**СОДЕРЖАНИЕ**

[ЛАБОРАТОРНАЯ РАБОТА №2. Часть 1. УСТАНОВКА СОЕДИНЕНИЯ С СЕРВЕРОМ MICROSOFT SQL SERVER И ПРИНЦИПЫ СОЗДАНИЯ БАЗ ДАННЫХ 3](#_Toc113796586)

[2.1. Цель лабораторной работы 3](#_Toc113796587)

[2.2. Исходные данные 3](#_Toc113796588)

[2.3. Используемые программы 3](#_Toc113796589)

[2.4. Теоретические сведения 3](#_Toc113796590)

[2.5. Задание 6](#_Toc113796591)

[2.6. Ход работы 6](#_Toc113796592)

[2.6.1. Создание соединения с сервером 6](#_Toc113796593)

[2.6.2. Общие сведения о базах данных MS SQL Server 8](#_Toc113796594)

[2.6.3. Создание и регистрация базы данных 8](#_Toc113796595)

[University. 9](#_Toc113796596)

[2.6.4. Подключение к базе данных 13](#_Toc113796597)

[2.6.5. Удаление базы данных 13](#_Toc113796598)

[2.6.6. Резервное копирование и восстановление 14](#_Toc113796599)

[2.7. Копирование и перенос на другой сервер БД 15](#_Toc113796600)

[2.8. Системные базы данных 15](#_Toc113796601)

[2.9. Основные принципы управления учетными записями и ролями в MS SQL Server 16](#_Toc113796602)

[Задание для лабораторной работы №2 (часть 1) 26](#_Toc113796603)

[ЛАБОРАТОРНАЯ РАБОТА №2. Часть 2. РАЗРАБОТКА ТАБЛИЦ И ОГРАНИЧЕНИЙ 27](#_Toc113796604)

[2.1 Цель практической работы 27](#_Toc113796605)

[2.2 Исходные данные 27](#_Toc113796606)

[2.3 Используемые программы 27](#_Toc113796607)

[2.4 Теоретические сведения 27](#_Toc113796609)

[2.5 Ход работы 31](#_Toc113796610)

[Задание для лабораторной работы №2 (часть 2) 37](#_Toc113796611)

## ЛАБОРАТОРНАЯ РАБОТА №2. Часть 1. УСТАНОВКА СОЕДИНЕНИЯ С СЕРВЕРОМ MICROSOFT SQL SERVER И ПРИНЦИПЫ СОЗДАНИЯ БАЗ ДАННЫХ

### Цель лабораторной работы

Познакомиться с основными принципами создания базы данных в MS SQL Server. Изучить операции, проводимые с базами данных в целом. Получить навыки использования программы "SQL Server Management Studio" для создания, удаления, регистрации, подключения, извлечения метаданных, резервного копирования и восстановления базы данных. Изучить SQL-операторы для создания, подключения и удаления базы данных. Познакомиться с основными принципами управления учетными записями и ролями.

### Исходные данные

Студент получает индивидуальный вариант исходных данных с кратким описанием предметной области, который используется при выполнении всех описанных в данном пособии лабораторных работ. При этом каждая очередная лабораторная работа является продолжением выполненной ранее и поэтому они должны обязательно выполняться последовательно.

### Используемые программы

1. Работающий на компьютере **сервер "MS SQL Server 2008 R2**" (MS SQL Server 2005-2019).
2. Установленная платформа **.NET Framework 2.0, 3.0, 3.5 или 4.0+**.
3. Операционная система **Microsoft Windows Windows 7/ 8/10+.**
4. Приложение "**SQL Server Management Studio 2008 rus+**", установленное на локальном компьютере.

### Теоретические сведения

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

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

**Microsoft SQL Server** — система управления реляционными базами данных (СУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов — **Transact-SQL**, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями.

Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

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

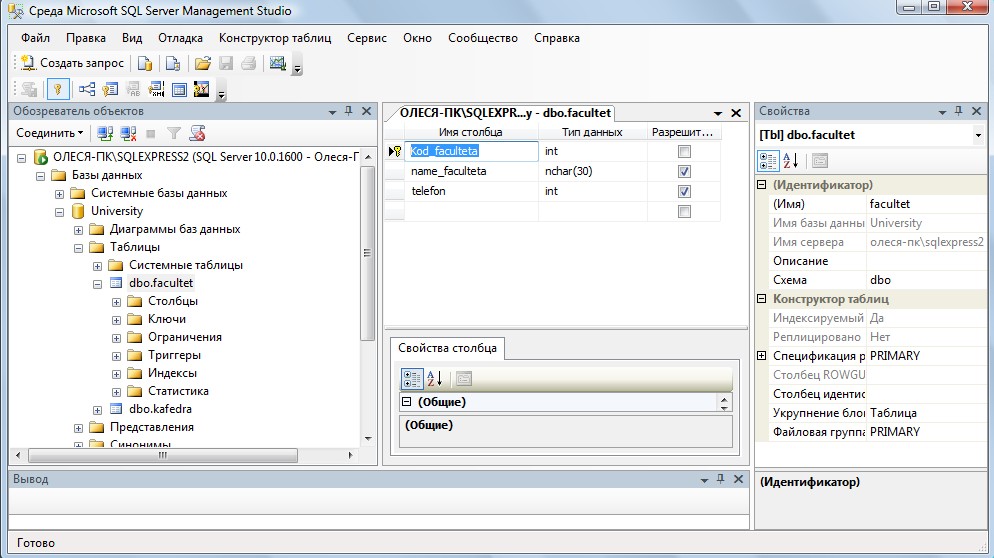
Система SQL Server 2008 позволяет обращаться к данным из любого приложения, разработанного с применением технологий Microsoft .NET и Visual Studio, а также в пределах сервисно-ориентированной архитектуры и бизнес-процессов — через Microsoft BizTalk Server. Сотрудники, отвечающие за сбор и анализ информации,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

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

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

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

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

Для выполнения операций с базой данных при проведении практических работ предлагается использовать программу " **SQL Server Management Studio 2008 Rus**" (рис. 1), представляющую собой наиболее распространенное и удобное средство администрирования баз данных под управлением MS SQL Server (Среда Management Studio Express доступна для свободной загрузки из центра загрузки Майкрософт - https://www.microsoft.com/ru-ru/download/details.aspx?id=30438).

*Рис. 1. Программа* SQL Server Management Studio

**Среда SQL Server Management Studio** — это интегрированная среда для доступа, настройки, управления, администрирования и разработки всех компонентов SQL Server. Среда SQL Server Management Studio объединяет большое число графических средств с набором полнофункциональных редакторов сценариев для доступа к SQL Server разработчиков и администраторов с любым опытом работы.

Среда SQL Server Management Studio обеспечивает следующие основные возможности:

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

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

### Задание

Практическую работу следует выполнять в следующем порядке:

1. Создать на сервере pi\_srv (или на локальном компьютере, если нет сервера) рабочую папку для хранения файлов, получаемых при выполнении практической работы. Эта папка должна располагаться в папке \**Базы данных\Группа\Студент** и соответствовать номеру выполняемой практической работы.
2. На основании индивидуального задания выбрать имя файла создаваемой базы данных. Для имени лучше всего выбрать одно или несколько английских слов, соответствующих наименованию предметной области. Использование для имени русских слов, записанных латинскими буквами, не допускается.
3. Открыть приложение " Среда SQL Server Management Studio ". Для этого можно либо воспользоваться меню Пуск (**Пуск/Программы/ Microsoft SQL Server 2008**

**/ Среда SQL Server Management Studio).**

1. Создать соединение с локальным или удаленным сервером.
2. Создать базу данных для своей предметной области с помощью диалога, выбрав сервер "pi\_srv" или локальный сервер “**Имя\_компьютера\SQLEXPRESS**”
3. Создать базу данных и указать в качестве имени файла **"\Базы данных\Группа\ФИО\_студента\Название\_БД"**.
4. Извлечь метаданные для автоматической генерации команды создания базы данных.
5. Удалить базу данных, выполнив команду "**Database/Drop Database**" (База данных/Удалить базу данных).
6. Создать базу данных вторым способом, выполнив в окне "**Script Executive**" операторы, полученные при извлечении метаданных перед предыдущим удалением.
7. Создать резервную копию базы данных.
8. Удалить базу данных.
9. Восстановить базу данных из резервной копии.
10. Сохранить файл сценария на сервере в папке "Студент", дав ему имя «лаб.№1» и стандартное расширение "\***.sql**".

### Ход работы

### Создание соединения с сервером

Выполните следующие инструкции:

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

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

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

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

Если компонент Database Engine является именованным экземпляром, то поле Имя сервера должно также содержать имя экземпляра в формате <имя\_компьютера>\<имя\_экземпляра>.

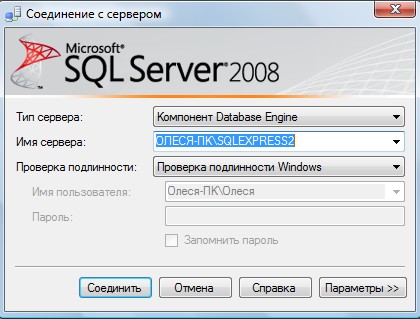
****

Рис. 2. Создание соединения сервером В параметрах указываем:

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

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

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

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

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

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

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

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

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

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

### Общие сведения о базах данных MS SQL Server

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

**База данных представляет собой:**

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

**Файлы базы данных**

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

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

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

**Группы файлов**

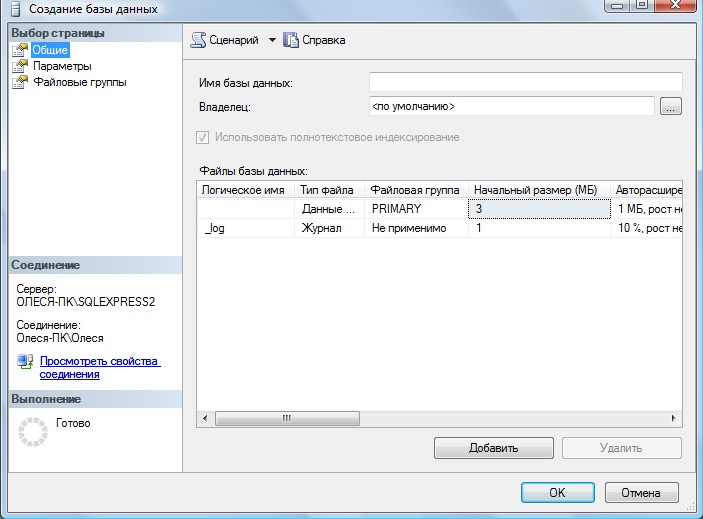
Группы файлов предназначены для объединения нескольких файлов. Каждый файл может входить не более чем в одну группу. Файлы журналов не могут принадлежать никаким группам. Группы файлов используются для распределения операций чтения/записи по нескольким дискам. Если группа содержит более одного файла, операции записи распределяются между файлами группы. Базы данных могут содержать до 32 768 групп файлов.

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

### Создание и регистрация базы данных

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

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



*Рис. 3. Диалоговое окно создания базы данных*

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

### University.

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

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

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

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

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

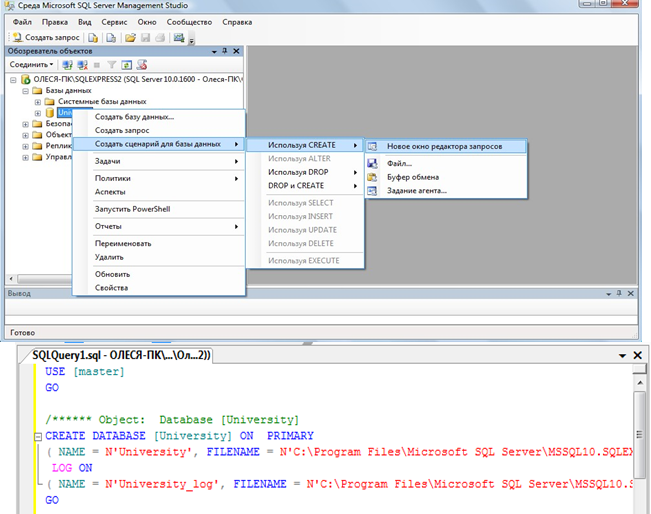


Рис. 4. Сгенерированный sql-код созданной базы данных

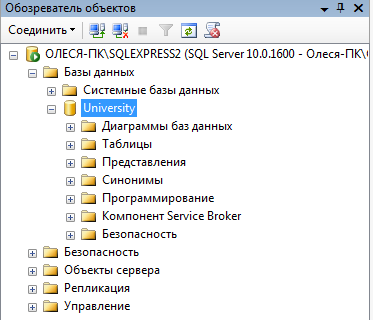
Содержащиеся в сценарии операторы отделяются друг от друга символом "**;**". Сценарий может содержать поясняющие комментарии двух видов:

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

При создании базы данных возможны следующие типичные ошибки:

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

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

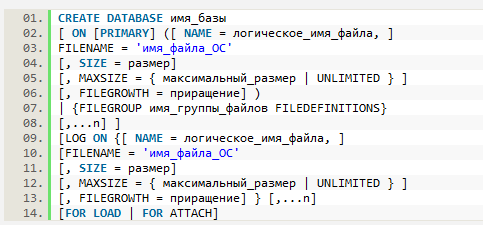


*Рис. 5. Перечень зарегистрированных баз данных в* SQL Server Management Studio

**Второй способ создания БД**. Выполнить в программе SQL Server Management Studio команду "**Создать запрос**"  на панели инструментов, затем ввести команду, создающую базу данных в окне "Script Execute" (рис. 3) и нажать кнопку .

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

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



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

Физические файлы будут находиться в стандартном каталоге.

Первичному файлу присваивается имя **имя\_базы.mdf**, а файлу журнала — **имя\_ базы\_log.ldf**.

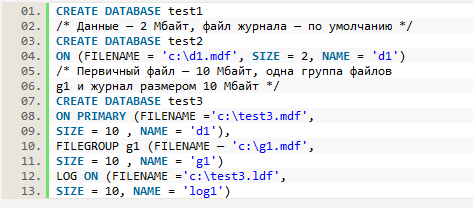
Если размер файлов не задан, то при создании размер первичного файла совпадает с размером первичного устройства базы **model**, а размер файла журнала и вторичных файлов данных равен 1 Мбайт. Он может быть и больше, если размер первичного файла базы данных model превышает 1 Мбайт. Хотя имена и размеры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

Команда **CREATE DATABASE** имеет следующие параметры:

* **PRIMARY** — файл определяется как первичное устройство.
* **NAME** — логическое имя; по умолчанию совпадает с именем файла.
* **FILENAME** — полное имя файла на диске.
* **SIZE** — исходный размер файла. Минимальный размер файла журнала равен 512 Кбайт.
* **MAXSIZE** — максимальный размер файла.
* **UNLIMITED** — размер файла не ограничивается.
* **FILEGROWTH** — приращение размера в мегабайтах (MB), килобайтах (KB) или процентах (%). По умолчанию приращение равно 10%.
* **FOR LOAD** — обеспечивает обратную совместимость со сценариями SQL, написанными для предыдущих версий SQL Server.
* **FOR ATTACH** — указывает, что файлы базы данных уже существуют.

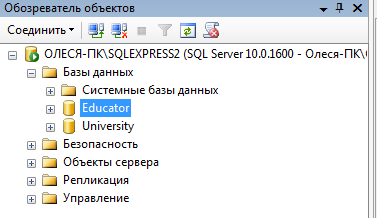
Пользователь, создавший базу данных, является ее владельцем. Все параметры конфигурации базы копируются из базы model, если только при создании базы не был указан параметр **FOR ATTACH.** В этом случае параметры конфигурации читаются из существующей базы данных. Рассмотрим некоторые примеры команды **CREATE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

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



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

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

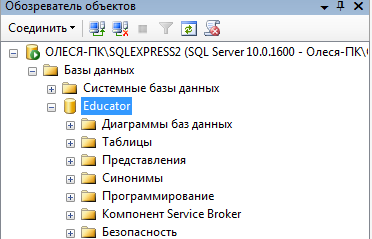


*Рис. 6. Окно проводника после выполнения сценария создания базы данных*

### Подключение к базе данных

Чтобы подключиться к зарегистрированной базе данных, надо выбрать нужную базу данных в списке (рис. 5) и сделать двойной щелчок мышкой на выбранной базе данных.

Если все параметры подключения были введены правильно, то произойдет подключение к базе данных, название подключенной базы данных в окне "Обозревателя объектов" будет выделено жирным шрифтом, а также появятся вложенные узлы c объектами, содержащимися в подключенной базе данных (рис. 7).



*Рис. 7. Зарегистрированные базы данных в* SQL Server Management Studio

После подключения к базе данных можно просматривать имеющиеся объекты, создавать новые, вносить и просматривать данные, а также проводить операции с имеющимися объектами.

После создания БД в окне **Обозревателя объектов** (его можно вызвать по **<F8>**) выбираем **DataBases (Базы данных)** и откроется список БД, в котором откроем созданную БД (если она не появилась, то в окне **Object Explorer** нажать **<F5>** для обновления списков), которая состоит из восьми вложенных разделов (некоторые содержат еще дополнительные разделы), соответствующих объектам СУБД SQL Server:

|  |  |  |
| --- | --- | --- |
| **Database Diagrams (Диаграммы БД)** | **Views (Представления)** | **Programmability (Объекты**  **программирования)** |
| **Tables (Таблицы)** | **Synonyms (Синонимы)** | **Security (Безопасность)** |
| **Service Broker** | **Storage** |  |

На начальном этапе раздел созданной БД пуст, за исключением некоторых объектов, которые создаются по умолчанию, например, в разделе **Security/ Users** создаются пользователи, которые имеют право на доступ к объектам БД, их можно изменить.

### Удаление базы данных

Для удаления базы данных можно использовать один из трех способов:

1. Выполнить в программе " SQL Server Management Studio " команду контекстного меню "**Удалить**", выбрав перед этим в списке базу данных, а затем подтвердить свое желание в диалоговом окне.
2. Выполнить оператор **DROP DATABASE** в SQL-редакторе.
3. Удалить файл с базой данных. Синтаксис оператора **DROP DATABASE:**

**DROP DATABASE database\_name;**

### Резервное копирование и восстановление

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.

Для создания резервной копии базы данных с помощью программы " SQL Server Management Studio " необходимо подключиться к базе данных, выбрать из контекстного меню базы данных **Задачи/ Создать резервную копию**. В открывшемся диалоговом окне "**Мастер резервного копирования**" задать несколько параметров и нажать кнопку [**Выполнить**], см. рис.8.

После выбора пути и файла для резервной копии в окне **Back Up Database** нажатием на OK 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.

|  |  |
| --- | --- |
|  |  |
| Рис.8. Создание резервной копии базы  данных | Рис.9. Восстановление базы данных |

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**], см.рис.9. Запускаем процесс восстановления. В случае успешного выполнения получим сообщение.

Резервное копирование и восстановление базы данных, наряду с процессом извлечения метаданных и последующего выполнения полученного сценария, можно использовать при переносе разрабатываемой базы данных между различными компьютерами для обеспечения самостоятельной работы студентов над практическими работами и курсовым проектом.

**Самостоятельно** Выполните вначале резервирование, а затем восстановление базы данных.

Удалите базу данных **Educator** c помощью скрипта сохраните sql-запрос.

### Копирование и перенос на другой сервер БД

Для просмотра, запуска, остановки служб MS SQL Server необходимо запустить утилиту **SQL Server Configuration Manager** (рис. 10).

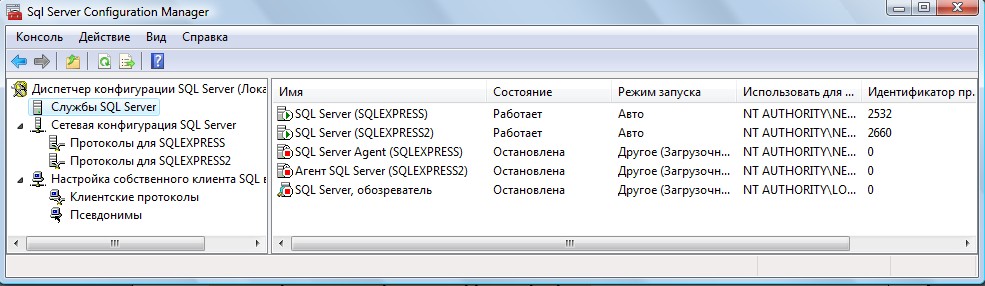


Рис.10. Список служб сервера БД

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 (рис. 5) выбираем **Attach…** (**Присоединить…**). В появившемся окне **Attach DataBases** (Присоединение базы данных) нажать <**Add>** и выбрать ваш файл БД с расширением **.mdf**.

### Системные базы данных

**Системные базы данных сервера, создаваемые при установке, и их файлы представлены в таблице 1.**



Все системные и пользовательские базы данных содержат в обязательном порядке 18 системных таблиц, которые хранят информацию, определяющие структуру и организацию соответствующей базы данных.

**MSSQL Server** поддерживает два основных класса приложений клиентского типа

:

1. приложения реляционных баз данных, использующие команды Transact - SQL с

расширениями ODBC и набор стандартных функций и объектно-ориентированных методов;

1. web - приложения, использующие команды Transact - SQL или запросы на языке Xpath и документы XML.

Оба класса приложений используют API интерфейс баз данных типа OLE DB или ODBC.

### Основные принципы управления учетными записями и ролями в MS SQL Server

Список системных процедур и команд, которые позволяют реализовать политику разделения прав между пользователя БД.

|  |  |
| --- | --- |
| **Название встроенной процедуры** | **Описание** |
| **sp\_grantlogin** | – позволяет использовать пользователей или группы ОС для соединения с Microsoft SQL Server™ , используя **Windows Authentication**.  Этот пример позволяет пользователю Windows NT Corporate\BobJ соединяться с SQL Server. Например,  **EXEC sp\_grantlogin 'Corporate\BobJ'** |
| **sp\_defaultdb** | Изменяет для пользователя БД по умолчанию  Этот пример устанавливает БД по умолчанию pubs для пользователя Victoria. Например,  **EXEC sp\_defaultdb 'Victoria', 'pubs'** |
| **sp\_grantdbaccess** | Добавляет учетную запись из раздела security в текущую БД, для учетных записей Microsoft Windows также дает разрешение на доступ к текущей БД.  **Синтаксис:**  **EXEC sp\_grantdbaccess [@loginame =] 'login' [,[@name\_in\_db =] 'name\_in\_db' [OUTPUT]]**  Этот пример добавляет учетную запись Corporate\GeorgeW в текущую БД и присваивает псевдоним внутри БД Georgie.  Например,  **EXEC sp\_grantdbaccess 'Corporate\GeorgeW', 'Georgie'** |
| **sp\_revokedbaccess** | Удаляет информацию об учетной записи из текущей БД.  **Синтаксис:**  **EXEC sp\_revokedbaccess [ @name\_in\_db = ] 'name'**  Этот пример удаляет учетную запись Corporate\GeorgeW из текущей БД.  **EXEC sp\_revokedbaccess 'Corporate\GeorgeW'** |
| **sp\_addrole** | Создает новую роль в текущей БД.  Этот пример создает новую роль в текущей БД с названием Managers.  **EXEC sp\_addrole 'Managers'** |
| **sp\_addrolemember** | В текущей БД назначает роль конкретному пользователю.  **Пример A.**  Этот пример добавляет учетную запись Corporate\JeffL из Windows NT в БД Sales как пользователя Jeff. Jeff затем получает роль Sales\_Managers в БД Sales.  **USE Sales** --сделать текущей БД Sales  **GO –**выполнить команду, а потом запустить следующую  **EXEC sp\_grantdbaccess 'Corporate\JeffL', 'Jeff' GO**  **EXEC sp\_addrolemember 'Sales\_Managers', 'Jeff'**  **Пример B.**  Этот пример добавляет пользователя SQL Server с именем Michael к роли Engineering в текущей БД.  **EXEC sp\_addrolemember 'Engineering', 'Michael'** |
| **sp\_helprotect** | Показывает список привилегий, ассоциированных с ролью. |
| **sp\_helprolemember** | Показывает список пользователей БД, входящих в указанную  роль |
| **sp\_addsrvrolemember** | Присвоение встроенной серверной роли для существующей учетной записи  **sp\_addsrvrolemember [ @loginame = ] 'login' , [ @rolename =** |
|  | **] 'role'**  Например:  **sp\_addsrvrolemember 'Admin\_DB', 'sysadmin'** |
| **sp\_dropsrvrolemembe r** | Удаление встроенной серверной роли для учетной записи или группы  **sp\_dropsrvrolemember [ @loginame = ] 'login' , [ @rolename =**  **] 'role'**  Например:  **sp\_dropsrvrolemember 'Admin\_DB' , 'sysadmin'** |
| **sp\_helpsrvrole** | Описание только встроенных ролей в SQL Server **sp\_helpsrvrole [ [ @srvrolename = ] 'role' ]** Например:  **sp\_helpsrvrole 'sysadmin'** |
| **sp\_helpsrvrolemember** | Возвращает список ролей и учетных записей, которым присвоены эти роли  **sp\_helpsrvrolemember [ [ @srvrolename = ] 'role' ]**  Например:  **sp\_helpsrvrolemember 'sysadmin'** |
| **sp\_srvrolepermission** | Возвращает список ролей и разрешений, которые присвоены этим ролям  **sp\_srvrolepermission [[@srvrolename =] 'role']**  Например:  **sp\_srvrolepermission 'sysadmin'** |
| **sp\_addlogin**  **sp\_adduser** | Создание новой учетной записи в SQL Server в разделе Sequrity:  **sp\_addlogin [ @loginame = ] 'login' [ , [ @passwd = ] 'password' ]**  **[ , [ @defdb = ] 'database' ]**  **[ , [ @deflanguage = ] 'language' ] [ , [ @sid = ] sid ]**  **[ , [ @encryptopt = ] 'encryption\_option' ]**  Например:  **sp\_addlogin 'login1',sysname, 'DB\_Books'**  Создает пользователя в SQL Server без PUBLIC в БД ' DB\_Books'.  Нужно еще использовать  **sp\_adduser [ @loginame = ] 'login' [ , [ @name\_in\_db = ] 'user' ]**  **[ , [ @grpname = ] 'group' ],**  **Пример:**  Создана база данных DB\_Books. В ней создан пользователь Admin\_DB с серверной ролью sysadmin, с ролью в БД db\_owner.  Cоздать в QueryAnalyzer нового пользователя с именем Public\_ |
|  | и паролем Public\_1 (пароль не должен совпадать с именем пользователя) с помощью следующих команд (не забудьте нажать F5 для запуска команд на выполнение):  **EXEC sp\_addlogin 'Public\_','Public\_1', 'DB\_Books' use DB\_Books**  **EXEC sp\_adduser 'Public\_','Public\_'**  В БД DB\_Books создан пользователь Public\_ с ролью в БД DB\_Books public. |
| **Deny (отрицание)** | Этот пример запрещает несколько системных привилегий для нескольких пользователей.  Пользователи не могут использовать системные привилегии  **CREATE DATABASE** or **CREATE TABLE**, если они не  наделены имим через команду **GRANT**. **Пример:**  **DENY CREATE DATABASE, CREATE TABLE**  **TO Mary, John, [Corporate\BobJ]**  **DENY SELECT, INSERT, UPDATE, DELETE ON authors TO**  **Mary, John, Tom** |
| **Grant (предоставлять)** | This example grants multiple statement permissions to the users Mary and John, and the Corporate\BobJ Windows NT group.  **GRANT CREATE DATABASE, CREATE TABLE**  **TO Mary, John, [Corporate\BobJ]**  Назначение разрешения на выборку (SELECT) для роли PUBLIC в таблице Authors:  **GRANT SELECT ON Authors TO public** |
| **Revoke (отменять)** | This example revokes multiple statement permissions from multiple users.  **REVOKE CREATE TABLE, CREATE DEFAULT**  **FROM Mary, John**  This example removes the denied permission from Mary and, through the SELECT permissions applied to the Budget role, allows Mary to use the SELECT statement on the table.  **REVOKE SELECT ON Budget\_Data TO Mary** |

Создание пользователей для доступа к серверу через утилиту Microsoft SQL Server Management Studio

**Создадим новую учетную запись** для нашей базы данных **University**. 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.

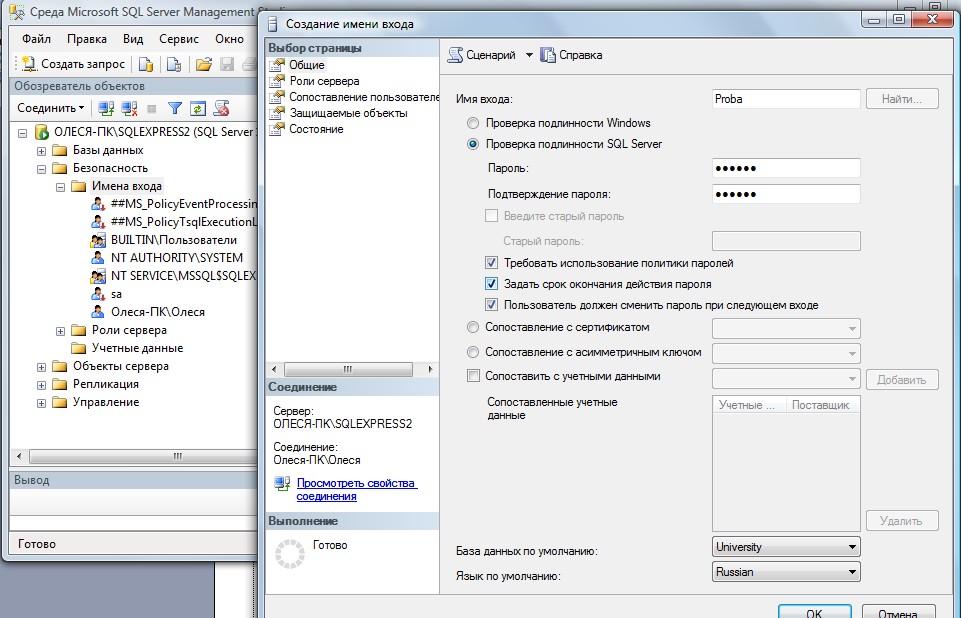


Рис. 11. Раздел Безопасность (Security) для работы с пользователями и создание нового пользователя (при SQL Server аутентификации нужно снять галочки с **Enforce password policy**)

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**.

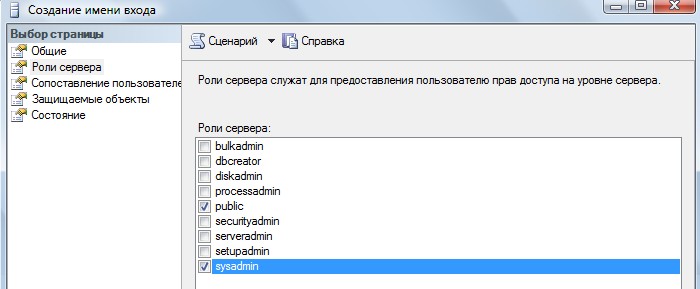


Рис. 12. Настройка серверной роли для нового пользователя (весь список серверных ролей с их привилегиями в конце работы)

Далее просмотрите раздел **Сопоставление пользователя**. Установите для базы данных **University** у пользователя **Proba** права доступа **Db\_owner**, означающие, что пользователь может выполнять **любые действия с БД**. Ниже перечислены все возможные варианты прав доступа.

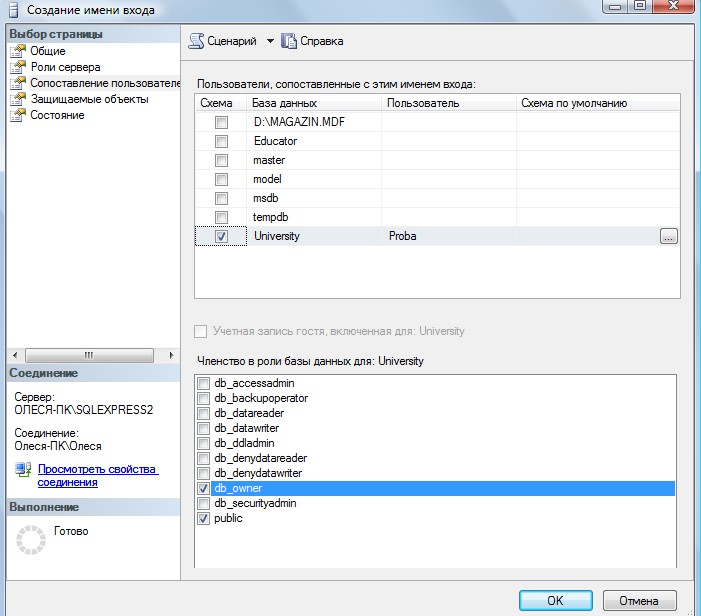


Рис. 13. Настройка роли базы данных для нового пользователя (весь список ролей баз данных с их привилегиями ниже)

**Перечень ролей БД:**

**Public** – минимальные права доступа к БД (на просмотр) **Db\_owner** – может выполнять любые действия с БД **Db\_accessadmin** – добавляет и удаляет пользователей БД

**Db\_sequrityadmin** – управляет ролями в БД и разрешениями на запуск команд и работу с объектами БД

**Db\_ddladmin** – добавляет, изменяет и удаляет объекты БД **Db\_backupoperator** – осуществляет резервное копирования БД **Db\_dataSTUDENT** – может просматривать все данные в каждой таблице в БД

**Db\_datawriter** - может добавлять, удалять и изменять данные в каждой таблице в

БД

**Db\_denydataSTUDENT** – запрет на просмотр всех данных в каждой таблице в БД

**Db\_denydatawriter** - запрет на добавление, удаление и изменение всех данных в

каждой таблице в БД

Далее перейдите на раздел **Состояние**. Установите опции **Разрешение к подключению к ядру СУБД** – предоставить и **имя входа** включить.

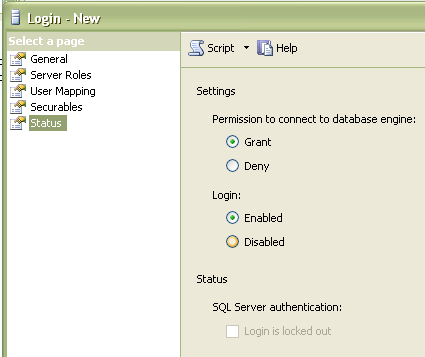


Рис. 14. Разблокирование создаваемой учетной записи

После нажатия на <OK> 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 найдите только что созданного пользователя.

**Создание ролей программно**

Для упрощения управления правами доступа в системе создаются **роли**, которые затем можно назначать **группе пользователей**.

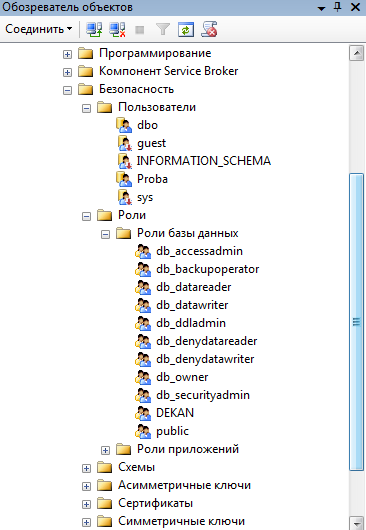
Создадим для нашего примера роли декана (**DEKAN**) и студента (**STUDENT**).

**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 . 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

— предоставитель). 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 (немного забегая вперед) создадим таблицу **Discuplinu**. Без объяснения синтаксиса 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 студент Petrov\_Stud и присвоения роли:

**EXEC sp\_addlogin 'Petrov\_Stud','Petrov', 'University'**

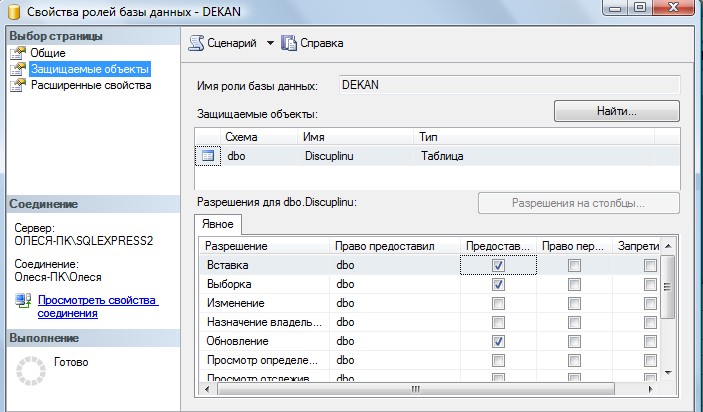
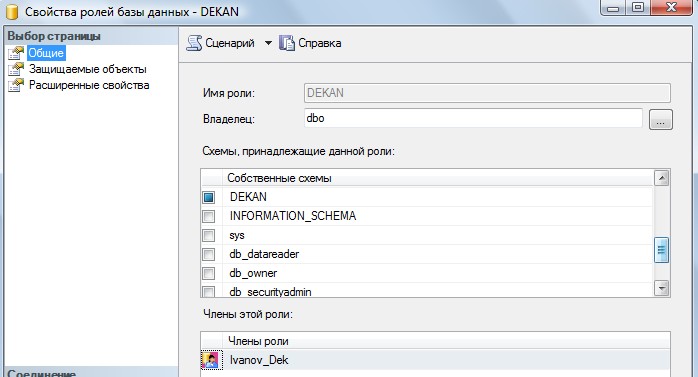
**use University**

**EXEC sp\_adduser 'Petrov\_Stud','Petrov\_Stud'**

**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.

**Самостоятельно** просмотрите свойства роли базы данных **Student.** 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 для лабораторной работы №2 (часть 1)

* Создать файл базы данных, согласно номеру варианта, выданного в лабораторной работе №1 с помощью sql-команды.
* Создать резервную копию базы данных.
* 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 с названием **ФамилияСтудента\_Лаб\_2.**
* Создать текстовый отчет, в котором отобразить sql-команды разработанных запросов и скриншоты результатов работы из СУБД **SQL Server Management Studio**.

## ЛАБОРАТОРНАЯ РАБОТА №2. Часть 2. РАЗРАБОТКА ТАБЛИЦ И ОГРАНИЧЕНИЙ

### Цель практической работы

Изучить способы создания, изменения и удаления таблиц. Получить навыки использования приложения " SQL Server Management Studio " для создания, удаления и изменения структуры таблиц. Изучить SQL-операторы для работы с таблицами и индексами. Изучить используемые в SQL Server типы ограничений. Получить навыки использования программы " SQL Server Management Studio " для создания, изменения и удаления ограничений. Изучить SQL-операторы для работы с ограничениями.

### Исходные данные

Исходными данными является индивидуальное задание и результат предыдущих практических работ.

### Используемые программы

Программа " SQL Server Management Studio " и установленный сервер Microsoft SQL Server .

**Создание таблиц**

### Теоретические сведения

* + 1. Логическая структура и физическая реализация баз данных MS SQL Server Данные в сервере SQL Server хранятся в базах данных. Структуру баз данных

необходимо рассматривать на двух уровнях: **логическом и физическом**.

**Логическая структура базы данных** определяет структуру таблиц, взаимосвязи между ними, список пользователей, хранимые процедуры, правила, умолчания и другие объекты базы данных.

**Физическая структура базы данных** включает в себя описание файлов и групп файлов базы данных, журнала транзакций, первоначальный размер файлов, шаг прироста базы данных, ее максимальный размер, параметры конфигурации и другие физические характеристики.

**Все логические компоненты** базы данных SQL Server называются объектами и подразделяются на **11 типов**.

Для объектов можно устанавливать различные правила доступа со стороны пользователей.

Функциональное назначение объектов базы данных сервера SQL Server можно кратко определить следующим образом:

1. **Table** - таблица, представляющая собой матрицу из строк и столбцов.

Каждая строка (или запись) состоит из значений атрибутов конкретного объекта.

Столбец (или поле записи) содержит совокупность атрибутов рассматриваемых объектов некоторой предметной области. Некоторые столбцы таблицы могут быть вычисляемыми. В этих случаях для них задается расчетная формула.

1. **User-defined data type** - пользовательский тип данных, создаваемый на основе системных. Имя нового типа должно быть уникальным в пределах владельца.
2. **View** - представление, являющееся виртуальной таблицей, содержимое которой определяется запросом. Эта таблица не содержит данных, а только их представляет, возможно из нескольких таблиц. Данные из представления не сохраняются в базе данных. Физически представление реализуется в виде запроса **SELECT**. Представления используются в следующих случаях:
   1. Для ограничения доступа пользователей к определенным строкам таблицы;
   2. Для ограничения доступа пользователей к определенным столбцам таблицы;
   3. Для представления данных столбцов разных таблиц в виде одного объекта;
   4. Для просмотра информации, получающейся в результате преобразования данных столбцов.
3. **Stored procedure** - хранимая процедура, представляющая собой группу команд Transact-SQL, объединенных в один модуль. Каждая хранимая процедура имеет уникальное , в пределах базы данных, имя, по которому она вызывается. Хранимая процедура может вызывать другие хранимые процедуры. В состав SQL Server входит большое количество встроенных процедур, которые называются системными и имена которых начинаются с префикса **sp\_**.
4. **Trigger** - триггер, представляющий собой специальную хранимую процедуру, автоматически запускаемую при добавлении, изменении или удалении данных из таблицы. Триггеры делятся на три категории:

**UPDATE TRIGGER** - триггеры изменения; **INSEART TRIGGER** - триггеры вставки; **DELETE TRIGGER** - триггеры удаления.

Действия, выполняемые в одном триггере, могут вызвать другие триггеры (вложенные триггеры).

1. **Index** - индекс, представляющий собой структуру, связанную с таблицей или представлением и предназначенную для ускорения поиска информации в этой таблице или представлении. Индекс определяется для одного или нескольких столбцов, называемых индексированными столбцами. Индекс содержит отсортированные значения индексированного столбца или столбцов со ссылкой на соответствующую строку исходной таблицы или представления. Алгоритмы поиска в отсортированных данных гораздо эффективнее, чем в неотсортированных.
2. **Rule** - правило, используемое для ограничения значений, хранимых в столбце таблицы или в пользовательском типе данных. Одно и тоже правило может связываться с множеством столбцов различных таблиц и пользовательских типов данных только в текущей базе данных. Правило создается командой **CREATE RULE** и связывается с объектом базы данных с помощью процедуры spbindrule. Правила оставлены в Transact- SQL для совместимости со старыми версиями сервера.
3. **Constraint** - ограничение целостности, представляющее собой механизм, обеспечивающий автоматический контроль соответствия данных установленным условиям, или ограничениям целостности. Ограничения целостности имеют приоритет над триггерами, правилами и значениями по умолчанию. Имеется пять ограничений целостности, различающихся по функциональности и области применения:

**NULL** - действует на уровне столбца и пользовательского типа данных и либо разрешает (NULL), либо запрещает (NOT NULL) хранение значений NULL.

**CHECK** - действует на уровне столбца и ограничивает диапазон значений, которые могут быть сохранены в столбце, путем проверки логического условия для вводимых данных. При вводе или изменении данных вводимое значение подставляется в условие. Если полученный результат

TRUE, то изменения данных принимаются, иначе - отвергаются и генерируется сообщение об ошибке. Для одного столбца можно задать несколько ограничений типа **CHECK** (проверок):

**CONSTRAINT humanavance**

**CHECK (human\_ avance BEETWEEN 0 and 700)).**

**UNIQUE** - действует на уровне столбца и гарантирует уникальность в столбце вводимых значений. В отличии от ограничения **PRIMARY KEY**, это ограничение допускает хранение значений NULL.

**PRIMARY KEY** - действует на уровне столбца или таблицы и гарантирует уникальность в пределах таблицы первичного ключа, состоящего из одного или нескольких столбцов. Ни для одного из столбцов ключа не должно быть установлено свойство NULL. Когда используется один столбец, то для него необходимо также задать и свойство UNIQUE. В таблице создается только один первичный ключ. При его выборе надо учитывать требования удобства и функциональности.

**FOREIGN KEY** - действует на уровне таблицы и связывается с одним из кандидатов на первичный ключ в другой таблице. Таблица, в которой определен внешний ключ с помощью этого ограничения, называется зависимой, а таблица с кандидатом на первичный ключ - главной. В зависимую таблицу нельзя вставить строку, если внешний ключ не имеет соответствующего значения в главной таблице. Из главной таблицы нельзя удалить строку, если с ней связана хотя бы одна строка в зависимой таблице. Формат задания ограничения таков:

**FOREIGN KEY REFERENCES** имя главной таблицы (кандидат на первичный ключ или ее ключ).

1. **Default** - умолчание, представляющее собой значение, которое будет присвоено элементу столбца таблицы при вставке строки (записи), если в команде вставки явно не указано значение для этого столбца. Умолчаниями могут быть константы, а также встроенные функции и математические выражения, возвращающие конкретные значения. Создание умолчания выполняется командой **CREATE DEFAULT**, а удалить само умолчание командой **DROP DEFAULT**.
2. **Function** - функция, представляющая собой программный модуль, выполняющий некоторые часто используемые действия над данными и возвращающий значение какого-либо типа. Имя функции, осуществляющее ее вызов, может указываться в любом выражении языка Transact-SQL.

**Встроенные функции** (built-in functions) являются составной частью среды программирования сервера, выполняют заранее предопределенную последовательность команд и не могут изменяться пользователем: **COUNT, SUM, MIN, MAX** и т. д. **Функции пользователя** (user\_ defind function) создаются пользователем по правилам языка Transact-SQL для реализации разрабатываемых алгоритмов.

Каждая копия, или **экземпляр сервера SQL Server имеет 4 системных базы данных** с именами **master, msdb, tempdb и model**, а также несколько созданных администратором пользовательских баз данных. Набор системных баз постоянен и не может быть изменен. Обращаться к этим базам напрямую запрещено. Обращаться к ним можно только с помощью специально разработанных интерфейсов:

* 1. С помощью системных хранимых процедур;
  2. С помощью интегрированной среды **Enterprice Manager**;
  3. Используя программные интерфейсы **Transact-SQL (API).**

При соединении с экземпляром сервера организуется связь с конкретной базой данных на сервере. Эта база данных называется ***текущей.*** Она определяется для каждого пользователя системным администратором.

Пользователь может переключаться на другую базу, используя команду

**USE < имя базы данных>**

или функции API для изменения текущей базы данных.

SQL Server позволяет соединить базы данных от одной копии сервера и подсоединить к другой, а затем восстановить подсоединение к прежней копии.

Все базы данных SQL Server, как системные, так и пользовательские, физически **организованы одинаково**. **Каждая база данных хранится в отдельных файлах**. В отдельном файле хранится **журнал транзакций**, создаваемому автоматически для каждой базы данных. Это повышает надежность системы. Если произойдет сбой и файлы базы данных будут повреждены, то можно восстановить базу данных из резервной копии, а затем восстановить сделанные изменения из журнала транзакций, который останется не поврежденным.

Таким образом, каждая база данных имеет, минимум, **два файла: один для базы данных и один для журнала транзакций**. Эти файлы имеют различную структуру и при работе с ними применяются разные правила.

Основная единица хранения данных на уровне файла базы данных - это страница, которая участвует в операциях ввода-вывода как единое целое даже тогда, когда требуется всего одна строка. Размер страницы равен 8 Кбайт.

**Файл журнала транзакций** не имеет страниц и экстентов. Он содержит только последовательность записей транзакций, выполняемых в базе данных.

Каждая страница файла базы данных имеет объем 8192 байт. Первые 96 байт страницы отводятся под заголовок, в котором хранится системная информация: тип страницы, объем свободного места на странице, идентификационный номер таблицы или индекса - владельца страниц:

Имеется шесть типов страниц:

1. **Data.** В страницах этого типа хранятся собственно данные, исключая данные типа text, ntext и image.
2. **Index.** Страницы этого типа используются для хранения информации об индексированных таблицах.
3. **Text/Image.** В страницах этого типа хранятся данные типа text, ntext и image.
4. **Global Allocation Map (GAM).** В страницах данного типа хранится информация об использовании экстентов ( групп страниц). Экстент состоит из 8 страниц (64 Кбайт).
5. **Page Free Space.** В страницах этого типа хранится информация о свободном пространстве на страницах.
6. **Index Allocation Map (IAM).** Страницы этого типа хранят информацию об экстентах, используемых таблицами или индексами.

Более сложные базы данных имеют несколько файлов для данных и для транзакций. В этом случае они объединяются в группы для упрощения администрирования базы данных. Любой из таких файлов может располагаться на отдельном диске.

**Многофайловая база данных имеет в своем составе файлы следующих типов:**

**Primary (\*.mdf)** - основной файл, который содержит системную информацию о самой базе данных и ее объектах. В этом файле размещаются системные таблицы и описание объектов базы данных. Здесь могут храниться и данные. В базе данных такой файл только один и его наличие в базе обязательно.

**Secondary (\*.ndf)** - вторичный файл, который используется только для хранения данных и не содержит системной информации. Он может отсутствовать в базе данных. Если задано несколько файлов этого типа, то они могут быть организованны в группы и распределены по разным физическим дискам.

**Transaction Log (\*.ldf)** - файл журнала транзакций. Можно использовать несколько таких файлов для ускорения операций ввода-вывода, так как транзакции записываются параллельно во все файлы.

Каждый файл, используемый в базе данных, имеет два имени:

**Logical File Name** - логическое имя файла, которое используется в командах Transact-SQL при ссылке на конкретный файл;

**OS File Name** - имя файла в операционной системе, которое используется в операционной системе.

Для каждого файла базы данных можно задать свойство автоматического роста и шаг прироста в мегабайтах или в процентах от первоначального роста, а также максимальный размер, до которого возможен рост файла.

SQL-Server 2000 обеспечивает создание групп следующих трех типов:

**Primary File Group** - основная группа файлов, которая включает первичный файл и все файлы, не включенные в другие группы. База данных может иметь только одну основную группу файлов.

* + 1. **Таблицы (Tables)**

**Microsoft SQL Server** – реляционная СУБД, поэтому все данные в Sql хранятся в виде **двумерных таблиц** со строками и столбцами. **Строки** называются кортежами или записями, а **столбцы** – доменами или полями.

В этой практической работе рассматриваются способы создания таблиц.

**Основные ограничения, которым должны удовлетворять таблицы:**

1. Каждый столбец в таблице имеет уникальное имя.
2. Все данные в столбце должны быть одного типа.
3. Порядок строк и столбцов в таблице не имеет значения.
4. В таблице не может быть двух одинаковых строк.
   * 1. **Индексы**

Microsoft SQL Server (как и другие реляционные СУБД) хранит записи в таблицах в неупорядоченном виде. Записи, добавляемые в таблицу одна за другой, не обязательно окажутся "рядом". Данные, извлекаемые из таблицы, также не имеют какого-либо порядка, кроме того, который явно указан в запросе на выборку информации.

**Индекс** – это упорядоченный указатель на записи таблицы. Индекс состоит из пар значений "значение поля" – "физическое расположение записи", поэтому по значению поля (или полей), входящего в индекс, при помощи индекса можно быстро найти место в таблице, где располагается запись, содержащая это значение.

Устройство индексов на физическом уровне для нас совершенно не имеет значения. Важно только знать, что создание индексов может привести к значительному ускорению процессов поиска и сортировки.

Не следует создавать индекс на поля с ограниченным набором значений – например, на поле, хранящие пол человека, которое содержит только два значения – "м" и "ж".

Использование индексов имеется два отрицательных последствия:

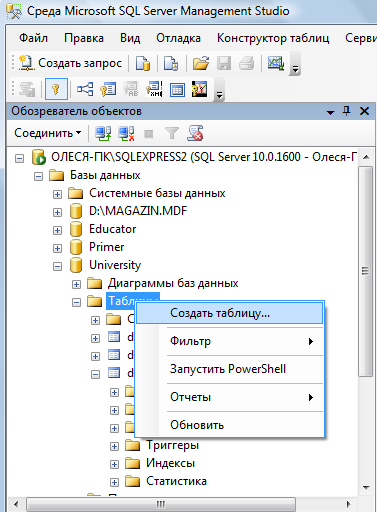
1. Для индексов дополнительно тратится дисковое пространство.
2. Наличие индексов замедляет модификацию данных в таблице.

### Ход работы

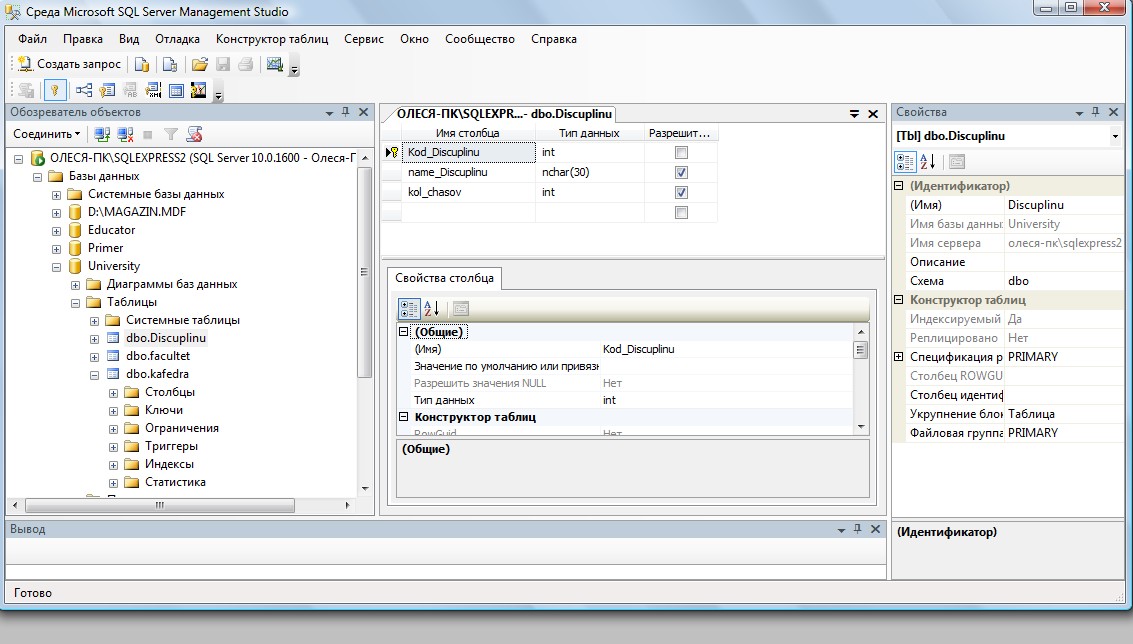
**Выполнение операций создания таблиц и индексов в диалоговом режиме.**

Откройте среду SQL Server Management Studio, выполните соединение с сервером, откройте созданную базу данных **University** в лаб.раб. №2 – часть 1.

**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)" (рис. 15). В результате откроется окно создания таблицы (рис. 16).



*Рис. 15. Окно "Проводник" с перечнем таблиц*



*Рис. 16. Окно формирования таблицы в диалоговом режиме*

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.

**Самостоятельно**

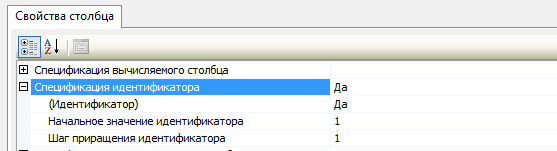
Создайте новую таблицу, хранящую информацию о всех факультетах в университете.

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 **varcha**r(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’).

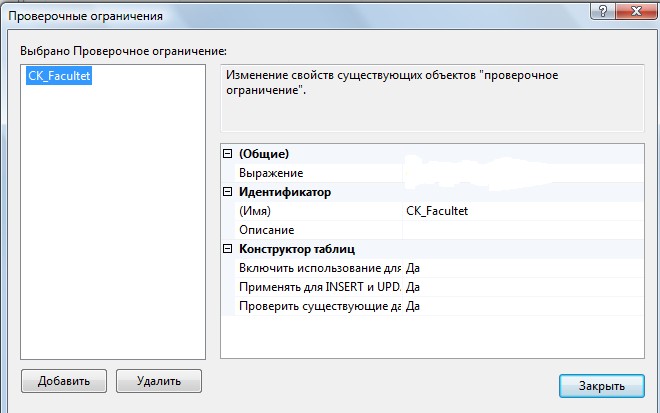
**Примечание.**

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



1. 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 ‘1 000 000’ будет иметь вид

**Tel\_decanata< 1 000 000.**

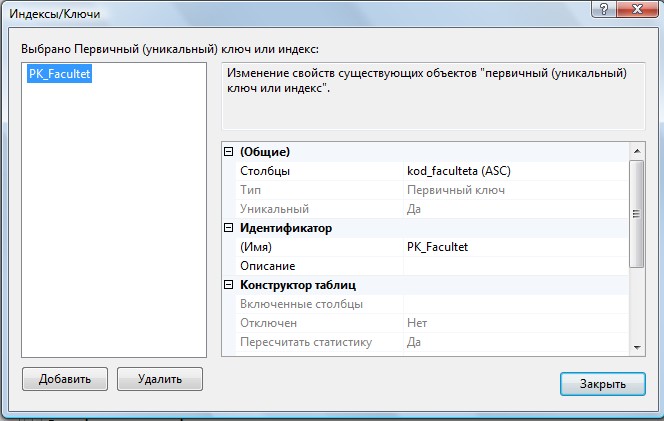


После введения данных о всех полях таблицы следует нажать кнопку [**Сохранить**] на панели инструментов.

Создадим для данной таблицы индексы (для других таблиц создавать индексы не нужно!!).

Для этого выберите из контекстного меню "**Индексы и ключи**" (рис. 17).

В нашем случае для первичного ключа автоматически присваивается уникальный индекс по возрастанию, см. рис.17.



*Рис. 17. Окно просмотра и редактирования индексов*

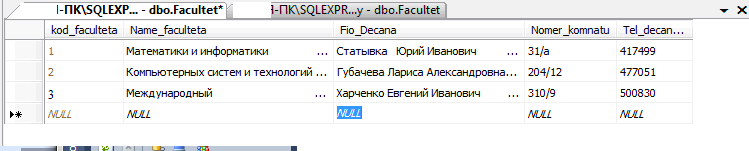
Мастер позволяет просматривать, редактировать, создавать и удалять индексы.

**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**" находиться 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.**

После создания индекса его можно в любой момент изменить, если изменить параметры индекса и снова нажать кнопку **Сохранить.**

Для заполнения таблиц данными вначале выделите таблицу в окне проводника и из контекстного меню выберите **Изменить первые 200 строк**. Вид окна представлен на рис. 18.



*Рис. 18. Диалог создания записей в таблице*

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

После ввода данных нажмите на панели кнопку о подтверждении изменения данных

 Выполнить SQL – код.

Примечание.

При вводе данных в поле **kod\_faculteta** заполнять числами не нужно, они будут добавлены автоматически согласно формуле инкремента.

При вводе данных в поле **tel\_decanata** значение можно оставить не заполненным. В результате выполнения sql-кода поле будет заполнено значением по-умолчанию.

При вводе данных в поле **tel\_decanata** значение более 1 000 000 будет отображаться ошибка ввода связанной с проверочным ограничением.

**Самостоятельно** 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’)**

**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 .

**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** (см. рис. 19) перейдите на поле

**Спецификация таблиц и столбцов** и нажмите кнопку напротив  для его изменения.

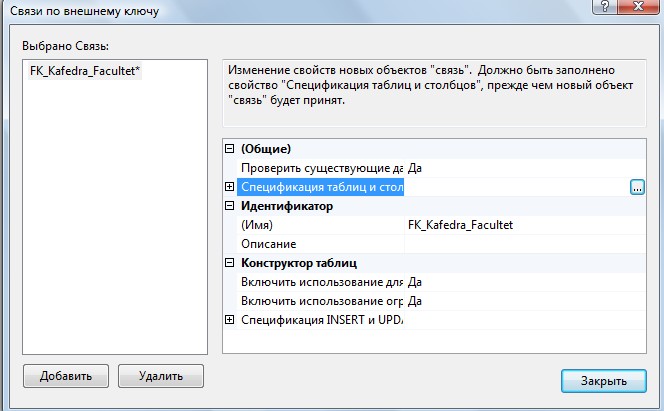


Рис. 19. *Диалог создания внешних ключей*

Вам будет открыто диалоговое окно создания **Внешних ключей** (рис. 20), 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.

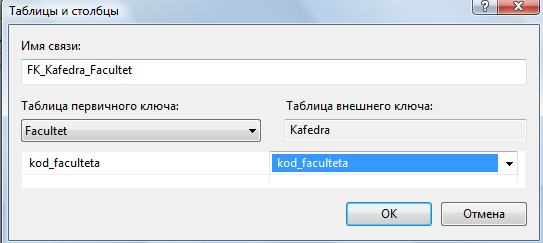
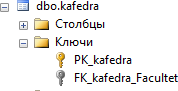


Рис. 20. Создание Внешних ключей

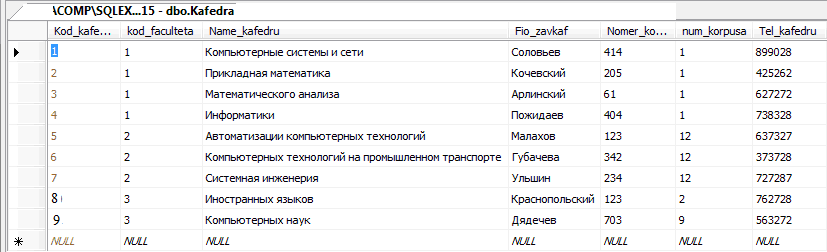
Установите правила удаления и обновления данных. Для этого перейдите на поле **Спецификации INSERT и UPDATE** и выберите правило **каскадного** удаления и обновления.

**Сохраните таблицу.**

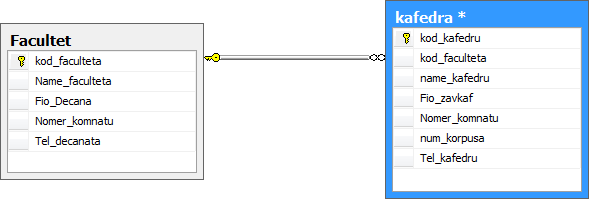
После подтверждения операции у вас в таблице **Кафедра** появится внешний ключ по полю **kod\_faculteta.** Для этого в окне проводника выберите в таблице **Кафедра** элемент **Ключи** и вы увидите первичный ключ и внешний ключ.



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



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**.

### Задание для лабораторной работы №2 (часть 2)

Для ранее созданной базы данных по номеру варианта (созданной в лаб. работе

№2 – часть 1) в СУБД **SQL Server Management Studio**:

* 1. Создайте все таблицы базы данных, ключи, ограничения и связи.
  2. Каждая таблица должна иметь ограничение первичного ключа.
  3. С помощью ограничений внешнего ключа должны быть заданы все имеющиеся связи между таблицами.
  4. В зависимости от условий выданного задания в некоторых таблицах могут быть наложены дополнительные ограничения целостности на столбцы или должны быть разработаны вычисляемые поля.
  5. Создайте диаграмму базы данных.
  6. Заполните таблицы данными не менее 5 записей в каждой.
  7. Создать текстовый отчет, в котором отобразить скриншоты результатов работы в СУБД **SQL Server Management Studio** (окно с базой данных с перечнем всех таблиц, проекты таблиц с перечнем столбцов, окна ограничений внешних ключей (создание), окно с перечнем ключей для каждой таблицы, окна с данными для каждой таблицы, диаграмма базы данных).

## 