого трафика осуществляется на отдельном хосте. Для этого используются программы «Анализаторы трафика», или «снифферы». Эти программы позволяют осуществить перехват всего трафика по выбранному сетевому интерфейсу и его деинкапсуляцию до прикладного уровня. Как правило, они обладают средствами фильтрации и поиска в перехваченном наборе кадров. Наиболее известным кроссплатформенным решением является Wireshark.

Кроме них существуют стандартные консольные утилиты `arp`, `netstat` (Windows, Linux), `ss`, `lsof` и `tcpdump` (Linux). Как правило, подобные утилиты работают на сетевом уровне и выше. К назначению средств анализа начального уровня относятся анализ текущих соединений на хосте и поиск неисправностей при сетевом взаимодействии.

#### **Содержание работы:**

##### **Задание.**

1. Установите на виртуальном хосте программу Wireshark.
2. Настройте виртуализацию сети в VirtualBox, так чтобы получать трафик приходящий на реальный сетевой адаптер (пропустите этот пункт если Wireshark работает на реальном хосте).
3. Настройте перехват трафика, так чтобы он завершился после сбора 15 Мб (для увеличения интенсивности генерации кадров открыть любой сайт в браузере).
4. Используя инструментарий статистики определите:
  - a. Узел с максимальной активностью (по объему переданных данных),
  - b. Узел осуществивший наибольшее количество широковещательных рассылок,
  - c. Самый активный TCP-порт на хосте (по количеству переданных пакетов)
  - d. Постройте на одной координатной сетке постройте графики интенсивности TCP и UDP трафика (пункт Io Graphs).
  - e. Постройте граф связей только для пакетов, содержащих сообщения протокола HTTP (пункт Flow Graph)
5. Напишите фильтры которые выделяют из общего числа пакеты:
  - a. Относящиеся к работе протоколов HTTP и FTP при работе в качестве клиента операционной системы на которой запущена среда виртуализации (или самого хоста если среда виртуализации не используется).
  - b. Все кадры Ethernet, отправленные с сетевого интерфейса хоста, на котором запущена среда виртуализации (или самого хоста, если среда виртуализации не используется).

- с. Напишите фильтр, отбирающий только широковещательные сообщения. Определите назначение как минимум 3-х широковещательных рассылок разных протоколов.
- d. Определить адреса, на которые поступают данные кадры и пакеты для канального и сетевого уровня
- е. Напишите фильтры для каждой из трех широковещательных рассылок, выбранных в пункте 6-с.
- f. На основании собранной статистики определить, к какому типу коммутационного оборудования подключен используемый компьютер (концентратор, коммутатор или маршрутизатор).
- 6. Запустите одновременно виртуальную машины Linux и Windows. Убедитесь, что на Windows есть ssh клиент putty, а на Linux telnet клиент. Если их нет, то установите клиенты. Программа putty доступна на <http://www.putty.org/>. Telnet клиент на Linux доступен в репозиториях (для CentOS команда `yum install telnet`).
- 7. Настройте между ними внутреннюю сеть и установите на сетевых интерфейсах IP адреса из сети 192.168.0.0/24 (маска 255.255.255.0).
- 8. Запустите на Windows Telnet-сервер (консоль Службы / Services)
- 9. С Windows с помощью терминального клиента Putty подключитесь к SSH серверу на Linux.
- 11. С Linux с помощью telnet клиента подключитесь к Windows машине.
- 12. Используя утилиту netstat или lsof (для Linux) вывести все активнее (прослушиваемые) порты на обеих платформах. Используя утилиту netstat или ss (для Linux) все открытые соединения на обеих платформах.
- 13. С помощью команды tcpdump на Linux настроить вывод на экран содержимого пакетов от Windows-хоста по протоколу telnet.
- 14. Завершите ssh и telnet соединения. На одном из хостов запустите перехват трафика Wireshark и начните ssh и telnet сессии заново.
- 15. С помощью фильтров отберите трафик telnet и ssh. Сравните содержимое сообщений прикладного уровня в обоих случаях.



## **Информационное обеспечение обучения по дисциплине**

### **Основные учебные издания:**

1. Данилова, Л. Ф. Проектирование и разработка баз данных: практикум для СПО / Л. Ф. Данилова, А. Н. Полетайкин. — 2-е изд. — Саратов: Профобразование, 2025. — 172 с. — ISBN 978-5-4488-2589-7. — Текст: электронный // Электронный ресурс цифровой образовательной среды СПО PROОбразование: [сайт]. — URL: <https://profspo.ru/books/152770>
2. Кумскова, И. А., Базы данных: учебник / И. А. Кумскова. — Москва: КноРус, 2026. — 400 с. — ISBN 978-5-406-15045-0. — URL: <https://book.ru/book/958783>

### **Дополнительные учебные издания:**

3. Молдованова, О. В. Информационные системы и базы данных: учебное пособие для СПО / О. В. Молдованова. — 2-е изд. — Саратов: Профобразование, 2024. — 177 с. — ISBN 978-5-4488-1177-7. — Текст: электронный // Электронный ресурс цифровой образовательной среды СПО PROОбразование: [сайт]. — URL: <https://profspo.ru/books/139095>
4. Ткаченко, С. Н., Основы проектирования баз данных: учебник / С. Н. Ткаченко. — Москва: КноРус, 2026. — 176 с. — ISBN 978-5-406-14991-1. — URL: <https://book.ru/book/958706>

### **Интернет-ресурсы:**

#### **Электронно-библиотечная система:**

5. ЭБС «Znanium»
6. ЭБС «PROОбразование»
7. ЭБС «Book.ru»