Кох Е. В.

Базы данных

Методические указания по выполнению работ

Екатеринбург

2021 г.

Оглавление

Введение

Информационные системы, использующие базы данных, в настоящее время представляют собой одну из важнейших областей современных компьютерных технологий. С этой сферой связана большая часть современного рынка программных продуктов. Одной из общих тенденций в развитии таких систем являются процессы интеграции и стандартизации, затрагивающие структуры данных и способы их обработки и интерпретации, системное и прикладное программное обеспечение, средства разработки взаимодействия компонентов баз, данных и т.п. Современные системы управления базами данных(СУБД) основаны на реляционной модели представления данных—в большой степени благодаря простоте и четкости ее концептуальных понятий и строгому математическому обоснованию

Реляционная база данных, определения и понятия

Реляционная база данных представляет собой множество **взаимосвязанных двумерных таблиц** – реляционных таблиц, называемых также **отношениями**, в каждой из которых содержатся сведения об одной сущности автоматизируемой предметной области.

Логическую структуру реляционной базы данных образует совокупность реляционных таблиц, между которыми установлены связи.

В таблицах базы должны сохраняться все данные, необходимые для решения задач предметной области. Причем каждый элемент данных должен храниться только в одном экземпляре.

Для создания таблиц, соответствующих реляционной модели данных, используется процесс, называемый нормализацией данных. **Нормализация** – это удаление из таблиц повторяющихся данных путем их переноса в новые таблицы, записи которых не содержат повторяющихся значений.

Структура реляционной таблицы определяется составом полей. Каждое поле отражает определенную характеристику сущности. Для поля указывается

тип и размер элементарного данного, размещаемого в нем, и ряд других свойств. Содержимое поля отображается в столбце таблицы. Столбец таблицы содержит данные одного типа.

Содержание таблицы заключено в ее строках, однотипных по структуре. Каждая строка таблицы содержит данные о конкретном экземпляре сущности и называется **записью**. Структура записи определяется составом входящих в нее полей. Для однозначного определения (идентификации) каждой записи таблица должна иметь уникальный (**первичный**) ключ. По значению ключа таблицы отыскивается единственная запись в таблице. Ключ может состоять из одного или нескольких полей таблицы. Значение уникального ключа не может повторяться в нескольких записях.

Логические связи между таблицами дают возможность объединять данные из разных таблиц. Связь каждой пары таблиц обеспечивается одинаковыми полями в них – ключом связи. Так обеспечивается рациональное хранение недублированных данных и их объединение в соответствии с требованиями решаемых задач.

Для двух таблиц, находящихся в отношении типа 1:M, устанавливается связь по уникальному ключу таблицы, представляющей в отношении сторону «один» - главной таблицы в связи. Во второй таблице, представляющей в отношении сторону «многие» и именуемой подчиненной, этот ключ связи может быть либо частью уникального ключа, либо не входить в состав ключа. В подчиненной таблице ключ связи называется еще **внешним ключом**.

РЕАЛИЗАЦИЯ БАЗЫ ДАННЫХ С ПОМОЩЬЮ СУБД MICROSOFT SQL SERVER

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

Microsoft SQL Server — система управления реляционными базами данных (СУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов —SQL, создан совместно Microsoft и Sybase. SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

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

В Microsoft SQL Server основным средством администратора БД служит утилита Microsoft SQL Server Management Studio.

SQL Management Studio for SQL Server – это законченное решение для администрирования и разработки баз данных. Для разработчиков приложений или баз данных, администраторов или аналитиков, в SQL Studio найдутся все необходимые средства, чтобы сделать вашу работу продуктивной как никогда ранее. SQL Studio соединяет эти средства в единую мощную и удобную рабочую среду.

SQL Studio предоставляет незаменимые средства для администрирования баз данных и управления их объектами, а также для миграции, сравнения и извлечения баз данных, импорта, экспорта и сравнения данных.

Для выполнения данных лабораторных работ вам понадобится установленная СУБД Microsoft SQL Server версии не ниже 2014.

Взаимодействие с СУБД Microsoft SQL Server осуществляется через графическую оболочку Microsoft SQL Server Management Studio.

УСТАНОВКА СОЕДИНЕНИЯ С СЕРВЕРОМ MICROSOFT SQL SERVER И ПРИНЦИПЫ СОЗДАНИЯ БАЗ ДАННЫХ

ЗАДАНИЕ

1. Создать соединение с локальным или удаленным сервером.
2. Изучить пользовательский интерфейс SQL Server Management Studio.
3. Познакомиться с основными принципами создания и удаления базы данных в MS SQL Server.
4. Создать БД с помощью мастера и с помощью запроса (в отчете отобразить создание с помощью обоих методов).
5. Создать резервную копию для восстановления БД.

Ход работы

1. После запуска среды разработки SQL Server Management Studio появится окно подключения к серверу (рис. 1).

В параметрах указываем:

Тип сервера – Компонент Database Engine.

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

проверка подлинности – Windows (по умолчанию),

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

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

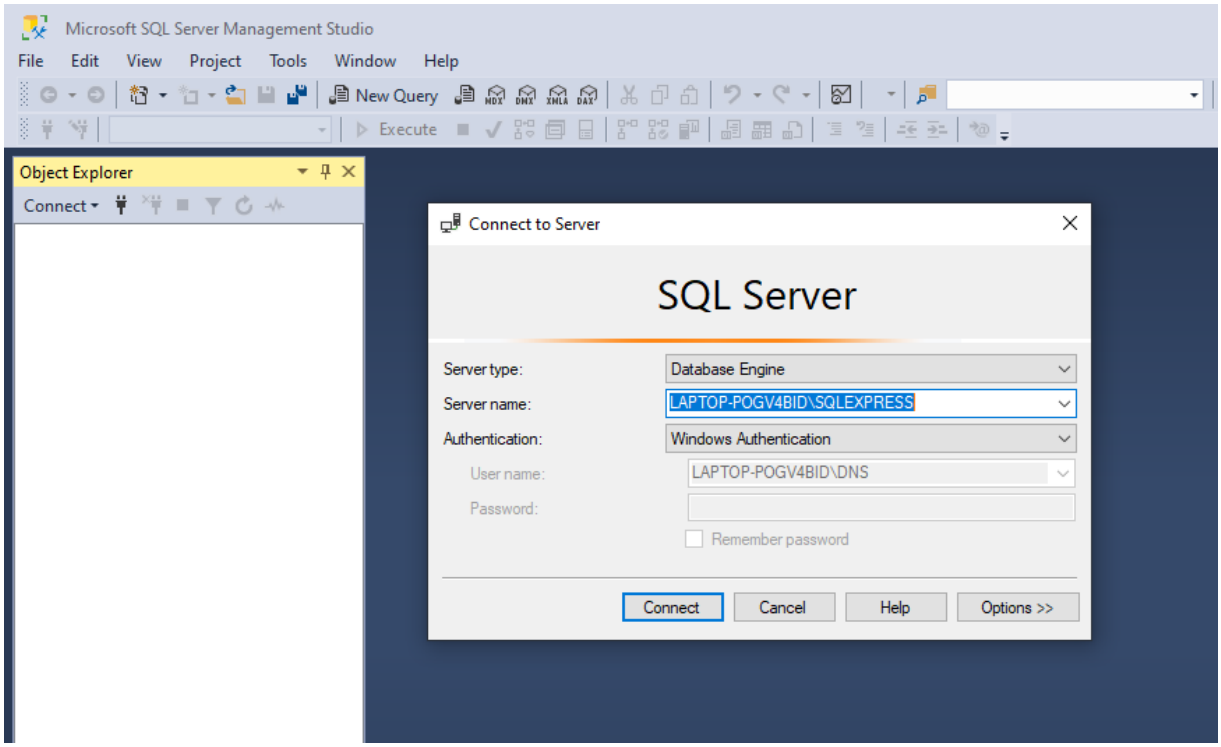


Рис. 1. Соединение с сервером

2. Нажмите `Connect` (Соединить), появится окно среды разработки `SQL Server Management Studio` (рис. 2). Если соединение будет совершенно успешно, то на экране появятся данные сервера.

3. Сведения о базе данных отображаются в обозревателе объектов и окнах документов. Обозреватель объектов является представлением в виде дерева, в котором отображаются все объекты базы данных на сервере.

Общие сведения о базах данных MS SQL Server

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

База данных представляет собой:

— набор взаимосвязанных таблиц;

— связанный набор страниц, выделенных для хранения данных MS SQL Server;

— совокупность данных при архивации;

— два и более файла;

— важную совокупность данных для целей защиты и управления.

Файлы базы данных

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

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

Файлы делятся на три типа:

— **первичные файлы**. Используются для хранения данных и информации, определяющих начальные действия с базой. База данных содержит лишь один первичный файл. Стандартное расширение — **.mdf.**

— **вторичные файлы**. Одна или несколько вспомогательных областей для хранения данных. Могут использоваться для распределения операций чтения/записи по нескольким дискам. Стандартное расширение — **.ndf.**

— **файлы журналов.** Содержат журналы транзакций базы данных. База данных содержит по крайней мере один файл журнала. Стандартное расширение — **.ldf.** Перед непосредственной записью транзакций в файл данных все вносимые изменения записываются в журнал.

Группы файлов

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

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

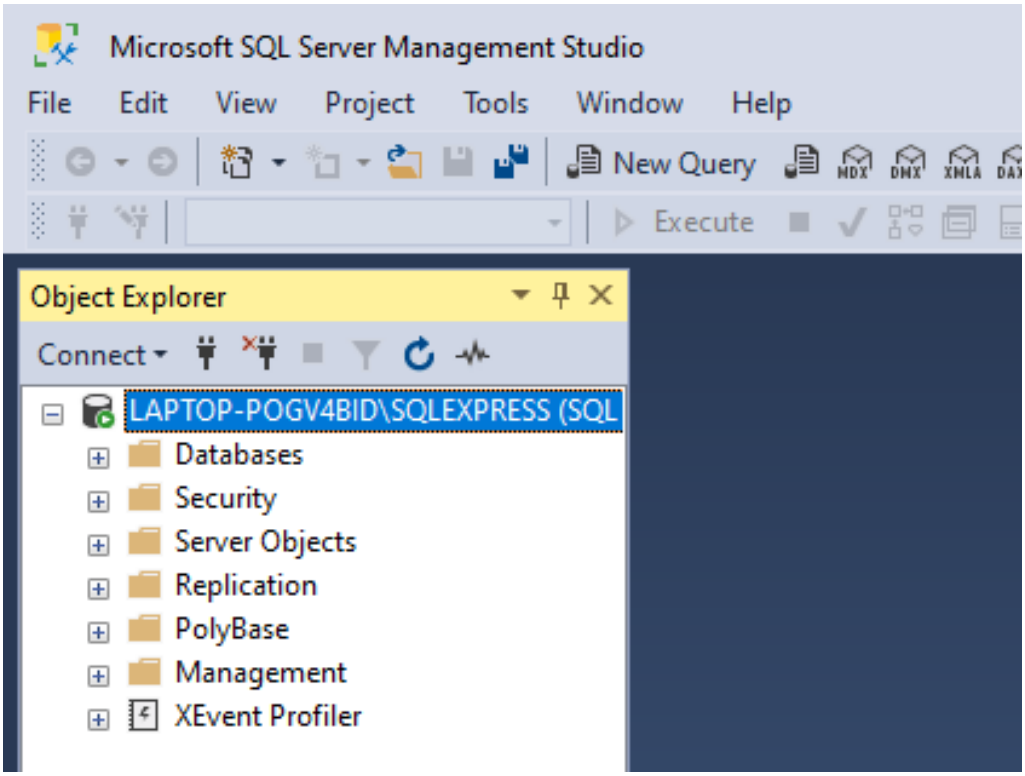


Рис. 2. Рабочее окно Server Management Studio

Создание и удаление базы данных

Для создания базы данных можно использовать один из двух способов:

Первый способ создания БД.

1. Выполнить команду Базы данных/Создать базу данных...(в контекстном меню папки Базы данных) в программе SQL Server Management Studio (рис.3), ввести параметры создаваемой базы данных в диалоговом окне Создание базы данных (рис. 4) и нажать кнопку OK.

2. В поле Имя базы данных введите имя будущей базы данных, например – `University` (!!! При создании новой базы данных используйте следующие правила присвоения имени: Название_БД_группа_Фамилия). Используйте только английский язык. Поле Владелец - задан по умолчанию, в зависимости от настройки сервера. Папка с базой данных будет создана по умолчанию на диске.

C:\Program Files\Microsoft SQL Server\MSSQL10.SQLEXPRESS2\MSSQL\DATA \.

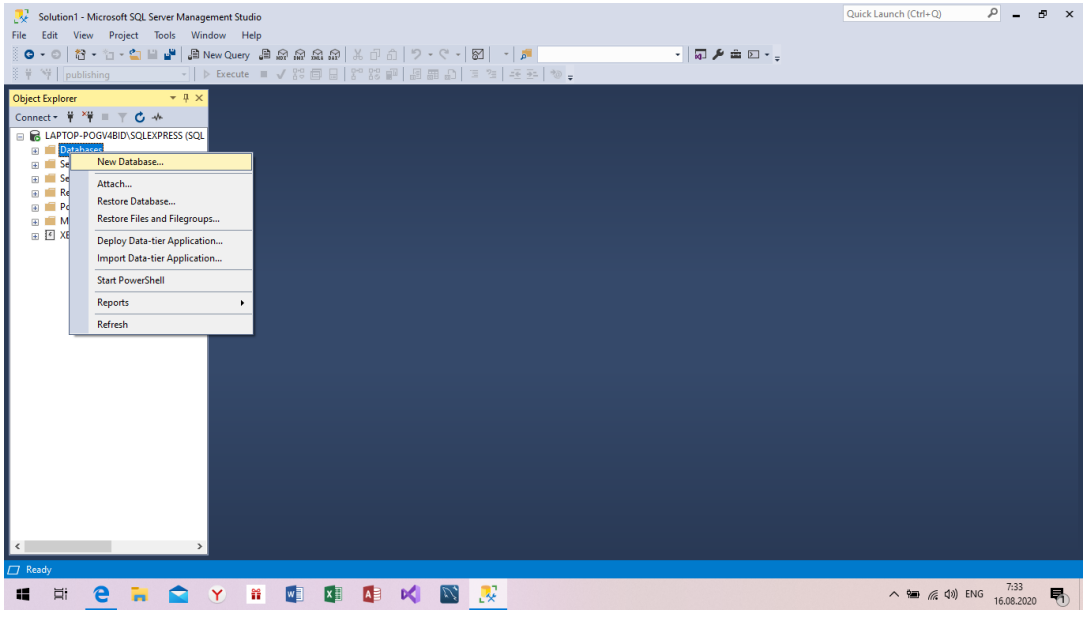


Рис.3. Выполнение команды Создать новую БД

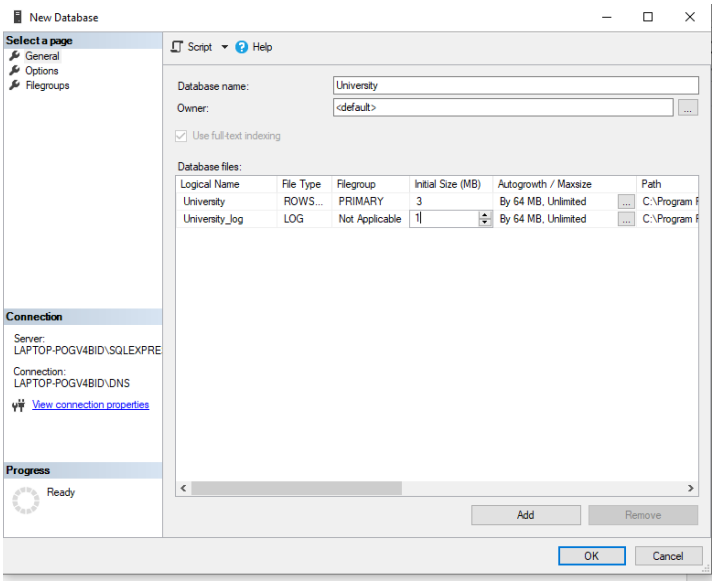
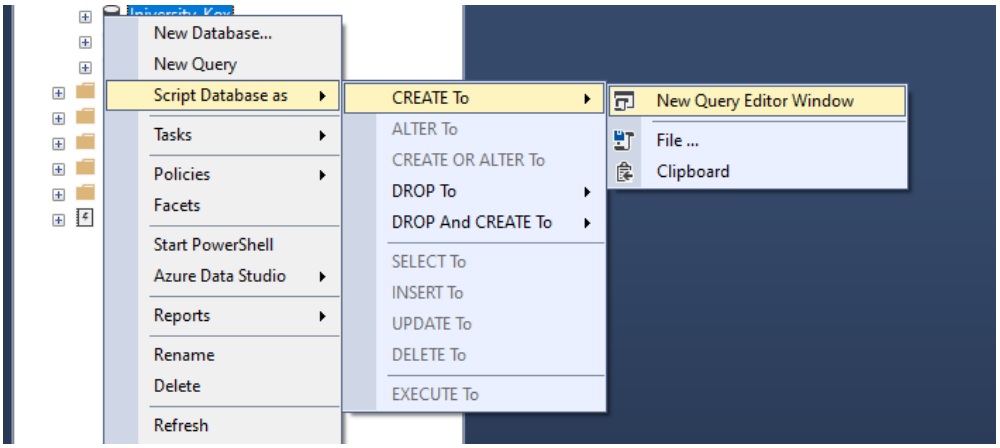


Рис. 4. Диалоговое окно создания базы данных

3. После нажатия на кнопку OK программа SQL Server Management Studio создаст базу данных, имя которой вы увидите в обозревателе объектов.

SQL Server Management Studio создает базу данных, а также генерирует необходимый SQL-код для создания базы данных с теми свойствами, которые указаны в этом диалоговом окне и передаст его серверу СУБД для выполнения.

4. Нажмите на имени базы данных University правой клавишей и из контекстного меню выберите Создать скрипт как.. CREATE To, рис.5.



С

Рис.5. Скрипт создания БД

5. Полученный скрипт (рис. 6.) на создание БД University **сохраните** в рабочей папке.

6. Удалите созданную БД, выбрав в конткстном меню команду Удалить.

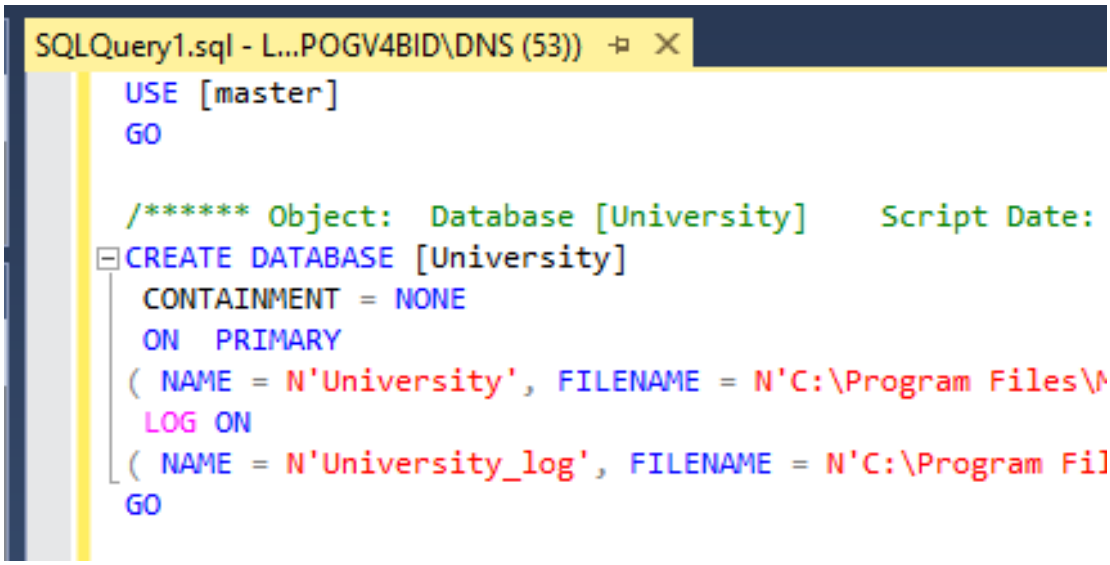


Рис.6. Сгенерированный sql-код созданной базы данных

Второй способ создания БД

Создание новой БД с помощью запроса Новую БД можно создать, используя стандартные команды языка SQL. Все команды языка SQL набираются на вкладке нового запроса (SQLQuery). Для того чтобы создать новый запрос на панели инструментов необходимо нажать кнопку NewQuery/Создать запрос. Для выполнения команд языка SQL на панели инструментов необходимо нажать кнопку Execute/Выполнить или на вкладке нового запроса набрать команду GO.

Создание баз данных разрешается любому пользователю с ролью системного администратора или всем, кому системный администратор предоставил такое право.

Пример: Создать БД «publishing», расположенную в файле D:\publishing.mdf и имеющую начальный размер файла данных 5 Мб., максимальный размер файла данных неограничен. и шаг увеличения файла данных равный 1 Мб. Файл журнала транзакций данной БД имеет имя publishing_log и расположен в файле D:\publishing_log.ldf. Данный файл имеет

начальный размер равный 1 Мб., максимальный размер равный 2 Гб. и шаг увеличения равный 100 Кб.

```
CREATE DATABASE [publishing] ON  PRIMARY
 (
    NAME = 'publishing',
    FILENAME = 'D:\publishing.mdf' ,
    SIZE = 5120KB ,
     MAXSIZE = UNLIMITED,
      FILEGROWTH = 1024KB )
    LOG ON
    (
    NAME = 'publishing_log',
      FILENAME = 'D:\publishing_log.ldf' ,
     SIZE = 1024KB ,
      MAXSIZE = 2048GB ,
      FILEGROWTH = 10%
    )
    GO
```

Если при создании базы не указан первичный файл данных и/или файл журнала, то отсутствующий файл (или файлы) создается с именем по умолчанию. Физические файлы будут находиться в стандартном каталоге. Первичному файлу присваивается имя имя_базы.mdf, а файлу журнала — имя_ базы_log.ldf. Если размер файлов не задан, то при создании размер первичного файла совпадает с размером первичного устройства базы model, а размер файла журнала и вторичных файлов данных равен 1 Мбайт. Он может быть и больше, если размер первичного файла базы данных model превышает 1 Мбайт. Хотя имена и размеры файлов указывать не обязательно, на практике это всегда следует делать. SQL Server создает базу данных за два этапа. На первом этапе база model копируется в новую базу данных, а на втором этапе инициализируется все неиспользуемое пространство.

Команда CREATE DATABASE имеет следующие параметры: • PRIMARY — файл определяется как первичное устройство.

NAME — логическое имя; по умолчанию совпадает с именем файла. • FILENAME — полное имя файла на диске.

SIZE — исходный размер файла. Минимальный размер файла журнала равен 512 Кбайт.

MAXSIZE — максимальный размер файла.

UNLIMITED — размер файла не ограничивается.

FILEGROWTH — приращение размера в мегабайтах (MB), килобайтах (KB) или процентах (%). По умолчанию приращение равно 10%.

FOR LOAD — обеспечивает обратную совместимость со сценариями SQL, написанными для предыдущих версий SQL Server.

FOR ATTACH — указывает, что файлы базы данных уже существуют.

Создание базы данных с помощью запроса

1. Выполнить в программе SQL Server Management Studio команду Создать запрос/ New Query. Наберите предложенный код на рис. 8. (имена и путь создания базы указывать свои).

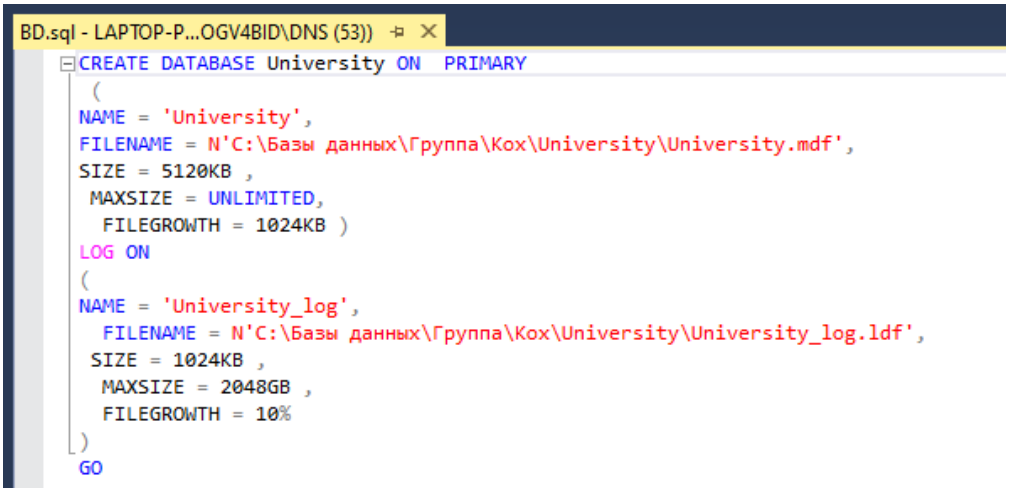


Рис.8.

2. Запустите на выполнение скрипт (Выполнить/Execute).

3. Сохраните созданный скрипт в рабочую папку.

4. После успешного выполнения и обновления проводника у вас должна появится база данных `University`.

5. Для переименования БД необходимо в Обозревателе объектов щёлкнуть по ней правой кнопкой мыши и в появившемся меню выбрать пункт Rename /Переименовать. Для обновления — пункт Refresh/Обновить, а для изменения свойств, описанных выше, — пункт Properties/Свойства.

Для удаления базы данных можно использовать один из трех способов:

Выполнить в программе SQL Server Management Studio команду контекстного меню Удалить, выбрав перед этим в списке базу данных, а затем подтвердить свое желание в диалоговом окне.

Выполнить оператор DROP DATABASE в SQL-редакторе.

Удалить файл с базой данных.

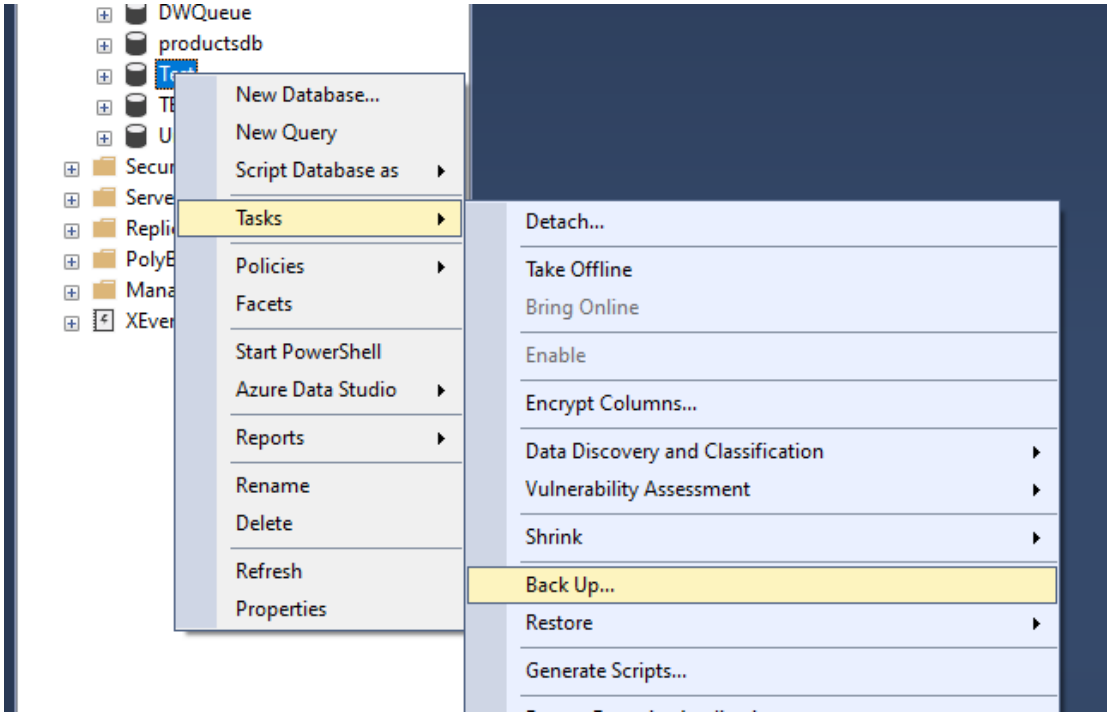
Синтаксис оператора `DROP DATABASE`:

```
DROP DATABASE database_name;
```

Резервное копирование и восстановление

Резервное копирование (backup) базы данных и восстановление из резервной копии (restore) – два важнейших и наиболее частых процесса, осуществляемых администраторами баз данных. Резервное копирование базы данных – единственный надежный способ предохранить данные от потери в результате поломки диска, сбоев электропитания, действий злоумышленников и ошибок в программах. В процессе резервного копирования создается независимый от платформы "снимок" базы данных, с помощью которого можно перенести данные на другую операционную систему или даже другую платформу. Полный цикл: резервное копирование и восстановление из резервной копии приводит к корректировке статистической информации, является средством от излишнего "разбухания" базы данных и необходимой операцией обслуживания базы данных. Кроме того, миграция от одной версии сервера к другой также происходит при помощи процесса backup/restore.

Для создания резервной копии базы данных с помощью программы " SQL Server Management Studio " необходимо подключиться к базе данных, выбрать из контекстного меню базы данных Задачи/ (Tasks) Создать резервную копию (Back Up Database), рис. 8.1. В открывшемся диалоговом окне "Мастер резервного копирования" задать несколько параметров и нажать кнопку [Выполнить], После выбора пути и файла для резервной копии в окне Back Up Database нажатием на OK запускаем процесс создания резервной копии. В случае успешной работы появится сообщение. В результате будет создан файл с резервной копией. Стандартным расширением таких файлов для " SQL Server Management Studio " является "*.bak". Файл с резервной копией базы данных обычно на порядок меньше оригинала.



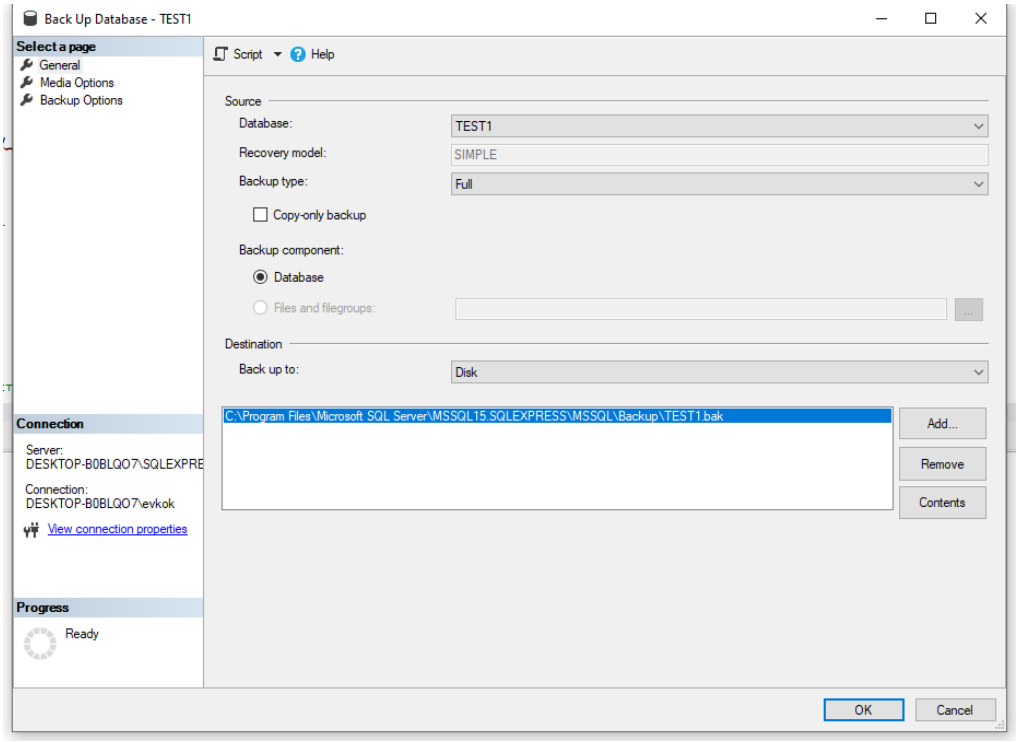
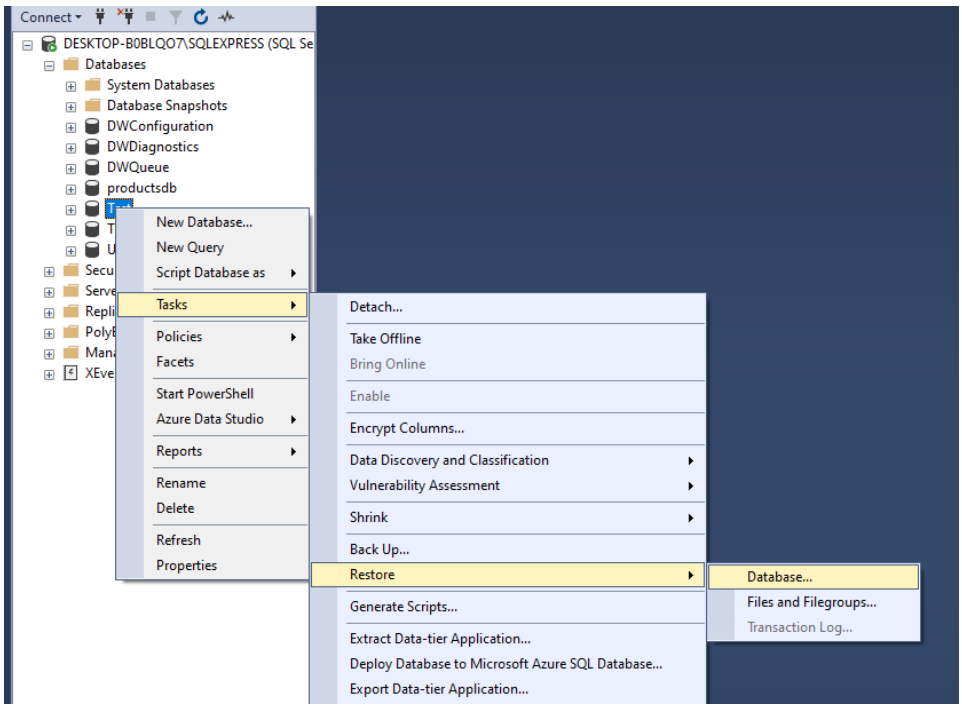
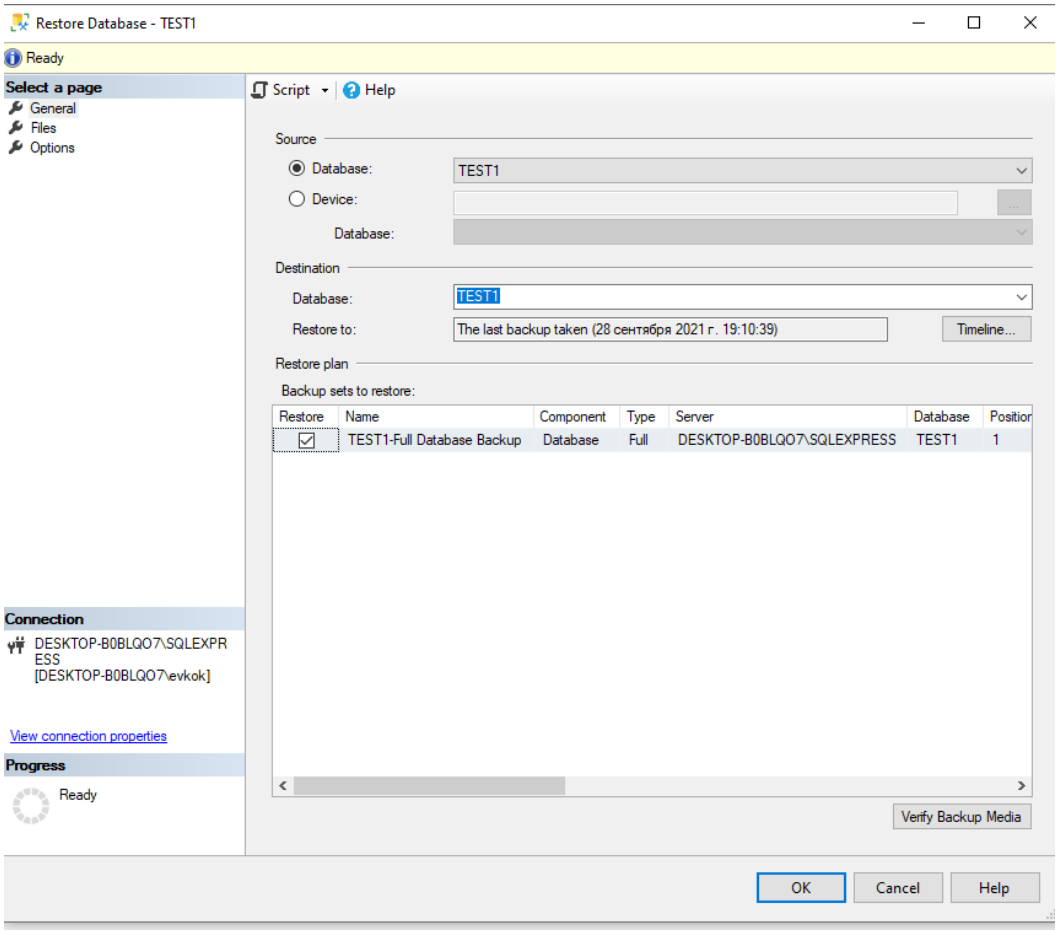
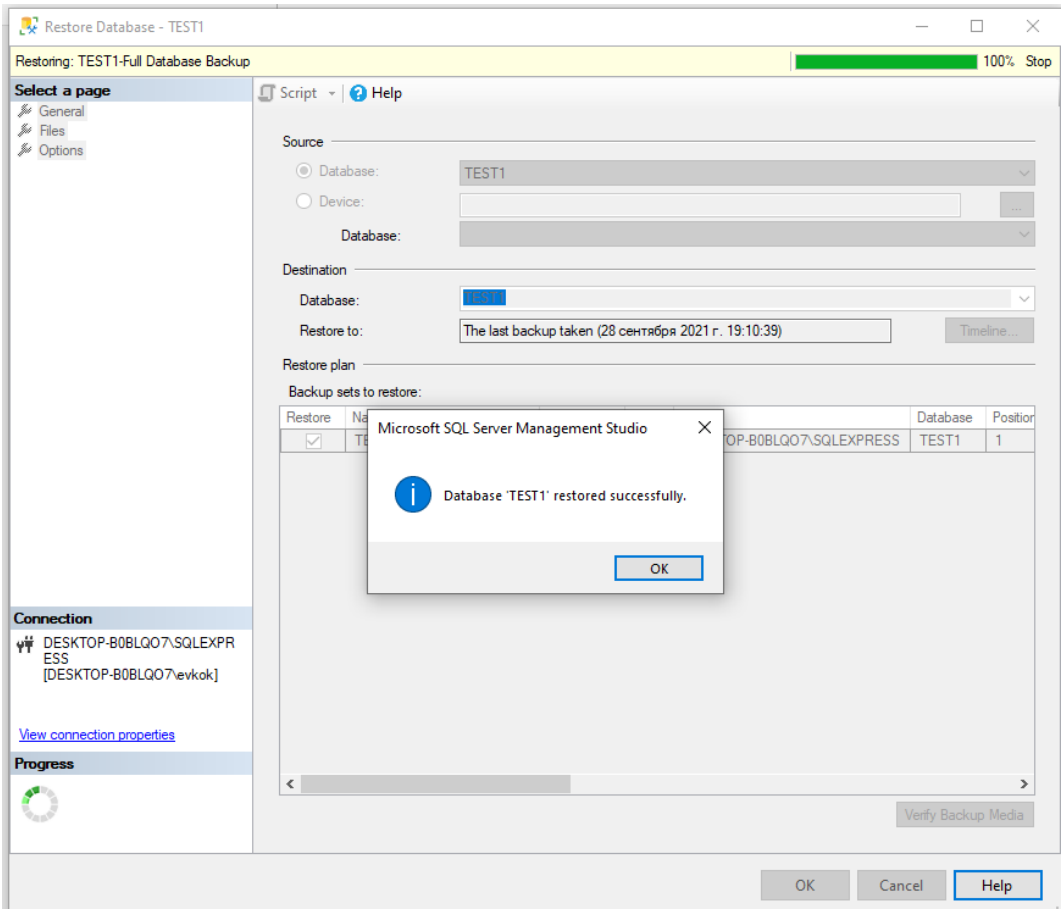


Рис. 8.1. Создать резервную копию

Для восстановления базы данных из резервной копии используется команда "База данных/ Восстановление базы данных (Restore), рис8.2. В результате откроется диалоговое окно "Мастер восстановления баз данных", в котором надо выбрать имя БД куда будет восстанавливаться база данных, в которую будет помещен результат, способ восстановления, файл, из которого будет восстанавливаться база данных, отмечаем выбранную резервную копию, и нажать кнопку [Восстановить]. Запускаем процесс восстановления.


Connect
DESKTOP-B0BLQO7\SQLEXPRESS (SQL Se
Databases
System Databases
Database Snapshots
DWConfiguration
DWDiagnostics
DWQueue
productsdb
New Database...
New Query
Script Database as
Tasks
Policies
Facets
Start PowerShell
Azure Data Studio
Reports
Rename
Delete
Refresh
Properties
Detach...
Take Offline
Bring Online
Enable
Encrypt Columns...
Data Discovery and Classification
Vulnerability Assessment
Shrink
Back Up...
Restore
Generate Scripts...
Extract Data-tier Application...
Deploy Database to Microsoft Azure SQL Database...
Export Data-tier Application...
Database...
Files and Filegroups...
Transaction Log...


Restore Database - TEST1
Ready
Select a page
General
Files
Options
Script
Help
Source
Database:
TEST1
Device:
Database:
Destination
Database:
TEST1
Restore to:
The last backup taken (28 сентября 2021 г. 19:10:39)
Timeline...
Restore plan
Backup sets to restore:
Restore
Name
Component
Type
Server
Database
Position
TEST1-Full Database Backup
Database
Full
DESKTOP-B0BLQO7\SQLEXPRESS
TEST1
1
Verify Backup Media
Connection
DESKTOP-B0BLQO7\SQLEXPRESS
[DESKTOP-B0BLQO7\evkok]
View connection properties
Progress
Ready
OK
Cancel
Help



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

Перенос базы данных между серверами MS SQL Server

Правой кнопкой мыши нажать на БД которую необходимо перенести и в контекстном меню выбрать «Задачи – Сформировать скрипты». Указать путь для сохранения скрипта.

Задание

1. Изучить способы создания, изменения и удаления таблиц. Получить навыки использования приложения " SQL Server Management Studio " для создания, удаления и изменения структуры таблиц.

2. Используя инструменты SQL Management Studio создать таблицы.

Язык запросов SQL

Появление и развитие языка SQL связано с созданием теории реляционных БД. Математической основой языка SQL является реляционная алгебра и реляционное исчисление. Прообраз языка возник в 1970 году в лаборатории Санта-Тереза фирмы IBM. В настоящее время популярность SQL настолько велика, что разработчики нереляционных СУБД снабжают свои системы SQL-интерфейсом. SQL сочетает в себе возможности языка определения данных, языка манипулирования данными и языка запросов. При этом он реализует и основные функции реляционных СУБД. SQL не является языком программирования в традиционном представлении. На нем пишутся не программы, а запросы к БД, поэтому этот язык называют языком запросов, языком декларативным, а не процедурным. Это означает, что с его помощью можно сформулировать, что необходимо получить, однако нельзя указать, как это следует сделать. В отличие от процедурных языков программирования (Си, Паскаль), в языке SQL отсутствуют алгоритмические конструкции, операторы цикла, условные переходы и т.д.

Язык SQL выходит в состав 4 семейств языков и выполняет их роли. SQL является языком **DDL** (Data Definition Language – язык описания данных) так как реализует следующие функции:

- **CREATE** – создание объектов в БД;
- **ALTER** – изменение структуры БД;
- **DROP** – удаление объектов из БД;
- **TRUNCATE** – удаление всех записей из БД;
- **COMMENT** – добавление комментариев;
- **RENAME** – изменение имени объекта БД.

SQL является языком **DML** (Data Manipulation Language – язык манипулирования данных) так как реализует следующие функции:

— **SELECT** – извлечение данных из БД;

— **UPDATE** – обновление данных в БД;

— **DELETE** – удаление данных из БД;

— **INSERT** – вставка данных в БД;

— **MERGE** – вставка и обновление;

— **CALL** – вызов **PL/SQL** или Java подпрограммы;

— **EXPLAIN PLAN** – создание планов выполнения запросов;

— **LOCK TABLE** – для блокирования таблиц.

SQL является языком **DCL** (Data Control Language – язык управления данными) так как реализует следующие функции:

— **GRANT** – предоставляет права доступа пользователям к БД;

— **REVOKE** – отмена прав доступа.

SQL является языком **TCL** (Transaction Control Language – язык транзакций) так как реализует следующие функции:

— **COMMIT** – сохранение работы;

— **SAVEPOINT** - определение точки транзакции;

— **ROLLBACK** – восстановление БД на оригинал с последним **COMMIT**;

— **SET TRANSACTION** – изменение параметров транзакций.

Запрос в языке SQL состоит из одного или нескольких операторов, следующих один за другим и разделенных точкой с запятой. Каждая последовательность операторов языка SQL реализует определенное действие над БД. Оно осуществляется за несколько шагов, на каждом из которых над таблицами выполняются определенные действия. Каждый оператор SQL начинается с ключевого слова, которое определяет, что делает этот оператор (SELECT, INSERT, DELETE). В операторе содержатся предложения,

содержащие сведения о том, над какими данными производятся операции. Каждое предложение начинается с ключевого слова, такого как `FROM`, `WHERE` и др. Структура предложения зависит от его типа: ряд предложений содержит имена полей или таблиц, некоторые могут включать дополнительные ключевые слова, константы или выражения.

Создание таблиц

После создания базы данных мы пока не может вносить в нее какие-либо данные т.к. пока еще не создана структура куда мы можем их поместить и работать с ними. Так ка Microsoft SQL Server – реляционная СУБД, поэтому все данные в Sql хранятся в виде двумерных таблиц со строками и столбцами. Строки называются кортежами или записями, а столбцы – доменами или полями.

Вся информация обрабатывается по записям. Каждая запись состоит из 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.

Основные ограничения, которым должны удовлетворять таблицы:

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

В SQL сервер используется следующие типы данных:

Битовые типы данных, которые содержат последовательности нулей и единиц: Binary(n) и Varbinary(n), где n длина. Содержимое полей типа Binary всегда равно n, разница заполняется пробелами. Varbinary размер поля равен n или большему;

Целочисленные типы данных – типы данных для хранения целых чисел (в скобках указан диапазон значений типа данных): Tinyint (0-255), Smallint (±32000), Int (±2000000000), Bigint (±263);

Типы данных для хранения дробных чисел:

Типы числовых данных с фиксированными точностью и масштабом. Типы decimal и numeric являются взаимозаменяемыми синонимами.

decimal[(p[, s])] и numeric[(p[, s])] Числа с фиксированной точностью и масштабом. При использовании максимальной точности числа могут принимать значения в диапазоне от -10^38+1 до 10^38-1. Синонимами типа decimal по стандарту ISO являются типы dec и dec(p, s). Тип numeric функционально эквивалентен типу decimal.

p (точность). Максимальное общее число хранимых десятичных разрядов. Это число включает символы слева и справа от десятичной запятой. Точность должна быть значением в диапазоне от 1 до максимум 38. Точность по умолчанию составляет 18.

s (масштаб). Максимальное число хранимых десятичных разрядов справа от десятичной запятой. Это число отнимается от *p* для определения максимального количества цифр слева от десятичной запятой. Масштаб должен иметь значение от 0 до *p* и может быть указан только при заданной точности. По умолчанию масштаб принимает значение 0, поэтому 0 <= *s* <= *p*. Максимальный размер хранилища зависит от точности.

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. Если указан параметр *n*, это должно быть значение в диапазоне от 1 до 53. Значение *n* по умолчанию — 53. В приложении SQL Server параметр n может принимать одно из двух возможных значений. Если 1<=n<=24, n принимает значение 24. Если 25<=n<=53, n принимает значение 53.

Специальные типы данных: Bit – логический тип данных является заменой логическому типу Boolean в Visual Basic, Text - тип для хранения

больших объемов текста, одно поле может хранить до 2 Гб текста, Image – тип данных для хранения до 2Гб рисунков, RowGUID – уникальный идентификатор строки таблицы, SQL_Variant - аналогичен типу Variant в Visual Basic;

Типы данных даты и времени: Datetime (от 1.01.1953 до 3.12. 1999). SmallDatetime (от 1.01.19 до 6.07 2079);

Денежные типы данных для хранения финансовой информации: Money (±1015 и 4 знака после нуля), Smallmoney (± 20000,0000);

Автоматически обновляемые типы данных - аналоги счетчиков, но в данной роли они не используются: RowVersion уникальный идентификатор строки. TimeStamp – закодированное дата и время создания строки.

Ход работы

1. Создайте новую БД `Training_base_Фамилия` (способ создания выберите самостоятельно).

2. Создайте таблицу `Salespeople (Продавцы)`. Для этого щёлкните правой кнопкой мыши по папке `Tables` («Таблицы») и в появившемся меню выберите пункт `New Table` (Создать таблицу), рис.10. Появится окно создания новой таблицы (рис. 11).

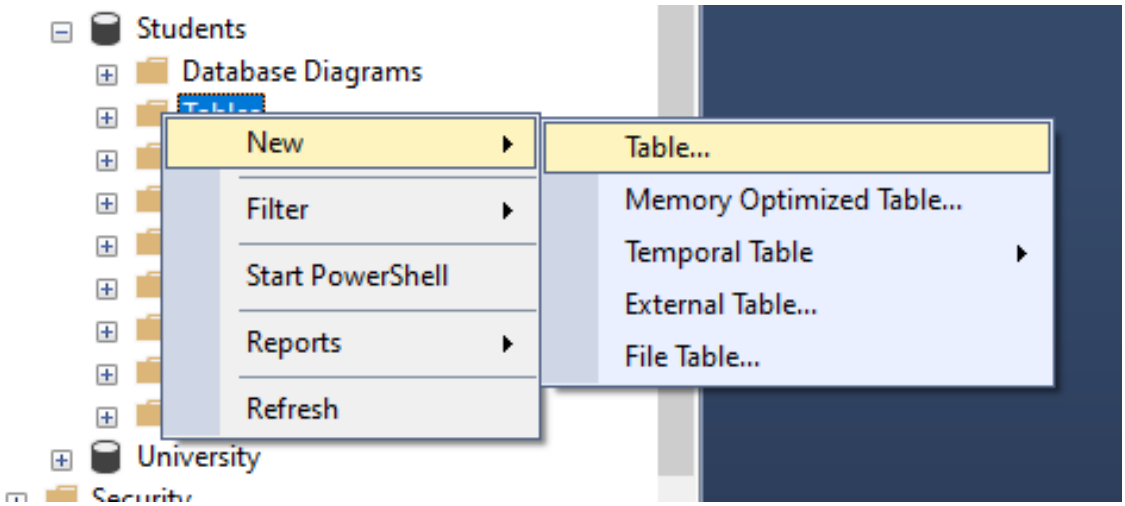


Рис. 10. Выбор команды Создать таблицу

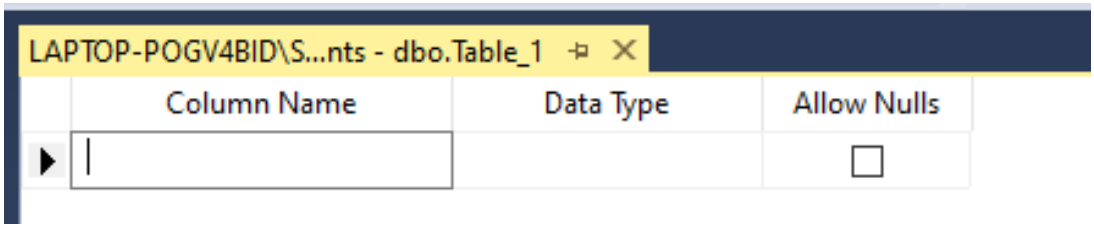


Рис. 11. Окно создания таблицы

Данная таблица имеет следующие столбцы:

— `Column Name` (Имя столбца) — имя столбца должно всегда начинаться с буквы и не должно содержать различных специальных символов и знаков препинания. Если имя поля содержит пробелы, то оно автоматически заключается в квадратные скобки.

— `Data Type` (Тип данных) — тип данных поля.

— `Allow Nulls` (Разрешить значения `Null`) — допуск значения `Null`. Если эта опция поля включена, то в случае незаполнения поля в него будет автоматически подставлено значение `Null`. То есть поле необязательно для заполнения. Замечание: Под таблицей определения полей располагается таблица свойств выделенного поля `Column Properties` (Свойства столбца). В данной таблице настраиваются свойства выделенного поля. Некоторые из них будут рассмотрены ниже. Перейдем к созданию полей и к настройке их свойств. В таблице определения полей задайте значения столбцов `Column Name` («Имя столбца»), `Date Type` (Тип данных) и `Allow Nulls` (Разрешить значения `Null`), как показано на рис. 12.

Из рис. 12 следует, что таблица `Salespeople` имеет четыре поля:

— `SNUM` — уникальный номер (ключевое поле), приписанный каждому продавцу (номер служащего), числовой, целое. Чтобы сделать поле ключевым, выделите поле, а затем на панели инструментов нажмите кнопку с изображением ключа (или в контекстном меню поля). В таблице определения полей, рядом с полем `SNUM` появиться изображение ключа, говорящее о том, что поле ключевое;

— SNAME — имя продавца, текстовое поле, предназначенное для хранения строк, имеющих длину не более 10 символов;

— CITY — Место расположения продавца, текстовое поле, предназначенное для хранения строк, имеющих длину не более 10 символов.

— COMM – вознаграждение (комиссионные) продавца, числовое поле с плавающей точкой (decimal (6,2) означает, 6 - максимальное общее число хранимых десятичных разрядов, включая символы слева и справа от десятичной запятой, 2 - число хранимых десятичных разрядов справа от десятичной запятой).

— Сохраните таблицу под именем Salespeople. Обновите папку Таблицы. Таблица Salespeople отобразится в обозревателе объектов в папке Tables (Таблицы) БД «Training_base» (Рис. 13).

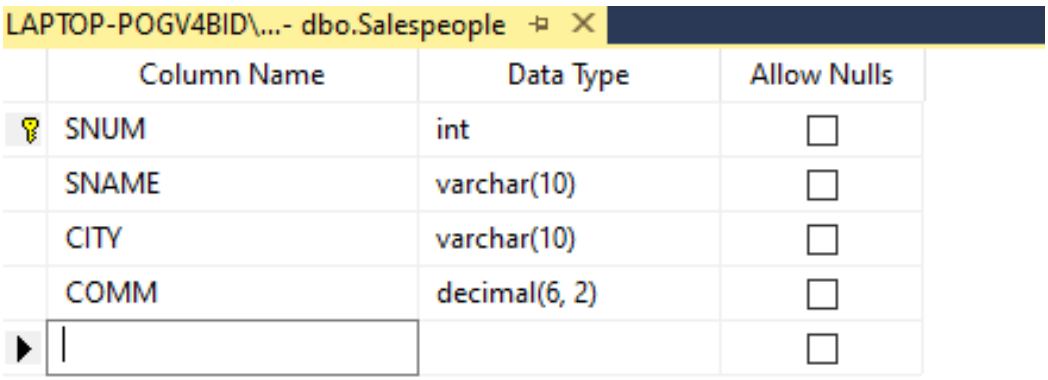


Рис. 12. Создание таблицы Salespeople

В обозревателе объектов таблица Salespeople отображается как dbo.Salespeople. Префикс dbo обозначает, что таблица является объектом БД (Data Base Object). В дальнейшем при работе с объектами БД префикс dbo можно опускать.

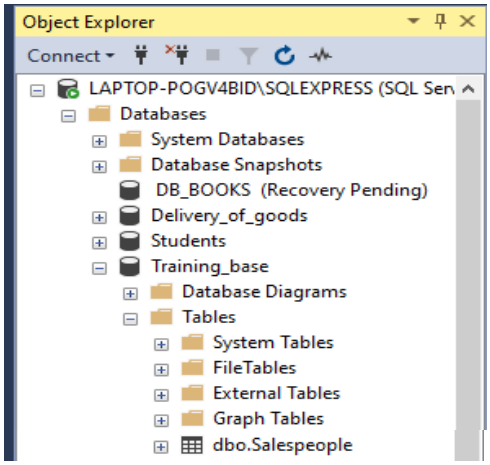


Рис. 13. Созданная таблица `Salespeople` в окне обозревателя

3. Создайте вторую таблицу- `Customers (Покупатели)`.

4. Добавьте в нее поля: (в соответствии с рис. 14).

— `CNUM` – уникальный номер, присвоенный покупателю;

— `CNAME` – имя покупателя;

— `CITY` – место расположения покупателя;

— `RATING` – цифровой код, определяющий уровень предпочтения данного покупателя. Чем больше, тем больше предпочтение.

— `SNUM` – номер продавца, назначенного данному покупателю (из таблицы `Salespeople)`.

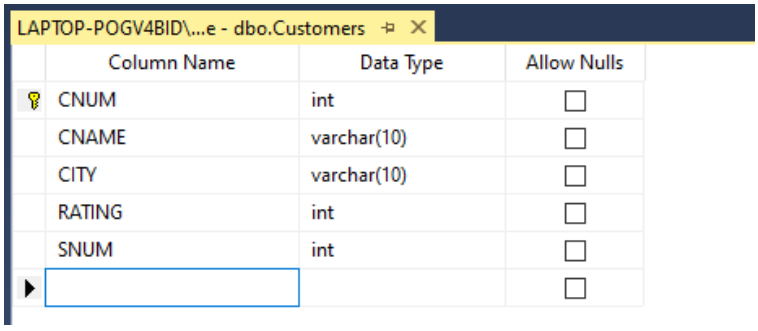


Рис. 14. Создание таблицы `Customers`

5. Создадите таблицу `Orders` (`Заказы`). Включите в нее следующие поля:

— `ONUM` – уникальный номер, присвоенный данной покупке;

— `AMT` – количество (сумма заказа);

— ODATE – дата заказа (покупки);

— CNUM – номер покупателя, сделавшего покупку (из таблицы (Customers);

— SNUM – номер продавца, обслуживающего покупателя (из таблицы Salespeople).

Поле ONUM (уникальный номер, присвоенный данной покупке) является первичным ключом, в данном примере будет числовым **счётчиком**. То есть данное поле должно автоматически заполняться числовыми значениями. Более того, оно должно быть **ключевым**.

Сделайте поле ONUM **счётчиком**. Для этого выделите поле, просто щёлкнув по нему мышкой в таблице определения полей. В таблице свойств поля отобразятся свойства поля ONUM. Разверните группу свойств Identity Specification (Спецификация идентификатора). Свойство (Is Identity) (Идентификатор) установите в значение Yes (Да). Задайте свойства Identity Seed («Начальное значение идентификатора») и Identity Increment («Шаг приращения идентификатора») равными 1 (Рис. 15). Эти настройки показывают, что значение поля ONUM у первой записи в таблице будет равным 1, у второй — 2, у третьей — 3 и т.д. Теперь сделайте поле ONUM ключевым.

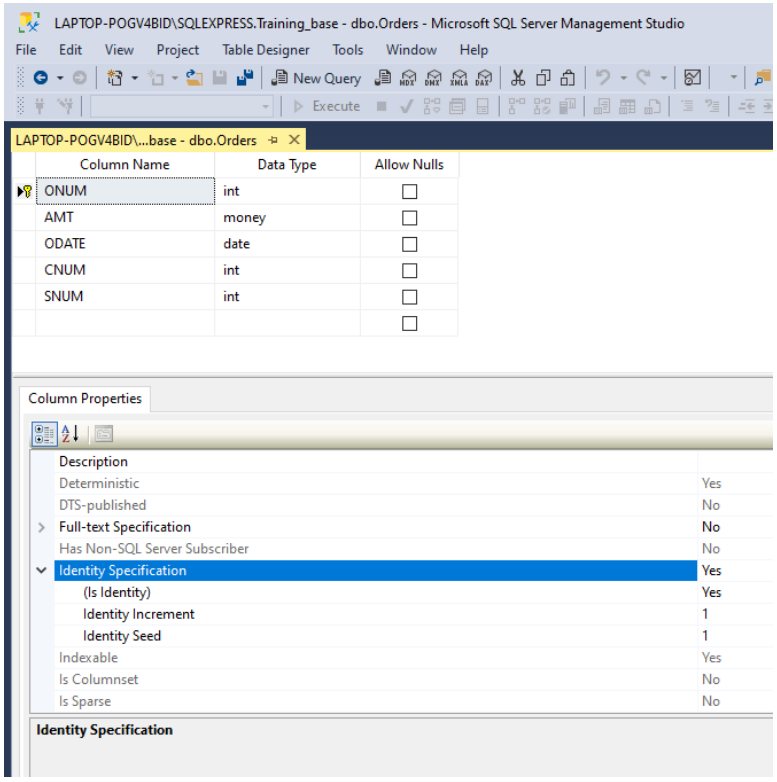


Рис.15. Таблица `Orders`

Создание таблиц с помощью запроса

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 (
    column1 datatype,
    column2 datatype,
    column3 datatype,
   ....
);
```

Здесь:

`table_name` – имя создаваемой таблицы;

`column1`– имя первого поля таблицы;

`datatype` – тип поля;

Если имя поля содержит пробел, то оно заключается в квадратные скобки.

Пример создания таблицы с помощью запроса приведен на рис.15.1.

```
CREATE TABLE Student (
ID_student INTEGER NOT NULL,
Surname VARCHAR(20) NOT NULL,
ID_specialization INTEGER CHECK(ID_specialization<12),
Note VARCHAR(20) DEFAULT 'зачислен'
CONSTRAINT PK_Stud PRIMARY KEY (ID_student)
);
```

Рис. 15.1. Запрос на создание таблицы Student

`NOT NULL` – столбец не может содержать значение `NULL`, т. е. значения этого столбца должны быть определены.

`PRIMARY KEY` – столбец является первичным ключом. В каждой таблице только один столбец (составной) может быть первичным ключом. Это означает, что он не может содержать значение NULL, а вводимое в него значение должно отличаться от всех остальных значений в этом столбце.

`DEFAULT значение` – устанавливает значение по умолчанию.

`CHECK (условие)`– позволяет производить проверку условия при вводе данных. Значение будет сохранено, если условие выполняется, в противном случае – нет.

Если необходимо использовать поле в качестве числового счетчика, то запрос на создание таблицы может быть создан, как показано на рис. 15.2.

Ограничение PRIMARY KEY может также быть применено для нескольких полей, составляющих уникальную комбинацию значений — составной первичный ключ. Пример запроса приведен на рис. 15.3.

```
CREATE TABLE Факультет
(
    Код_факультета int IDENTITY (1,1) NOT NULL,
    Наименование_факультета varchar(50) NOT NULL,
    Фамилия_декана varchar(50) NOT NULL,
    Телефон_декана int NOT NULL
 CONSTRAINT PK_Факультет PRIMARY KEY (Код_факультета)
);
```

Рис. 15.2. Запрос на создание таблицы Факультет

```
CREATE TABLE NEW_EXAM_MARKS
(EXAM_ID INT NOT NULL,
STUDENT_ID INT NOT NULL,
SUBJ_ID INT NOT NULL,
MARK INT,
DATA DATE,
CONSTRAINT EX_PR_KEY PRIMARY KEY (EXAM_ID, STUDENT_ID));
```

Рис. 15.3. Запрос на создание таблицы с составным первичным ключом

6. Создайте запрос на создание таблиц `Salespeople_1`, `Customers_1` и `Orders_1`. Выполните его.

Заполнения таблиц начальными данными

1. Для начала заполните таблицу `Salespeople`. Для заполнения этой таблицы в обозревателе объектов в контекстном меню таблицы `Salespeople` (Рис. 22) и в появившемся меню выберите пункт Edit Top 200 Rows (Изменить первые 200 строк). В рабочей области MS SQL Server Management Studio появится окно заполнения таблиц.

2. Заполните таблицу `Salespeople`, как показано на рис.23.

3. Заполним таблицу `Customers` (рис.24).

4. Заполним таблицу `Orders` (рис.25). Так как поле `ONUM` является первичным полем связи и **ключевым числовым счётчиком**, то оно заполняется автоматически (заполнять его не нужно).

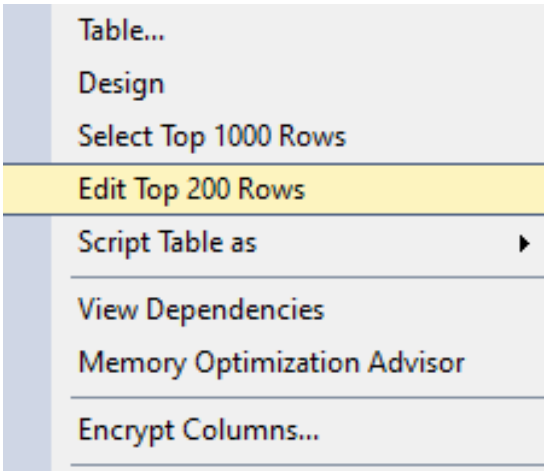


Рис. 22. Изменить данные в таблице

LAPTOP-POGV4BID\...- dbo.Salespeople

SNUM	SNAME	CITY	COMM
1001	Peel	London	0,12
1002	Serres	San Jose	0,13
1003	Axelrod	New York	0,10
1004	Motika	London	0,11
1007	Rifkin	Barselona	0,15
NULL	*NULL*	*NULL*	*NULL*

Рис. 23. Заполнение таблицы `Salespeople`

LAPTOP-POGV4BID\...e - dbo.Customers

CNUM	CNAME	CITY	RATING	SNUM
2001	Hoffman	London	100	1001
2002	Giovanni	Rome	200	1003
2003	Liu	San Jose	200	1002
2004	Grasse	Berlin	300	1002
2006	Clemens	London	100	1001
2007	Pereira	Rome	100	1004
2008	Cisneros	San Jose	300	1007
NULL	*NULL*	*NULL*	*NULL*	*NULL*

Рис. 24. Заполнение таблицы `Customers`

LAPTOP-POGV4BID\...base - dbo.Orders

ONUM	AMT	ODATE	CNUM	SNUM
1	18,6900	2019-03-10	2008	1007
2	767,1900	2019-03-10	2001	1001
3	1900,1000	2019-03-10	2007	1004
4	5160,4500	2019-03-10	2003	1002
5	1098,1600	2019-03-10	2008	1007
6	1713,2500	2019-04-10	2002	1003
7	78,7800	2019-04-10	2004	1002
8	4756,2300	2019-05-10	2006	1001
9	1309,2500	2019-06-10	2004	1002
10	9897,8800	2019-06-10	2006	1001
NULL	NULL	NULL	NULL	NULL

Рис. 25. Заполнение таблицы `Orders`

Добавление данных в таблицу

В SQL Server 20XX заполнение таблиц производится при помощи следующей команды:

```
INSERT INTO <Имя таблицы> (<Список полей>)
VALUES (<Значения полей>);
```

<Имя таблицы> – таблица, куда вводим данные.

<Список полей> – список полей, куда вводим данные, если не указываем, то подразумевается заполнение всех полей, в списке полей поля указываются через запятую.

<Значения полей> – значение полей через запятую.

Пример

```
INSERT INTO Posts (P_POST, P_SAL)
VALUES ('Менеджер', 15000);
```

Или для добавления множества записей можно использовать конструкция из следующего примера

```
INSERT INTO Posts VALUES
('Менеджер', 15000),
('Редактор', 20000),
('Программист', 50000),
('Главный редактор', 150000)
```

Строковые данные, как и даты в запросах указываются в одинарных кавычках, числовые данные указываются без кавычек. VALUES позволяет использовать один или несколько списков вставляемых значений данных. Список значений должен быть заключен в скобки. Если нет необходимости заполнять все поля, то можно указать только те имена полей, значения которых необходимо изменить. В перечислении можно указывать поля в любом порядке, но в списке VALUES значения должны идти в том же порядке, в котором перечислены поля.

```
INSERT INTO tbPeoples (vcSurname, idPosition, vcName)
VALUES ('Смирнов', 12, 'Сергей')
```

5. Создайте запрос на добавление записей в таблицы Salespeople_1, Customers_1 и Orders_1.

Удаление отдельных столбцов и отдельных строк из таблицы

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

```
WHERE <Условие>
```

Примечание: WHERE – означает условие, т.е. в данном случае мы удаляем данные, если выполняется условие, которое находится после ключевого слова WHERE.

Минимальная команда для удаления выглядит следующим образом:

```
DELETE <Имя таблицы>
```

Например,

```
DELETE tbPeoples
```

Эта команда удаляет все строки из таблицы tbPeoples.

Либо для удаления таблицы полностью:

```
DROP TABLE <Имя таблицы>
```

Для полной очистки таблицы и сброса счетчика AUTOINCREMENT используется конструкция

```
TRUNCATE TABLE <Имя таблицы>
```

Пример использования этой конструкции представлен ниже:

```
TRUNCATE TABLE Posts;
```

Инструкция TRUNCATE TABLE выполняется быстрее, чем инструкция DELETE, и использует меньше системных ресурсов и ресурсов журнала транзакций. Если условие указано, то удаляются записи поля, которые соответствуют условию. Ключевое слово TOP задает количество или процент удаляемых случайных строк. Если с инструкцией DELETE применяется предложение TOP (n), то операция удаления производится над n случайно выбранных строк.

Пример: Удалить 20 случайных строк из таблицы `PurchaseOrderDetail`, имеющих дату ранее 1 июля 2006 г

```
DELETE TOP (20)
FROM Purchasing.PurchaseOrderDetail
WHERE DueDate < '2006/07/01';
```

Пример:

Удалить записи из таблицы `Posts`, у которых значение поля `P_SAL > 100000`

```
DELETE Posts
WHERE P_SAL > 100000
```

Пример:

Удалить записи из таблицы `Posts`, у которых значение поля `P_Post` будет содержать часть слова «джер»

```
DELETE FROM Posts WHERE P_Post LIKE '%джер';
```

Пример:

Удалить записи из таблицы `Posts`, у которых значение поля `P_ID` будет между 5 и 8

```
DELETE FROM Posts WHERE P_ID BETWEEN 5 AND 8;
```

Для этого используется следующая команда:

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

<Имя поля1>, <Имя поля2> - имена изменяемых полей,

<Выражение1>, <Выражение2> - либо конкретные значения, либо NULL, либо операторы SELECT. Здесь SELECT применяется как функция.

<Условие> – условие, которым должны соответствовать записи, поля которых изменяем.

Пример:

В таблице Posts переименовать название Менеджер в Редактора

```
UPDATE Posts
SET P_POST = 'Редактор'
WHERE P_Post = 'Менеджер'
```

Здесь командой SET в поле P_POST устанавливаем значение Редактор там, где по условию WHERE в этом поле значение равно Менеджер.

Пример:

Увеличить значение даты у всех строк на 1

```
UPDATE peoples
SET dayBirthday = dayBirthday +1
```

Из примера видно, что оператор UPDATE позволяет использовать математические вычисления.

Создание запросов

Оператор SELECT (ВЫБРАТЬ) языка **SQL** является самым важным и наиболее часто используемым оператором. Он предназначен для выборки информации из таблиц базы данных. Упрощенный синтаксис оператора **SELECT** выглядит следующим образом:

SELECT [DISTINCT] <список выражений над атрибутами и константами>

FROM <список таблиц>

[WHERE <условие выборки или соединения>]

[GROUP BY <список атрибутов>]

[HAVING <условие для группы>]

[UNION <выражение с оператором SELECT>]

[ORDER BY <список атрибутов, по которым упорядочить вывод>];

В квадратных скобках указаны элементы, которые могут отсутствовать в запросе.

Ключевое слово SELECT сообщает базе данных, что данное предложение является запросом на выборку информации. После слова SELECT через запятую перечисляются наименования полей (список атрибутов), содержимое которых запрашивается.

Обязательным ключевым словом в предложении-запросе SELECT является слово FROM (ИЗ). За ключевым словом FROM указывается список разделенных запятыми имен таблиц, из которых извлекается информация.

Задание

Создать запросы на извлечение данных.

Ход работы

Выполнение простых sql – запросов.

1. Подключиться к базе данных и выполнить команду `New Query`/Создать запрос.

2. Ввести текст запроса согласно рис.27.

3. Нажать на панели инструментов кнопку `Execute`/Выполнить.

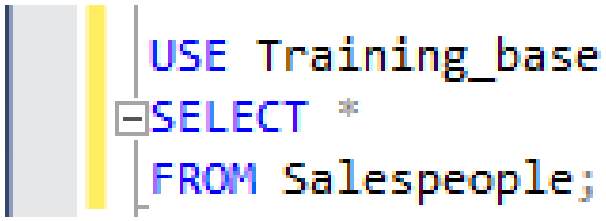
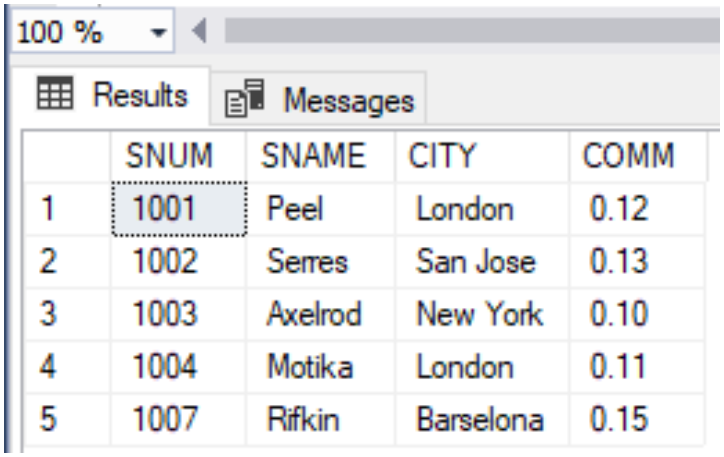


Рис. 27. Окно запроса 1

`USE` <Имя_БД> Обращение к БД

4. Если запрос правильный, то в результате произойдет его выполнение и Results / Результат будет отображен на вкладке Результаты (рис. 28).



	SNUM	SNAME	CITY	COMM
1	1001	Peel	London	0.12
2	1002	Serres	San Jose	0.13
3	1003	Axelrod	New York	0.10
4	1004	Motika	London	0.11
5	1007	Rifkin	Barselona	0.15

Рис. 28. Окно с результатом выполнения запроса

Приведенный запрос осуществляет выборку всех значений из таблицы Salespeople. Сохраните в текущей папке.

Запрос 2. Вывод данных таблицы Customers с присвоением псевдонима, рис. 29.

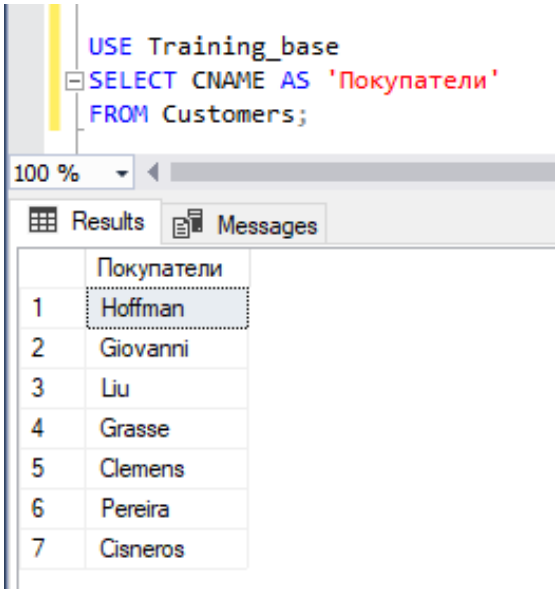


Рис. 29. Окно запроса 2

Команда AS изменяет имя столбца в результирующей таблице на имя, указанное после этой команды в кавычках (CNAME изменяется на Покупатели).

Запрос 3. Вывести информацию о продавцах с вознаграждением больше 0,12 (рис. 30).

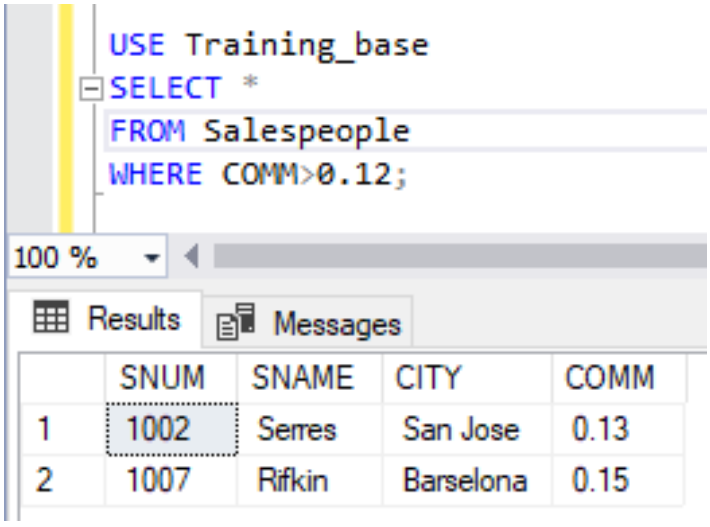


Рис. 30. Окно запроса 3

Предложение `WHERE` команды `SELECT` позволяет определить предикат, условие, которое может быть либо истинным, либо ложным для каждой строки

таблицы. Команда извлекает только те строки из таблицы, для которых предикат имеет значение «истина».

Запрос 4. Составное условие: извлечение из таблицы Salespeople записей, чьи коды больше 1001, но меньше или равен 1004, рис. 31.

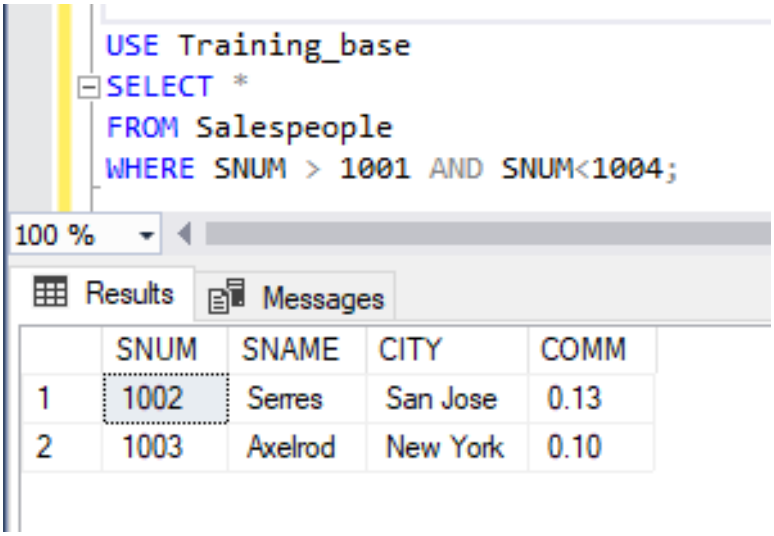


Рис. 31. Окно запроса 4

AND, OR, NOT – это стандартные булевы операторы. Они связывают одно или несколько значений «истина/ложь» и в результате получают единственное значение «истина/ложь».

AND берет два булевых выражения (в виде A AND B) в качестве аргументов и дает в результате истину, если они оба истинны.

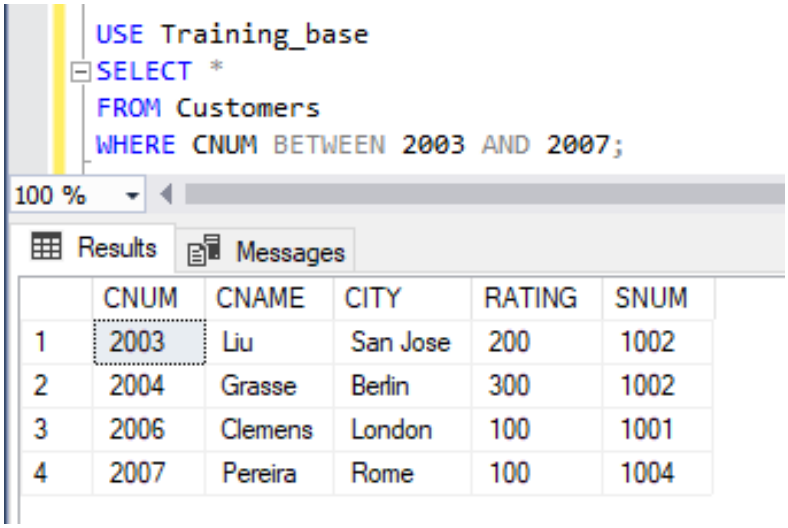
OR берет два булевых выражения (в виде A OR B) в качестве аргументов и оценивает результат как истину, если хотя бы один из них истинен.

NOT берет единственное булево выражение (в виде NOT A) в в качестве аргумента и изменяет его значение с истинного на ложное или с ложного на истинное.

Запрос 5. Составное условие: извлечение из таблицы Customers записей, чьи коды находятся между 2003 и 2007, рис. 32.

Оператор BETWEEN задает границы, в которые должно попадать значение, чтобы предикат был истинным 9граничные значения тоже включаются в этот диапазон).

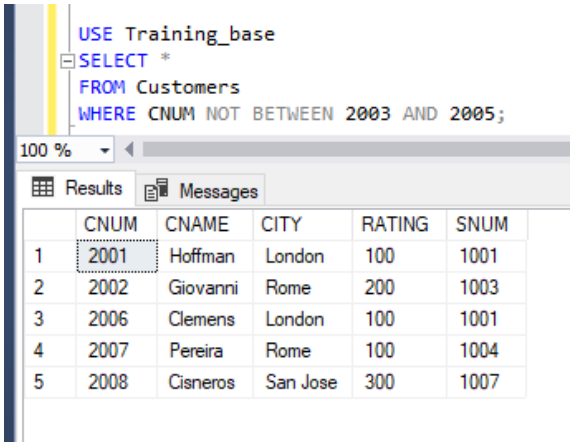
Оператор IN полностью определяет множество, которому данное значение может принадлежать.



	CNUM	CNAME	CITY	RATING	SNUM
1	2003	Liu	San Jose	200	1002
2	2004	Grasse	Berlin	300	1002
3	2006	Clemens	London	100	1001
4	2007	Pereira	Rome	100	1004

Рис. 32. Окно запроса 5

Запрос 6. Составное условие: требуется что бы выводились строки где CNUM вне указанного далее диапазона т.е. то, что находится между 2003 и 2005 не выводилось, рис. 36.



	CNUM	CNAME	CITY	RATING	SNUM
1	2001	Hoffman	London	100	1001
2	2002	Giovanni	Rome	200	1003
3	2006	Clemens	London	100	1001
4	2007	Pereira	Rome	100	1004
5	2008	Cisneros	San Jose	300	1007

Рис. 36. Окно запроса 6

Запрос 7. Вывод записей, удовлетворяющих не диапазону, а списку `IN` указывает, что необходимо в результирующую таблицу поместить только те строки, значения поля SNUM, которых равны значениям, указанным в скобках после `IN`, рис. 37.

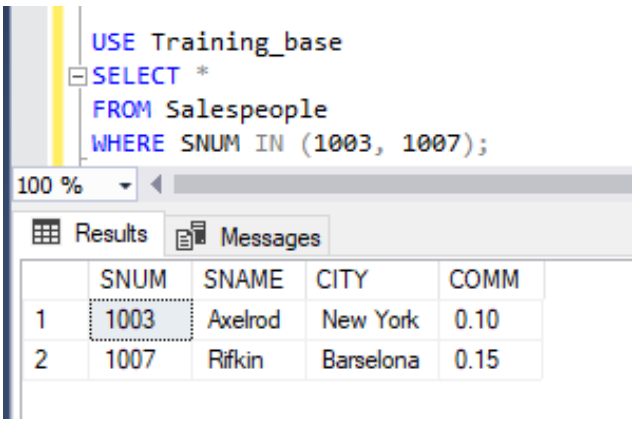


Рис. 37. Окно запроса 7

Запрос 8. Вывод записей, удовлетворяющих условию, заданному текстом: вывести все записи о продавцах, с именемAxelrod, рис. 38.

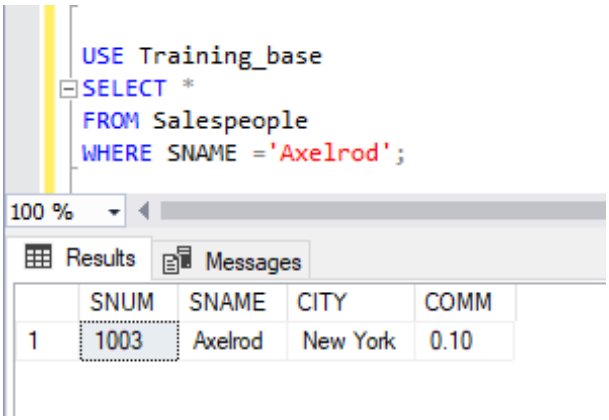


Рис. 38. Окно запроса 8

Запрос 9. Вывод записей, удовлетворяющих условию, заданному частью текста. `LIKE` дает возможность указать какую-либо часть значения. В данном случае мы получаем все строки, значения поля CNAME (имя покупателя), которых будет начинаться с буквы G, рис. 39.

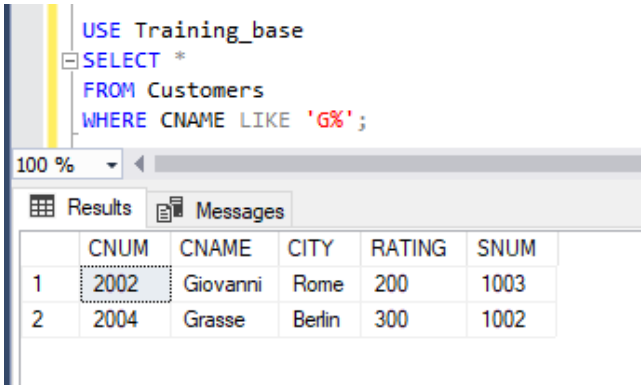


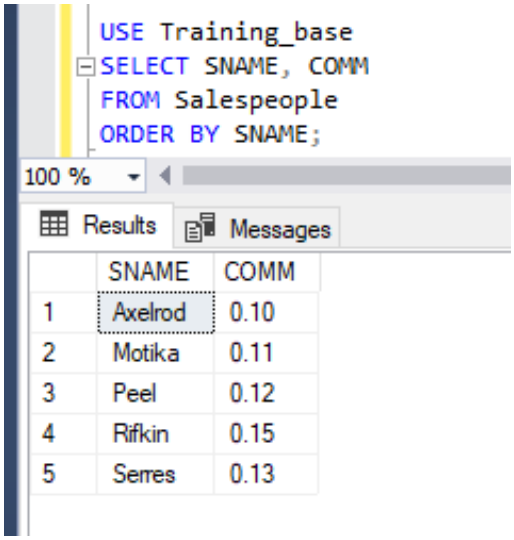
Рис. 39. Окно запроса 9

Оператор LIKE применим только к символьным типам полей, поскольку он используется для поиска подстрок. Он осуществляет просмотр строки для выяснения: входит ли заданная подстрока в указанное поле. С этой же целью используются шаблоны – специальные символы, которые могут обозначать все, что угодно.

Символ «подчеркивание» (_) заменяет один любой символ. Например, образцу ‘b_t’ соответствуют ‘bat’ или ‘bit, но не соответствует ‘brat’.

Символ «процент» (%) заменяет последовательность символов произвольной длины, в том числе и нулевой. Например, образцу ‘%p%t’ соответствуют ‘put’, ‘posit’, ‘opt’, но не ‘spite’

Запрос 10. Сортировка по значению одного из столбцов. В данном случае сортировка по полю SNAME, рис. 40.



	SNAME	COMM
1	Axelrod	0.10
2	Motika	0.11
3	Peel	0.12
4	Rifkin	0.15
5	Serres	0.13

Рис. 40. Окно запроса 10

Таблицы являются неупорядоченными множествами, и исходящие из них данные необязательно представляются в какой-либо определенной последовательности. Команда ORDER BY упорядочивает в соответствии со значениями одного или нескольких выбранных столбцов. Можно задавать возрастающую (ASC) или убывающую (DESC) последовательность сортировки для каждого из столбцов.

Запрос 11. Извлечение из таблицы Salespeople записей, номер которых от 1003 до 1007 с сортировкой по полю SNUM и полю SNAME, рис. 41.

```
USE Training_base
SELECT SNUM, SNAME
FROM Salespeople
WHERE SNUM BETWEEN 1003 AND 1007 ORDER BY SNUM, SNAME;
```

100 %

Results | Messages

	SNUM	SNAME
1	1003	Axelrod
2	1004	Motika
3	1007	Rifkin

Рис. 41. Окно запроса 11

Запрос 12. Изменение порядка сортировки. `DESC` означает сортировка по убыванию, рис. 42.

```
USE Training_base
SELECT CNAME
FROM Customers
 ORDER BY CNUM DESC;
```

100 %

Results | Messages

	CNAME
1	Cisneros
2	Pereira
3	Clemens
4	Grasse
5	Liu
6	Giovanni
7	Hoffman

Рис. 42. Окно запроса 12

Запрос 13. Извлечение трех записей с обратной сортировкой по полю CNUM, рис. 43.

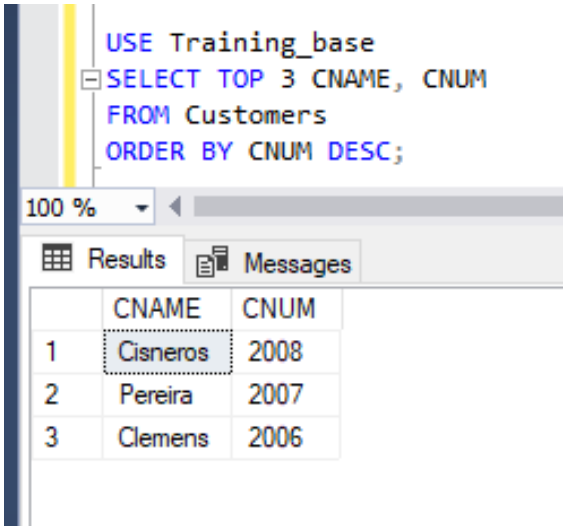


Рис. 43. Окно запроса 13

Функции агрегирования:

— `COUNT` определяет количество строк или значений полей;

— `SUM` вычисляет арифметическую сумму;

— `AVG` вычисляет среднее значение для всех выбранных значений данного поля;

— `MAX` вычисляет наибольшее значение;

— `MIN` вычисляет наименьшее значение.

Запрос 14. Подсчет среднего значения по всем комиссионным, с присвоением псевдонима, рис. 44.

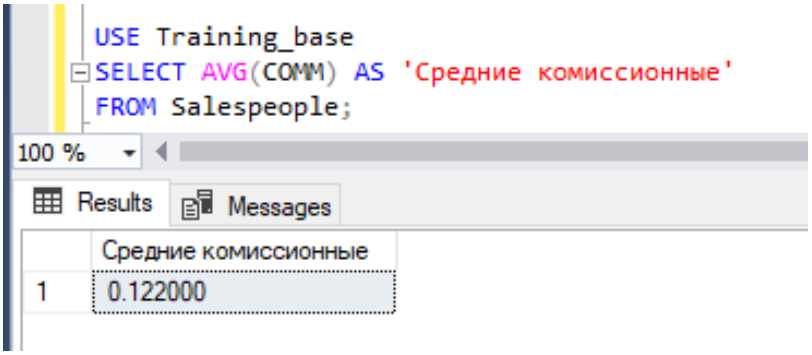


Рис. 44. Окно запроса 14

Запрос 15. Подсчет количества заказов (числа строк) в таблице Orders, значения столбца которых отличны от NULL, с присвоением псевдонима, рис. 45.

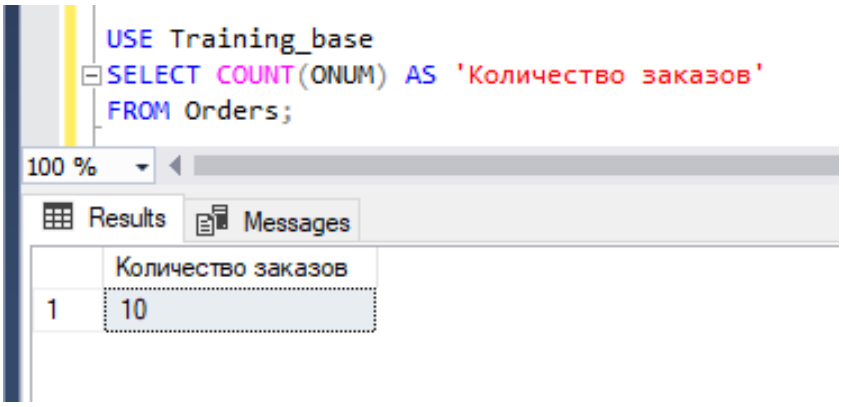


Рис. 45. Окно запроса 15

Запрос 16. Извлечение максимального значения столбца AMT, рис. 46.

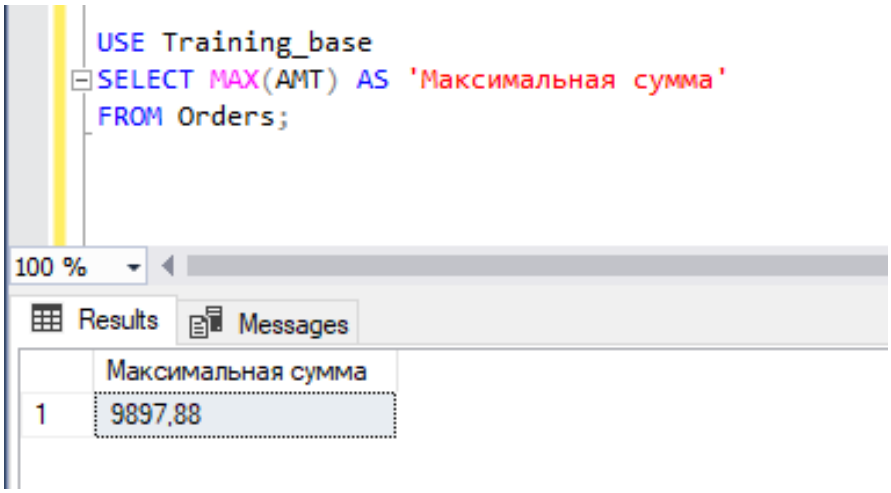
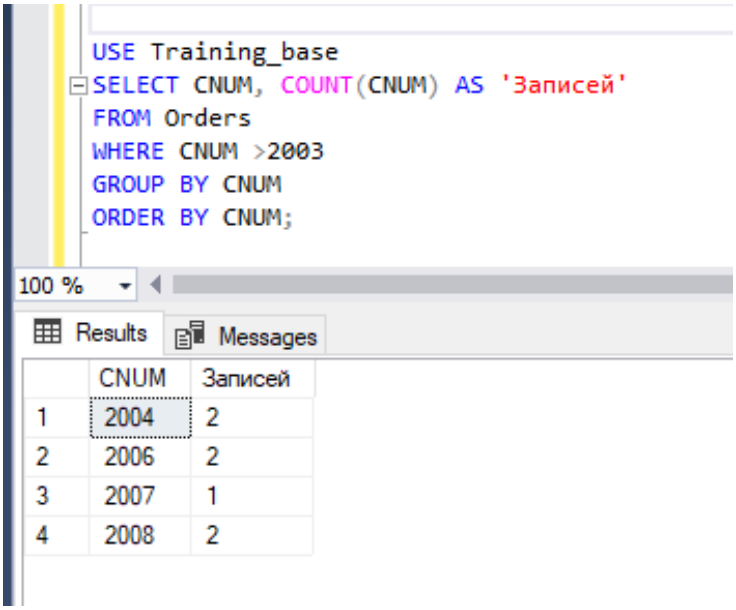


Рис. 46. Окно запроса 16

Предложение `GROUP BY` (группировать по) позволяет определять подмножество значений отдельного поля в терминах другого поля и применять функции агрегирования к полученному подмножеству.

Запрос 17. Вывод числа записей, соответствующих каждому из уникальных значений CNUM больше 2003, рис. 47.



```
USE Training_base
SELECT CNUM, COUNT(CNUM) AS 'Записей'
FROM Orders
WHERE CNUM >2003
GROUP BY CNUM
ORDER BY CNUM;
```

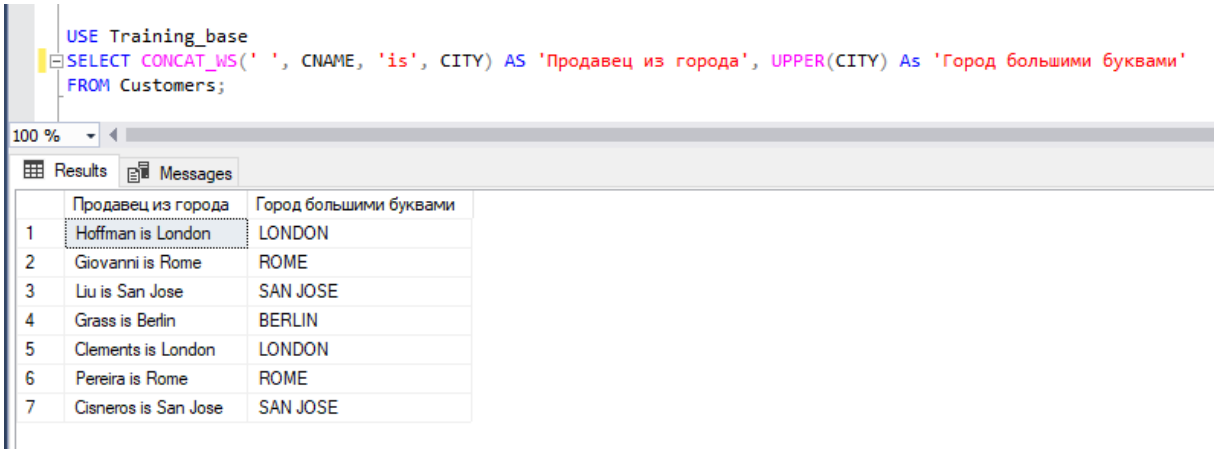
100 %

Results | Messages

	CNUM	Записей
1	2004	2
2	2006	2
3	2007	1
4	2008	2

Рис. 47. Окно запроса 17

Запрос 18. Выборка данных из таблицы Customers, где сцеплены CNAME (имя покупателя) и CITY (город), населенный пункт записан прописными буквами, рис. 48.



```
USE Training_base
SELECT CONCAT_WS(' ', CNAME, 'is', CITY) AS 'Продавец из города', UPPER(CITY) As 'Город большими буквами'
FROM Customers;
```

100 %

Results | Messages

	Продавец из города	Город большими буквами
1	Hoffman is London	LONDON
2	Giovanni is Rome	ROME
3	Liu is San Jose	SAN JOSE
4	Grass is Berlin	BERLIN
5	Clements is London	LONDON
6	Pereira is Rome	ROME
7	Cisneros is San Jose	SAN JOSE

Рис. 48. Окно запроса 18