

# Лабораторный практикум

МДК 11.01 Технология разработки  
и защиты баз данных

SQL

Куропаткина О.П.

## **ЛАБОРАТОРНАЯ РАБОТА №2**

### **СОЗДАНИЕ БАЗЫ ДАННЫХ, ТАБЛИЦ И ОГРАНИЧЕНИЙ**

## 1. СОЗДАНИЕ БАЗЫ ДАННЫХ

### 1.1 ПОДГОТОВКА К СОЗДАНИЮ ПОЛЬЗОВАТЕЛЬСКОЙ БАЗЫ ДАННЫХ

#### Утилита SQL Server Management Studio

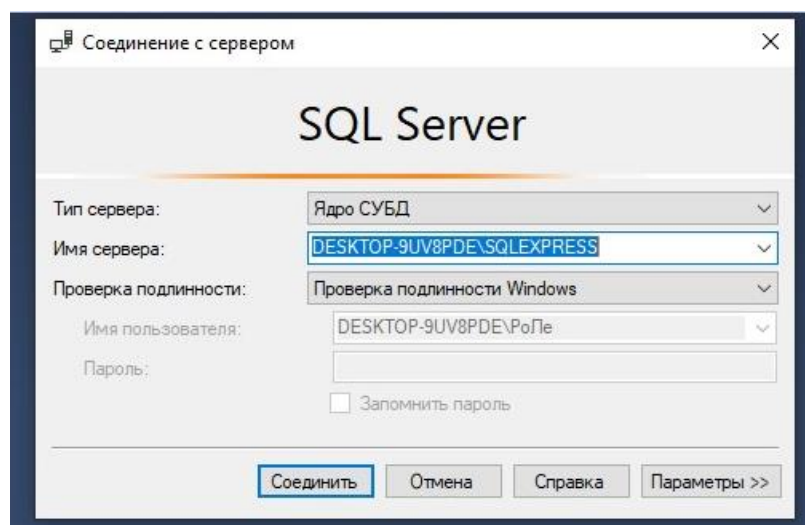
Подавляющую массу задач администрирования SQL Server можно выполнить в графической утилите SQL Server Management Studio. В ней можно создавать базы данных и все ассоциированные с ними объекты (таблицы, представления, хранимые процедуры и др.). Здесь вы можете выполнить последовательности инструкций Transact-SQL(запросы). В этой утилите можно выполнить типовые задачи обслуживания баз данных, такие как резервирование и восстановление. Здесь можно настраивать систему безопасности базы данных и сервера, просматривать журнал ошибок и многое другое.

Для запуска Management Studio в меню «Пуск» операционной системы выберите пункт «Microsoft SQL Server Tools 18 / Microsoft SQL Server Management Studio».

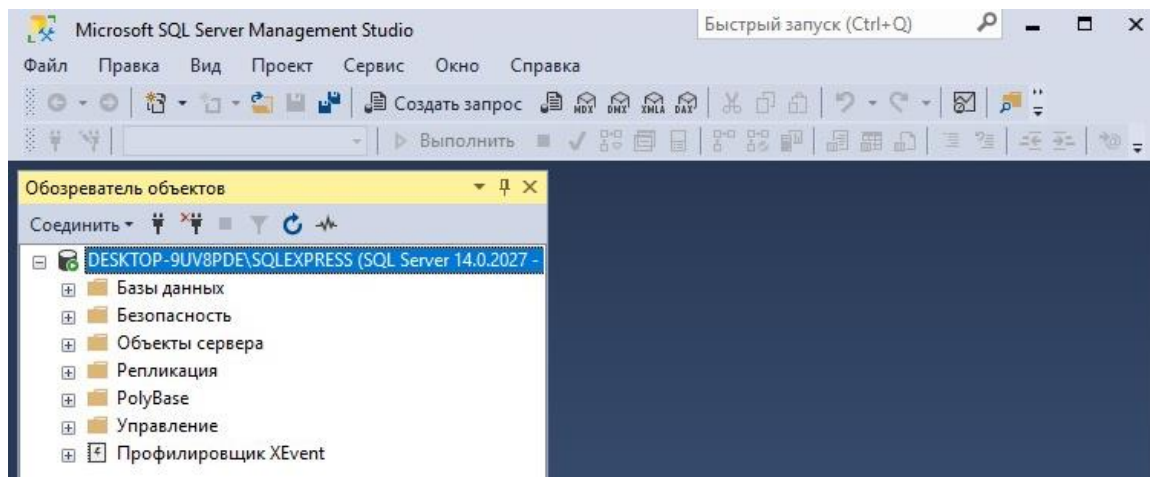
#### Подключение к серверу

В окне «Соединение с сервером» необходимо указать следующую информацию:

- *Тип сервера.* Здесь следует выбрать, к какой именно службе необходимо подключиться. Оставьте вариант «Ядро СУБД».
- *Имя сервера.* Позволяет указать, к какому серверу будет осуществляться подключение. По умолчанию имя SQL Server совпадает с именем компьютера. Выберите ваш локальный компьютер.
- *Проверка подлинности.* Способ аутентификации, можно выбрать «Проверка подлинности Windows» или «Проверка подлинности SQL Server». Первый способ использует учетную запись, под которой текущий пользователь осуществил вход в Windows. Вариант SQL Server использует свою собственную систему безопасности. Оставьте вариант проверки подлинности Windows.



После подключения экземпляр сервера будет отображаться на панели «Обозреватель объектов».



Прежде чем перейти к созданию своих собственных рабочих баз данных рассмотрим служебные базы данных SQL Server, которые создаются автоматически в процессе его установки. Если мы раскроем узел «Базы данных – Системные базы данных» в обозревателе объектов, то увидим следующий **набор служебных баз данных**:

- *master*. Главная служебная база данных всего сервера. В ней хранится общая служебная информация сервера: настройки его работы, список баз данных на сервере с информацией о настройках каждой базы данных и ее файлах, информация об учетных записях пользователей, серверных ролях и т.п.
- *msdb*. Эта база данных в основном используется для хранения информации службы SQL Server Agent (пакетных заданий, предупреждений и т.п.), но в нее записывается и другая служебная информация (например, история резервного копирования).
- *model*. Эта база данных является шаблоном для создания новых баз данных в SQL Server. Если внести в нее изменения, например, создать набор таблиц, то эти таблицы будут присутствовать во всех создаваемых базах данных.
- *tempdb*. Эта база данных предназначена для временных таблиц и хранимых процедур, создаваемых пользователями и самим SQL Server. Эта база данных создается заново при каждом запуске SQL Server.

### Создание пользовательских баз данных

**База данных** представляет собой группу файлов, хранящихся на жестком диске. Эти файлы могут относиться к трем типам: файлы с первичными данными, файлы с вторичными данными и файлы журнала транзакций. Любая база данных SQL Server **содержит, по крайней мере, два файла: первичный файл данных (с расширением .mdf) и файл журнала транзакций (с расширением .ldf)**. Существует два способа их создания:

- графически с помощью SQL Server Management Studio
- посредством кода Transact-SQL

В данной лабораторной работе рассмотрим способ создания базы данных с помощью SQL Server Management Studio.

## 1.2 СОЗДАНИЕ БАЗЫ ДАННЫХ В SQL SERVER MANAGEMENT STUDIO (SSMS)

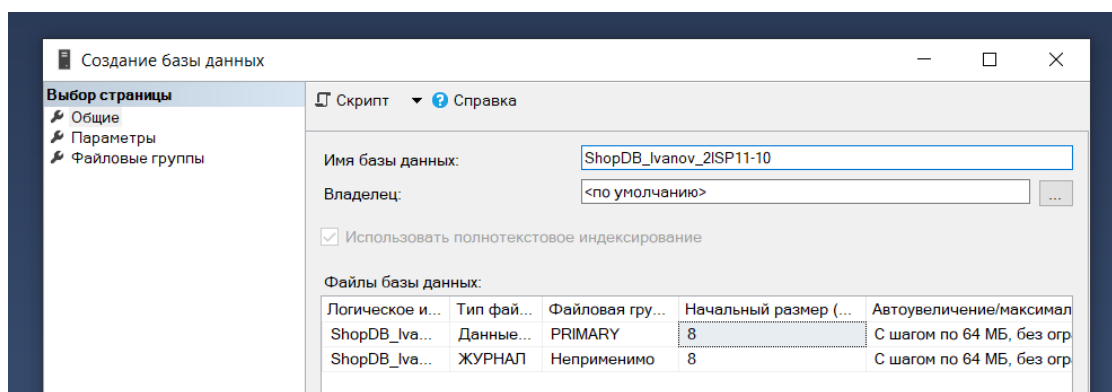
Использование данной утилиты является самым простым способом создания базы данных. Создадим базу данных **CarServis (Автосервис)**, которую позже заполним таблицами, представлениями и другими объектами, предназначенными для отдела продаж.

1. В окне «Обозреватель объектов» найдите и раскройте папку «Базы данных». Щелкните на ней правой кнопкой мыши и выберите команду «Создать базу данных...».
2. В открывшемся диалоговом окне «Создание базы данных» на странице «Общие» введите следующую информацию:

**Имя базы данных:** CarServis\_XX\_YY (где XX – ваша фамилия, YY – номер группы)  
**Владелец:** по умолчанию.

В таблице «Файлы базы данных» измените путь к файлам данных и журнала на ваш каталог.

Для всех остальных параметров оставьте значения по умолчанию.

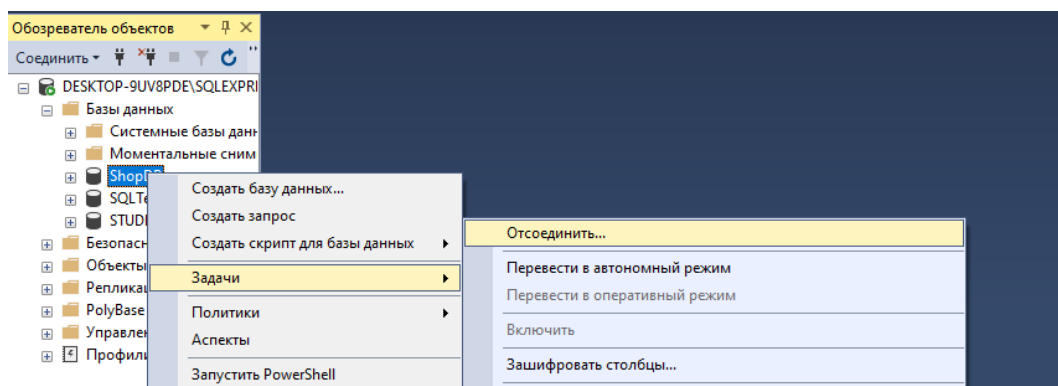


3. Для создания базы данных щелкните «ОК». Вы должны увидеть свою новую базу данных в окне «Обозреватель объектов».

## 1.3 ОТСОЕДИНЕНИЕ/ПРИСОЕДИНЕНИЕ БАЗЫ ДАННЫХ

Для переноса базы данных на другой сервер необходимо отсоединить ее от текущего сервера.

Для этого в контекстном меню базы данных **CarServis** выберите команду «Задачи - Отсоединить...». В диалоговом окне «Отсоединение базы данных» нажмите кнопку «ОК» и убедитесь, что CarServis исчезла из списка баз данных в дереве обозревателя объектов. Теперь файлы базы данных могут быть перенесены на другой сервер.



Файлы базы данных хранятся в системной папке:

**C:\Program Files\Microsoft SQL Server\MSSQL15.SQLEXPRESS\MSSQL\DATA**

Для присоединения базы данных к серверу выберите в контекстном меню узла «Базы данных» команду «Присоединить...». В диалоговом окне «Присоединение базы данных» с помощью кнопки «Добавить...» выберите созданный на предыдущих этапах файл **CarServis.mdf** (ldf файл будет определен системой автоматически) и нажмите кнопку «ОК». База данных **CarServis** должна появиться в списке дерева обозревателя объектов.

## 2. СОЗДАНИЕ ТАБЛИЦ И ОГРАНИЧЕНИЙ

*Таблицы* представляют собой объекты базы данных, используемые непосредственно для хранения всех данных. Одним из самых главных правил организации баз данных является то, что в одной таблице должны храниться данные лишь об одном конкретном типе сущности (например, клиенты, товары, заказы и т. п.).

Данные в таблицах организованы по полям и записям. Поля (или столбцы таблицы) содержат определенный тип информации, например, фамилию, адрес, телефонный номер. Запись (или строка таблицы) - группа связанных полей, содержащих информацию об отдельном экземпляре сущности.

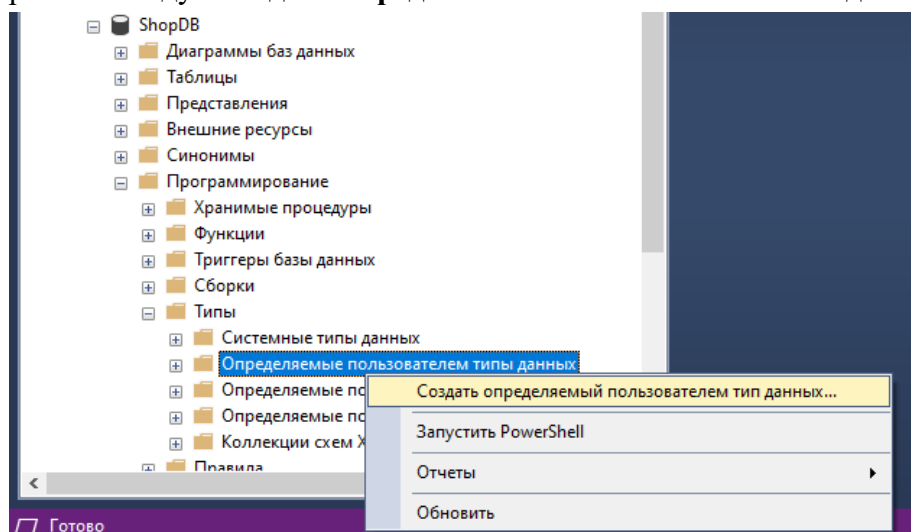
Любое поле таблицы характеризуется как минимум тремя обязательными свойствами:

- *Имя столбца.* Реализует способ обращения к конкретному полю в таблице. Рекомендуется всегда присваивать полям смысловые имена.
- *Тип данных.* Определяет, информация какого типа может храниться в данном поле.
- *Разрешить значения null.* Определяет, допустимо ли для данного поля отсутствие фактических данных, для обозначения которого используется так называемый маркер пустого значения null.

### 2.1 СОЗДАНИЕ ПОЛЬЗОВАТЕЛЬСКИХ ТИПОВ ДАННЫХ

SQL Server позволяет на основе системных типов данных создавать пользовательские типы со всеми предварительно заданными параметрами, включая все ограничения и умолчания. В качестве примера **создадим тип данных PathFile**, который будет использоваться для хранения ссылки на файлы (фото, медиа-файлы, документы и т.д.). Для его создания воспользуемся графическим интерфейсом утилиты Management Studio.

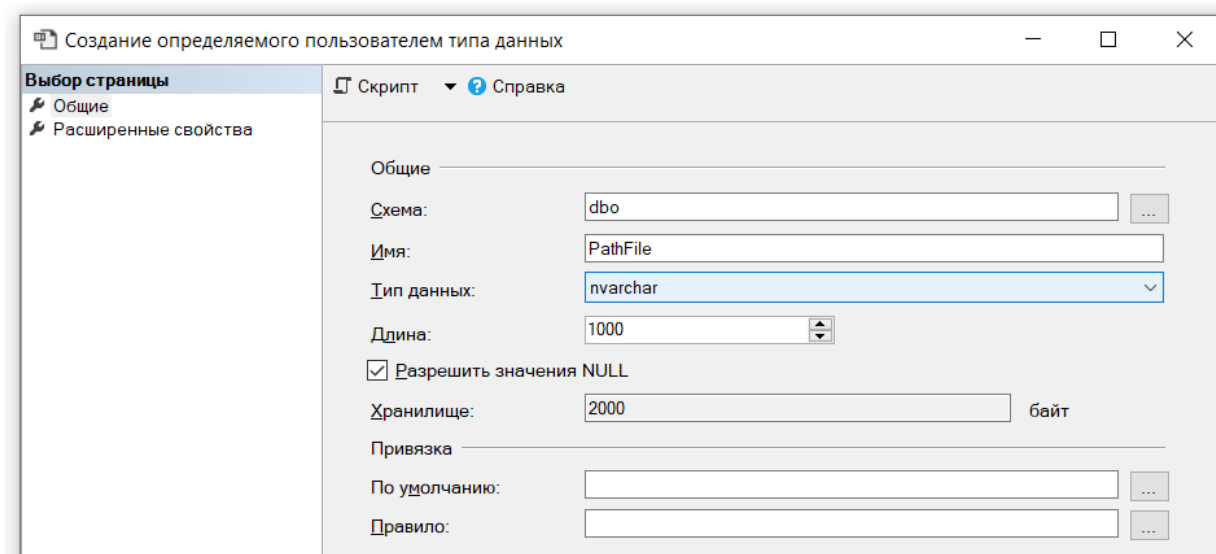
1. В дереве обозревателя объектов раскройте папки «Базы данных – CarServis – Программирование - Типы». В контекстном меню узла «Определяемые пользователем типы данных» выберите команду «Создать определяемый пользователем тип данных».



2. В появившемся окне в текстовом поле «Имя» введите **PathFile**. В раскрывающемся списке «Тип данных» выберите **nvarchar**. В качестве длины введите **1000**. Отметьте параметр

«Разрешить значения null», чтобы иметь возможность не указывать ссылку на файл при заполнении таблицы.

3. В секции «Привязки» оставьте пустые значения и щелкните на кнопке Ок. Созданный пользовательский тип данных должен появиться в дереве обозревателя объектов.



## 2.1 СОЗДАНИЕ ТАБЛИЦ

Прежде чем начать проектирование нашей базы данных, познакомимся с **описанием предметной области**.

*Вашей задачей является разработка системы для компании, которая оказывает услуги клиентам и продаёт определенные товары.*

### *Подсистема работы с клиентами*

*Подсистема работы с клиентами включает в себя возможность добавления новых клиентов, отслеживание их посещений, а также контроль их бонусной программы.*

*Запись о клиенте содержит следующие данные: фамилию, имя, отчество, дату рождения, телефон, электронную почту, пол, дату первого посещения (регистрации), фотографию клиента.*

*В связи с тем, что у компании большое количество клиентов, то их удобно собирать с помощью определенных ярлыков (тегов), которые позволят очень удобно пометать клиентов (например, новый, постоянный, проблемный, горячий). Теги помимо названия должны иметь еще и определенный цвет.*

*Посещения клиента в рамках оказания услуг должны обязательно фиксироваться администраторами.*

### *Подсистема работы с услугами*

*Дополнительная подсистема позволяет клиентам изучать услуги, которые предоставляет компания, а также знакомиться с доступными акциями. Кроме этого некоторые услуги могут содержать подробный список фотографий-примеров предоставления данной услуги.*

Запись об услуге содержит следующие данные: наименование, подробное описание, стоимость, скидку, продолжительность, основную фотографии услуги и множество дополнительных.

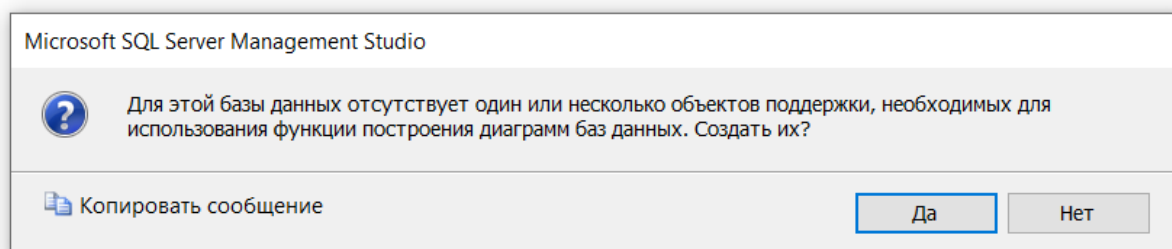
В рамках записи клиентов на услуги необходимо указывать клиента и услугу соответственно, а также время/дату начала и комментарий при наличии.

Приступаем к непосредственному созданию объектов базы данных.

Наиболее быстрый и удобный способ создания таблиц базы данных осуществляется с применением диаграммы базы данных.

Правой кнопкой мыши кликните по объекту «Диаграммы базы данных» и выберите «Создать диаграмму базы данных».

Вам будет отображено представленное ниже сообщение, так как на данный момент в нашей базе данных нет ни одной таблицы. **Нажмите «Да».**



Для создания первой таблицы кликните правой кнопкой мыши по пустому листу диаграммы и выберите «Создать таблицу». Назовите ее «**Client**».

**Важно! При именовании объектов базы данных используйте имена только на английском языке, избегайте сокращений, НЕ используйте знаки подчеркивания, дефисы, пробелы и другие символы, не являющиеся буквенно-цифровыми. Соблюдайте стиль CamelCase. Наименования объектов должны быть в единственном числе.**

Создайте атрибуты таблицы Client в соответствии с приведенным ниже примером.

Client			
	Имя столбца	Тип данных	Разрешить знач...
?	ID	int	<input type="checkbox"/>
	FirstName	nvarchar(50)	<input type="checkbox"/>
	LastName	nvarchar(50)	<input type="checkbox"/>
	Patronymic	nvarchar(50)	<input checked="" type="checkbox"/>
	Birthday	date	<input checked="" type="checkbox"/>
	RegistrationDate	datetime	<input type="checkbox"/>
	Email	nvarchar(255)	<input checked="" type="checkbox"/>
	Phone	nvarchar(20)	<input type="checkbox"/>
	GenderCode	nchar(1)	<input type="checkbox"/>
	PhotoPath	PathFile:nvarchar(1000)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>



Основными инструментами обеспечения целостности сущностей являются первичные ключи и ограничения уникальности.

**Первичный ключ** используется для обеспечения гарантии уникальности каждой записи в таблице. Он состоит из одного (простой ключ) или нескольких (составной ключ) столбцов с гарантированно уникальными значениями для каждой записи таблицы. Если пользователь попытается ввести в поля первичного ключа дублирующее значение будет сгенерирована ошибка и модификация данных будет отменена.

В качестве примера создадим первичный ключ для таблицы **Client**. Для этого правой кнопкой мыши кликните по атрибуту ID и выберите «Задать первичный ключ».

**Свойство «Идентификатор» (Identity)**, обычно используемое совместно с типом данных int, предназначено для автоматического приращения значения на единицу при добавлении каждой новой записи. К примеру, клиент, добавленный в таблицу первым, будет иметь значение идентификатора 1, вторым – 2, третьим – 3, и т.д.

Для автоматического заполнения поля ID кликните правой кнопкой мыши по полю ID и выберите «Свойства». Далее в поле «Спецификация идентификатора» выберите «Да».

Client			
	Имя столбца	Тип данных	Разрешить знач...
PK	ID	int	<input type="checkbox"/>
	FirstName	nvarchar(50)	<input type="checkbox"/>
	LastName	nvarchar(50)	<input type="checkbox"/>
	Patronymic	nvarchar(50)	<input checked="" type="checkbox"/>
	Birthday	date	<input checked="" type="checkbox"/>
	RegistrationDate	datetime	<input type="checkbox"/>
	Email	nvarchar(255)	<input checked="" type="checkbox"/>
	Phone	nvarchar(20)	<input type="checkbox"/>
	GenderCode	nchar(1)	<input type="checkbox"/>
	PhotoPath	PathFile:nvarchar(1000)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Конструктор баз данных	
RowGuid	Нет
Детерминированный	Да
Имеет подписчик, отличный от подпис...	Нет
Индексируемый	Да
Набор столбцов	Нет
Не для репликации	Нет
Описание	
Опубликован слиянием	Нет
Опубликован через службы DTS	Нет
Параметры сортировки	<база данных по
Размер	4
Разряженный	Нет
Реплицировано	Нет
Сжатый тип данных	int
Спецификация вычисляемого столбца	
Спецификация идентификатора	
(Идентификатор)	Да
Начальное значение идентификатора	1
Шаг приращения идентификатора	1
Спецификация полнотекстового столбца	
	Нет

Настоятельно рекомендуется **сохранять диаграмму базы данных после создания каждой таблицы**. Сделать это можно при помощи кнопки «Сохранить» или комбинации клавиш **Ctrl+S**.

## ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ:

В соответствие с нижеприведенным описанием создайте оставшиеся шесть таблиц для описанной предметной области.

Tag			
Имя столбца	Сжатый тип	Допускает значения NULL	Идентификатор
ID	int	Нет	<input checked="" type="checkbox"/>
Title	nvarchar(30)	Нет	<input type="checkbox"/>
Color	nchar(7)	Нет	<input type="checkbox"/>

TagOfClient			
Имя столбца	Сжатый тип	Допускает значения NULL	Идентификатор
ClientID	int	Нет	<input type="checkbox"/>
TagID	int	Нет	<input type="checkbox"/>

Gender			
Имя столбца	Сжатый тип	Допускает значения NULL	Идентификатор
Code	nchar(1)	Нет	<input type="checkbox"/>
Name	nvarchar(10)	Да	<input type="checkbox"/>

ServicePhoto			
Имя столбца	Сжатый тип	Допускает значе...	Идентификатор
ID	int	Нет	<input checked="" type="checkbox"/>
ServiceID	int	Нет	<input type="checkbox"/>
PhotoPath	PathFile:nvarchar(1000)	Нет	<input type="checkbox"/>

Service			
Имя столбца	Сжатый тип	Допускает знач...	Идентификатор
ID	int	Нет	<input checked="" type="checkbox"/>
Title	nvarchar(100)	Нет	<input type="checkbox"/>
Cost	decimal(10, 2)	Нет	<input type="checkbox"/>
DurationInS...	int	Нет	<input type="checkbox"/>
Description	nvarchar(MAX)	Да	<input type="checkbox"/>
Discount	float	Да	<input type="checkbox"/>
MainImageP...	PathFile:nvarchar(1000)	Да	<input type="checkbox"/>

ClientService			
Имя столбца	Сжатый тип	Допускает значения NULL	Идентификатор
ID	int	Нет	<input checked="" type="checkbox"/>
ClientID	int	Нет	<input type="checkbox"/>
ServiceID	int	Нет	<input type="checkbox"/>
StartTime	datetime	Нет	<input type="checkbox"/>
Comment	nvarchar(MAX)	Да	<input type="checkbox"/>

## 2.1 СОЗДАНИЕ ОГРАНИЧЕНИЙ

Перед тем как начать работать с таблицами следует ограничить вводимые в них данные в целях обеспечения так называемой целостности данных, т.е. ограничить возникновение в базе данных некорректных или противоречивых данных вследствие добавления, изменения или удаления какой-либо записи, например, ввод отрицательной цены или количества товара. Существует четыре типа целостности данных: доменная, сущностная, ссылочная и пользовательская (или бизнес-правила). Рассмотрим основные инструменты, предоставляемые в SQL Server для их реализации.

**Обеспечение доменной целостности.** Ограничение диапазона данных, вводимых пользователем в поле. Основными инструментами обеспечения доменной целостности являются ограничения проверки и значения по умолчанию.

### 2.1.1 ИСПОЛЬЗОВАНИЕ ПРОВЕРОЧНЫХ ОГРАНИЧЕНИЙ

Ограничения на проверку используются для ограничения данных, принимаемых полем, даже если они имеют корректный тип. Например, поле **Color** имеет тип nchar(7), т.е. теоретически оно может содержать любые символы. Рассмотрим, как **создать ограничение на проверку, позволяющее вставлять в столбец только значения, начинающиеся на знак # за которыми следуют шесть любых символов (код цвета может содержать как цифры, так и буквы).**

1. В контекстном меню папки «Ограничения» таблицы Tag выберите команду «Создать ограничение».
2. В открывшемся окне «Проверочные ограничения» заполните следующие поля:

Проверочные ограничения

Выбрано Проверочное ограничение:

CK\_Tag

Изменение свойств существующих объектов "проверочное ограничение".

▼ (Общие)	
Выражение	[[Color] like '#_____']
▼ Идентификатор	
(Имя)	CK_Tag
Описание	
▼ Конструктор таблиц	
Включить использовани	Да
Применять для INSERT и	Да
Проверить существующи	Да

Добавить Удалить Закреть

Выражение: `[[Color] like '#_____']` содержит шесть знаков нижнего подчеркивания \_ (позволяющее вставлять в столбец только значения, начинающиеся на знак # за которыми следуют шесть любых символов).

Щелкните на кнопке «Закреть» и закройте конструктор таблиц (он был открыт, когда вы начали создавать ограничение) с сохранением изменений.

Пример: Следующий код создает ограничение, позволяющее вставлять в столбец только значения, начинающиеся на два любых символа, за которыми следуют дефис (-), любое число символов или не следует никаких символов, и завершающиеся целым числом из диапазона от 0 до 9. `Value LIKE '___-%[0-9]'`

## ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ:

1. Создайте ограничения для поле Cost таблицы Service, запрещающие ввод в них отрицательных значений.
2. Создайте ограничение для поля Email таблицы Client, позволяющее вставлять в столбец только значения, начинающиеся на любое число символов, за которыми следует знак @, далее любое число символов, после чего знак точки и заканчивается любым числом символов.

### 2.1.2 ИСПОЛЬЗОВАНИЕ ЗНАЧЕНИЙ ПО УМОЛЧАНИЮ

Установка для полей значений по умолчанию — это отличный способ избавить пользователя от излишней работы, если значения этих полей во всех записях, как правило, принимают одни и те же значения. Так в таблице клиентов **Client** вполне логично определить по умолчанию значение поля **RegistrationDate** (дата регистрации клиента) в виде текущей даты. В этом случае при добавлении записи о новом клиенте в случае

пропуска этого поля оно будет автоматически заполняться значением системной даты. Для создания такого свойства выполните следующие шаги:

1. Правой кнопкой мыши кликните по полю **RegistrationDate**, выберите «Свойства».
2. В свойстве столбца «Значение по умолчанию или привязка» введите **GETDATE()**. Эта функция T-SQL возвращает текущую системную дату.
3. Щелкните на кнопке Сохранить.

Client *				
Имя столбца	Сжатый тип	Допускает значения NULL	Идентификатор	
ID	int	Нет	<input checked="" type="checkbox"/>	
FirstName	nvarchar(50)	Нет	<input type="checkbox"/>	
LastName	nvarchar(50)	Нет	<input type="checkbox"/>	
Patronymic	nvarchar(50)	Да	<input type="checkbox"/>	
Birthday	date	Да	<input type="checkbox"/>	
RegistrationDate	datetime	Нет	<input type="checkbox"/>	
Email	nvarchar(255)	Да	<input type="checkbox"/>	
Phone	nvarchar(20)	Нет	<input type="checkbox"/>	
GenderCode	nchar(1)	Нет	<input type="checkbox"/>	
PhotoPath	PathFile:nv...	Да	<input type="checkbox"/>	


<b>(Общие)</b>	
(Имя)	RegistrationDate
Значение по умолчанию или привязка	GETDATE()
Разрешить значения NULL	Нет
Тип данных	datetime
<b>Конструктор баз данных</b>	
RowGuid	Нет
Детерминированный	Да
Имеет подписчик, отличный от подписи	Нет
Индексируемый	Да
Набор столбцов	Нет
Не для репликации	Нет
Описание	

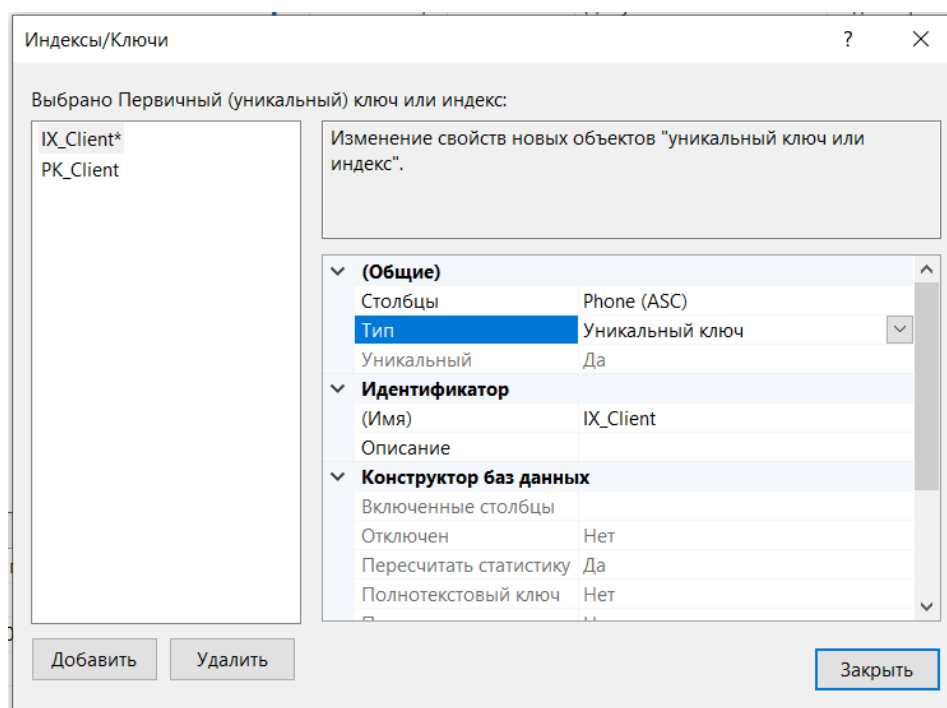
## ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ:

Установите для поля **Discount** (размер скидки) таблицы Service в качестве значения по умолчанию ноль.

## 2.1.3 ИСПОЛЬЗОВАНИЕ ОГРАНИЧЕНИЙ НА УНИКАЛЬНОСТЬ

Между ограничениями первичного ключа и ограничениями на уникальность существует два отличия. Первое состоит в том, что первичные ключи используются вместе с внешними ключами для обеспечения целостности ссылок (рассматривается в следующем разделе). Второе отличие заключается в том, что ограничения на уникальность позволяют вставлять в его поля пустые значения (null), чего нельзя делать с первичными ключами. Во всем остальном они служат одной цели – обеспечить уникальность данных, вставляемых в поле. Ограничение на уникальность следует использовать в тех случаях, когда нужно гарантировать, что дублирующие значения не будут добавляться в поле, не являющееся частью первичного ключа, в частности, все потенциальные ключи должны быть организованы в виде ограничений уникальности. Хорошим примером такого поля, требующего ограничение на уникальность, является поле ИНН или серия и номер паспорта, поскольку эти поля должны быть уникальными у каждого человека. Такого идеального кандидата на роль **уникального ограничения** в нашей таблице **Client** нет. Поэтому создадим его по полю **Phone**, которое также повторяться у разных клиентов не должно.

1. Правой кнопкой мыши кликните по полю Phone, выберите «Индексы и ключи» .
2. В открывшемся окне «Индексы и ключи» щелкните кнопку «Добавить» и введите следующие параметры для нового уникального ключа:



### ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ:

Аналогичным образом создайте ограничение уникальности по полю Title таблицы Service, чтобы обеспечить отсутствие в справочнике услуг с одинаковыми названиями.

## 2.1.4 СОЗДАНИЕ ВНЕШНИХ КЛЮЧЕЙ (СВЯЗЕЙ МЕЖДУ ТАБЛИЦАМИ)

Используя диаграмму базы данных ограничения внешнего ключа можно создавать значительно быстрее: лишь перетаскивая поля из одной таблицы в другую. В качестве примера создадим внешний ключ в таблице Client по полю GenderCode для связи с таблицей Gender:

1. Выделите в таблице Gender поле GenderCode и, не отпуская кнопку мыши, перетащите его на поле GenderCode таблицы Client.
2. В диалоговых окнах «Таблицы и столбцы» и «Связь по внешнему ключу» примите настройки по умолчанию.

### ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ:

Аналогичным образом создайте связи между оставшимися таблицами.



## 2.1.5 ОБЕСПЕЧЕНИЕ ЦЕЛОСТНОСТИ ССЫЛОК

Суть обеспечения целостности ссылок очевидна из названия: данные в одной таблице, ссылающиеся на данные из другой таблицы, защищены от некорректного обновления. В терминологии SQL Server это называется *декларативной ссылочной целостностью* и достигается путем связывания первичного ключа одной из таблиц с внешним ключом другой таблицы (создается так называемое ограничение внешнего ключа).

**Внешний ключ** используется в комбинации с первичным для связывания двух таблиц по общему столбцу (столбцам). Кроме того, при отсутствии каскадирования (рассматривается в следующем разделе) невозможно удалить запись из таблицы первичного ключа при наличии связанных с ней записей в таблице внешнего ключа.

## Использование каскадной ссылочной целостности

При наличии ограничения внешнего ключа с параметрами по умолчанию вы не можете удалить запись или изменить значение первичного ключа главной таблицы в случае наличия связанных записей в подчиненной таблице (в которой организовано ограничение внешнего ключа). Однако это поведение можно изменить, используя *каскадную ссылочную целостность*.

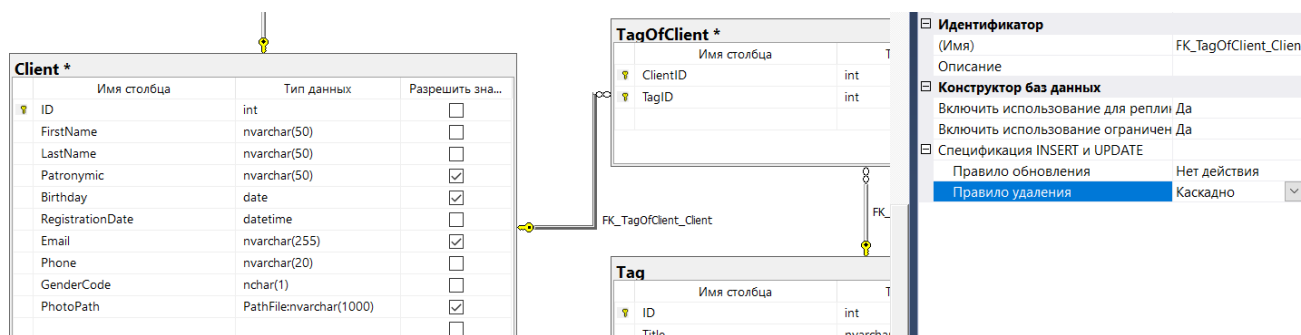
Настроить правила каскадирования можно при создании ограничения внешнего ключа в окне «Связи по внешнему ключу» изменяя значения параметров «Правило обновления» и «Правило удаления» блока «Спецификация INSERT и UPDATE». Оба этих параметра могут содержать четыре значения, описанные в следующей таблице.

Настройка	Правило удаления	Правило обновления
Нет действия	<b>Невозможно удалить</b> в главной таблице строку, на которую есть ссылки в подчиненной	<b>Невозможно обновить</b> значения полей первичного ключа главной таблицы при наличии связанных записей в подчиненной
Каскадное	При удалении строки в главной таблице <b>все связанные строки в подчиненной также будут удалены</b>	При обновлении значений полей первичного ключа главной таблицы соответствующим образом <b>будут изменены и их значения во всех связанных строках подчиненной таблицы</b>
Присвоить Null	При удалении строки в главной таблице во всех связанных строках подчиненной полям вторичного ключа <b>будет присвоено значение Null</b>	При обновлении значений полей первичного ключа главной таблицы во всех связанных строках подчиненной таблицы полям вторичного ключа <b>будет присвоено значение Null</b>
Присвоить значение по умолчанию	При удалении строки в главной таблице во всех связанных строках подчиненной полям вторичного ключа <b>будут присвоены значения по умолчанию</b>	При обновлении значений полей первичного ключа главной таблицы во всех связанных строках подчиненной таблицы полям вторичного ключа <b>будут присвоены значения по умолчанию</b>

Создадим правило каскадного удаления для связи таблиц **Client** и **TagOfClient**, которое будет обозначать: при удалении сведений о клиенте из таблицы **Client** каскадно будут удалены все записи о тегах этого клиента из таблицы **TagOfClient**.

Для этого необходимо кликнуть правой кнопкой мыши по связи между этими таблицами и выбрать «Свойства».

В появившемся окне свойств раскрыть **спецификацию INSERT и UPDATE** и в меню Правила удаления выбрать «Каскадно».



## ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ:

Создайте ограничение внешнего ключа для связи таблиц **Service** и **ServicePhoto** по полю **ServiceID**. При этом настройте правило каскадного удаления, установив в качестве параметра «Спецификация INSERT и UPDATE» **Правило удаления – значение «Каскадно»**, что приведет к автоматическому удалению всех фотографий услуги при удалении самой услуги.

## ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ:

После настройки всех ограничений можно заполнить таблицы данными. Для этого в контекстном меню таблицы выберите команду «Изменить первые 200 строк» и появившейся в рабочей области вкладке введите новые записи, заполняя все необходимые столбцы. В процессе внесения данных проверьте работоспособность всех созданных ранее ограничений (**в каждой таблице заполните не менее 2 строк**):

- Ограничений проверки: попробуйте ввести в поле **Email** таблицы **Client** значения, не соответствующие условию, а в поля **Discount** таблицы **Service** – отрицательные.
- Значений по умолчанию: убедитесь, что при пропуске поля **RegistrationDate** таблицы **Client** для него устанавливается значения по умолчанию в виде текущей системной даты.
- Ограничений первичного и уникального ключа: попробуйте ввести в таблицы записи с дублирующими значениями первичного или уникального ключа.
- Ограничений внешнего ключа: попробуйте ввести несогласованные данные в связанные таблицы, например, заказ для несуществующего клиента или удалить запись из любой главной таблицы при наличии связанных записей в подчиненной при отсутствии правил каскадирования.
- Правил каскадирования: убедитесь, что при удалении записи из таблицы **Service** все связанные записи из таблицы **ServicePhoto** удаляются автоматически.