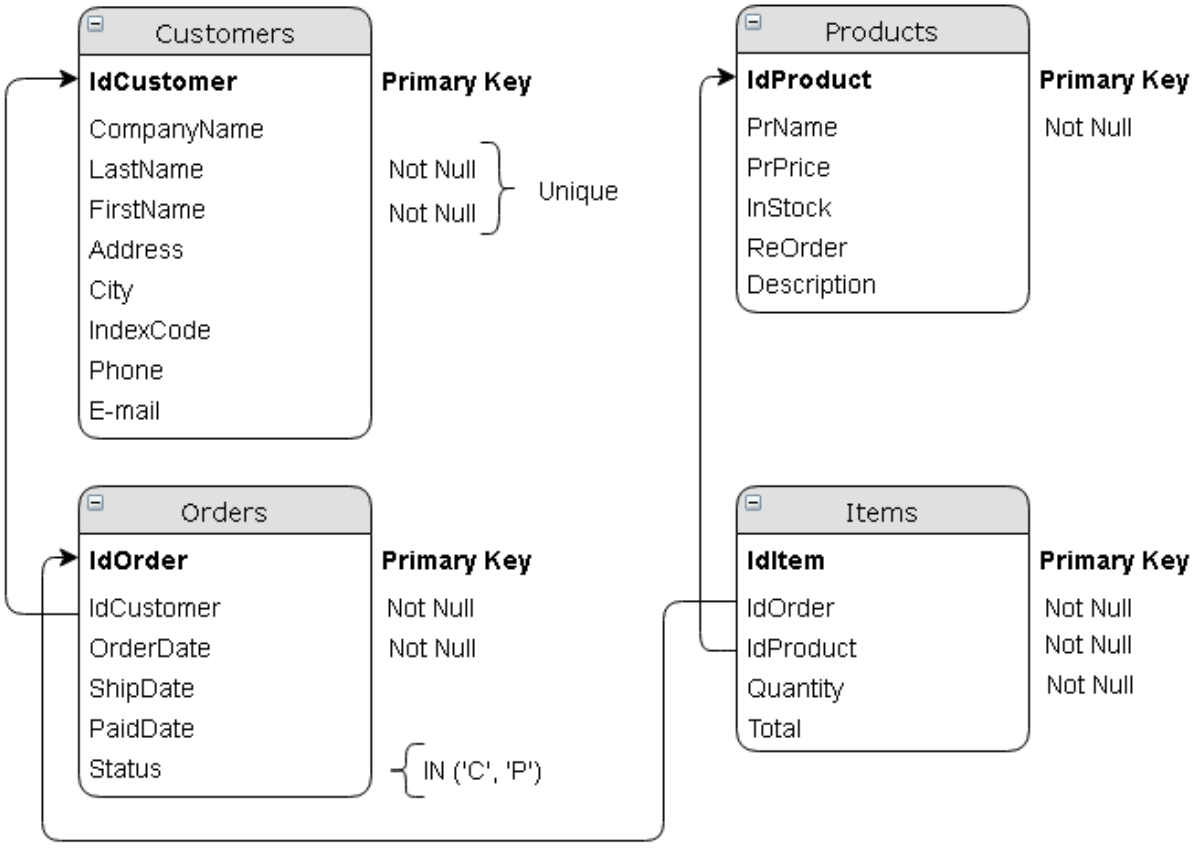
ЛАБОРАТОРНАЯ РАБОТА №2 СОЗДАНИЕ БАЗЫ ДАННЫХ В СУБД MS SQL SERVER. ОСНОВЫ ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ SQL: ДОБАВЛЕНИЕ, ИЗМЕНЕНИЕ и УДАЛЕНИЕ ДАННЫХ.

Цель: изучить основные принципы организации MS SQL Server, получить теоретические и практические навыки создания базы данных в СУБД MS SQL Server, изучить основные понятия и операторы, научиться работать в среде SQL Server Management Studio, преобразовать базу данных MS Access в базу MS SQL Server, сформировать знания и умения по программирования на языке SQL, приобрести практические навыки работы со средствами языка SQL для обновления, удаления и вставки данных в БД.

Содержание лабораторной работы:

1. Изучить теоретические сведения лабораторной работы.
2. Создать базу данных в среде MS SQL Server Management Studio.
3. Создать таблицы (с помощью запроса и графического способа), первичные ключи, обеспечить ссылочную целостность базы данных (ограничение по внешним ключам) в соответствии с приведенным ниже рисунком.



4. Создать диаграмму БД.
5. Создать ограничение на проверку данных (задание и пример на стр. 13).
6. Создать значение по умолчанию для поля OrderDate (пример на стр. 14).
7. **Дополнительное задание (необязательное):** преобразование базы данных MS ACCESS в базу MS SQL SERVER.
8. Сохранить БД
9. Заполнить каждую таблицу базы данных (минимум 15 записей в каждую таблицу) с помощью команды INSERT.

   ПРИМЕЧАНИЕ: для просмотра содержания таблиц необходимо использовать

запрос `SELECT *FROM <название таблицы>`

10. Изучить команду для обновления данных в таблицах.
11. Изучить команду для удаления данных из таблиц.
12. Создать скрипт БД.
13. Защитить лабораторную работу.
    - Любой вопрос по выполнению лабораторной работы;
    - Любой контрольный вопрос.

Краткий вспомогательный материал лабораторной работы №2

Microsoft SQL Server – реляционная СУБД. Под архитектурой базы данных будем понимать ее основные составляющие и принципы их взаимодействия. Архитектуру рассмотрим на двух уровнях абстракции (см. рис. 1.1):

- Логический уровень (логическая архитектура базы данных) – данный уровень рассмотрения подразумевает изучение базы данных на уровне ее содержательных объектов.
- Физический уровень (физическая архитектура базы данных) – любая база данных представляет собой набор файлов, которые хранятся на жестком диске одного или нескольких компьютеров. Данный уровень рассмотрения подразумевает изучение базы данных на уровне файлов.

Рисунок 1.1 – Уровни абстракций MS SQL Server

Компоненты логического уровня.

Tables (Таблицы). Таблицы базы данных предназначены для хранения данных. Подразделяются на две категории: User (пользовательские) и System (системные). Пользовательские таблицы хранят данные из предметной области, системные – служебную информацию.

Views (Представления). По сути своей являются «виртуальными таблицами». Представление - это поименованный запрос SELECT.

Indexes (Индексы). Существуют для поддержания вместе с данными информации об их упорядоченности по различным критериям, что позволяет существенно повысить производительность некоторых операций, в частности поиска данных. Индексы существуют непосредственно вместе с таблицами и не имеют смысла сами по себе. Индексирование может быть выполнено по одному или нескольким столбцам и произведено в любой момент.

Diagrams (Диаграммы). Специальные визуальные средства изучения и описания структуры базы данных. При помощи диаграмм можно изучать структуру таблиц и связи между ними, а также вносить в схему БД изменения.

Keys (Ключи). Подобно индексам, ключи не существуют сами по себе. Ключ – один

из типов ограничений целостности.

Defaults (Умолчания). Не существуют отдельно от таблиц. Умолчания определяют, какие значения будут подставлены в поле данных при добавлении строки, если значение не задано явно.

Rules (Правила). Механизм, предназначенный для установления ограничений на диапазон возможных значений поля таблицы или нового типа данных, определяемого пользователем.

Constraints (Ограничения целостности). Определяют диапазон возможных значений полей таблицы.

Stored Procedures (Хранимые процедуры). Поименованный набор команд SQL. Хранимые процедуры располагаются на сервере вместе с базой данных и могут запускаться различными пользователями, имеющими соответствующие права.

Triggers (Триггеры). Не существуют отдельно от таблицы. Триггер – специальный вид хранимой процедуры, предназначенный для производства действий при выполнении операций вставки, удаления, редактирования данных или из.

User-defined data types, UDDT (Определяемые пользователем типы данных). Предоставляет специальный аппарат для создания пользовательских типов данных.

User-defined functions, UDF (Определяемые пользователем функции). Представляет собой набор команд SQL, сохраненных в специальном виде. Наряду с пользовательскими функциями, существует набор стандартных функций Microsoft SQL Server, которыми можно пользоваться в работе с базой данных.

Users (Пользователи). Определяет список пользователей базы данных. Служит для работы механизма защиты данных.

Roles (Роли). Роли пользователей – важный механизм для организации разграничения доступа. Пользователю может быть назначена одна или больше ролей.

Физический уровень.

На физическом уровне база данных в Microsoft SQL Server представляет собой набор файлов с данными и набор файлов, содержащий журнал транзакций. Каждая база данных состоит из собственного уникального набора файлов, что существенно повышает надежность системы.

Рассмотрим физическую структуру на уровне ее основных абстракций. Основными элементами структуры являются файлы и группы файлов. Все файлы делятся на два типа:

- Файлы данных содержат данные и различные объекты, входящие в логическую структуру базы данных, такие, как триггеры, хранимые процедуры и т.д.
- Файлы журнала транзакций содержат сведения о ходе выполнения транзакций. В них содержится информация о том, когда началась и закончилась транзакция, какие ресурсы, кем и когда были заблокированы или разблокированы и т.д.

Каждая база данных содержит как минимум два файла: файл с данными и файл журнала транзакций. Имеется возможность создания при необходимости новых файлов с данными и для журнала транзакций. При этом можно указать, какие данные в каком файле хранить.

Файлы с данными бывают двух типов:

- Основной или главный файл (Primary File) присутствует в любой базе данных. Если в базе данных всего один файл с данными, то он является главным.

Главный файл содержит метаданные – данные о том, как устроена база данных (системные таблицы и т.д.). Кроме системной информации, главный файл может хранить и пользовательские данные. Главный файл имеет расширение mdf (Master Data File).

- Вторичный или дополнительный файл (Secondary File) – дополнительный файл для хранения пользовательских данных. Не содержит системной информации. Такие файлы могут создаваться администратором, например, при больших объемах базы, расширение может быть создано на другом жестком диске. Имеет расширение ndf (secoNdary Data File).

Файлы журнала транзакций содержат информацию о транзакциях и бывают только одного типа – (Transaction Log File). Имеют расширение ldf (Log Data File).

Для удобства администрирования и с целью повышения эффективности обработки файлы с данными могут объединяться в группы. По умолчанию существует одна группа Primary File Group – основная группа файлов и все файлы добавляются в нее. При необходимости возможно создание пользовательских групп файлов (User File Group). Существует также группа файлов по умолчанию (Default File Group), содержащая объекты базы данных, не приписанные явно ни к одной группе.

Для работы с файлами в Microsoft SQL Server применяется принцип постраничной организации файлов. В его рамках файл рассматривается как набор последовательно расположенных страниц с данными. Таким образом, страница является минимальным блоком (квантом) данных, с которым может идти работа (физической записью). Размер страницы в Microsoft SQL Server составляет 8 Кб. Это позволяет использовать разные алгоритмы буферизации и существенно повысить эффективность работы с файлами. Постраничная организация применяется лишь для файлов с данными. Файлы журнала транзакций не организованы в страницы, т.к. содержат последовательно расположенные записи о происходящих транзакциях.

Страницы файлов данных могут использоваться не только для хранения собственно данных, но и для представления различной служебной информации. Так, страницы могут быть одного из следующих типов:

- *Data.* Хранение данных таблиц, (Image, Text, nText).
- *Index*. Хранение индексов таблиц и представлений.
- *Text/Image*. Хранение «тяжелых» данных Image, Text, nText, занимающих обычно более одной страницы, место сразу же выделяется постранично.
- *Global Allocation Map (GAM).* Служебный тип страницы. Содержит информацию об использовании групп страниц.
- *Page Free Space (PFS)*. Содержит сведения о расположении свободного пространства в страницах файла.
- *Index Allocation Map (IAM).* Содержит информацию о группах страниц, которые используются таблицами.

Каждая страница имеет внутреннюю архитектуру. У любой страницы присутствует заголовок, содержащий служебную информацию. Структура заголовка одинакова для страниц разных типов.

Для уменьшения фрагментации базы данных SQL Server использует *экстент*. Экстент содержит 8 последовательных страниц и имеет размер 64 Кб. Под фрагментацией понимается факт разброса по всему файлу базы данных страниц, соответствующих одной и той же таблице или индексу. Во избежание фрагментации SQL Server выделяет пространство сразу экстентами. Таким образом, по крайней мере восемь страниц расположены физически рядом друг с другом, что упрощает для сервера поиск. SQL Server ис-

пользует следующие типы экстентов.

- *Однородные экстенты* являются собственностью одного объекта. К примеру, если одна таблица владеет всеми восемью страницами расширения, оно считается однородным.
- *Смешанные экстенты* используются для объектов, которые слишком малы, чтобы занять собой все восемь страниц. В данном случае SQL Server распределяет отдельные страницы экстента для разных объектов.

Утилита SQL Server Management Studio

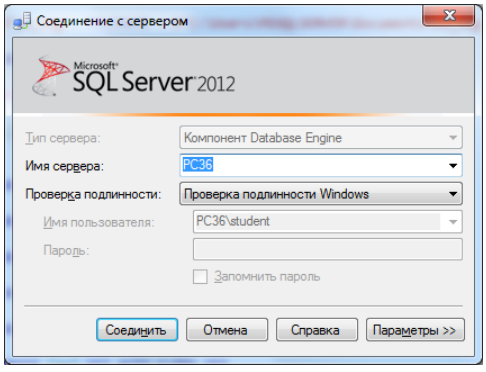
Подавляющую массу задач администрирования SQL Server можно выполнить в графической утилите SQL Server Management Studio. В ней можно создавать базы данных и все ассоциированные с ними объекты (таблицы, представления, хранимые процедуры и др.). Здесь вы можете выполнить последовательности инструкций Transact-SQL (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 « SQL Server Management Studio».

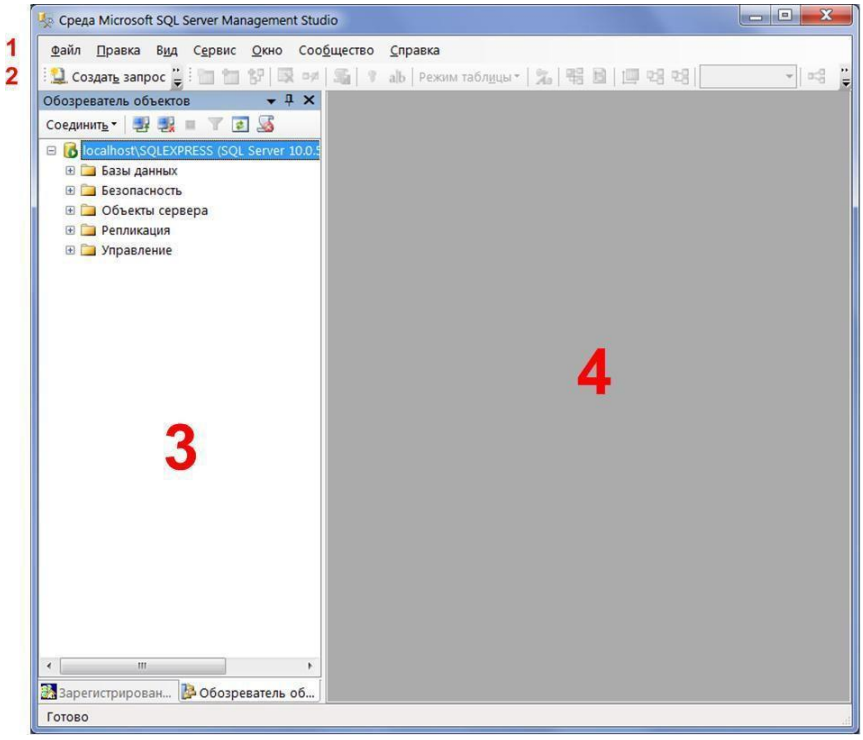
Подключение к серверу

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

- Тип сервера. Здесь следует выбрать, к какой именно службе необходимо подключится. Оставьте вариант «Компонент Database Engine».
- 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.



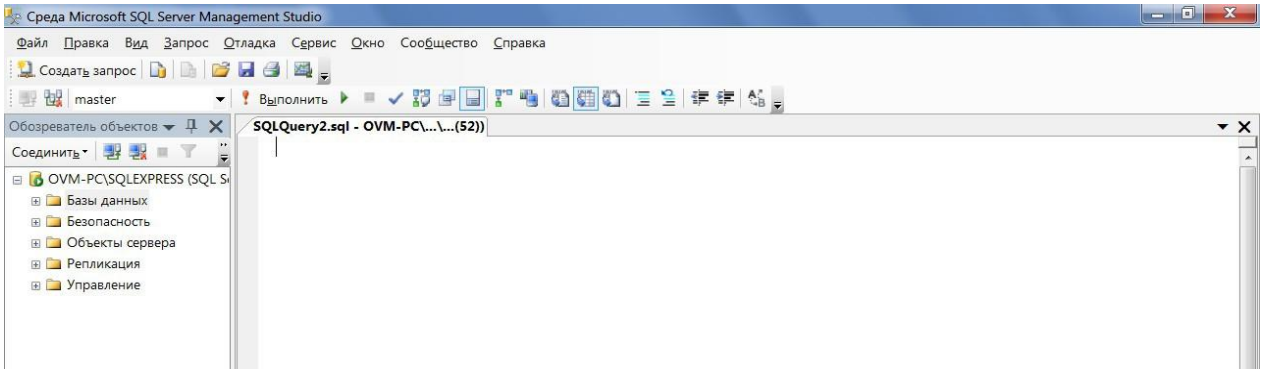
После подключения экземпляр сервера будет отображаться на панели «Обозреватель объектов».



Данное окно имеет следующую структуру:

1. Оконное меню – содержит полный набор команд для управления сервером и выполнения различных операций.
2. Панель инструментов – содержит кнопки для выполнения наиболее часто производимых операций. Внешний вид данной панели зависит от выполняемой операции.
3. Панель «Обозреватель объектов» – это панель с древовидной структурой, отображающая все объекты сервера, а также позволяющая производить различные операции, как с самим сервером, так и с БД. Обозреватель объектов является основным инструментом для разработки БД.
4. Рабочая область. В рабочей области производятся все действия с БД, а также отображается её содержимое.

Нажатие кнопки «Создать запрос» приводит к открытию окна запросов.



Для выполнения запроса, введенного в этом окне, нужно нажать на кнопку «Выполнить».

Для создания базы данных вам нужно лишь создать первичный файл данных и файл журнала транзакций. Задачу можно выполнить двумя способами:

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

Пример создания базы посредством кода:

```
IF DB_ID ('Sales') IS NULL CREATE DATABASE Sales
```

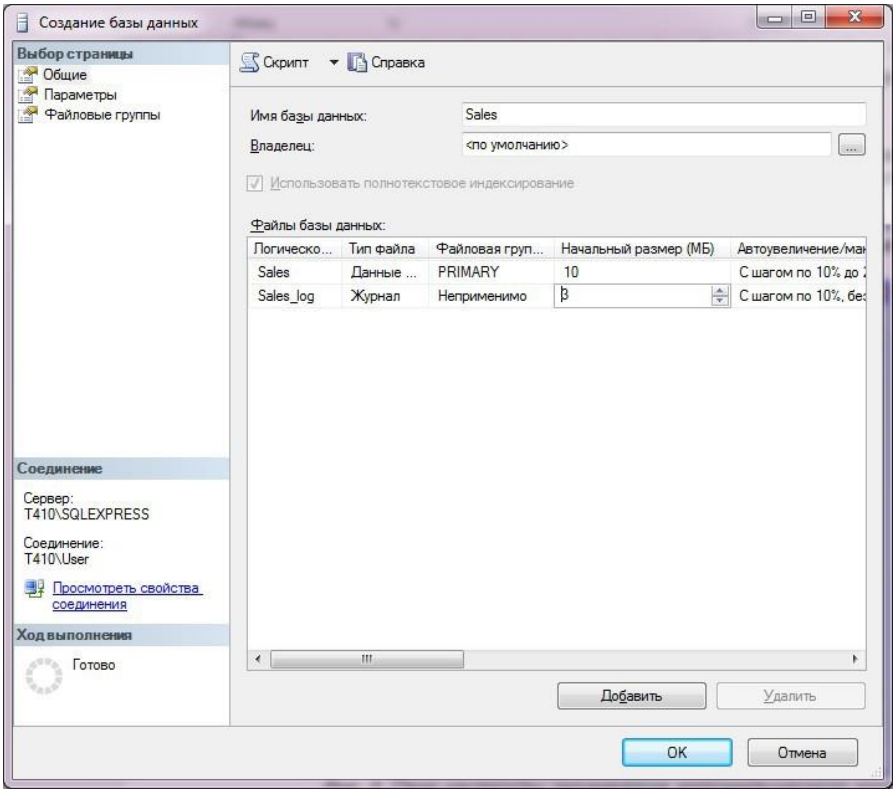
Если базы данных с именем `Sales` не существует, приведенный код создаст новую БД. Функция `DB_ID` принимает имя базы данных в качестве входного параметра и возвращает внутренний ID базы данных. Если БД с именем входного параметра не существует, функция возвращает значение NULL. Это простой способ проверить наличие заданной БД.

В инструкции `CREATE DATABASE` можно использовать установочные параметры файла для задания его местоположения и начального размера. После выполнения данного запроса в окне среды разработки SQL Server Management Studio на панели обозревателя объектов в папке «Базы данных» появится новая БД .

В лабораторной работе будем использовать графический метод.

Создадим базу данных Sales:

1. В окне *Обозреватель объектов* найдите и раскройте папку *Базы данных*.
2. Щелкните правой кнопкой мыши на папке *Базы данных* и выберите команду *Создать базу данных*.



3. В левой панели вы увидите список Выбор страницы. Перейдите во вкладку *Общие* и введите следующую информацию:
   Имя базы данных: **Sales**
   Владелец: **<default>**
4. Перейдите во вкладку *Общие* и введите следующую информацию:
   Параметры сортировки (Collation): ***<server*** **default>**
   Модель восстановления (Recovery Model): **Full**
5. В окне *Файлы базы данных (Database Files)* вы должны увидеть две строки: одну для файла данных и одну для файла журнала. Измените исходный размер файла данных на 10.
6. В столбце *Автоувеличение (Autogrowth)* для файла данных щелкните на кнопке с многоточием, установите переключатель в положение *Ограниченное (Restricted File*

Growth) и измените максимальный размер на 100. Если вы оставите настройку *Без ограничений (Unrestricted File Growth)*, файл данных может заполнить весь жесткий диск, что приведет к неработоспособности компьютера, если другие программы находятся на том же диске (например, сама ОС Windows) (рис. 2).

7. В столбце *Автоувеличение (Autogrowth)* для файла журнала щелкните на кнопке с многоточием, установите переключатель в положение *Ограниченное (Restricted File Growth)*, задайте максимальный размер 10 и измените значение *Увеличение размера файлов (File Growth)* на 10 процентов (рис.3).

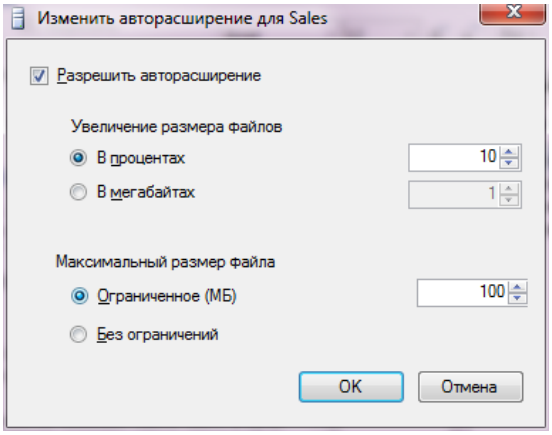


Рис. 3. Параметры автоматического роста файла настроены

8. ***Создайте папку для хранения файлов БД*** и измените путь к файлам БД, указав созданную папку. Для создания новой базы данных щелкните ОК. (В зале 306э можно оставить файл в исходной папке или выбрать папку STUDENT)
9. Вы должны увидеть свою новую базу данных в окне *Обозреватель объектов* приложения Management Studio. Выберите созданную базу данных Sales и для просмотра ее свойств щелкните правой кнопкой мыши на свободном поле рас- крывшейся вкладки, выберите *Отчеты -> Стандартный отчет -> Занято место на диске (Report->Standart Reports-> DiskUsage)* (рис. 4).
10. Посмотрите, как выглядит занятое место для системной БД, например, ***model.***

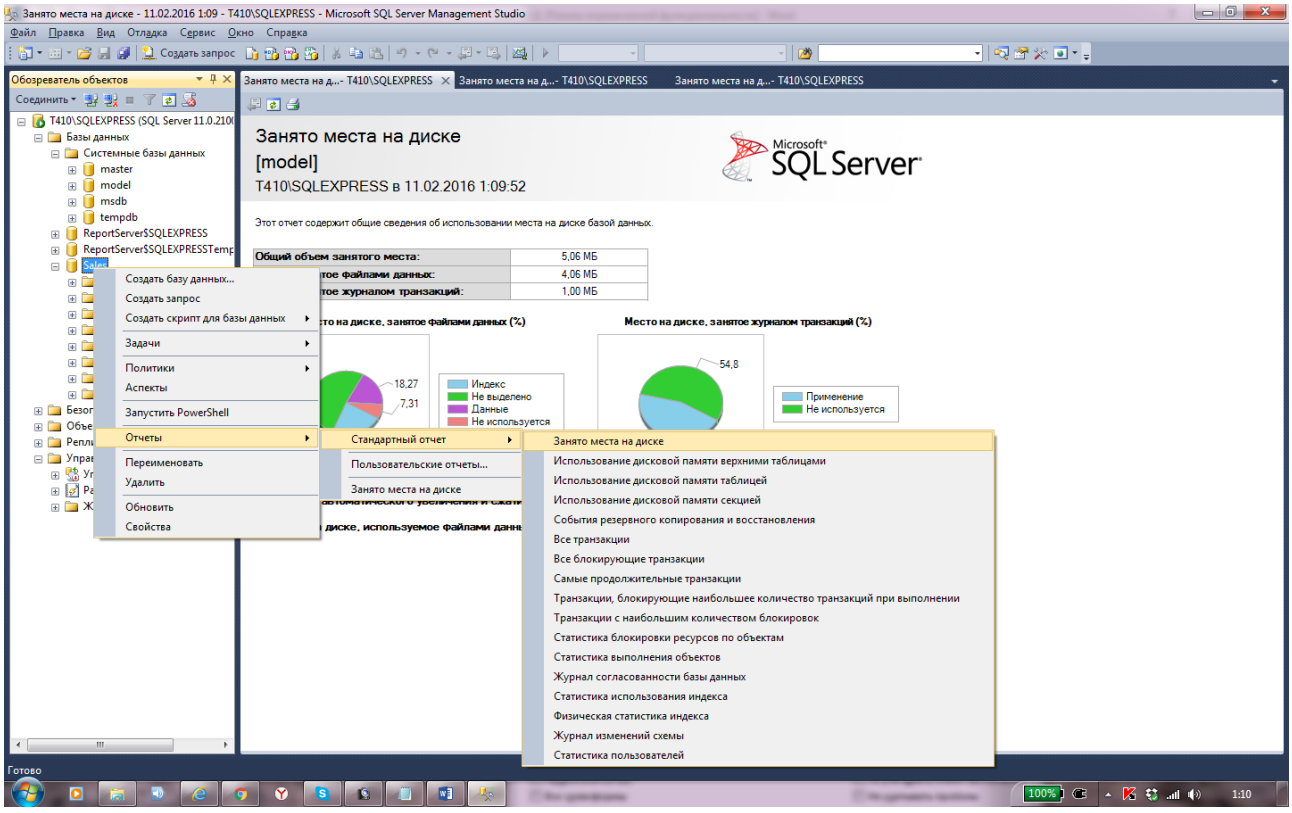


Рис. 4. Свойства базы данных продаж

При создании нового объекта в SQL Server вы можете и не увидеть его сразу в окне

Обозреватель объектов. Щелкните правой кнопкой на уровне, где должен быть новый объект, и выберите в контекстном меню пункт *Обновить (Refresh)*, чтобы программа SQL Server вновь проверила системные таблицы и отобразила все новые объекты базы данных.

Итак, основной строительный блок SQL Server — сама база данных — уже под вашим управлением. ***База данных представляет собой контейнер для других объектов***, таких как таблицы и представления, и без базы данных все эти объекты будут находиться в беспорядочном состоянии.

Мы узнали, что базы данных состоят из трех типов файлов: первичные файлы данных, вторичные файлы данных и файлы журналов транзакций. Первичные файлы данных используются для хранения пользовательских данных и системных объектов, необходимых SQL Server для доступа к базе данных. Вторичные файлы данных хранят лишь пользовательскую информацию и применяются для расширения базы данных на множество физических жестких дисков. Файлы журналов транзакций используются для обеспечения возможности восстановления путем отслеживания всех модификаций данных, сделанных в системе перед записью в файлы данных.

Вся информация в базе данных хранится в таблицах, которые представляют собой средство хранения данных. Таблицы состоят из строк или записей и столбцов или полей. Каждое поле имеет три характеристики:

- Имя поля – используется для обращения к полю;
- Значение поля – определяет информацию, хранимую в поле;
- Тип данных поля – определяет, какой вид информации можно хранить в поле.

В SQL сервер используется следующие типы данных:

- **Двоичные типы данных**, которые содержат последовательности нулей и единиц: 1) binary(n) – двоичный тип фиксированной длины размером в n байт, где n — значение от 1 до 8000; размер при хранении составляет n байт; 2) varbinary(n) – двоичный тип с переменной длиной, n может иметь значение от 1 до 8000, max указывает, что максимальный размер при хранении составляет 2^{31}-1 байт;
- **Целочисленные типы данных** – типы данных для хранения целых чисел (в скобках указан диапазон значений типа данных): tinyint (0..255), smallint (-32768..+32767), int (-2^{31}..+(2^{31}–1)), bigint (-2^{63}…+(2^{63}–1));
- **Типы данных для хранения чисел с плавающей запятой**: real занимает в памяти 4 байта; float(n), где n — это количество битов, используемых для хранения мантиссы числа в формате float при экспоненциальном представлении, определяет точность данных и размер для хранения; значение параметра n должно лежать в пределах от 1 до 53; значением по умолчанию для параметра n является 53;
- **Типы данных для хранения чисел с фиксированными точностью и масштабом**: decimal (p, s) и numeric (p, s), где p (точность) – максимальное количество десятичных разрядов числа (как слева, так и справа от десятичной запятой). Точность должна быть значением в диапазоне от 1 до 38. По умолчанию это значение равно 18. s (масштаб) – максимальное количество десятичных разрядов числа справа от десятичной запятой. Масштаб может принимать значение от 0 до p и может быть указан только совместно с точностью. По умолчанию масштаб принимает значение 0; поэтому $0 \le s \le p$;
- **Символьные типы данных**: char(n) – строковые данные фиксированной

длины не в Юникоде, аргумент n определяет длину строки и должен иметь значение от 1 до 8000, размер при хранении составляет n байт; varchar(n | max) – строковые данные переменной длины не в Юникоде, аргумент n определяет длину строки и должен иметь значение от 1 до 8000, значение max указывает, что максимальный размер при хранении составляет 2^{31}–1 байт (2 ГБ); text –данные переменной длины не в Юникоде в кодовой странице сервера и с максимальной длиной строки 231–1; **nchar** ([n]) - строковые данные постоянной длины в Юникоде. *n* определяет длину строки и должен иметь значение от 1 до 4000. Размер хранения — два раза *n* байт. Если кодовая страница параметров сортировки использует двухбайтовые символы, размер хранилища по-прежнему *n* байт. В зависимости от строки, размер хранилища *n* байтов может быть меньше, чем значение, указанное для *n*; **nvarchar** [(n | **max**)]Строковые данные переменной длины в Юникоде. *n* определяет длину строки и может принимать значение от 1 до 4000. **Max** указывает, что максимальный размер хранилища равен 2 ^ 31-1 байт (2 ГБ). Размер хранилища в байтах вдвое больше числа введенных символов + 2 байта.

- **Специальные типы данных**: bit – целочисленный тип данных, который может принимать значения 1, 0 или NULL; image – тип данных для хранения рисунка размером до 2ГБ;
- **Типы данных даты и времени**: date (от 01.01.0001 до 31.12.9999); datetime (диапазон даты – от 01.01.1753 до 31.12.1999, диапазон времени – от 00:00:00 до 23:59:590,997); smalldatetime (диапазон даты – от 01.01.1900 до 6.06.2079, диапазон времени – от 00:00:00 до 23:59:59); time (от 00:00:00.0000000 до 23:59:59.9999999);
- **Денежные типы данных для хранения финансовой информации**: money (8 байт) и smallmoney (4 байта) – типы данных, представляющие денежные (валютные) значения.

Подробнее типы данных можно изучить в документации:
https://docs.microsoft.com/ru-ru/sql/t-sql/data-types/data-types-transact-sql?view=sql-server-ver15

Создание таблиц

Создадим в базе данных Sales 4 таблицы. Первая таблица, **Customers**, будет хранить такую информацию о клиентах как название фирмы, фамилия, имя, адрес, город, индекс, телефон. Вторая таблица, **Orders**, будет содержать подробную информацию о заказах — номер заказа, идентификатор покупателя, даты отправки и оплаты, а так же статус заказа (‘C’ – Cancelled отменен, ‘P’ – Processed обработан), идентификатор продукта и заказанное количество. Третья таблица, **Products**, будет хранить такую информацию о продукте, как название, цена продукта, его количество на складе, необходимость перезаказа и описание. Таблица **Items** содержит идентификатор заказа, идентификатор продукта в заказе и его количество, общую сумму по данной позиции.

Для создания таблиц в SQL Server в первую очередь необходимо сделать активной ту БД, в которой создается таблица. Для этого в новом запросе можно набрать команду: `USE <Имя БД>`, либо на панели инструментов необходимо выбрать в выпадающем списке рабочую БД. После выбора БД можно создавать таблицы.

Таблицы создаются командой

```
CREATE TABLE table_name  ( { <column_definition> } ) [ ; ]
```

```
<column_definition> ::= column_name <data_type> [ NULL| NOT NULL ]
[DEFAULT constant_expression ] | [ IDENTITY [ ( seed,increment ) ] ]
[ <column_constraint> [ ...n ] ]
<column_constraint> ::= [ CONSTRAINT constraint_name ] { PRIMARY KEY | UNIQUE }
```

Здесь table_name – имя таблицы; column_name – имя столбца в таблице; data_type – тип данных для столбца; IDENTITY указывает, что новый столбец является столбцом идентификаторов, при этом *seed* – значение, используемое для самой первой строки, загружаемой в таблицу, *increment* – значение приращения, добавляемое к значению идентификатора предыдущей загруженной строки; CONSTRAINT – необязательное ключевое слово, указывающее на начало определения ограничения, *constraint_name* – имя ограничения; NULL | NOT NULL определяет, допустимы ли для столбца значения NULL; PRIMARY KEY – ограничение, которое определяет столбец первичным ключом таблицы. Если имя поля содержит пробел, то оно заключается в квадратные скобки.

Пример. Создать таблицу **Orders**, содержащую поля:

- IdOrder, IdCustomer, OrderDate, ShipDate, PaidDate, Status.

```
USE Sales;
IF OBJECT_ID('dbo.Orders', 'U') IS NOT NULL DROP TABLE dbo.Orders;
CREATE TABLE dbo.Orders
(
IdOrder BIGINT IDENTITY(1,1) CONSTRAINT pk_order PRIMARY KEY,
IdCustomer INT NOT NULL,
OrderDate  DATE NOT NULL,
ShipDate DATE NULL, PaidDate DATE NULL,
[Status] char(1)
);
```

Инструкция **USE** изменяет связь с текущей БД на связь с **Sales**. Включение этой инструкции в сценарии создания объектов очень важно, т.к. гарантирует создание объектов в требуемой БД.

Инструкция **IF** запускает функцию **OBJECT_ID**, которая в качестве входных параметров принимает имя объекта и его тип. Тип **'U'** представляет пользовательские таблицы. Данная функция возвращает внутренний ID объекта, если объект с заданным именем и типом уже существует, и значение **NULL** в противном случае.

При создании таблицы используется схема с именем **dbo**, которая создается автоматически в каждой базе данных и используется как схема по умолчанию. Если опустить имя схемы при создании таблицы, то SQL Server свяжет с таблицей схему, используемую по умолчанию для имени пользователя, выполняющего программный код.

Для каждого атрибута сущности **Orders** задается его имя, тип и допустимость значений **NULL**.

Для столбца **IsOrder** определено ограничение в виде первичного ключа (**pk_order**), при этом значения **IdOrder** будут начинаться с 1 и увеличиваться при каждом добавлении новых строк в таблицу тоже на 1 (**IDENTITY(1,1)**).

Для выполнения запросов, введенных в среде SQL Server Management Studio, нужно нажать на кнопку «Выполнить». Можно выполнять только часть инструкций, для этого нужные достаточно выделить курсором.

Используя этот пример, создайте таблицу **Customers**.

Таблицы можно создавать и в графическом интерфейсе. Рассмотрим на примере таблицы Products.

1. В окне *Обозреватель объектов* раскройте папки *Базы данных*=>Sales.
2. Щелкните правой кнопкой на папке *Таблицы (Tables)* и примените команду *Создать таблицу (New Table),* чтобы открыть конструктор таблиц.
3. В первую строку в столбце *Имя столбца (Column Name)* введите имя IdProduct.
4. В столбец *Тип данных (Data Type)* введите тип int.
5. Убедитесь, что параметр *Разрешить значения Null (Allow Nulls)* не включен. С включенной этой опцией поле может быть вообще без данных, а этого нам не нужно.
6. Установим аналог счетчика в Access. Свойство Identity, для этого в нижней половине экрана в разделе *Конструктор таблиц (Table Designer)* блока *Свойства столбца (Column Properties)* разверните *Спецификация идентификатора (Identity Specification)* и измените значение параметра *Идентификатор (Is Identity)* на **Да**.
7. Введите в столбец *Имя столбца (Column Name)* второй строки под IdProduct имя PrName.
8. В столбец *Тип данных (Data Type)* введите тип nvarchar.
9. В разделе *Общие* блока Column Properties определите для настройки *Длина (Length)* значение 100.
10. Убедитесь, что параметр *Разрешить значения Null (Allow Nulls)* отключен.
11. В столбец *Имя столбца (Column Name)* введите для третьей строки имя PrPrice и тип данных smallmoney.
12. Четвертая строка *Имя столбца (Column Name)* – InStock тип int.
13. Пятая строка – ReOrder тип bit и шестая - Description тип nvarchar(100)
14. Щелкните на кнопке *Save* в левой части панели инструментов (в виде дискеты).
15. В открывшийся блок *Выбор имени (Choose Name)* введите имя Products (рис. 5)

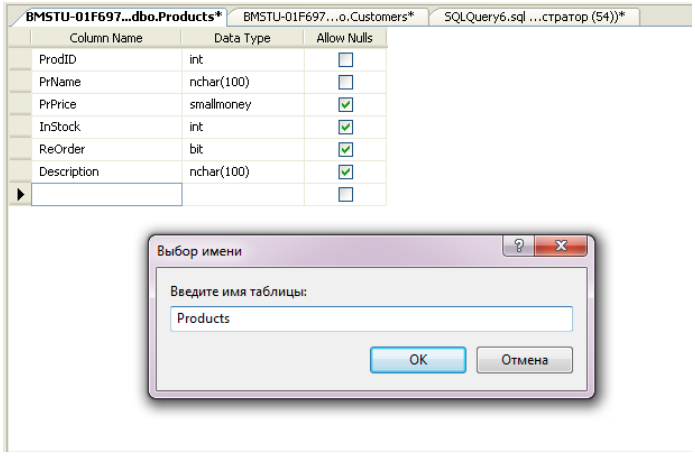


Рис. 5. Окно создания новой таблицы Products

16. Щелкните на кнопке OK.

Создайте оставшиеся две таблицы, одну с помощью T-Sql, вторую - графически.

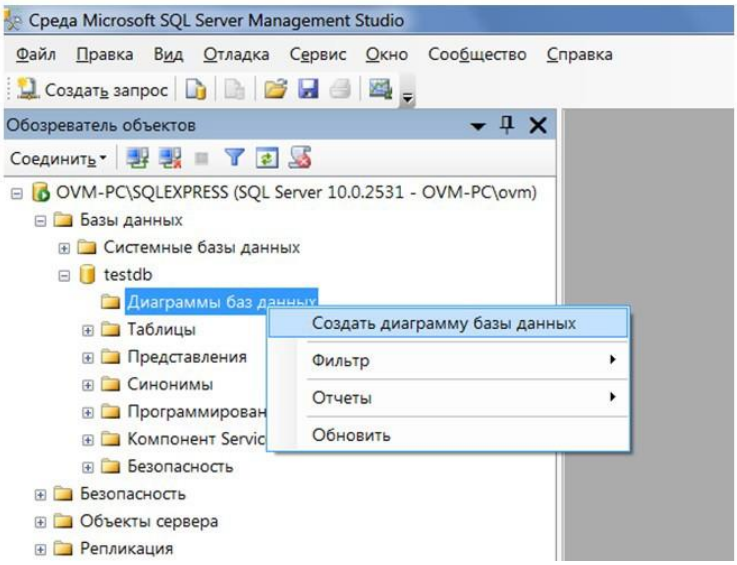
Для того чтобы обеспечить ссылочную целостность в БД, нужно добавить в созданные таблицы ограничение по внешним ключам.

Пример. Связать таблицы Customers и Orders по столбцу IdCustomer при помощи SQL

```
ALTER TABLE dbo.Orders
ADD CONSTRAINT FR_Ord_Cust      -- FK_Ord_Cust - имя внешнего ключа для СУБД
  FOREIGN KEY (IdCustomer) REFERENCES Customers (IdCustomer)
```

Когда установлены связи между всеми таблицами, можно создать описывающую их диаграмму. Для создания диаграммы нужно щелкнуть правой кнопкой мыши на

элементе БД «Диаграммы баз данных» и в открывшемся контекстном меню выбрать пункт «Создать диаграмму базы данных».

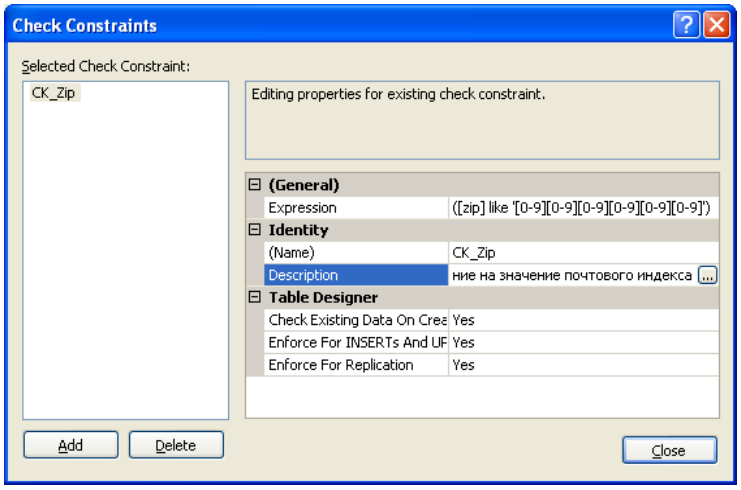


В открывшемся диалоговом окне нужно выбрать таблицы, которые будут добавлены на диаграмму.

Использование проверочных ограничений (задание лабораторной работы)

Ограничения на проверку используются для ограничения данных, принимаемых полем, даже если они имеют корректный тип. Например, поле Zip (почтовый индекс) имеет тип nchar(6), т.е. чисто теоретически оно может принимать буквы. Это может стать проблемой, поскольку не существует почтовых индексов с буквами. Рассмотрим, как создать ограничение на проверку, запрещающее вводить в это поле буквы.

1. В контекстном меню папки «Ограничения» таблицы Customers выберите команду «Создать ограничение».
2. В открывшемся окне «Проверочные ограничения» заполните следующие поля:
   - Имя: CK_Zip
   - Выражение: ([zip] like '[0-9][0-9][0-9][0-9][0-9][0-9]'). Данное выражение описывает ограничение, принимающее шесть символов, которыми могут быть только цифры от 0 до 9.
   - Описание: Ограничение на значения почтового индекса
3. Щелкните на кнопке «Закрыть» и закройте конструктор таблиц (он был открыт, когда вы начали создавать ограничение) с сохранением изменений.



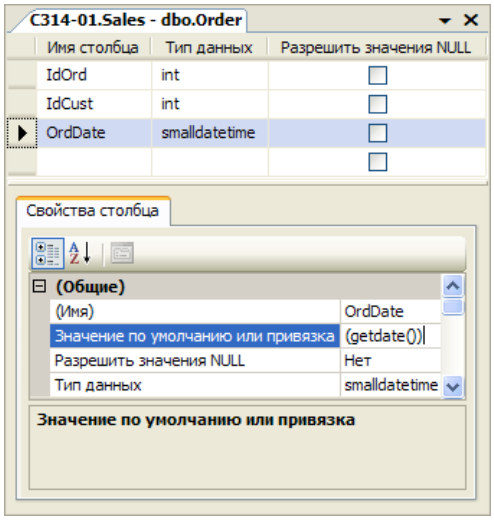
Создайте ограничения для полей PrPrice, InStock таблицы Products и Quantity таблицы Items, запрещающие ввод в них отрицательных значений.

Протестируйте созданное вами ограничение, введя в таблицу несколько новых записей, с использованием инструкции INSERT.

Использование значений по умолчанию (задание лабораторной работы)

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 заказов Orders вполне логично определить по умолчанию значение поля OrdDate (дата заказа) в виде текущей даты. В этом случае при добавлении записи о новом заказе в случае пропуска этого поля оно будет автоматически заполняться значением системной даты. Для создания такого свойства выполните следующие шаги:

1. Раскройте папку «Столбцы» таблицы Orders и в контекстном меню поля «OrderDate» выберите команду «Изменить».
2. В свойстве столбца «Значение или привязка по умолчанию» введите getdate(). Эта функция T-SQL возвращает текущую системную дату.
3. Щелкните на кнопке Сохранить и выйдите из конструктора таблиц.



Сохранение БД

Чтобы сохранить БД для дальнейшей работы, сначала нужно закрыть все активные соединения БД, затем можно отсоединить БД. В контекстном меню базы Sales выбрать *Задачи -> Отсоединить (Tasks -> Detach).*

Теперь можно скопировать файл базы и журнала в любое удобное место. Перенос файлов журналов обязателен, даже если нужно создать новые файлы журналов.

Для присоединения базы используется команда *Присоединить (Attach)* контекстного меню Базы данных.

Использование операторов INSERT, UPDATE, DELETE

В SQL Server заполнение таблиц производится при помощи следующей команды:

```
INSERT INTO <имя_таблицы>(<список_столбцов>) VALUES (<список_значений>);
```

где <имя_таблицы> – таблица, куда вставляются данные, <список_столбцов> – список полей, в которые вставляются данные, если он не указывается, то подразумевается заполнение всех полей, в списке полей поля указываются через запятую, <список_значений> – значение полей для вставки через запятую.

Пример: Добавление записи в таблицу «Orders»:

```
use Sales19
INSERT INTO dbo.Orders (IdCustomer, OrderDate, ShipDate, PaidDate, [Status])
VALUES ( 3,'12.01.2019', '12.01.2019','12.01.2019','P');
```

Из таблицы можно удалить все столбцы, либо отдельные записи. Это осуществляется командой

```
DELETE FROM <Имя таблицы> [WHERE <Условие>]
```

где <Условие> – условие, которым удовлетворяют удаляемые записи, если условие не указано, то удаляются все столбцы таблицы.

Пример: Удалить записи из таблицы Orders, у которых поле IdCustomer < 10.

```
DELETE FROM dbo.Orders WHERE IdCustomer < 10;
```

Значение полей таблицы можно обновить (изменить), используя следующую команду:

```
UPDATE <Имя таблицы> SET
<Имя поля1> = <Выражение1>, [<Имя поля2> = <Выражение2>,]
…
[WHERE <Условие>]
```

Здесь <Имя поля1>, <Имя поля2> – имена изменяемых полей; <Выражение1>, <Выражение 2> – значения, которые должны принять поля; <Условие> – условие, которым должны соответствовать записи, поля которых изменяем. В качестве выражения можно использовать математические формулы.

Пример: для таблицы Orders, установить сегодняшнюю дату отгрузки для всех обработанных заказов

```
UPDATE Orders SET ShipDate = GETDATE()
WHERE [Status]='P'
```

Если необходимо из таблицы удалить все записи, но сохранить ее структуру, нужно воспользоваться командой

```
TRUNCATE TABLE <Имя таблицы>
```

при этом все данные будут удалены, но сама таблица останется.

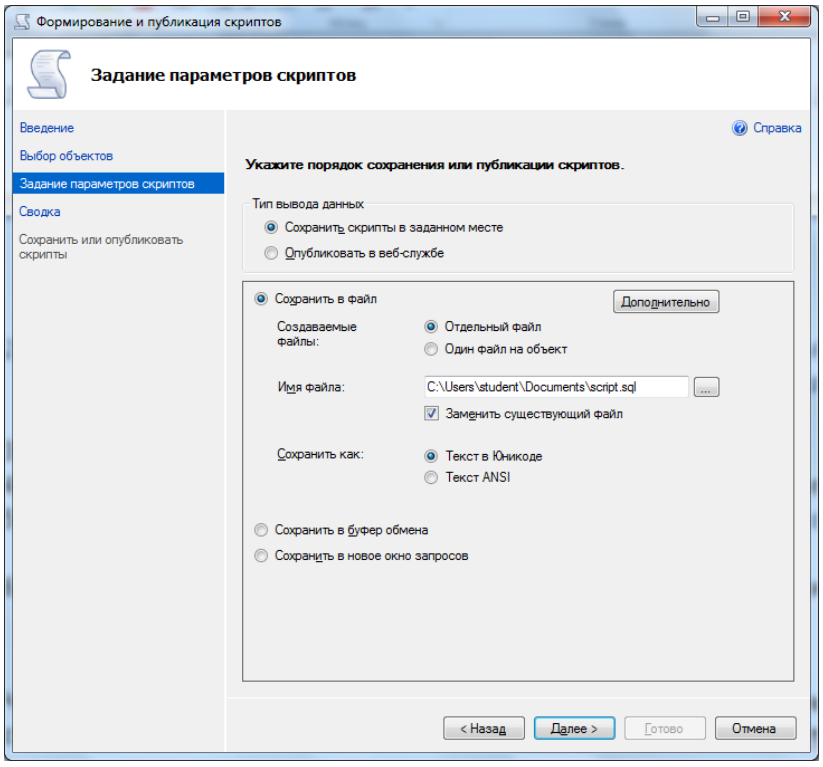
Создание скрипта БД

Иногда бывает необходимо получить структуру готовой базы в виде скрипта. Создадим скрипт, позволяющий заново создавать БД и заполнить ее данными.

Для этого выполните следующие действия:

1. Щелкните правой кнопкой мыши вашу базу данных и выберите *Задачи - > Сформировать скрипты (Tasks -> Generate Scripts)*

2. В мастере создания и публикации скриптов выберите опцию *Выбор объектов –> Внести в скрипт всю базу данных и все объекты базы данных» (Script all objects in the selected database)*
3. На следующем экране выберите, куда вы хотите сохранить скрипт. Затем нажмите кнопу *«Дополнительно».* В дополнительных параметрах *«Типы данных для внесения в скрипт -> Схема и данные*



4. Последнее окно предлагает проверить все указанные опции, нажав *Завершить (Finish),* получаем скрипт.

Чтобы восстановить базу данных, просто создайте новую базу данных и измените первую строку сгенерированного сценария на USE [Your_New_Database_Name], а затем выполните.

Контрольные вопросы

1. Что такое СУБД?
2. Отличие реляционной СУБД от не реляционной?
3. Какие существуют уровни абстракции архитектуры базы данных и что они из себя представляют?
4. Назовите компоненты основных типов объектов БД на логическом уровне. Какие функции они выполняют?
5. В чем заключается отличие таблиц от представлений?
6. Опишите основные элементы физической структуры.
7. Объясните суть принципа постраничной организации файлов и для чего он необходим.
8. Назовите ваши действия при подключении к серверу.
9. Опишите структуру среды MS SQL Server Management Studio?
10. Какие типы данных используются в MS SQL?
11. Опишите структуру запроса для создания таблицы.
12. Что такое первичный ключ? Почему первичный ключ должен быть уникальным?
13. Как реализуется связь между таблицами? Объяснить на примере запроса создания связи между таблицами с помощью внешних ключей.
14. Что отображается на диаграмме базы данных?
15. Для чего используются проверочные ограничения? Как создать ограничение на проверку?
16. Зачем используются значения по умолчанию? Как установить значение по умолчанию?
17. Опишите структуру команды INSERT для заполнения таблицы.
18. Опишите структуру команды DELETE для удаления записей из таблицы.
19. Опишите структуру команды UPDATE для записей из таблицы.
20. Каково назначение команды TRUNCATE TABLE?