

РЯЗАНСКИЙ СТАНКОСТРОИТЕЛЬНЫЙ КОЛЛЕДЖ
ФГБОУ ВО "РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ
РАДИОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ В.Ф. УТКИНА"

Основы проектирования баз данных

Пр. раб. №1 – Создание базы
данных, таблиц, связей

ОПЕРАТОРЫ CREATE, DROP



РОДИН Е.Н.
2022 Г.

СОДЕРЖАНИЕ

ЦЕЛИ РАБОТЫ:	3
ТЕОРЕТИЧЕСКАЯ ЧАСТЬ:	3
1. ЗНАКОМСТВО С MICROSOFT SQL SERVER MANAGEMENT STUDIO.....	3
2. РАБОТА С БАЗАМИ ДАННЫХ	7
2.1 Создание БД.....	8
2.1.1 Создание БД средствами SQL Server Management Studio	8
2.1.2 Создание БД SQL-запросом.....	10
2.2 Удаление БД	11
2.2.1 Удаление БД средствами SQL Server Management Studio.....	11
2.2.2 Удаление БД SQL-запросом	12
3. РАБОТА С ТАБЛИЦАМИ.....	12
3.1 Создание таблиц	13
3.1.1 Создание таблиц БД средствами SQL Server Management Studio ...	13
3.1.2 Создание таблиц БД SQL-запросом	16
3.2 Удаление таблиц.....	21
3.2.1 Удаление таблиц БД средствами SQL Server Management Studio ...	21
3.2.2 Удаление таблиц БД SQL-запросом	21
4. РАБОТА СО СВЯЗЯМИ	22
4.1 Создание связей.....	22
4.1.1 Создание связей средствами SQL Server Management Studio	23
4.1.2 Создание связей SQL-запросом.....	27
4.2 Удаление связей	29
4.2.1 Удаление связей средствами SQL Server Management Studio	29
4.2.2 Удаление связей SQL-запросом.....	30
5. ПЕРЕНОС БАЗЫ ДАННЫХ	32
5.1 Подключение и отключение БД	32
5.2 Создание скрипта	34
5.3 Резервное копирование	37
ЗАДАНИЕ:	41
ВАРИАНТЫ:	42
КОНТРОЛЬНЫЕ ВОПРОСЫ:	53
ПОРЯДОК ОЦЕНИВАНИЯ:	54

Цели работы:

- приобрести начальные навыки работы с MS SQL Server Management Studio;
- научиться создавать базы данных, таблицы, связи между таблицами средствами MS SQL Server Management Studio, а также средствами языка T-SQL;
- изучить возможности операторов CREATE и DROP.

Теоретическая часть:

1. Знакомство с Microsoft SQL Server Management Studio

Основной утилитой MS SQL Server 2019 является приложение SQL Server Management Studio (рисунок 1).

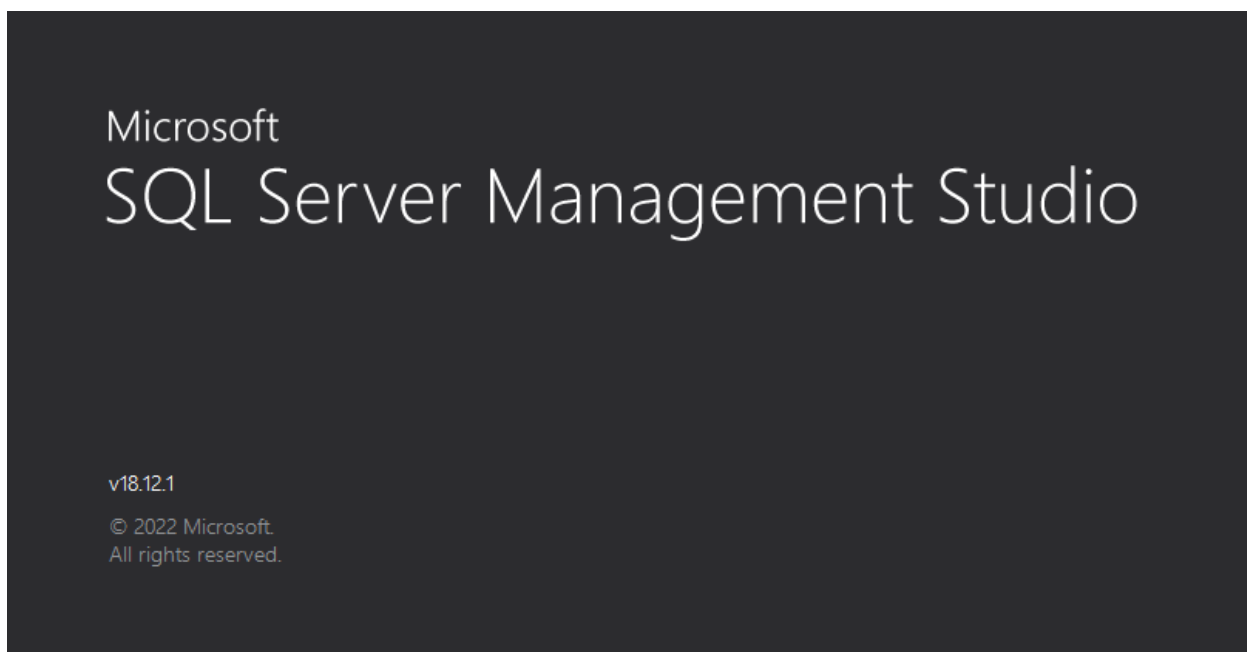


Рисунок 1 – Запуск SQL Server Management Studio

К возможностям программы можно отнести:

- выполнение основных действий над базами данных с помощью визуальных средств;
- отладка и выполнение SQL-запросов к БД;
- управление и настройка SQL Server.

В рамках выполнения практической работы предлагается познакомиться с визуальными возможностями SQL Server Management Studio по созданию БД, таблиц БД, диаграмм, а также изучить работу операторов CREATE и DROP.

После запуска SQL Server Management Studio на экране открывается окно подключения к серверу, показанное на рисунке 2.

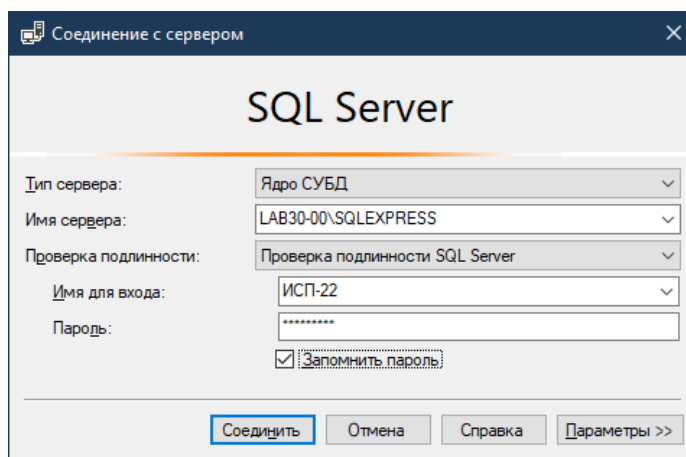


Рисунок 2 – Соединение с сервером

Для продолжения работы следует выбрать в выпадающем списке *Тип сервера* службу сервера, с которой вы хотите работать. Далее в выпадающем списке *Имя сервера* выбрать имя сервера БД, с которым осуществляется соединение. Если такого имени сервера нет в списке, то его следует набрать с клавиатуры. Следует учитывать, что имя сервера указывается в виде <имя_компьютера>\<имя_экземпляра_сервера>. Затем необходимо выбрать тип аутентификации в списке *Проверка подлинности*, ввести имя пользователя и пароль и нажать кнопку *Соединить*.

Пример:

На рисунке 2 происходит подключение к ядру базы данных экземпляра SQL Server с именем SQLEXPRESS, находящегося на компьютере с именем LAB30-00. При этом выбирается проверка подлинности SQL Server. Соединение происходит под учетной записью ИСП-22.

При успешной аутентификации появляется рабочее окно SQL Server Management Studio, показанное на рисунке 3.

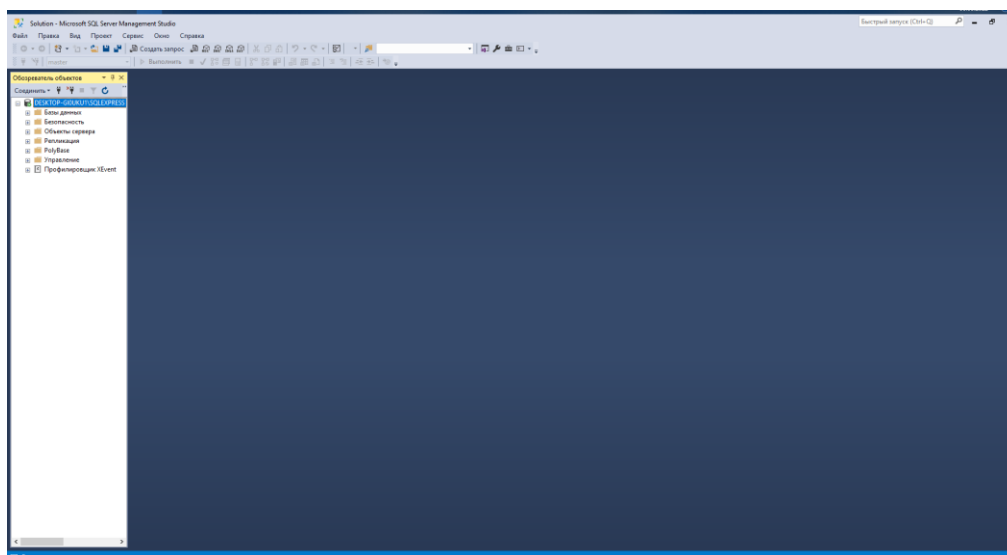


Рисунок 3 – Рабочая область SQL Server Management Studio

В левой части окна SQL Server Management Studio расположена панель *Обозреватель объектов*. Эта панель используется для создания и администрирования объектов баз данных SQL Server 2019. При закрытии панели её можно вызвать через пункт главного меню *Вид* или нажав клавишу *F8*.

Обозреватель объектов включает сведения по всем серверам, к которым он подключен. С помощью *Обозревателя объектов* можно установить или завершить соединение с выбранным сервером, а также зарегистрировать новый сервер. Выполнить данные действия можно с помощью контекстного меню соответствующего сервера.

Таким образом, с помощью *Обозревателя объектов* можно одновременно управлять несколькими серверами.

Для того, чтобы управлять всеми решениями и проектами, Management Studio создан *Обозреватель решений* (рисунок 4).

Решение в SQL Server Management Studio – это набор из одного или нескольких взаимосвязанных проектов.

Проекты в SQL Server Management Studio – это контейнеры для организации взаимосвязанных файлов, например, файлов с SQL инструкциями, которые используются при разработке того или иного функционала в базах данных.

Открыть *Обозреватель решений* можно из меню *Вид* или нажав сочетание клавиш *Ctrl+Alt+L*.

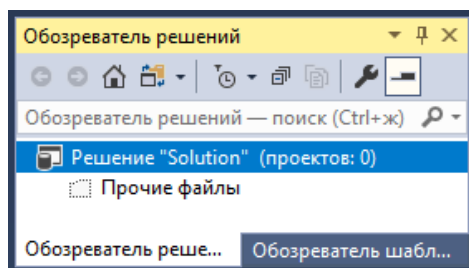


Рисунок 4 – Обозреватель решений

Таким образом, с помощью *Обозревателя решений* мы можем все свои SQL скрипты сгруппировать в проект, тем самым систематизировать все файлы и иметь к ним более удобный доступ.

Для создания проекта необходимо выполнить пункт контекстного меню соответствующего решения – *Добавить / Создать проект* (рисунок 5).

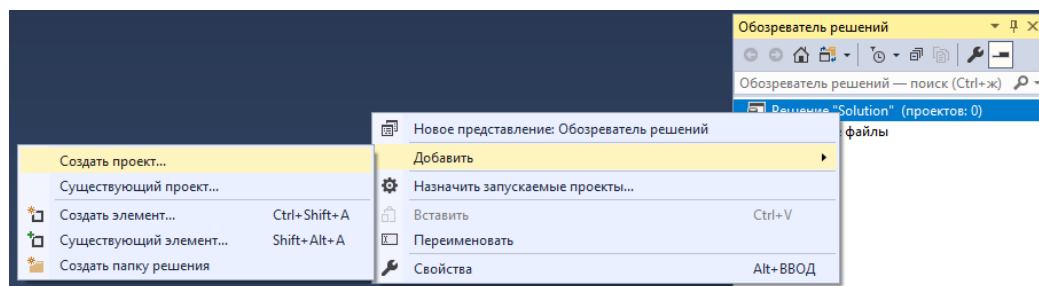


Рисунок 5 – Создание проекта

Далее необходимо выбрать тип проекта – *Скрипты SQL Server* (рисунок 6).

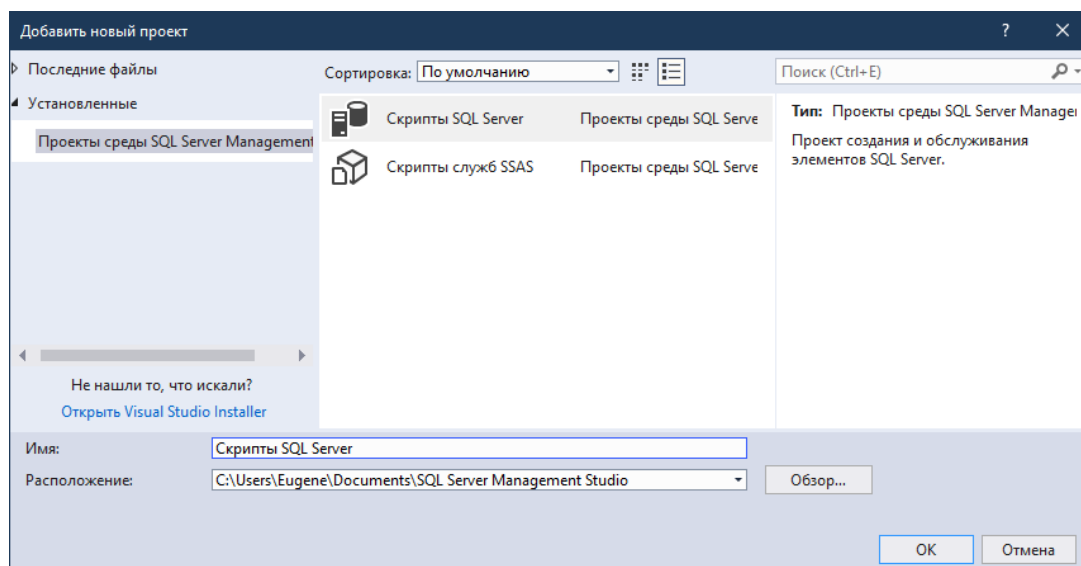


Рисунок 6 – Выбор типа проекта

Запросы в проект можно добавлять 2-мя способами:

Способ 1:

Для узла *Запросы* выполнить пункт контекстного меню *Создать запрос* (рисунок 7).

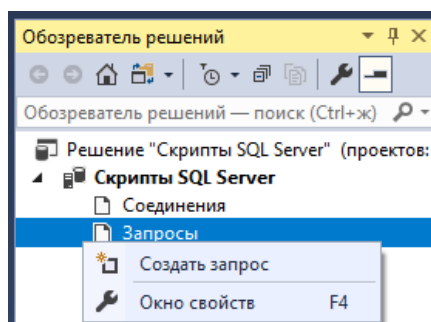


Рисунок 7 – Добавление запросов через узел «Запросы»

Способ 2:

Для проекта выполнить пункт контекстного меню – *Добавить / Создать элемент* или нажать сочетание клавиш *Ctrl+Shift+A* (рисунок 8) и выбрать шаблон запроса (рисунок 9).

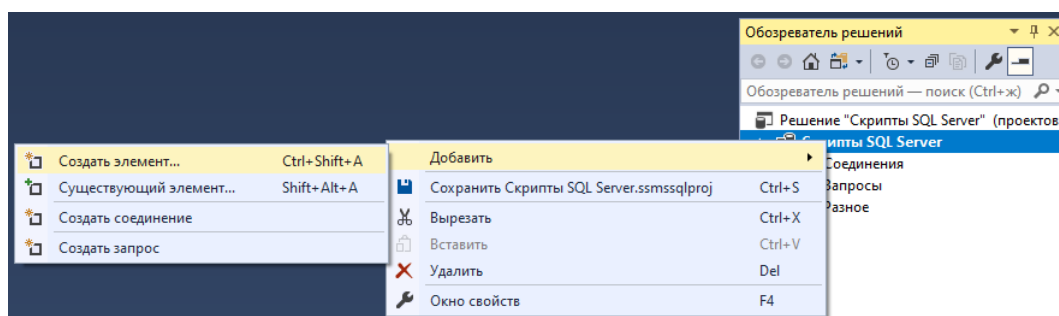


Рисунок 8 – Добавление запроса в проект

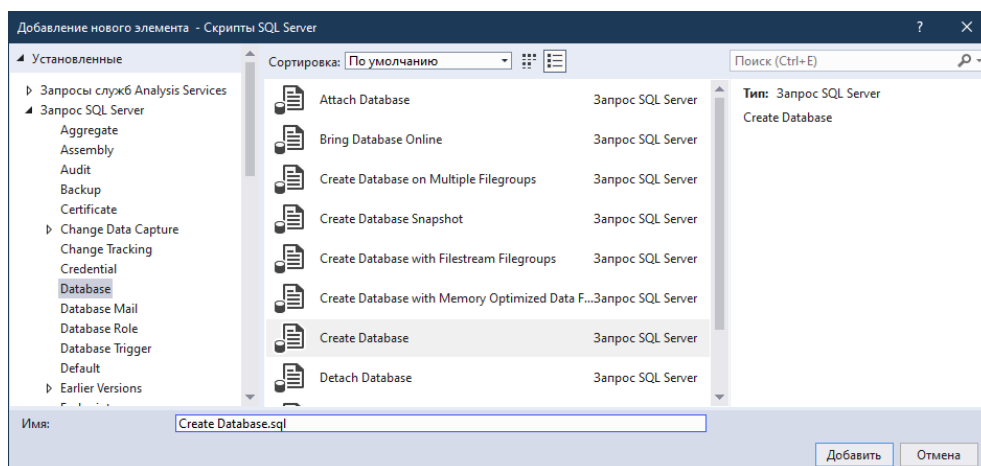



Рисунок 9 – Создание запроса на основе шаблона

Если же просто требуется написать запрос, не добавляя его в проект, можно воспользоваться соответствующей командой  **Создать запрос** в меню.

При выполнении запросов учитывайте контекст выполнения (по умолчанию – master), проверить или изменить его можно с помощью соответствующего выпадающего списка (рисунок 10).

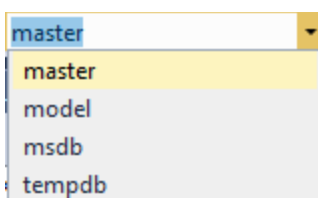


Рисунок 10 – Контекст выполнения запроса

Контекст выполнения также можно изменить в начале запроса командой **USE**, после которой указать имя БД, в которой будет выполняться запрос.

2. Работа с базами данных

SQL Server сохраняет информацию о создаваемых БД на диске компьютера-сервера в соответствующих файлах с расширениями *.mdf (*.ndf) и *.ldf.

Файл *.mdf (первичный файл – Master Data File) представляет, собственно, БД. В нем хранится информация обо всех объектах БД, включая таблицы, индексы, хранимые процедуры и т.д.

Файлы *.ndf (вторичные файлы – Not Master Data File). Данные типы файлов БД может и не содержать, они создаются дополнительно к первичному файлу.

Файл *.ldf (Log Data File) называется журналом транзакций. В нем хранится информация обо всех изменениях, произведенных над данными в базе, а также обо всех транзакциях, вызвавших эти изменения.

Файлы БД по умолчанию располагаются в каталоге Data, который, в свою очередь, размещается в каталоге, куда установлен сервер. Например, C:\MSSQL\Data. При желании можно расположить файлы БД в любом другом месте.

В Microsoft SQL Server есть возможность объединять файлы данных в файловые группы (файлы журнала транзакций не могут входить в файловые группы). По умолчанию в SQL сервере создана файловая группа PRIMARY. Один файл данных может входить в состав только одной файловой группы.

2.1 Создание БД

Рассмотрим процесс создания новой БД.

Задание:

Пусть требуется создать БД для хранения информации о работе торговой фирмы.

Название БД – «Поставки».

2.1.1 Создание БД средствами SQL Server Management Studio

В дереве *Обозревателя объектов* выберем узел *Базы данных*. Выполним пункт контекстного меню *Создать базу данных* (рисунок 10).

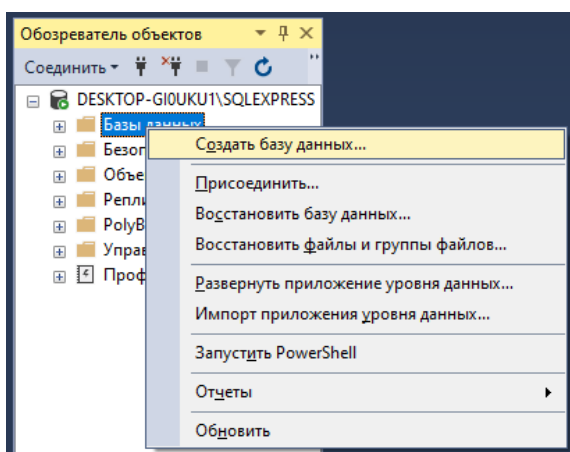


Рисунок 10 – Создание БД средствами интерфейса

В открывшемся окне *Создание базы данных* (рисунок 11) задаются параметры для новой БД: имя, владелец и файлы БД.

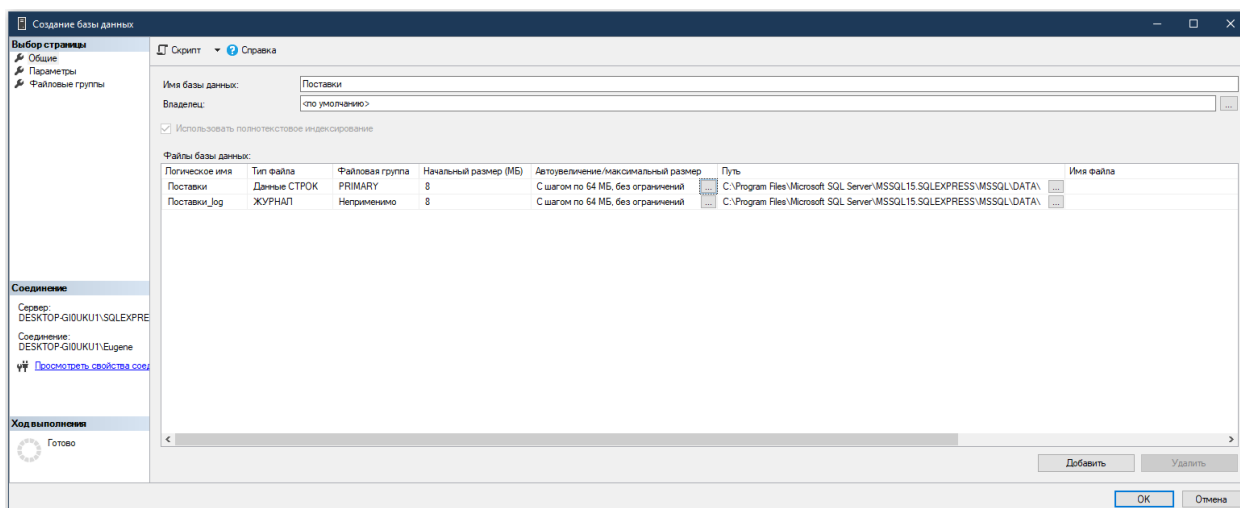



Рисунок 11 – Настройка параметров создаваемой БД

В таблице *Файлы БД* приведены параметры файлов:

- логические имена для файлов данных и для журнала транзакций;
- тип файла. Для файла данных указывается тип *Данные СТРОК*, а для журнала транзакций – *ЖУРНАЛ*;
- первоначальный размер файла в МБ;
- увеличение файла в мегабайтах или процентах. Каждый раз, когда в файле БД заканчивается свободное место, размер этого файла будет увеличен на заданную величину. Для того чтобы задать параметры увеличения БД следует нажать кнопку  и в появившемся окне *Изменение авторасширения* выбрать увеличение файла в процентах или в мегабайтах. Кроме того, в этом же окне можно задать максимальный размер файла неограниченный или ограниченный заданным числом мегабайт (окно приведено на рисунке 12);
- пути к файлам.

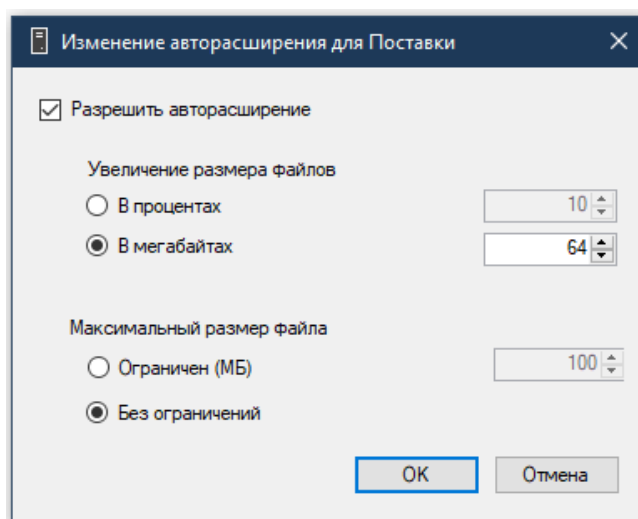


Рисунок 12 – Изменение авторасширения файлов

При желании можно изменить имена файлов, их первоначальный размер и автоматическое приращение, а также путь к файлам. После нажатия кнопки *ОК* новая БД будет создана, а ее объект появится на дереве объектов в *Обозревателе объектов* (рисунок 13).

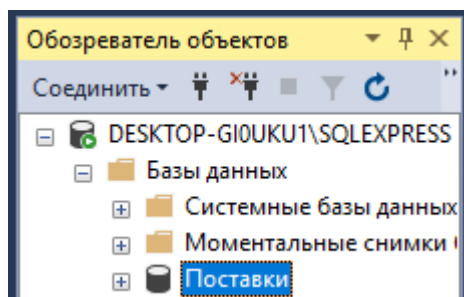



Рисунок 13 – Созданная БД «Поставки»

Если БД не появилась в списке, необходимо обновить список баз данных в *Обозревателе объектов*, нажав на  или на *F5*.

2.1.2 Создание БД SQL-запросом

Выполним действия, показанные на рисунке 7, открыв тем самым редактор запросов.

В SQL Server инструкция **CREATE DATABASE** создает базу данных, используемые для нее файлы и их файловые группы.

Базовый синтаксис:

```
CREATE DATABASE database_name
    [ON [PRIMARY] {file_spec1}, ...]
    [LOG ON {file_spec2}, ...]
```

где:

database_name – имя новой базы данных. Имена баз данных должны быть уникальны внутри экземпляра SQL Server и соответствовать правилам для идентификаторов. Аргумент **database_name** может иметь максимальную длину 128 символов.

Если инструкция **CREATE DATABASE** содержит параметр **ON**, все файлы базы данных указываются явно.

Параметр **PRIMARY** указывает первый (и наиболее важный) файл, который содержит системные таблицы и другую важную внутреннюю информацию о базе данных. Если параметр **PRIMARY** отсутствует, то в качестве первичного файла используется первый файл, указанный в спецификации.

Параметр **file_spec1** представляет спецификацию файла и сам может содержать дополнительные опции, такие как логическое имя файла, физическое имя и размер.

Опция **LOG ON** (относится к параметру **dbo** – владельцу БД) определяет один или более файлов (**file_spec2**) в качестве физического хранилища журнала транзакций базы данных. Если опция **LOG ON** отсутствует, то журнал транзакций базы данных все равно будет создан, поскольку каждая база данных должна иметь, по крайней мере, один журнал транзакций.

Пример:

```
CREATE DATABASE Поставки
```

либо

```
CREATE DATABASE Поставки
    ON (NAME = supplies_data,
        FILENAME = 'D:\Поставки.mdf',
        SIZE = 10,
        MAXSIZE = 100,
        FILEGROWTH = 5)
    LOG ON (NAME = supplies_log,
        FILENAME = 'D:\Поставки_log.ldf',
        SIZE = 40,
        MAXSIZE = 100,
        FILEGROWTH = 10);
```

Для выполнения запроса необходимо нажать **Выполнить**, после чего появится сообщение о статусе выполнения запроса (рисунок 14).

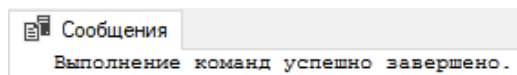


Рисунок 14 – Информационное сообщение о статусе выполнения запроса

Для просмотра списка существующих баз данных на сервере можно использовать следующий SQL-запрос:

Пример:

```
SELECT name
FROM sys.databases
```

В результате будет выведен список имён баз данных (рисунок 15).

	name
1	master
2	model
3	msdb
4	tempdb
5	Поставки

Рисунок 15 – Список БД, существующих на сервере

2.2 Удаление БД

Рассмотрим процесс удаления существующей БД.

2.2.1 Удаление БД средствами SQL Server Management Studio

В дереве *Обозревателя объектов* в узле *Базы данных* выберем нужную для удаления БД, выполним пункт контекстного меню *Удалить*.

Затем в открывшемся окне *Удаление объекта* необходимо поставить галочку на пункте *Закрывать существующие соединения* и нажать *ОК* (рисунок 16).

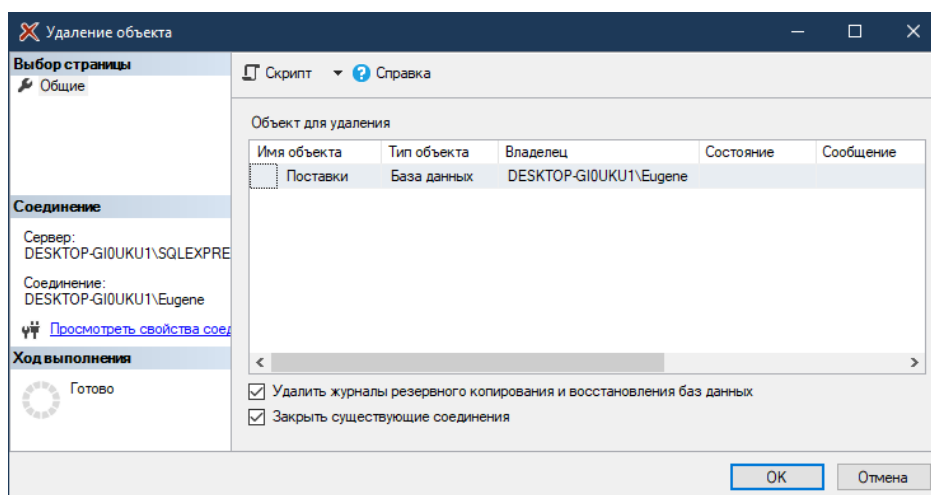


Рисунок 16 – Удаление БД средствами интерфейса

После проделанных действия БД будет удалена и пропадёт из дерева объектов.

2.2.2 Удаление БД SQL-запросом

В SQL Server инструкция **DROP DATABASE** удаляет одну или несколько пользовательских баз данных или моментальных снимков базы данных из экземпляра SQL Server.

Базовый синтаксис:

```
DROP DATABASE
[IF EXISTS] {database_name | database_snapshot_name} [, ...]
```

где:

IF EXISTS – условное удаление базы данных только в том случае, если она уже существует. Если попытаться удалить несуществующую базу данных без указания IF EXISTS, SQL Server выдаст сообщение об ошибке.

database_name – имя удаляемой базы данных.

database_snapshot_name – имя удаляемого моментального снимка базы данных.

Перед удалением базы данных необходимо убедиться в соблюдении следующих важных моментов:

- **DROP DATABASE** удаляет базу данных, а также файлы на физическом диске, используемые базой данных. Поэтому у вас должна быть резервная копия базы данных на случай, если вы захотите восстановить ее в будущем.
- вы не можете удалить базу данных, которая используется в данный момент.

Пример:

```
DROP DATABASE Поставки
```

После выполнения запроса БД будет удалена и пропадёт из дерева объектов.

3. Работа с таблицами

Таблицы являются основным видом объектов БД. В SQL Server Management Studio для каждой таблицы имеются следующие сведения: имя, владелец, тип и дата создания.

Владельцем называют пользователя, создавшего таблицу. Указание на владельца dbo (database owner), означает, что таблицей владеет пользователь, создавший эту БД. Владелец может производить над таблицей любые действия.

Тип таблицы бывает двух видов: System и User (системные и пользовательские). Таблицы типа User создаются пользователями. Таблицы типа System называют системными. В любой БД всегда присутствует целый ряд специальных системных таблиц, необходимых для внутренних

потребностей по обслуживанию БД сервером. В них хранится информация об объектах БД, их свойствах, пользователях, которым разрешен доступ к БД и т.д. Не следует без особой необходимости изменять данные в этих таблицах.

3.1 Создание таблиц

Рассмотрим процесс создания таблицы.

Задание:

В БД «Поставки» необходимо хранить информацию о поставщиках (табельный номер, наименование, адрес, телефон), поставляемых ими товарах (код товара, наименование, фирма-производитель, цена товара), а также о дате и объеме каждой поставки.

3.1.1 Создание таблиц БД средствами SQL Server Management Studio

Рассмотрим процесс создания новой таблицы, например, таблицы «Поставщик». Выберем для БД «Поставки» группу объектов *Таблицы*. Выполним пункт контекстного меню *Создать / Таблица* (рисунок 17).

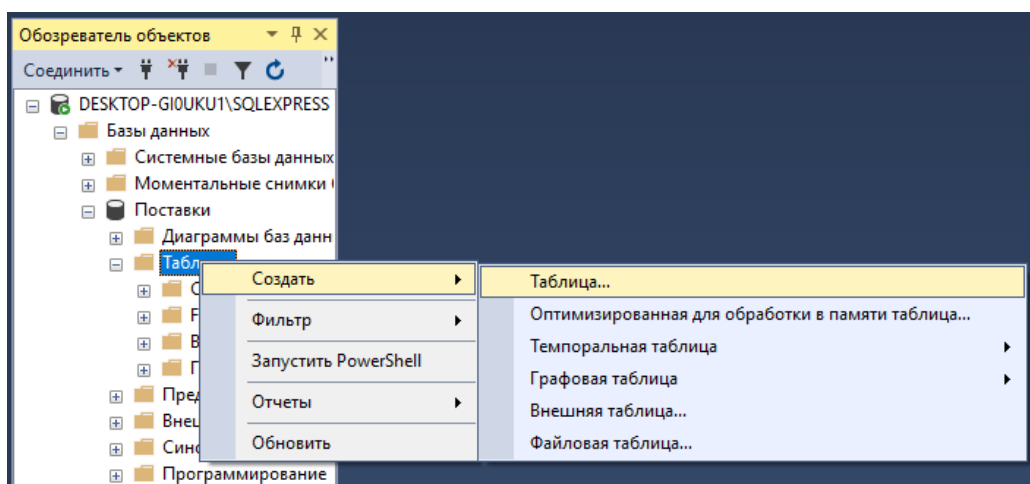


Рисунок 17 – Создание таблицы средствами интерфейса

После этого на правой стороне окна SQL Server Management Studio появляется вкладка *Поставки – dbo.Table_1* (рисунок 18). Далее в открывшемся окне следует ввести информацию об атрибутах таблицы.

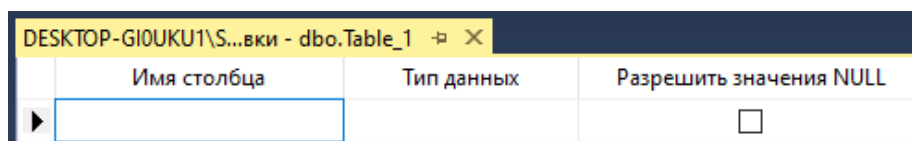


Рисунок 18 – Окно редактирования таблицы

Кроме того, в окне *Свойства столбца* можно задать дополнительные свойства (рисунок 19).

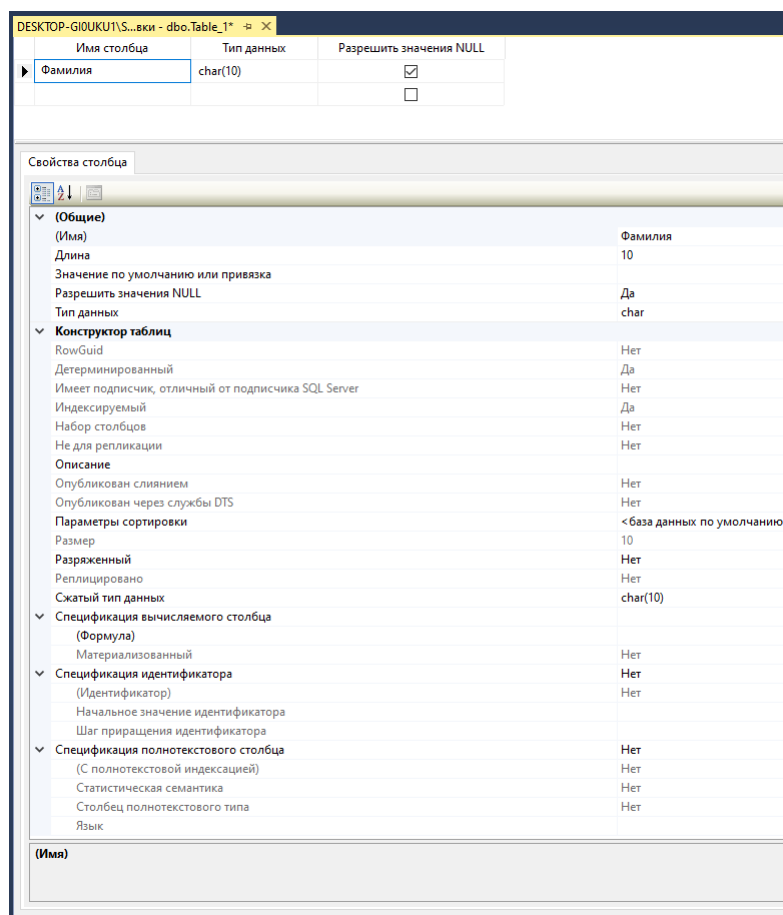



Рисунок 19 – Окно свойств столбца

Чтобы задать первичный ключ таблицы, следует выделить один или несколько атрибутов (для выделения нескольких атрибутов удерживать клавишу *CTRL*) и выбрать на панели инструментов кнопку  *Задать первичный ключ*.

Задание:

Зададим атрибуты: Табельный номер, Наименование, Адрес, Телефон – указав необходимые сведения о них, как показано на рисунке 20. В качестве первичного ключа выберем атрибут Табельный номер.



DESKTOP-GIOUKU1\S...ски - dbo.Table_1* -> X			
	Имя столбца	Тип данных	Разрешить значения NULL
	[Табельный номер]	int	<input type="checkbox"/>
	Наименование	char(25)	<input type="checkbox"/>
	Адрес	char(25)	<input checked="" type="checkbox"/>
	Телефон	char(25)	<input checked="" type="checkbox"/>

Рисунок 20 – Создание таблицы «Поставщик»

Для того, чтобы сохранить таблицу, нажмем кнопку  *Сохранить* на панели инструментов (*Ctrl + S*) или выберем пункт *Сохранить* контекстного меню вкладки *Поставки – dbo.Table_1*.

Имя таблицы можно задать в свойствах, как показано на рисунке 21.

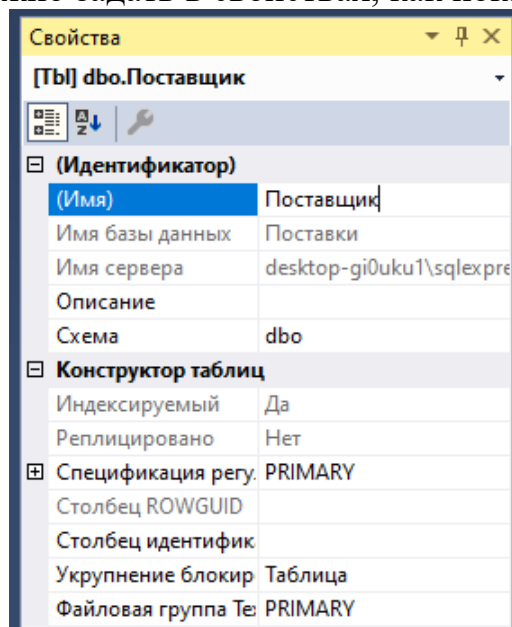


Рисунок 21 – Окно свойств таблицы

Если не задать имя в свойствах, то это можно сделать при сохранении в диалоговом окне *Выбор имени* (рисунок 22).

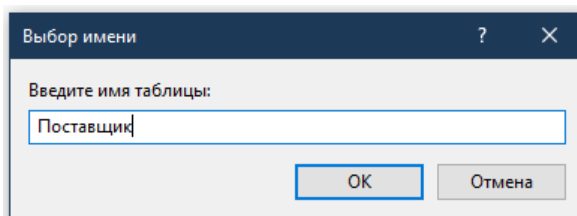


Рисунок 22 – Выбор имени при сохранении таблицы

Обратите внимание, что новая таблица появилась в группе *Таблицы* базы данных «Поставки» (рисунок 23).

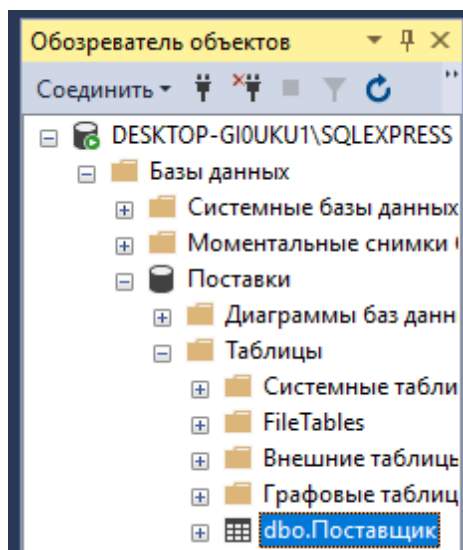


Рисунок 23 – Созданная таблица «Поставщик»

При необходимости можно внести изменения в структуру таблицы, выбрав ее среди таблиц в группе объектов *Таблицы* и выполнив пункт контекстного меню *Проект*.

3.1.2 Создание таблиц БД SQL-запросом

Выполним действия, показанные на рисунке 7, открыв тем самым редактор запросов.

В SQL Server инструкция **CREATE TABLE** создает новую таблицу.

Базовый синтаксис:

```
CREATE TABLE [database_name.][schema_name.]table_name (
    pk_column data_type [PRIMARY KEY],
    column_1 data_type [NOT NULL],
    column_2 data_type,
    ...,
    column_n data_type,
    [table_constraints]
)
```

где:

database_name – имя существующей базы данных (необязательно). Если вы его не укажете, по умолчанию будет использоваться текущая база данных.

schema_name – схема, к которой принадлежит новая таблица (необязательно).

table_name – имя новой таблицы.

Пример:

```
CREATE TABLE Товар (
    [Код товара] INT PRIMARY KEY,
    Наименование NVARCHAR(25) NOT NULL,
    Производитель NVARCHAR(25),
    Цена MONEY
)
```

Т.к., каждая таблица должна иметь первичный ключ, состоящий из одного или нескольких столбцов, обычно сначала перечисляются столбцы первичного ключа (**pk_column**), а затем другие столбцы (**column_n**). Если первичный ключ содержит только один столбец, вы можете использовать ключевые слова **PRIMARY KEY** после имени столбца. Если первичный ключ состоит из двух или более столбцов, вам необходимо указать ограничение **PRIMARY KEY** в качестве ограничения таблицы (**table_constraints**).

Каждый столбец имеет связанный тип данных, указанный после его имени в инструкции (**data_type**). Столбец может иметь одно или несколько ограничений, таких как **NOT NULL** / **NULL** и т.д.

При создании столбцов в T-SQL мы можем использовать ряд атрибутов, ряд которых являются ограничениями (**table_constraints**):

1) С помощью выражения **PRIMARY KEY** столбец можно сделать первичным ключом. Первичный ключ уникально идентифицирует строку в

таблице. В качестве первичного ключа необязательно должны выступать столбцы с типом INT, они могут представлять любой другой тип.

Базовый синтаксис:

pk_column data_type PRIMARY KEY,

либо

PRIMARY KEY(pk_column_1, pk_column_2, ... , pk_column_n)

Пример:

Поставщик INT PRIMARY KEY,

Товар INT PRIMARY KEY,

Дата DATE PRIMARY KEY,

либо

PRIMARY KEY(Поставщик, Товар, Дата)

2) Атрибут **IDENTITY** позволяет сделать столбец идентификатором. Этот атрибут может назначаться для столбцов числовых типов INT, SMALLINT, BIGINT, TINYINT, DECIMAL и NUMERIC.

Базовый синтаксис:

pk_column data_type PRIMARY KEY IDENTITY,

либо

pk_column data_type PRIMARY KEY IDENTITY(seed, increment)

где:

seed – указывает на начальное значение, с которого будет начинаться отсчет.

increment – определяет, насколько будет увеличиваться следующее значение.

По умолчанию атрибут использует значения (1, 1).

Пример:

[Код товара] INT PRIMARY KEY IDENTITY,

либо

[Код товара] INT PRIMARY KEY IDENTITY(10, 5),

3) Атрибут **UNIQUE** позволяет определить столбец с уникальными значениями.

Базовый синтаксис:

column_1 data_type UNIQUE,

либо

`UNIQUE(column_1, column_2, ... , column_n)`**Пример:**Адрес `NVARCHAR(50) UNIQUE`,
Телефон `NVARCHAR(20) UNIQUE`,

либо

`UNIQUE(Адрес, Телефон)`

4) Чтобы указать, может ли столбец принимать значение **NULL**, при определении столбца ему можно задать атрибут **NULL** или **NOT NULL**. Если этот атрибут явным образом не будет использован, то по умолчанию столбец будет допускать значение **NULL**. Исключением является тот случай, когда столбец выступает в роли первичного ключа – в этом случае по умолчанию столбец имеет значение **NOT NULL**.

Базовый синтаксис:`column_1 data_type NOT NULL,`
`column_2 data_type NULL,`**Пример:**Наименование `NVARCHAR(25) NOT NULL`,
Производитель `NVARCHAR(25) NULL`,

5) Атрибут **DEFAULT** определяет значение по умолчанию для столбца. Если при добавлении данных для столбца не будет предусмотрено значение, то для него будет использоваться значение по умолчанию.

Базовый синтаксис:`column_1 data_type DEFAULT default_value,`**Пример:**Объём `INT DEFAULT 50`,

6) Атрибут **CHECK** задает ограничение для диапазона значений, которые могут храниться в столбце. Для этого после слова **CHECK** указывается в скобках условие (предикат), которому должен соответствовать столбец или несколько столбцов. Для соединения условий используется ключевое слово **AND**. Условия можно задать в виде операций сравнения больше (**>**), больше или равно (**>=**), меньше (**<**), меньше или равно (**<=**), не равно (**!=** или **<>**), не больше (**!>**), не меньше (**!<**).

Базовый синтаксис:`column_1 data_type CHECK (constraint),`

либо

`CHECK (constraint)`

где:

constraint – само ограничение.

Пример:

Объём `INT DEFAULT 50 CHECK (Объём > 0 AND Объём < 200)`,
 Адрес `NVARCHAR(50) UNIQUE CHECK(Адрес != '')`,
 Телефон `NVARCHAR(20) UNIQUE CHECK(Телефон != '')`,

либо

`CHECK((Объём > 0 AND Объём < 200) AND (Адрес != '') AND (Телефон != ''))`

С помощью ключевого слова **CONSTRAINT** можно задать имя для ограничений.

Ограничения могут носить произвольные названия, но, как правило, для применяются следующие префиксы:

- PK_ – для PRIMARY KEY;
- FK_ – для FOREIGN KEY;
- CK_ – для CHECK;
- UQ_ – для UNIQUE;
- DF_ – для DEFAULT.

Имена ограничений можно задать на уровне столбцов. Они указываются после **CONSTRAINT** перед атрибутами. Также можно задать все имена ограничений через атрибуты таблицы.

В принципе необязательно задавать имена ограничений, при установке соответствующих атрибутов SQL Server автоматически определяет их имена. Но, зная имя ограничения, мы можем к нему обращаться, например, для его удаления или изменения.

Базовый синтаксис:

`column_1 data_type`
`CONSTRAINT prefix_table_column keyword(constrain),`

либо

`CONSTRAINT prefix_table_column keyword(constrain),`

где:

prefix_table_column – имя ограничения, состоящее из префикса (**prefix**), имени таблицы (**table**) и имени столбца (**column**).

keyword – имя атрибута, устанавливающего ограничение.

constrain – само ограничение.

Пример:

```
[Табельный номер] INT
CONSTRAINT PK_Поставщик_ТабельныйНомер PRIMARY KEY IDENTITY(1, 1),

Объём INT
CONSTRAINT DF_Поставщик_Объём DEFAULT 50
CONSTRAINT CK_Поставщик_Объём CHECK(Объём > 0 AND Объём < 200),

Адрес NVARCHAR(50)
CONSTRAINT UQ_Поставщик_Адрес UNIQUE,

Телефон NVARCHAR(20)
CONSTRAINT UQ_Поставщик_Телефон UNIQUE,

либо

CONSTRAINT PK_Поставщик_ТабельныйНомер PRIMARY KEY ([Табельный номер]),
CONSTRAINT CK_Поставщик_Объём CHECK(Объём > 0 AND Объём < 200),
CONSTRAINT UQ_Поставщик_Адрес UNIQUE(Адрес),
CONSTRAINT UQ_Поставщик_Телефон UNIQUE(Телефон)
```

Рассмотрим процесс создания новой таблицы «Товар».

Задание:

Зададим атрибуты: Код товара, Наименование, Фирма-производитель, Цена товара – указав необходимые сведения о них. В качестве первичного ключа выберем атрибут Код товара.

Код товара – поле-счётчик, наименование товара – уникальное значение, цена – положительное значение не может быть меньше 1 рубля.

Пример:

```
CREATE TABLE Товар (
    [Код товара] INT IDENTITY(1, 1),
    Наименование NVARCHAR(25) NOT NULL,
    Производитель NVARCHAR(25),
    Цена MONEY
    CONSTRAINT DF_Товар_Цена DEFAULT (1),

    CONSTRAINT PK_Товар_КодТовара PRIMARY KEY ([Код товара]),
    CONSTRAINT UQ_Товар_Наименование UNIQUE (Наименование),
    CONSTRAINT CK_Товар_Цена CHECK (Цена > 0),
)
```

После выполнения запроса новая таблица появится в группе *Таблицы* базы данных «Поставки» (рисунок 24).

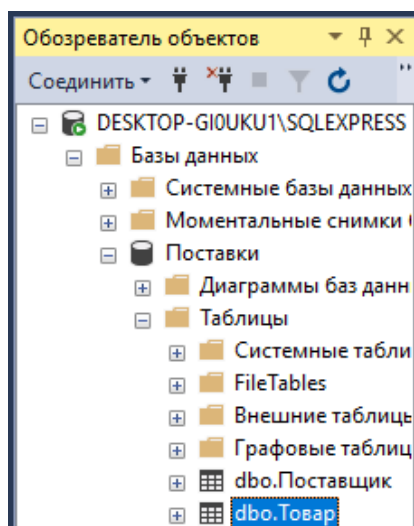


Рисунок 24 – Созданная таблица «Товар»

3.2 Удаление таблиц

Рассмотрим процесс удаления существующей таблицы.

3.2.1 Удаление таблиц БД средствами SQL Server Management Studio

В дереве *Обозревателя объектов* в узле *Таблицы* выберем нужную для удаления, выполним пункт контекстного меню *Удалить*.

Затем в открывшемся окне *Удаление объекта* необходимо нажать *OK* (рисунок 25).

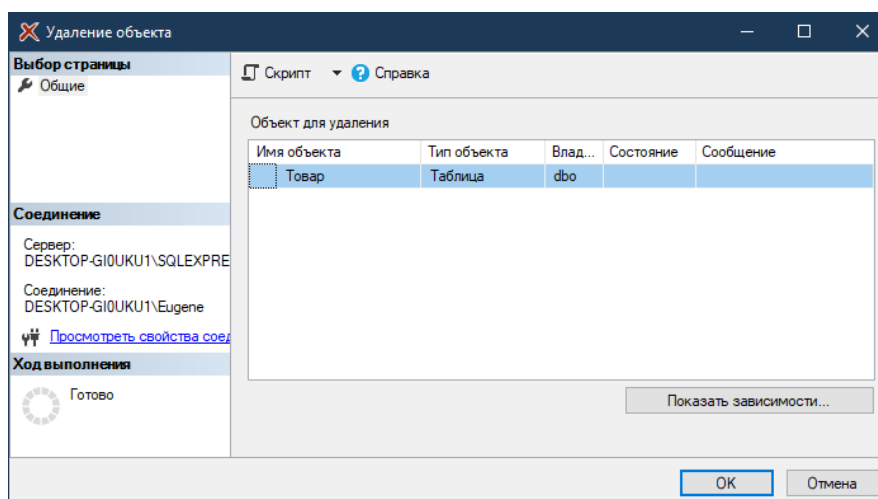


Рисунок 25 – Удаление таблицы средствами интерфейса

После проделанных действия таблица будет удалена и пропадёт из дерева объектов.

3.2.2 Удаление таблиц БД SQL-запросом

В SQL Server инструкция **DROP DATABASE** удаляет одну или несколько пользовательских баз данных или моментальных снимков базы данных из экземпляра SQL Server.

Базовый синтаксис:**DROP TABLE**

[IF EXISTS] [database_name.][schema_name.]table_name [, ...]

где:

IF EXISTS – условное удаление таблицы только в том случае, если она уже существует. Если попытаться удалить несуществующую таблицу без указания IF EXISTS, SQL Server выдаст сообщение об ошибке.

database_name – имя существующей базы данных (необязательно). Если вы его не укажете, по умолчанию будет использоваться текущая база данных.

schema_name – схема, к которой принадлежит удаляемая таблица (необязательно).

table_name – имя удаляемой таблицы.

Когда SQL Server удаляет таблицу, он также удаляет все данные, триггеры, ограничения, разрешения этой таблицы. Более того, SQL Server явно не удаляет представления и хранимые процедуры, которые ссылаются на удаленную таблицу. Поэтому, чтобы явно удалить эти зависимые объекты, вы должны использовать оператор **DROP VIEW** и **DROP PROCEDURE**.

Если вы используете инструкцию **DROP TABLE** для удаления связанных таблиц, дочерняя таблица должна быть указана перед родительской.

Пример:**DROP TABLE** Поставка

После выполнения запроса таблица будет удалена и пропадёт из дерева объектов.

4. Работа со связями

Базы данных могут содержать таблицы, которые связаны между собой различными связями. Связь представляет ассоциацию между сущностями разных типов.

При выделении связи выделяют главную или родительскую таблицу (primary key table / master table) и зависимую, дочернюю таблицу (foreign key table / child table). Дочерняя таблица зависит от родительской.

Для организации связи используются внешние ключи. Внешний ключ представляет один или несколько столбцов из одной таблицы, который одновременно является потенциальным ключом из другой таблицы. Внешний ключ необязательно должен соответствовать первичному ключу из главной таблицы. Хотя, как правило, внешний ключ из зависимой таблицы указывает на первичный ключ из главной таблицы.

4.1 Создание связей

Рассмотрим процесс создания связей между таблицами.

4.1.1 Создание связей средствами SQL Server Management Studio

Для создания связей между таблицами и контроля целостности связей используются объекты-диаграммы. Диаграмма по своей сути является схемой реляционной БД. Чтобы создать новую диаграмму следует выбрать группу объектов *Диаграммы баз данных* и выполнить пункт контекстного меню *Создать диаграмму базы данных* (рисунок 26).

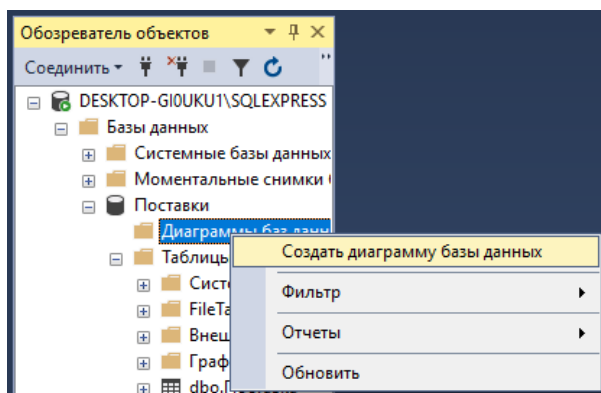


Рисунок 26 – Создание диаграммы базы данных

При первом открытии группы будет выведено диалоговое окно (рисунок 27), в котором нужно нажать *Да*.

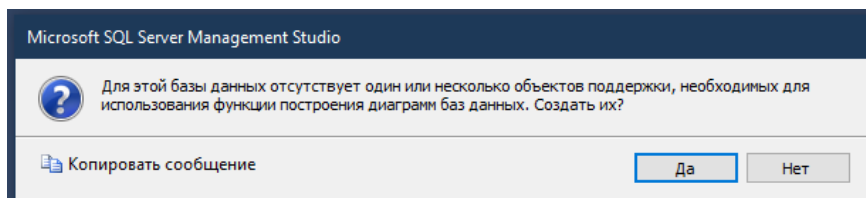


Рисунок 27 – Создание объектов поддержки для БД

Далее будет предложен мастер по добавлению таблиц на диаграмму (рисунок 28).

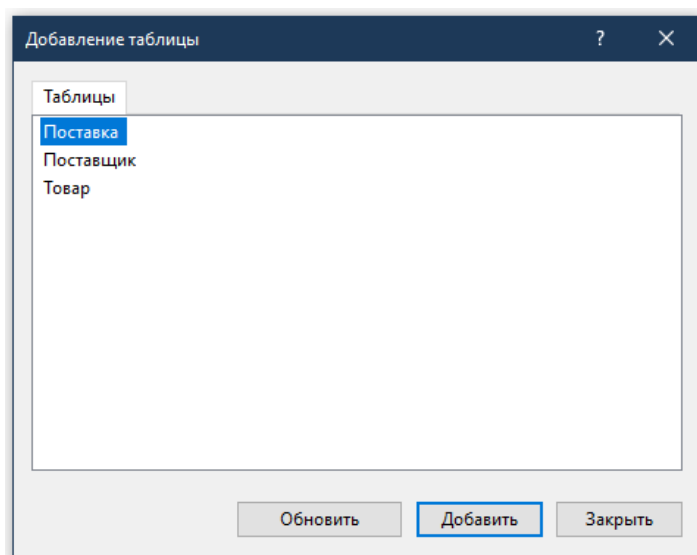


Рисунок 28 – Добавление таблиц на диаграмму

В окне *Добавление таблицы* из списка *Таблицы* следует выбрать необходимые и нажать кнопку *Добавить*. На рабочей области SQL Server Management Studio появится новая вкладка *Diagram*, где будут размещены выбранные таблицы (рисунок 29).

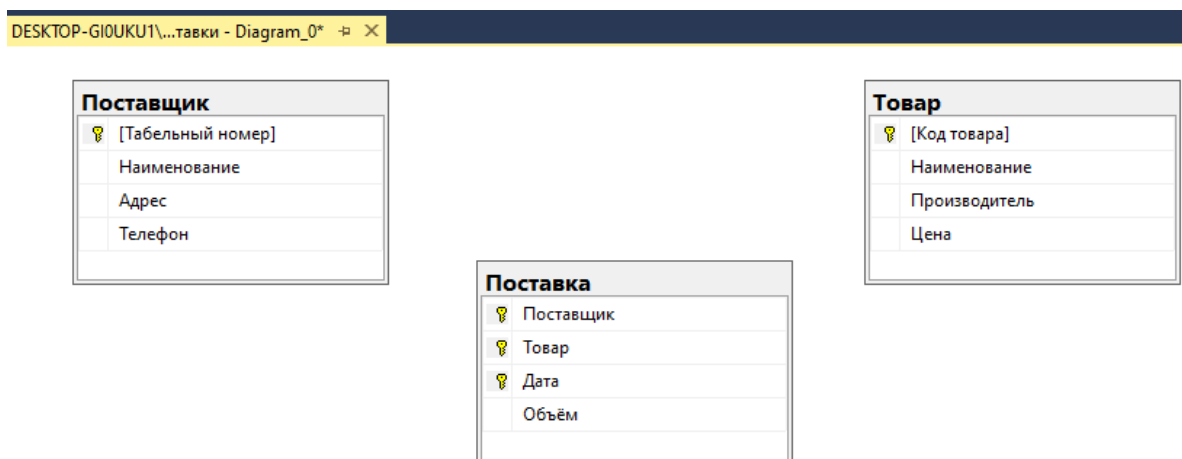


Рисунок 29 – Добавленные на диаграмму таблицы

Теперь можно расставить необходимые связи между таблицами, указывая мышью поля связи.

Указатель мыши подводят к внешнему ключу одной таблицы, нажимают левую кнопку мыши и тянут связь к внешнему ключу другой таблицы. При создании новой связи появится диалоговое окно *Таблицы и столбцы* (рисунок 30). В нем можно изменить название связи, которое по умолчанию задается следующим образом: *FK_<имя первой таблицы>_<имя второй таблицы>*. В этом окне полезно проверить имена соединяемых таблиц и атрибутов, по которым соединяются таблицы. Это имя главной таблицы – *Таблица первичного ключа* и имя подчиненной таблицы – *Таблица внешнего ключа*. Соответствующие атрибуты таблиц записываются под именами таблиц.

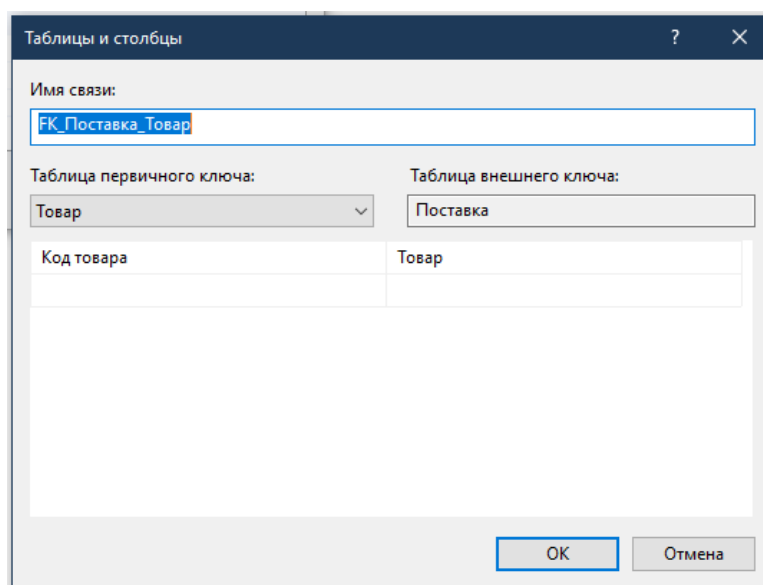


Рисунок 30 – Создание связи средствами интерфейса

В следующем окне *Связь по внешнему ключу* можно задать параметры связи. В группе *Общие* можно задать проверку существования данных при создании связи, установив свойство *Проверить существующие данные при создании или повторном включении* на *Да* (рисунок 31).

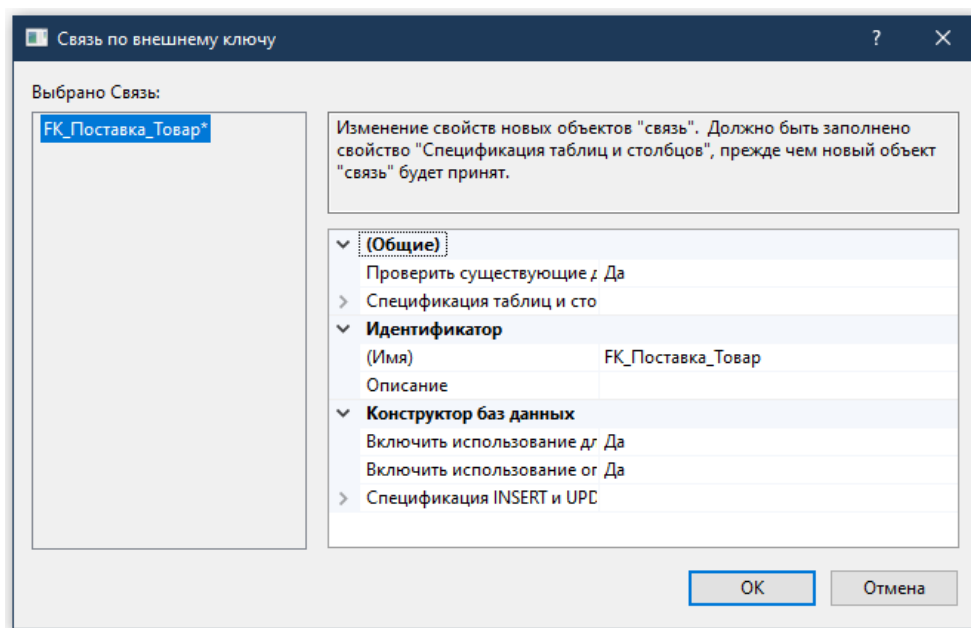


Рисунок 31 – Настройка связи

В группе *Конструктор баз данных* можно задать тип стратегии поддержания ссылочной целостности при выполнении операций Delete и Update в главной таблице для связи (рисунок 32).

Могут быть заданы следующие стратегии:

- 1) Нет действия – не разрешается выполнение операции, приводящей к нарушению ссылочной целостности.
- 2) Каскадно – разрешается выполнение операции, приводящей к нарушению ссылочной целостности, но при этом вносятся соответствующие поправки в другие отношения. Изменение начинается в главном отношении, и каскадно выполняется в подчиненном.
- 3) Присвоить NULL – разрешается выполнение операции, приводящей к нарушению ссылочной целостности, но при этом все возникающие некорректные значения внешних ключей устанавливаются в NULL.
- 4) Присвоить значение по умолчанию – разрешается выполнение операции, приводящей к нарушению ссылочной целостности, но при этом все возникающие некорректные значения внешних ключей изменяются на некоторые значения, принятые по умолчанию.

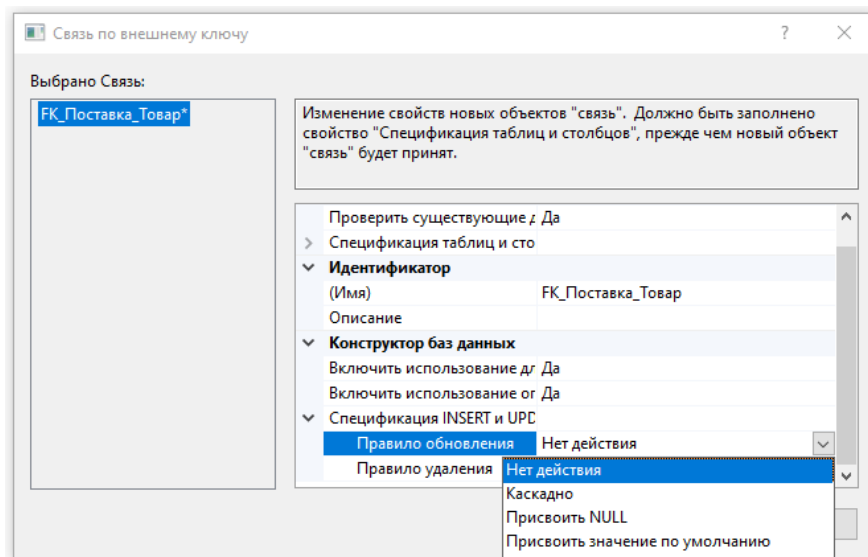


Рисунок 32 – Настройка стратегии поддержания ссылочной целостности

Если осуществляется связь по составным внешним ключам, то в окне параметров связи следует указать атрибуты, входящие в состав внешних ключей для обеих таблиц. Пример созданной диаграммы представлен на рисунке 33.

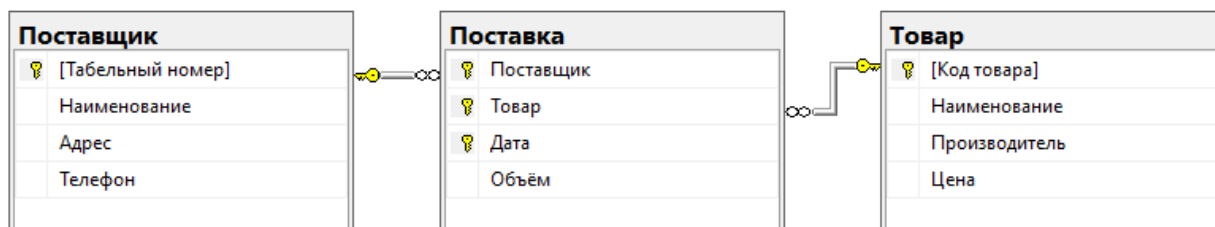



Рисунок 33 – Создание диаграммы БД

Созданную диаграмму сохраняют, нажав кнопку  *Сохранить* на панели инструментов (*Ctrl + S*). При этом необходимо указать имя диаграммы (рисунок 34).

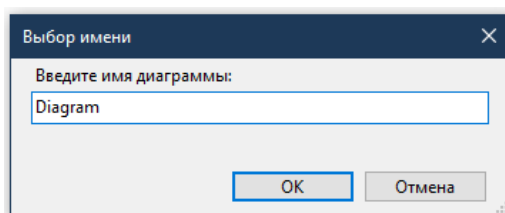


Рисунок 34 – Ввод имени диаграммы

Новая диаграмма появится в списке группы объектов *Диаграммы баз данных* (рисунок 35). При необходимости можно внести изменения в диаграмму, открыв ее двойным щелчком мыши.

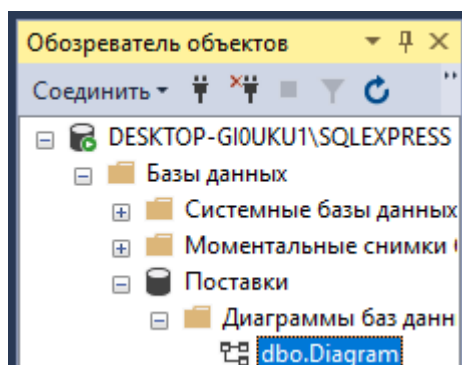


Рисунок 35 – Созданная диаграмма

4.1.2 Создание связей SQL-запросом

Установить связи можно на этапе создания таблиц, используя ограничение **FOREIGN KEY**.

Внешние ключи применяются для установки связи между таблицами. Внешний ключ устанавливается для столбцов из зависимой, подчиненной таблицы, и указывает на один из столбцов из главной таблицы. Хотя, как правило, внешний ключ указывает на первичный ключ из связанной главной таблицы, но это необязательно должно быть неперенным условием. Внешний ключ также может указывать на какой-то другой столбец, который имеет уникальное значение.

Базовый синтаксис:

На уровне столбца:

```
[FOREIGN KEY] REFERENCES master_table (master_table_column)
    [ON DELETE {CASCADE|NO ACTION}]
    [ON UPDATE {CASCADE|NO ACTION}]
```

На уровне таблицы:

```
FOREIGN KEY (child_table_column)
REFERENCES master_table (master_table_column)
    [ON DELETE {CASCADE|NO ACTION}]
    [ON UPDATE {CASCADE|NO ACTION}]
```

где:

master_table – главная таблица.

child_table – подчиненная таблица

master_table_column – столбец главной таблицы, на которую ссылается подчиненная таблица.

child_table_column – столбец подчиненной таблицы, который ссылается на главную таблицу.

REFERENCES – ссылка на главную таблицу.

ON DELETE / ON UPDATE – действия, которые будут выполняться соответственно при удалении и изменении связанной строки из главной таблицы.

Пример:

```
Поставщик INT REFERENCES Поставщик ([Табельный номер])
           ON DELETE CASCADE
           ON UPDATE CASCADE,
```

либо:

```
FOREIGN KEY (Товар)
REFERENCES Товар ([Код товара])
ON DELETE CASCADE
ON UPDATE CASCADE
```

Для определения действия мы можем использовать следующие опции:

- 1) **CASCADE**: автоматически удаляет или изменяет строки из зависимой таблицы при удалении или изменении связанных строк в главной таблице.
- 2) **NO ACTION**: предотвращает какие-либо действия в зависимой таблице при удалении или изменении связанных строк в главной таблице. То есть фактически какие-либо действия отсутствуют.
- 3) **SET NULL**: при удалении связанной строки из главной таблицы устанавливает для столбца внешнего ключа значение **NULL**.
- 4) **SET DEFAULT**: при удалении связанной строки из главной таблицы устанавливает для столбца внешнего ключа значение по умолчанию, которое задается с помощью атрибуты **DEFAULT**. Если для столбца не задано значение по умолчанию, то в качестве него применяется значение **NULL**.

По умолчанию, если на строку из главной таблицы по внешнему ключу ссылается какая-либо строка из зависимой таблицы, то мы не сможем удалить эту строку из главной таблицы.

Так как первичные ключи, как правило, изменяются очень редко, да и в принципе не рекомендуется использовать в качестве первичных ключей столбцы с изменяемыми значениями, то на практике выражение **ON UPDATE** используется редко.

Рассмотрим процесс создания новой таблицы «Поставка», связанной с таблицами «Поставщик» и «Товар».

Задание:

Зададим атрибуты: Поставщик (ссылка на Табельный номер поставщика), Товар (ссылка на Код товара), Дата, Объем – указав необходимые сведения о них. В качестве первичного ключа выберем атрибуты Поставщик, Товар, Дата.

Поле Объем – положительное значение не может быть меньше 1 единицы.

При удалении/обновлении записей из главной таблицы, удалять/обновлять записи в подчиненной.

Пример:

```

CREATE TABLE Поставка (
    Поставщик INT,
    Товар INT,
    Дата DATE,
    Объем INT
    CONSTRAINT DF_Поставка_Объем DEFAULT (1),

    CONSTRAINT PK_Поставка PRIMARY KEY (Поставщик, Товар, Дата),
    CONSTRAINT CK_Поставка_Объем CHECK (Объем > 0),

    CONSTRAINT FK_Поставка_Поставщик FOREIGN KEY (Поставщик)
        REFERENCES Поставщик ([Табельный номер])
        ON DELETE CASCADE
        ON UPDATE CASCADE,

    CONSTRAINT FK_Поставка_Товар FOREIGN KEY (Товар)
        REFERENCES Товар ([Код товара])
        ON DELETE CASCADE
        ON UPDATE CASCADE
)

```

После выполнения запроса будет создана новая таблица, а также связи с уже существующими таблицами. Для проверки можно создать новую диаграмму и добавить существующие таблицы, если всё сделано верно, то связи также отобразятся на диаграмме.

4.2 Удаление связей

Рассмотрим процесс удаления связей между таблицами.

4.2.1 Удаление связей средствами SQL Server Management Studio

Удалить созданную связь можно, выбрав её на диаграмме и выполнив действие *Удалить связи из базы данных* контекстного меню (рисунок 36). После чего подтвердить удаление выбранной связи.

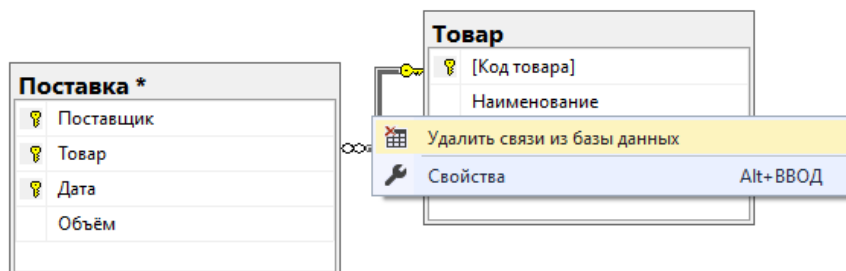


Рисунок 36 – Удаление связи

Если требуется удалить диаграмму целиком, то в дереве *Обозревателя объектов* в узле *Диаграммы баз данных* выберем нужную для удаления, выполним пункт контекстного меню *Удалить*.

Затем в открывшемся окне *Удаление объекта* необходимо нажать *ОК* (рисунок 37).

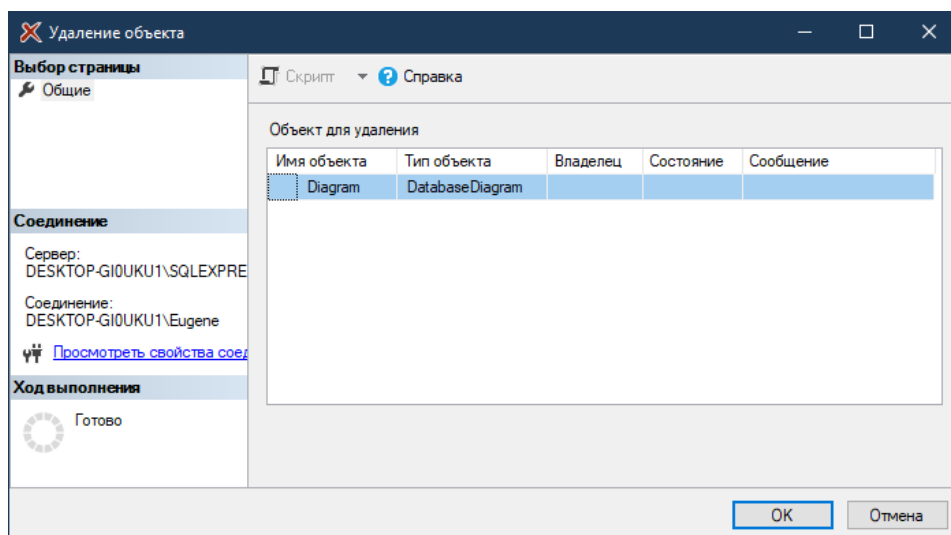


Рисунок 37 – Удаление диаграммы средствами интерфейса

После проделанных действия диаграмма будет удалена и пропадёт из дерева объектов.

4.2.2 Удаление связей SQL-запросом

Удаление связей происходит по внешнему ключу. При удалении ограничения внешнего ключа удаляется требование принудительного создания ссылочной целостности.

В SQL Server инструкция **DROP CONSTRAINT** удаляет ограничение таблицы. Однако напрямую её использовать нельзя, необходимо редактировать таблицу, где это ограничение задано, используя инструкцию **ALTER TABLE**.

Базовый синтаксис:

```
ALTER TABLE table_name
DROP CONSTRAINT constraint
```

где:

ALTER TABLE – инструкция для редактирования существующей таблицы.

table_name – имя редактируемой таблицы.

constraint – имя удаляемого ограничения.

Пример:

```
ALTER TABLE Поставка
DROP CONSTRAINT FK_Поставка_Поставщик
```

После выполнения запроса связь будет удалена. Проверить это можно, развернув узел *Ключи* выбранной таблицы (рисунок 38).

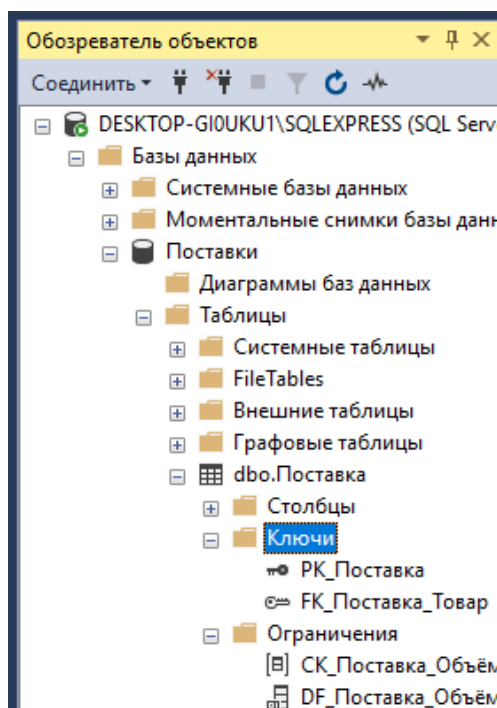


Рисунок 38 – Просмотр ключей и ограничений таблицы

Аналогичным способом можно удалять и другие ограничения, существующие в таблице.

Таким образом, мы сначала создали базу данных, а затем в эту БД добавляли таблицы с помощью отдельных команд SQL. Но можно сразу совместить в одном скрипте несколько команд. В этом случае отдельные наборы команд называются пакетами (batch).

Каждый пакет состоит из одного или нескольких SQL-выражений, которые выполняются как одно целое. В качестве сигнала завершения пакета и выполнения его выражений служит команда **GO**.

Смысл разделения SQL-выражений на пакеты состоит в том, что одни выражения должны успешно выполниться до запуска других выражений. Например, при добавлении таблиц мы должны бы быть уверены, что была создана база данных, в которой мы собираемся создать таблицы.

Для БД из примера скрипт будет выглядеть следующим образом:

Пример:

```
CREATE DATABASE Поставки

GO

USE Поставки

CREATE TABLE Товар (
    [Код товара] INT IDENTITY(1, 1) NOT NULL,
    Наименование NVARCHAR(25) NOT NULL,
    Производитель NVARCHAR(25) NULL,
    Цена MONEY
    CONSTRAINT DF_Товар_Цена DEFAULT (1) NULL,
```

```

CONSTRAINT PK_Товар_КодТовара PRIMARY KEY ([Код товара]),
CONSTRAINT UQ_Товар_Наименование UNIQUE (Наименование),
CONSTRAINT CK_Товар_Цена CHECK (Цена > 0),
)

CREATE TABLE Поставщик (
    [Табельный номер] INT IDENTITY(1, 1) NOT NULL,
    Наименование NVARCHAR(25) NOT NULL,
    Адрес NVARCHAR(25) NULL,
    Телефон NVARCHAR(25) NULL,

    CONSTRAINT PK_Поставщик_ТабельныйНомер PRIMARY KEY ([Табельный номер]),
    CONSTRAINT UQ_Поставщик_Наименование UNIQUE (Наименование),
    CONSTRAINT UQ_Поставщик_Адрес UNIQUE(Адрес),
    CONSTRAINT UQ_Поставщик_Телефон UNIQUE(Телефон)
)

CREATE TABLE Поставка (
    Поставщик INT NOT NULL,
    Товар INT NOT NULL,
    Дата DATE NOT NULL,
    Объем INT
    CONSTRAINT DF_Поставка_Объем DEFAULT (1) NULL,

    CONSTRAINT PK_Поставка PRIMARY KEY (Поставщик, Товар, Дата),
    CONSTRAINT CK_Поставка_Объем CHECK (Объем > 0),

    CONSTRAINT FK_Поставка_Поставщик FOREIGN KEY (Поставщик)
        REFERENCES Поставщик ([Табельный номер])
        ON DELETE CASCADE
        ON UPDATE CASCADE,

    CONSTRAINT FK_Поставка_Товар FOREIGN KEY (Товар)
        REFERENCES Товар ([Код товара])
        ON DELETE CASCADE
        ON UPDATE CASCADE
)

```

Вначале создается БД «Поставки». Затем идет команда GO, которая сигнализирует, что можно выполнять следующий пакет выражений. Далее выполняется второй пакет, который переключает контекст на созданную БД, добавляет в нее таблицы и ограничения.

5. Перенос базы данных

Часто возникает необходимость копирования файлов БД с одного компьютера на другой. Сделать это можно несколькими способами.

5.1 Подключение и отключение БД

Пусть требуется скопировать БД с сервера А на сервер В. Для этого необходимо выполнить следующие действия:

- 1) Отключить БД от сервера А.
- 2) Скопировать файлы БД на носитель информации.
- 3) Подключить БД обратно к серверу А.
- 4) Скопировать файлы БД с носителя на сервер В.
- 5) Подключить БД к серверу В.

Отключение БД с помощью SQL Server Management Studio осуществляется следующим образом:

- 1) Выбрать БД.

2) Выполнить пункт контекстного меню *Задачи / Отсоединить* (рисунок 39).

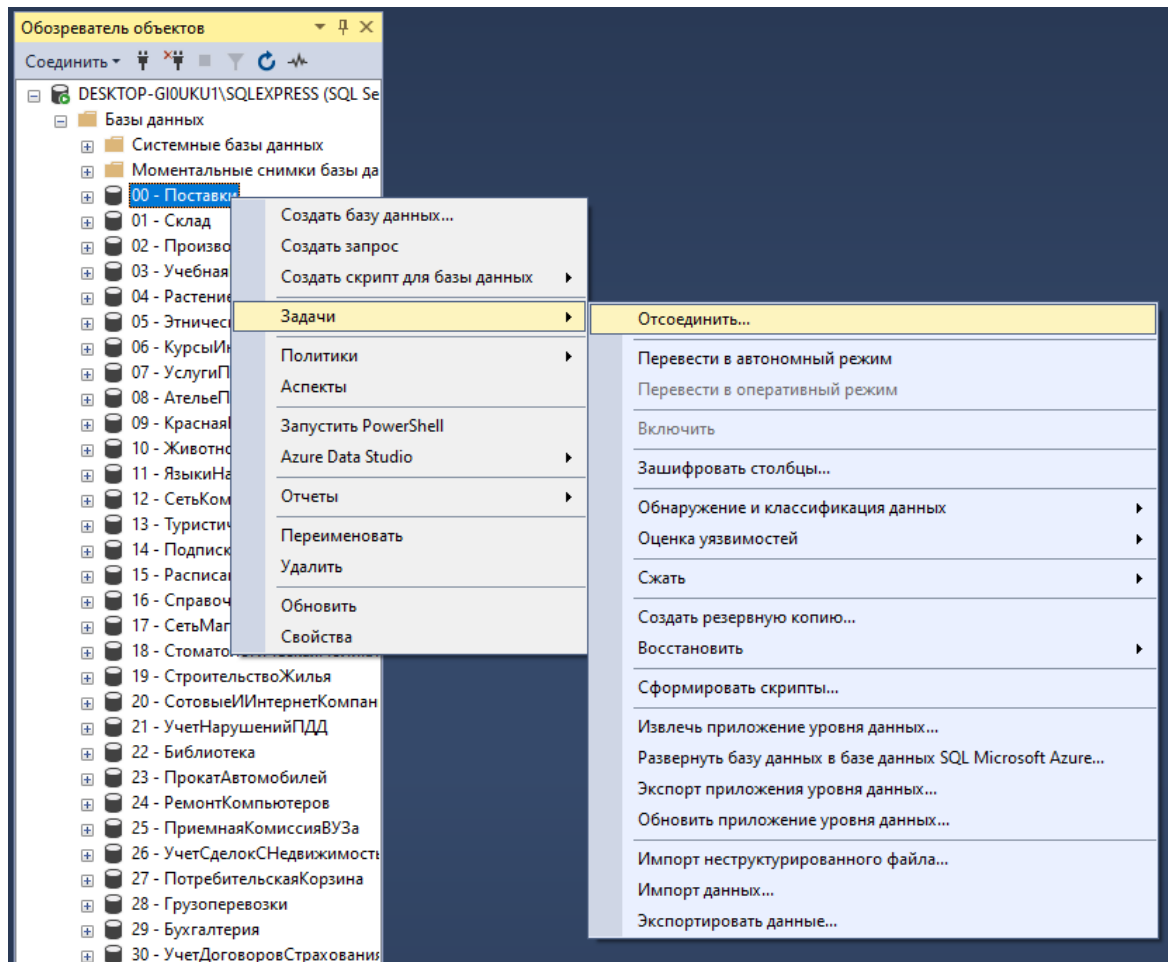


Рисунок 39 – Отсоединение БД от сервера

3) В окне *Отсоединение базы данных* подтвердить действие (рисунок 40).

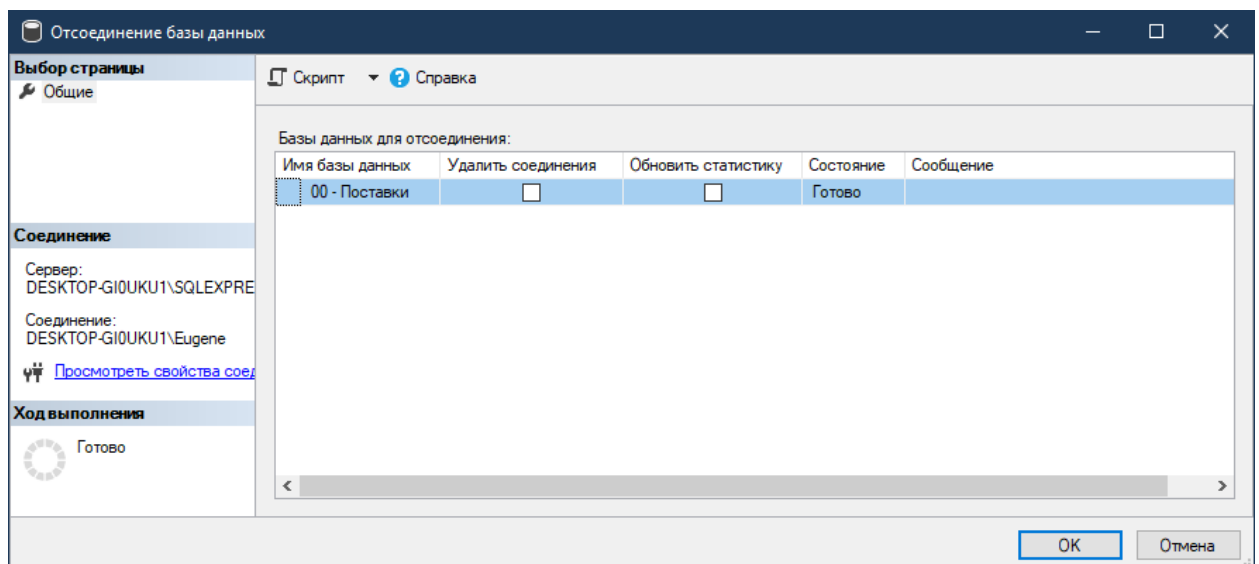


Рисунок 40 – Подтверждение отсоединения

Подключение БД к серверу с помощью SQL Server Management Studio осуществляется следующим образом:

- 1) Выбрать узел *Базы данных*.
- 2) Выполнить пункт *Присоединить* из контекстного меню (рисунок 41).

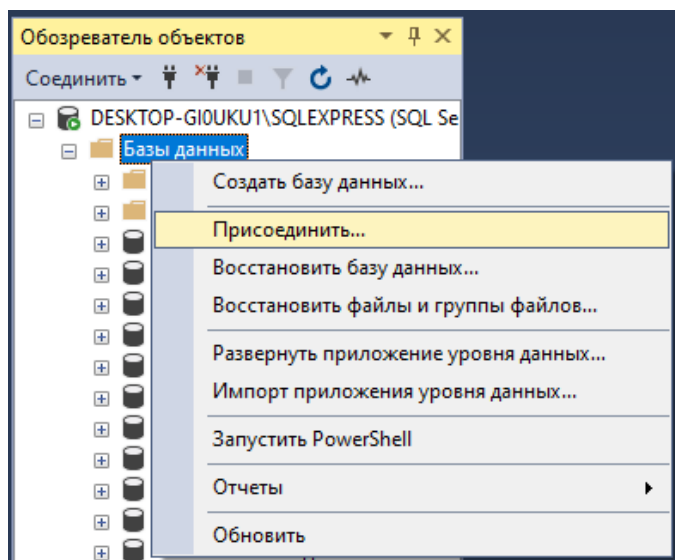


Рисунок 40 – Присоединение БД

- 3) В открывшемся окне *Присоединение баз данных* выбрать файл для добавления через кнопку *Добавить*, указать имя, под которым БД подключается к серверу, а также физическое имя файла. Если файл журнала транзакций находится в том же каталоге, то он подключается автоматически (рисунок 42).

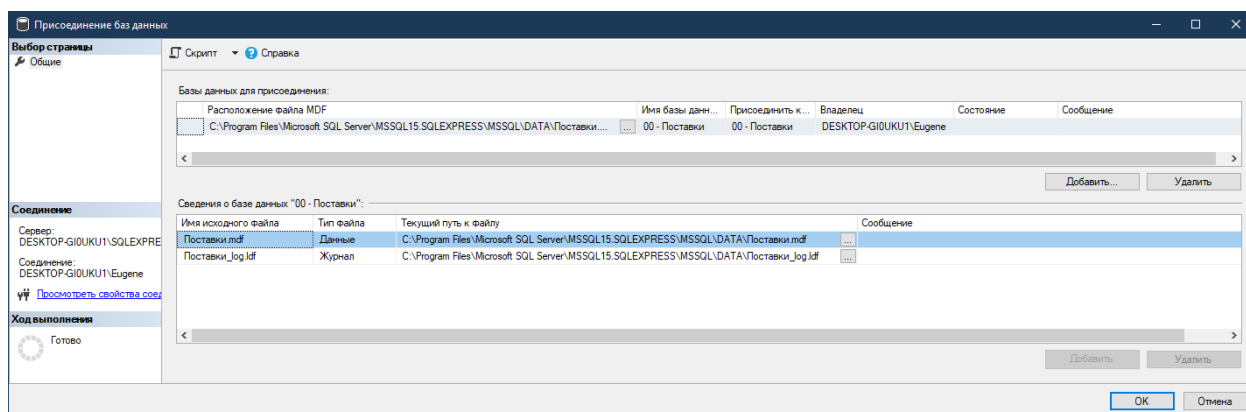


Рисунок 42 – Выбор файла для присоединения

5.2 Создание скрипта

Осуществить перенос информации в пустую БД можно, создав скрипт. Для этого необходимо выполнить следующие действия:

- 1) Выбрать нужную БД в *Обозревателе объектов*.
- 2) Выполнить пункт контекстного меню *Задачи / Сформировать скрипты* (рисунок 43).

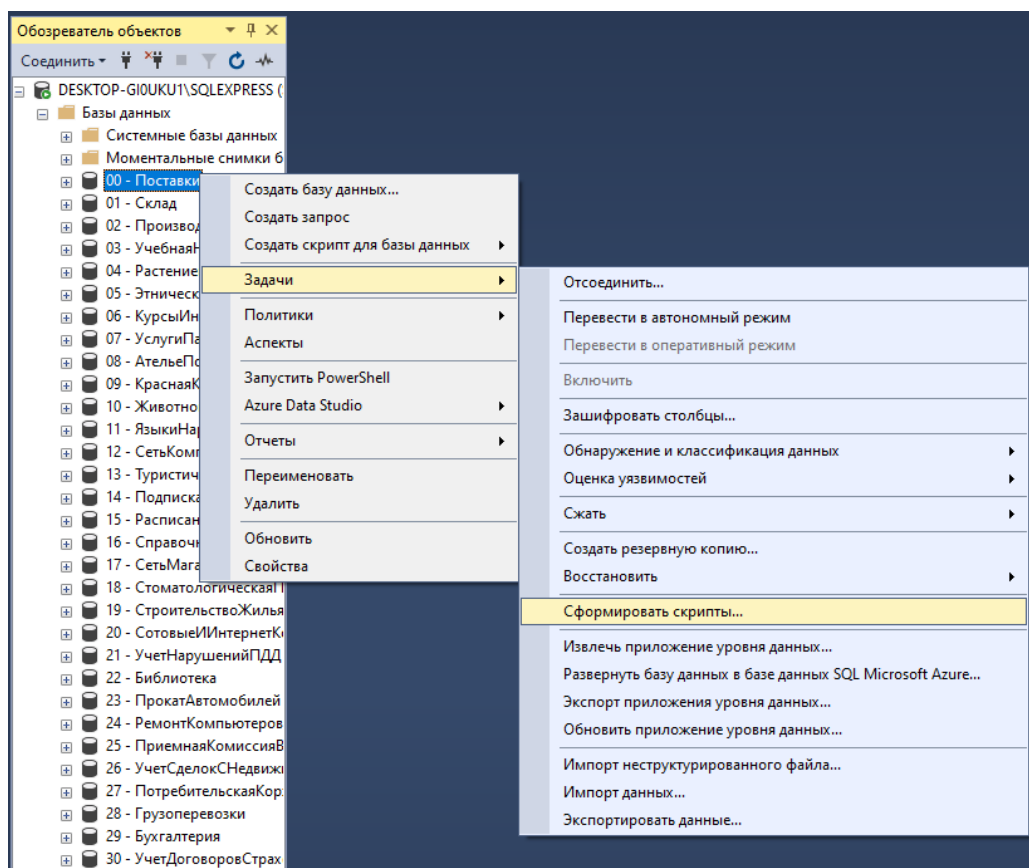


Рисунок 43 – Создание скрипта для БД

3) В открывшемся мастере *Создание скриптов* на 1 шаге нажать *Далее* (рисунок 44).

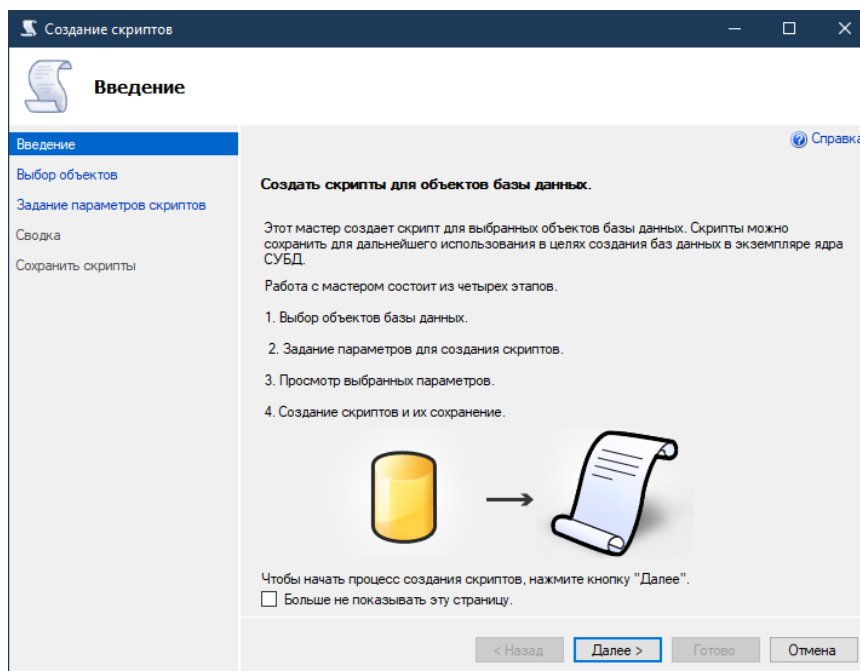


Рисунок 44 – Шаг 1 мастера «Создание скриптов»

4) Далее необходимо выбрать объекты, которые будут включены в скрипт. Выбираем все объекты (рисунок 45).

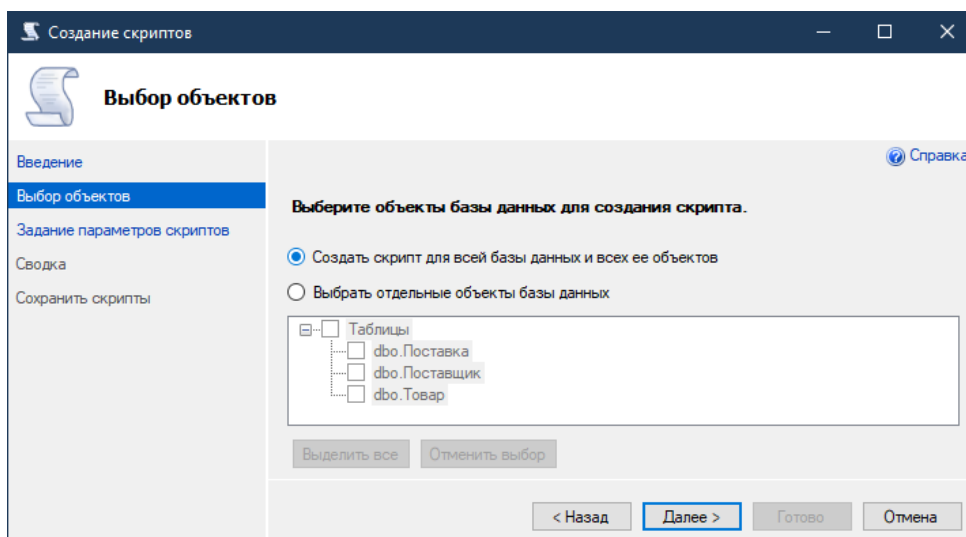


Рисунок 45 – Шаг 2 мастера «Создание скриптов»

5) Затем нужно включить данные в скрипт, нажав на *Дополнительно* на текущем шаге и выбрать соответствующую настройку (рисунки 46-47).

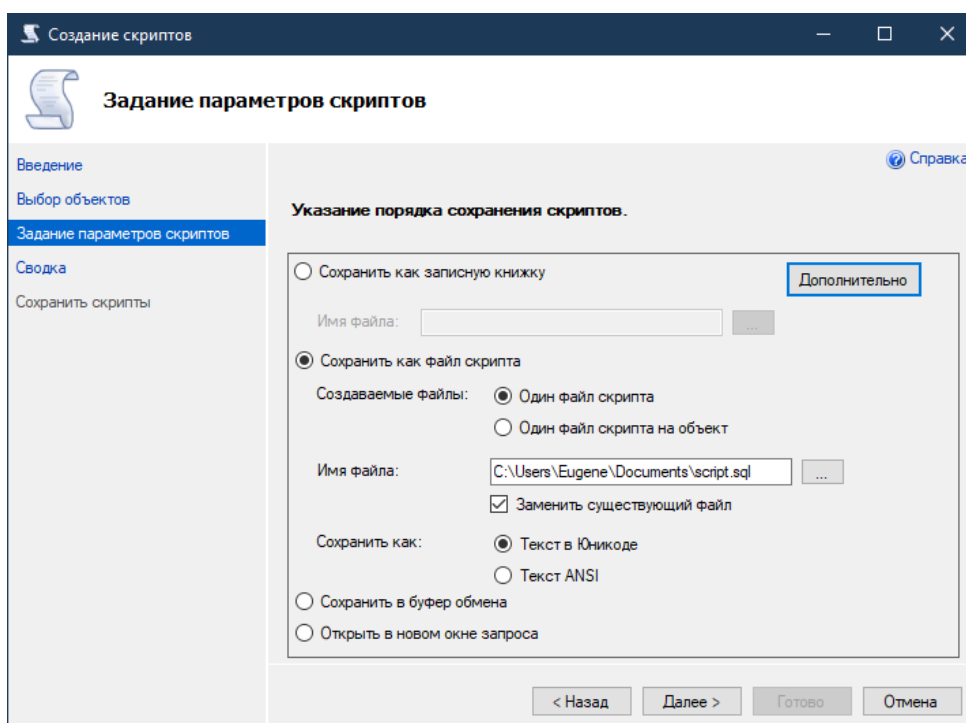


Рисунок 46 – Шаг 3 мастера «Создание скриптов»

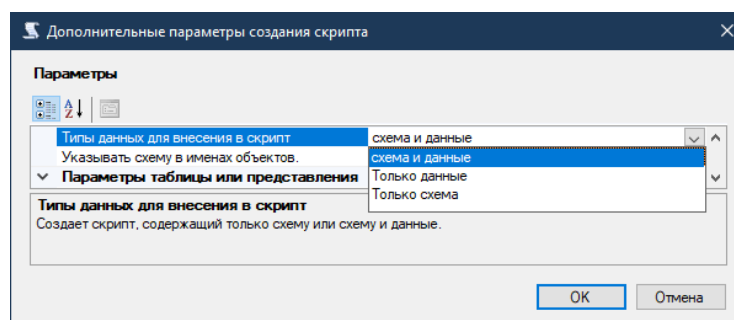


Рисунок 47 – Настройка дополнительных параметров

- 6) Проверить все параметры и нажать *Далее* (рисунок 48).

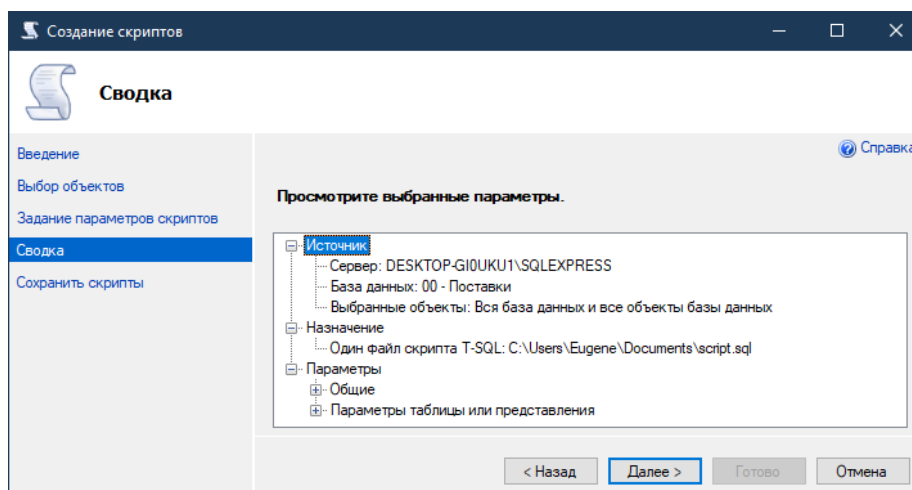


Рисунок 48 – Шаг 4 мастера «Создание скриптов»

- 7) Дождаться окончания создания и сохранения скрипта (рисунок 49), после чего нажать *Готово*.

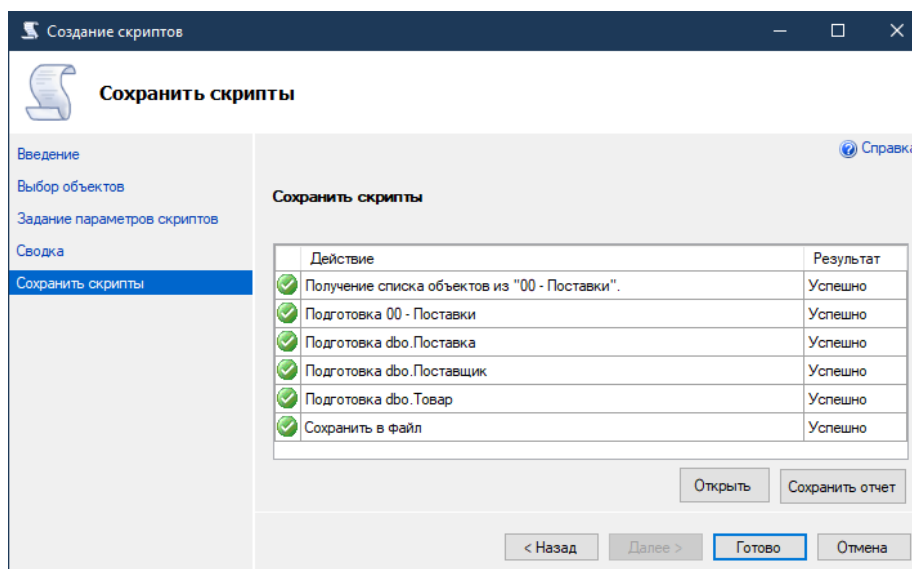


Рисунок 49 – Шаг 5 мастера «Создание скриптов»

- 8) Для окончания переноса необходимо создать пустую БД, открыть и выполнить файл скрипта. После этого все данные будут занесены в новую БД.

5.3 Резервное копирование

В Microsoft SQL Server принята практика разных типов резервного копирования. Пользователям доступно:

- полное – делается резервная копия всей БД;
- дифференциальное или разностное – осуществляется копирование данных с того момента, когда осуществлялось ее последнее полное резервирование;
- логов или инкрементальное.

Чтобы создать резервную копию базы данных, выполните следующие действия:

1) Выберите нужную БД, выполните пункт контекстного меню *Задачи / Резервное копирование* (рисунок 50).

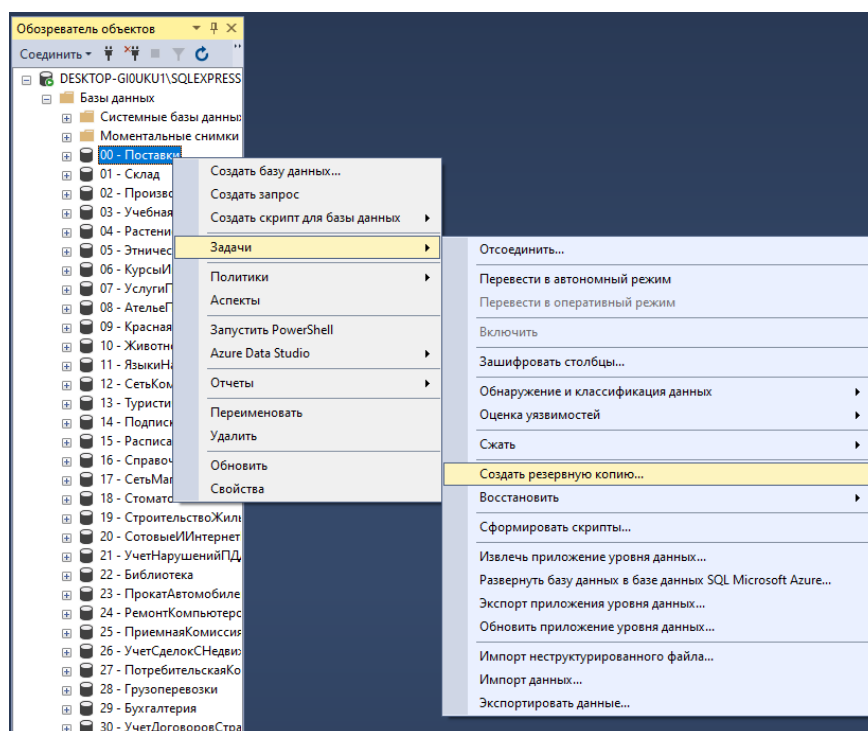


Рисунок 50 – Создание резервной копии средствами интерфейса

2) В разделе *Назначение* проверьте правильность пути к резервной копии. Если вам нужно изменить его, выберите *Удалить* для удаления существующего пути, а затем *Добавить*, чтобы ввести новый путь (рисунок 51). Затем нажмите ОК.

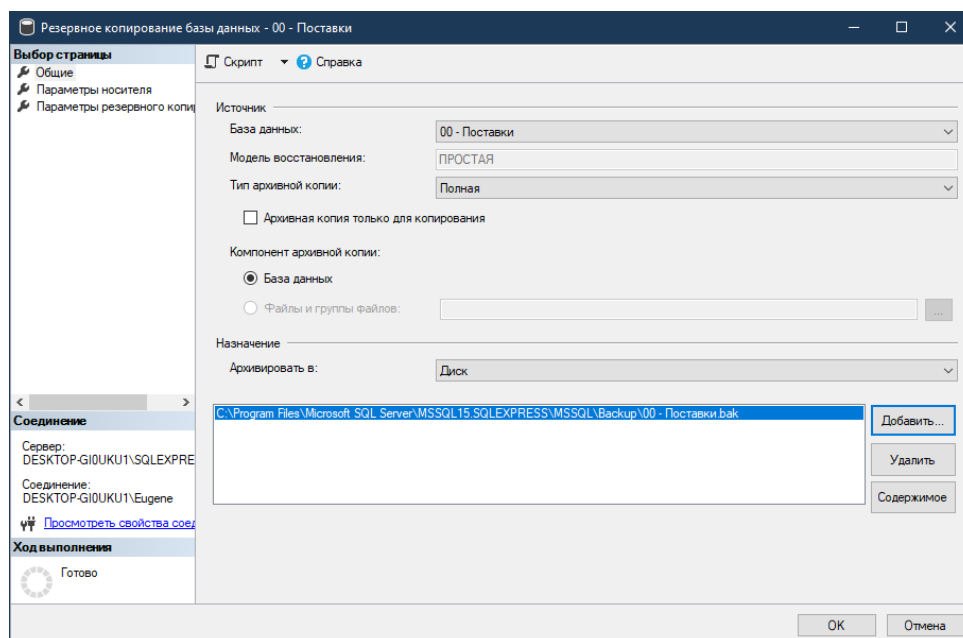


Рисунок 51 – Настройка резервного копирования

Кроме того, резервную копию базы данных можно создать с помощью команды T-SQL – **BACKUP**.

В SQL Server инструкция **BACKUP** создает резервную копию всей базы данных SQL Server (чтобы получить резервную копию базы данных) либо файлов или файловых групп базы данных (чтобы получить резервную копию файлов (**BACKUP DATABASE**)). Кроме того, при использовании модели полного восстановления или модели восстановления с неполным протоколированием создается резервная копия журнала транзакций (**BACKUP LOG**).

Базовый синтаксис:

```
BACKUP DATABASE database_name
TO backup_device [, ... n]
[WITH { DIFFERENTIAL | general_WITH_options [, ... n] }]

BACKUP LOG database_name
TO backup_device [, ... n]
[WITH { general_WITH_options | \log-specific_optionspec } [, ... n]]
```

где:

DATABASE – указывает, что должна быть создана резервная копия всей базы данных. Если указан список файлов и файловых групп, то только они включаются в резервную копию. Во время создания полной или разностной резервной копии SQL Server создает резервную копию той части журнала транзакций, которая достаточна для создания согласованной базы данных во время ее восстановления. Во время восстановления резервной копии, созданной с помощью инструкции **BACKUP DATABASE** (резервной копии данных), восстанавливается вся резервная копия. Восстановление на определенный момент времени или к определенной транзакции возможно только для резервной копии журналов.

LOG – указывает только на резервное копирование журнала транзакций. Создается резервная копия части журнала, начинающейся с конца последней успешно созданной копии и заканчивающейся текущим концом журнала. До создания первой резервной копии журнала необходимо создать полную резервную копию базы данных.

database_name – база данных, журнал транзакций и часть данных или все данные, которые подвергаются резервному копированию.

backup_device – указывает логическое или физическое устройство резервного копирования, используемое для создания резервной копии.

DIFFERENTIAL – указывает, что резервная копия базы данных или файлов должна состоять только из тех частей базы данных или файлов, которые были изменены с момента создания последней полной резервной копии.

general_WITH_options – параметры инструкции WITH.

log-specific_optionspec – спецификация параметра журнала.

FORMAT – указывает, что должен быть создан новый набор носителей.

Пример:

BACKUP DATABASE Поставки

TO DISK = N'C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\MSSQL\Backup\Поставки.bak'

WITH FORMAT

Чтобы восстановить базу данных, сделайте следующее:

- 1) В *Обозревателе объектов* выберите узел *Базы данных*, выполните пункт контекстного меню *Восстановить базу данных* (рисунок 52).

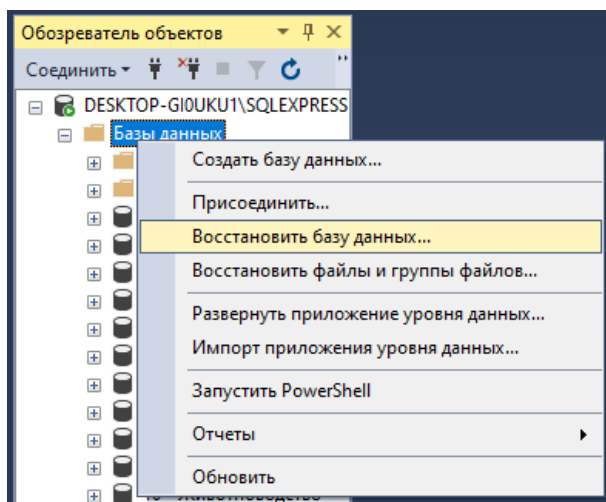



Рисунок 52 – Восстановление резервной копии средствами интерфейса

- 2) В окне *Восстановление базы данных* Выберите *Устройство* и нажмите кнопку , чтобы найти файл резервной копии.

- 3) В окне *Выбор устройств резервного копирования* выберите *Добавить* и перейдите в расположение вашего файла .bak. Выберите файл .bak и затем нажмите *OK*.

- 4) Снова нажмите *OK* в диалоговом окне *Выбор устройств резервного копирования*, чтобы закрыть его. Затем нажмите *OK* (рисунок 53).

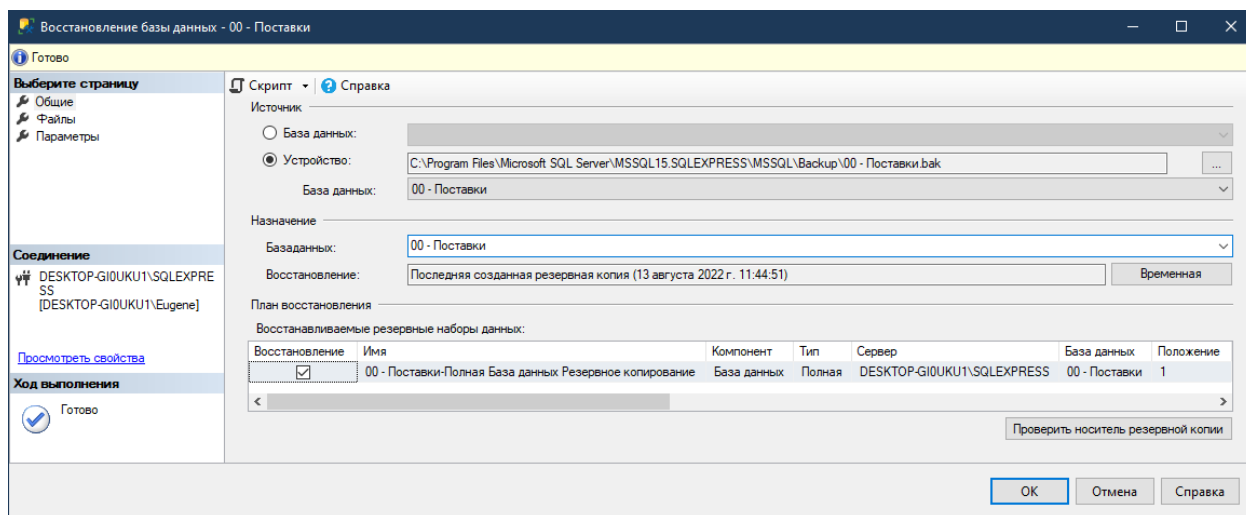


Рисунок 53 – Настройка восстановления БД

Кроме того, резервную копию базы данных можно восстановить с помощью команды T-SQL – **RESTORE**.

В SQL Server инструкция **RESTORE** позволяет выполнить один из сценариев восстановления.

Базовый синтаксис:
<pre>RESTORE DATABASE database_name [FROM backup_device [, ... n] [WITH general_WITH_options [, ... n] RESTORE LOG database_name [FROM backup_device [, ... n] [WITH general_WITH_options [, ... n]</pre>

где:

database_name – база данных, журнал транзакций и часть данных или все данные, которые подвергаются восстановлению.

backup_device – указывает логическое или физическое устройство резервного копирования, используемое для восстановления резервной копии.

general_WITH_options – параметры инструкции WITH.

Пример:
<pre>RESTORE DATABASE Поставки FROM DISK = N'C:\Program Files\Microsoft SQL Server\MSSQL15.MSSQLSERVER\MSSQL\Backup\Поставки.bak'</pre>

Задание:

- 1) Запустить SQL Server Management Studio, выполнить подключение к серверу (параметры соединения получить у преподавателя).
- 2) Создать новую БД (название БД должно соответствовать предметной области).
- 3) Создать таблицы БД в соответствии с заданной предметной областью (типы данных столбцов определить самостоятельно на основании их назначения; ограничения первичного и внешнего ключа выставить согласно диаграмме, прилагаемой к варианту задания, остальные ограничения выбрать самостоятельно).
- 4) Создать диаграмму, включающую таблицы и все необходимые связи между таблицами (должна соответствовать диаграмме, прилагаемой к варианту задания).

В отчет предоставить:

- текст SQL-запроса, выполняющий создание БД, таблиц (включая необходимые ограничения);
- скриншот диаграммы БД (на скриншоте обязательно должно присутствовать имя (номер) компьютера).

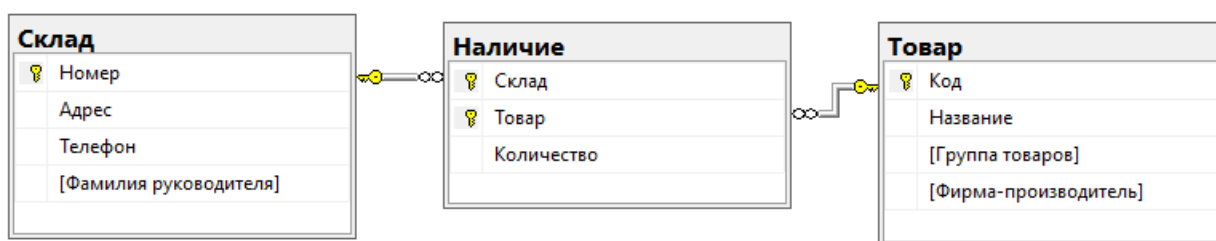
Варианты:

1 подгруппа:

1 вариант – «Склад»

Необходимо хранить информацию о существующих складах (номер склада, адрес, телефон, фамилия руководителя склада), товарах (код, название, группа товара, фирма-производитель), а также о наличии товаров на конкретных складах с указанием их количества.

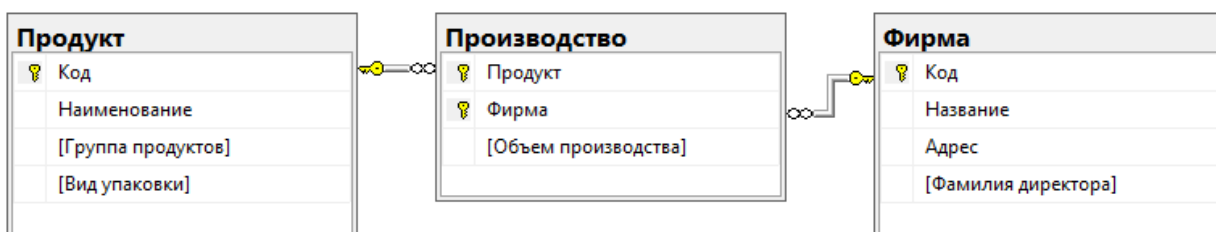
Схема базы данных:



2 вариант – «Производство продуктов питания»

Необходимо хранить информацию о фирмах-производителях продуктов (код фирмы, название фирмы, адрес, фамилия директора), продуктах (код, название, группа продуктов, вид упаковки), а также об объеме производства продуктов фирмами.

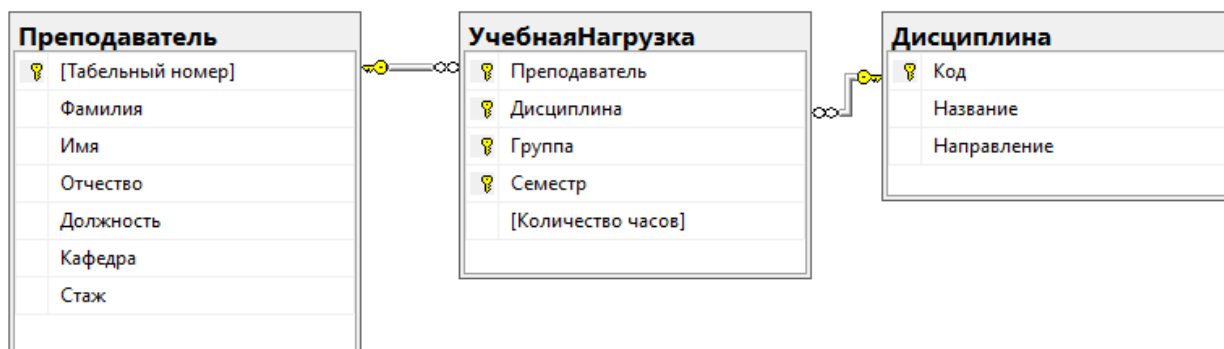
Схема базы данных:



3 вариант – «Учебная нагрузка»

Необходимо хранить информацию о преподавателях (табельный номер, фамилия, имя, отчество, должность, кафедра, стаж), дисциплинах (код, название, направление (гуманитарное, техническое и т.д.)), а также о распределении нагрузки по преподавателям с указанием номера группы студентов, семестра и количества часов.

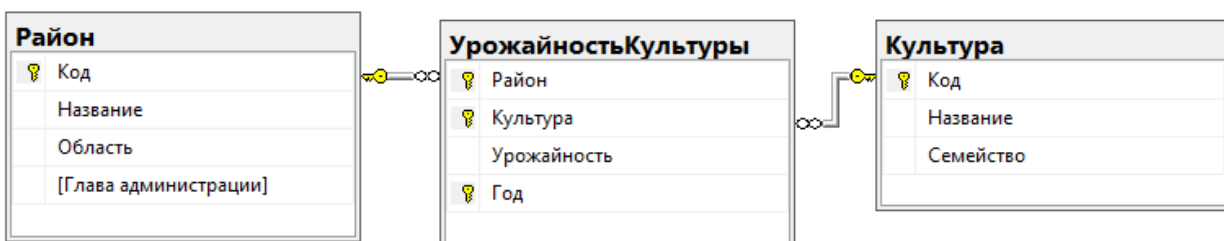
Схема базы данных:



4 вариант – «Растениеводство»

Необходимо хранить информацию о районах различных областей (код, название района, название области, фамилия главы администрации района), культурах (код, название, семейство), а также об урожайности культур в ц/га.

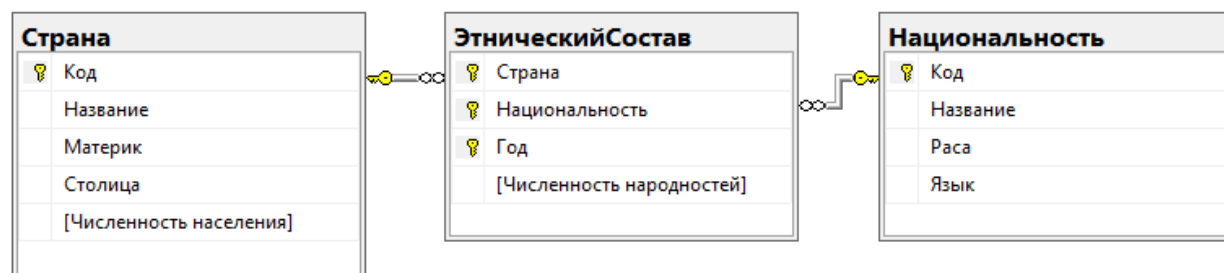
Схема базы данных:



5 вариант – «Этнический состав стран»

Необходимо хранить информацию о национальностях (код, название, раса, язык), странах (код страны, название, материк, столица, численность населения в млн), а также об этническом составе каждой отдельной страны (в тыс.).

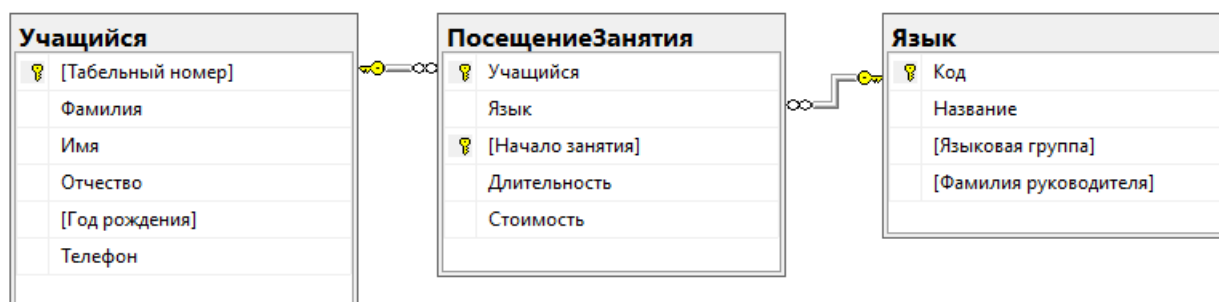
Схема базы данных:



6 вариант – «Курсы иностранных языков»

Необходимо хранить информацию об учащихся (табельный номер, фамилия имя, отчество, год рождения, контактный телефон), предлагаемых языках (код, название, языковая группа, фамилия руководителя направления), а также о посещении учащимися занятий по конкретным языкам с указанием даты начала занятий, длительности и стоимости курсов.

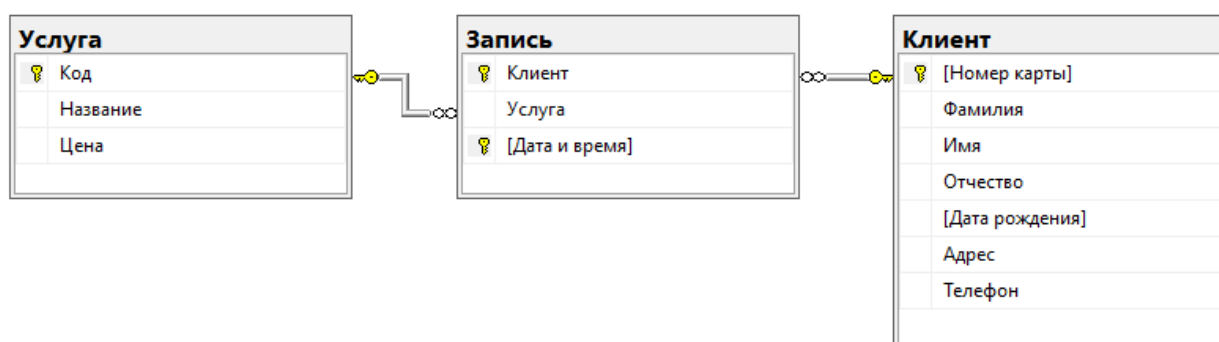
Схема базы данных:



7 вариант – «Услуги парикмахера»

Необходимо хранить информацию о предоставляемых в салоне видах услуг (код, название, цена), клиентах (номер клубной карты, фамилия имя, отчество, адрес, контактный телефон, год рождения), а также о предварительной записи клиентов на услуги с указанием даты и времени посещения.

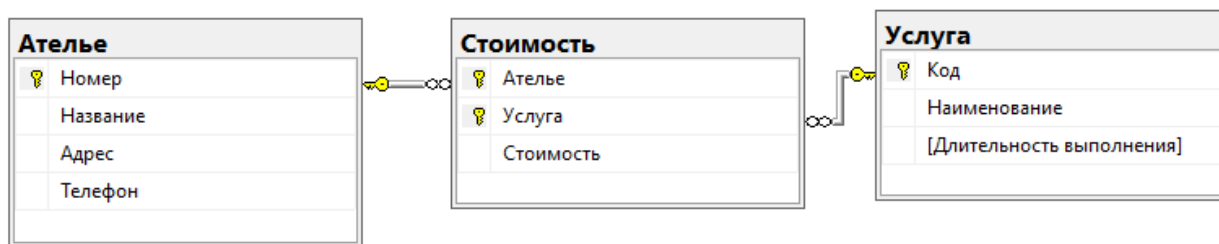
Схема базы данных:



8 вариант – «Ателье по пошиву и ремонту одежды»

Необходимо хранить информацию о существующем ателье (номер ателье, название, адрес, телефон), видах предоставляемых услуг (код, наименование, длительность выполнения), а также о стоимости таких услуг в конкретном ателье.

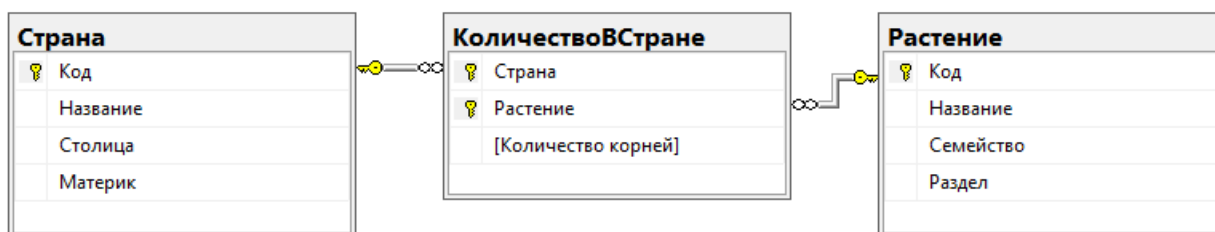
Схема базы данных:



9 вариант – «Красная книга растений»

Необходимо хранить информацию о редких растениях (название, семейство, раздел), странах (название, столица, материк), а также о произрастании растений в отдельных странах с указанием приблизительного количества корней.

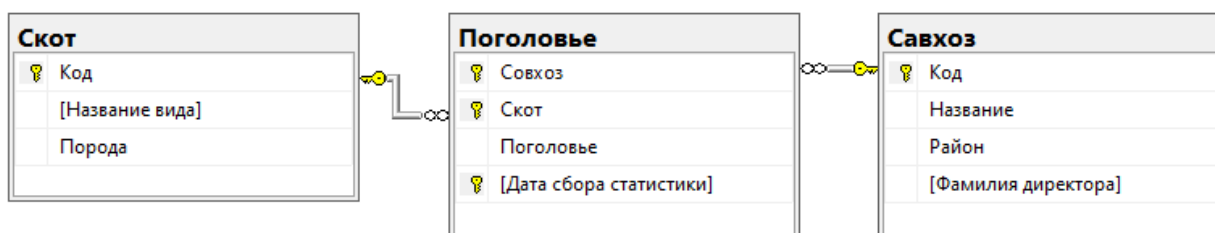
Схема базы данных:



10 вариант – «Животноводство»

Необходимо хранить информацию о совхозах области (код, название совхоза, название района, фамилия председателя), видах скота (код, название вида, порода), а также о поголовье скота в различных совхозах.

Схема базы данных:

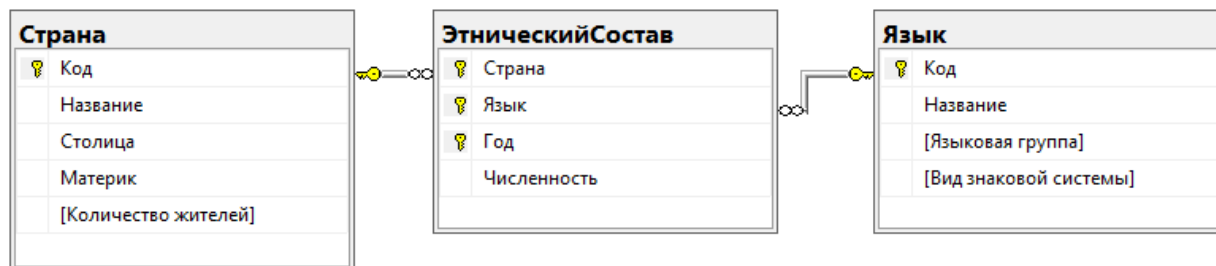


11 вариант – «Языки народов мира»

Необходимо хранить информацию о языках (код, название, языковая группа, вид знаковой системы (кириллица, латиница, иероглифы и т.п.)),

странах (код страны, название, столица, материк, количество жителей), а также об этническом составе каждой отдельной страны с указанием численности населения, говорящего на каждом из языков.

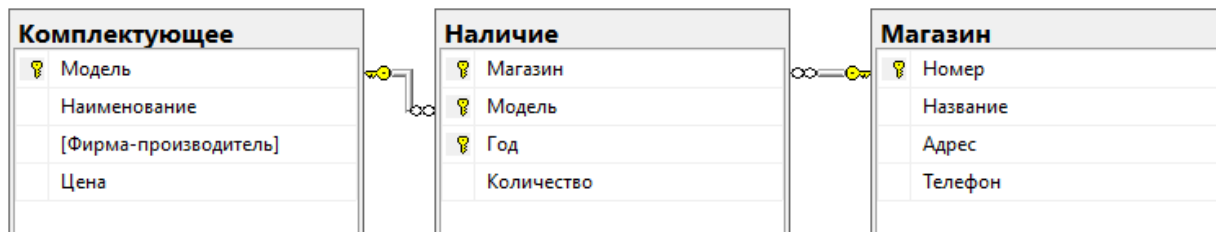
Схема базы данных:



12 вариант – «Сеть компьютерных магазинов города»

Необходимо хранить информацию о существующих магазинах (номер магазина, название, адрес, телефон), комплектующих (модель, наименование вида, фирма-производитель, цена), а также о наличии комплектующих в конкретных магазинах с указанием их количества.

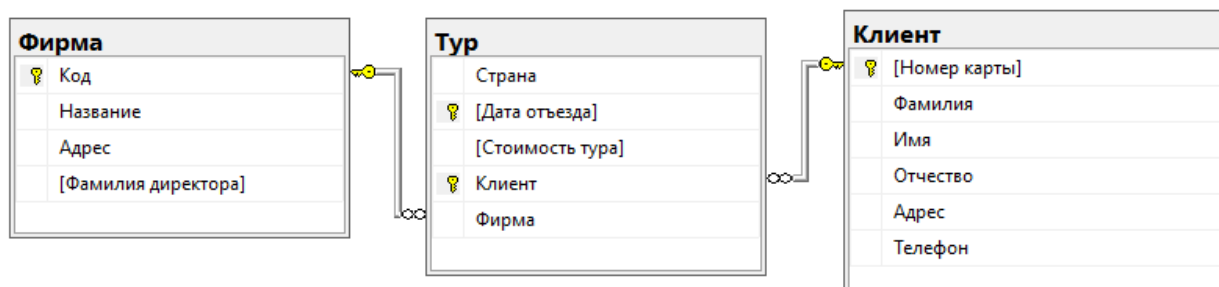
Схема базы данных:



13 вариант – «Туристические фирмы города»

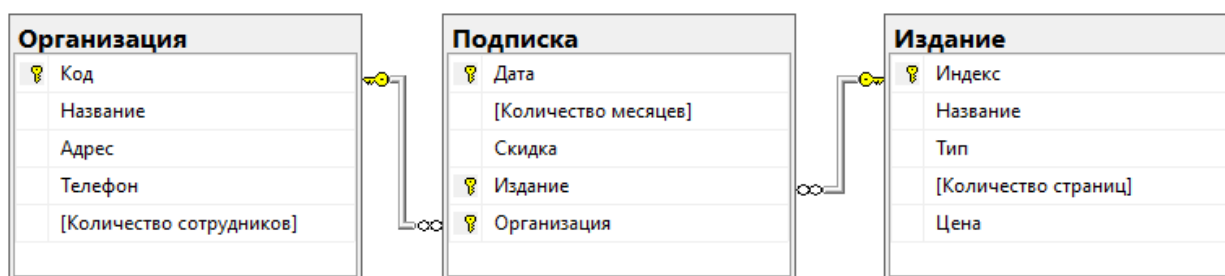
Необходимо хранить информацию о существующих в городе фирмах (код, название, адрес, фамилия директора), постоянных клиентах (номер клубной карты, фамилия, имя, отчество, адрес, телефон), а также о турах, заказанных клиентами с указанием страны, даты отъезда и стоимости тура.

Схема базы данных:

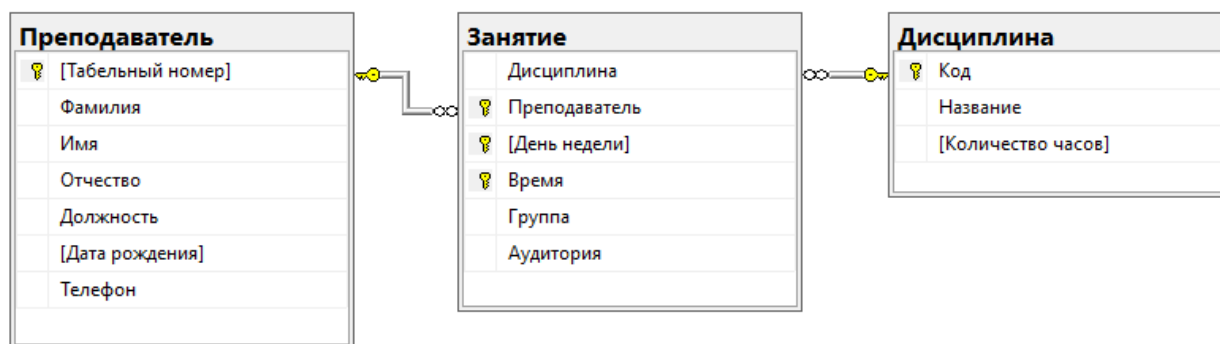


14 вариант – «Подписка на периодические издания города»

Необходимо хранить информацию о существующих организациях города (код, название, адрес, телефон, количество сотрудников), о распространяемых изданиях (индекс, название, тип (газета, журнал и т.п.), количество страниц, цена), а также о подписке организаций на издания с указанием даты начала подписки, количества месяцев и скидки на стоимость подписки в процентах.

Схема базы данных:**15 вариант – «Расписание занятий в ВУЗе»**

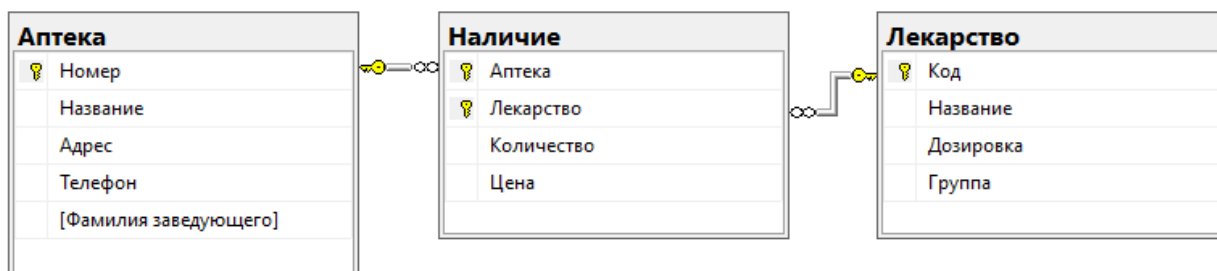
Необходимо хранить информацию о дисциплинах (код, название, количество часов), о преподавателях (табельный номер, фамилия, имя, отчество, должность, дата рождения, телефон), а также о проведении занятий с указанием дня недели, времени, группы и аудитории.

Схема базы данных:**2 подгруппа:****16 вариант – «Справочная служба аптек»**

Необходимо хранить информацию о существующих аптеках города (номер, название, телефон, адрес, фамилия заведующего), о поставляемых в аптеки лекарствах (код, название, дозировка, группа (антибиотики,

антисептики, сульфаниламиды, анальгетики и т.п.)), а также о наличии лекарств в аптеках с указанием количества упаковок и цены.

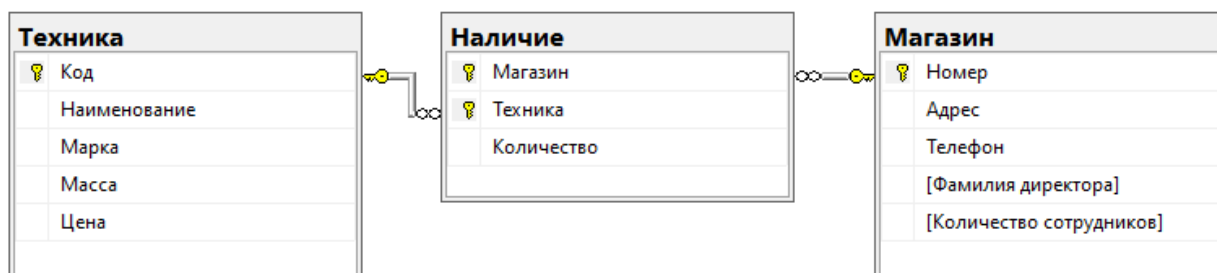
Схема базы данных:



17 вариант – «Сеть магазинов по продаже бытовой техники»

Необходимо хранить информацию о существующих магазинах (номер, адрес, телефон, фамилия директора, количество сотрудников), о продаваемой технике (код, наименование, марка, масса, цена), а также о наличии техники в магазинах с указанием количества.

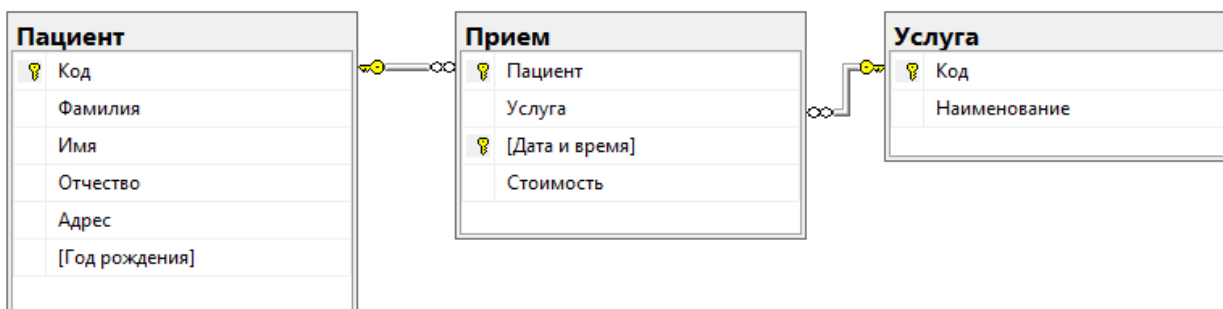
Схема базы данных:



18 вариант – «Стоматологическая поликлиника»

Необходимо хранить информацию о пациентах (код пациента, фамилия, имя, отчество, адрес, год рождения), о предоставляемых видах услуг (код, наименование), а также об оказании услуг пациентам с указанием стоимости, даты и времени приёма.

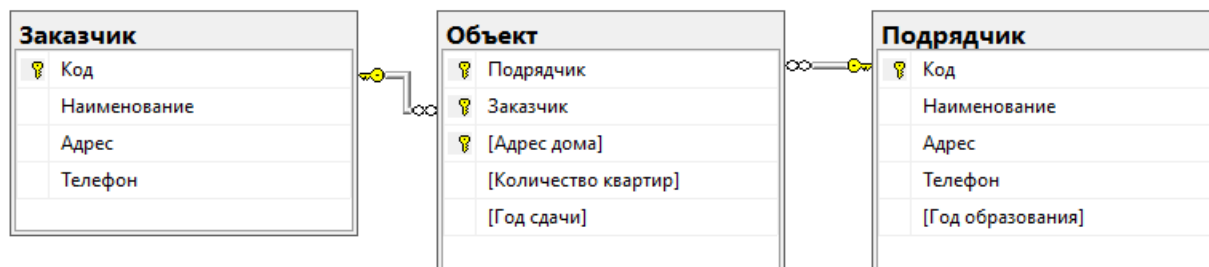
Схема базы данных:



19 вариант – «Строительство жилья»

Необходимо хранить информацию о существующих подрядчиках (код, название юридического лица, адрес, телефон, год образования фирмы), о заказчиках (код, название юридического лица, адрес, телефон), а также о строящихся зданиях с указанием адреса дома, кол-ва квартир и года сдачи.

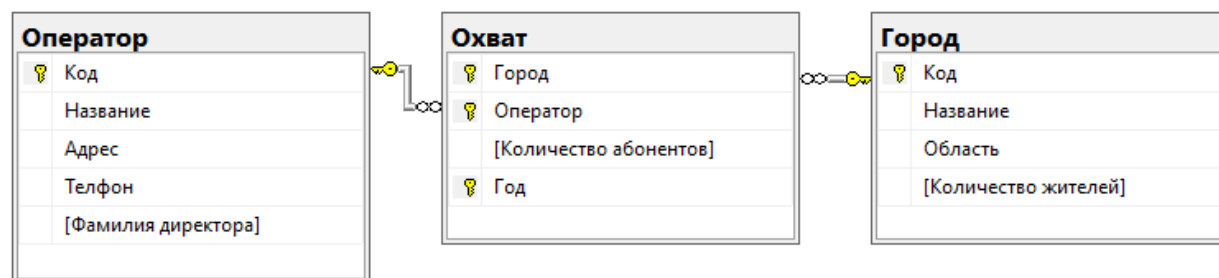
Схема базы данных:



20 вариант – «Сотовые и интернет компании России»

Необходимо хранить информацию об операторах сотовой и интернет связи (название, адрес центрального представительства, телефон, фамилия генерального директора), города (код города, название города, название области, количество жителей), а также о количестве абонентов каждой сотовой компании в отдельных городах.

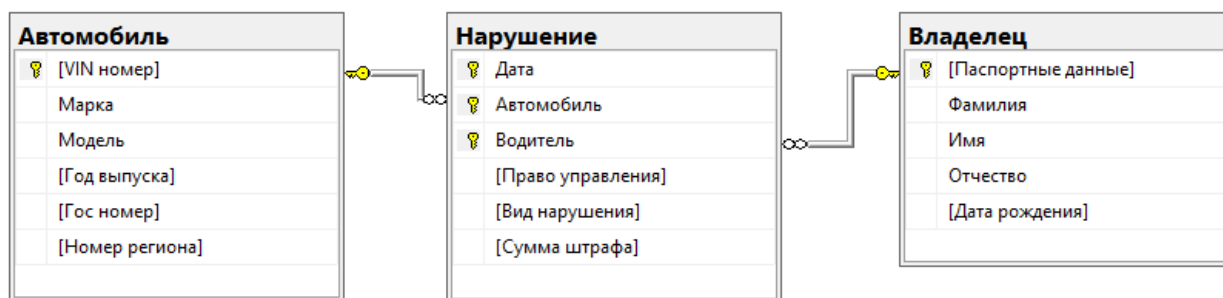
Схема базы данных:



21 вариант – «Учет нарушений правил дорожного движения»

Необходимо хранить информацию об автомобилях (VIN-номер, марка, модель, год выпуска, гос номер, номер региона), владельцах автомобилей (паспортные данные, фамилия, имя, отчество, дата рождения), а также о совершенных нарушениях с указанием даты, вида нарушения и суммы штрафа.

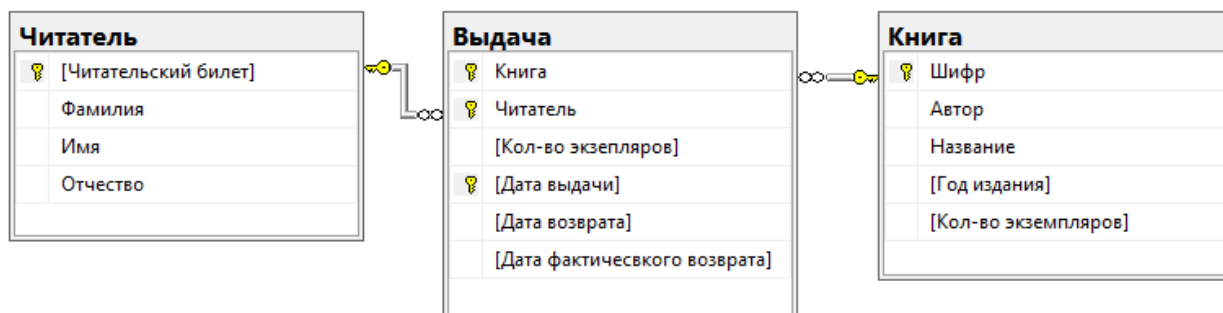
Схема базы данных:



22 вариант – «Библиотека»

Необходимо хранить информацию о читателях (номер читательского билета, фамилия, имя, отчество), книгах (шифр, автор, название, год издания, количество экземпляров), а также о выдаче книг читателям с указанием дат выдачи и возврата.

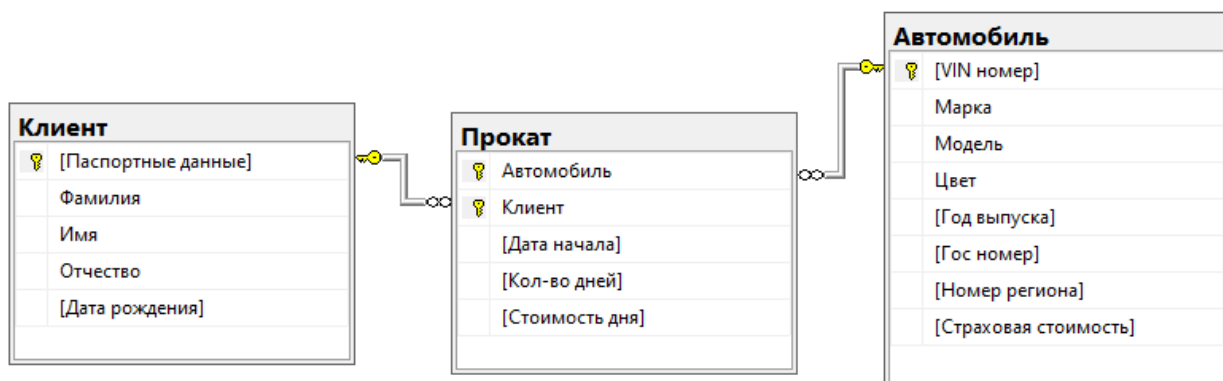
Схема базы данных:



23 вариант – «Прокат автомобилей»

Необходимо хранить информацию о клиентах (паспортные данные, фамилия, имя, отчество, дата рождения), автомобилях (VIN-номер, марка, модель, цвет, год выпуска, гос номер, номер региона, страховая стоимость), а также о прокатах с указанием даты, количества дней и стоимости одного дня.

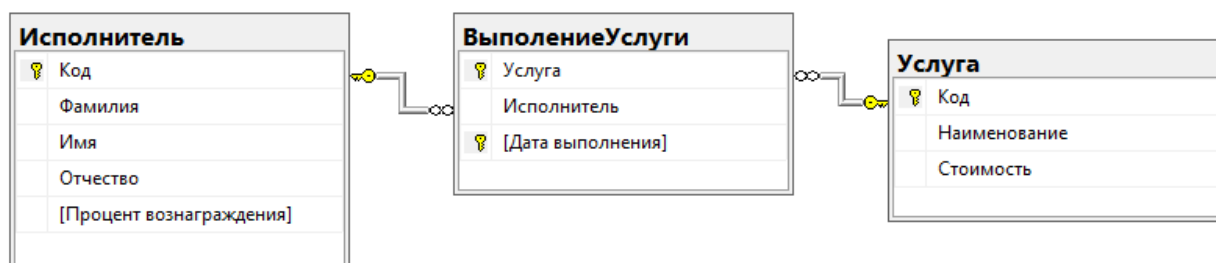
Схема базы данных:



24 вариант – «Ремонт компьютеров»

Необходимо хранить информацию о предоставляемых услугах в фирме по ремонту компьютеров (код услуги, наименование, стоимость), сотрудниках фирмы, исполняющих услуги (код, фамилия, имя, отчество, процент вознаграждения), а также о выполненных заказах (услуга, исполнитель, дата выполнения).

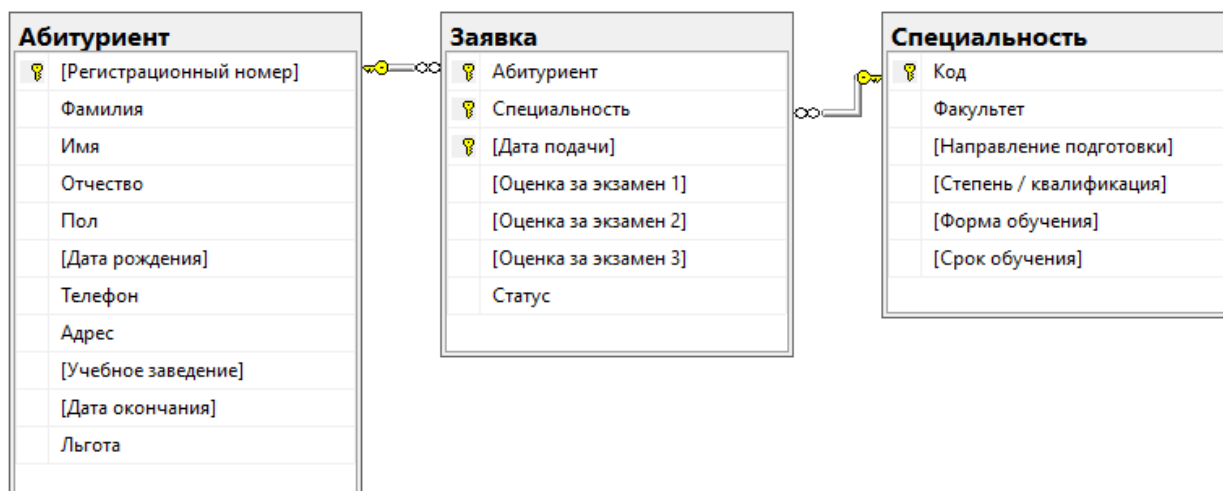
Схема базы данных:



25 вариант – «Приемная комиссия ВУЗа»

Необходимо хранить информацию об абитуриентах (регистрационный номер, фамилия, имя, отчество, пол, дата рождения, телефон, адрес, наименование оконченного учебного заведения, дата окончания, имеющаяся льгота), имеющих специальности (код, факультет, направление подготовки, степень / квалификация (бакалавриат, специалитет, магистратура, аспирантура), форма обучения (очная, заочная, очно-заочная), срок обучения в месяцах), а также заявках, поданных абитуриентом на поступление с указанием даты подачи, оценках за вступительных экзамена и статуса заявки (зачислен, не зачислен).

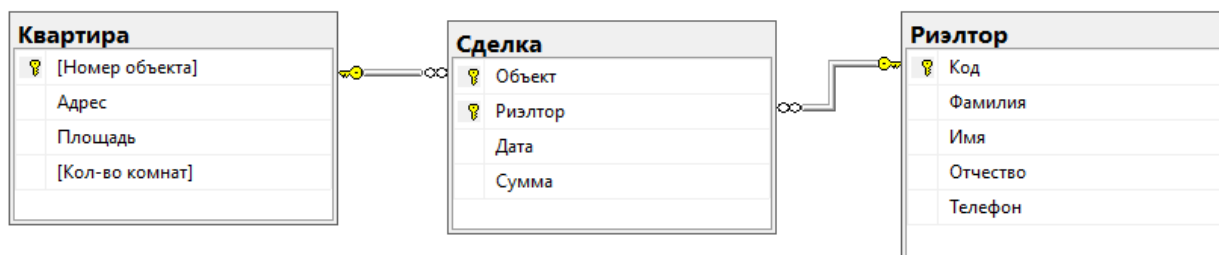
Схема базы данных:



26 вариант – «Учет сделок с недвижимостью»

Необходимо хранить информацию о квартирах (номер объекта, адрес, площадь, количество комнат), риэлторах (код, фамилия, имя, отчество, телефон), а также проведенных риэлторами сделках (объект, дата, сумма).

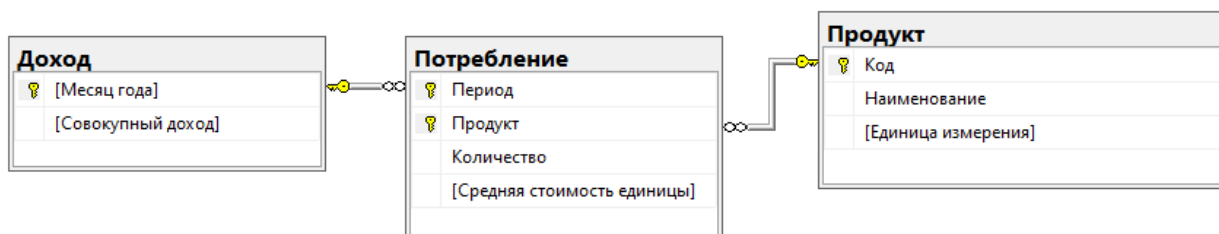
Схема базы данных:



27 вариант – «Потребительская корзина»

Необходимо хранить информацию о продуктах (код, наименование, единица измерения), доходах семьи (месяц и год (период), совокупный доход за период), а также информацию для анализа уровня жизни в семье, зависящего от соотношения доходов семьи и цен на потребляемые продукты.

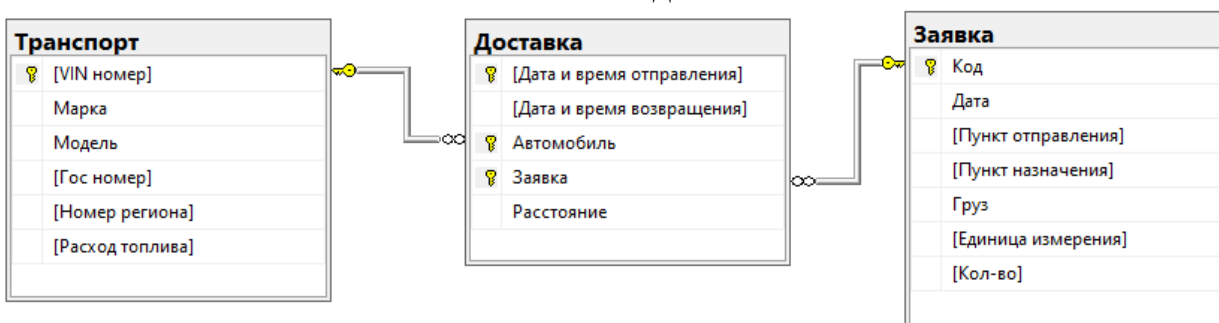
Схема базы данных:



28 вариант – «Грузоперевозки»

Необходимо хранить информацию о заявках от других организаций на перевозку различных грузов (код, дата, пункт отправления, пункт назначения, груз, единица измерения, количество груза), об используемом на предприятии транспорте (VIN-номер, марка, модель, гос номер, номер региона, расход топлива), а также о совершённых доставках (дата и время отправления, дата и время возвращения, используемый автомобиль, номер заявки, пройденное расстояние).

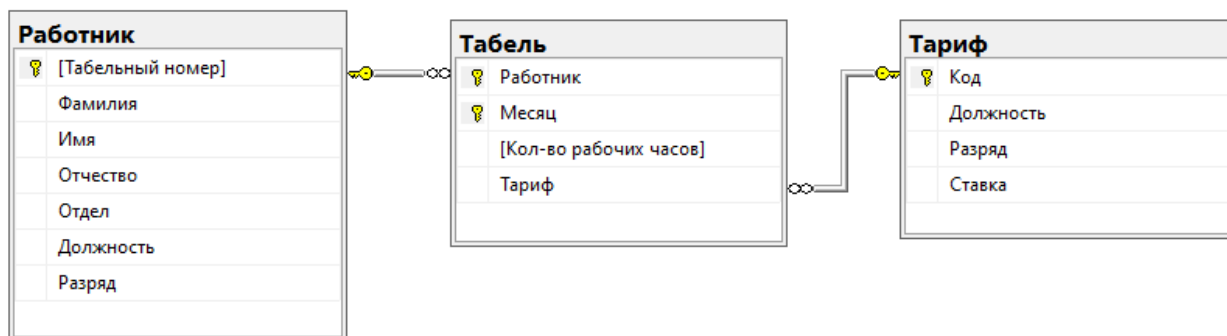
Схема базы данных:



29 вариант – «Бухгалтерия»

Необходимо хранить информацию о тарифах (код, должность, разряд, ставка), работников предприятия (табельный номер, фамилия, имя, отчество, отдел, должность, разряд), а также табелях для начисления з/п(работник, месяц, количество рабочих часов, тариф).

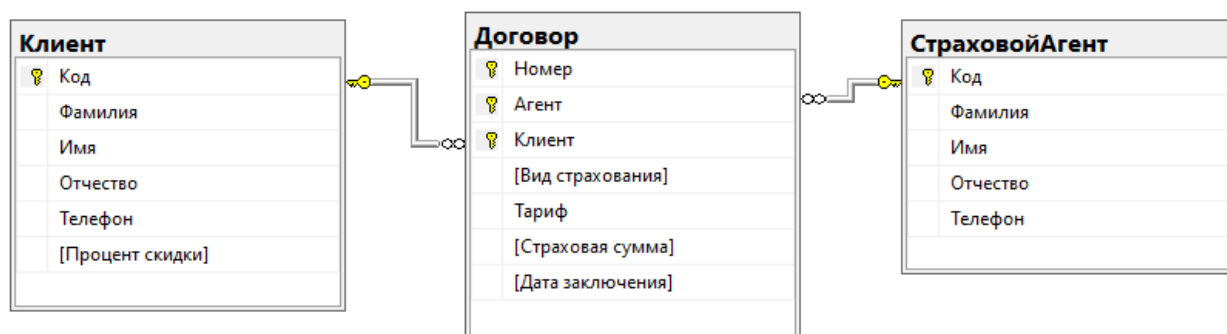
Схема базы данных:



30 вариант – «Учет договоров страхования»

Необходимо хранить информацию о клиентах страхового агентства (код, фамилия, имя, отчество, телефон, процент скидки), страховых агентах (код, фамилия, имя, отчество, телефон), а также о заключенных ими договорами страхования (номер, вид страхования, тариф, страховая сумма, дата заключения).

Схема базы данных:



Контрольные вопросы:

- 1) Какие основные действия можно выполнять с помощью приложения Management Studio.
- 2) Каким образом выполняется подключение Management Studio к серверу?
- 3) Перечислите параметры, которые указываются при подключении к серверу.

- 4) Как задается имя сервера при подключении?
- 5) Назовите основные панели приложения Management Studio?
- 6) Что представляет собой панель Обозреватель объектов?
- 7) Для чего предназначен Обозреватель решений?
- 8) Что такое контекст выполнения запроса? Для чего используется команда USE?
- 9) Как создать новую БД?
- 10) Опишите базовый синтаксис инструкции CREATE DATABASE.
- 11) Какие файлы входят в состав БД?
- 12) Перечислите основные параметры файлов БД.
- 13) Что включает файл с расширением mdf?
- 14) Что включает файл с расширением ldf?
- 15) Как удалить существующую БД?
- 16) Опишите базовый синтаксис инструкции DROP DATABASE.
- 17) Как создать новую таблицу в БД?
- 18) Опишите базовый синтаксис инструкции CREATE TABLE.
- 19) Какие ограничения можно задать для атрибута таблицы?
- 20) Для чего необходимо ограничение первичного ключа PRIMARY KEY?
- 21) Для чего необходимы связи между таблицами?
- 22) Опишите базовый синтаксис ограничения FOREIGN KEY.
- 23) Какие опции могут использоваться при задании ограничения внешнего ключа?
- 24) Как удалить существующую таблицу?
- 25) Опишите базовый синтаксис инструкции DROP TABLE.
- 26) Как удалить связь между таблицами?
- 27) Для чего предназначены диаграммы? Как их создавать?
- 28) Что такое пакет команд? Для чего используется команда GO?
- 29) Какими способами можно осуществить перенос БД?
- 30) Какое расширение имеет файл резервной копии БД?

Порядок оценивания:

Оценка «**Отлично**» выставляется, если были выполнены все пункты задания, отчёт содержит всю необходимую информацию по проделанной работе, оформлен согласно требованиям, работа сдана и защищена в срок, в ходе защиты даны правильные ответы на 5 вопросов.

Оценка «**Хорошо**» выставляется, если были выполнены все пункты задания, отчёт содержит всю необходимую информацию по проделанной работе, оформлен согласно требованиям, работа сдана и защищена в срок, в ходе защиты даны правильные ответы на 4 вопроса.

Оценка «**Удовлетворительно**» выставляется, если были выполнены все пункты задания, отчёт содержит минимально необходимую информацию по проделанной работе, работа сдана и защищена с нарушением сроков, в ходе защиты даны правильные ответы на 2-3 вопроса.

Оценка **«Неудовлетворительно»** выставляется, если были выполнены не все пункты задания, отчёт содержит информацию, не позволяющую оценить ход выполнения работы, работа сдана с нарушением сроков, не защищена, либо в ходе защиты даны правильные ответы менее чем на 2 вопроса.

Работы, получившие оценку «Удовлетворительно» и выше являются засчитанными.

Работы, получившие оценку «Неудовлетворительно» являются обязательными к доработке и повторной сдаче. Итоговая оценка при повторной сдаче работы будет снижена на 1 балл.