

ОГЛАВЛЕНИЕ

Введение.....	5
1 Общие сведения о СУБД MS SQL Server.....	7
1.1 Информация о выпусках	7
1.2 Минимальные системные требования	7
1.3 Компоненты сервера.....	8
2 Администрирование сервера на уровне операционной системы ...	10
2.1.1 Развертывание виртуальной машины.....	10
2.1.2 Управление запуском и остановкой сервера	12
2.1.3 Предоставление прав доступа к файловой системе	15
2.1.4 Добавление новых жестких дисков.....	17
2.2 Контрольные задания	22
3 Ядро СУБД Database Engine	23
3.1 Подключение к Database Engine	23
3.1.1 Программное обеспечение для работы с экземпляром сервера.....	23
3.1.2 Подключение к экземпляру сервера	25
3.2 Системные базы данных MS SQL Server.....	30
3.2.1 База данных master	30
3.2.2 База данных model	31
3.2.3 База данных tempdb	32
3.2.4 База данных msdb.....	33
3.2.5 Перенос файлов системных баз данных.....	33
3.3 Пользовательские базы данных	36
3.3.1 Создание пользовательской базы данных с помощью MS SSMS	36
3.3.2 Создание базы данных средствами Transact SQL.....	39
3.4 Управление размещением базы данных	42
3.4.1 Файлы базы данных	42
3.4.2 Файловые группы	44
3.4.3 Добавление и изменение файлов и файловых групп	46
3.5 Контрольные задания	50
4 Управление доступом	52
4.1 Разрешение дополнительных соединений.....	52
4.1.1 Создание имён входа.....	52
4.1.2 Серверные роли	56
4.2 Управление доступом к данным.....	58
4.2.1 Пользователи базы данных.....	58
4.2.2 Роли базы данных.....	64
4.2.3 Схемы базы данных	65
4.3 Контрольные задание	67
5 Резервное копирование и восстановление баз данных	69
5.1 Модели восстановления	69

5.1.1	<i>Виды моделей восстановления</i>	69
5.1.2	<i>Смена модели восстановления с помощью MS SSMS.....</i>	70
5.1.3	<i>Смена модели восстановления с помощью Transact-SQL.....</i>	73
5.2	<i>Резервное копирование базы данных.....</i>	74
5.2.1	<i>Оценка размера файла резервной копии</i>	74
5.2.2	<i>Виды резервных копий</i>	75
5.2.3	<i>Выполнение резервного копирования средствами MS SSMS</i>	76
5.2.4	<i>Резервное копирование с помощью средств языка Transact-SQL.....</i>	85
5.3	<i>Восстановление баз данных</i>	86
5.3.1	<i>Восстановление с помощью MS SSMS</i>	87
5.3.2	<i>Восстановление с помощью средств языка Transact-SQL</i>	96
5.4	<i>Контрольные задания</i>	97
Список рекомендованных источников		99

ВВЕДЕНИЕ

Современные предприятия среднего и крупного бизнеса, активно используют информационные технологии в деятельности организации. Аппаратные и программные комплексы, упрощающие выполнение каждодневных рутинных задач и расчётов прочно вплелись в каждый процесс, протекающий на предприятиях. В большинстве систем прикладного назначения неотъемлемой частью является база данных, которая обеспечивает надёжное хранение корпоративной информации и обеспечивает доступ к ней при возникновении необходимости. Учитывая широкую распространенность и сферу применения баз данных, базовые знания об их устройстве и особенностях, а также базовые навыки управления пригодятся практически каждому техническому специалисту в области информационных технологий. Каждая база данных требует для работы наличия системы управления (СУБД), которая, при правильной настройке, обеспечит безопасное и надёжное хранение информации организации. Однако, для правильной настройки необходимо иметь хотя бы базовое представление о внутреннем устройстве СУБД.

В данном пособии рассматриваются базовые аспекты администрирования популярной системы управления базами данных Microsoft SQL Server 2019.

Microsoft SQL Server 2019 — универсальная платформа по управлению данными, представляющее собой комплексное решение по организации хранения, предоставления и обработки информации. В настоящее время платформа поддерживает службы машинного обучения и средства аналитики данных. Выпуск полноценной версии ядра СУБД под операционные системы семейства Linux способствует популяризации данного программного продукта.

Пособие состоит из пяти разделов.

В первом разделе даётся краткая характеристика СУБД: информация о доступных выпусках, системных требованиях и службах, входящих в её состав.

Второй раздел посвящен особенностям настройки операционной системы Ubuntu Server 20.04, позволяющей администратору выполнять поставленные перед ним задачи по обеспечению размещения физических файлов по файловой системе.

В третьем разделе рассматривается ядро СУБД, объекты, которые его составляют и основные приёмы работы с ними. Как и в предыдущем разделе, здесь рассматривается работа с файлами, но уже со стороны сервера. Также, даётся описание метаязыка, применяемого в документации Microsoft при описании синтаксических конструкций.

Четвертый раздел посвящён управлению доступом к серверу и хранящемуся на нём объектам. Рассматриваются все уровни модели безопасности SQL Server: сервера, базы данных и схемы. Рассмотрены вопросы распределения прав доступа между пользователями СУБД.

Заключительный, пятый раздел посвящен вопросам организации резервного копирования и восстановления баз данных, что позволяет обеспечить сохранность данных и снизить риски, связанные с повреждением данных или важных объектов сервера.

Таким образом, пособие раскрывает базовые аспекты администрирования сервера баз данных на различных уровнях и может быть использовано при выполнении практических задач по дисциплине «Проектирование и администрирование хранилищ и баз данных» студентами бакалавриата, специальности 09.03.02 «Информационные системы и технологии».

1 ОБЩИЕ СВЕДЕНИЯ О СУБД MS SQL SERVER

1.1 Информация о выпусках

На текущий момент актуальной версией СУБД Microsoft SQL Server является версия MS SQL Server 2019. Компания Microsoft для каждой версии представляет пять различных выпусков СУБД, имеющих различные условия пользовательского соглашения. Большинство выпусков MS SQL Server являются платными, но два выпуска EXPRESS и DEVELOPER могут быть использованы совершенно бесплатно, при условии соблюдения условий лицензионного соглашения. Информация о бесплатных выпусках приведена в таблице ниже.

Примеры, рассмотренные в данном пособии, могут быть выполнены с помощью любого выпуска MS SQL Server, поэтому выбор зависит только от собственных предпочтений, целей и задач. Загрузить нужный выпуск можно с официального сайта компании [1].

Таблица 1 — Информация о бесплатных выпусках MS SQL Server 2019

Выпуск	Характеристика
DEVELOPER	Выпуск даёт возможность создавать любые типы приложений базирующихся на SQL Server. Функционально выпуск эквивалентен коммерческой Enterprise Edition (являющейся наиболее полной версией SQL Server), но условия лицензии не позволяют использовать её в качестве рабочего сервера. Назначение выпуска DEVELOPER — разработка и тестирование приложений баз данных.
EXPRESS	Бесплатный базовый выпуск СУБД, подходящий для применения в обучении или создания приложений баз данных, предназначенных для локальной работы или работы на небольших серверах. Выпуск подходит для непрофессиональных разработчиков, студентов, создающих клиентские приложения. При необходимости выпуск может быть расширен до выпусков более высокого класса. Существует упрощённая версия Express — SQL Server LocalDB, включающая в себя все программные функции. Она работает в пользовательском режиме, не требует настройки и обладает небольшим количеством предварительных требований.

1.2 Минимальные системные требования

Как и любое другое программное обеспечение, СУБД MS SQL Server предъявляет определённые требования к целевой аппаратно-программной платформе.

Аппаратные требования немного отличаются в зависимости от версии и выпуска. В таблице ниже Приводятся аппаратные требования для версии SQL Server 2019:

Компонент	Требование
Жесткий диск	6 Гб
ОЗУ	1 Гб (рекомендуется 4 Гб)
Процессор	x64 1,4 ГГц (Рекомендуется 2,0 ГГц)
Тип процессора	AMD Opteron, AMD Athlon 64, Intel Xeon с поддержкой Intel EM64T, Intel Pentium IV с поддержкой EM64T.
Монитор	Super VGA с разрешением 800x600 пикселей или более высоким
Интернет	При необходимости поддержки интернет-средств MS SQL Server

Примечание: для выпуска Express минимально допустимым объёмом оперативной памяти является 512 Мб, а рекомендуемый 1 Гб.

Кроме требований к аппаратной части, для функционирования СУБД требуется предварительно установить следующее программное обеспечение:

Тип	Требование
Операционная система	Windows 10 TH1 1507 или более поздней версии Windows Server 2016 или более поздней версии
.NET Framework	Версия зависит от операционной системы
Сетевое программное обеспечение	Поддерживаемые операционные системы для SQL Server содержат встроенное сетевое программное обеспечение. Именованные экземпляры и экземпляры по умолчанию изолированной установки поддерживают следующие сетевые протоколы: Shared memory, Named Pipes и TCP/IP.

Примечание: Начиная с MS SQL Server 2017 серверная часть выпускается и для некоторых дистрибутивов Linux, например Ubuntu 18.04 и выше.

Microsoft SQL Server состоит из множества компонентов, каждый из которых выполняет определённые функции и должен быть настроен администратором перед использованием. Далее будут рассмотрены компоненты, которые могут входить в состав СУБД.

1.3 Компоненты сервера

Компоненты, описанные в этом разделе, могут быть выбраны при установке нового экземпляра сервера.

Компонент Database Engine — основной, обязательный к установке компонент, который по существу является экземпляром сервера. Компонент содержит все необходимое для работы с СУБД.

Службы Analysis Services — содержит средства создания приложений оперативной аналитической обработки (OLAP) и приложений интеллектуального анализа данных, а также средства управления ими.

Службы Reporting Services — включают в себя серверные и клиентские компоненты для создания, управления и развертывания табличных, матричных и графических отчетов, а также отчетов в свободной форме.

Службы Integration Services — Службы Integration Services представляют

собой набор графических средств и программируемых объектов для перемещения, копирования и преобразования данных.

Службы Master Data Services (MDS) — это решение SQL Server по управлению основными данными. MDS можно настроить для управления любой структурой (товары, заказчики, счета). Поддерживаются иерархии, детальная настройка безопасности, транзакции, управление версиями данных и бизнес-правила, а также использование Надстройка для Excel для управления данными.

Служба машинного обучения (в базе данных) — Службы машинного обучения (в базе данных) поддерживают распределенные и масштабируемые решения машинного обучения, использующие корпоративные источники данных. В SQL Server 2016 поддерживался язык R. SQL Server 2019 (15.x) поддерживает язык R и Python.

Сервер машинного обучения (автономный) поддерживает развертывание распределенных масштабируемых решений машинного обучения на множестве платформ и использование разных корпоративных источников данных, включая Linux и Hadoop.

Для выполнения практических заданий, приведённых в данном приложении необходимым и достаточным компонентом, является только Database Engine, все остальные службы и компоненты могут быть установлены по желанию. Стоит отметить, что большинство компонентов не работают в выпусках Express.

2 АДМИНИСТРИРОВАНИЕ СЕРВЕРА НА УРОВНЕ ОПЕРАЦИОННОЙ СИСТЕМЫ

Учитывая возрастающую популярность open source решений, для организации практических работ по администрированию баз данных была подготовлена виртуальная машина, файл конфигурации и образ жесткого диска можно получить по ссылке, зашифрованной в QR-коде (рис. 2.1).



Рисунок 2.1 — Ссылка на файлы виртуальной машины

Для запуска виртуальной машины требуется установить гипервизор VMWare или Oracle VirtualBox. Использование виртуальной машины позволит без рисков повреждения системы производить манипуляции с файлами на жестких дисках, а также познакомится с моделью безопасности ОС Ubuntu Server, в части распределения прав доступа к файловой системе между пользователями.

2.1.1 Развертывание виртуальной машины

Запустите гипервизор (в примере используется Oracle Virtual Box, для операционной системы macOS).

В главном меню гипервизора необходимо выбрать пункт «Импорт конфигурации...» (рис. 2.2).

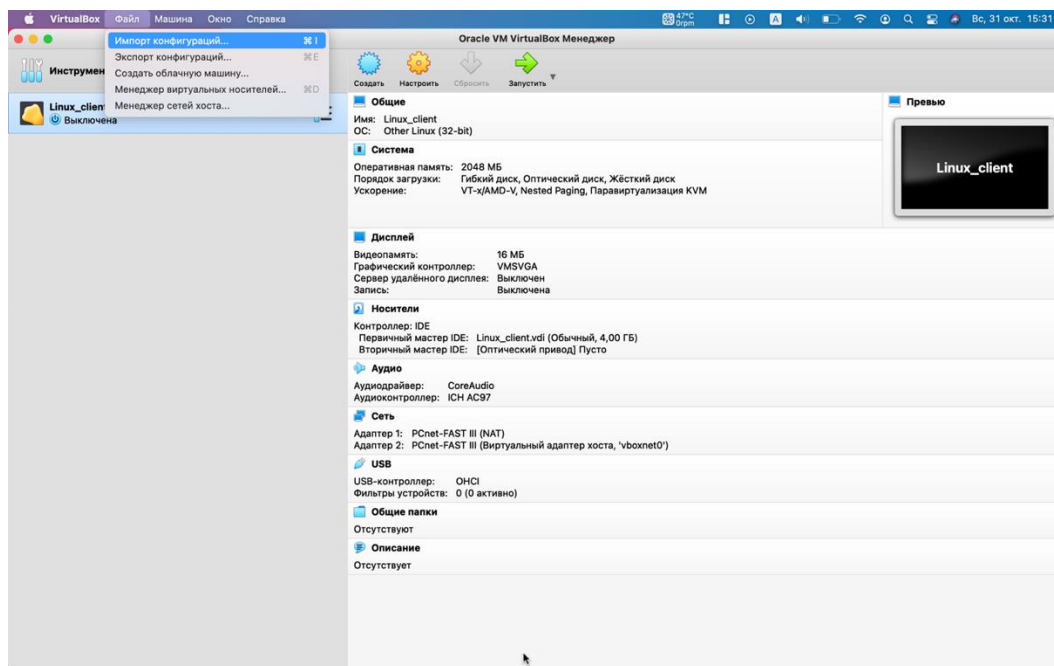


Рисунок 2.2 — Импорт конфигурации виртуальной машины

В открывшемся диалоговом окне необходимо выбрать источник, в котором храниться файл с конфигурацией виртуальной машины и указать путь к файлу с расширением **.ova** (**.ovf**). (рис. 2.3). После указания пути к файлу конфигурации, станет доступна кнопка **«Продолжить»**, на которую следует нажать, чтобы перейти к проверке параметров импортируемой конфигурации.

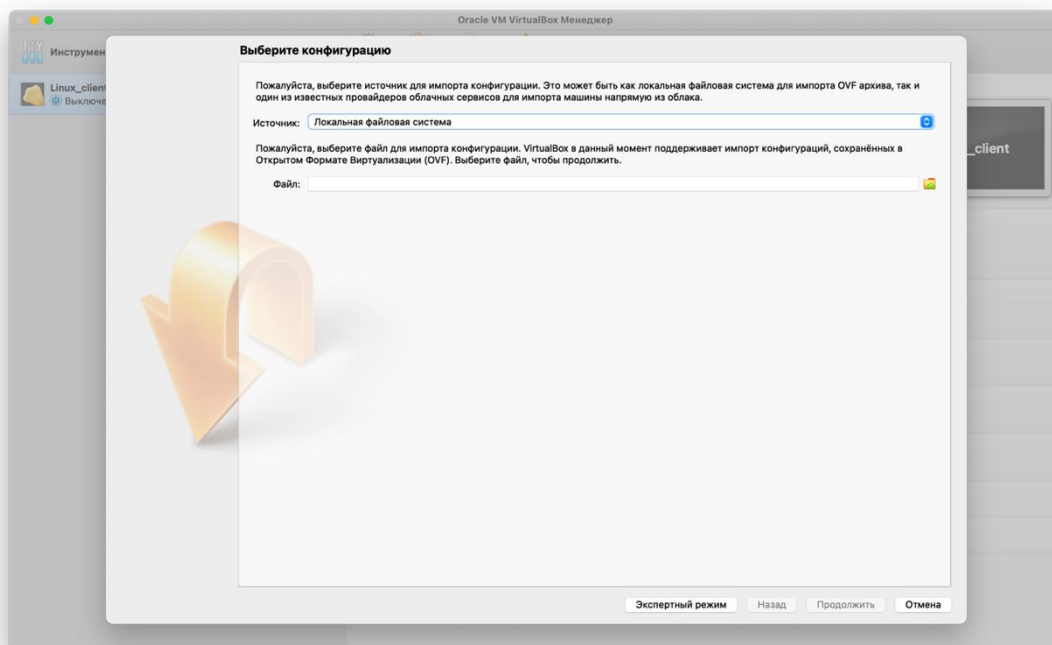


Рисунок 2.3 — Выбор импортируемой конфигурации

Далее необходимо проверить и, при необходимости, изменить параметры импорта (рис. 2.4):

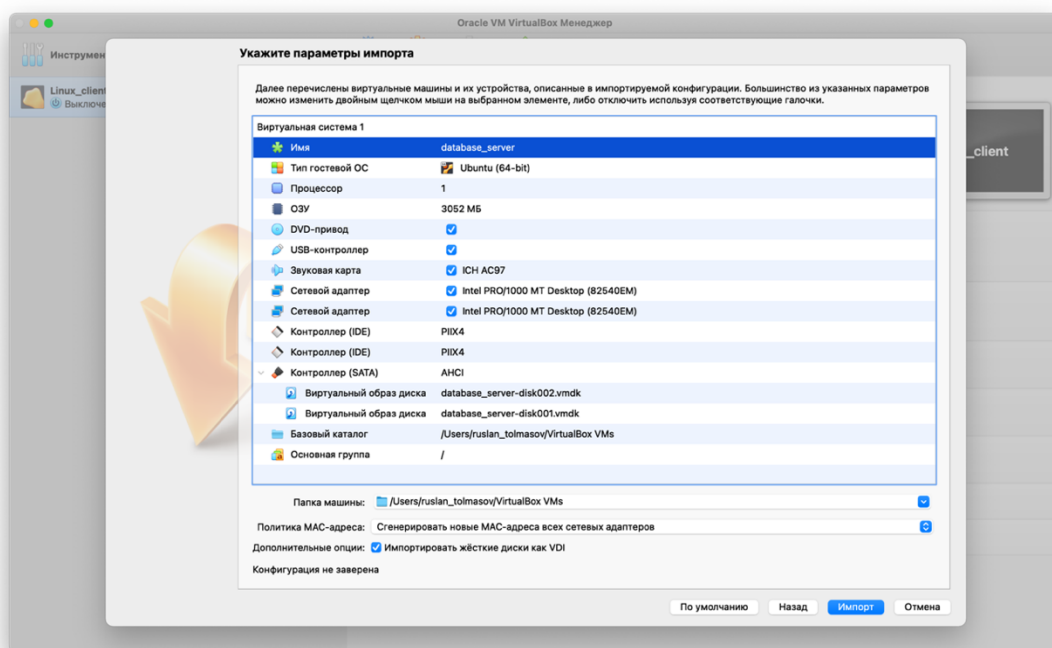


Рисунок 2.4 — Настройка параметров импорта

Обязательно нужно сгенерировать новые *MAC*-адреса устройств, остальные параметры можно настроить исходя из личных предпочтений. После проверки и изменения настроек, необходимо нажать на кнопку «**Импорт**».

Процедура импорта начнётся незамедлительно, дождитесь окончания импорта (рис.2.5).

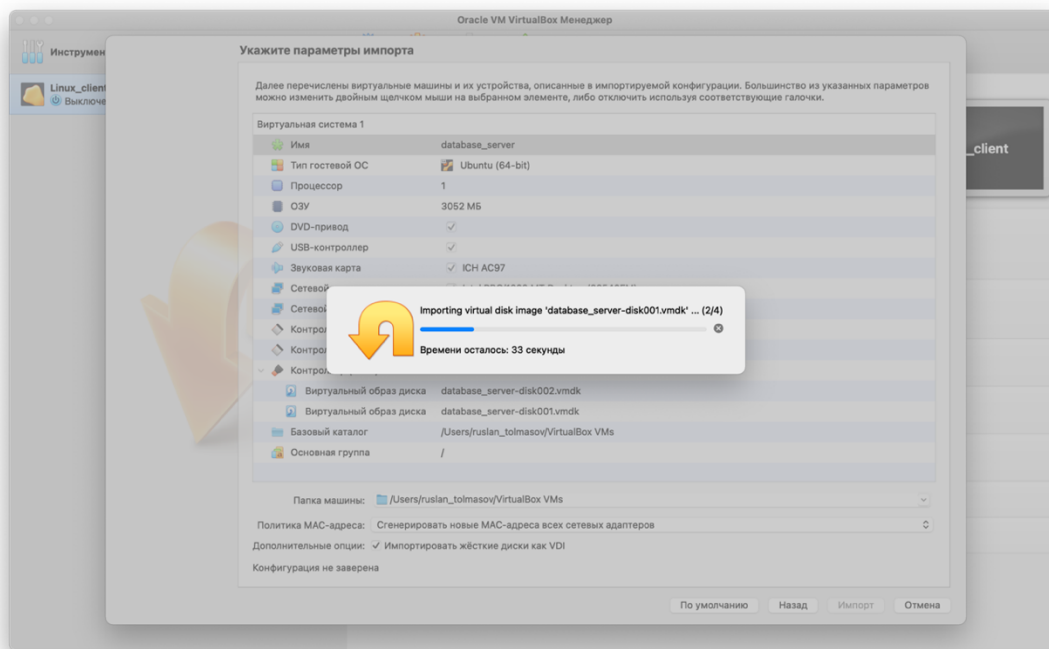


Рисунок 2.5 — Процесс импорта

После завершения импорта, импортированная конфигурация появится в списке виртуальных машин гипервизора.

На импортированной виртуальной машине установлены и доступны для использования 3 СУБД (в скобках указаны названия соответствующих процессов):

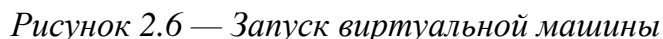
- MS SQL Server 2019 Express (mssql-server);
- MongoDB (mongodb);
- Postgre SQL (postgresql).

Для реляционных СУБД *MS SQL Server* и *PostgesSQL* установлены учебные базы данных: *AdventureWorks2019* и *AdventureWorksDW2019* для *MS SQL Server* и одна учебная база данных средних размеров для *PostgreSQL*.

2.1.2 Управление запуском и остановкой сервера

Управление запуском и остановкой имеющихся СУБД осуществляется с помощью утилиты управления службами (демонами) *systemctl*. Практически все действия в системе выполняются от имени суперпользователя или в режиме суперпользователя (что крайне не рекомендуется). Для начала запустим виртуальную машину и проверим состояние установленных СУБД используя утилиту

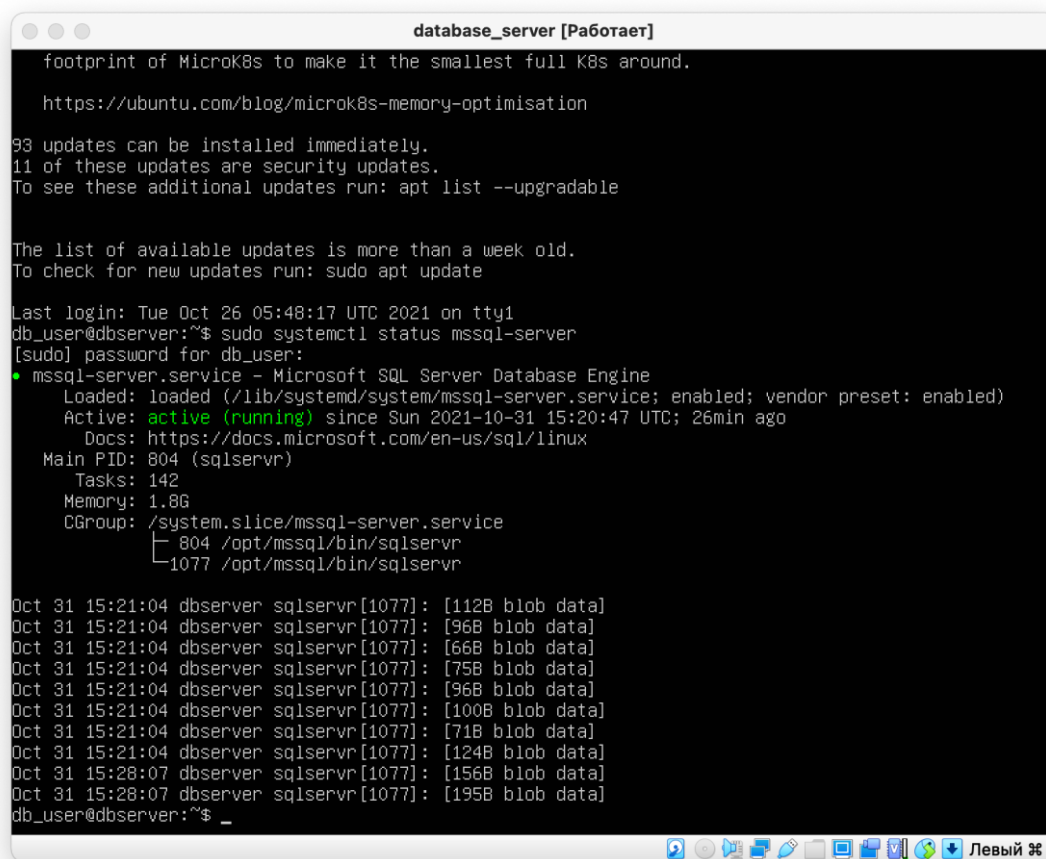
Для запуска виртуальной машины необходимо выбрать нужную виртуальную машину из списка доступных машин и нажать кнопку «*Запустить*», расположенную на панели инструментов (Рис. 2.6).



логин: db user; пароль:¹ dbStudy;

```
sudo systemctl status mssql-server
```

² При выполнении команды может потребоваться ввести пароль



```
database_server [Пабогае]
footprint of MicroK8s to make it the smallest full K8s around.
https://ubuntu.com/blog/microk8s-memory-optimisation

93 updates can be installed immediately.
11 of these updates are security updates.
To see these additional updates run: apt list --upgradable

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Tue Oct 26 05:48:17 UTC 2021 on tty1
db_user@dbserver:~$ sudo systemctl status mssql-server
[sudo] password for db_user:
• mssql-server.service - Microsoft SQL Server Database Engine
   Loaded: loaded (/lib/systemd/system/mssql-server.service; enabled; vendor preset: enabled)
   Active: active (running) since Sun 2021-10-31 15:20:47 UTC; 26min ago
     Docs: https://docs.microsoft.com/en-us/sql/linux
   Main PID: 804 (sqlservr)
      Tasks: 142
     Memory: 1.8G
    CGroup: /system.slice/mssql-server.service
             └─ 804 /opt/mssql/bin/sqlservr
                1077 /opt/mssql/bin/sqlservr

Oct 31 15:21:04 dbserver sqlservr[1077]: [112B blob data]
Oct 31 15:21:04 dbserver sqlservr[1077]: [96B blob data]
Oct 31 15:21:04 dbserver sqlservr[1077]: [66B blob data]
Oct 31 15:21:04 dbserver sqlservr[1077]: [75B blob data]
Oct 31 15:21:04 dbserver sqlservr[1077]: [96B blob data]
Oct 31 15:21:04 dbserver sqlservr[1077]: [100B blob data]
Oct 31 15:21:04 dbserver sqlservr[1077]: [71B blob data]
Oct 31 15:21:04 dbserver sqlservr[1077]: [124B blob data]
Oct 31 15:28:07 dbserver sqlservr[1077]: [156B blob data]
Oct 31 15:28:07 dbserver sqlservr[1077]: [195B blob data]
db_user@dbserver:~$
```

Рисунок 2.7 — Вывод утилиты *systemctl*

Зелёная надпись «**active (running)**» сообщает пользователям факт того, что сервер запущен. Для проверки состояния других СУБД необходимо выполнить ту же команду, но изменить имя службы, на необходимую в данный момент.

Для остановки сервера необходимо выполнить следующую команду:

```
sudo systemctl stop mssql-server
```

После выполнения команды, если в процессе не возникло ошибок, в консоль не будет передано никаких данных. Проверив состояние *mssql-server* можно убедиться, что сервер действительно больше не активен. Для ручного запуска сервера потребуется заменить слово «stop» на слово «start»:

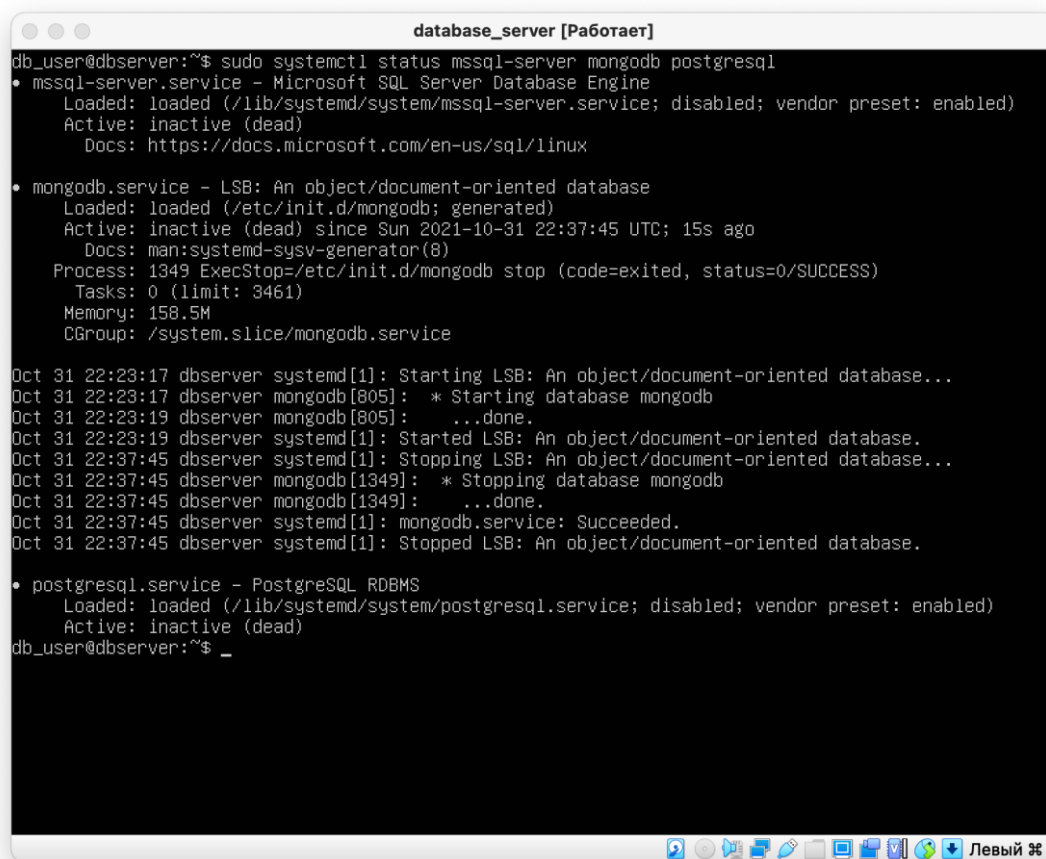
```
sudo systemctl start mssql-server
```

Таким образом можно вручную управлять состоянием серверов, установленных на данной виртуальной машине. Тем не менее, при перезапуске виртуальной машины все три сервера вновь станут активными. Для того, чтобы после перезагрузки виртуальной машины активными оставались только нужные сервера, необходимо настроить их автозапуск. Это возможно сделать применяя все ту же системную утилиту *systemctl* указав вместо слова «start» слово «enabled» для автоматического запуска сервера и «disable» для запрета автоматической

загрузки. Следующие инструкции установят автоматическую загрузку для mssql-server и отключат её для двух других СУБД:

```
sudo systemctl enabled mssql-server
sudo systemctl disabled mongodb3 postgresql
```

И после перезагрузки проверяя статус трех серверов будет получен вывод как представлен на рис. 2.8



```
db_user@dbserver:~$ sudo systemctl status mssql-server mongodb postgresql
• mssql-server.service - Microsoft SQL Server Database Engine
   Loaded: loaded (/lib/systemd/system/mssql-server.service; disabled; vendor preset: enabled)
   Active: inactive (dead)
     Docs: https://docs.microsoft.com/en-us/sql/linux

• mongodb.service - LSB: An object/document-oriented database
   Loaded: loaded (/etc/init.d/mongodb; generated)
   Active: inactive (dead) since Sun 2021-10-31 22:37:45 UTC; 15s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 1349 ExecStop=/etc/init.d/mongodb stop (code=exited, status=0/SUCCESS)
    Tasks: 0 (limit: 3461)
   Memory: 158.5M
    CGroup: /system.slice/mongodb.service

Oct 31 22:23:17 dbserver systemd[1]: Starting LSB: An object/document-oriented database...
Oct 31 22:23:17 dbserver mongodb[805]: * Starting database mongodb
Oct 31 22:23:19 dbserver mongodb[805]: ...done.
Oct 31 22:23:19 dbserver systemd[1]: Started LSB: An object/document-oriented database.
Oct 31 22:37:45 dbserver systemd[1]: Stopping LSB: An object/document-oriented database...
Oct 31 22:37:45 dbserver mongodb[1349]: * Stopping database mongodb
Oct 31 22:37:45 dbserver mongodb[1349]: ...done.
Oct 31 22:37:45 dbserver systemd[1]: mongodb.service: Succeeded.
Oct 31 22:37:45 dbserver systemd[1]: Stopped LSB: An object/document-oriented database.

• postgresql.service - PostgreSQL RDBMS
   Loaded: loaded (/lib/systemd/system/postgresql.service; disabled; vendor preset: enabled)
   Active: inactive (dead)
db_user@dbserver:~$ _
```

Рисунок 2.8 — Пример вывода проверки статуса при отключенных серверах СУБД

Далее мы познакомимся с особенностями организации доступа к файлам в операционной системе Ubuntu Server 20.04.

2.1.3 Предоставление прав доступа к файловой системе

В системах семейства Linux возможно настроить права доступа к папкам или отдельным файлам пользователю или группе пользователей. Подробно это рассматривается в соответствующих руководствах и дисциплинах. С точки зрения администрирования баз данных, нам необходимо научиться создавать папки

³ С отключением автозапуска MongoDB могут возникнуть трудности. СУБД можно отключить вручную и исключить из списка автозагрузки, но высока вероятность, что автозапуск mongodb.service может управляться другими утилитами.

и предоставлять к ним доступ приложению, а также перечень прав, которые могут быть назначены

Операционная система может предоставлять следующие права на файлы и папки каждому пользователю в отдельности или целой группе пользователей в любой комбинации:

- чтение;
- запись;
- выполнение;
- разрешение SUID;
- разрешение SGID;
- sticky bit.

Кроме того, у каждого файла или папки должен быть владелец. По умолчанию владельцем назначается пользователь создавший объект, но в процессе работы часто бывает необходимо сменить владельца объекта. Примером подобной ситуации при работе с СУБД Microsoft SQL Server может служить необходимость размещения файлов пользовательских баз данных в отличном от стандартного месторасположения или смены папки для хранения резервных копий.

Предположим, что некоторые пользовательские базы данных требуется хранить отдельно от остальных. Для решения этой задачи администратор системы создаёт папку доступную по следующему пути:

```
/home/db_user/mydb/
```

Для этого администратор создаёт новый каталог с помощью ввода в консоль следующей инструкции:

```
$ sudo mkdir /home/db_user/mydb
```

В результате, по указанному пути будет создан каталог. Если создать в нём файл или подкаталог с помощью консоли, то система позволит это сделать, но если попытаться создать базу данных в этом расположении используя утилиту `sqlcmd` или любое клиентское приложение, то система укажет на отсутствие необходимых разрешений. Проблема вызвана тем, что попытку создания базы данных осуществляет группа `mssql`, которая не является владельцем созданного ранее каталога и не имеет никаких прав на его содержимое. Для решения данной проблемы необходимо назначить группу `mssql` владельцем данного каталога выполнив следующие инструкции в консоли:

```
sudo chown mssql /home/db_user/mydb/*  
sudo chgrp mssql /home/db_user/mydb/*
```

Таким образом, мы можем создавать новые места размещения для объектов,

создаваемых сервером СУБД. Более подробно о распределении прав на файлы и каталоги можно в технической документации к операционной системе Ubuntu Server 20.04 [2] или на специализированных сайтах, например losst.ru [3]

2.1.4 Добавление новых жестких дисков

В процессе эксплуатации может возникнуть необходимость добавлять новые жесткие диски в сервер, которые будут использоваться, например, для размещения файлов баз данных. Для того чтобы обеспечить возможность использования новых жестких дисков в операционной системе их необходимо правильно отформатировать и смонтировать, при необходимости добавить автоматическое монтирование при запуске операционной системы.

Рассмотрим процесс добавления нового диска на примере используемой нами виртуальной машины. Для начала добавим новый образ жесткого диска к виртуальной машине. Для этого необходимо перейти в настройки виртуальной машины и выбрать раздел «**Носители**» и нажать на кнопку (2) (рис. 2.9).

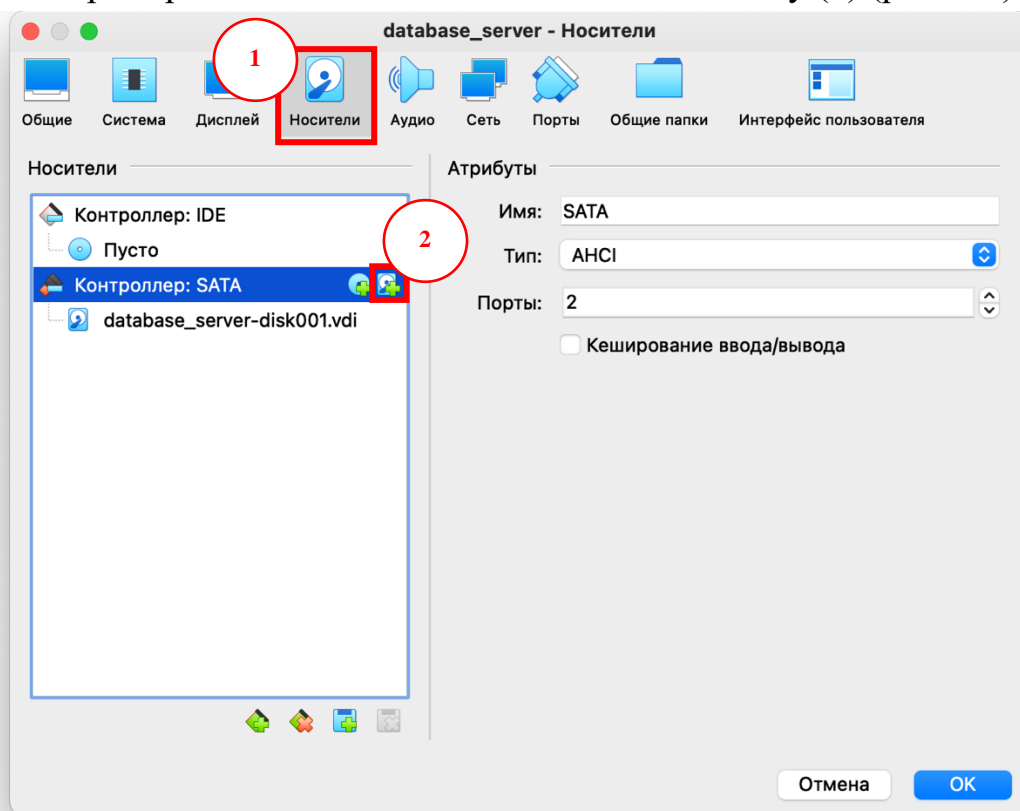


Рисунок 2.9 — Меню настроек "Носители"

В открывшемся окне необходимо выбрать существующий образ диска или создать новый (рис. 2.10). После добавления, закройте окно настроек (рис 2.11)

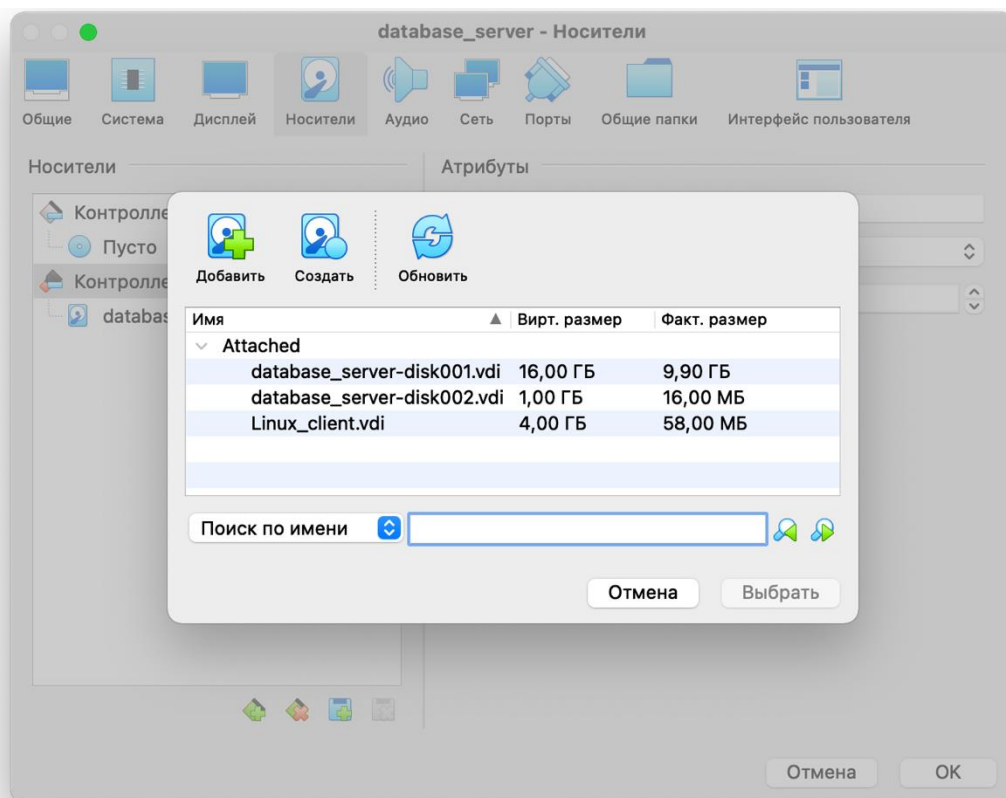


Рисунок 2.10 — Диалоговое окно добавления нового диска

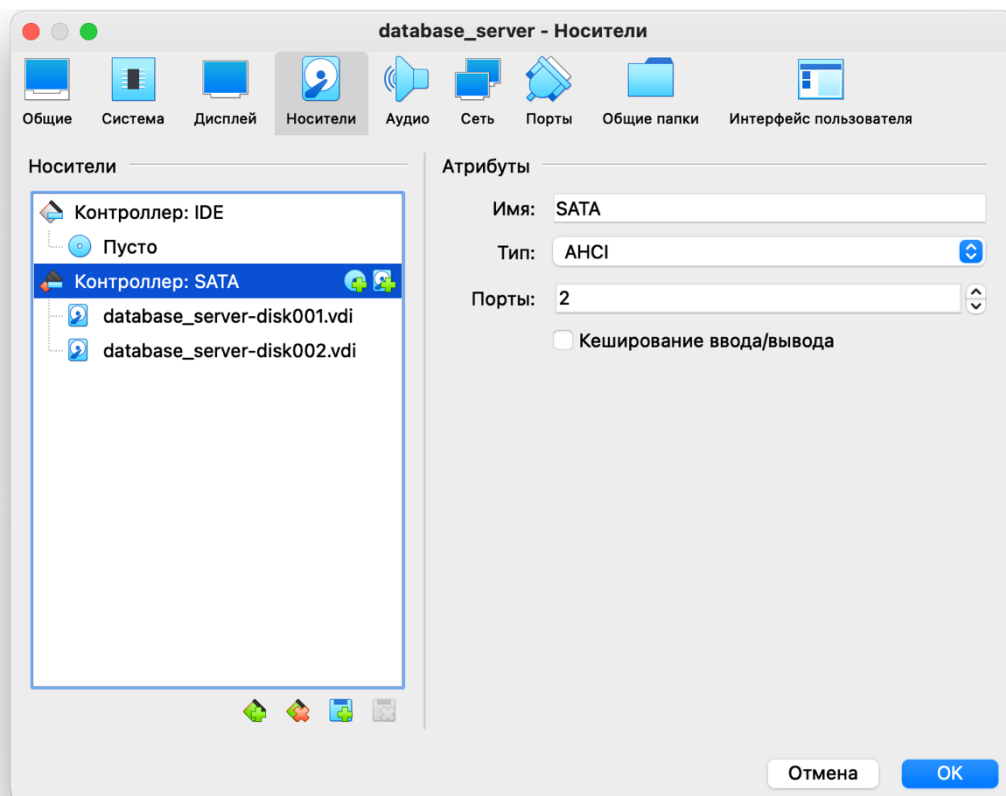


Рисунок 2.11 — Результат добавления нового диска

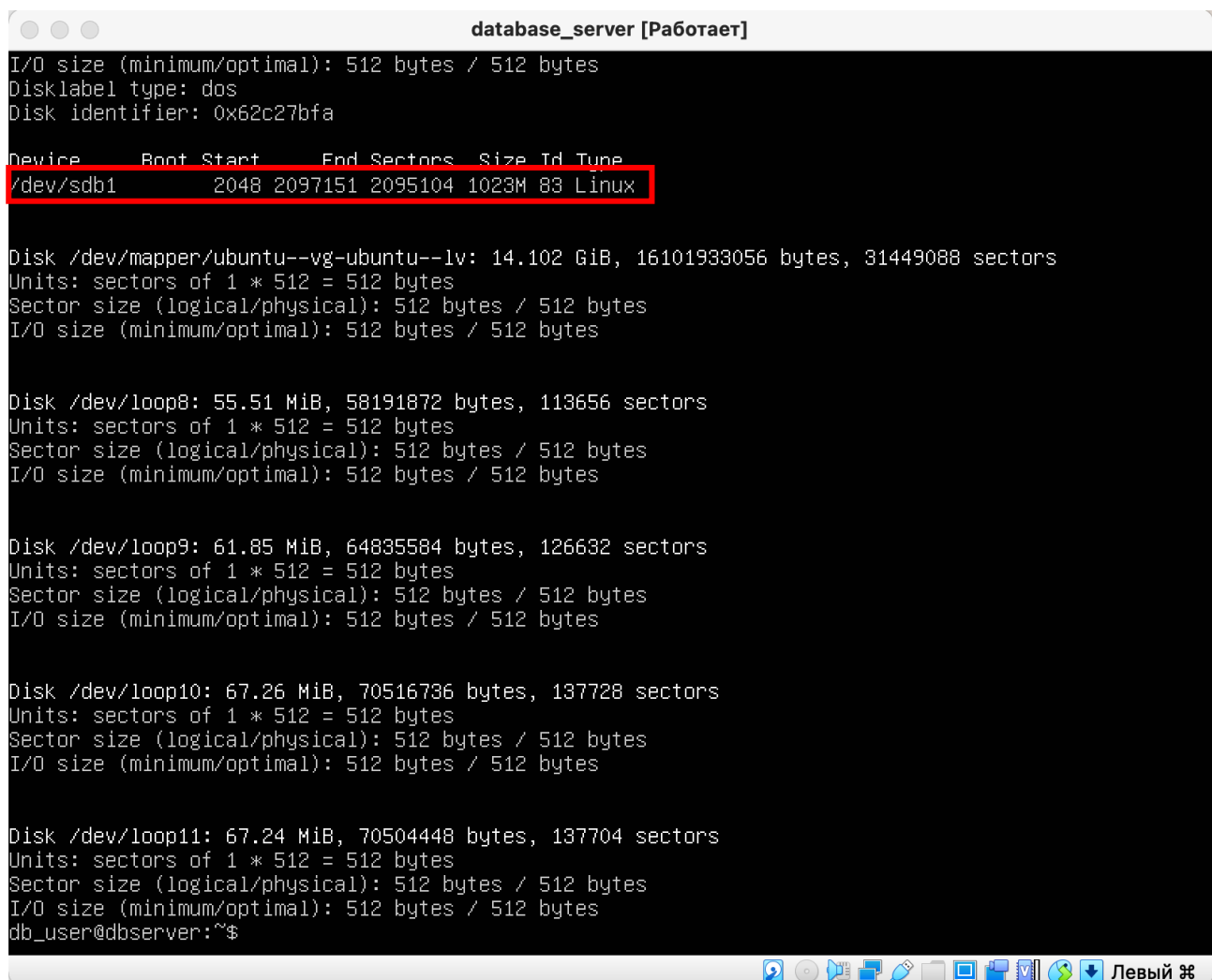
Новый диск успешно подключён к виртуальной машине. Запустите

виртуальную машину и дождитесь загрузки операционной системы. Напомним, что работать можно как из консоли виртуальной машины, так и с помощью внешних терминалов, подключаясь по протоколу SSH (для этого требуется знать IP-адрес машины в локальной сети).

Авторизуйтесь на сервере и далее воспользуйтесь утилитой просмотра и управления дисками и разделами в системах *Linux* *fdisk*, для этого введите в командной строке:

```
sudo fdisk -l
```

Вывод утилиты представлен на рис. 2.12



```
database_server [Работает]
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x62c27bfa

Device            Boot  Start      End  Sectors  Size Id Type
/dev/sdb1          2048 2097151 2095104 1023M 83 Linux

Disk /dev/mapper/ubuntu--vg-ubuntu--lv: 14.102 GiB, 16101933056 bytes, 31449088 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/loop8: 55.51 MiB, 58191872 bytes, 113656 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/loop9: 61.85 MiB, 64835584 bytes, 126632 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/loop10: 67.26 MiB, 70516736 bytes, 137728 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/loop11: 67.24 MiB, 70504448 bytes, 137704 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
db_user@dbserver:~$
```

Рисунок 2.12 — Результат работы утилиты *fdisk*

Добавленный диск будет отображаться как */dev/sdb#* (где # — порядковый номер устройства). Перед началом использования подключённый диск нужно разбить на разделы и отформатировать, для этого воспользуемся функционалом утилиты *fdisk* и выполним следующую команду:

```
sudo fdisk /dev/sdb1
```

В результате будет активирована программа форматирования диска и в

консоль будет выведено сообщение приведённое на рис. 2.13.



```
Changes will remain in memory only, until you decide to write them.  
Be careful before using the write command.  
  
The old ext4 signature will be removed by a write command.  
  
Device does not contain a recognized partition table.  
Created a new DOS disklabel with disk identifier 0xd95424da.  
  
Command (m for help): _
```

Рисунок 2.13 — Форматирование жесткого диска

Команды данной программы представлены прописными буквами латинского алфавита. Подробно все команды в данном пособии рассматриваться не будут, ознакомиться с ними можно в документации (введя *m* в качестве команды).

Для создания нового раздела вводим *n*. Далее предлагается выбрать тип разделов (первичный или расширенный). Выбираем предлагаемое по умолчанию значение (*p*). Номер раздела тоже устанавливаем по умолчанию (1), так как это новый диск. Если первичный раздел будет добавляться дополнительный раздел к существующему, то значение нужно будет увеличить на 1 (максимум поддерживается 4 первичных разделов). Далее выбирается размер первого сектора, можно оставить по умолчанию. Далее следует указать размер последнего сектора или размер раздела (рекомендуется указывать точное значение с префиксом +). Далее можно сохранить изменения и выйти из программы (*w*) (Рис. 2.14).

```
database_server [Работает]
db_user@dbserver:~$ sudo fdisk /dev/sdb1

Welcome to fdisk (util-linux 2.34).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

The old ext4 signature will be removed by a write command.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0xd95424da.

Command (m for help): n
Partition type
  p   primary (0 primary, 0 extended, 4 free)
  e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-2095103, default 2048): 2048
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-2095103, default 2095103): +2095103
Value out of range.
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-2095103, default 2095103): +1022M
Value out of range.
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-2095103, default 2095103): +1000M

Created a new partition 1 of type 'Linux' and of size 1000 MiB.

Command (m for help): w_
```

Рисунок 2.14 — Процесс создания раздела диска

Следующим шагом необходимо отформатировать диск, для чего следует воспользоваться утилитой `mkfs`. Введите и выполните следующую команду:

```
sudo mkfs.ext4 /dev/sdb1
```

После выполнения разметки и форматирования диска его можно приступить к его монтированию. Создадим каталог, который будет выступать точкой доступа к диску:

```
sudo mkdir mydb
```

Теперь смонтируем диск в созданную папку

```
sudo mount /dev/sdb1 /mydb
```

Теперь диск смонтирован в раздел `/data` но данную операцию придётся повторять при каждом перезапуске системы. Чтобы система автоматически выполняла монтирование диска, необходимо открыть текстовым редактором следующий файл:

```
sudo vi /etc/fstab
```

и добавить в него запись:

```
/dev/sdb1 /mydb ext4 defaults 0 0
```

Таким образом, мы можем подключать новые физические диски к системе

при необходимости. Более подробно данная тема рассматривается в соответствующих разделах документации операционной системы и на специализированных интернет-ресурсах.

Рассмотренных в данном разделе материалов хватит для выполнения базового администрирования операционной системы Ubuntu Server 20.04 для обеспечения возможности переноса файлов баз данных и размещения их на разных жестких дисках.

2.2 Контрольные задания

1. Скачайте и разверните на своём компьютере образ виртуальной машины.
2. Подключитесь к виртуальной машине и проверьте состояние сетевых интерфейсов.
3. Подключитесь к виртуальной машине через SSH.
4. Проверьте состояние служб с помощью утилиты `systemctl`. Отключите неиспользуемые СУБД и настройте автозапуск нужной СУБД. Перезагрузите машину и проверьте результат.
5. Создайте папку `mssqldb`. Проверьте владельца папки и права, которыми он обладает.
6. Смените владельца папки на `mssql` и предоставьте ему полный доступ к её содержимому.
7. Выключите виртуальную машину и добавьте новый виртуальный жесткий диск объёмом 1 Гб.
8. Выполните разбиение созданного диска на разделы. Отформатируйте его в формат `ext4`.
9. Смонтируйте диск в операционной системе и создайте папки под хранение резервных копий и файлов пользовательских баз данных.
10. Назначьте владельца папки и предоставьте ему соответствующие права. После выполнения этих действий, любой пользователь СУБД, обладающий необходимыми разрешениями на уровне сервера базы данных, должен иметь возможность создавать и изменять файлы в этих папках.

3 ЯДРО СУБД DATABASE ENGINE

3.1 Подключение к Database Engine

3.1.1 Программное обеспечение для работы с экземпляром сервера

Для подключения к серверу необходимо наличие установленного соответствующего клиентского программного обеспечения. Обычно вместе с экземпляром сервера производится установка инструментов командной строки, которых достаточно для работы с сервером, но они значительно уступают в удобстве работы специализированным клиентским программам. Рассмотрим наиболее популярные инструменты для работы с экземпляром сервера *MS SQL Server*.

SQL Server Management Studio (MS SSMS) — это бесплатная интегрированная среда для управления инфраструктурой *MS SQL Server*, а также разработки различных сценариев на языке T-SQL.



Рисунок 3.1 — Логотип SQL Server Management Studio

SSMS является полнофункциональным инструментом, являющимся стандартным, для работы с *Microsoft SQL Server* используемое как профессиональными разработчиками, так и администраторами *SQL Server*.

Таблица 2 — Минимальные системные требования SSMS

Требование	Значение
Операционная система	Windows Server 2022 (64-разрядная версия) Windows 11 (64-разрядная) Windows 10 (64-разрядная) как минимум версии 1607 (10.0.14393) Windows 8.1 (64-разрядная) Windows Server 2019 (64-разрядная версия) Windows Server 2016 (64-разрядная версия) Windows Server 2012 R2 (64-разрядная версия) Windows Server 2012 (64-разрядная версия) Windows Server 2008 R2 (64-разрядная версия)
Процессор	x86 с частотой 1,8 ГГц (рекомендуется использовать минимум двухъядерный процессор)
ОЗУ	2 Гб (минимум 2,5 Гб при использовании виртуальной машины), рекомендуется 4 Гб ОЗУ
HDD	10 Гб свободного места на диске

Для пользователей, которым не требуется богатый функционал Management Studio или использующие другие операционные системы, следует использовать альтернативные клиенты, один из которых также разработан компанией Microsoft.

Azure Data Studio — легкий, кроссплатформенный клиент для работы с *MS SQL Server*, а так же имеется возможность подключаться к другим СУБД. Клиент предназначен в первую очередь для разработчиков приложений и обладает широким функционалом по работе с кодом: автоматическое дополнение команд (технология *IntelliSense*), автоматическая генерация фрагментов часто используемых инструкций и форматирование файла скрипта. К преимуществам данного клиента также относится возможность графического отображения результатов запросов, а также их экспорт в различные форматы (*XML*, *CSV*, *JSON*). Встроенный терминал позволяет выполнять большинство задач по администрированию БД с помощью *SQLCMD* или *PowerShell*, но выполнение тонкой настройки параметров сервера может быть недоступна.



Рисунок 3.2 — Логотип Azure Data Studio

Функционала *Azure Data Studio* достаточно для знакомства с основами проектирования и администрирования хранилищ и баз данных и может быть использовано в дальнейшей профессиональной деятельности, связанной с разработкой SQL-скриптов. Кроме того, *Azure Data Studio* является отличной альтернативой пользователям операционных систем *Linux* и *macOS*.

DBeaver — свободно распространяемое ПО для разработчиков, администраторов баз данных, аналитиков и других людей, использующих базы данных в своей работе. Клиент поддерживает подключение ко всем популярным реляционным СУБД: *MS SQL Server*, *MySQL*, *PostgreSQL*, *Oracle* и многим другим.



Рисунок 3.3 — Логотип DBeaver

Программное средство обладает низкими системными требованиями, поэтому при определении возможности его установки следует убедиться, что поддерживается операционная система, используемая на целевом компьютере.

Программное средство обладает широким функционалом, например с его помощью можно извлекать и отображать диаграммы баз данных, что упрощает процессы перепроектирования баз данных. Подсветка синтаксиса и автодополнение команд в проекте тоже реализовано, но работает не так быстро и точно, как в том же Azure Data Studio. В итоге программное средство пользуется большой популярностью у разработчиков и администраторов баз данных. У него практически нет конкурентов, если требуется поддерживать базы данных размещенных в СУБД разных вендоров.

Однако, выполнение практических заданий в рамках изучения дисциплины возможно с использованием любого клиента, но в случае DBeaver могут возникнуть проблемы с визуальным отображением плана запросов.

3.1.2 Подключение к экземпляру сервера

В зависимости от выбранного клиента последовательность действий при подключении будет немного отличаться. Рассмотрим процесс подключения к серверу с помощью двух основных клиентских приложений, рекомендованных Microsoft для работы с СУБД Microsoft SQL Server.

Подключение с помощью Microsoft SQL Server Management Studio

При запуске MS SQL Server Management Studio, пользователю будет предложено авторизоваться на одном из локальных или сетевых серверов ядра СУБД MS SQL Server или сервере другого типа, поддерживаемых клиентом (Рисунок 3.6). Имя сервера можно ввести вручную в строке «Имя сервера» или выбрать подходящий сервер из списка выбора.

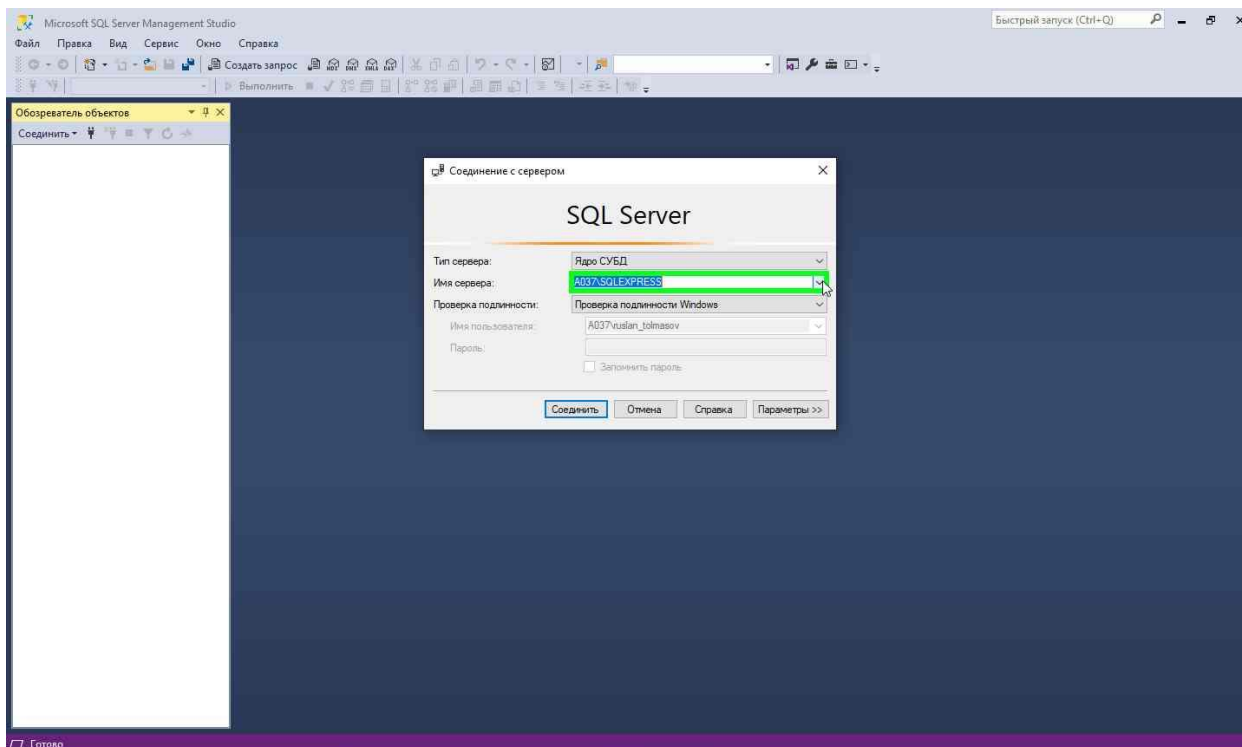


Рисунок 3.4 — Окно "Соединение с сервером"

Если имя сервера ранее не использовалось, то оно не будет отображаться в списке выбора и его необходимо добавить, нажав на пункт «Продолжить обзор...» (Рисунок 3.5)

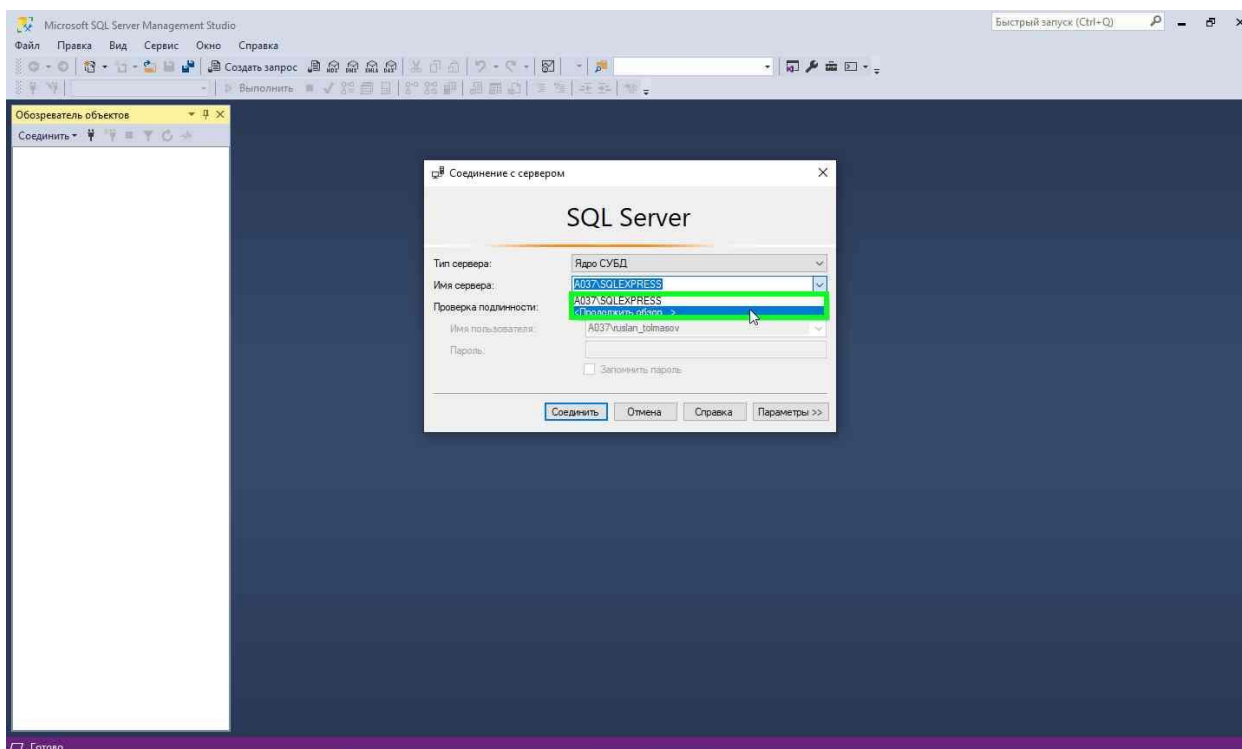


Рисунок 3.5 — Выбор сервера для подключения

В диалоговом окне необходимо выбрать тип сервера и раскрыв связанный список нажатием на «+» слева от названия типа, выбрать нужный экземпляр

локального сервера (Рисунок 3.6). При необходимости подключения к удалённому серверу, нужно перейти на соответствующую вкладку и заполнить параметры подключения (адрес сервера и порт).

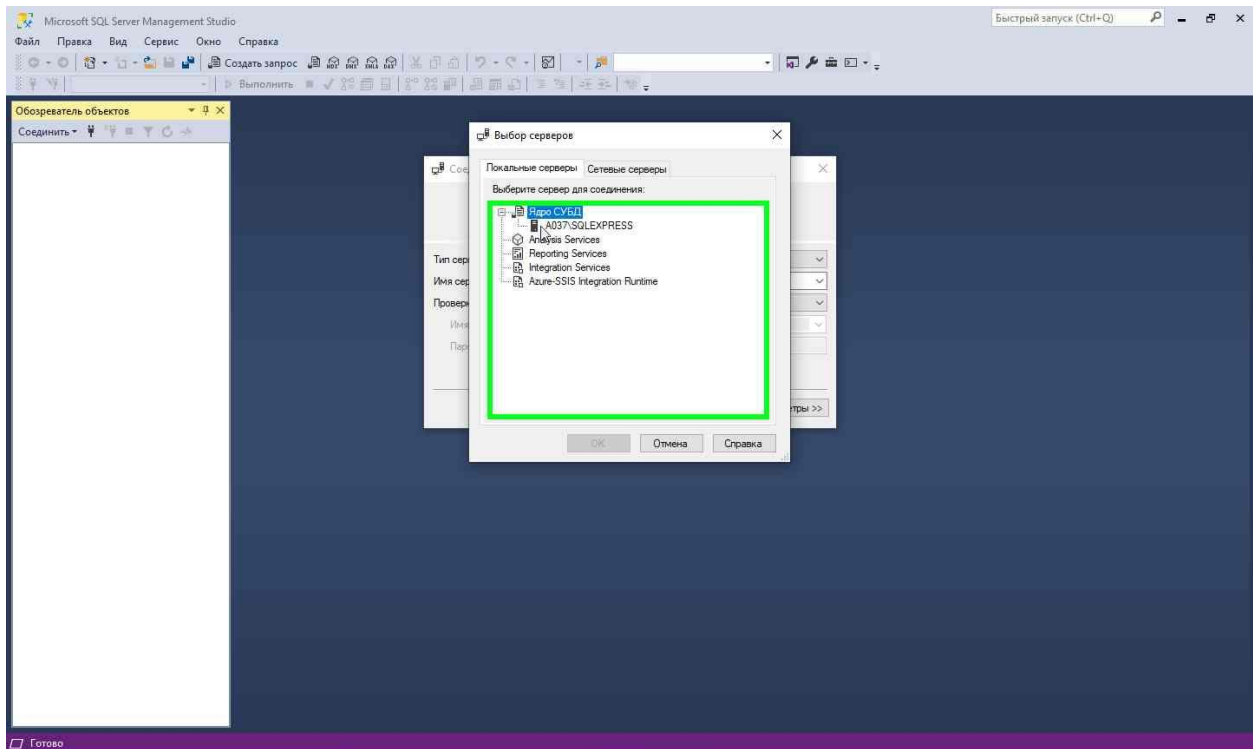


Рисунок 3.6 — Окно выбора сервера

Выбрав нужный экземпляр, необходимо нажать на кнопку «Соединить» (Рисунок 3.7)

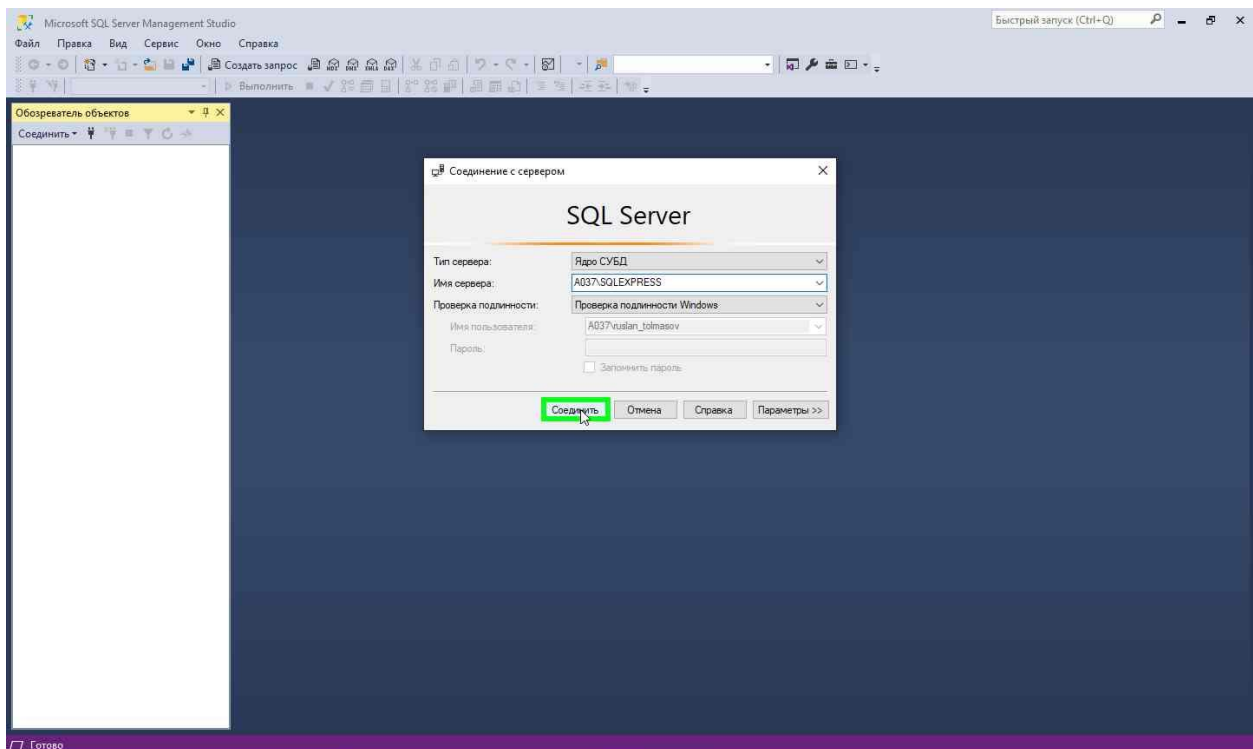


Рисунок 3.7 — Установление подключения

Логин и пароль вводить не нужно, если установлен тип аутентификации «Проверка подлинности Windows». В большинстве случаев соединения с локальным сервером этого достаточно, так как авторизация пользователя проводится благодаря учётной записи Windows. Если системный администратор предоставил логин и пароль, необходимо предварительно выбрать подходящий способ проверки пользователя и ввести логин и пароль.

Подключение с помощью Azure Data Studio

Подключение к серверу с помощью *Azure Data Studio* отличается от подключения к серверу с помощью SSMS.

При первом запуске *Azure Data Studio* автоматически будет открыта приветственная страница. Для создания нового подключения на данной странице необходимо выбрать пункт «*New connection*» (Рисунок 3.8)

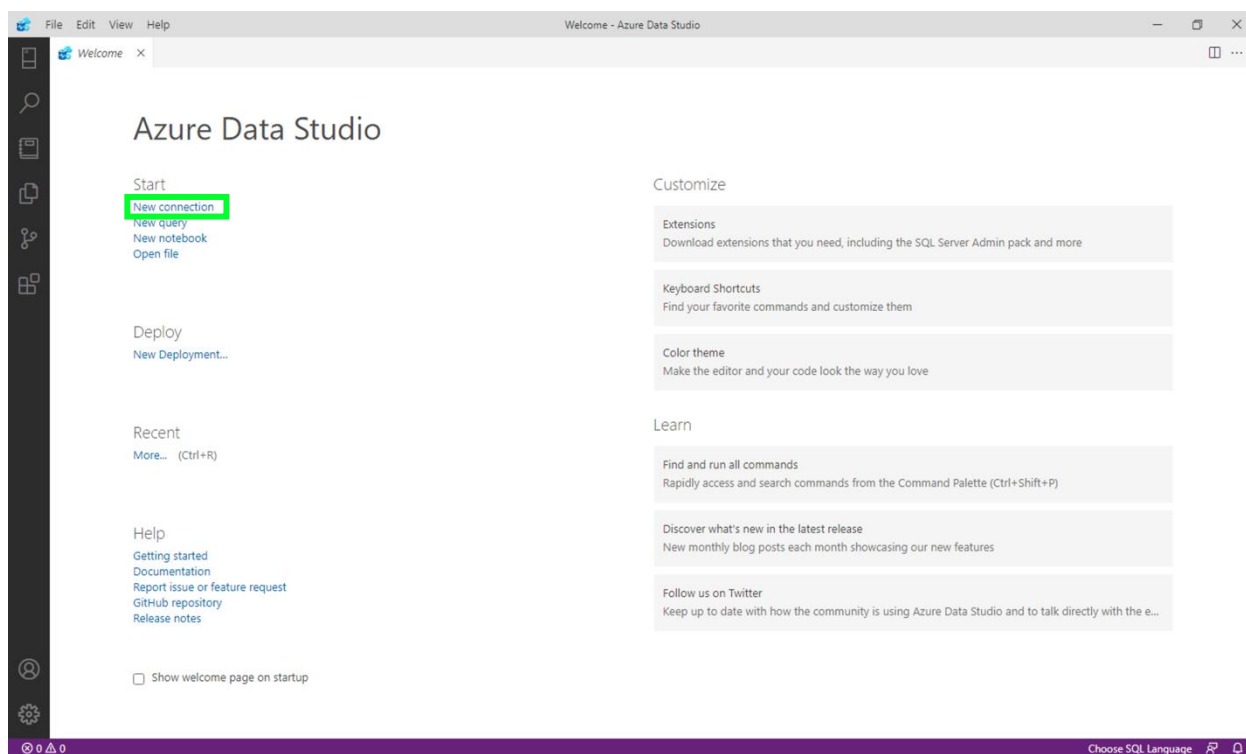


Рисунок 3.8 — Главная страница Azure Data Studio

В открывшемся окне необходимо ввести параметры подключения аналогичные тем, что предлагалось ввести в SSMS, но в данном программном средстве имя сервера необходимо вводить вручную (Рисунок 3.9). Имя локального сервера обычно записывается в следующем виде:

`<Имя_компьютера>\<Имя_экземпляра_сервера>`

В случае локального сервера возможно указание «. \» вместо имени компьютера.

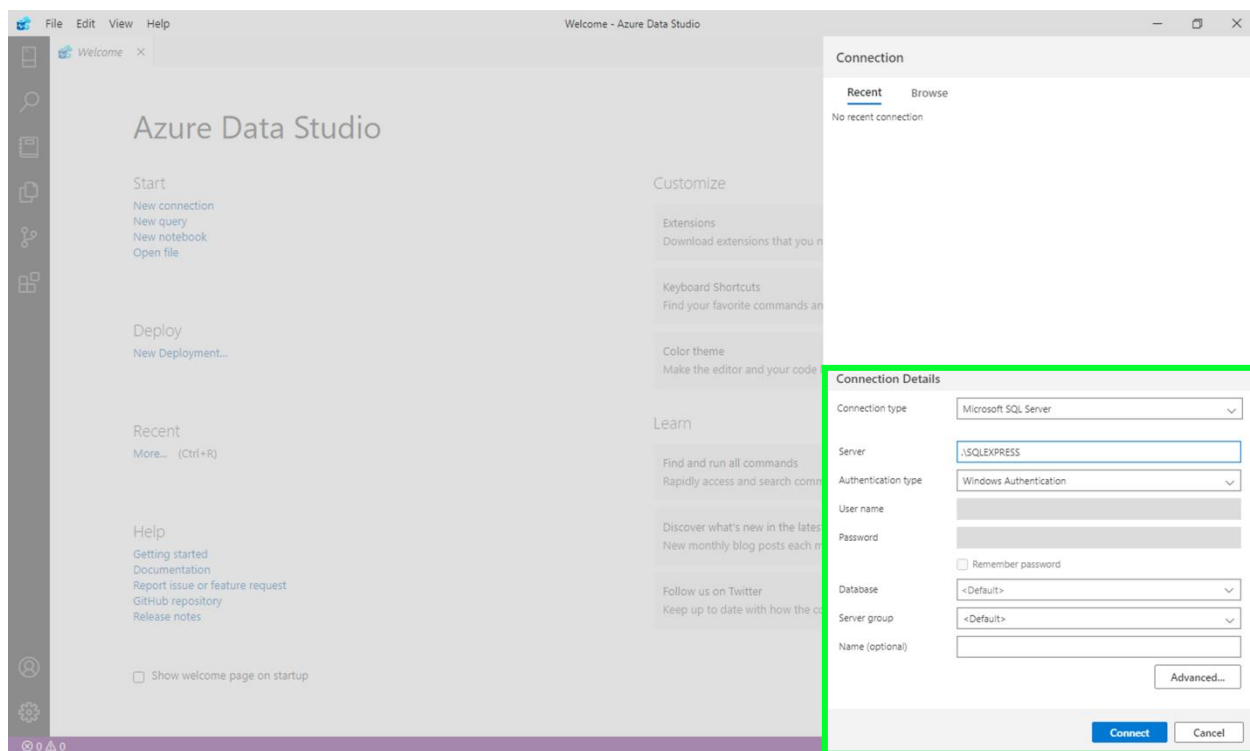


Рисунок 3.9 — Ввод параметров подключения

Для подключения к серверу необходимо нажать на кнопку «Connect» внизу диалогового окна «Connection». Если подключение выполнено успешно, то будет открыт монитор сервера (Рисунок 3.10).

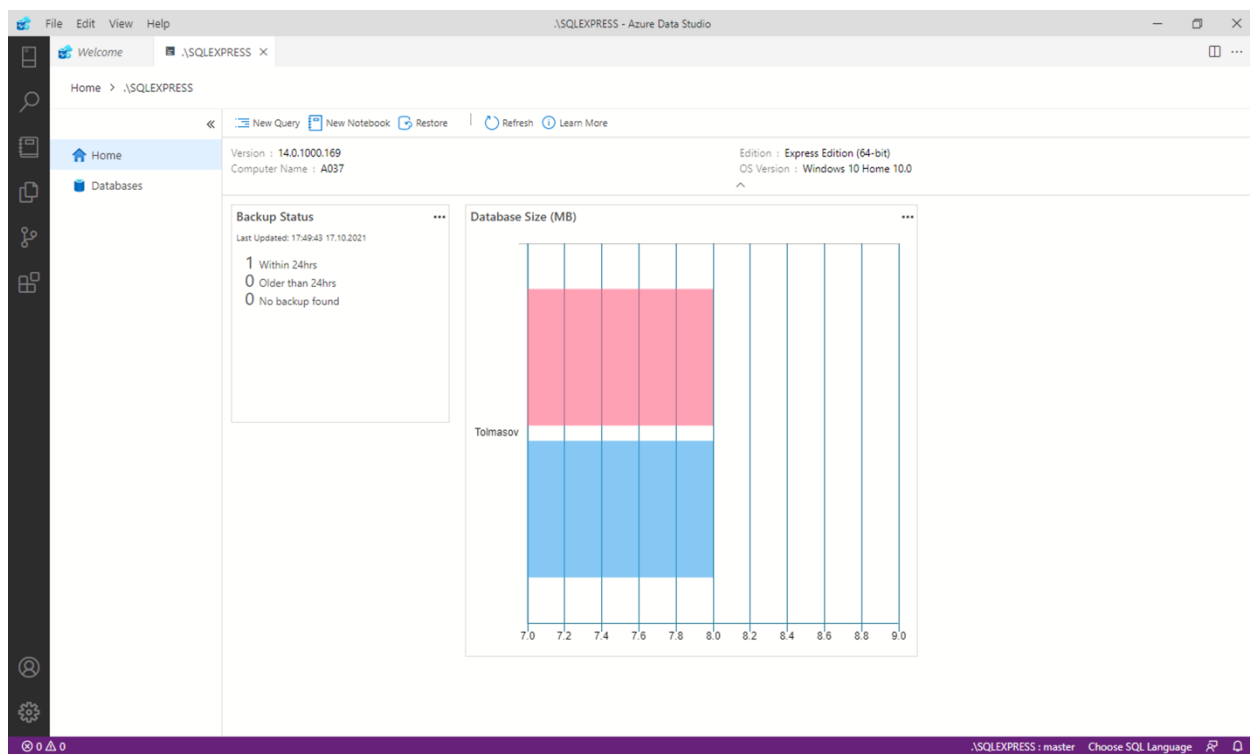


Рисунок 3.10 — Монитор сервера

Подключение DBeaver выполняется аналогично.

3.2 Системные базы данных MS SQL Server

Каждый экземпляр сервера содержит четыре доступных для просмотра и изменения системных баз данных и одну скрытую системную базу. В данном пособии кратко рассмотрено назначение основных системных баз данных, с которыми пользователь СУБД может взаимодействовать, а также ограничения, которые наложены на них.

3.2.1 База данных *master*

Системная база данных *master* является критически важной базой данных, повреждение которой вызовет отказ всего сервера и потери доступа ко всем пользовательским базам данных, расположенным на сервере.

База данных *master* содержит всю системную информацию об SQL Server: метаданные сервера, сведения об учётных записях и именах входа, параметры конфигурации системы и многие другие важные для инициализации и функционирования сервера объекты и информацию. В базе данных *master* регистрируются все базы данных, размещенные на данном сервере, а также информация о месте расположения файлов баз данных на жестком диске системы. Именно по причине хранения столь важной информации, повреждения базы данных *master* приведут сервер в нерабочее состояние.

Microsoft даёт следующие рекомендации по работе с данной базой данных:

- всегда имейте в наличии актуальную резервную копию базы данных *master*;
- после выполнения следующих операций как можно быстрее создавайте резервную копию базы данных *master*: создание, изменение или удаление базы данных; изменение значений параметров конфигурации сервера или базы данных; изменение или удаление учетных записей входа;
- не создавайте в базе данных *master* пользовательские объекты. Если сделать это, придется чаще создавать резервные копии базы данных *master*.
- не устанавливайте в базе данных *master* параметр `TRUSTWORTHY` в значение `ON`.

Учитывая критическую важность рассматриваемой базы данных, поддержка ряда операций с ней была отключена. Приведём список неподдерживаемых операций, указанный в официальной документации СУБД:

- добавление файлов или файловых групп;
- резервное копирование, для базы данных *master* может быть выполнено только полное резервное копирование;
- изменение параметров сортировки. Параметрами сортировки по

умолчанию являются параметры сортировки сервера;

- изменение владельца базы данных. Владелец *master* является *sa*;
- создание полнотекстового каталога или полнотекстового индекса;
- создание триггеров для системных таблиц базы данных;
- удаление базы данных;
- удаление пользователя *guest* из базы данных;
- включение системы отслеживания измененных данных;
- участие в зеркальном отображении базы данных;
- удаление первичной файловой группы, первичного файла данных или файла журнала;
- переименование базы данных или первичной файловой группы;
- перевод базы данных в режим «вне сети» (OFFLINE);
- перевод базы данных или первичной файловой группы в режим READ_ONLY.

Таким образом, следует соблюдать предельную осторожность при работе с системной базой данных *master*, так как её повреждение приведёт к значительным потерям, которые в некоторых случаях могут стать невосполнимыми.

3.2.2 База данных *model*

В процессе работы с СУБД часто требуется создавать новые базы данных. В процессе создания базы данных необходимо явно указывать значение каждого её свойства, что может представлять существенные трудности ввиду их большого количества. Кроме того, большинство из свойств одинаковы для многих создаваемых баз данных, следовательно, представляется логичным задать все свойства один раз для одной базы данных и в последствии копировать её, меняя только нужные в настоящий момент свойства. Именно такой базой данных и является *model*.

Рассматриваемая база данных является шаблоном для всех пользовательских баз данных, которые создаются на данном сервере. При создании пользовательской базы данных содержимое *model* полностью копируется в создаваемую базу данных, включая параметры. Параметры, которые явно указаны пользователем при создании, замещают параметры, указанные в шаблоне, и таким образом становится возможным сократить размер скрипта, необходимого для выполнения операции.

На базу данных *model* накладываются ограничения на выполнение некоторых операций:

- добавление файлов или файловых групп;
- изменение параметров сортировки. Параметрами сортировки по

умолчанию являются параметры сортировки сервера;

- изменение владельца базы данных. Владелец *model* является *sa*;
- удаление базы данных;
- удаление пользователя *guest* из базы данных;
- включение системы отслеживания измененных данных;
- участие в зеркальном отображении базы данных;
- удаление первичной файловой группы, первичного файла данных или файла журнала;
- переименование базы данных или первичной файловой группы.
- перевод базы данных в режим «вне сети» (OFFLINE);
- перевод первичной файловой группы в режим READ_ONLY;
- создание процедур, представлений или триггеров с помощью параметра WITH ENCRYPTION. Ключ шифрования привязывается к базе данных, в которой был создан объект. Зашифрованные объекты, созданные в базе данных *model* могут быть использованы только в базе данных *model*.

Повреждения базы данных *model* менее критичны, чем *master*, но также создадут определённые трудности при работе с сервером баз данных, так как она участвует в процессе создания системной базы данных *tempdb* и пользовательских баз данных. Поэтому рекомендуется создавать её резервную копию перед внесением изменений.

3.2.3 База данных *tempdb*

Несмотря на своё название, которое ассоциируется с чем-то временным, ненужным и т.д. системная база данных *tempdb* является важной системной базой данных. Она представляет собой глобальный ресурс, который содержит временные пользовательские объекты и задействована в некоторых системных операциях, выполняемых сервером в процессе работы. В отличие от других системных баз данных *tempdb* пересоздаётся каждый раз при запуске сервера. Именно по этой причине база данных *model* должна поддерживаться в исправном состоянии, так как для создания *tempdb* необходимо применение шаблона, определяемого *model*.

Учитывая факт того, что сервер при каждом запуске пересоздаёт *tempdb*, резервное копирование и восстановление данной базы не целесообразно и отключено.

Подробнее с параметрами системной базы данных *tempdb* можно ознакомиться на соответствующей странице документации [4].

3.2.4 База данных msdb

База данных *msdb* используется в задачах автоматизации сервера. Она используется SQL-агентом сервера при создании расписаний предупреждений и заданий. Эта база данных присутствует во всех выпусках *SQL Server*, но не применяется в выпусках *Express*, так как данные выпуски не позволяют пользователю управлять агентом.

На базу данных наложены следующие ограничения:

- Изменение параметров сортировки. Параметрами сортировки по умолчанию являются параметры сортировки сервера.
- Удаление базы данных.
- Удаление пользователя *guest* из базы данных.
- Включение системы отслеживания измененных данных.
- Участие в зеркальном отображении базы данных.
- Удаление первичной файловой группы, первичного файла данных или файла журнала.
- Переименование базы данных или первичной файловой группы.
- Перевод базы данных в режим «вне сети» (OFFLINE).
- Перевод первичной файловой группы в режим *READ_ONLY*.

3.2.5 Перенос файлов системных баз данных

Иногда возникают ситуации, когда необходимо перенести физические файлы системных баз данных. Например, это может пригодиться при повреждении системных баз в случае сбоя или при обслуживании дисков. Кроме того, перемещение файлов может быть выполнено плановое перемещение, когда администратор считает важным размещение файлов системных баз данных вне стандартного каталога СУБД.

Отработку навыков переноса файлов системных баз данных лучше всего производить на виртуальной машине, так как данная операция может привести к выходу из строя экземпляра сервера.

Для планового перемещения файлов системных баз данных, кроме *master* и скрытой системной базы данных *resources* (не рассматривается в данном пособии) необходимо применять следующий алгоритм:

1. Подготовить операционную систему к переносу. На данном этапе необходимо создать папку, куда планируется перенести файлы, и предоставить к ней полномочный доступ для пользователей, входящих в группу *mssql*.
2. Удостовериться, что пользователь, осуществляющий перемещение файлов

баз данных имеет необходимые разрешения ALTER на уровне сервера.

3. Узнать логические имена переносимых файлов, выполнив для этого следующий запрос:

```
SELECT name, physical_name
FROM sys.master_files
WHERE database_id = DB_ID(N'<database_name>');
```

4. Для каждого файла выполнить следующий скрипт:

```
ALTER DATABASE <database_name> MODIFY FILE
(NAME = logical_name ,
FILENAME = '<new_path\os_file_name>')
```

5. Остановить работу экземпляра сервера. Процесс остановки экземпляра сервера зависит от используемой операционной системы.
6. Выполнить перемещение файлов в новое расположение. В случае переносов файлов tempdb выполнение перемещения файлов необязательно и по окончании процедуры старые файлы tempdb можно удалить.
7. Запустить экземпляр сервера и проверить изменения файлов с помощью запроса:

```
SELECT name, physical_name AS CurrentLocation, state_desc
FROM sys.master_files
WHERE database_id = DB_ID(N'<database_name>');
```

Таким образом возможен перенос файлов системных баз данных в новое расположение, но управляться они будут тем же экземпляром сервера, что и раньше. Если возникает необходимость переноса системных баз данных на новый экземпляр сервера, то необходимо применить процедуру резервного копирования и восстановления.

После переноса файлов системных баз данных могут возникнуть следующие проблемы:

- ошибки при создании новых пользовательских баз данных;
- ошибки запуска агента SQL Server (не актуально для выпусков *Express*).

Для решения проблемы с созданием пользовательских баз данных необходимо воспользоваться средой SSMS.

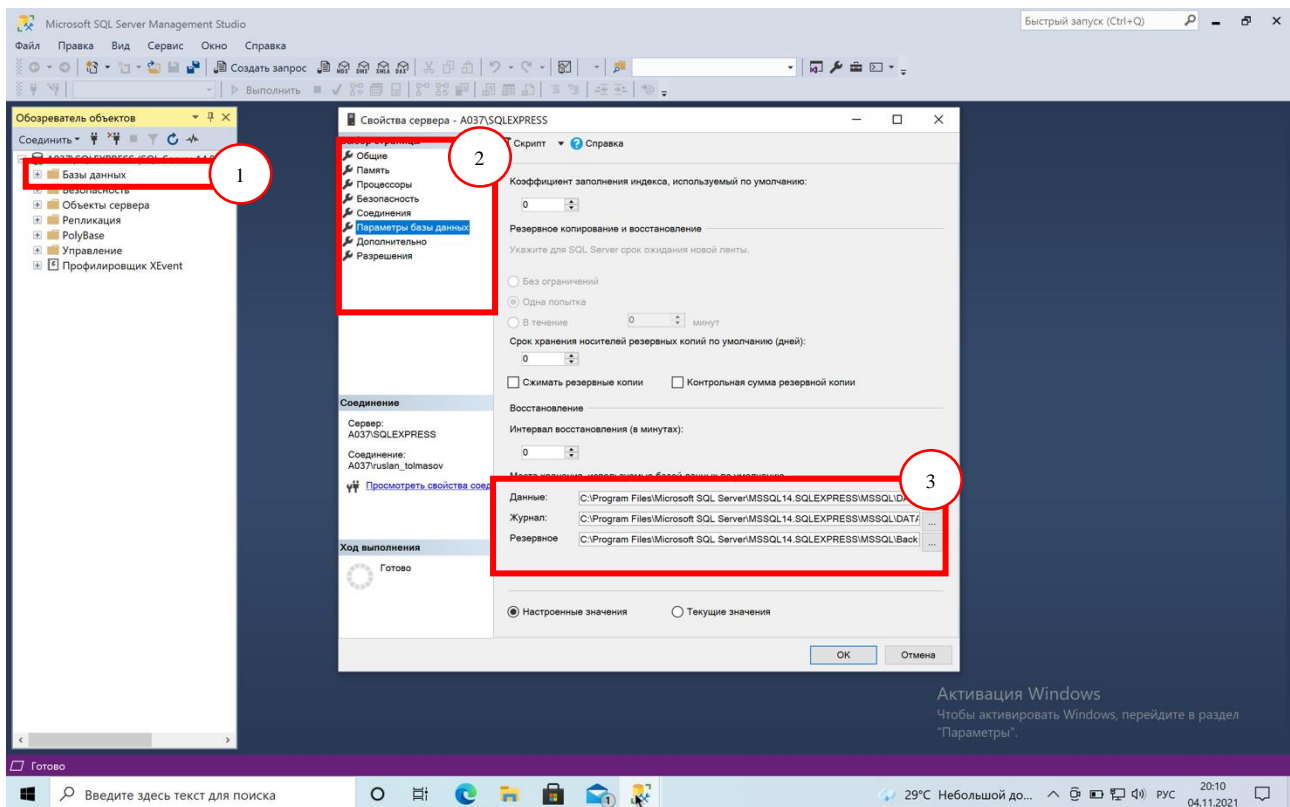


Рисунок 3.11 — Изменение расположения файлов баз данных по умолчанию

Необходимо выполнить следующую последовательность действий:

1. Подключиться к экземпляру сервера используя SSMS.
2. Нажать правой кнопкой мыши по имени экземпляра сервера (1, рис.3.11) и выбрать в контекстном меню пункт «**Свойства**».
3. В окне свойства сервера выбрать страницу «**Параметры базы данных**» (2, рис. 3.11).
4. В нижней части формы можно посмотреть пути к местам хранения, используемым по умолчанию (3, рис. 3.11).
5. Измените пути, при необходимости.
6. Для вступления изменений в силу необходимо перезагрузить экземпляр сервера.

Следует напомнить, что тонкую настройку сервера необходимо выполнять с помощью SSMS, так как в остальных клиентских приложениях существенно урезан функционал, связанный с администрированием. Это существенно ограничивает возможность выбора операционной системы для выполнения функций администратора SQL Server. Ситуация постепенно изменяется за счёт плагинов, которые активно разрабатываются для Azure Data Studio и возможно, что в ближайшее время выполнять тонкую настройку можно будет и в других операционных системах.

3.3 Пользовательские базы данных

Создание пользовательской базы данных возможно через интерфейс SSMS или с помощью средств языка T-SQL.

При создании пользовательских баз данных следует учитывать следующие ограничения и рекомендации:

- для создания базы данных любым из описанных ниже способов, пользователь должен обладать разрешением CREATE DATABASE в базе данных master или разрешения CREATE/ALTER ANY DATABASE;
- каждый экземпляр сервера может содержать не более 32 767 баз данных;
- рекомендуется создавать резервную копию базы данных master каждый раз после создания, изменения или удаления пользовательской БД.

3.3.1 Создание пользовательской базы данных с помощью MS SSMS

Для создания новой базы данных, используя графический интерфейс SSMS, необходимо выполнить следующую последовательность действий: нажать правой кнопкой мыши по вкладке «Базы данных» окна «Обозреватель объектов» и выбрать в контекстном меню «Создать базу данных...» (Рисунок 3.12)

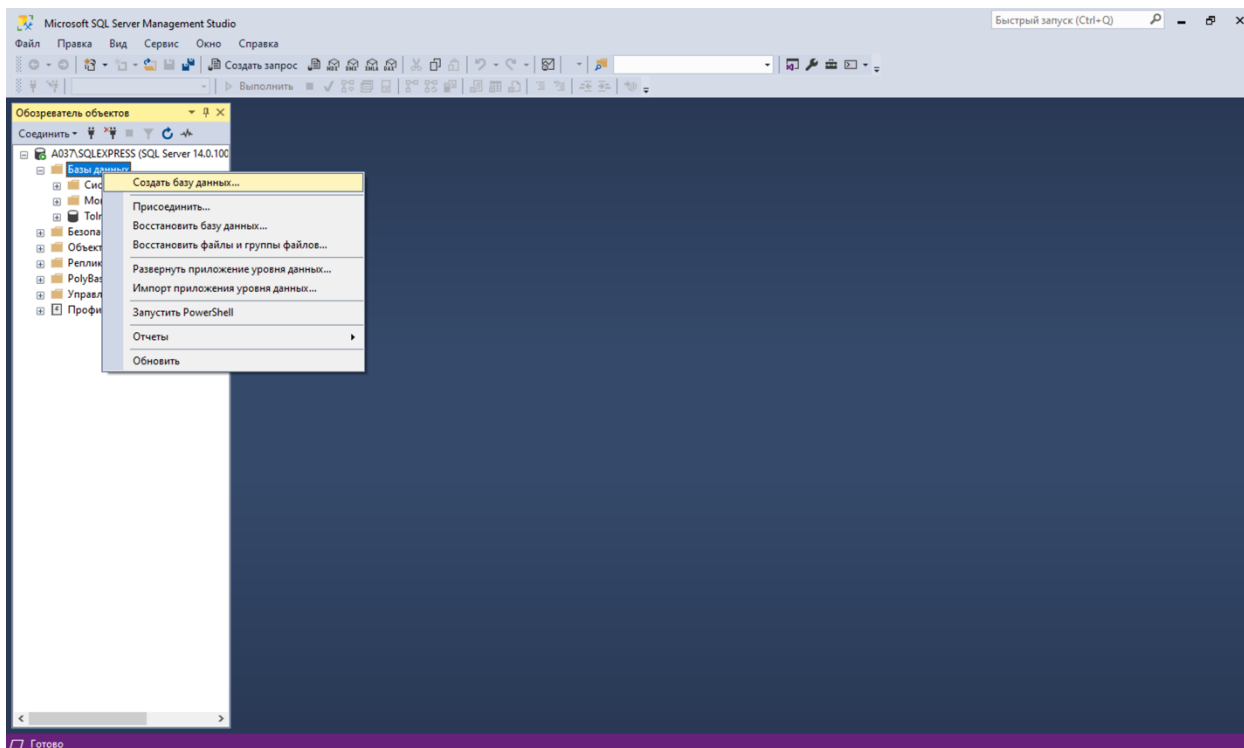


Рисунок 3.12 — Контекстное меню коллекции объектов "Базы данных"

В открывшемся диалоговом окне на странице «Общие» необходимо задать имя базы данных и при необходимости указать владельца БД. На данной странице также возможно изменить параметры файлов баз данных (Рисунок 3.13).

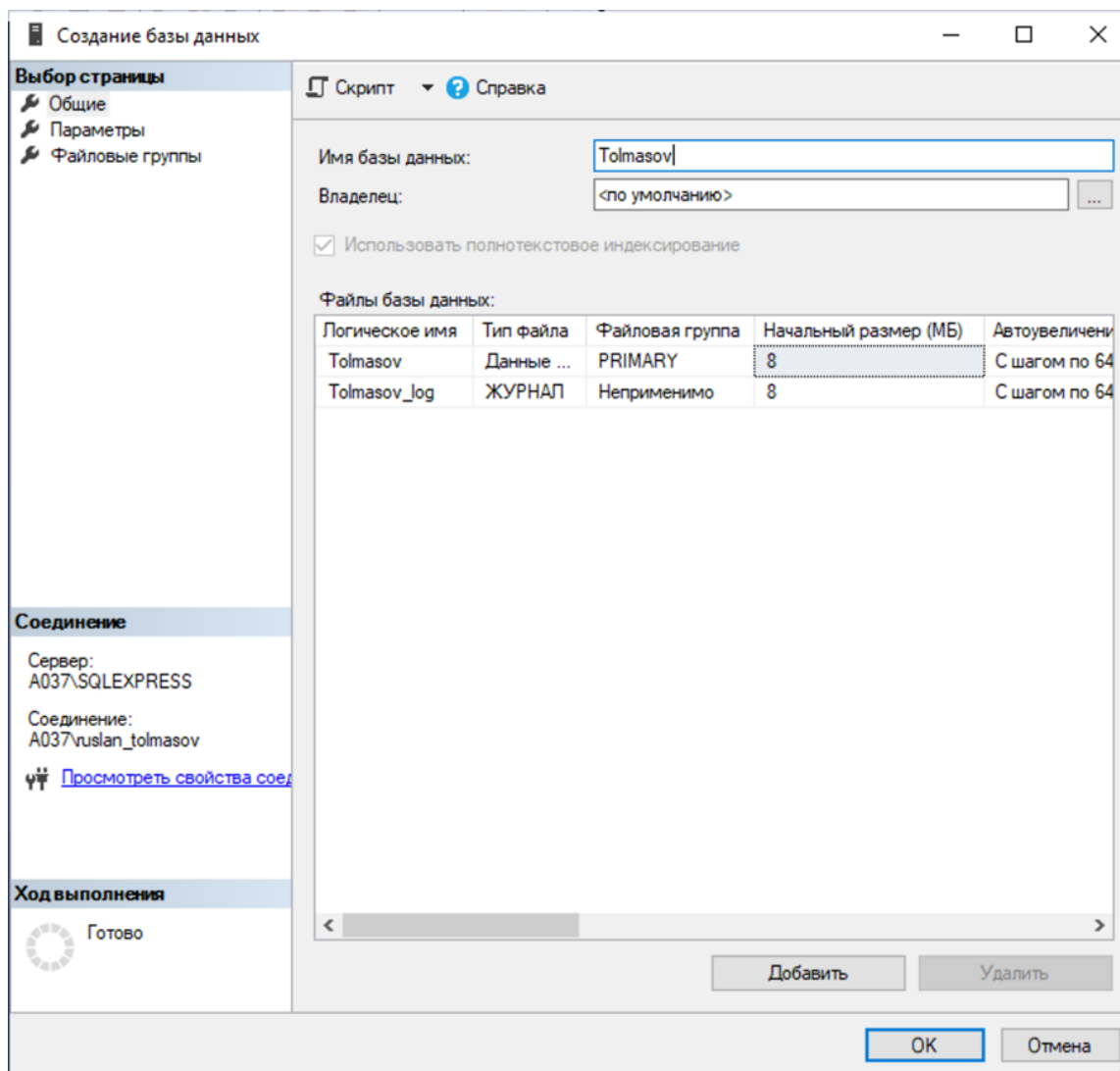


Рисунок 3.13 — Ввод основных параметров БД

Если требуется более точная настройка, то все параметры, связанные с автоматизацией, восстановлением и другие доступны на странице «**Параметры**», доступ к которой можно получить, используя панель «**Выбор страницы**» в левой верхней части экранной формы «**Создание баз данных**».

По умолчанию значение параметров устанавливается в соответствии с настройками базы данных *model*, рассмотренной ранее в пункте 3.2.2.

Пример экранной формы с открытой страницей «**Параметры**» приведён на рисунке ниже (Рисунок 3.14).

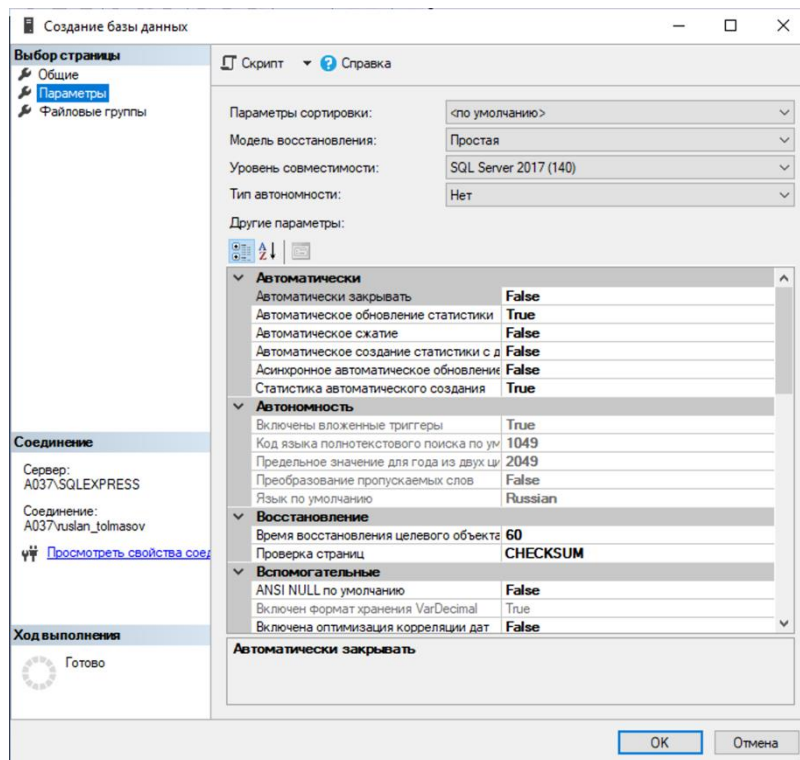


Рисунок 3.14 — Выбор значений основных и прочих параметров БД

Последняя страница («**Файловые группы**») позволяет настроить файловые группы, применяемых в создаваемой базе данных. Кроме файловых групп база данных может содержать и специальные объекты.

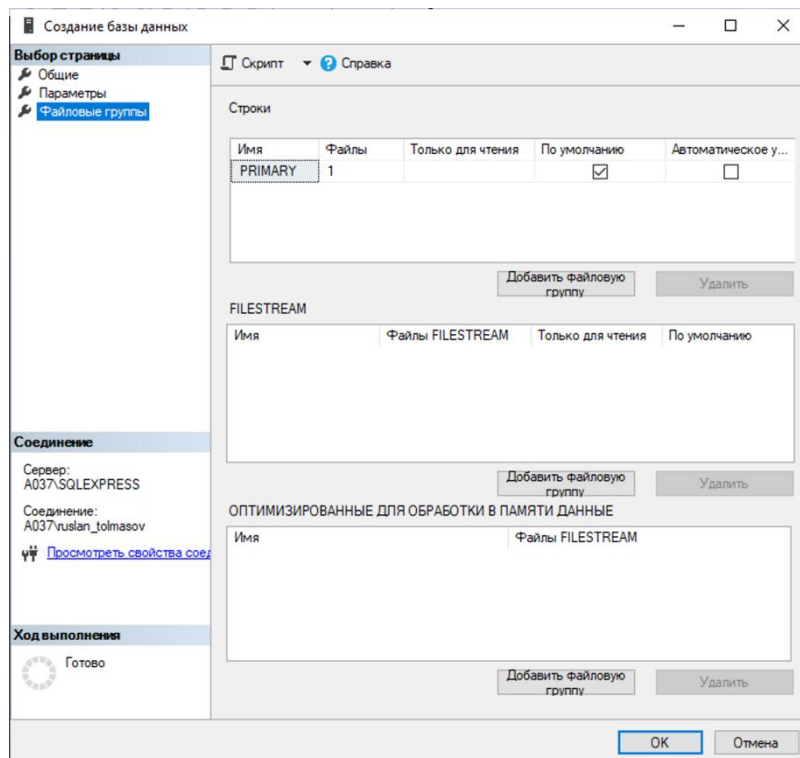


Рисунок 3.15 — Настройка файловых групп

К специальным объектам относятся файловые потоки и данные оптимизированные для хранения в памяти.

3.3.2 Создание базы данных средствами Transact SQL

В условиях, когда нет возможности воспользоваться созданием базы данных через графический интерфейс SSMS, необходимо использовать язык определения данных *DDL*. Рассмотрим полный синтаксис инструкции создания базы данных, приведённой в официальной документации MS SQL Server.

Метаязык описания синтаксических конструкций

Все инструкции описаны в документации с помощью специального метаязыка, который представляет собой набор общепринятых обозначений, позволяющих компактно описать синтаксис любой инструкции для любого языка.

Перед рассмотрением синтаксиса инструкции **CREATE DATABASE**, познакомимся с конструкциями метаязыка, применяемого для их описания:

- необязательные фрагменты кода заключаются в квадратные скобки. Например:

```
CREATE DATABASE db_name [.] --точка с запятой не обязательна
```

- если в инструкции должно стоять одно из нескольких значений или одна из нескольких инструкций или команд, то для изображения списка выбора, все возможные варианты заключаются в фигурные скобки, а варианты отделяются друг от друга вертикальной чертой, которая эквивалентно союзу «или». Например:

```
DEFAULT_FULLTEXT_LANGUAGE = {lcid | language_name | language_alias}
```

- для сокращения длины приводимого шаблона инструкции, используются псевдонимы. Особенно полезны псевдонимы в случае, если они содержат длинные списки выбора или множество различных опций. Псевдонимы обычно присваиваются группам объектов, обладающих одинаковыми свойствами или выполняющих одинаковую функцию, но структурно отличающихся друг от друга. Псевдонимы заключаются в угловые скобки:

```
<option>;
```

- каждый псевдоним в инструкции должен быть заменён любым допустимым значением. Обычно все допустимые варианты описываются сразу после описания синтаксиса рассматриваемой инструкции с помощью специального обозначения «:=», которое эквивалентно фразе «это есть»;
- существуют ситуации, когда некоторая синтаксическая конструкция может быть указана один и более раз. Поясним на примере ситуации, когда на месте некоторого псевдонима ожидается один или несколько

вариантов:

[PRIMARY] <filespec> [,...n]

что может быть раскрыто в следующую строку:

[PRIMARY] <filespec>, <filespec>, <filespec>, ..., <filespec>

Синтаксис инструкции CREATE DATABASE

Рассмотрев все необходимые конструкции метаязыка, можно рассмотреть синтаксис инструкции создания базы данных *Transact SQL*. Приведённый Листинг иллюстрирует все поддерживаемые варианты написания инструкции:

```
CREATE DATABASE database_name
[ CONTAINMENT = { NONE | PARTIAL } ]
[ ON
    [ PRIMARY ] <filespec> [ ,...n ]
    [ , <filegroup> [ ,...n ] ]
    [ LOG ON <filespec> [ ,...n ] ]
]
[ COLLATE collation_name ]
[ WITH <option> [ ,...n ] ]
[;]

<option> ::=
{
    FILESTREAM ( <filestream_option> [ ,...n ] )
| DEFAULT_FULLTEXT_LANGUAGE = { lcid | language_name | language_alias }
| DEFAULT_LANGUAGE = { lcid | language_name | language_alias }
| NESTED_TRIGGERS = { OFF | ON }
| TRANSFORM_NOISE_WORDS = { OFF | ON }
| TWO_DIGIT_YEAR_CUTOFF = <two_digit_year_cutoff>
| DB_CHAINING { OFF | ON }
| TRUSTWORTHY { OFF | ON }
| PERSISTENT_LOG_BUFFER=ON ( DIRECTORY_NAME='<Filepath to folder on DAX formatted volume>' )
}

<filestream_option> ::=
{
    NON_TRANSACTED_ACCESS = { OFF | READ_ONLY | FULL }
| DIRECTORY_NAME = 'directory_name'
}

<filespec> ::=
{
(
```

```

NAME = logical_file_name ,
FILENAME = { 'os_file_name' | 'filestream_path' }
[ , SIZE = size [ KB | MB | GB | TB ] ]
[ , MAXSIZE = { max_size [ KB | MB | GB | TB ] | UNLIMITED } ]
[ , FILEGROWTH = growth_increment [ KB | MB | GB | TB | % ] ]
)
}

<filegroup> ::=
{
FILEGROUP filegroup name [ [ CONTAINS FILESTREAM ] [ DEFAULT ] | CONTAINS MEMORY_OPTI-
MIZED_DATA ]
    <filespec> [ ,...n ]
}

```

Назначение каждого параметра подробно рассматривается в документации СУБД.

Благодаря наличию системной базы данных *model*, инструкция может быть записана в короткой форме:

```
CREATE DATABASE database_name
```

при этом все остальные параметры будут заимствованы из базы данных *model*. Однако, рекомендуется при создании баз данных явно указывать создаваемые файлы, их расположение, размер, увеличение размера, а также максимальный размер, который занимает файл. Приведём пример типового скрипта создания базы данных:

```

USE master
GO
-- Проверка существования базы данных, если существует - удалить
IF NOT EXISTS (
    SELECT [name]
FROM sys.databases
WHERE [name] = N'Tolmasov'
)
CREATE DATABASE Tolmasov
ON --определяем первичный файл БД
(
    NAME = tolmasov,
    FILENAME = N'/home/db_user/mydb/Tolmasov.mdf',
    SIZE = 1 Mb, MAXSIZE = 10 Mb,
    FILEGROWTH = 10%
)
LOG ON --определяем файл журнала транзакций
(

```

```
NAME = tolmasov_log,  
FILENAME = N'/home/db_user/mydb/Tolmasov_log.ldf',  
SIZE = 1 Mb, MAXSIZE = 10 Mb,  
FILEGROWTH = 10%
```

)

Перед созданием базы данных рекомендуется производить проверку существования этого объекта. Часто на этапе разработки БД, при её отладке, необходимо пересоздавать базу, чтобы избежать ошибок при выполнении сложного, многострочного скрипта, рекомендуется производить проверку существования объекта. Данную проверку существования можно производить для любого объекта, что может быть полезно при разработке скриптов.

После ключевого слова ON указывается первичный файл базы данных, его логическое и физическое имя, расположение и устанавливается начальный и максимальный размер и величина приращения размера файла, при его заполнении. Файл журнала транзакций описывается после ключевых слов LOG ON параметры, которыми описывается файл журнала транзакций аналогичны главному файлу.

Рассмотрим работу с файлами баз данных более подробно.

3.4 Управление размещением базы данных

3.4.1 Файлы базы данных

Для функционирования БД СУБД MS SQL Server необходимо наличие минимум двух файлов: файла данных (или первичного файла) и файла журнала транзакций. Если при создании файлов БД путь к ней не задавался явным образом, то они будут помещены в расположении по умолчанию (данный вопрос рассматривался в пункте 3.2.5). Для промышленных систем рекомендуется размещать файлы данных и файлы журналов транзакций на отдельных жестких дисках, если это решение будет оптимальным.

Основными свойствами файлов баз данных являются: *имя файла* (логическое и физическое), *размер* и *файловая группа*. Чтобы получить информацию о файлах, которые входят в состав базы данных, необходимо выполнить следующий скрипт, находясь в контексте интересующей базы данных:

```
SELECT * FROM sys.database_files
```

Познакомившись со структурой системной таблицы, рекомендуется запомнить некоторые поля и отказаться в дальнейшем от запросов, содержащих * вместо перечисления столбцов в пользу выборки только необходимых в данный момент столбцов таблицы.

Имена файлов

Каждый файл базы данных имеет два имени:

- логическое (*logic_file_name*) — уникальный идентификатор (то есть не повторяется среди логических имен других файлов базы данных), который используется в инструкциях T-SQL для обозначения ссылки на физический файл. Имя файла должно соответствовать правилам наименования объектов SQL Server;
- физическое имя (*os_file_name*) — идентификатор, содержащий полный путь к файлу, заданный в соответствии с правилами операционной системы⁴, на которой установлен сервер.

Таким образом, каждому файлу должно быть присвоено при создании два имени, которые в скрипте на языке T-SQL задаются свойствами NAME для логических имен и FILENAME для физического имени (пример создания файлов с помощью скрипта приведён выше, в пункте 3.3.2).

Каждый физический файл имеет расширение (обозначение типа файла, принятого во всех операционных системах). Расширение первичного файла базы данных образованно из первых букв словосочетания «*main database file*» (главный файл базы данных) и обозначается **.mdf**. Файл с таким разрешением должен быть единственным в базе данных, в нем хранится служебная информация о базе данных, а также сами данные.

Вторым важным файлом является файл журнала транзакций, который имеет расширение **.ldf** (*log database file*) и содержит в себе необходимую информацию о совершенных транзакциях. Данный журнал является важной частью базы данных и его утрата, повреждение или переполнение могут привести к выходу базы данных из строя, поэтому не следует удалять или переименовывать данный файл.

При создании базы данных под каждый файл выделяется указанный в скрипте или в *model* объём дискового пространства, при этом сами файлы заполняются пустыми страницами, которые будут заполнены в процессе работы с базой данных. СУБД поддерживает постепенное увеличение объёма выделенного под файлы дискового пространства, что явно должно быть указано при создании файлов.

Размер

Параметр «размер» определяет сколько дискового пространства займёт файл после создания. В T-SQL размер файла задаётся с помощью свойства файла

⁴ В приведённых в пособии примерах физические имена, заданные с помощью SSMS, соответствуют правилам и требованиям операционной системы Windows, в приведённых скриптах T-SQL — Ubuntu Server 20.04.

SIZE и может быть задан в килобайтах, мегабайтах, гигабайтах. Размер файла не может быть меньше 8 Кб, ограничение вызвано особенностью организации хранения информации в SQL Server.

Каждый файл состоит из страниц, каждая из которых занимает 8 Кб. Страница — основная единица хранения информации в базе данных. Каждая страница занимает 8 Кб. В системной таблице `sys.database_file` размер файла указывается именно в страницах, что может ввести начинающих администраторов в заблуждение, но зная этот факт и размер страницы, не составит особого труда пересчитать размер в привычные единицы (мегабайтах и другие). Созданный файл сразу занимает всё выделенное для него пространство, даже если в него не было помещено ни одной строки данных. Это происходит потому, что при создании файл заполняется пустыми страницами, которые будут заполняться в процессе работы с базой данных.

Но размер файлов непостоянный и может автоматически увеличиваться на заданную величину каждый раз, когда выделенное дисковое пространство заканчивается. Величина приращения задаётся свойством `FILEGROWTH` в процентах от текущего размера или фиксированной величиной. Чтобы ограничить рост файла необходимо указывать максимальный размер файла с помощью свойства `MAXSIZE`. Если максимальный размер не задан или указан как `UNLIMITED`, то файл будет увеличиваться до тех пор, пока на диске будет свободное место. В случае невозможности дальнейшего расширения, будет сгенерирована исключительная ситуация и новые данные не будут сохраняться. В случае переполнения журнала транзакций база данных не сможет нормально функционировать.

3.4.2 Файловые группы

Создание файловых групп позволяет физически разделить хранение логически связанных данных. Например, данные, которые относятся к одной таблице, могут храниться в разных файлах и даже на разных логических дисках, при условии, что они принадлежат одной файловой группе. В процессе создания базы данных в ней создается первичная файловая группа `PRIMARY`, которая используется по умолчанию. Это значит, что каждый последующий созданный файл без указания файловой группы будет включен в группу `PRIMARY`. Далее мы рассмотрим назначение и основные характеристики файловых групп, а в следующем разделе рассмотрим процесс их создания и изменения.

Существует четыре типа файловых групп, из которых в данном пособии будут рассмотрены наиболее часто применяемые и простые для ознакомления группы:

- первичная файловая группа — группа создаваемая одновременно с созданием базы данных, независимо от того, объявят её явно или нет. В ней хранится первичный файл, все системные таблицы базы и все файлы, для которых в момент их создания не была указана файловая группа;
- пользовательская группа — файловая группа, явно объявленная пользователем в процессе создания базы данных или при её изменении. Здесь хранятся дополнительные файлы баз данных. Пользовательская файловая группа может быть назначена группой по умолчанию, в этом случае все созданные несистемные объекты будут относиться к этой файловой группе автоматически.

Информацию о других типах файловых групп (данные, оптимизированные для памяти и файловые потоки) можно найти в официальной справочной системе Microsoft [5].

Как было упомянуто выше, в базе данных всегда присутствует файловая группа по умолчанию, в которую попадают объекты, для которой не указана никакая другая файловая группа. В момент создания базы данных в качестве группы по умолчанию задается первичная файловая группа. Необходимо следить, чтобы размеры файлов в группе по умолчанию были достаточно большими, чтобы вместить в себя новые объекты, для которых группа не назначена.

Объединение файлов в группы позволяет распределять запросы к базе данных и тем самым повысить производительность, за счет того, что файлы в составе одной файловой группы будут находиться на разных физических носителях.

Объединение файлов в файловые группы позволяет обеспечить их равномерное наполнение данными за счёт применения стратегии пропорционального наполнения файлов. При пропорциональном заполнении файлов в файловой группе заполнение осуществляется пропорционально свободному месту в этих файлах, вместо записи всех данных в один файл⁵. Таким образом достигается равномерное распределение данных по файлам.

Файловые группы также влияют и на рост размера файлов. Файлы расширяются и заполняются поочередно. Увеличивается первый добавленный в группу файл, после заполнения выделенного объёма, расширяется следующий файл и так по кругу, до тех пор, пока каждый из файлов не достигнет максимального размера или не закончится место на дисках. Администратор может изменить политику расширения файлов входящих в файловую группу, указав в процессе создания файловой группы опцию `AUTOGROW_ALL_FILES`, в этом случае будет

⁵ Для того чтобы наглядно убедиться в этом проведите простой эксперимент, выполнив задания в конце раздела.

происходить автоматическое расширение всех файлов.

Существует несколько правил, которые необходимо соблюдать при проектировании файловых групп:

- каждый файл базы данных может одновременно принадлежать только одной файловой группе;
- файл журнала транзакций не может входить ни в одну файловую группу;
- файлы и файловые группы могут быть использованы несколькими базами данных.

В большинстве случаев для баз данных достаточно одного файла данных и файла журнала транзакций, и пользовательские файловые группы в них не используются. При необходимости использования в базе данных более одного файла рекомендуется создать новую файловую группу, в который поместить новый файл базы данных и сделать ее группой по умолчанию. Таким образом можно добиться разделения системной части базы данных от пользовательских объектов.

Также Microsoft даёт и следующие рекомендации по оптимизации работы базы данных применяя файловые группы:

- по возможности следует разносить файлы и файловые группы по разным физическим дискам;
- объекты конкурирующие за свободное пространство желательно размещать в разных файловых группах;
- применять файловые группы следует для целенаправленного размещения файлов базы данных на разных дисках;
- следует помещать разные таблицы, использующиеся в одних и тех же запросах с соединениями, в разные файловые группы. Это увеличит производительность, так как для поиска соединяемых данных можно будет использовать параллельный ввод-вывод;
- не следует помещать файлы журнала транзакций на тот же физический диск, где находятся другие файлы и файловые группы.

Рассмотрев основные элементы базы данных, перейдём к вопросу организации создания и изменения файлов и файловых групп.

3.4.3 Добавление и изменение файлов и файловых групп

Файлы баз данных можно изменять в процессе работы базы данных. Можно менять их расположение, размер и другие параметры. Для этого необходимо, чтобы пользователь, выполняющий данные действия имел разрешение ALTER DATABASE.

Добавление или изменение файлов с помощью MS SSMS

Так как файл — это объект, входящий в состав базы данных, работа с ними осуществляется через окно свойств на странице файлы (рис. 3.16):

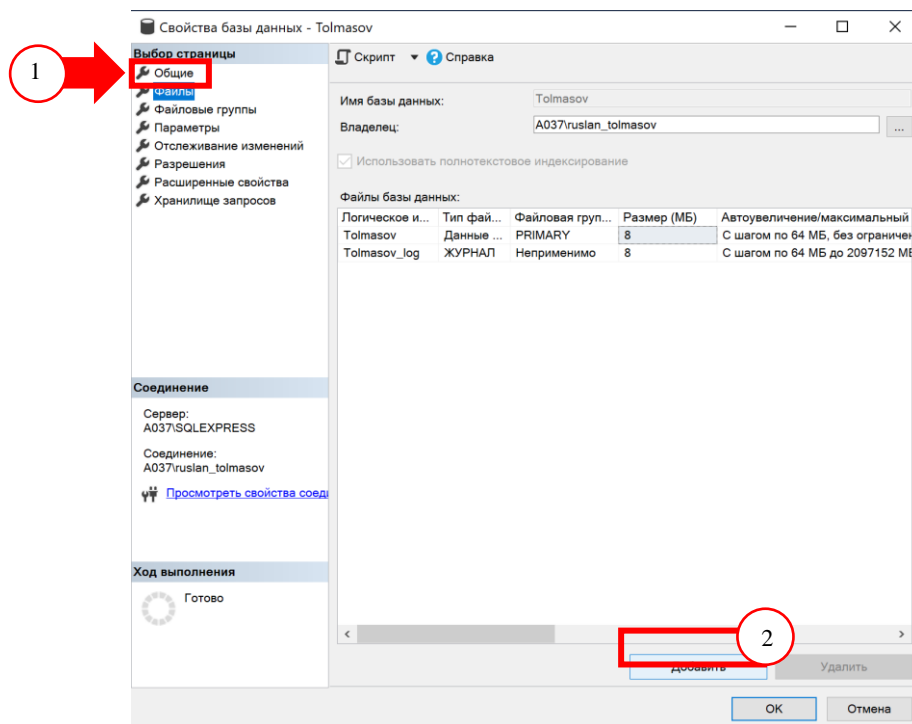


Рисунок 3.16 — Окно свойств базы данных

Для доступа к окну свойств базы данных выполните следующие действия:

1. Подключитесь к серверу с помощью SSMS.
2. Перейдите в каталог баз данных в инспекторе объектов и правой кнопкой мыши щелкните на базе данных, к которой необходимо добавить файл.
3. В контекстном меню выберите пункт «Свойства».
4. В открывшемся окне перейдите на страницу «Файлы» (1, рис. 3.16).

С помощью данной формы можно отредактировать свойства существующих файлов (например, изменить размер файла или его расположение), а также добавить новый файл, нажав на кнопку «Добавить» внизу формы (2, рис. 3.16).

Файлы базы данных:						
Логическое и...	Тип файла	Файловая груп...	Размер (МБ)	Автоувеличение/максимальный размер	Путь	Имя файла
Tolmasov	Данные СТРОК	PRIMARY	8	С шагом по 64 МБ, без ограничений	C:\Program Files\Microsoft SQL Server\MSSQL14.SQLEXPRESS\MSSQL\DATA	Tolmasov.mdf
Tolmasov_log	ЖУРНАЛ	Неприменимо	8	С шагом по 64 МБ до 2097152 МБ	C:\Program Files\Microsoft SQL Server\MSSQL14.SQLEXPRESS\MSSQL\DATA	Tolmasov_log.ldf
Tolmasov_new	Данные СТРОК	PRIMARY	8	С шагом по 64 МБ, без ограничений	C:\Program Files\Microsoft SQL Server\MSSQL14.SQLEXPRESS\MSSQL\DATA\	

Рисунок 3.17 — Результат выполнения операции «Добавить»

В добавленной строке обязательно требуется задать только логическое имя, остальные параметры, кроме физического имени файла, уже заполнены данными по умолчанию, но их можно изменить при необходимости. Обычно для новых файлов вручную устанавливают значения следующих параметров: *логическое*

имя, тип файла, файловую группу, размер и настройки авто увеличения файлов.

Новый файл базы данных получает расширение *.ndf*, файлов с таким расширением может быть создано несколько, ограничения на количество файлов, прикреплённых к БД, в явном виде не установлено, но оно должно быть обоснованно, так как наличие большого количества файлов создаёт дополнительные сложности администрирования и угрозы целостности базы данных.

Для подтверждения изменений следует нажать «**Ок**» в нижней части формы. Теперь в базе данных создан ещё один файл для хранения строк.

Добавление или изменение файлов и файловых групп с помощью Transact SQL

Рассмотрим синтаксис инструкции для изменения параметров файлов и файловых групп:

```
ALTER DATABASE database_name
{
    <add_or_modify_files>
  | <add_or_modify_filegroups>
}
<add_or_modify_files>::=
{
    ADD FILE <filespec> [ ,...n ]
      [ TO FILEGROUP { filegroup_name } ]
  | ADD LOG FILE <filespec> [ ,...n ]
  | REMOVE FILE logical_file_name
  | MODIFY FILE <filespec>
}
<filespec>::=
(
    NAME = logical_file_name
  [ , NEWNAME = new_logical_name ]
  [ , FILENAME = { 'os_file_name' | 'filesystem_path'
                  | 'memory_optimized_data_path' } ]
  [ , SIZE = size [ KB | MB | GB | TB ] ]
  [ , MAXSIZE = { max_size [ KB | MB | GB | TB ] | UNLIMITED } ]
  [ , FILEGROWTH = growth_increment [ KB | MB | GB | TB | % ] ]
  [ , OFFLINE ]
)
<add_or_modify_filegroups>::=
{
  | ADD FILEGROUP filegroup_name
    [ CONTAINS FILESTREAM | CONTAINS MEMORY_OPTIMIZED_DATA ]
  | REMOVE FILEGROUP filegroup_name
  | MODIFY FILEGROUP filegroup_name
    { <filegroup_updatability_option>
```

```

| DEFAULT
| NAME = new_filegroup_name
| { AUTOGROW_SINGLE_FILE | AUTOGROW_ALL_FILES }
}
}
<filegroup_updatability_option>::=
{ { READONLY | READWRITE }
  | { READ_ONLY | READ_WRITE }
}

```

При помощи инструкции ALTER DATABASE можно добавлять изменять файлы и файловые группы в пользовательской базе данных. Рассмотрим несколько примеров создания и изменения элементов.

Добавим новый файл к базе данных. Изучив синтаксис инструкции, становится очевидно, как построить инструкцию для добавления файла. Приведём вариант инструкции добавления файла размером 1 Мб к базе данных *Tolmasov*:

```

ALTER DATABASE Tolmasov ADD FILE
(
  NAME = N'new',
  FILENAME = N'/home/db_user/mydb/new.ndf',
  SIZE = 1 Mb, MAXSIZE = 15 Mb,
  FILEGROWTH = 10%
)

```

Файл добавится к файловой группе PRIMARY, но предположим, что требуется добавить файл к другой файловой группе. В этом случае её необходимо предварительно создать. Принадлежность файла к файловой группе изменить нельзя, поэтому мы пересоздадим файл new.ndf, в первую очередь, чтобы продемонстрировать процесс удаления файла. После этого будет создана новая файловая группа, которая будет использоваться по умолчанию. В завершении создадим файл заново, но при создании явно укажем его принадлежность к новой файловой группе:

```

--Удаление пользовательского файла
ALTER DATABASE Tolmasov REMOVE FILE new;

--создание пользовательской файловой группы
ALTER DATABASE Tolmasov ADD FILEGROUP new_filegroup;

--Пересоздаём файл БД в новой файловой группы
ALTER DATABASE Tolmasov ADD FILE
(

```

```

NAME = N'new',
FILENAME = N'/home/db_user/mydb/new.ndf',
SIZE = 1 Mb, MAXSIZE = 15 Mb,
FILEGROWTH = 10%
) TO FILEGROUP new_filegroup

```

--устанавливаем новую файловую группу для использования по умолчанию
ALTER DATABASE Tolmasov MODIFY FILEGROUP new_filegroup DELAULT;

Теперь каждый созданный в базе данных объект будет попадать в файловую группу new_filegroup. Рассмотрим так же процесс изменения логического имени файла.

Следующая инструкция изменит логическое имя созданного ранее файла данных для базы данных *Tolmasov*, созданной ранее⁶:

```

ALTER DATABASE Tolmasov MODIFY FILE
(Name = Tolmasov, NEWNAME = N'Tolmasov_new');

```

Рассмотрим так же синтаксис добавления объекта в определённую файловую группу базы данных. Создадим новую таблицу, которую явно разместим в созданной выше файловой группе:

```

CREATE TABLE Person
(
    ID INT IDENTITY PRIMARY KEY,
    LastName NVARCHAR(30)
) ON new_filegroup

```

Грамотно используя механизм файловых групп, можно увеличить производительность баз данных и оптимизировать расход дискового пространства, выделяемого под файлы.

3.5 Контрольные задания

1. Создайте подключение к экземпляру сервера с помощью клиента Azure Data Studio.
2. Создайте подключение к экземпляру сервера используя DBEaver.
3. Изучите свойства системной базы данных *model*. Запомните размер файлов базы данных *model*.
4. Создайте пользовательскую базу данных *test1* используя минимальную синтаксическую конструкцию инструкции CREATE DATABASE.
5. С помощью запроса определите размер файлов созданной базы данных *test1*.

⁶ Следует отметить, что новый размер файла базы данных не может быть меньше, чем размер существующего файла.

6. Измените размер файлов базы данных *model* используя графический интерфейс SSMS.
7. Создайте базу данных *test2*. Узнайте размер файлов созданной базы данных. Сравните результаты.
8. Удалите базу данных *test2*.
9. Добавьте файл с начальным размером 10 Мб, максимальным в 128 МБ и приращением размера в 15% в базу данных.
10. Перенесите созданный файл базы данных в новое расположение.
11. Проверьте файловую группу файла. Если файл принадлежит к группе PRIMARY — удалите его.
12. Добавьте новую файловую группу и сделайте её группой по умолчанию.
13. Создайте новый файл базы данных с расположением отличным от расположения по умолчанию. Проверьте файловую группу нового файла.
14. Создайте таблицу с явным указанием файловой группы.
15. Заполните таблицу случайными данными. Наблюдайте за процессом наполнения файлов с помощью запроса к системной таблице *sys.database_files*. Сформулируйте вывод на основе наблюдений.
16. Перенесите файлы системной базы данных *tempdb* в новое расположение.
17. Найдите и приведите описание не менее двух хранимых процедур для работы с файлами базы данных (или просмотра информации об этих файлах).
18. Найдите информацию о переносе системной базы данных *master* в операционной системе Ubuntu Server.

4 УПРАВЛЕНИЕ ДОСТУПОМ

В СУБД *MS SQL Server* имеется встроенная система безопасности позволяющая защитить данные от несанкционированных действий. Система безопасности включает в себя механизмы авторизации, разграничения прав доступа, логического разделения хранимых данных и другие.

В этом разделе рассмотрены вопросы организации доступа к серверу, предопределенные роли, к которым могут быть присоединены пользователи, связь пользователей сервера с конкретными базами данных и процесс распределения прав на уровне баз данных, то есть будут рассмотрены три уровня модели безопасности: уровень сервера, уровень базы данных, уровень схемы.

4.1 Разрешение дополнительных соединений

Доступ на только что установленный экземпляр сервера обычно разрешен для двух пользователей: *sa*⁷ и пользователя Windows, который осуществлял установку экземпляра сервера. Но базы данных обычно используются множеством пользователей, которые часто не имеют возможности подключения к базе с компьютера, где установлен экземпляр. Кроме того, локальное подключение не является оптимальным при организации многопользовательского доступа.

Таким образом, перед администратором стоит задача по обеспечению доступа к серверу всех необходимых пользователей, для чего необходимо создать достаточное количество имён входа. В приведённых примерах будут рассматриваться создание имён входа SQL, на сервере установлен смешанный способ проверки подлинности (смена способа проверки подлинности рассматривается далее).

4.1.1 Создание имён входа

Авторизация пользователей на сервере осуществляется с помощью специальных субъектов — LOGIN (имя входа в русской версии MS SQL Server Management Studio). Документация MS SQL Server определяет имя входа следующим образом:

«Имя входа — это субъект безопасности, с помощью которого система безопасности может проверить подлинность лица или сущности» [6].

Имя входа даёт пользователям возможность подключаться к всему серверу

⁷ *sa* (*system administrator*) — учетная запись дающая право выполнения любых действий на сервере. Запись по умолчанию отключена из соображений безопасности. Активировать её можно при установке сервера, выбрав в качестве способа авторизации смешанный тип. *sa* по умолчанию включена для сервера устанавливаемого в ОС Ubuntu. После выполнения начальных настроек сервера и создания нового имени входа для администратора сервера, *sa* рекомендуется отключать (удалять это имя входа нельзя).

и каждое имя входа может быть наделено определёнными правами, тем не менее, первоначально не существует прямой связи между именем входа и базой данных.

Как и любой другой объект на сервере, имя входа может быть создано с помощью *MS SSMS* или средствами языка *Transact SQL*. Независимо от способа, пользователь, выполняющий действие должен иметь разрешения ALTER ANY LOGIN или ALTER LOGIN на уровне сервера.

Для примера создадим нового пользователя, который будет является администратором сервера и подлинность которого будет проверяться средствами SQL Server.

Создание имён входа с помощью MS SSMS

Для создания нового имени входа необходимо авторизоваться на сервере под именем входа, обладающего соответствующими разрешениями. После успешной авторизации необходимо раскрыть ниспадающий список нужного экземпляра сервера, в обозревателе объекта, нажав на «+» слева от имени сервера (рис. 4.1).

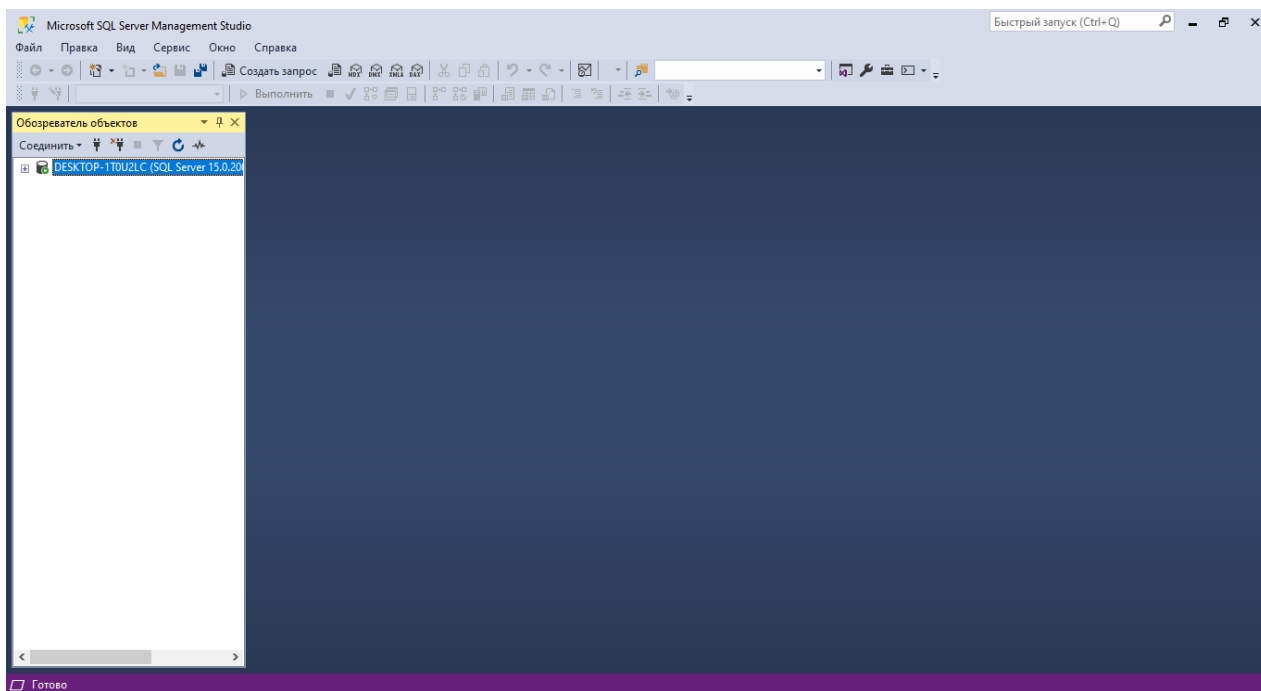


Рисунок 4.1 — Экран *SQL Server Management Studio* после авторизации

Имена входа относятся к субъектам безопасности сервера, следовательно их можно найти в разделе «**Безопасность**». Необходимо найти в обозревателе объектов папку «**Безопасность**» и нажать на ней правой кнопкой мыши. В открывшемся контекстном меню выбрать «**Создать / Вход...**» (рис. 4.2)

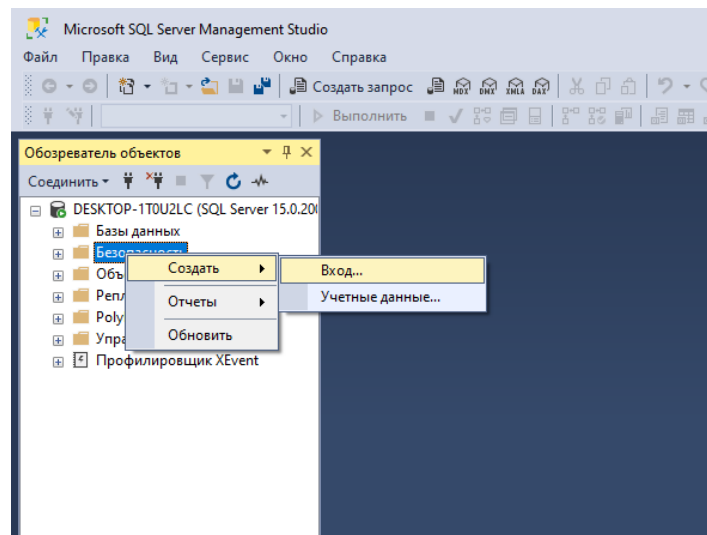


Рисунок 4.2 — Контекстное меню «Безопасность»

В результате откроется окно «Создание имени для входа» (Рис. 4.3).

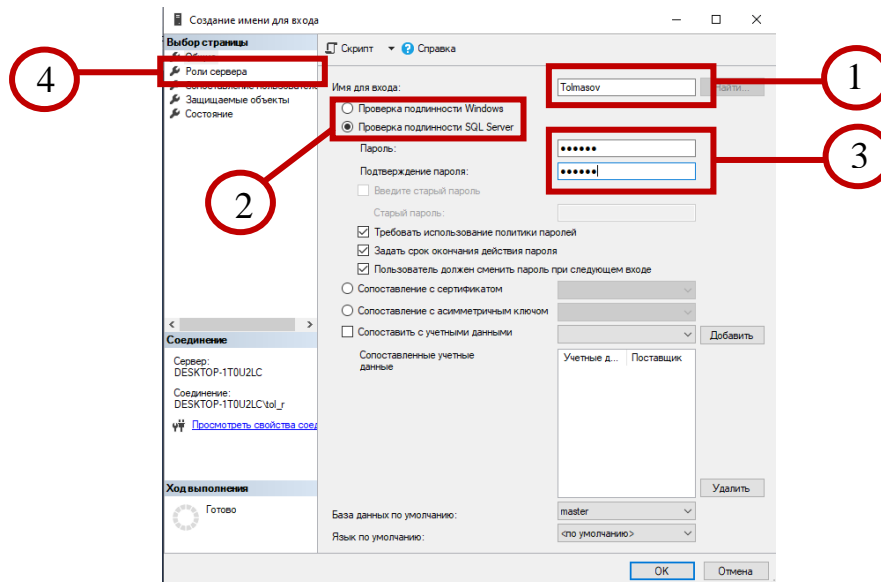


Рисунок 4.3 — Диалоговое окно создания имени входа

Для создания имени входа, с требуемыми параметрами, необходимо ввести уникальный (среди имён входа данного экземпляра сервера) идентификатор, соответствующий правилам присвоения имен (1). Выбрать вариант способа проверки подлинности. Для имён входа, выдаваемых людям, применяется два основных способа: «Проверка подлинности Windows» и «Проверка подлинности SQL Server». Так как по условию необходимо проверить подлинность паролем, то выбираем второй вариант (2). В ставших доступными полях ввода (3), вводим пароль дважды. Остальные настройки оставим без изменения.

Далее переходим к выбору ролей, в которых созданное имя входа будет числиться. Для этого переходим на страницу «Роли сервера» (4).

На этой странице необходимо выбрать роль sysadmin (Рис. 4.4)

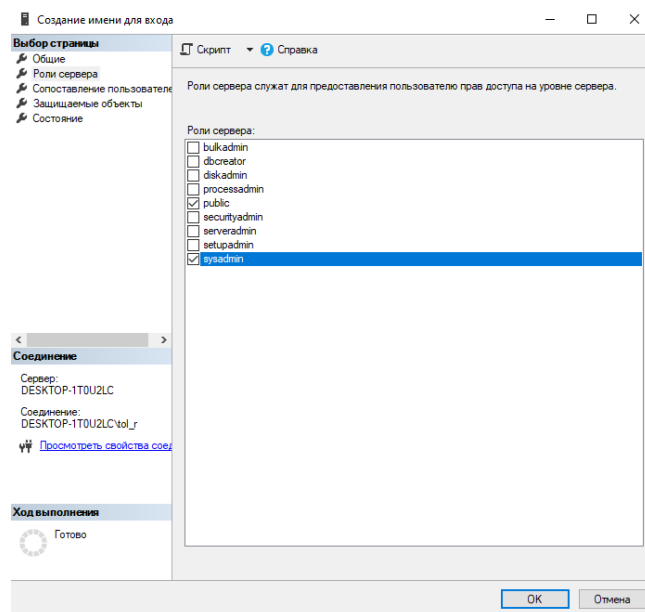


Рисунок 4.4 — Включение имени входа в роль администратора сервера

Для подтверждения введённых параметров и завершения создания субъекта безопасности следует нажать на кнопку «**Ок**».

Создание имени входа средствами Transact SQL

Решить задачу по созданию нового имени входа можно средствами языка Transact SQL. Для этого необходимо выполнить инструкцию:

```
CREATE LOGIN Tolmasov
WITH
    PASSWORD = '<password_word>' MUST_CHANGE,
    DEFAULT_DATABASE = master,
    CHECK_POLICY = ON,
    CHECK_EXPIRATION = ON
ALTER SERVER ROLE sysadmin ADD MEMBER Tolmasov
```

Выполнение данного скрипта приведёт к результатам, аналогичным полученным с помощью *MS SSMS*. Рассмотрим построчно свойства, заданные после ключевого слова WITH:

- **PASSWORD** — задаём способ проверки с помощью пароля, что автоматически определяет способ проверки подлинности средствами *SQL Server*. Свойство **MUST_CHANGE** указывает на необходимость смены пароля после первой авторизации на сервере.
- **CHECK_POLICY** — требовать использование политики паролей (определяет требования к сложности пароля).
- **CHECK_EXPIRATION** — включает контроль сроков смены пароля для создаваемого имени входа.

Следующая строка после создания имени входа, включает созданное имя в

состав роли sysadmin. Членство в роли public предоставляется автоматически.

4.1.2 Серверные роли

В каждом экземпляре сервера присутствует ряд заранее предопределённых ролей, к которым могут быть присоединены имена входа. Серверные роли определяют допустимый набор действий на сервере, которые может выполнять пользователь, авторизовавшийся под соответствующим именем. Предопределённые роли можно изменять, но нельзя удалять. В таблице ниже представлены все предопределённые роли и их краткое описание.

Таблица 3

Роль	Описание
sysadmin	Разрешено выполнять любые действия на сервере
dbcreator	Разрешено создавать БД
bulkadmin	Может выполнять BULK INSERT. Роль bulkadmin или разрешения ADMINISTER BULK OPERATIONS не поддерживаются в Linux. Операции массовой вставки на Linux может выполнять только sysadmin.
diskadmin	Позволяет управлять файлами на диске
processadmin	Позволяет управлять подключениями, запускать и приостанавливать экземпляры SQL Server
securityadmin	Создание и управление учётными записями, право на сброс пароля учётной записи, управление разрешениями на уровне сервера и на уровне баз данных (при наличии доступа к БД)
serveradmin	Члены роли могут изменять параметры конфигурации на уровне сервера, а также выключать сервер
setupadmin	Добавление и удаление связанных серверов с помощью инструкций T-SQL
public	Каждое имя входа принадлежит этой роли, но эта роль не даёт пользователям никаких прав.

Разрешения, предоставленные серверным ролям приведены на **рис. 4.5** [7]

SERVER LEVEL ROLES AND PERMISSIONS: 9 fixed server roles, 34 server permissions

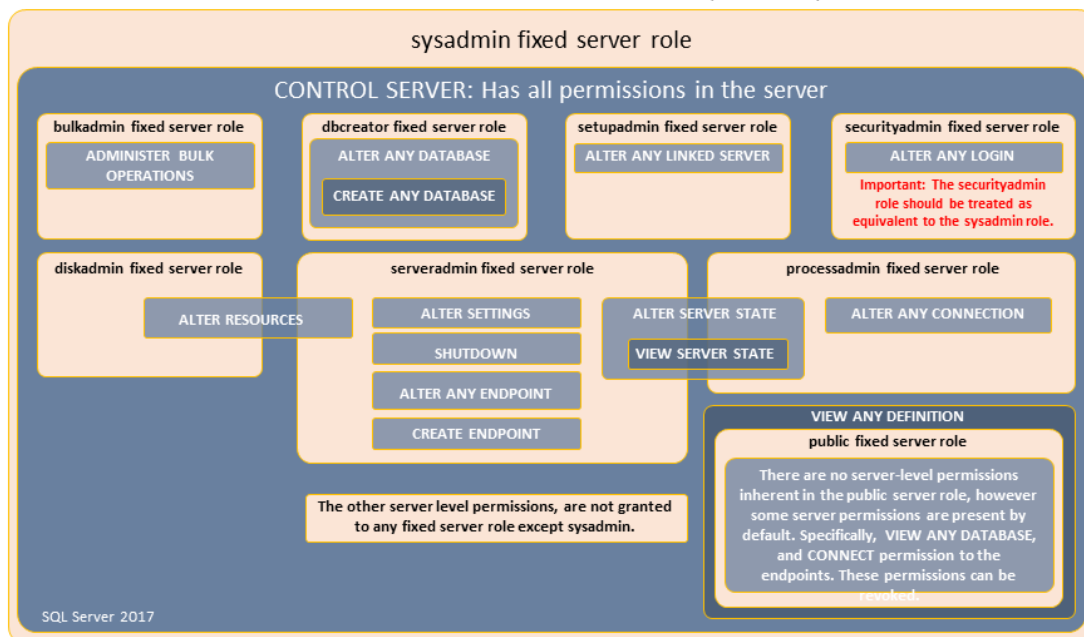


Рисунок 4.5 — Разрешения, предоставленные предопределённым ролям

На рис. 4.1 представлены 9 предопределённых ролей (светлые прямоугольники со скруглёнными углами) и 34 серверных разрешения (темные прямоугольники с прямыми углами). Если прямоугольник накладывается на соответствующую роль (вложен или пересекает его границу), то члены данной роли имеют соответствующее разрешение. Ситуации вложенных друг в друга решений трактуются так: если предоставлено внешнее разрешение, то автоматически предоставляются и все вложенные в него. Аналогично и с вложенностью ролей. Таким образом наглядно продемонстрировано, что роль **sysadmin** имеет все возможные разрешения и имеет полный контроль над сервером (**CONTROL SERVER**).

Серверные роли, определяемые пользователем

При необходимости пользователь может определять собственные серверные роли. Рассмотрим синтаксис инструкции для создания ролей определяемых пользователем:

```
CREATE SERVER ROLE role_name [ AUTHORIZATION server_principal ]
```

Ключевое слово **AUTHORIZATION** и следующее за ним имя входа определяет принадлежность роли. То есть можно явно указать какому из существующих имён входа принадлежит роль. Можно не указывать принципала явно, в этом случае роль будет принадлежать имени входа, создавшего роль.

Создадим роль **test**, владельцем которой назначим имя входа **sa**:

```
CREATE SERVER ROLE test AUTHORIZATION sa
```

Данному типу ролей можно давать разрешения уровня сервера (рис. 4.5) с

помощью инструкций GRANT, DENY и REVOKE. Общий синтаксис команды GRANT имеет вид:

```
GRANT permission [ ,...n ]  
  TO <grantee_principal> [ ,...n ] [ WITH GRANT OPTION ]  
  [ AS <grantor_principal> ]
```

Предоставим новой роли право создавать базы данных и запретим выключать сервер, для этого необходимо выполнить следующие инструкции:

```
CREATE SERVER ROLE test AUTHORIZATION sa  
GRANT CREATE ANY DATABASE TO test  
DENY SHUTDOWN ON SERVER::dbserver TO test
```

Для удаления созданной серверной роли следует воспользоваться инструкцией DROP языка *DDL*:

```
DROP SERVER ROLE Test
```

4.2 Управление доступом к данным

4.2.1 Пользователи базы данных

Пользователь с именем входа имеет право на доступ к экземпляру сервера, но в большинстве случаев серверные роли не предоставляют никаких прав для взаимодействия с объектами внутри баз данных. Кроме того, система разрешений внутри баз данных немного отличается от серверной и в первую очередь направлена на защиту данных, хранящихся в базах.

При проектировании баз данных, их разработке и развертывании строго привязываться к именам входа не удобно, так как они с высокой степенью вероятности отличаются от сервера к серверу. Но при работе с базой данных каждое действие должно быть выполнено от имени конкретного пользователя, разрешения которого проверяются перед каждым его действием. Для решения этой проблемы внутри базы данных существуют собственные субъекты безопасности — «*Пользователь*»⁸ (*USER*). Пользователи баз данных соотносятся с конкретным именем входа, обеспечивая таким образом возможность ассоциации внешних пользователей с необходимыми базами данных. Данная взаимосвязь наглядно представлена в технической документации MS SQL Server (рис. 4.6).

⁸ Так как ранее мы использовали термин «*Пользователь*» для обозначения человека применяющего экземпляр сервера, для обозначения субъекта безопасности базы данных будет употребляться «*Пользователь базы данных*».

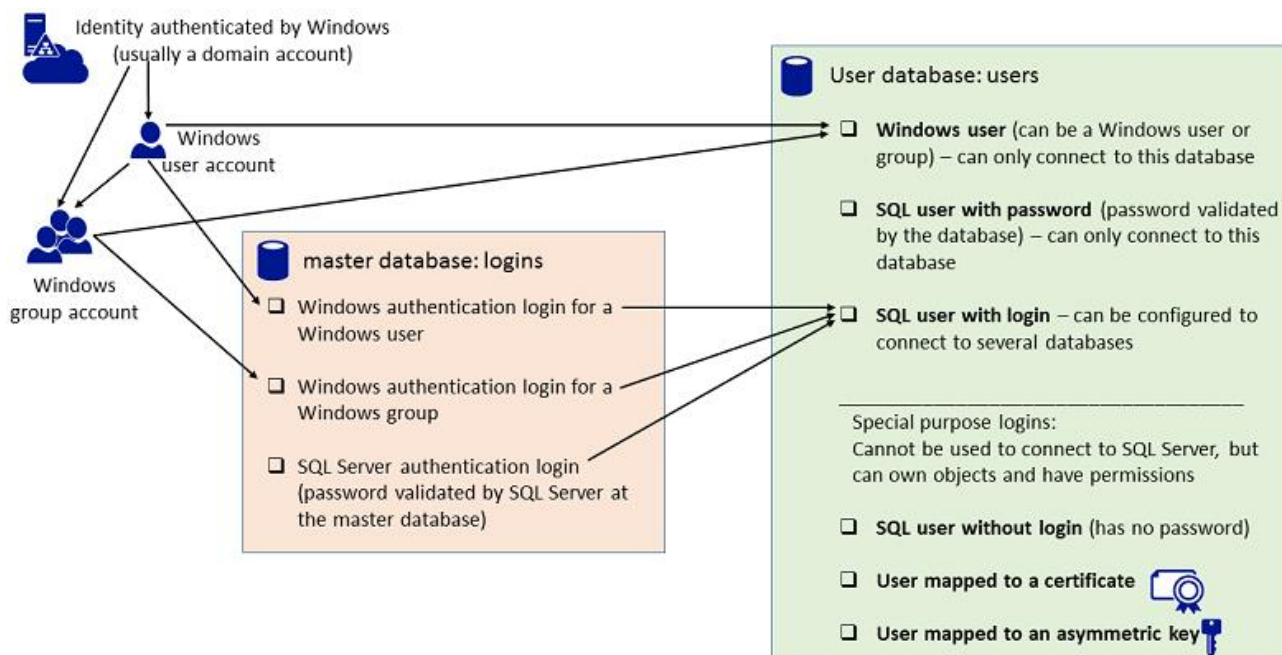


Рисунок 4.6 — Варианты связи имени входа и пользователя базы данных

На рис. 4.6 приведены варианты связи внешних пользователей и пользователей базы данных. Анализируя рисунок можно сделать следующие выводы:

- 1) Информация об именах входа, которые имеют доступ к серверу, хранится в системной базе данных master.
- 2) Доступ к конкретной базе данных может быть предоставлен авторизованным пользователям Windows или группе пользователей, без предоставления доступа к серверу.
- 3) Существуют специальные способы доступа к конкретной базе данных, которые используются для предоставления доступа к данным для приложений: доступ без пароля, использование сертификатов или ассиметричные ключи.

Далее мы рассмотрим создание пользователя базы данных ассоциированного с созданным ранее именем входа и сделаем его владельцем базы данных. Для этого необходимо иметь разрешение ALTER ANY USER в соответствующей базе данных. В нашем случае это может быть имя входа, которое было создано при установке экземпляра сервера.

Создание пользователя базы данных с помощью MS SSMS

Авторизуйтесь на сервере именем входа, созданным при установке сервера. Разверните список объектов нужного экземпляра сервера (рис.4.7) и перейдите в папку «**Базы данных**».

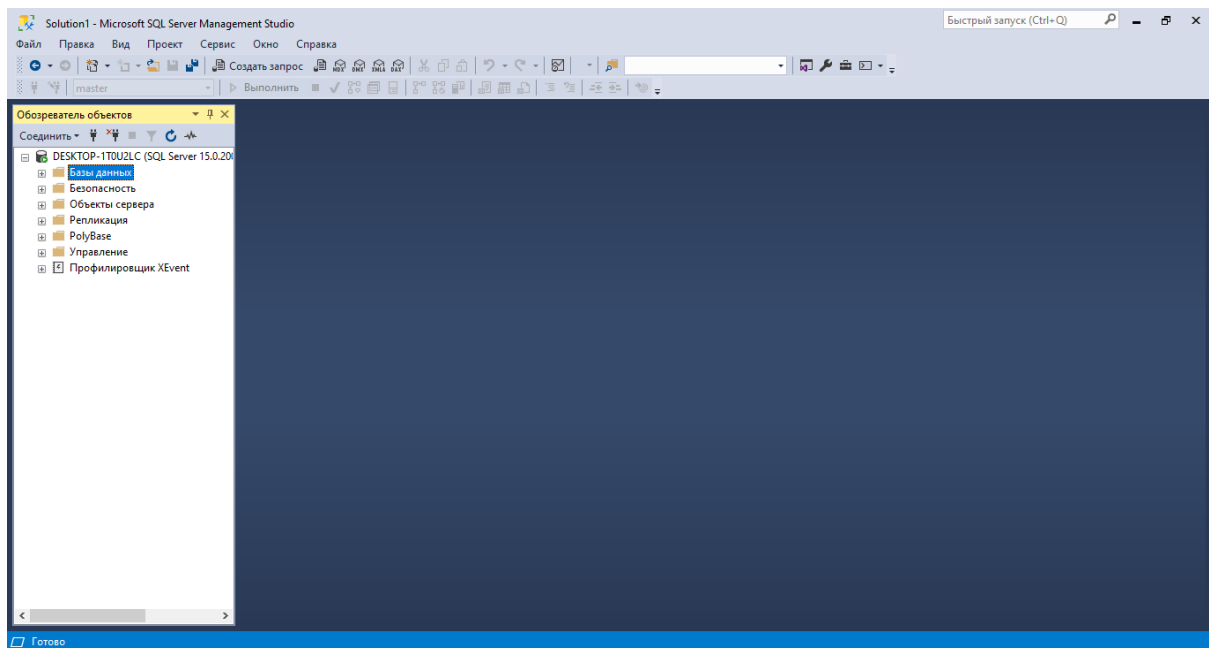


Рисунок 4.7 — Состояние SQL Server Management Studio перед началом процесса

Найдите в списке базу данных, для которой требуется создать пользователя и разверните список объектов базы данных, нажав на «+» слева от названия базы данных. Щелкните правой кнопкой мыши по папке «**Безопасность**» (Рис. 4.8) базы данных. В развернутом контекстном меню необходимо выбрать «**Создать / Пользователь...**».

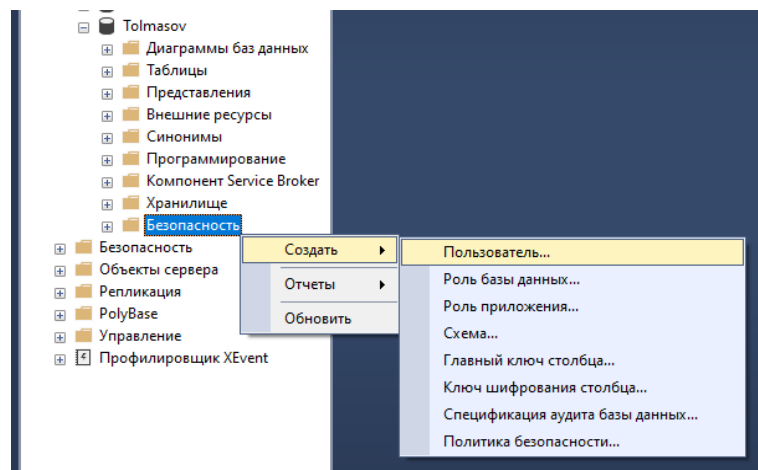


Рисунок 4.8 — Открытие диалогового окна создания пользователя базы данных

В диалоговом окне необходимо обязательно заполнить поля «**Имя пользователя**» и «**Имя для входа**» (Рис. 4.9).

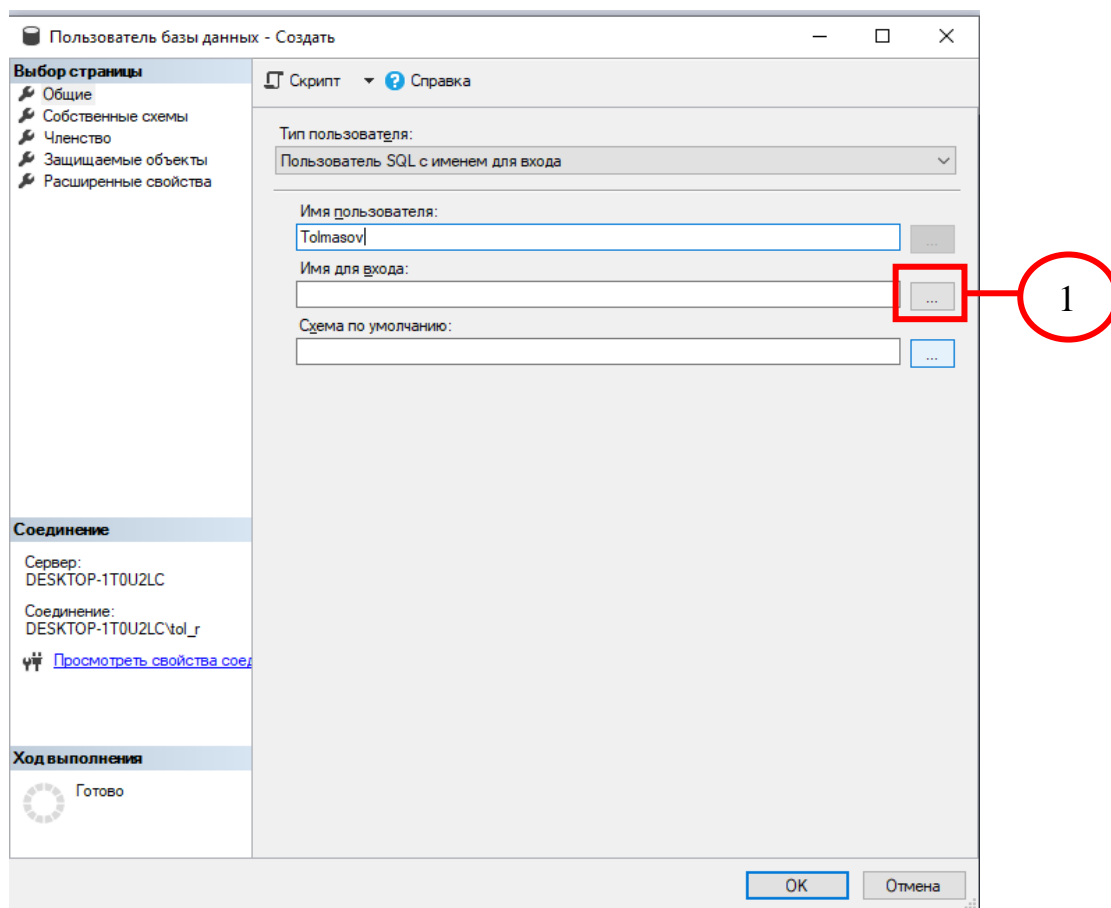


Рисунок 4.9 — Диалоговое окно создания пользователя базы данных

Имя для входа лучше выбрать с помощью диалогового окна, чтобы избежать ошибок в заполнении поля (особенно при использовании имен входа, ассоциированных с учётной записью или группой учетных записей Windows). Для вызова диалогового окна **«Выбор имени для входа»** нажмите на кнопку «...» справа от соответствующего поля ((1) Рис. 4.9).

В окне **«Выбор имени для входа»** нажмите на кнопку **«Обзор»** (Рис. 4.10)

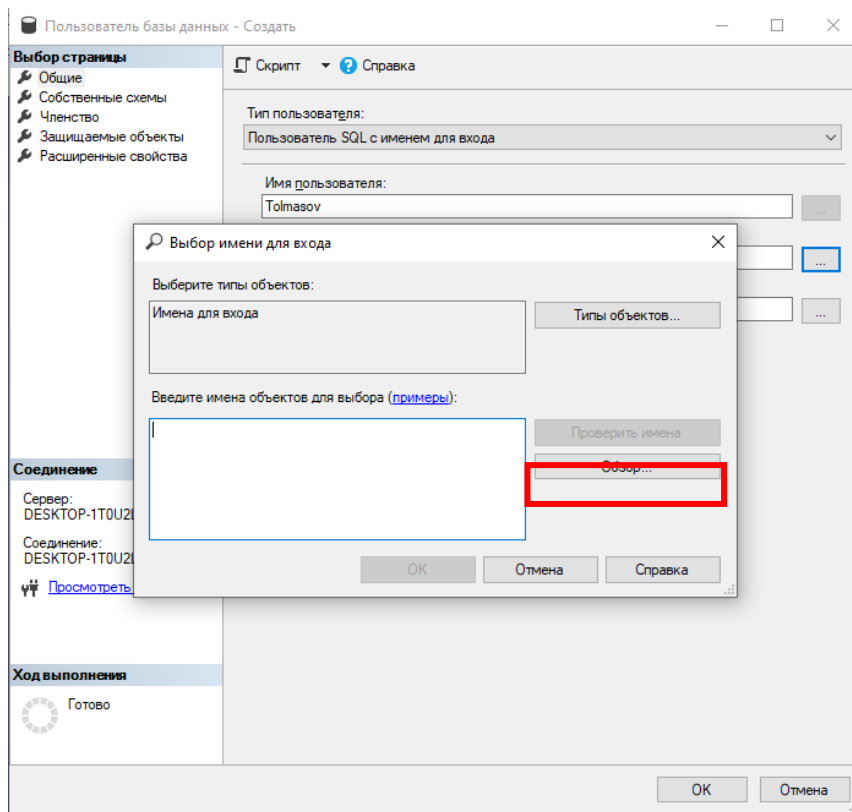


Рисунок 4.10 — Выбор имени для входа

В следующем диалоговом (Рис. 4.11) окне необходимо найти созданное ранее имя входа (1) и выбрать его. После этого нажать «**Ок**» (2).

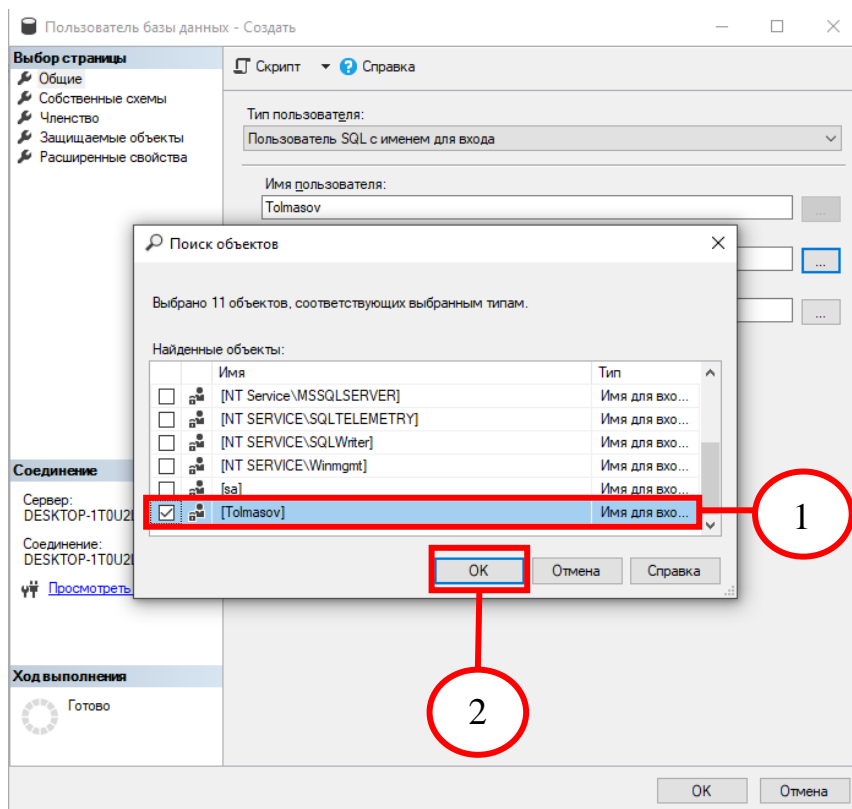


Рисунок 4.11 — Поиск созданного ранее имени входа

Схему по умолчанию можно оставить без изменений. Следующим шагом

необходимо предоставить все права на эту базу данных для пользователя. Чтобы сделать это нужно перейти на страницу «Членство» (Рис. 4.12).

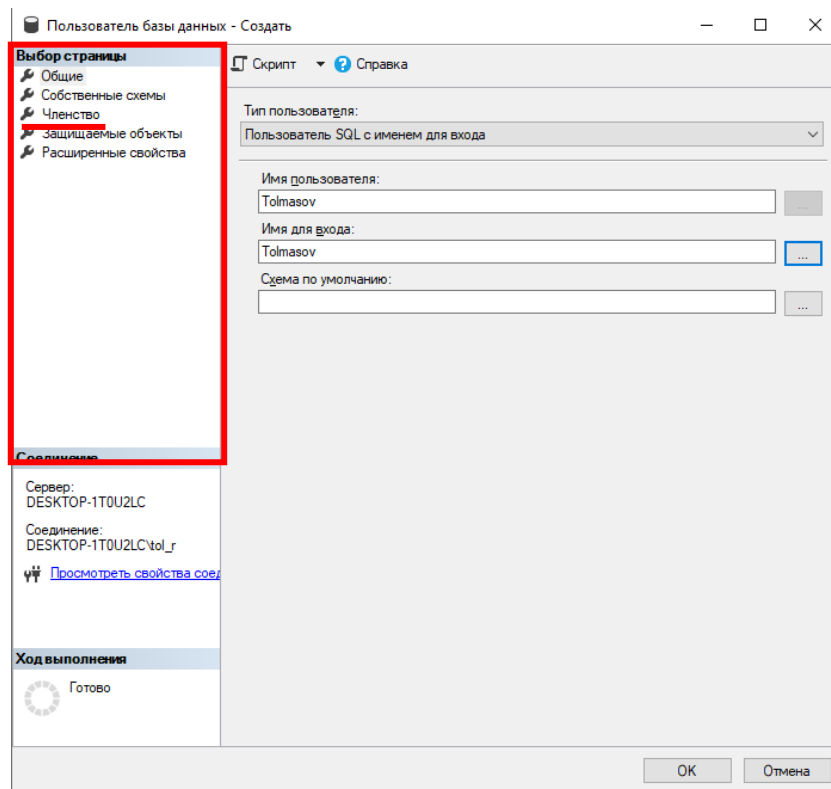


Рисунок 4.12 — Переход к странице предоставления членства в роли

На рис. 4.13 приведена страница «Членство в роли базы данных».

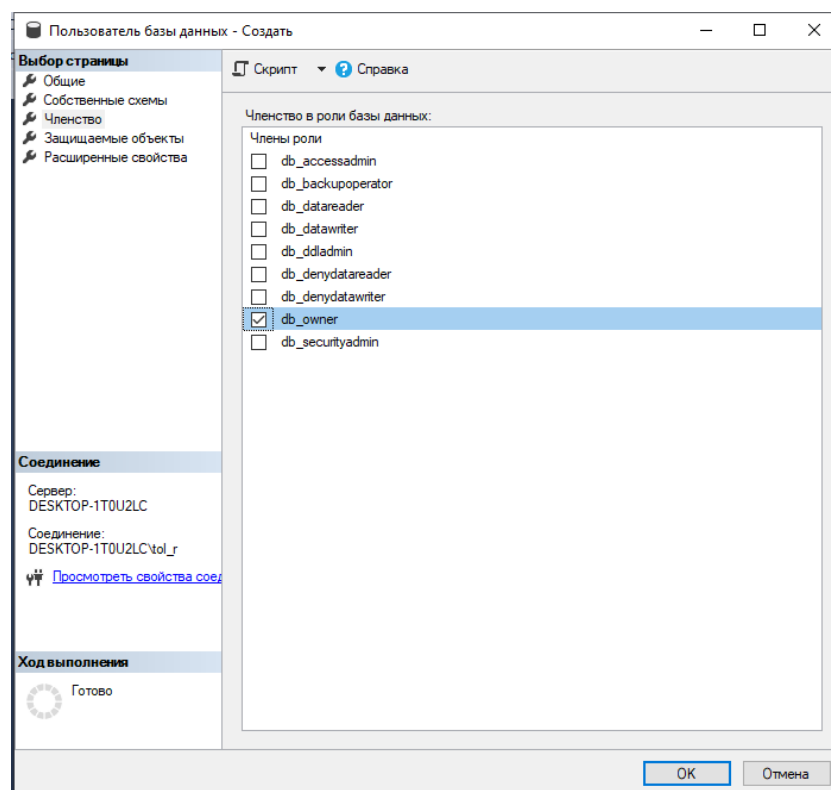


Рисунок 4.13 — Включение пользователя базы данных в роли базы данных

Выбираем роль *db_owner*⁹ (владелец базы данных). Членство в роли позволит пользователю совершать любые действия над базой, в том числе и удалять её, но не гарантирует ему возможность создания базы данных заново. Выбрав нужную роль, можно подтвердить создание нового пользователя нажав «**Ок**» в правом нижнем углу формы.

Создание пользователя базы данных средствами Transact SQL

Аналогичные действия можно выполнить с помощью скрипта. Процесс состоит из последовательного выполнения нескольких инструкций:

- 1) Перейти в контекст базы данных, где будет создаваться пользователь базы данных.
- 2) Выполнить инструкцию CREATE USER и указать имя пользователя в соответствии с правилами наименования сервера и связать его с именем входа указав нужное имя после FOR LOGIN. Данный синтаксис инструкции применим только к именам входа с проверкой подлинности SQL Server.
- 3) Предоставить пользователю членство в роли *db_owner* с помощью инструкции ALTER ROLE.

Следовательно, скрипт, выполняющий те же действия, что были выполнены в MS SSMS будет выглядеть следующим образом:

```
USE Tolmasov
GO
CREATE USER Tolmasov FOR LOGIN Tolmasov
ALTER ROLE db_owner ADD MEMBER Tolmasov
```

Более подробно с инструкцией создания пользователя можно ознакомиться в соответствующем разделе официальной документации [9]

4.2.2 Роли базы данных

Для каждой базы данных, как и для экземпляра сервера, существуют предопределённые роли. Роли позволяют удобно и эффективно распределять права на выполнение требуемых действий между пользователями. Изменения в предопределённые роли вносить нельзя, но существует риск непреднамеренного расширения предоставленных им прав. Во избежание подобных ситуаций рекомендуется избегать добавления пользовательских ролей в качестве членов предопределённых ролей.

Рассмотрим предопределённые роли баз данных для удобства сведя информацию о них в таблицу:

⁹ Роли базы данных отличаются от серверных ролей. Их назначение и процесс создания пользовательских ролей будет рассмотрено далее.

Таблица 4

Роль	Описаниеы
<i>db_owner</i>	Обладают полными правами на базу данных. Могут выполнять все действия по настройке и обслуживанию базы данных, а также удалять базу данных
<i>db_securityadmin</i> ¹⁰	Могут изменять членство в роли (только для настраиваемых ролей) и управлять разрешениями.
<i>db_accessadmin</i>	Могут добавлять или удалять права удаленного доступа к базе данных для имен входа и групп <i>Windows</i> , а также имен входа <i>SQL Server</i>
<i>db_backupoperator</i>	Могут создавать резервные копии базы данных
<i>db_ddladmin</i>	Могут выполнять любые команды языка определения данных (<i>DDL</i>) в базе данных
<i>db_datawriter</i>	Могут добавлять, удалять или изменять данные во всех пользовательских таблицах
<i>db_datareader</i>	Могут считывать все данные из всех пользовательских таблиц и представлений
<i>db_denydatawriter</i>	Не могут добавлять, удалять или изменять данные во всех пользовательских таблицах
<i>db_denydatareader</i>	Не могут считывать все данные из всех пользовательских таблиц и представлений

Несмотря на общее сходство предопределенных ролей баз данных и серверных ролей, роли баз данных более направлены на разграничение прав доступа к данным и ограничения манипуляции с ними.

4.2.3 Схемы базы данных

Ранее понятие «Схема» уже упоминалось, но до настоящего момента никакой характеристики объекту не давалось. Определим это понятие:

Схема (SCHEMA) — это коллекция объектов базы данных, имеющая одного владельца и формирующая одно пространство имен.

Схемы, как один из объектов, используемых в модели безопасности SQL Server. Тем не менее схема является важным понятием в контексте управления доступом к данным, так как позволяет скрыть часть базы данных от определённых пользователей, групп пользователей или ролей баз данных.

Рассмотрим процесс создания схем средствами языка T-SQL.

Каждая новая схема должна принадлежать одному из участников уровня баз данных: пользователю баз данных, роли базы данных или приложению. Для создания схем, создающий должен обладать разрешением CREATE SCHEMA, а при необходимости назначения владельцем схемы другого пользователя, у создателя должно быть разрешение IMPERSONATE на этого пользователя. Если владельцем схемы определяется роль, то вызывающий схему объект должен либо иметь

¹⁰ Необходимо внимательно следить за элементами входящими в состав этой роли, так как они потенциально могут повышать свои права доступа.

членство в этой роли, либо иметь разрешение ALTER ROLE.

Инструкция создания схемы имеет следующий вид:

```
CREATE SCHEMA schema_name_clause [ <schema_element> [ ...n ] ]
```

```
<schema_name_clause> ::=
```

```
{  
  schema_name  
  | AUTHORIZATION owner_name  
  | schema_name AUTHORIZATION owner_name  
}
```

```
<schema_element> ::=
```

```
{  
  table_definition | view_definition | grant_statement |  
  revoke_statement | deny_statement  
}
```

Следует сделать ряд замечаний по оформлению и выполнению инструкции:

- 1) Объявление схемы должно быть единственной инструкцией в пакете (находящиеся в схеме защищаемые объекты и настройку прав доступа ограничение не распространяется).
- 2) В состав инструкции входит все что указано между CREATE SCHEMA и ближайшим GO или концом файла.
- 3) Транзакция CREATE SCHEMA является атомарной, если в процессе выполнения произойдет ошибка при создании одного из защищаемых объектов, то не будет создан ни один объект, описанный в инструкции.
- 4) Объекты в схеме можно объявлять в произвольном порядке, то есть создание представления можно выполнить раньше, чем описать таблицу, исключением является представление, ссылающееся на другое представление.
- 5) Порядок объявления GRANT, DENY и REVOKE играет роль, так как будут применяться в порядке их появления, но они могут предоставлять разрешения на доступ к объекту до того, как он будет объявлен.

Создадим схему в базе данных *Tolmasov* с двумя таблицами *test1* и *test2* в каждой по одному целочисленному столбцу. Владелец схемы будет назначен автоматически. Для демонстрации синтаксиса объявления продемонстрируем выдачу прав на защищаемые таблицы созданному ранее пользователю базы данных.

Для таблицы *test1* предоставим пользователю базы данных все права, кроме возможности вставки данных в таблицу, а для таблицы *test2* наложим запрет на

выборку.

```
GO
CREATE SCHEMA schema_test
CREATE TABLE test1
(
    C1 INT
)
CREATE TABLE test2
(
    C2 INT
)
DENY SELECT ON test2 TO Tolmasov
GRANT SELECT, UPDATE, DELETE ON test1 TO Tolmasov
DENY INSERT ON test1 TO Tolmasov
GO
```

Права можно также предоставлять и ролям.

Для удаления схемы необходимо предварительно удалить все защищаемые ей объекты. После чего использовать инструкцию *DDL* DROP SCHEMA.

4.3 Контрольные задание

1. Создайте два новых имени входа, имеющих членство в разных серверных ролях с помощью *MS SSMS*.
2. Создайте два имени входа средствами *T-SQL*. Предоставьте им членство в разных предопределённых серверных ролях.
3. Попробуйте выполнить следующие действия используя разные имена входа: создать базу данных, изменить базу данных, создать новое имя входа, включить новое имя входа в состав серверной роли (на выбор). Составьте таблицу, где зафиксируйте удавшиеся и неудавшиеся действия.
4. Создайте серверную роль и предоставьте ей разрешения на создание баз данных и изменение имен входа.
5. Предоставьте членство в созданной серверной роли для любого созданного ранее имени входа.
6. Исключите имя входа из состава других ролей (кроме *public*).
7. Удалите созданную серверную роль.
8. Удалите одно из созданных ранее имён входа.
9. Для каждого оставшегося имени входа создайте по одному пользователю базы данных с разным набором свойств. Если база данных не существует, то предварительно создайте её создайте.

10. Создайте пользовательскую роль уровня баз данных. Предоставьте ей три разрешения.
11. Предоставьте пользователю баз данных членство в созданной роли.
12. Авторизуйтесь именем входа, связанного с пользователем, принадлежащим к созданной роли базы данных. Проверьте правильность работы прав доступа.
13. Удалите пользователя баз данных.
14. Создайте схему владельцем которой назначьте созданную ранее роль базы данных. В состав схемы включите три таблицы и два представления. Назначьте права доступа на защищаемые объекты для разных пользователей.
15. Переназначьте владельца созданной схемы.
16. Проверьте работу механизмов безопасности на уровне схем.
17. Удалите схему баз данных.
18. Найдите информацию об инструкциях GRANT, DENY, REVOKE, опишите их назначение и назначение параметров.
19. Предоставьте членство двум пользователям в роли баз данных db_owner. Запретите одному из пользователей применять инструкции *DDL*. Проверьте правильность выполнения инструкции.
20. Удалите все созданные в ходе заданий объекты. Оставьте на сервере только одну пользовательскую базу данных¹¹

¹¹ Имеются ввиду базы данных созданных в процессе выполнения упражнения, если на сервере есть и используются другие базы данных — удалять их не нужно

5 РЕЗЕРВНОЕ КОПИРОВАНИЕ И ВОССТАНОВЛЕНИЕ БАЗ ДАННЫХ

Резервное копирование и восстановление играет важнейшую роль в обеспечении надёжной защиты и хранения данных. Резервное копирование полностью определяется планом (стратегией) резервного копирования, который должен быть определён исходя из нужд конкретной организации, а также регламентам, устанавливающий порядок действий при исполнении стратегии резервного копирования. Хорошо спроектированная стратегия резервного копирования защитит базы данных от потери данных в случае различных сбоев.

В данном разделе рассматриваются технические аспекты реализации процедуры резервного копирования и восстановления баз данных и смежные понятия, необходимые для успешного выполнения данной процедуры.

5.1 Модели восстановления

5.1.1 Виды моделей восстановления

Резервное копирование тесно связано с *моделью восстановления*¹² базы данных. Модель восстановления предназначена для управления обслуживанием журналов транзакций. В документации к MS SQL Server даётся следующее определение модели восстановления:

«*Модель восстановления* — это свойство базы данных, которое управляет процессом регистрации транзакций, определяет, требуется ли для журнала транзакций резервное копирование, а также определяет, какие типы операций восстановления доступны» [9].

СУБД поддерживает три модели восстановления: простая, полная и модель восстановления с неполным протоколированием. Краткая характеристика моделей приводится ниже.

Простая модель восстановления характеризуется в первую очередь отсутствием резервных копий журналов транзакций. Это накладывает некоторые ограничения на возможности восстановления из резервных копий, например, невозможно восстановление на определённый момент времени. При использовании этой модели, в процессе резервного копирования происходит автоматическое освобождение места на диске, занятого другими журналами, что позволяет отказаться от ручного управления размеров журналов транзакций. При

¹² Не смотря на название «модель восстановления», данное свойство определяется при создании базы данных и влияет не столько на процесс восстановления, сколько на процесс регистрации транзакций. Оно так же влияет на перечень доступных вариантов резервных копий. Поэтому изучение вопросов резервного копирования и восстановления следует начать именно с определения этого свойства.

использовании простой модели восстановления изменения, произошедшие с момента создания последней резервной копии, не защищаются и будут потеряны в случае возникновения сбоя, восстановление этих данных придётся производить повторным внесением изменений.

Полная модель восстановления базы данных отличается от простой в первую очередь строгим протоколированием всех производимых операций с базой данных в журнале транзакций. С одной стороны это обстоятельство приводит к резкому увеличению занимаемого журналом транзакций места на жестком диске, но с другой — позволяет откатывать базы данных в состояние до определённого момента времени, например, до момента сбоя приложения или ошибки пользователей. Возможность восстановления в произвольный момент времени достигается за счёт необходимости создания резервной копии журнала транзакций. Используя данную модель восстановления практически исключена, потеря результатов работы из-за повреждения файлов данных.

Рост объёма физического файла журнала транзакций может быть существенной проблемой, особенно в случаях, когда часто осуществляются операции массового копирования. Для решения этой проблемы можно ограничить операции, регистрируемые в журнале транзакций, выбрав в качестве модели восстановления модель с неполным протоколированием.

Используя модель с неполным протоколированием, администратор может добиться уменьшения занимаемого журналами дискового пространства, так как к большинству массовых операций будет применено минимальное протоколирование. Список операций, поддерживающих минимальное протоколирование, приведён в разделе «Журнал транзакций (SQL Server)» официальной документации [9]. Данная модель не поддерживает восстановление до заданной временной отметки. Опытные администраторы стараются не применять эту модель на постоянной основе, включая ее только перед выполнением массовых операций.

Смена модели для базы данных может быть произведена в любой момент, при наличии у пользователя базы данных разрешения ALTER DATABASE. Смену модели можно произвести как при помощи графического интерфейса MS SQL Server Management Studio, так и с помощью средств языка T-SQL, оба способа приведены ниже в соответствующих разделах.

5.1.2 Смена модели восстановления с помощью MS SSMS

Для смены модели восстановления базы данных необходимо выполнить следующую последовательность действий:

1. Нажатием правой кнопкой мыши на нужной базе данных вызвать

контекстное меню и выбрать пункт «**Свойства**» (Рис. 5.1)

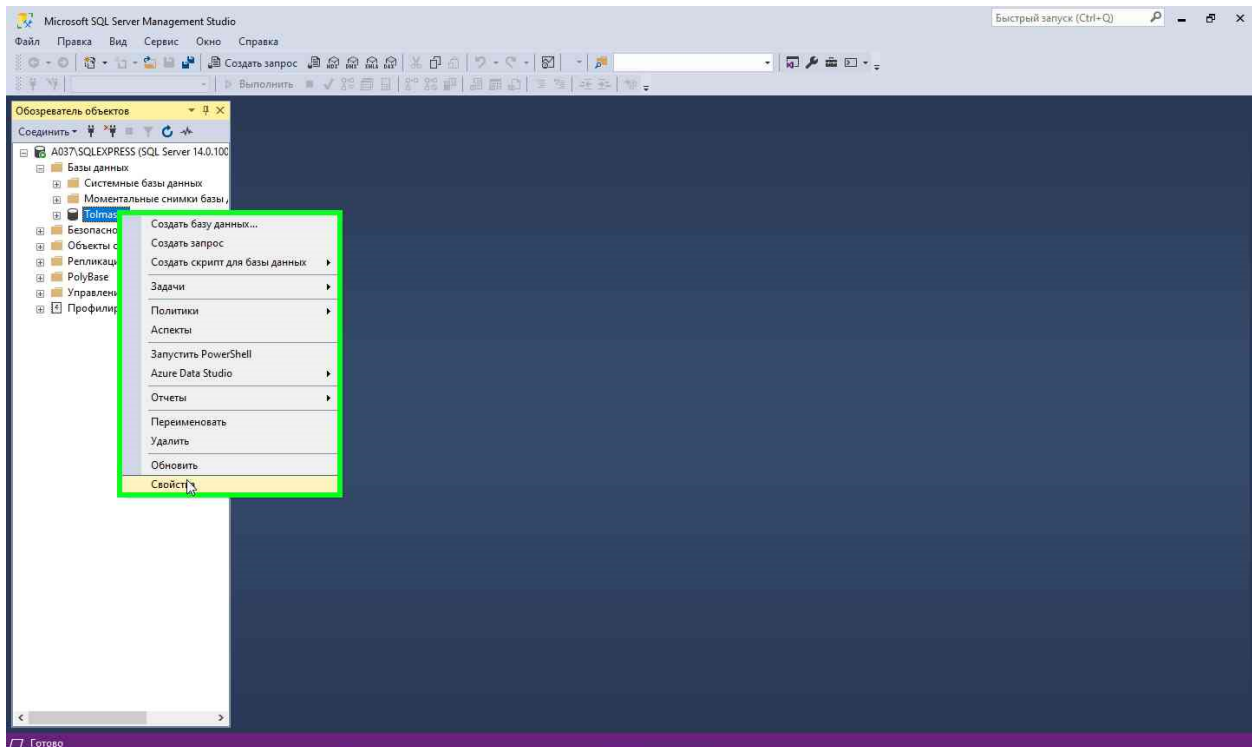


Рисунок 5.1 — Контекстное меню базы данных

2. В открывшемся окне, в разделе «Выбор страницы», выбрать пункт «**Параметры**» (Рис. 5.2).

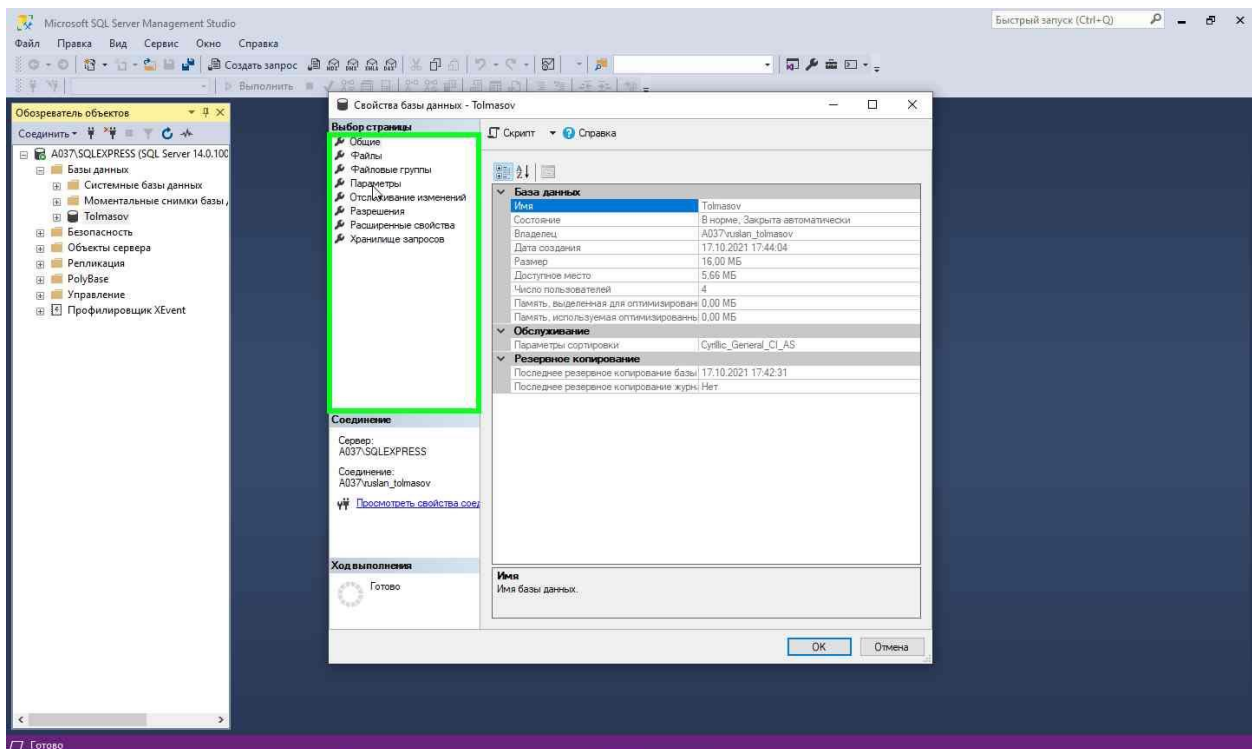


Рисунок 5.2 — Выбор страницы свойств базы данных

3. Найти поле «**Модель восстановления**» (Рис. 5.3).

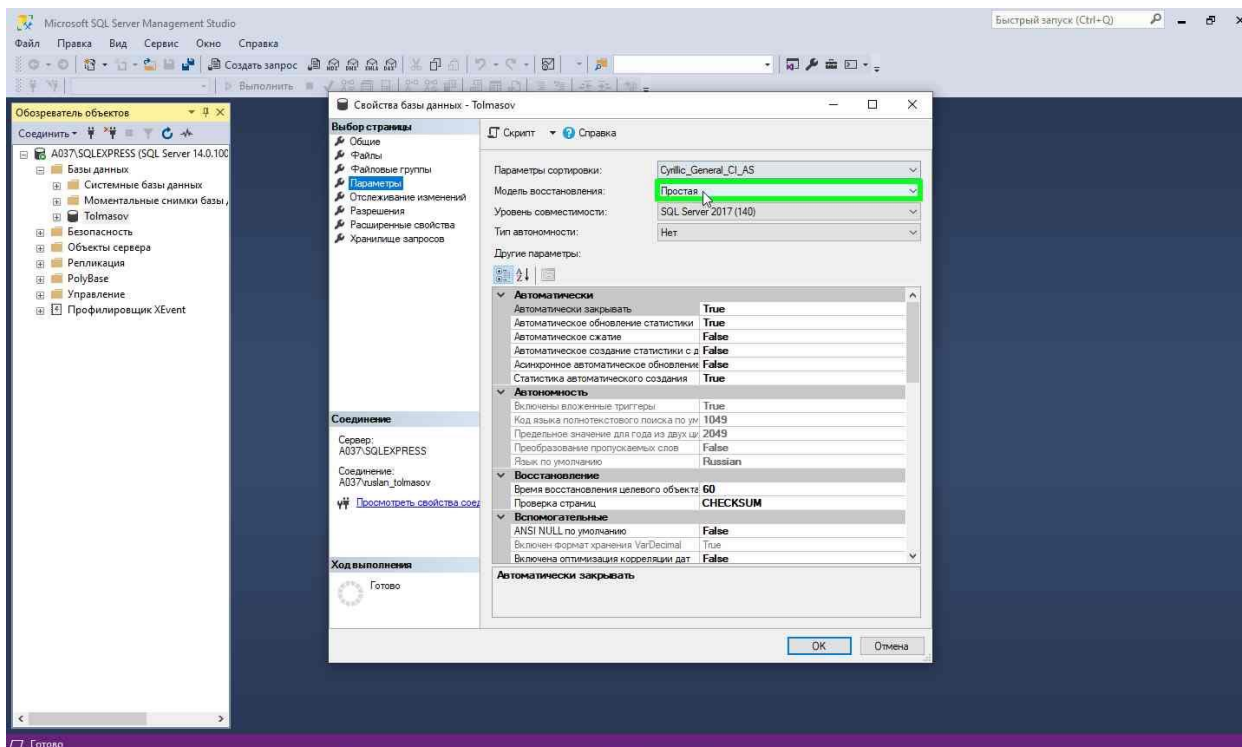


Рисунок 5.3 — Параметры базы данных

4. Выбрать из списка необходимую модель восстановления (Рис. 5.4).

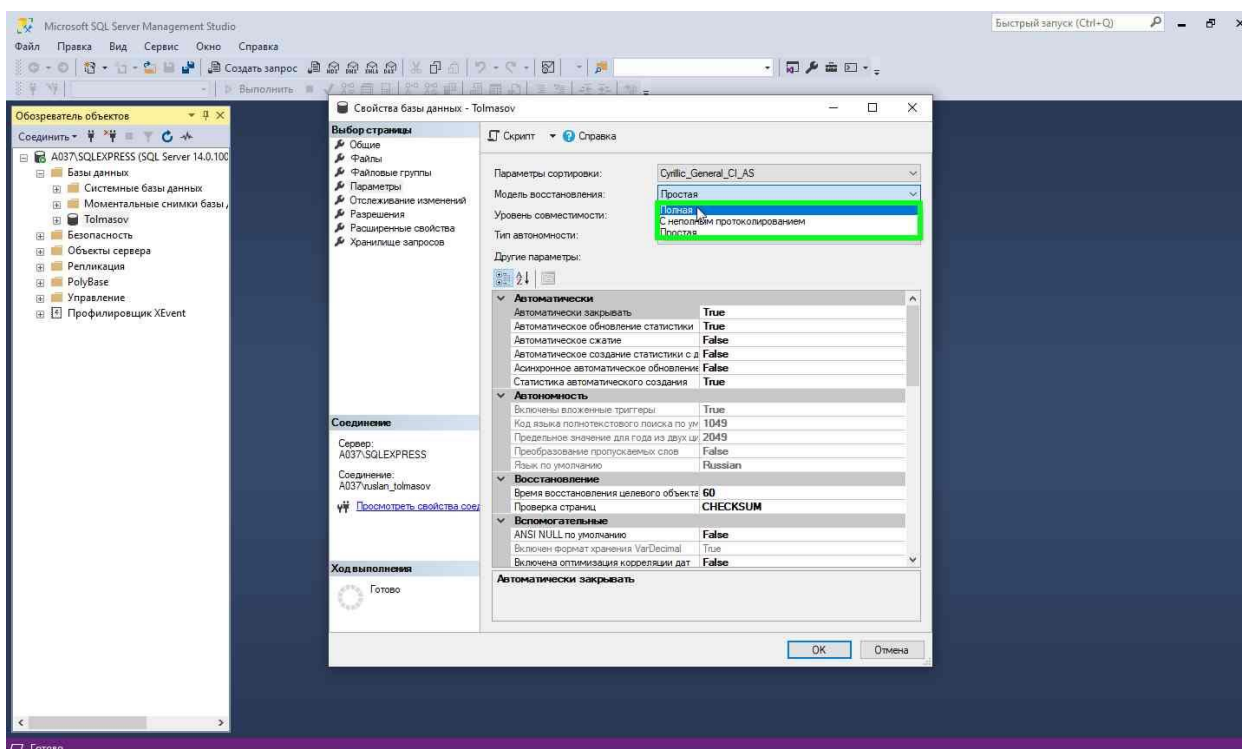


Рисунок 5.4 — Смена модели восстановления

5. Нажать кнопку «Ok», чтобы изменения вступили в силу (Рис.5.5).

МОМЕНТ СВОЙСТВА: МОДЕЛИ ВОССТАНОВЛЕНИЯ:

```
<recovery_option> ::=  
{  
    RECOVERY { FULL | BULK_LOGGED | SIMPLE }  
    | TORN_PAGE_DETECTION { ON | OFF }  
    | PAGE_VERIFY { CHECKSUM | TORN_PAGE_DETECTION | NONE }  
}
```

В приведённом выше определении объекта собраны свойства отвечающие за восстановление базы данных. Для изменения модели восстановления достаточно использовать только верхнюю строчку. Приведём пример скрипта изменения модели восстановления созданной ранее базы данных *Tolmasov*:

```
ALTER DATABASE Tolmasov  
SET  
    RECOVERY FULL
```

Подробное описание других свойств баз данных приведено на странице официальной документации SQL Server 2019.

5.2 Резервное копирование базы данных

5.2.1 Оценка размера файла резервной копии

Перед началом процедуры резервного копирования полезно оценить предполагаемый размер копии с помощью хранимой процедуры `sp_spaceused`, которая позволяет определить количество строк, зарезервированный и используемый на объём памяти используемое объектом базы данных (таблицей или представлением) или всей базой данных. Приведём сигнатуру вызова процедуры:

```
sp_spaceused [ [ @objname = ] 'objname' ]  
[ [ @updateusage = ] 'updateusage' ]  
[ [ @mode = ] 'mode' ]  
[ [ @onerresultset = ] onerresultset ]  
[ [ @include_total_xtp_storage = ] include_total_xtp_storage ]
```

В качестве объекта должно быть указано имя базы данных, таблицы, индексированного представления или очереди. Поиск указанных объектов осуществляется внутри активной базы данных, поэтому попытка указать её имя в качестве идентификатора объекта приведёт к ошибке.

Процедура может быть выполнена без указания каких-либо параметров, в этом случае она вернёт две таблицы, которые содержат сведения об объёме дискового пространства, выделенного под размещение базы данных, размере индекса и оставшемся нераспределённом под объекты базы данных месте.

	database_name ▾	database_size ▾	unallocated space ▾
1	Tolmasov	10.00 MB	6.01 MB

	reserved ▾	data ▾	index_size ▾	unused ▾
1	3064 KB	1064 KB	1264 KB	736 KB

Рисунок 5.6 — Результат работы *sp_spaceused*

Используя эту информацию, можно оценить размер файла резервной копии. В файл резервной копии не попадает нераспределённое и неиспользуемое дисковое пространство и, следовательно, размер резервной копии будет примерно равен значению, указанному в поле *reserved* второй таблицы (рис. 5.6). Тем не менее, размер итогового файла может отличаться в большую сторону примерно на 10%.

5.2.2 Виды резервных копий

В MS SQL Server поддерживается три вида резервных копий баз данных (резервные копии файлов в данном пособии не рассматриваются):

- полная;
- разностная (дифференцированная);
- копия журнала транзакций.

Наличие трех видов резервных копий позволяет решать комплекс задач по обеспечению сохранности данных и позволяют достичь необходимой гибкости при восстановлении. При этом, грамотно спланированный процесс создания резервных копий позволяет оптимизировать использование дискового пространства, а также сократить время выполнения операций.

Не для каждой базы данных доступны все виды резервного копирования. Для того чтобы тот или иной вид стал доступен, необходимо выполнение определённых условий. Для удобства, описание и условия выполнения того или иного вида резервной копии сведены в таблицу (Таб. 5)

Таблица 5

Вид резервной копии	Описание	Условие доступности	Модель восстановления
Полная	Создаётся копия всей базы данных целиком. Включает часть журнала транзакций. Отражает состояние БД на момент завершения резервного копирования	Нет специальных условий	Любая

Таблица 6 (окончание)

Вид резервной копии	Описание	Условие доступности	Модель восстановления
Разностная	Включает только изменённую часть БД с момента последнего резервного копирования	Должна существовать полная копия	Любая
Журнала транзакций	Включает в себя копию всех записей журнала транзакций, которые были сделаны после последней резервной копии журнала.	Должна существовать полная копия, модель восстановления не SIMPLE	BULK-LOGGED, FULL

Для выполнения резервного копирования пользователь должен обладать разрешениями BACKUP DATABASE и BACKUP LOG.

5.2.3 Выполнение резервного копирования средствами MS SSMS

После подключения к экземпляру сервера, в окне обозревателя объектов нужно раскрыть список «Объекты сервера», и щелкнуть правой кнопкой мыши на «Устройства резервного копирования» и выбрать пункт «Создать устройство резервного копирования...» (Рисунок 5.7)

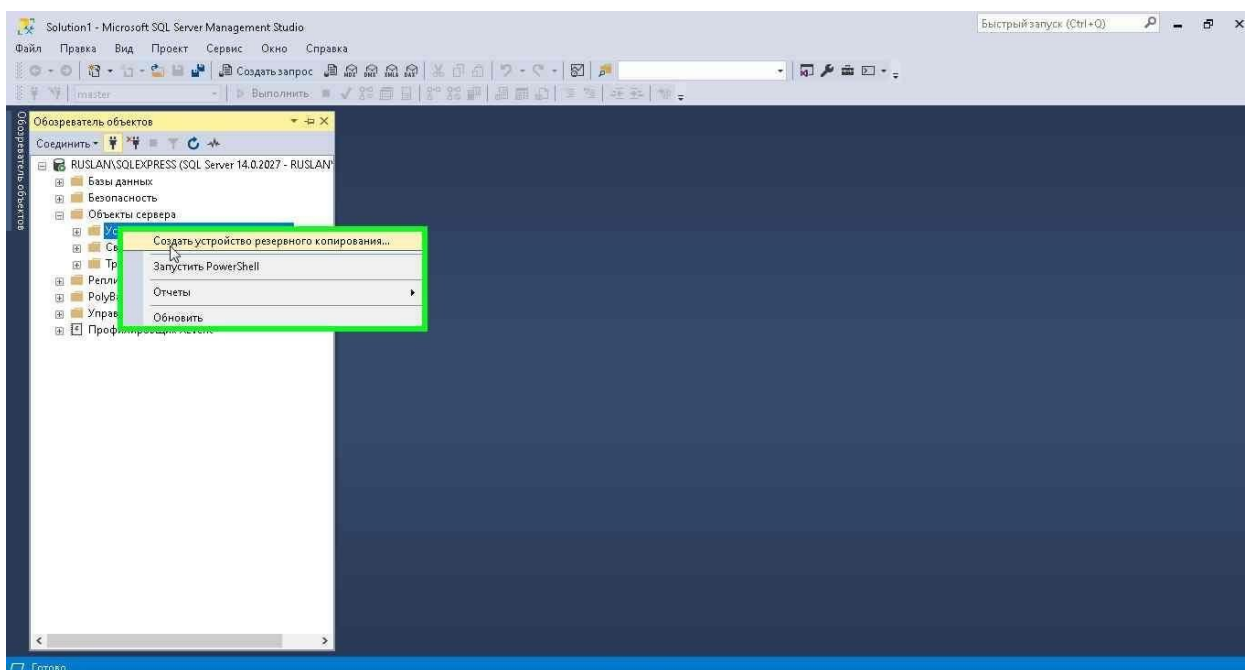


Рисунок 5.7 — Создание устройства резервного копирования

В открывшемся диалоговом окне можно указать путь, до места размещения устройства хранения, а также его тип. Database Engine поддерживает два типа носителей: магнитная лента и жесткие диски. Подробнее об устройствах можно прочитать в [10]. На рисунках ниже показан процесс выбора места размещения.

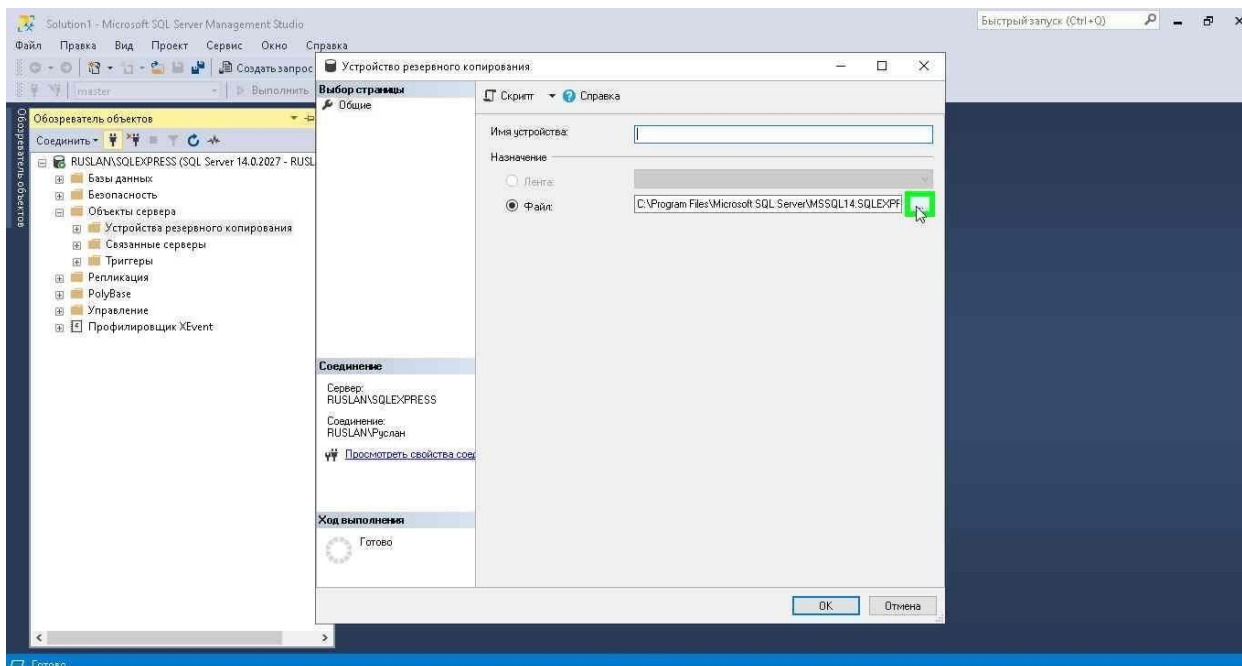


Рисунок 5.8 — Выбор пути к размещению файла резервной копии

Выбор жесткого диска (в том числе и логического) для размещения, а также задание имени файла

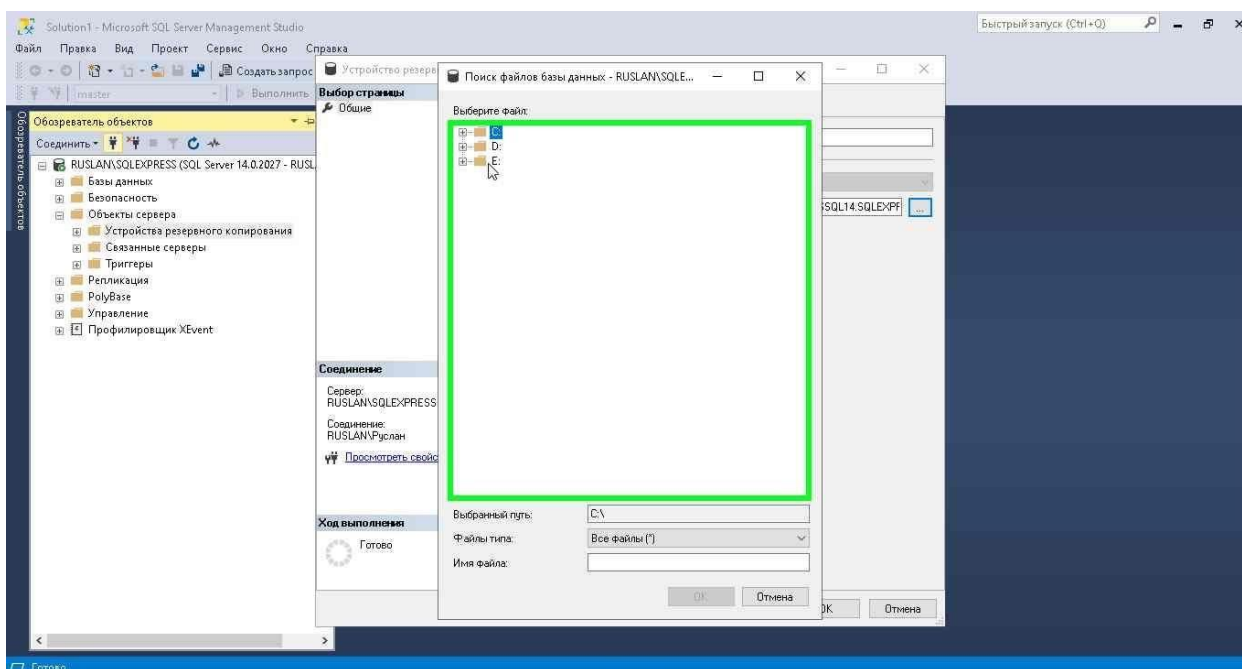


Рисунок 5.9 — Выбор места размещения и имени файла

После – нажмите «**Ок**»

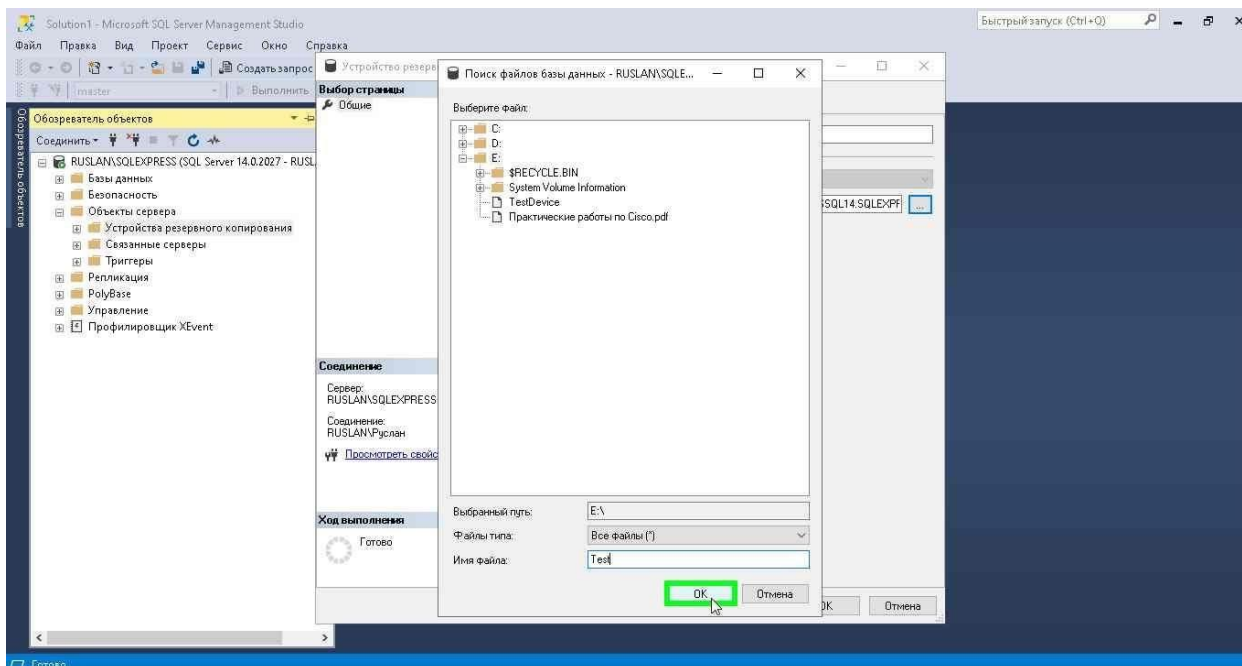


Рисунок 5.10 — Завершение выбора размещения файла резервной копии

Также необходимо указать имя устройства хранения, оно может быть произвольным в рамках принятой системы наименований и не обязано совпадать с именем базы данных или физического файла.

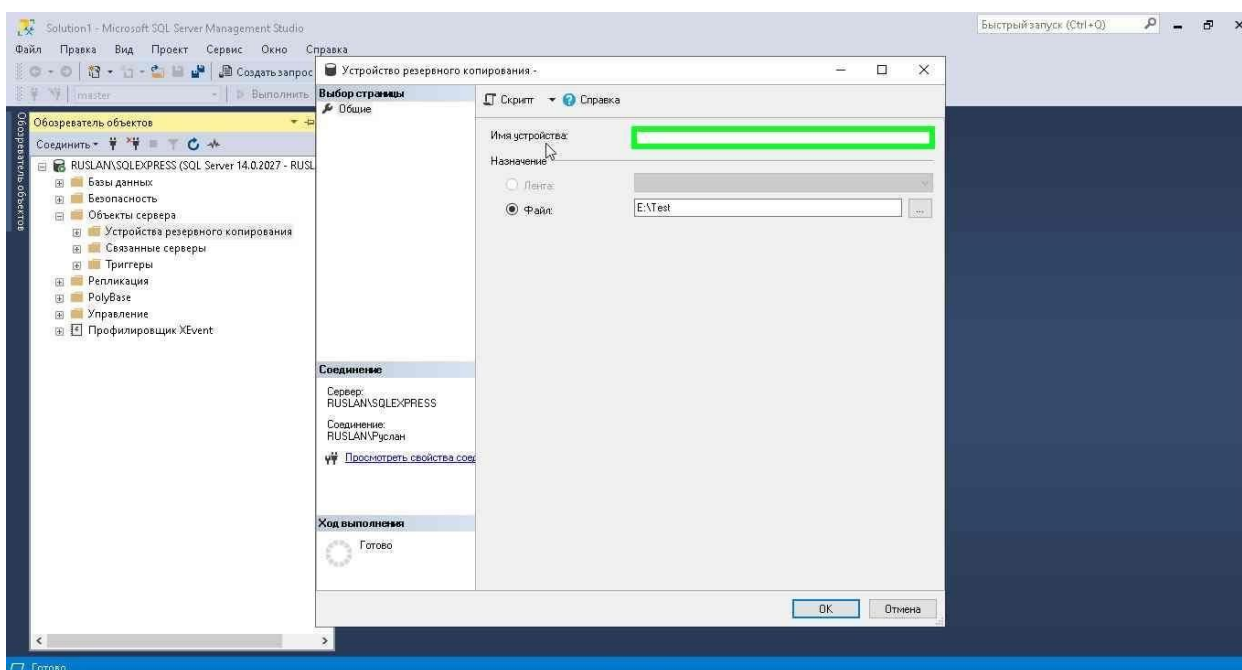


Рисунок 5.11 — Указание имени устройства

По окончании следует нажать на кнопку «Ок».

Разверните список баз данных, установленных на вашем сервере, и щёлкните правой кнопкой мыши по базе, которую Вам необходимо скопировать и выберите «Задачи/Создать резервную копию...»

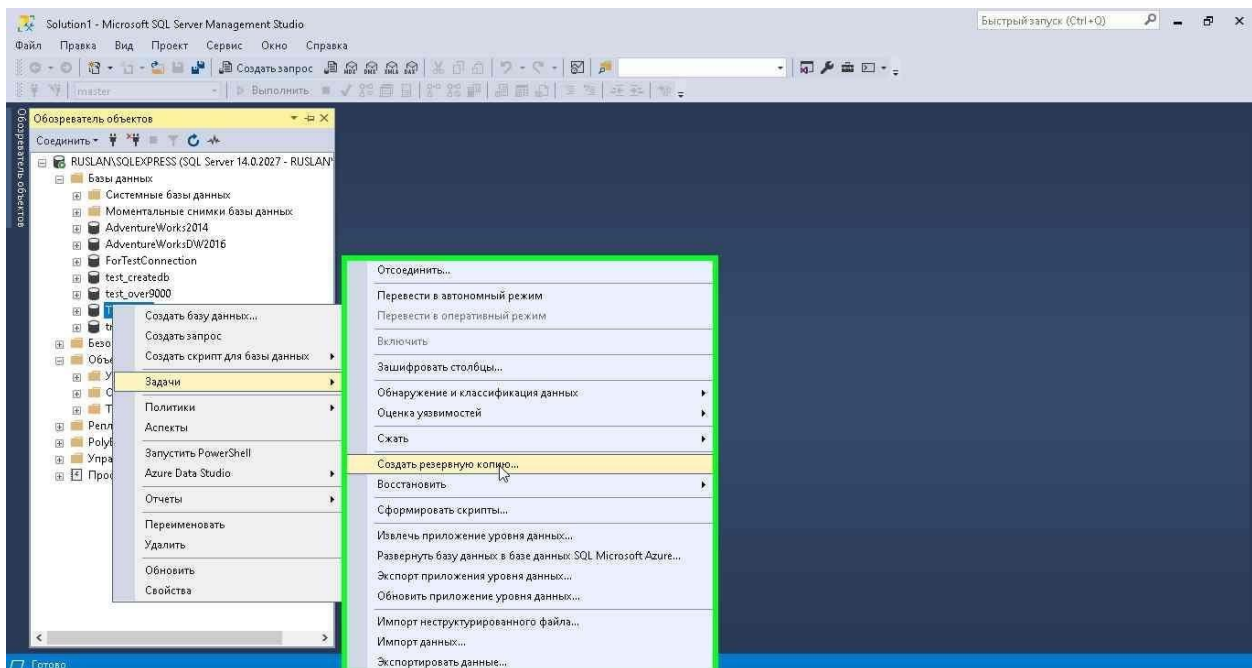


Рисунок 5.12 — Открытие диалогового окна создания резервной копии

В открывшемся диалоговом окне можно выбрать:
базу данных если Вы вдруг ошиблись;

- модель восстановления;
- тип резервной копии;
- компонент архивной копии;
- назначение (куда будем сохранять).

Список может содержать различные точки назначения, но в данном случае он пуст. Для того чтобы добавить его, следует нажать на кнопку в правой части формы (Рисунок 5.8)

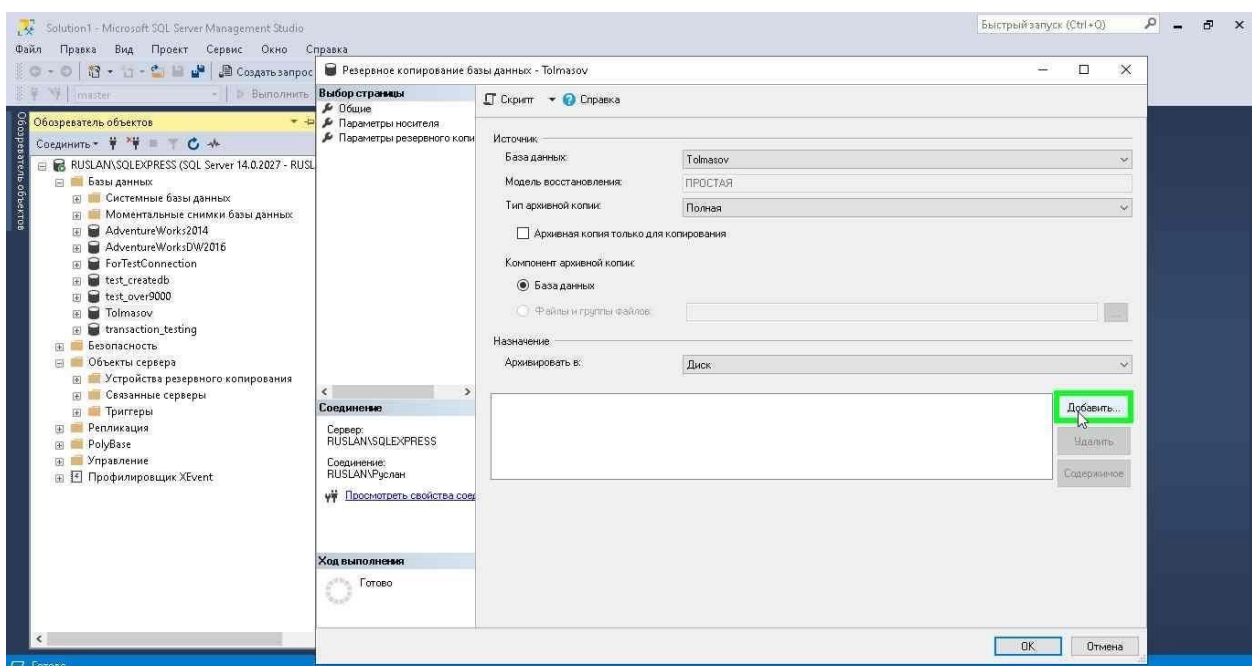


Рисунок 5.13 — Добавление места назначения

В открывшемся диалоговом окне можно выбрать либо конкретный файл, либо логическое устройство, созданное ранее. Выберите пункт «*Устройство резервного копирования*» (Рисунок 5.14 — Рисунок 5.16).

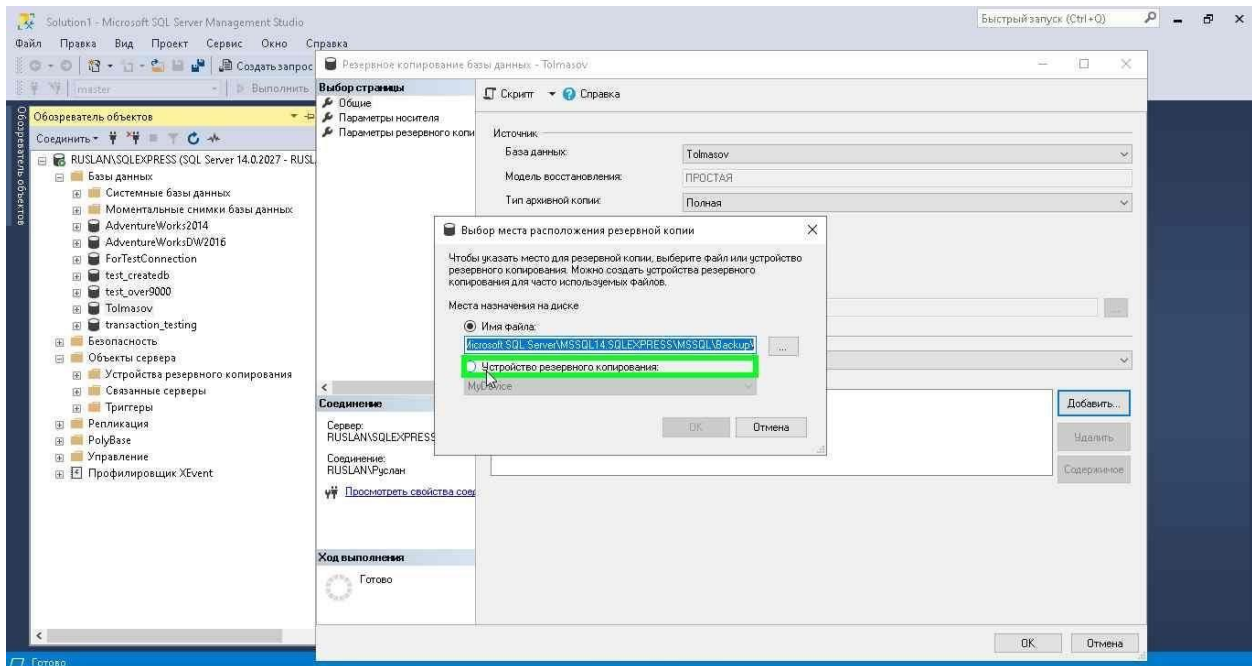


Рисунок 5.14 — Выбор места расположения резервной копии

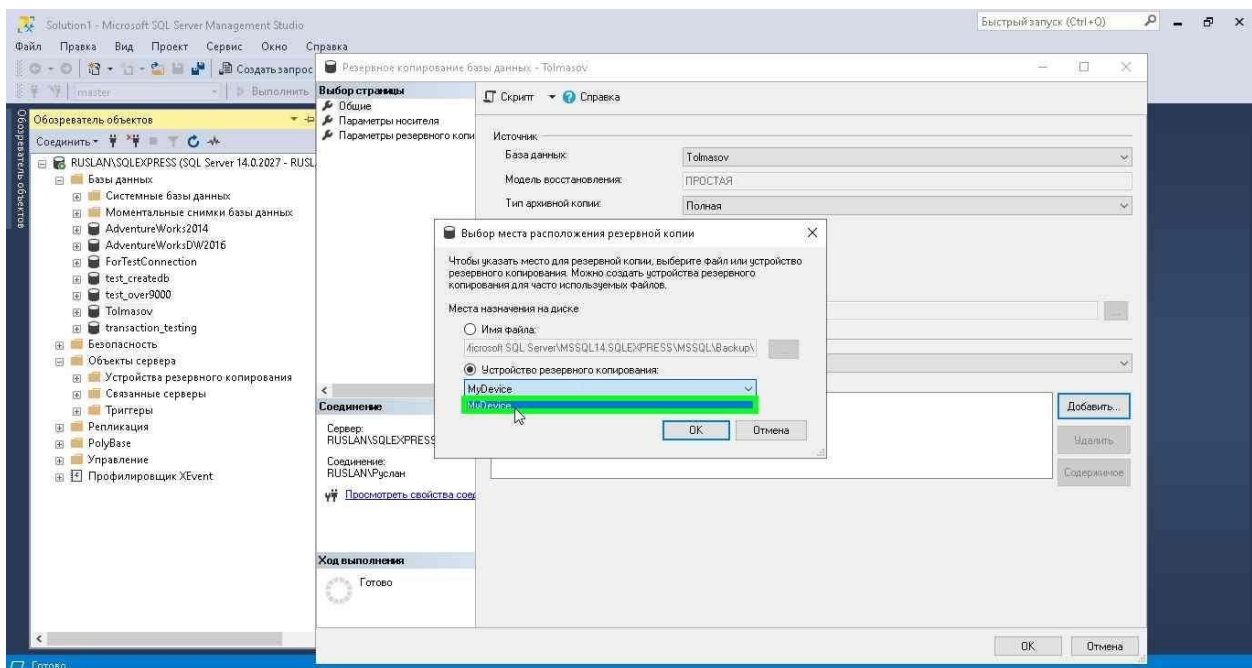


Рисунок 5.15 — Выбор ранее созданного устройства

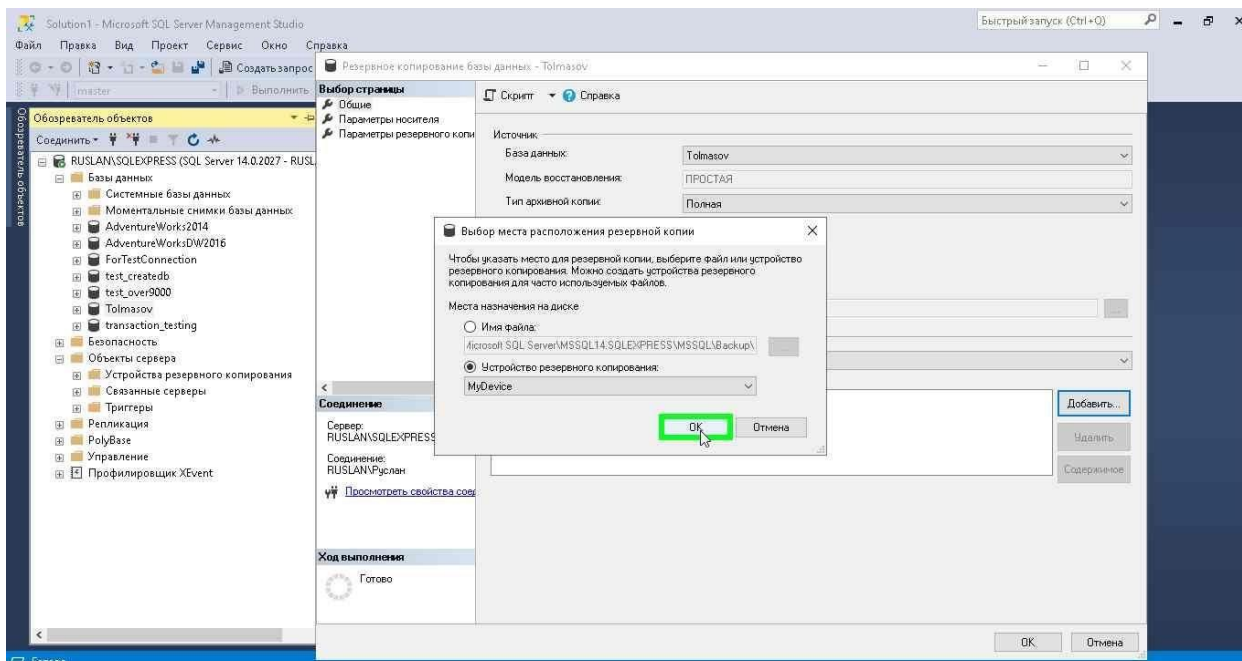


Рисунок 5.16 — Завершение выбора место расположения

Теперь устройство стало доступным для выбора, щёлкните по нему левой кнопкой мыши.

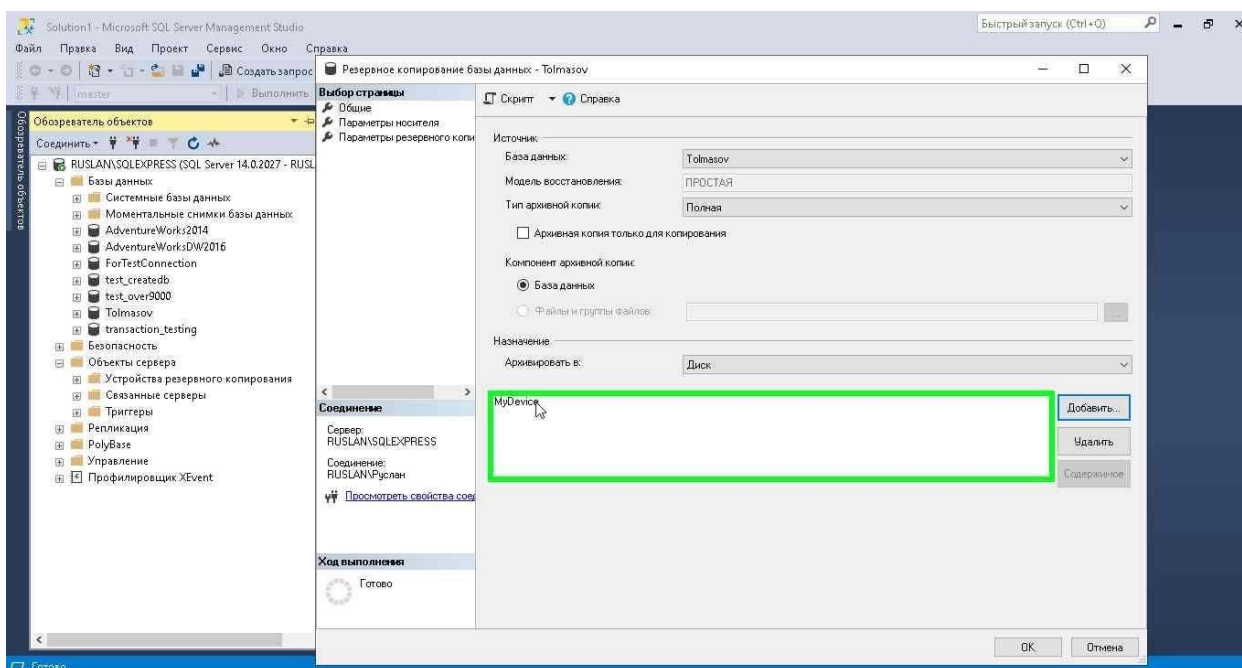


Рисунок 5.17 — Выбор добавленного устройства

Перед тем как начать процедуру резервного копирования следует перейти в «**Параметры резервного копирования**» выбрав из списка в левой части формы.

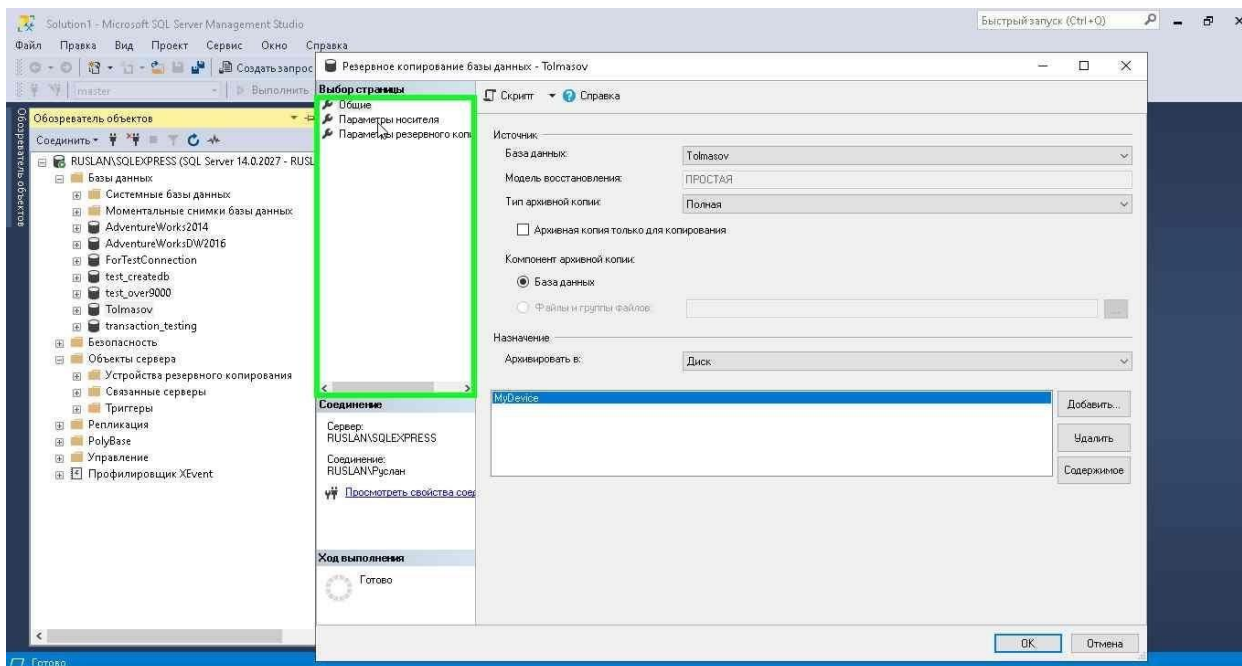


Рисунок 5.18 — Навигация по диалоговому окну настроек резервного копирования

В пункте «**Сжимать резервные копии**» следует выбрать вариант: «**Не сжимать резервные копии**», так как *SQL Express* не поддерживает резервное копирование со сжатием.

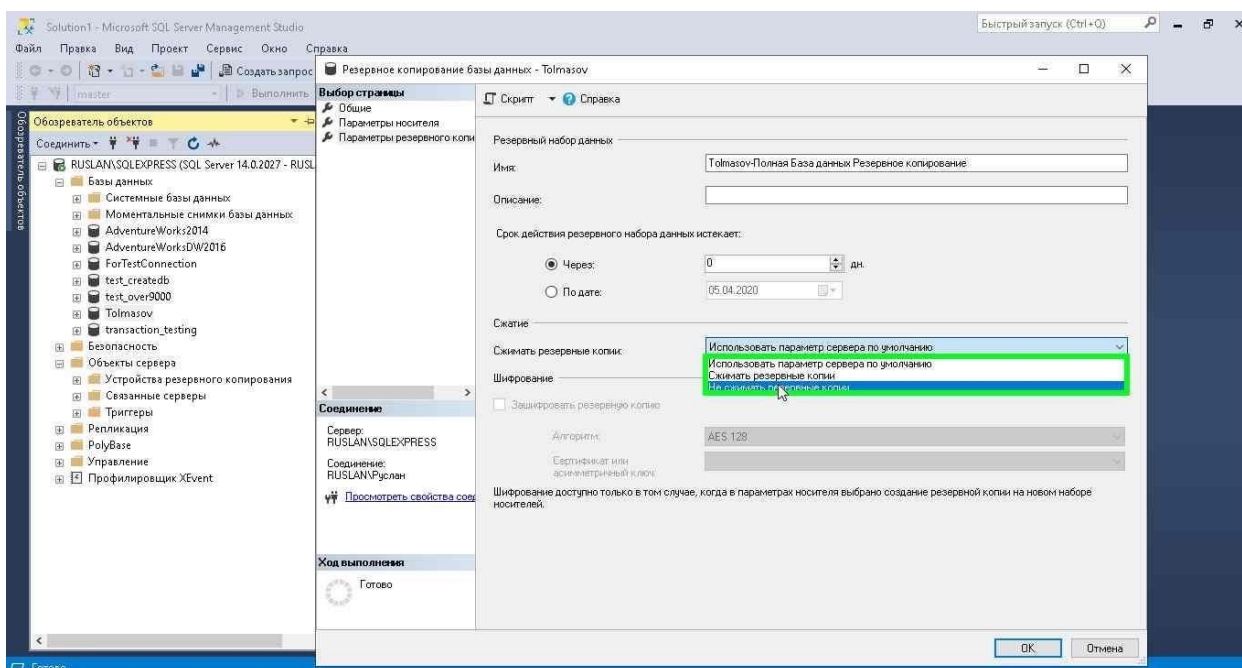


Рисунок 5.19 — Выбор режима сжатия

После, следует нажать на кнопку "OK"

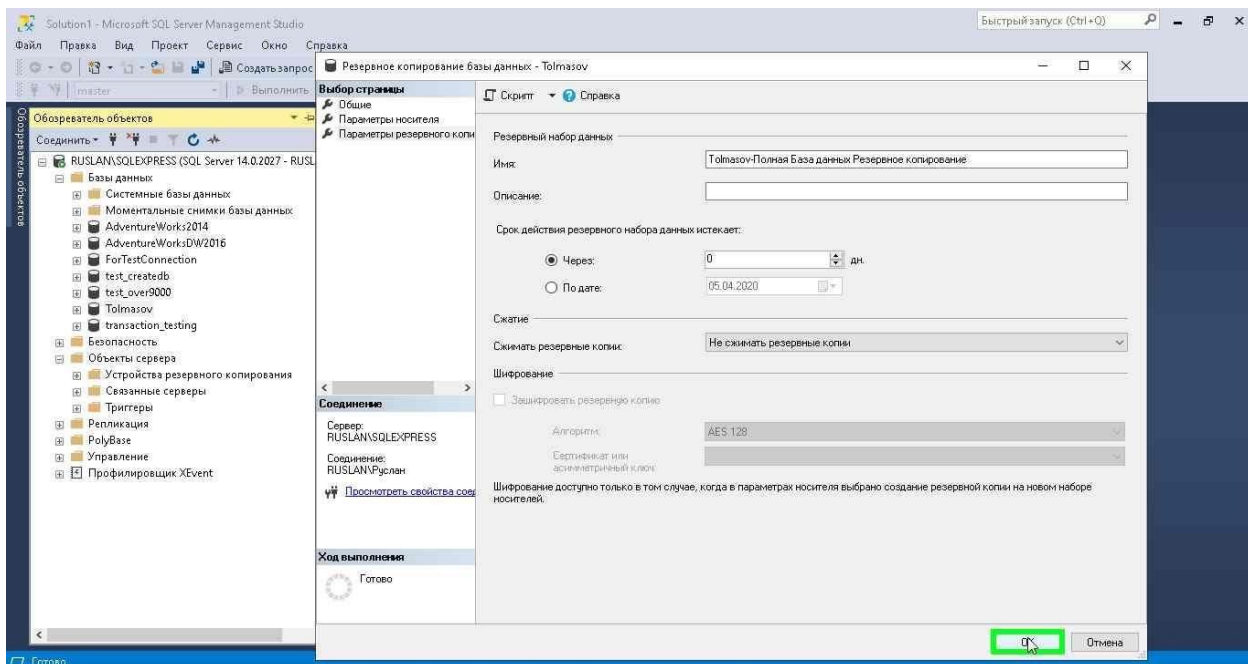


Рисунок 5.20 — Окончание настроек резервного копирования

Резервное копирование завершено.

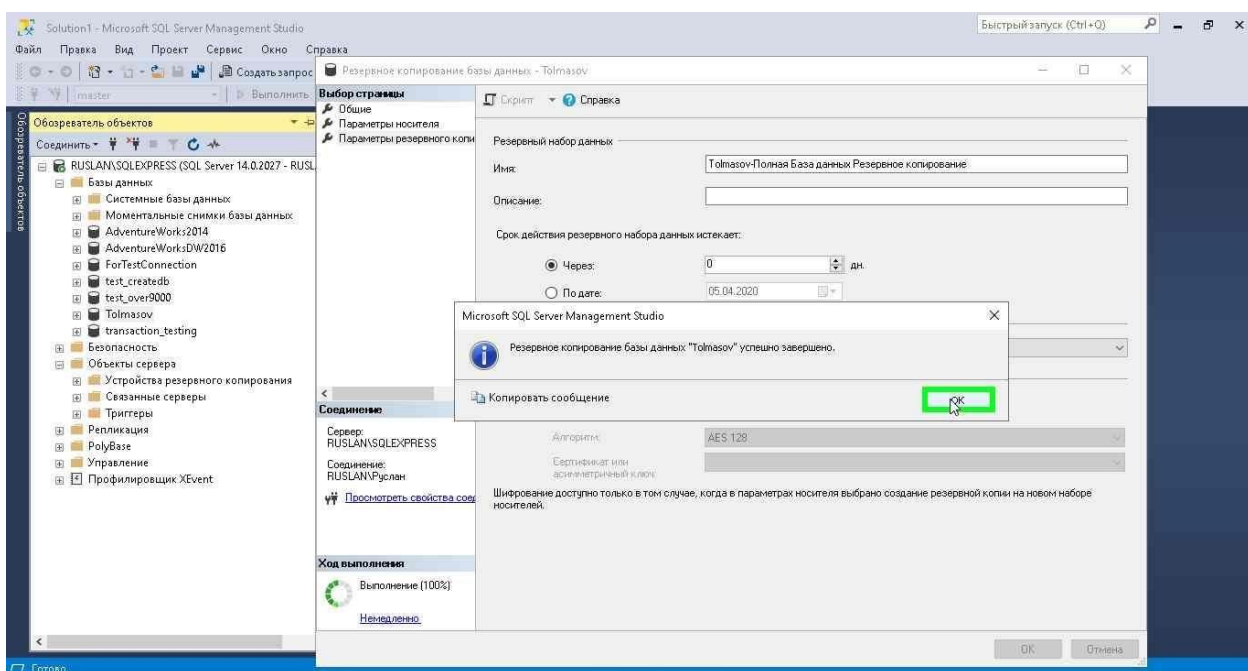


Рисунок 5.21 — Резервное копирование успешно завершено

Аналогично выполняются разностные копии и копии журнала транзакций. Но следует помнить, что эти типы копий доступны только после выполнения полной резервной копии и не могут применяться вместо неё на постоянной основе.

Для пользователей операционной системы Linux (и других операционных систем) воспользоваться средой *MS SSMS* на данный момент не получится, однако программное средство *Azure Data Studio* поддерживает возможность

выполнения резервной копии. Подробно разбирать процесс создания резервной копии не имеет смысла, укажем только интерфейсные различия в выполнении процесса.

Для создания резервной копии базы данных необходимо выбрать в меню подключений нужное подключение и развернув его найти в раскрывшемся дереве нужную базу данных. Нажатием правой кнопкой мыши по имени базы данных выбрать пункт «*Резервное копирование*»

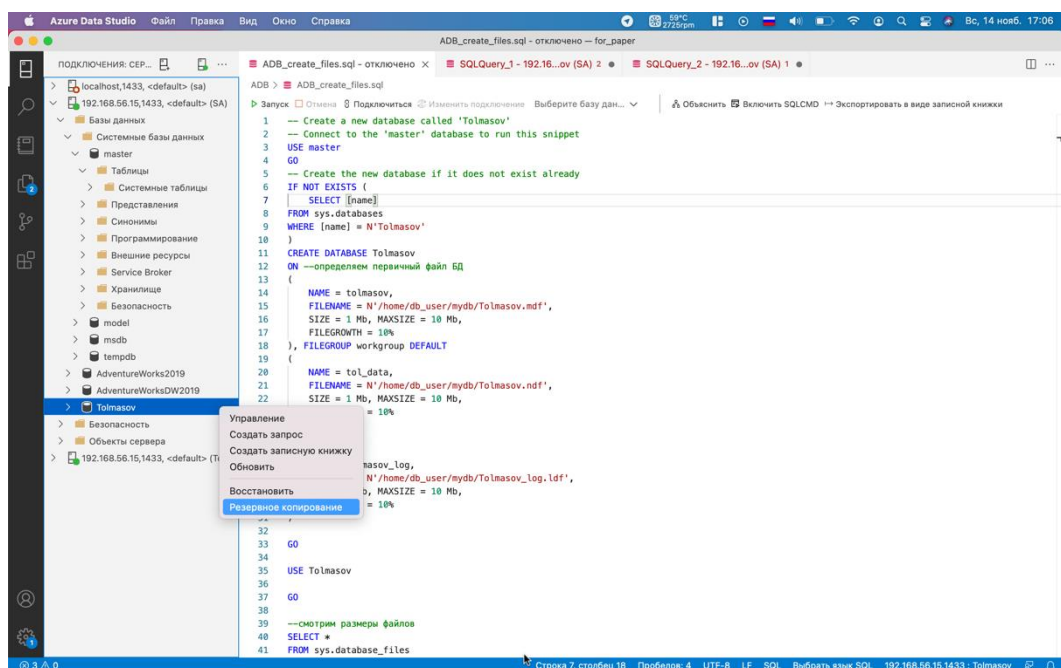


Рисунок 5.22 — Открытие настроек резервного копирования в Azure Data Studio

В правой части экрана появится окно настроек резервного копирования (рис.5.23)

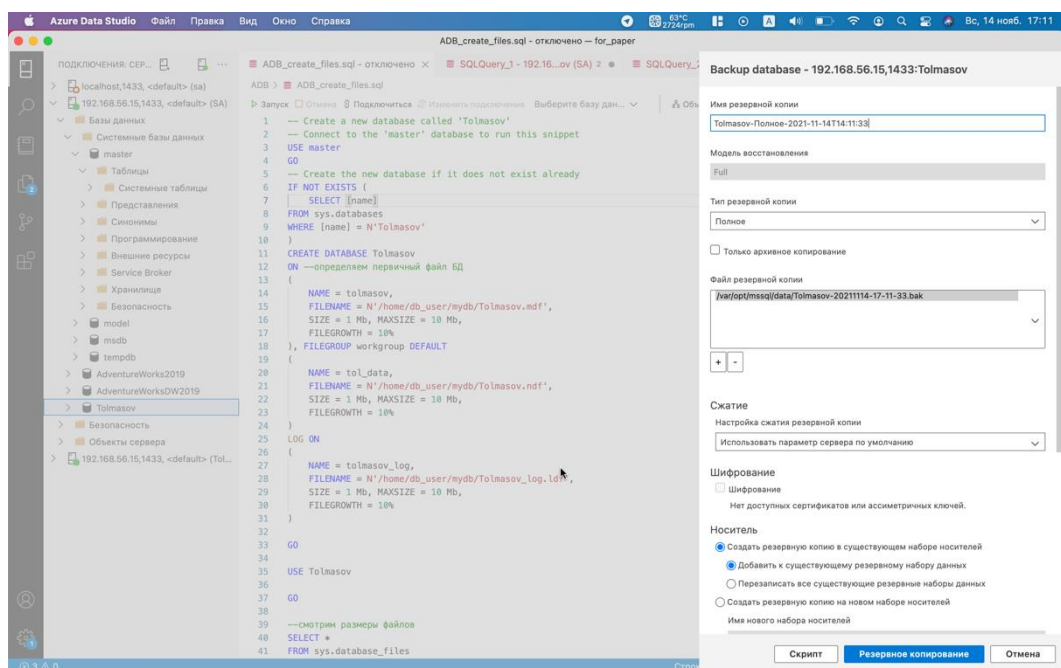


Рисунок 5.23 — Редактор настроек резервного копирования

Пункты, разнесённые по разным экранным формам и окошкам в MS SSMS в Azure Data Studio представлены на одной странице и имеют практически идентичные заголовки. Аккуратно заполнив форму необходимыми параметрами следует нажать на кнопку «*Резервное копирование*» внизу страницы.

5.2.4 Резервное копирование с помощью средств языка Transact-SQL

Приведём синтаксис инструкции для резервного копирования базы данных:

```
BACKUP DATABASE { database_name | @database_name_var }  
TO <backup_device> [ ,...n ]  
[ WITH { DIFFERENTIAL | <general_WITH_options> [ ,...n ] } ]  
[;]
```

Выше приведена синтаксическая конструкция скрипт создания полной резервной копии. Первая строка указывает на выполняемую операцию, в данном случае выполняется резервное копирование базы данных, имя которой должно быть указано сразу после этого объявления. На следующей строке необходимо указать не менее одного устройства резервного копирования.

В качестве устройства могут быть указаны логические устройства (которые создавались в предыдущем разделе), так и прямой путь к файлу резервной копии или ссылка на BLOB-облако.

```
<backup_device>::=  
{  
  { logical_device_name | @logical_device_name_var }  
  | { DISK | TAPE | URL } =  
    { 'physical_device_name' | @physical_device_name_var | 'NUL' }  
}
```

Не обязательно, но часто необходимо указывать свойства резервного копирования после ключевого слова WITH. Если сразу после WITH указать DIFFERENTIAL, то вместо полного копирования будет осуществлено разностное резервное копирование базы данных. Другие возможные варианты приведены ниже:

```
<general_WITH_options> [ ,...n ]::=  
--Backup Set Options  
  COPY_ONLY  
  | { COMPRESSION | NO_COMPRESSION }  
  | DESCRIPTION = { 'text' | @text_variable }  
  | NAME = { backup_set_name | @backup_set_name_var }  
  
--Media Set Options  
  { NOINIT | INIT }
```

```
| { NOSKIP | SKIP }  
| { NOFORMAT | FORMAT }  
| MEDIADESCRIPTION = { 'text' | @text_variable }  
| MEDIANAME = { media_name | @media_name_variable }  
| BLOCKSIZE = { blocksize | @blocksize_variable }
```

--Log-specific Options

```
{ NORECOVERY | STANDBY = undo_file_name }  
| NO_TRUNCATE
```

Backup set options определяют общие параметры резервной копии, такие как сжатие, описание содержимого, логическое имя резервной копии. Опции группы Media set option определяют порядок работы с устройством резервного копирования, например, добавив опцию FORMAT можно предварительно отчистить устройство резервного копирования от предыдущих копий. Перечень всех свойств и их значений можно найти соответствующей документации.

Приведем пример скрипта выполнения резервной копии базы данных по физическому адресу файла резервной копии:

```
BACKUP DATABASE Tolmasov  
TO DISK = N'/home/db_user/mydb/Tolmasov.bak'  
WITH FORMAT,  
MEDIANAME = 'Tolmasov_backup',  
MEDIADESCRIPTION = 'Testbackup';
```

В приведённом примере создаётся полная резервная копия созданной ранее базы данных, предварительно очищая файл от предыдущих вариантов резервных копий и добавляет к записи логическое имя для носителя и описание содержимого.

5.3 Восстановление баз данных

В ситуациях повреждения рабочей базы данных необходимо прибегнуть к процедуре восстановления, для успешного выполнения которой потребуется некоторый набор резервных копий, состав которого зависит от того какой тип резервной копии применялся для этой базы данных.

В случае использования полных резервных копий потребуется только последняя версия резервной копии. В случае если использовалась разностная или копия журнала транзакций, то кроме полной копии потребуется и все файлы частичных копий и/или копий журнала транзакций в хронологическом порядке, созданных с момента последней полной копии.

В случае повреждения файла в этой цепочке, восстановление будет возможно только в состоянии базы данных, соответствующее состоянию

зафиксированному в файле предшествующего поврежденному.

Восстановление баз данных возможно выполнить с помощью MS SSMS, Azure Data Studio или T-SQL.

5.3.1 Восстановление с помощью MS SSMS

Авторизуйтесь на сервере именем входа, имеющего соответствующие разрешения и нажмите правой кнопкой мыши на «**Базы данных**» (рис. 5.24).

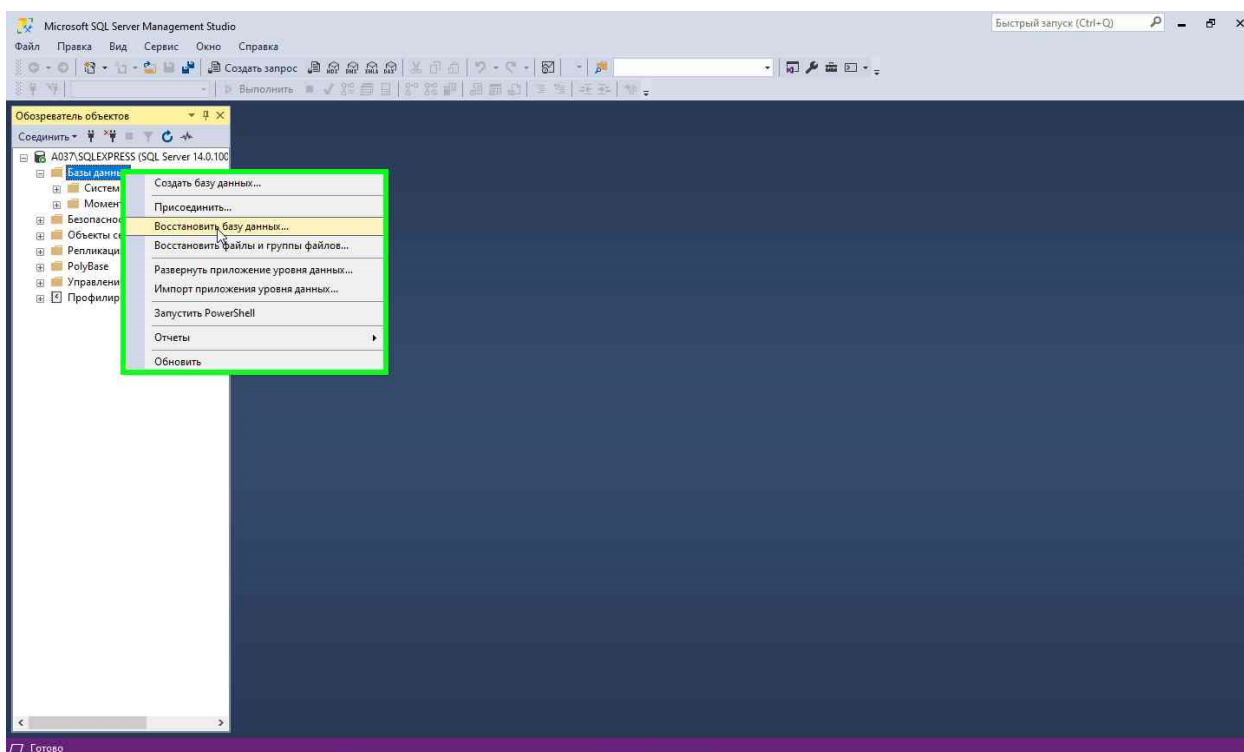


Рисунок 5.24 — Контекстное меню объекта сервера "Базы данных"

В контекстном меню выберете пункт «**Восстановить базу данных...**» и дождитесь открытия окна «**Восстановление баз данных**» (рис. 5.24)

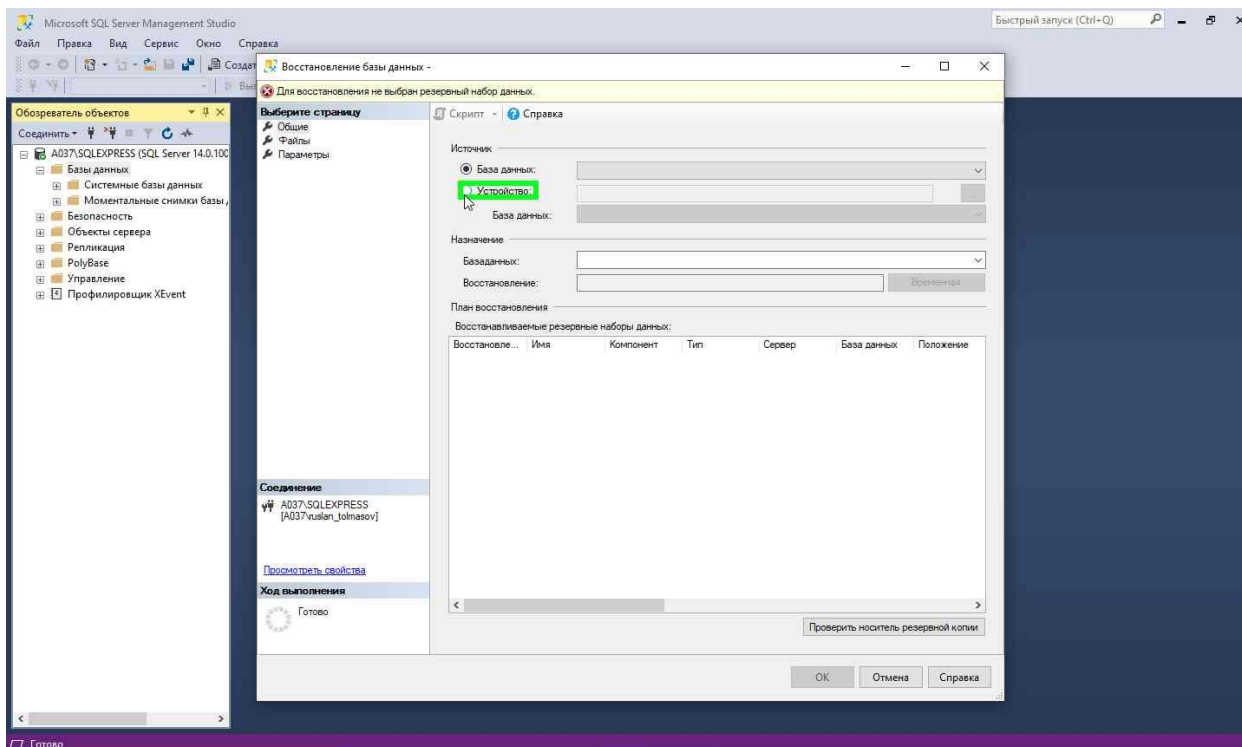


Рисунок 5.25 — Окно "Восстановление базы данных"

Выберите пункт «**Устройство**» (рис. 5.25) и далее щелкните левой мышью по кнопке «...» (рис. 5.26), вызвав диалоговое окно выбора устройства резервной копии.

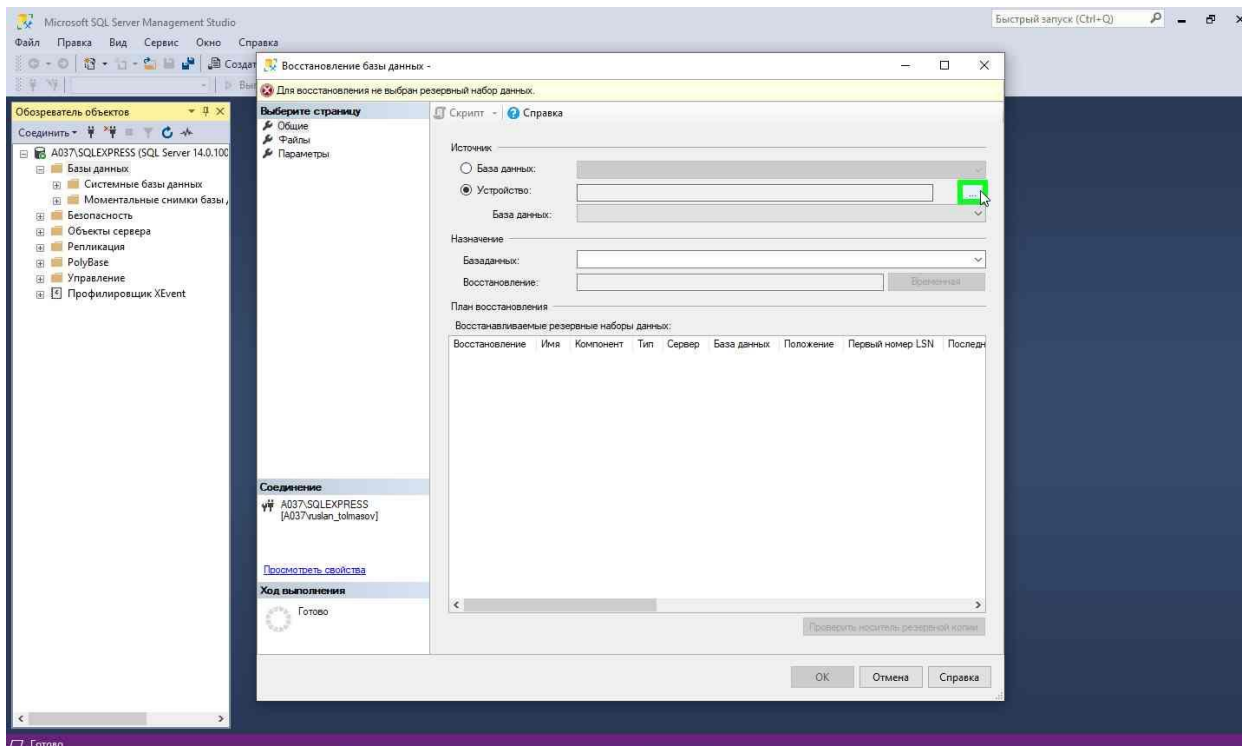


Рисунок 5.26 — Выбор устройства резервного копирования

В диалоговом окне «**Выбор устройства резервного копирования**» необходимо нажать на кнопку «**Добавить**» (рис. 5.7)

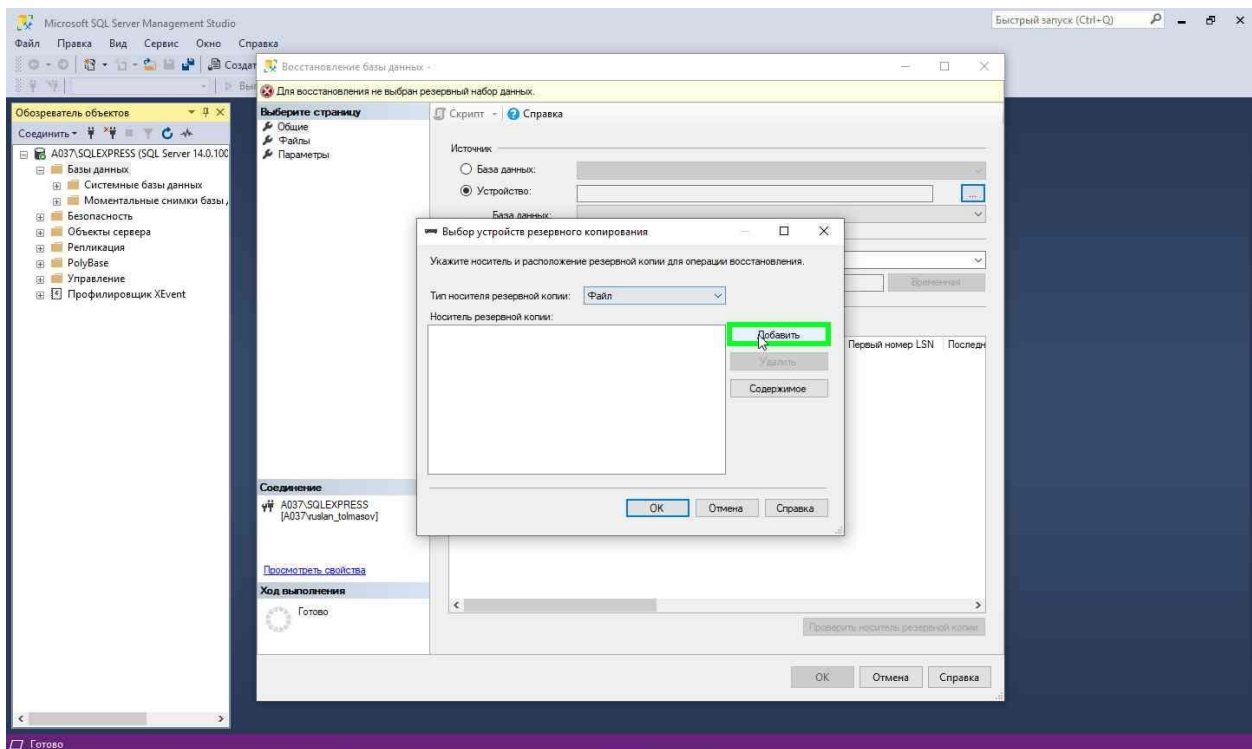


Рисунок 5.27 — Окно добавления устройства/файлов резервных копий

Далее необходимо найти *bak*-файл содержащий необходимую резервную копию (рис. 5.28)

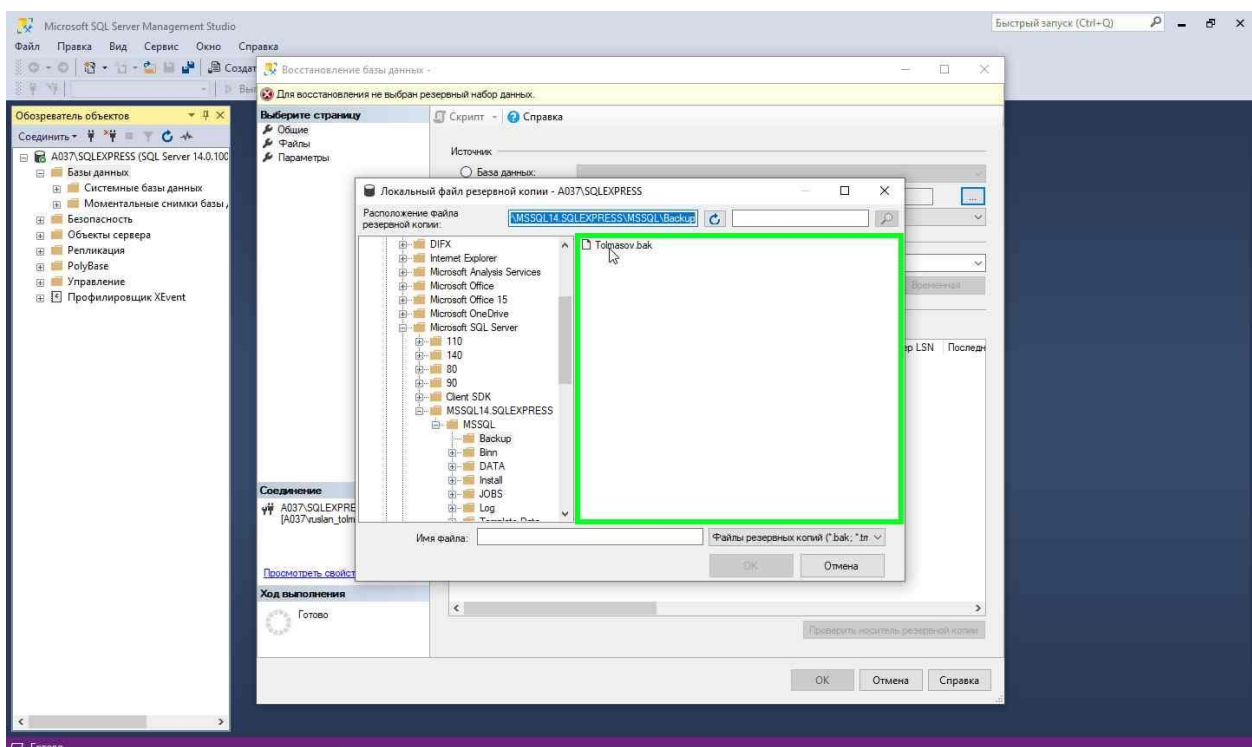


Рисунок 5.28 — Поиск и выбор файла резервной копии

Подтвердите выбор файла нажав на кнопку «**OK**» (Рис. 5.29)

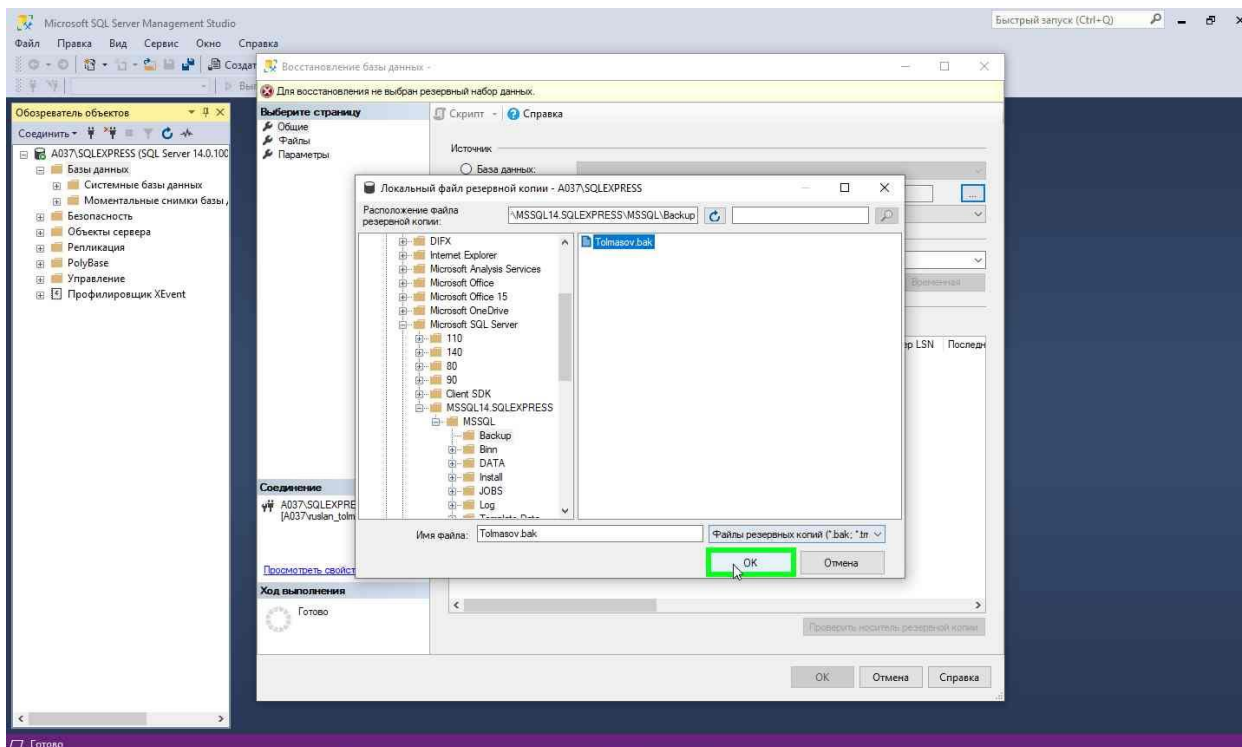


Рисунок 5.29 — Подтверждение выбора

Подтверждаем выбор нажатием на кнопку «OK» в окне выбора устройства резервного копирования (Рис. 5.30).

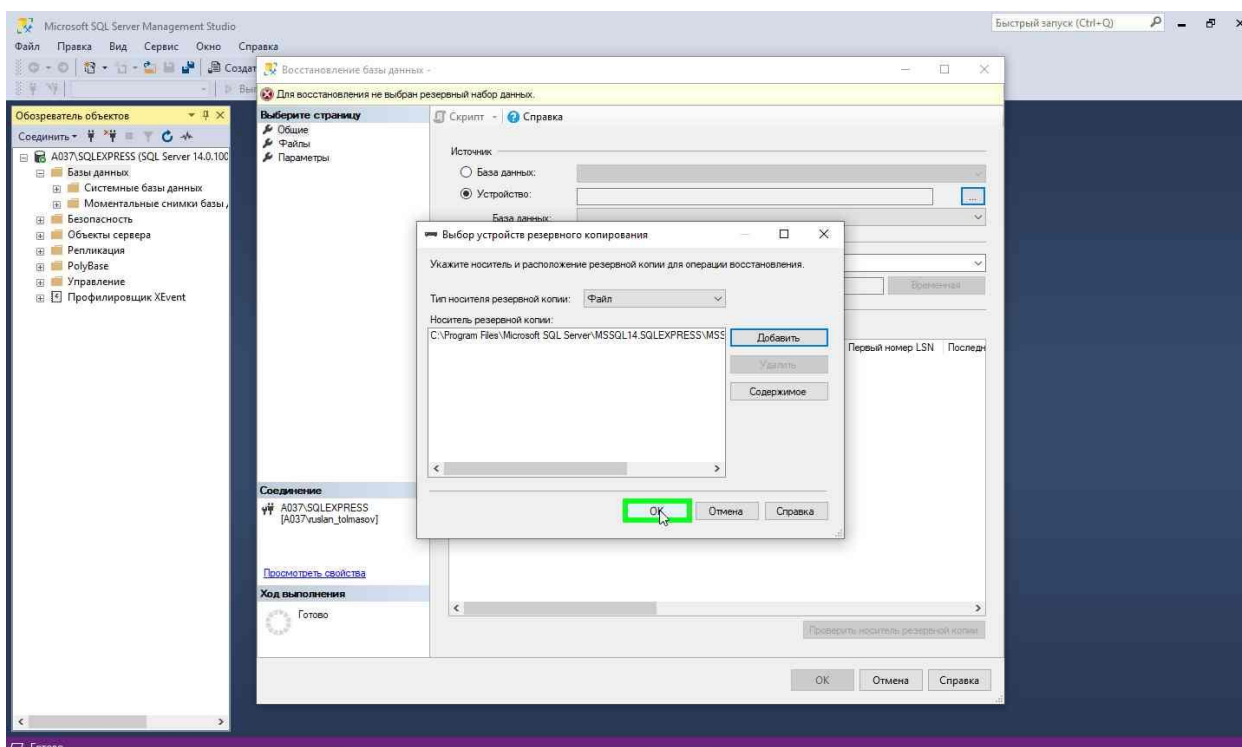


Рисунок 5.30 — Подтверждение выбранного файла

Проверьте параметры, прочитанные из файла резервной копии. Кроме того, можно провести предварительное тестирование носителя резервной копии. Если всё исправно, а метаданные прочитаны корректно, можно начинать

восстановление нажатием на кнопку «OK» (Рис. 5.31)

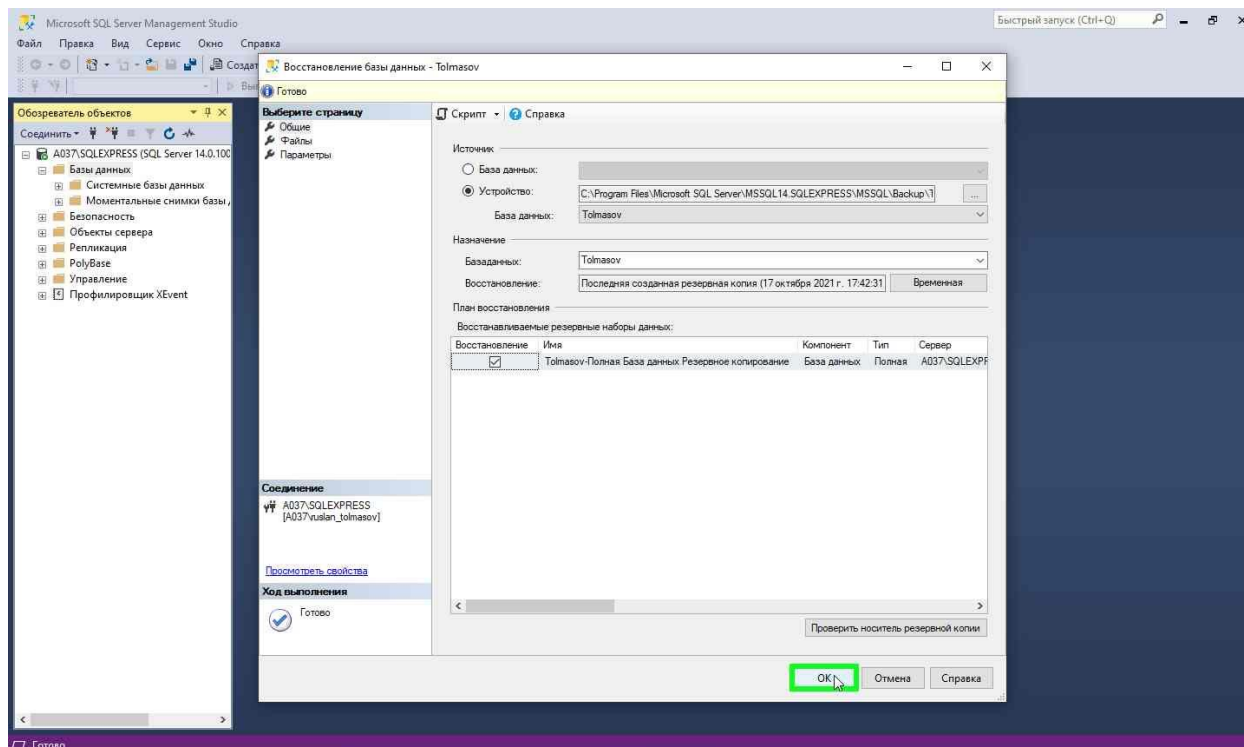


Рисунок 5.31 — Запуск процесса восстановления базы данных

Дождитесь окончания процесса восстановления и закройте информационное окно нажатием на кнопку «OK» (Рис. 5.32)

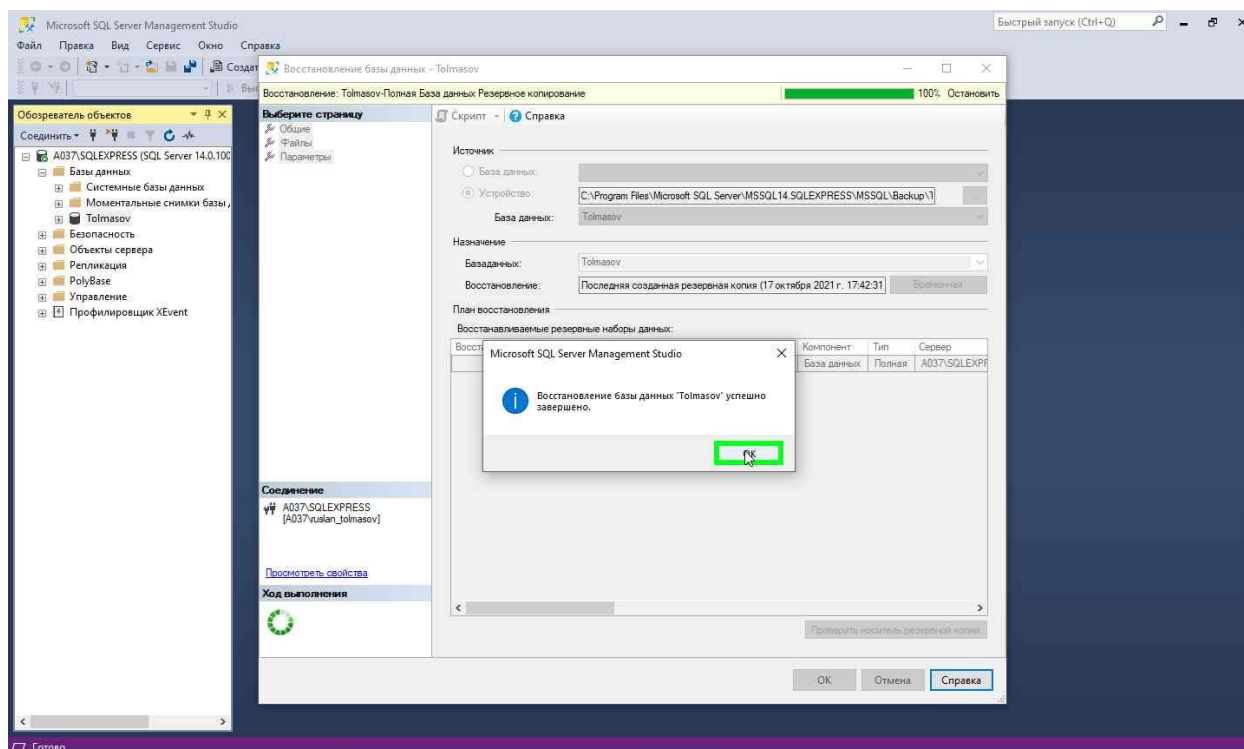


Рисунок 5.32 — Успешное завершение процесса восстановления базы данных

Рассмотрим способ восстановления базы данных с помощью Azure Data Studio.

Нажмите правой кнопкой мыши на подключении к нужному серверу и в контекстном меню выберите пункт «Управление» (Рис. 5.33)

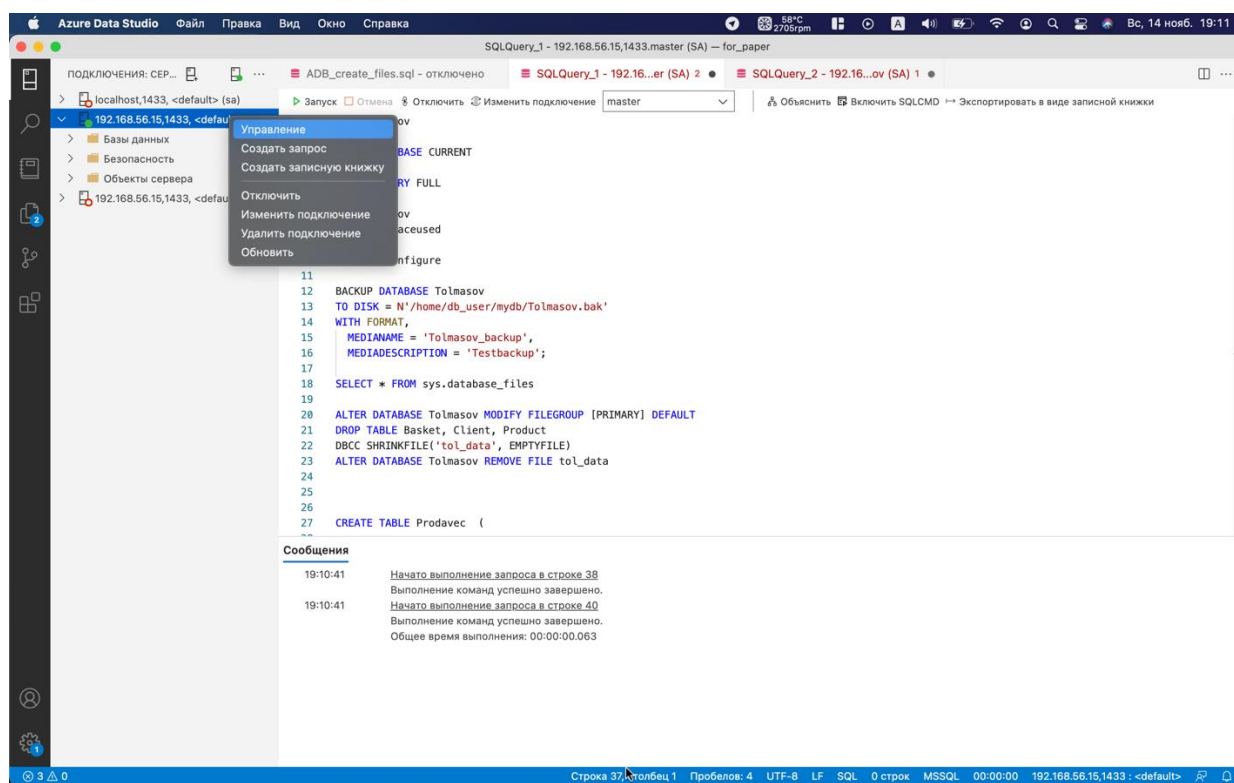


Рисунок 5.33 — Вызов страницы управления сервером

На открывшейся странице необходимо нажать на кнопку «Восстановить» (Рис. 5.34)

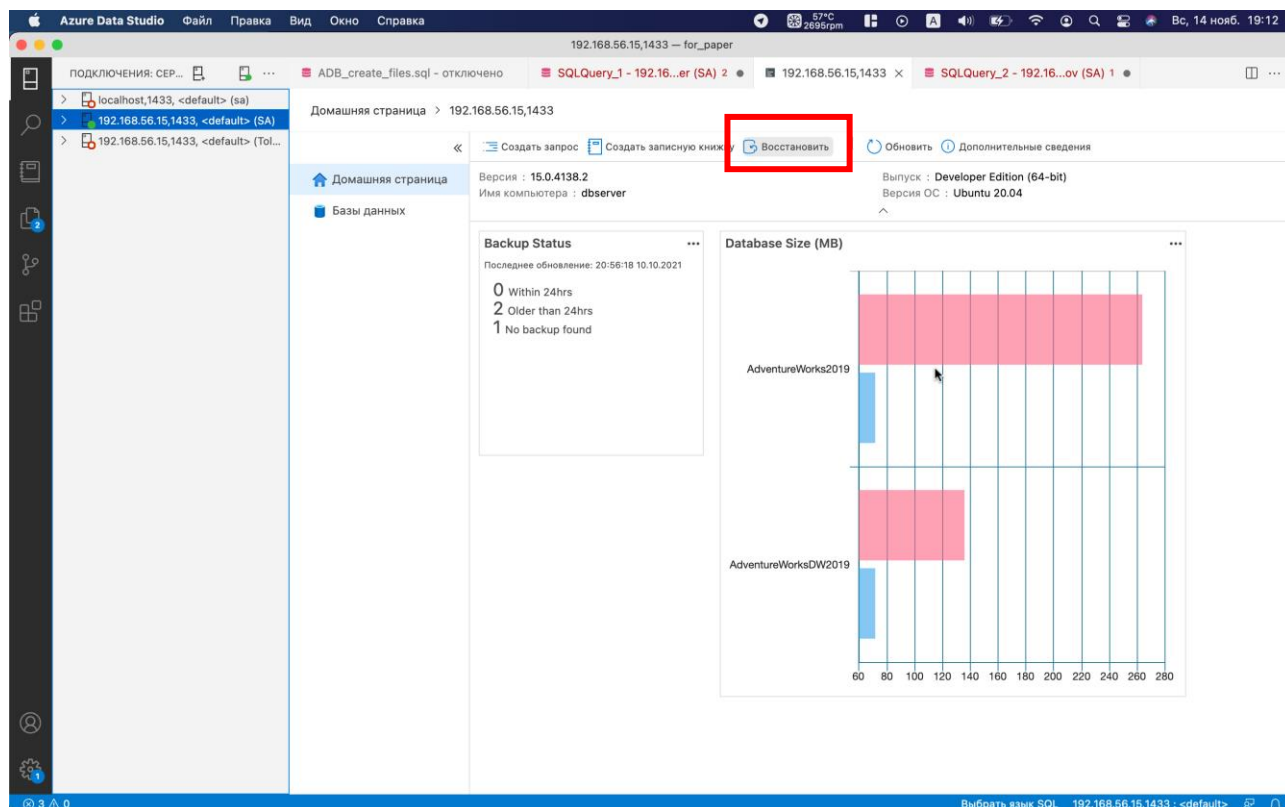


Рисунок 5.34 — Страница управления сервером

Откроется диалоговое окно настроек параметров восстановления (Рис. 5.35)

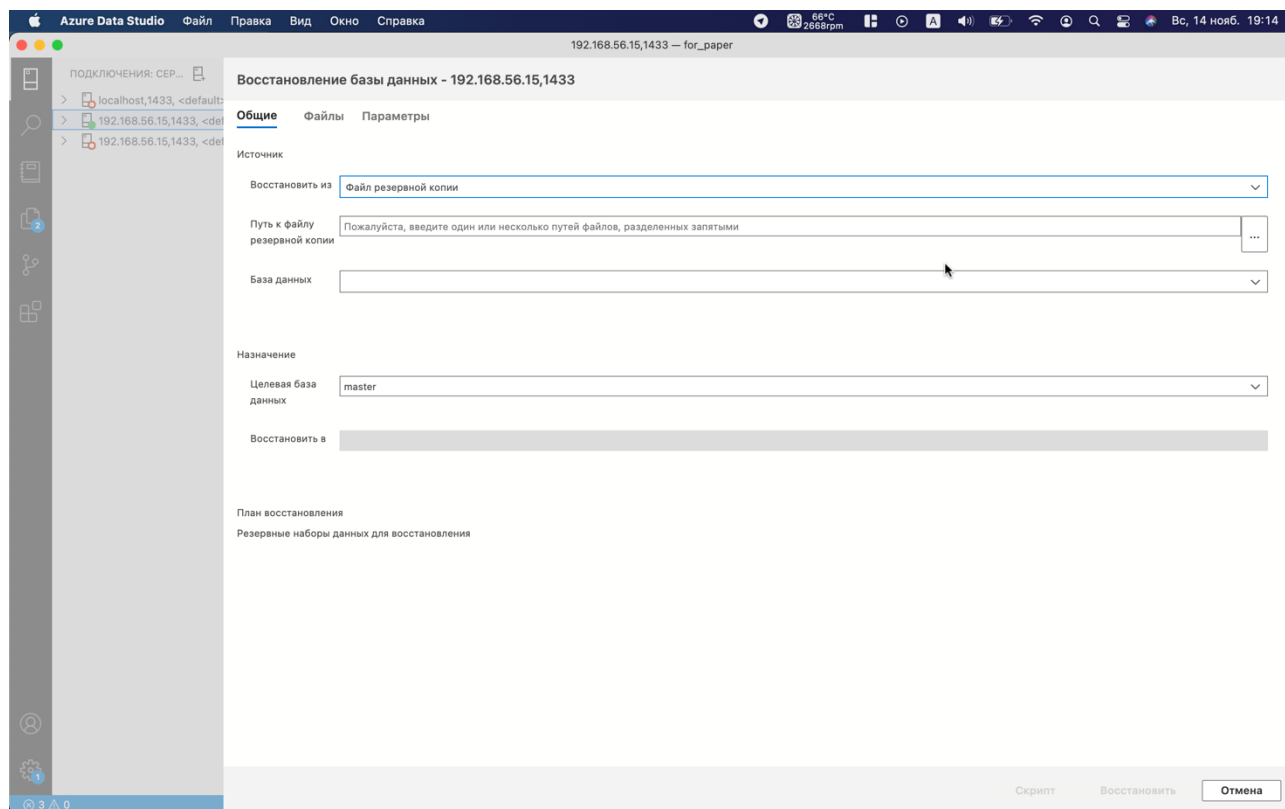


Рисунок 5.35 — Окно восстановления базы данных

На вкладке «**Общие**», которая открыта при появлении диалогового окна необходимо выбрать источник восстановления (в нашем случае из файла) и указать путь к файлу резервной копии, вызвав диалоговое окно нажав на кнопку «...» справа поля ввода «**Путь к файлу резервной копии**». В правой части экрана появится панель открытия файла, с её помощью необходимо отыскать файл с резервной копией.

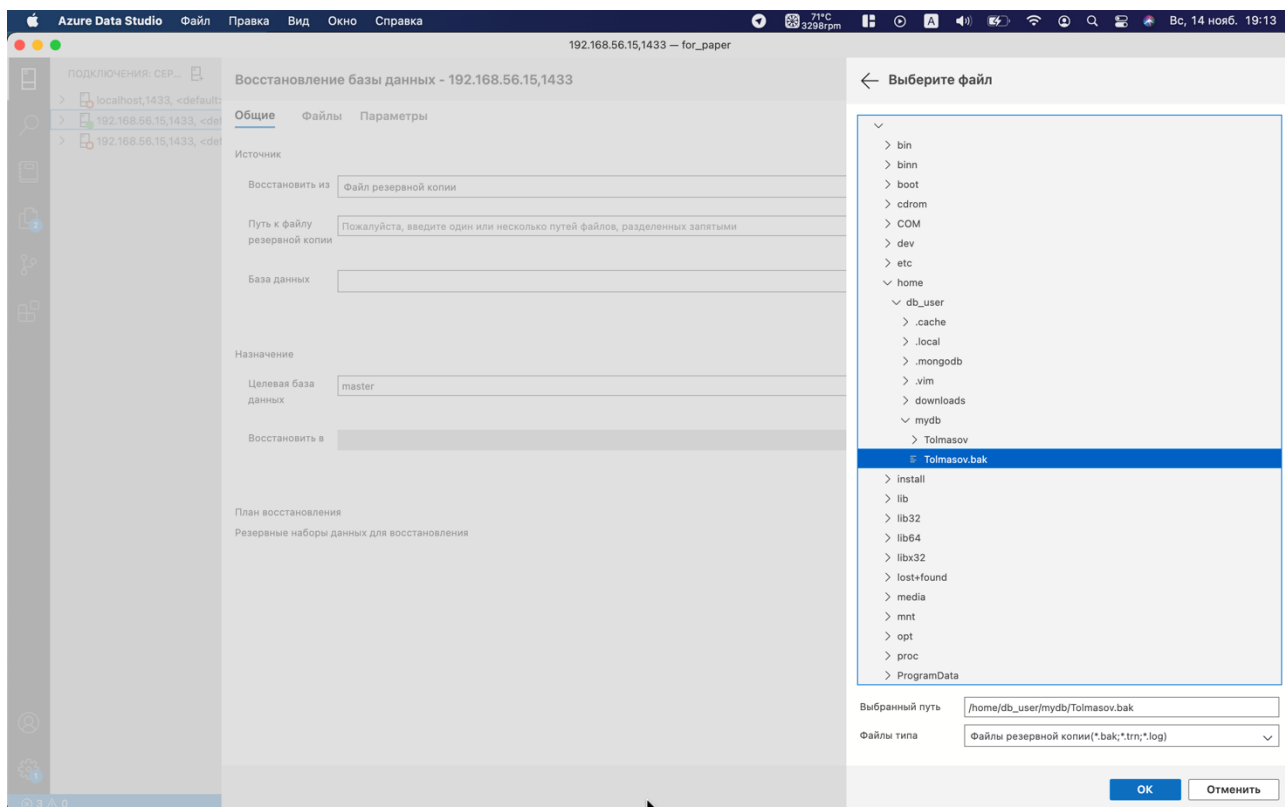


Рисунок 5.36 — Поиск файла резервной копии

Обратите внимание, что поиск осуществляется в файловой системе сервера. Если есть необходимость восстановиться из удалённой копии, то для этого способа её необходимо предварительно загрузить на сервер. При этом следует помнить, что СУБД не обладает правами администратора и некоторые папки или файлы могут быть недоступны. В случае возникновения ошибки проверки прав доступа, в первую очередь следует проверить имеет ли пользователь необходимые разрешения, во-вторую очередь разрешения на чтение из папки на уровне операционной системы.

Выбрав нужный файл следует нажать на кнопку «**OK**» в нижней части панели.

Часть элементов окна восстановления заполнится данными, полученными из файла резервной копии и появится возможность ознакомиться с метаданными выбранного файла (Рис. 5.37). При необходимости выполнения тонкой настройки операции восстановления или ознакомится с файлами входящими в состав файла резервной копии, перейдите на соответствующие вкладки ((1,2) Рис. 5.37)

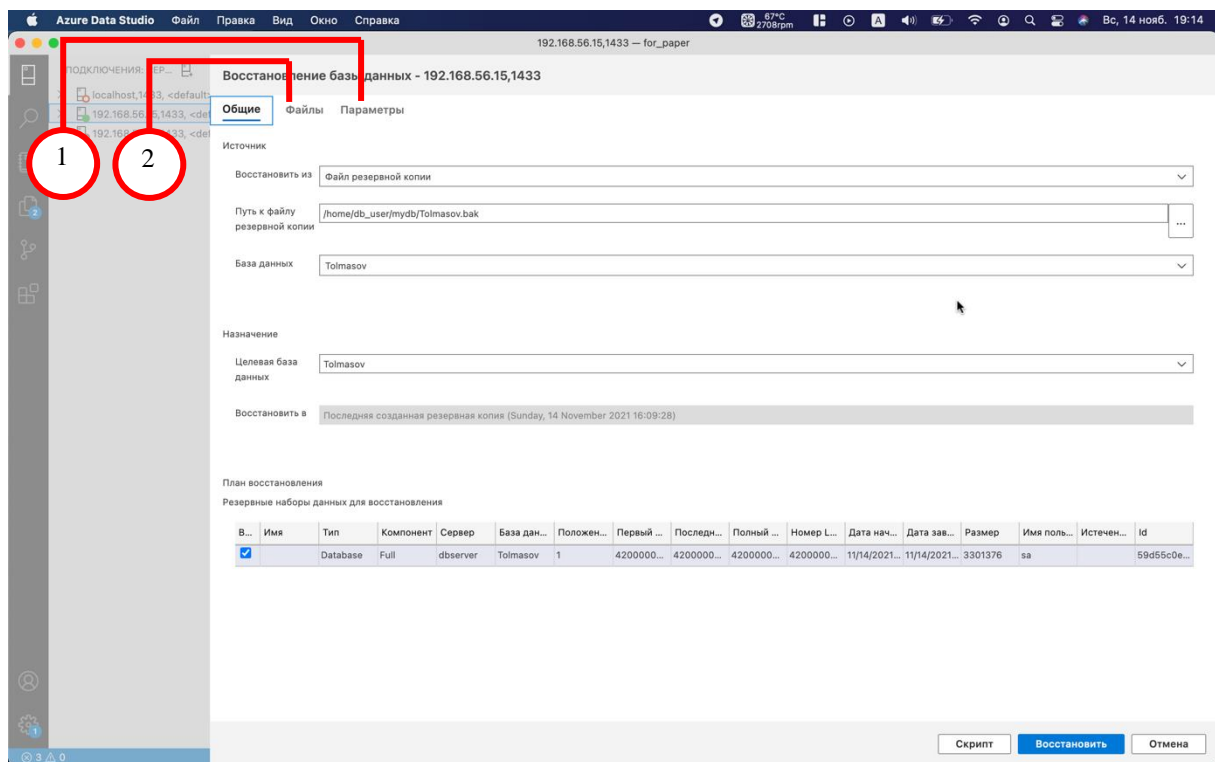


Рисунок 5.37 — Заполненная форма восстановления базы данных

Нажмите на кнопку «**Восстановить**» в правом нижнем углу формы для начала процесса восстановления. Об успешном окончании процесса можно узнать на вкладке «**Задачи**» окна терминала.

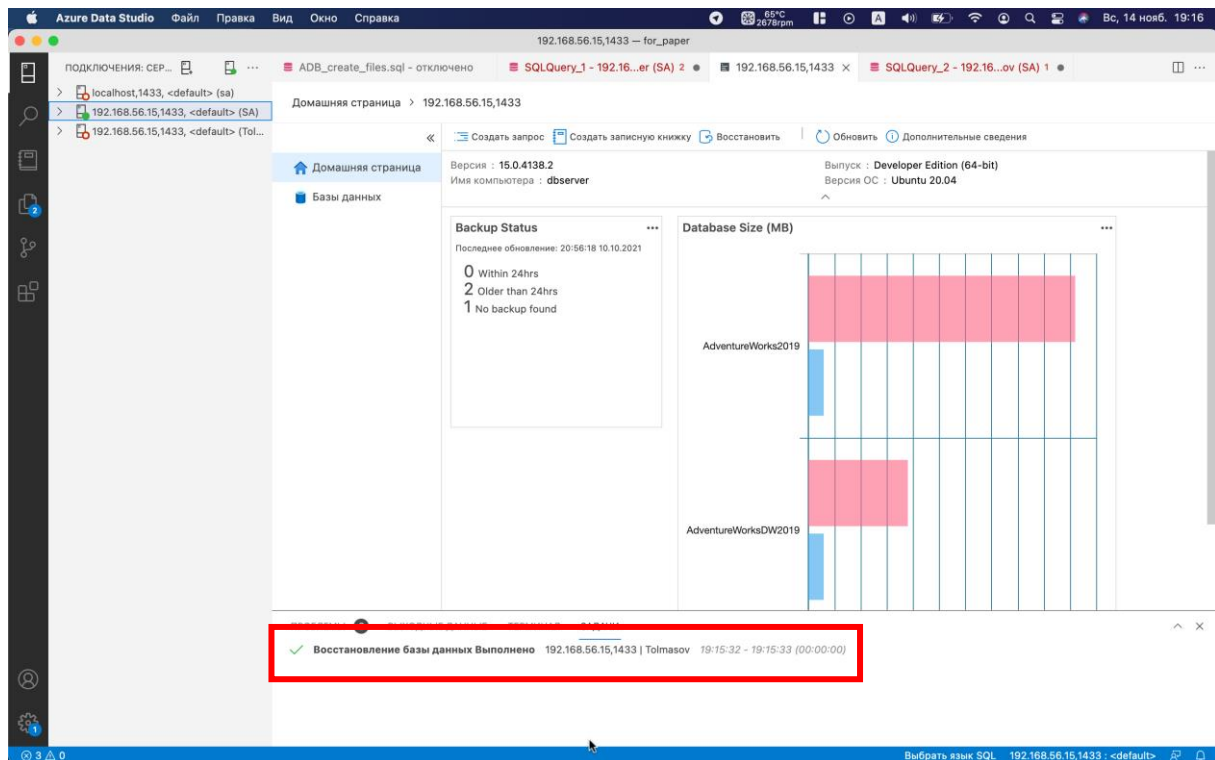


Рисунок 5.38 — Завершение восстановления базы данных

Процедура восстановления базы данных завершена.

5.3.2 Восстановление с помощью средств языка Transact-SQL

Проведем процедуру восстановления из полной резервной копии. Для этого необходимо воспользоваться инструкцией RESTORE DATABASE, которая в большинстве случаев имеет следующий синтаксис:

```
RESTORE DATABASE { database_name | @database_name_var }
[ FROM <backup_device> [ ,...n ] ]
[ WITH
    , <general_WITH_options> [ ,...n ]
]

<backup_device>::=
{
    { logical_backup_device_name |
        @logical_backup_device_name_var }
| { DISK
    | TAPE
    | URL
    } = { 'physical_backup_device_name' |
        @physical_backup_device_name_var }
}

--Опции операции размещения файлов
MOVE 'logical_file_name_in_backup' TO 'operating_system_file_name'
    [ ,...n ]
| REPLACE | RESTART | RESTRICTED_USER | CREDENTIAL

--Настройки файла бекапа
| FILE = { backup_set_file_number | @backup_set_file_number }

--Опции мониторинга
| STATS [ = percentage ]
```

С полным набором свойств можно ознакомиться на странице документации посвященной инструкции RESTORE.

Составим инструкцию восстановления базы данных из файла полной резервной копии с настройкой отслеживания состояния каждые 5%.

```
RESTORE DATABASE Tolmasov
FROM DISK = N'/home/db_user/mydb/Tolmasov.bak'
WITH FILE = 1, STATS = 5
```

В результате выполнения инструкции в консоль будет осуществляться следующий вывод (Рис. 5.39):

20:34:07 Начато выполнение запроса в строке 38

6 percent processed.

10 percent processed.

16 percent processed.

20 percent processed.

25 percent processed.

31 percent processed.

35 percent processed.

41 percent processed.

46 percent processed.

50 percent processed.

56 percent processed.

60 percent processed.

66 percent processed.

71 percent processed.

75 percent processed.

81 percent processed.

85 percent processed.

92 percent processed.

96 percent processed.

100 percent processed.

Processed 392 pages for database 'Tolmasov', file 'tolmasov' on file 1.

Processed 2 pages for database 'Tolmasov', file 'tolmasov_log' on file 1.

RESTORE DATABASE successfully processed 394 pages in 0.055 seconds (55.894 MB/sec).

Общее время выполнения: 00:00:00.587

Рисунок 5.39 — Вывод информации о ходе процесса восстановления

В данном разделе мы рассмотрели базовые вопросы выполнения резервного копирования и восстановления данных с применением различных программных средств и способов.

5.4 Контрольные задания

1. Оцените размер резервных копий системных баз данных. Выполните процедуру полной копии системных баз данных и с помощью файлового менеджера определите фактический размер. Определите среднюю погрешность метода.
2. Создайте две базы данных. Смените модель восстановления одной из баз данных на модель с полным протоколированием, а у другой на простую модель. Уточните размер журнала транзакций. Создайте в базах данных по таблице и заполните их случайными значениями не менее чем на 5000 тысяч строк. Уточните размер файла журнала транзакций ещё раз. Сравните результаты для разных моделей восстановления.
3. Выполните процедуру полного резервного копирования в файл, расположенный в отдельной папке, созданной специально для выполнения задания.
4. Повредите или удалите один из файлов базы данных, резервную копию

которой вы создали в предыдущем задании. Попробуйте обратиться к базе данных.

5. Выполните процедуру восстановления из полной резервной копии.
6. Найдите в официальной документации к MS SQL Server хранимые процедуры, таблицы, представления, которые могут быть полезны при выполнении операций резервного копирования и восстановления.
7. Используя хранимые процедуры изучите содержимое файлов резервных копий.
8. Внесите изменение в одну из созданных баз. Откатите изменения проведя процедуру восстановления базы данных.
9. Внесите изменения в базу данных и выполните разностную копию базы данных. Повторите процедуру ещё пять раз, затем удалите базу данных.
10. Выполните восстановление из разностной резервной копии.
11. Создайте новую полную копию базы данных. Передайте файл в ОС, установленной на виртуальной машине. Подключитесь к серверу на виртуальной машине и выполните процедуру восстановления базы данных.
12. Выполните копию журнала транзакций. Внесите изменения в базу. Выполните копию журнала транзакций, повторяйте процедуру не менее 5 раз.
13. Восстановите базу данных из полной копии и копии журналов транзакций.
14. Выполните резервное копирование в несколько файлов одновременно. Изучите содержимое обоих файлов. Подумайте, когда можно применить данный способ копирования.
15. Сделайте полную резервную копию базы данных *master* сервера виртуальной машины. Повредите системную базу данных мастер. Пользуясь документацией СУБД и полной резервной копией, восстановите работоспособность сервера.

СПИСОК РЕКОМЕНДОВАННЫХ ИСТОЧНИКОВ

1. Data Platphorm // Microsoft.com — URL: <https://www.microsoft.com/ru-ru/sql-server/sql-server-2019> (дата обращения: 10.11.2021)
2. Ubuntu Server Guide // Ubuntu.com — URL: <https://ubuntu.com/server/docs> (дата обращения: 10.11.2021)
3. Сайт проекта Losst // Losst.ru — URL: <https://losst.ru/about> (дата обращения: 10.11.2021)
4. Системная база данных *tempdb* // Microsoft.com — URL: <https://docs.microsoft.com/ru-ru/sql/relational-databases/databases/tempdb-database?view=sql-server-ver15> (дата обращения: 10.11.2021)
5. Файлы и файловые группы // Microsoft.com — URL: <https://docs.microsoft.com/ru-ru/sql/relational-databases/databases/database-files-and-filegroups?view=sql-server-ver15> (дата обращения: 10.11.2021)
6. Создание имени входа // Microsoft.com — URL: <https://docs.microsoft.com/ru-ru/sql/relational-databases/security/authentication-access/create-a-login?view=sql-server-ver15> (дата обращения: 10.11.2021)
7. Роли уровня сервера // Microsoft.com — URL: <https://docs.microsoft.com/ru-ru/sql/relational-databases/security/authentication-access/server-level-roles?view=sql-server-ver15> (дата обращения: 10.11.2021)
8. Инструкция CREATE USER // Microsoft.com — URL: <https://docs.microsoft.com/ru-ru/sql/t-sql/statements/create-user-transact-sql?view=sql-server-ver15> (дата обращения: 10.11.2021)
9. Модели восстановления // Microsoft.com — URL: <https://docs.microsoft.com/ru-ru/sql/relational-databases/backup-restore/recovery-models-sql-server?view=sql-server-ver15> (дата обращения: 10.11.2021)
10. Петкович, Душан. "Microsoft SQL Server 2012. Руководство для начинающих." СПб.: БХВ-Петербург (2013).

Сведения об авторах

1. Смирнов Михаил Вячеславович, кандидат экономических наук, доцент кафедры “Предметно-ориентированные информационные системы” Института комплексной безопасности и специального приборостроения РТУ-МИРЭА.
2. Толмасов Руслан Сергеевич, преподаватель кафедры «Предметно-ориентированные информационные системы» Института комплексной безопасности и специального приборостроения РТУ-МИРЭА.