Лекция 15.1 (основная)
Администрирование данных MS SQL Server

1. ОБЩИЕ СВЕДЕНИЯ О СУБД MS SQL SERVER

Информация о выпусках

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 бесплатно.

Загрузить нужный выпуск можно с официального сайта компании.

Таблица 1

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 тестирвание приложений баз данных.
EXPRESS	Бесплатный базовый выпуск СУБД, подходящий для приме- нения в обучении или создания приложений баз данных, предназначенных для локальной работы или работы на небольших серверах. Выпуск подходит для непрофессиональ-ных разработчиков, студентов, создающих клиентские приложения. При необходимости выпуск может быть расширен до выпусков более высокого класса.

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 SQLServer 2019:

Таблица 2

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

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

Таблица 3

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 экзем- пляры по умолчанию изолированной установки поддерживают следующие сетевые протоколы: Shared memory, Named Pipes и TCP/IP.

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 струк- турой (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 Server2016 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.

Системные базы данных MS SQL Server

База данных master

Системная база данных master является важной базой данных, повреждение которой **вызовет отказ всего сервера и потери доступа ко всем пользовательским базам данных**, расположенных 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 функцио- нирования 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***.

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.

База данных tempdb

Несмотря на своё название, которое ассоциируется с чем-то временным, 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 отличии от других системных баз **данных tempdb пересоздаётся каждый раз при запуске сервера.** Учитывая факт того, что сервер при каждом запуске пересоздаёт tempdb, резервное копирование и восстановление данной базы не целесообразно и отключено.

База данных msdb. База данных msdb используется в задачах автоматизации сервера. Она используется SQL-агентом сервера при создании расписаний предупреждений и заданий.

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.

Рассмотрим 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

три уровня модели безопасности:

уровень сервера
уровень базы данных
уровень схемы

Разрешение дополнительных соединений

Доступ на только что установленный экземпляр сервера обычно разрешен **для двух пользователей: *sa* 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 ниже примерах будут рассматриваться создание имён входа SQL, на сервере установлен смешанный способ проверки подлинности.

Создание имён входа

Авторизация пользователей на сервере осуществляется с помощью специальных субъектов — LOGIN.

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».

Имя входа даёт пользователям возможность подключаться к всему серверу 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 проверятся средствами SQL Server.

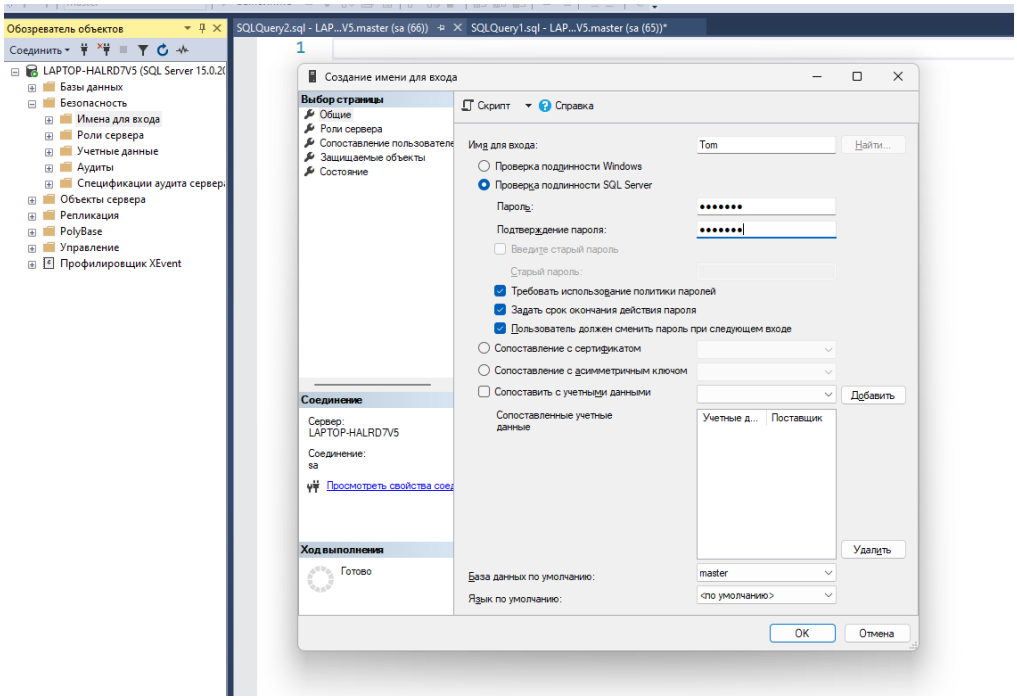
Создание имён входа с помощью MS SSMS

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.

Имена входа относятся к субъектам безопасности сервера, следовательно их можно найти в разделе «***Безопасность***».

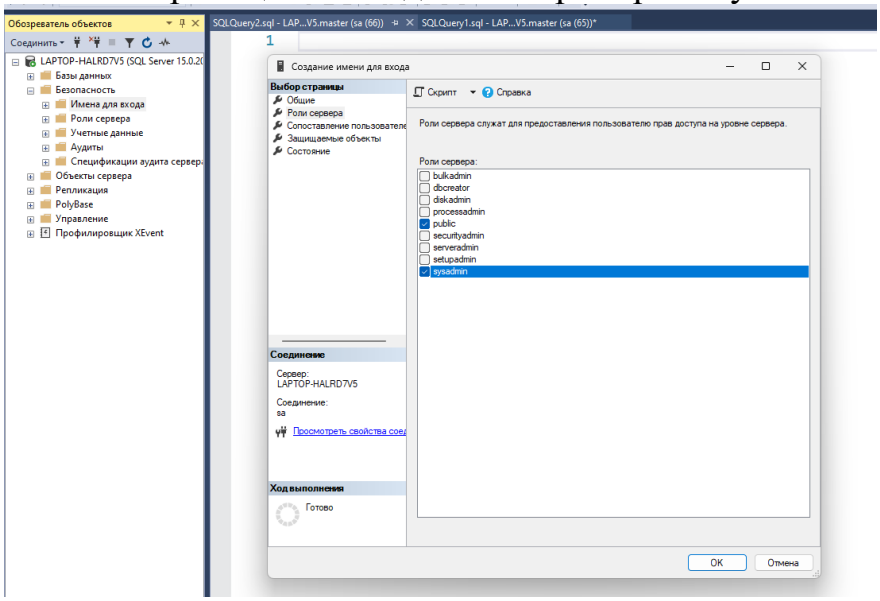
Необходимо найти в обозревателе объектов папку «***Безопасность***» и нажать на ней правой кнопкой мыши. В открывшемся контекстном меню выбрать «***Создать / Вход…»***

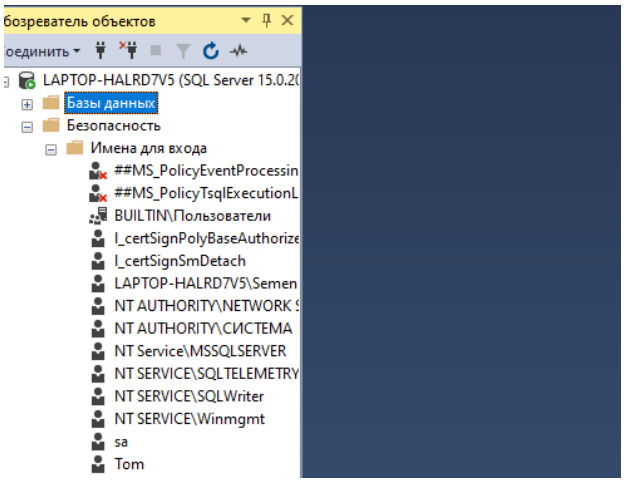


Для создания имени входа, с требуемыми параметрами, необходимо ввести уникальный (среди имён входа данного экземпляра сервера) идентификатор. Выбрать вариант способа проверки подлинности. Для имён входа, выдаваемых людям, применяется два основных способа: «Проверка подлинности Windows» и «Проверка подлинности SQL Server». Так как по условию необходимо проверить подлинность паролем, то выбираем второй вариант. В ставших доступными полях ввода, 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».

На этой странице необходимо выбрать роль sysadmin/





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 TOM1 WITH
PASSWORD=’<password_word>’MUST_CHANGE,
DEFAULT_DATABASE = master,
CHECK_POLICY = ON, CHECK_EXPIRATION = ON
ALTER SERVER ROLE sysadmin ADD MEMBER TOM1
```

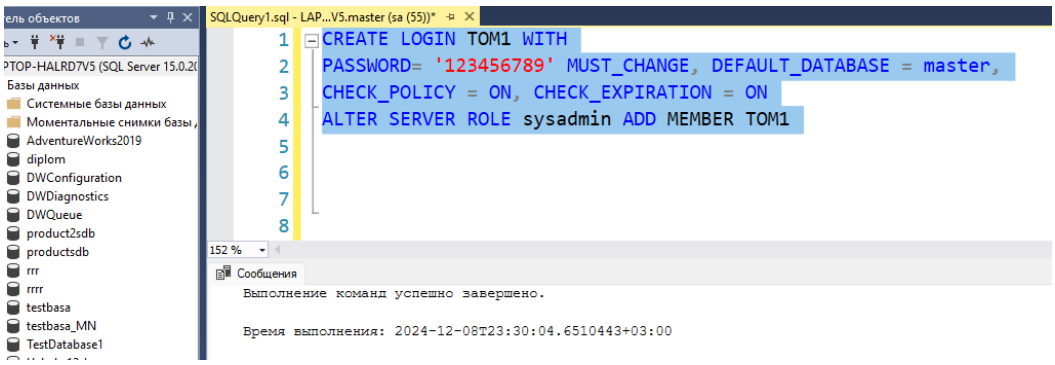
Выполнение данного скрипта приведёт к результатам, аналогичным полученным с помощью MS SSMS.

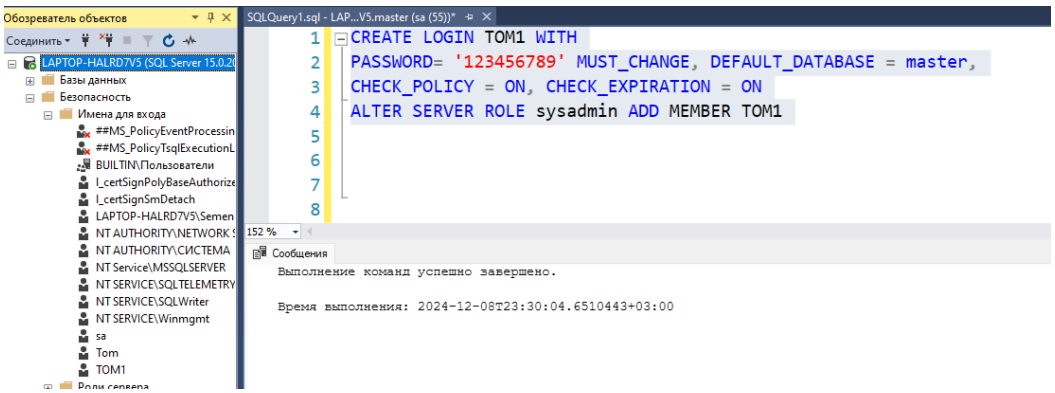
Рассмотрим построчно свойства, заданные после ключевого слова WITH:

- PASSWORD — задаём способ проверки с помощью пароля, что автоматически определяет способ проверки подлинности средствами SQL Server. Свойство MUST_CHAHGE 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.





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 пред-определённые роли и их краткое описание.

Таблица 5

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 экземпляр 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.

Серверные роли, определяемые пользователем

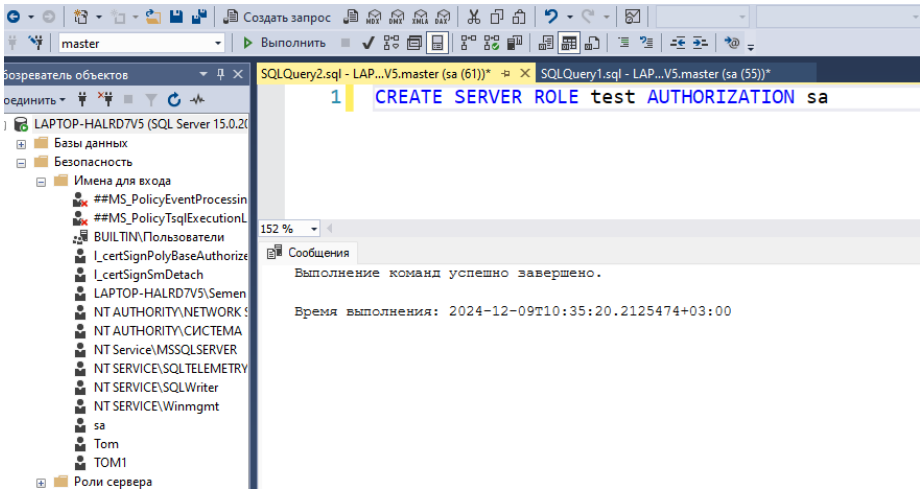
При необходимости пользователь **<u>может определять собственные серверные роли.</u>** Рассмотрим синтаксис инструкции для создания ролей, определяемых пользователем:

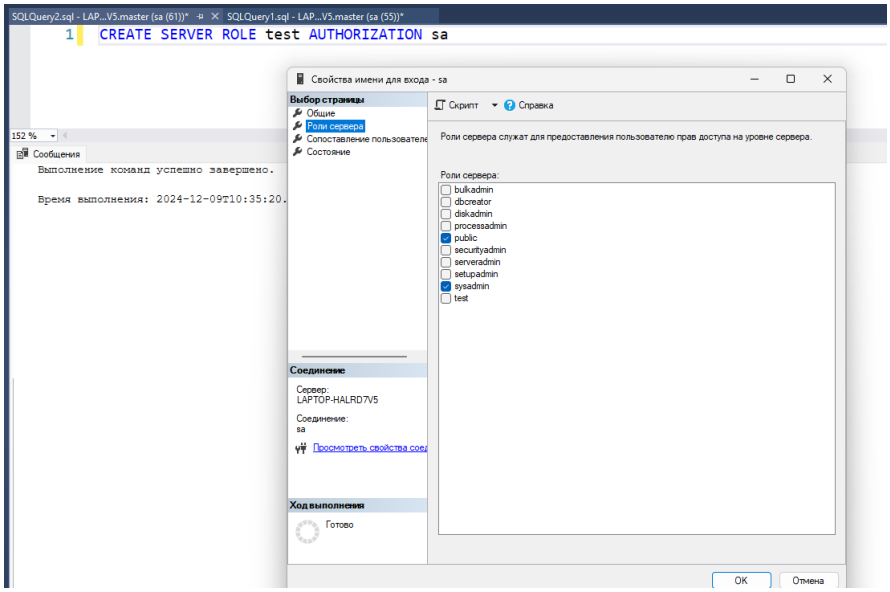
```
CREATE SERVER ROLE role_name
[ AUTHORIZATION server_principal ]
```

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 (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:

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

Для удаления созданной серверной роли следует воспользоваться инструк- цией DROP языка *DDL*:

```
DROP SERVER ROLE Test
```

Управление доступом к данным

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.

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 авторизированным 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.***

Пример

Найдите в списке базу данных, в которой требуется создать пользователя и разверните список объектов базы данных.

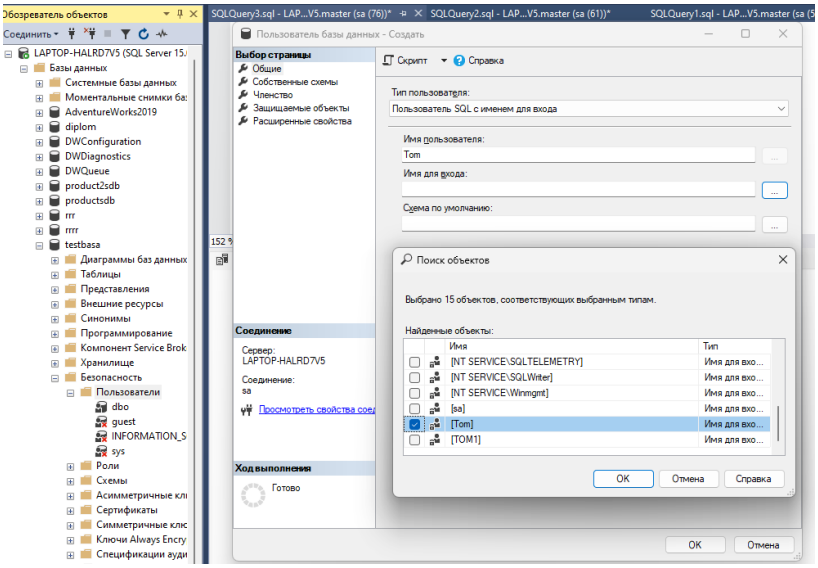
Щелкните правой кнопкой мыши по папке «***Безопасность***» базы данных. В контекстном меню необходимо выбрать «***Создать / Пользователь…***».

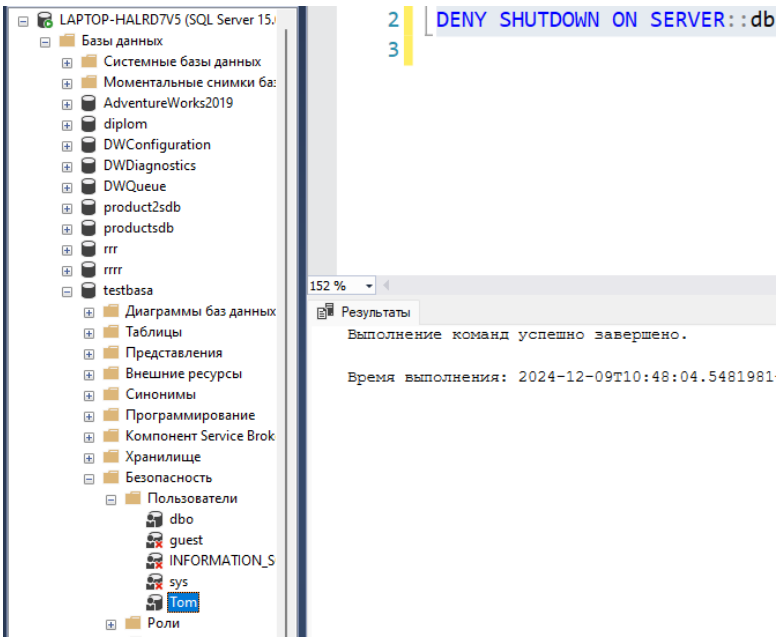
В диалоговом окне необходимо обязательно заполнить поля «***Имя пользователя***» и «***Имя для входа***».

Имя для входа лучше выбрать с помощью диалогового окна, чтобы избе- жать 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.

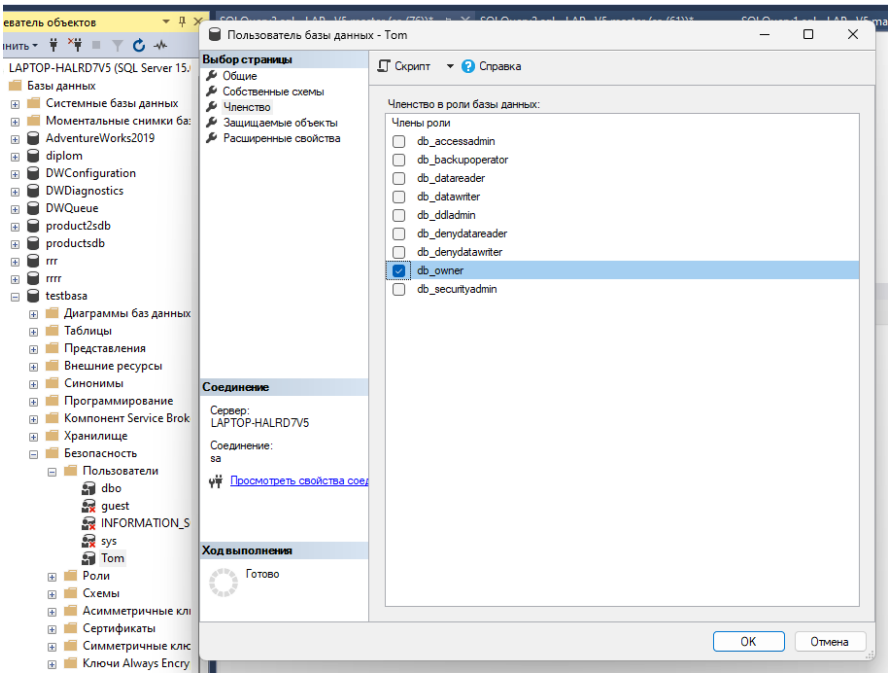
В окне «***Выбор имени для входа***» нажмите на кнопку «***Обзор***».

В следующем диалоговом окне необходимо найти созданное ранее имя входа и выбрать его. После этого нажать «Ок». 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***». Выбираем роль *db_owner* (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

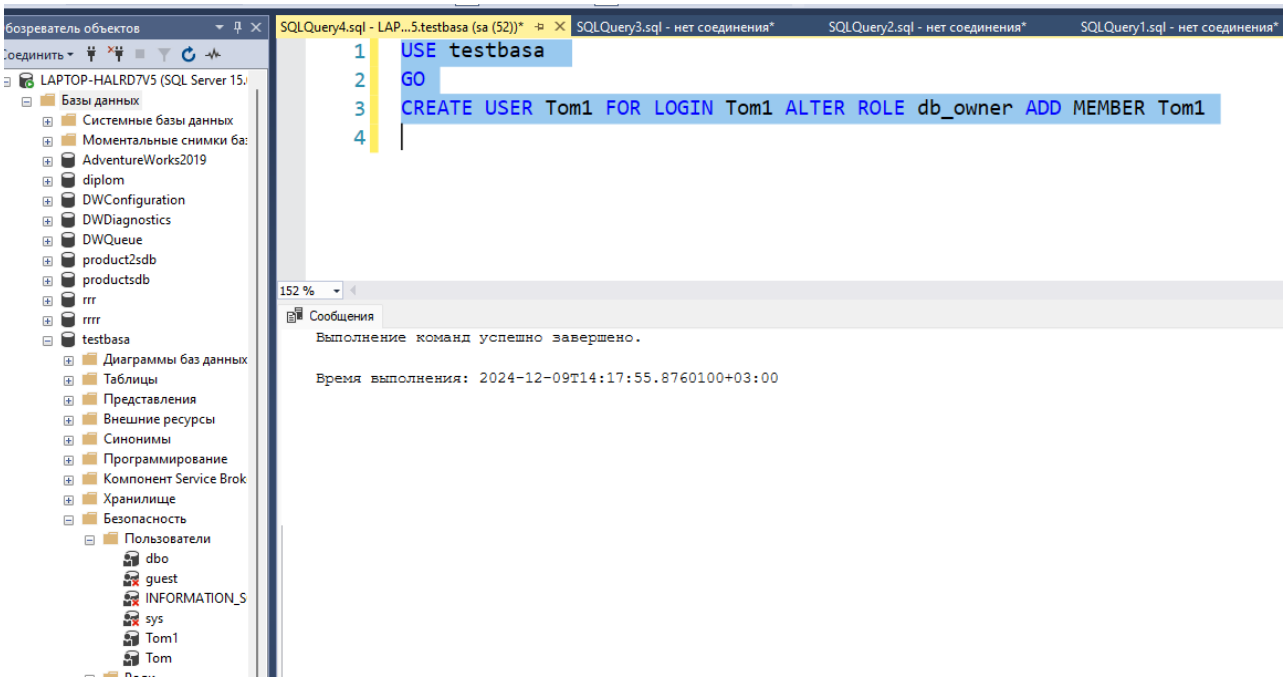
Аналогичные действия можно выполнить с помощью скрипта.

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 testbasa GO
CREATE USER Tom1 FOR
LOGIN Tom1 ALTER ROLE
db_owner ADD MEMBER
Tom1
```



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 таб

Таблица

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

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 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
    }
```

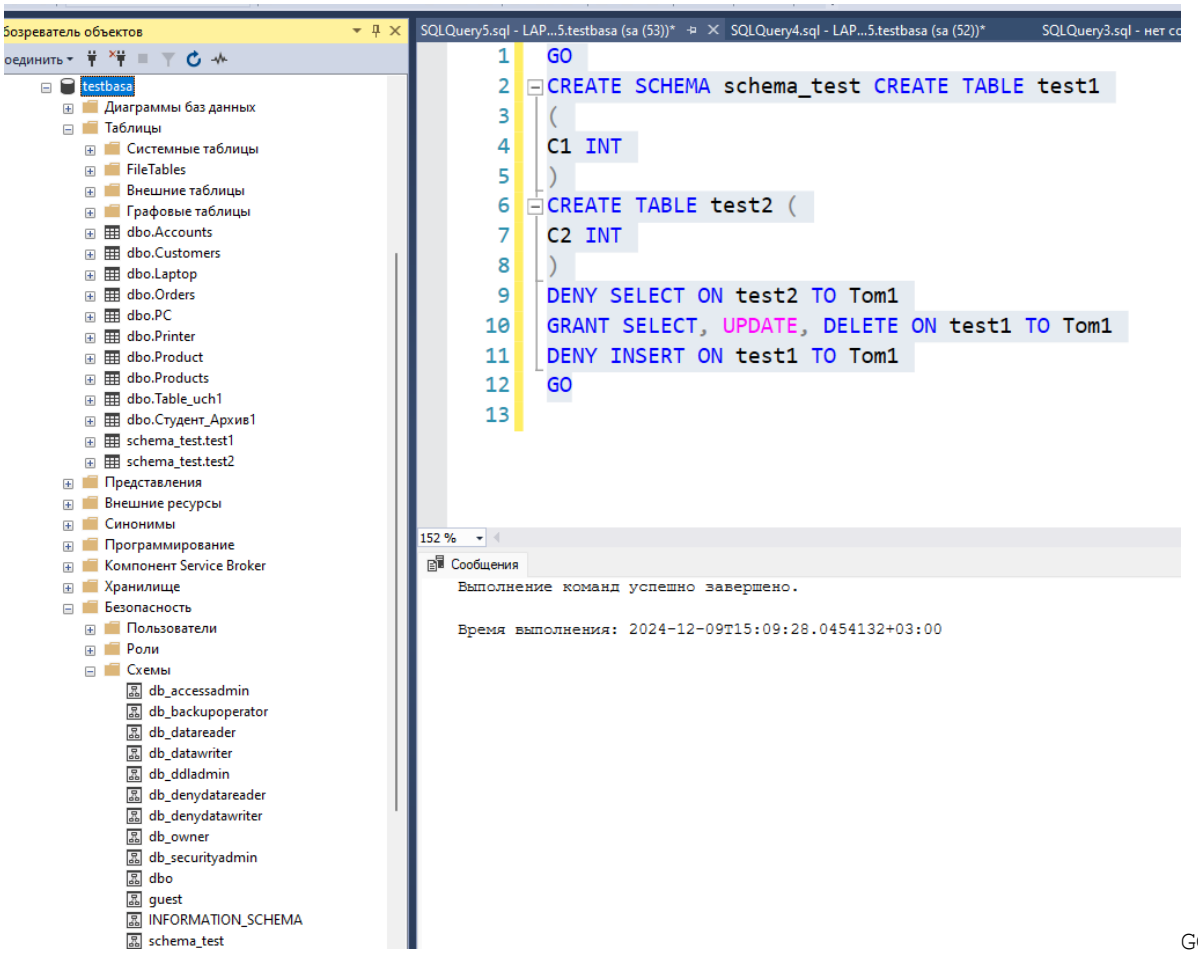
Следует сделать ряд замечаний по оформлению и выполнению инструкции:

1) *Объявление схемы должно быть единственной инструкцией в пакете (на входящие в схему защищаемые объекты и настройку прав доступа ограничение не распространяется).*
2) *В состав инструкции входит все что указанно между* `CREATE SCHEMA` *и ближайшим* `GO` *или концом файла.*
3) *Транзакция* `CREATE SCHEMA` *является атомарной, если в процессе выполнения произойдёт 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.*

ПРИМЕР

Создадим схему в базе данных *testbasa* 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

Права можно также предоставлять и ролям.

Для удаления схемы необходимо предварительно удалить все защищаемые ей объекты.

После чего использовать инструкцию *DDL*

`DROP SCHEMA` Имя схемы

РЕЗЕРВНОЕ КОПИРОВАНИЕ И ВОССТАНОВЛЕНИЕ БАЗ ДАННЫХ

Резервное копирование и восстановление играет важнейшую роль в обеспечении надёжной защиты и хранения данных.

Модели восстановления

Виды моделей восстановления

Резервное копирование тесно связано с *моделью восстановления* 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».**

СУБД поддерживает три модели восстановления: **простая, полная и модель восстановления с неполным протоколированием.**

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 использовании простой модели восстановления изменения, произошедшие с мо- мента 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.**

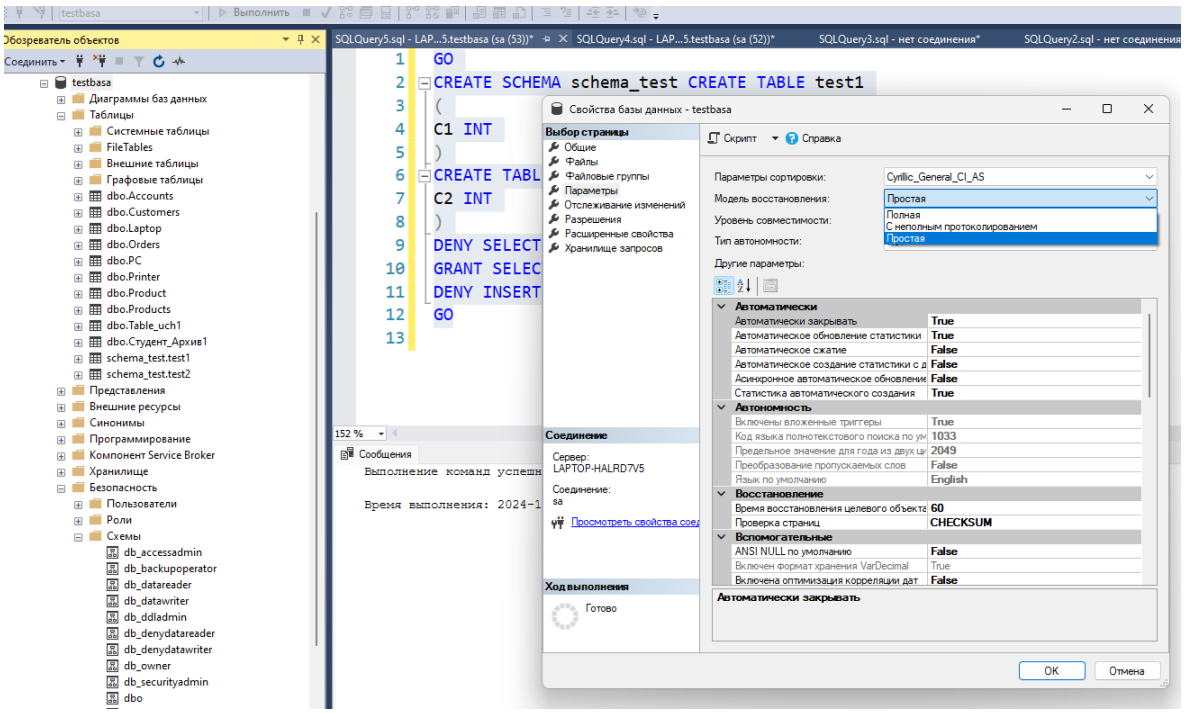
Используя модель с неполным протоколированием, администратор может добиться уменьшения занимаемого журналами дискового пространства, так как к большинству массовых операций будет применено минимальное протоколирование. Опытные администраторы стараются не применять эту модель на постоянной основе, включая 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.

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***»


Обозреватель объектов
Соединить
testbasa
Диаграммы баз данных
Таблицы
Системные таблицы
FileTables
Внешние таблицы
Графовые таблицы
dbo.Accounts
dbo.Customers
dbo.Laptop
dbo.Orders
dbo.PC
dbo.Printer
dbo.Product
dbo.Products
dbo.Table_uch1
dbo.Студент_Архив1
schema_test.test1
schema_test.test2
Представления
Внешние ресурсы
Синонимы
Программирование
Компонент Service Broker
Хранилище
Безопасность
Пользователи
Роли
Схемы
db_accessadmin
db_backupoperator
db_datareader
db_datawriter
db_ddladmin
db_denydatareader
db_denydatawriter
db_owner
db_securityadmin
dbo
SQLQuery5.sql - LAP...5.testbasa (sa (53))*
SQLQuery4.sql - LAP...5.testbasa (sa (52))*
SQLQuery3.sql - нет соединения*
SQLQuery2.sql - нет соединения
GO
CREATE SCHEMA schema_test CREATE TABLE test1
(
C1 INT
)
CREATE TABL
C2 INT
)
DENY SELECT
GRANT SELEC
DENY INSERT
GO
152 %
Сообщения
Выполнение команд успешн
Время выполнения: 2024-1
Свойства базы данных - testbasa
Выбор страницы
Общие
Файлы
Файловые группы
Параметры
Отслеживание изменений
Разрешения
Расширенные свойства
Хранилище запросов
Скрипт
Справка
Параметры сортировки:
Cyrillic_General_CI_AS
Модель восстановления:
Простая
Полная
С неполным протоколированием
Простая
Уровень совместимости:
Тип автономности:
Другие параметры:
Автоматически
Автоматически закрывать
True
Автоматическое обновление статистики
True
Автоматическое сжатие
False
Автоматическое создание статистики с д
False
Асинхронное автоматическое обновлени
False
Статистика автоматического создания
True
Автономность
Включены вложенные триггеры
True
Код языка полнотекстового поиска по ум
1033
Предельное значение для года из двух ци
2049
Преобразование пропускаемых слов
False
Язык по умолчанию
English
Восстановление
Время восстановления целевого объекта
60
Проверка страниц
CHECKSUM
Вспомогательные
ANSI NULL по умолчанию
False
Включен формат хранения VarDecimal
True
Включена оптимизация корреляции дат
False
Автоматически закрывать
Соединение
Сервер:
LAPTOP-HALRD7V5
Соединение:
sa
Просмотреть свойства сое
Ход выполнения
Готово
OK
Отмена

Виды резервных копий

В *MS SQL Server* поддерживается три вида резервных копий баз данных (ре- зервные 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.

Таблица 7

Вид резервной копии	Описание
Полная	Создаётся копия всей базы данных целиком. Включает часть журнала транзакций. Отражает состояние БД на момент завершения резервного копирования
Разностная	Включает только изменённую часть БД с момента последнего резервного копирования
Журнала транзакций	Включает в себя копию всех записей журнала транзакций, которые были сделаны после последней резервной копии журнала.

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