Microsoft SQL Server 2017 для поддержки системы «1С:Предприятие 8»: администрирование, оптимизация, обеспечение безопасности

Учебные материалы

Подготовлено с использованием материалов Microsoft и 1С
Москва 2019

Тел.: (499)253-58-38, (495)542-19-94

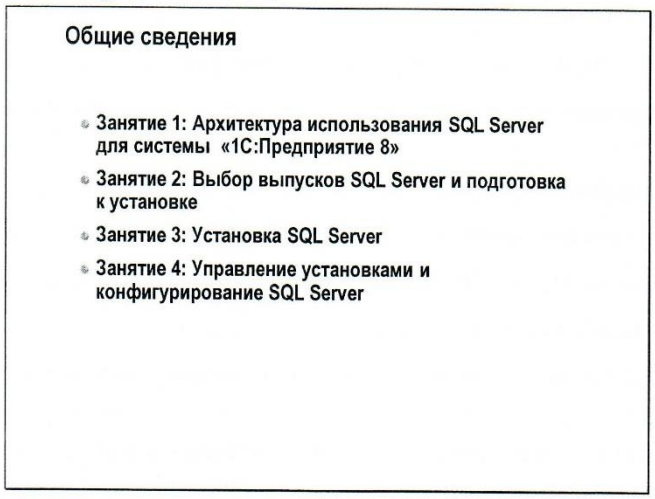
Автор методических материалов – Крамарская Т. А.

По вопросам совершенствования методических материалов
просьба обращаться в
ООО «1С-Учебный центр №3» www.1c-uc3.ru, uc3@1c.ru
ООО «АЛЕСТАСОФТ» www.alesta.ru, marketing@alesta.ru

Содержание

Microsoft SQL Server 2017 для поддержки системы «1С:Предприятие 8»: администрирование, оптимизация, обеспечение безопасности

Раздел 1: Установка SQL Server 2017



Для обслуживания системы «1С:Предприятие 8» важно, чтобы администраторы познакомились с требованиями для установки сервера SQL Server, процедурами добавления и удаления компонентов SQL Server и проблемами сосуществования с предыдущими версиями. В этом разделе слушатели узнают, как планировать и выполнять установку SQL Server 2017 и проводить последующее обслуживание. Они также узнают о средствах администрирования SQL Server 2017, включая диспетчер конфигурации SQL Server, среду SQL Server Management Studio и служебную программу sqlcmd.

Цели

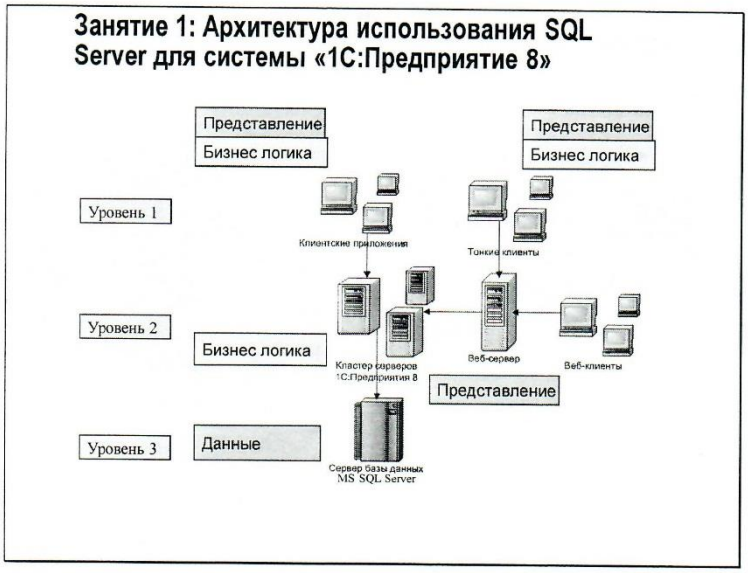
После изучения данного раздела вы сможете:

• объяснить архитектуру использования SQL Server для системы «1С:Предприятие 8»;

• объяснить, как подготовить оборудование и другие необходимые ресурсы к установке SQL Server 2017;

• установить SQL Server 2017;

• управлять сервером SQL Server 2017 и настраивать его.



Клиент-серверный вариант системы «1С:Предприятие 8» предназначен для использования в рабочих группах или в масштабе предприятия. Он реализован на основе трехуровневой архитектуры «клиент-сервер».

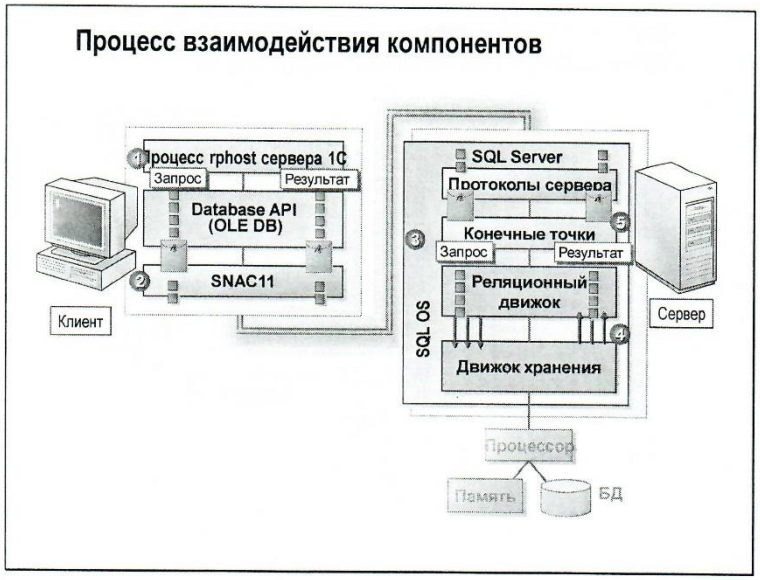
Программа, работающая у пользователя, (клиентское приложение) взаимодействует с кластером серверов, а кластер, обращается к серверу баз данных SQL Server. При этом физически кластер серверов системы «1С:Предприятие 8» и сервер баз данных могут располагаться как на одном компьютере, так и на разных. Это позволяет администратору при необходимости распределять нагрузку между серверами. Использование кластера серверов позволяет сосредоточить на нем выполнение наиболее объемных операций по обработке данных. Например, при выполнении даже весьма сложных запросов программа, работающая у пользователя, будет получать только необходимую ей выборку, а вся промежуточная обработка будет выполняться на сервере. Обычно увеличить мощность кластера серверов гораздо проще, чем обновить весь парк клиентских машин.

Другим важным аспектом использования 3-х уровневой архитектуры является удобство администрирования и упорядочивание доступа пользователей к информационной базе. В этом варианте пользователь не должен знать о физическом расположении конфигурации или базы данных. Весь доступ осуществляется через кластер серверов системы «1С:Предприятие 8». При обращении к той или иной информационной базе пользователь должен указать только имя кластера и имя информационной базы, а система запрашивает соответственно имя и пароль пользователя.

Система «1С:Предприятие 8» использует возможности SQL Server для эффективной выборки информации:

- механизм запросов ориентирован на максимальное использование MS SQL Server для выполнения расчетов и составления отчетов;
- просмотр больших динамических списков обеспечивается без выполнения большого количества обращений к базе данных; при этом пользователю предоставляются возможности эффективного поиска, а также настройки отбора и сортировки.

В системе «1С:Предприятие 8» клиентом СУБД SQL Server является кластер серверов.



Интерфейс, используемый системой «1С:Предприятие 8» для работы с SQL Server

Приложения для работы с реляционными БД обращаются к SQL Server при помощи интерфейса прикладного программирования БД (database API), который определяет на уровне кода приложения, каким образом это приложение будет подключаться к SQL Server и передавать команды в СУБД. Система «1С:Предприятие 8» традиционно использовала интерфейс OLE DB, входящий в MDAC и ориентированный на SQL Server 2000. OLE DB — это интерфейс прикладного программирования, позволяющий приложениям, использующим технологию COM, использовать данные из источников данных OLE DB. OLE DB-поставщик представляет собой COM-компонент, который получает вызовы, адресованные интерфейсу прикладного программирования OLE DB, и выполняет все необходимые действия по обработке запроса к источнику данных. Этот поставщик поддерживает приложения, написанные с использованием технологии OLE DB или других интерфейсов прикладного программирования, использующих OLE DB, например ADO. SQL Server 2017 поддерживает такое подключение. OLE DB-поставщик использует клиентскую сетевую библиотеку, или клиентский протокол, для обмена данными с серверной сетевой библиотекой из состава SQL Server 2017. Обмен данными может выполняться как на одном компьютере, так и по сети. Сетевые библиотеки инкапсулируют запросы, которыми обмениваются клиентские компьютеры и серверы, для последующей передачи этих запросов в нижележащий сетевой протокол. Обмен данными может осуществляться с шифрованием по протоколу Secure Sockets Layer (SSL). Релизы платформы, ориентированные на использование SQL Server 2017, требуют использования собственного клиента (SNAC).

Собственный клиент (SNAC) и сетевые протоколы

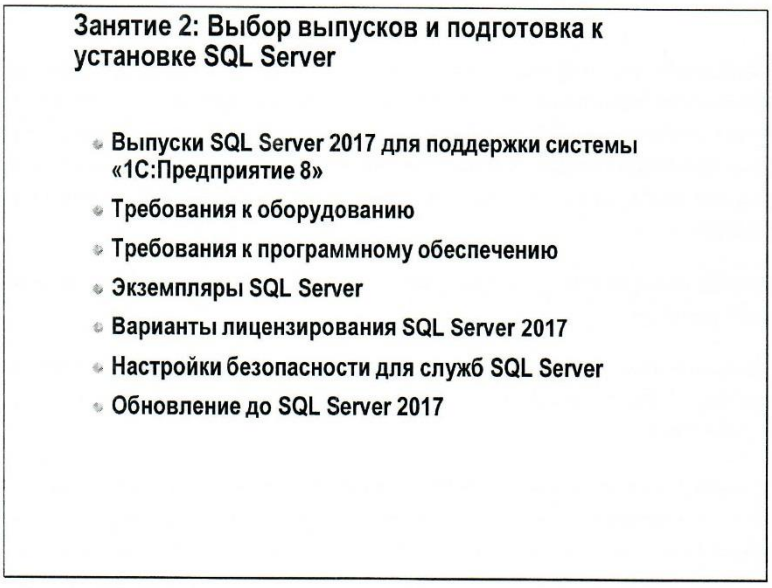
На клиентской машине должен быть установлен собственный клиент SQL Server 2017 (SNAC), содержащий в одной динамической библиотеке SQLNCLI11.DLL OLE DB и ODBC. Собственный клиент SQLNCLI11.DLL один для SQL Server 2017 - 2012.

SQL Server 2017– это реляционная СУБД. В основе реляционной базы данных лежит идея о связанных двумерных таблицах, состоящих из строк (записей) и столбцов (полей). При создании нескольких таблиц со связанной информацией можно выполнять сложные и мощные операции над данными. Мощность базы данных заключается, скорее в связях между частями информации, чем в самих частях. Программы, обрабатывающие реляционные базы данных, были созданы для работы с большими и сложными наборами тех данных, которые являются наиболее общими в деловой жизни общества. Даже если база данных содержит десятки и тысячи элементов, единственная команда SQL предоставит необходимую информацию практически мгновенно.

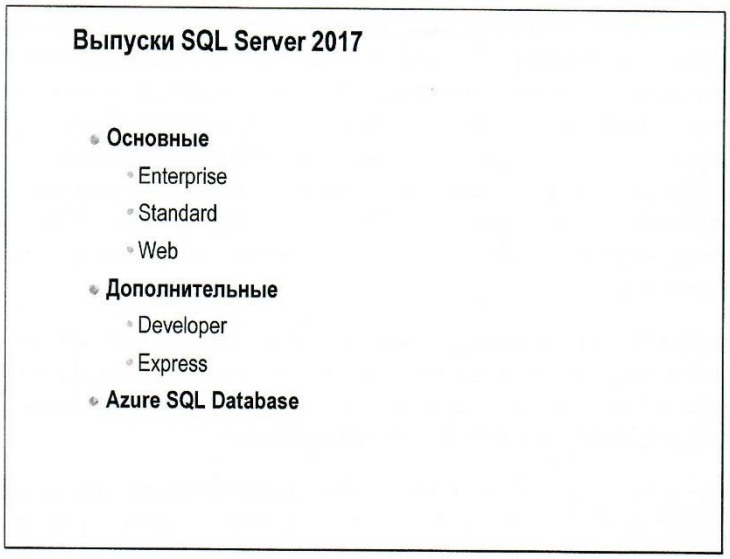
Программирование в реляционной модели не является процедурным, и программа оперирует множеством строк. Команды, применяемые для доступа к данным, вставки и изменения данных, описывают всего лишь результирующее множество. Ориентированность на множества упрощает получение сразу нескольких записей из реляционной базы.

Реляционный подход привел к созданию структурированного языка запросов Structured Query Language (SQL). Реализация языка SQL в Microsoft SQL Server называется Transact-SQL (T-SQL).

SQL - это язык, ориентированный специально на реляционные базы данных. Он позволяет исключить большую работу, выполняемую при исполнении языка программирования общего назначения. Команды SQL могут выполняться над целой группой таблиц, как над единственным объектом, а также могут оперировать любым количеством информации, которая извлекается или выводится из них как из единого целого.



Перед установкой SQL Server 2017 следует убедиться в наличии необходимого оборудования и программного обеспечения и определить необходимые вашей организации варианты установки и ее сценарий. На этом занятии анализируются различные требования, о которых необходимо знать до начала установки SQL Server 2017. Знакомство с этими требованиями и вариантами установки поможет в ее планировании.



Имеется несколько выпусков SQL Server 2017, каждый из которых предназначен для конкретной среды или определенной задачи. Все выпуски только 64-разрядные. Важно понимать различия между имеющимися выпусками, чтобы можно было выбрать тот, который лучше всего отвечает поставленным требованиям. Краткое описание выпусков приведено ниже.

Enterprise. Полноценная производительность обработки в памяти для критически важных приложений, беспрецедентный уровень безопасности, комплексное решение корпоративной бизнес-аналитики со встроенными возможностями мобильной бизнес-аналитики, а также расширенная аналитика внутри базы данных в требуемом масштабе. Выпуск Enterprise Edition обеспечивает высочайшие уровни обслуживания и производительности. Дополнительно имеется пробный 180-дневный выпуск SQL Server 2017 Enterprise Edition.

Standard. Ключевые возможности управления данными и бизнес-аналитикой с использованием минимальных ИТ-ресурсов.

Web. Безопасная, экономичная и в высокой степени масштабируемая платформа для обработки данных общедоступных вебсайтов. Выпуск Web Edition предоставляется только сторонним поставщикам программных продуктов.

Developer. Обладает всеми возможностями выпуска Enterprise Edition, но лицензируется для использования в системах разработки и тестирования, а не в качестве производственного сервера. Этот выпуск, доступный бесплатно зарегистрированным разработчикам, следует применять для разработки и тестирования решений баз данных. Выпуск можно обновить до выпуска Enterprise Edition, который можно использовать для производственных целей.

Express. Бесплатная версия SQL Server 2017 для неподключенных клиентов и изолированных приложений с объемом базы до 10ГБ, использует только 1410Мб оперативной памяти и 4 ядра.

Azure SQL Database. Версия для облачных вычислений

Требования к оборудованию для SQL Server 2017

Оборудование	Требования
Процессор 	• x64: 1,4 ГГц и выше • Рекомендуется 2 ГГц или выше
Память 	• Выпуски Enterprise, Developer, и Standard: **минимум 1 ГБ.** рекомендуется 4 Гб и выше • Выпуск Express Edition **512** МБ , рекомендуется 1Гб
Диск 	• Свободное место на диске 6 ГБ

При планировании установки SQL Server 2017 следует убедиться, что компьютер, на котором будет устанавливаться SQL Server, отвечает минимальным требованиям, предъявляемым к оборудованию, и обеспечивает текущие и будущие потребности организации. Несоответствие минимальным требованиям может воспрепятствовать успешной установке некоторых или всех компонентов.

Требования к процессору

Специалисты корпорации Майкрософт рекомендуют использовать процессор, работающий на частоте 2 ГГц и выше. Для SQL Server 2017 следует использовать процессор, работающий на частоте не менее 1,4 ГГц.

Требования к памяти

У SQL Server 2017 должно быть не менее 1 ГБ памяти дополнительно к той, что требуется для операционной системы. Специалисты корпорации Майкрософт рекомендуют не менее 4 ГБ памяти.

Примечание. Приведенные здесь требования к памяти относятся только к SQL Server 2017. В них не предусмотрены ресурсы памяти, необходимые для операционной системы и другого программного обеспечения, установленного на компьютере.

Требования к жесткому диску

Обычно для установки требуется 6 ГБ свободного места на диске

Виртуализация. SQL Server 2017 поддерживается в среде виртуальных машин.

Требования к программному обеспечению для SQL Server 2017

Операционная система	Express	Standard	Enterprise	Developer
Windows Server 2016, 2019	✓	✓	✓	✓
Windows Server 2012, 2012 R2	✓	✓	✓	✓
Windows 10, Windows 8, Windows 8.1	✓	✓		✓
Red Hat Enterprise Linux 7.3 или 7.4 SUSE Enterprise Linux Server v12 SP2 Ubuntu 16.04 LTS	✓	✓	✓	✓
Подсистема docker 1.8 + в Windows, Mac или Linux	✓	✓	✓	✓

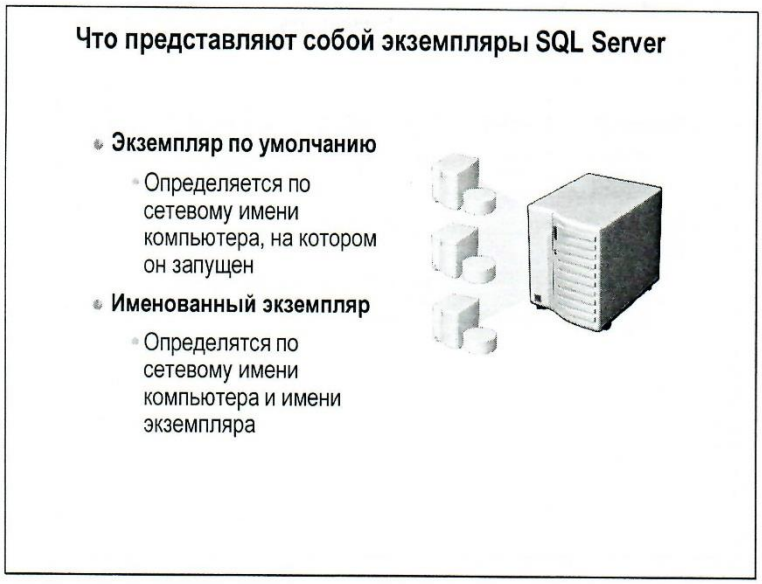
SQL Server 2017 следует устанавливать на компьютер, работающий под управлением Microsoft Windows или Linux. SQL Server 2017 можно также установить в виде контейнера в подсистеме docker 1.8 и выше. Требования относительно конкретных версий операционной системы Windows зависят от устанавливаемого выпуска SQL Server 2017.

В таблице, показанной на рисунке, перечислены имеющиеся выпуски SQL Server 2017 и операционные системы, которые их поддерживают.

Требования к дополнительному программному обеспечению

Для SQL Server 2017 требуется также следующее программное обеспечение:

- Поддержка сети TCP/IP.
- Microsoft .NET Framework 3.5 SP1 (Разрешить в операционной системе)
- Microsoft .NET Framework 4.6 (SQL Server при необходимости выполнит установку).
- Установщик Windows 4.5 (SQL Server при необходимости выполнит установку).
- Windows PowerShell 4.0 или 5.0 в Windows Server 2016.
- Пакет обновления 1 (SP1) для компонентов доступа к данным WDAC 2.8 или более поздняя версия.



Установка SQL Server 2017 может содержать один или несколько отдельных экземпляров. Экземпляр ядра СУБД SQL Server, используемый по умолчанию или именованный, имеет свой собственный набор специфичных для данного экземпляра файлов программ и данных, а также набор общих файлов, используемых всеми экземплярами, находящимися на данном компьютере. Каждый экземпляр работает независимо от других экземпляров, находящихся на том же компьютере, и приложения могут подключаться к любому из этих экземпляров.

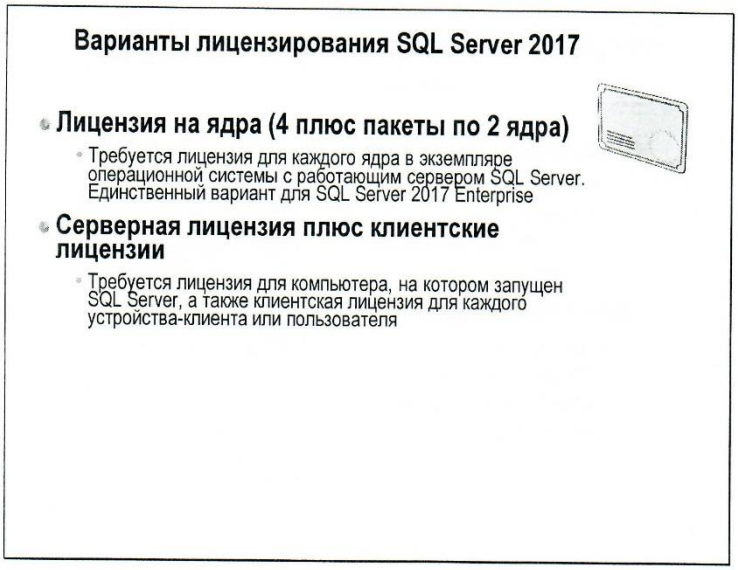
Экземпляр по умолчанию

Этот экземпляр идентифицируется сетевым именем компьютера, на котором он запущен. Именем экземпляра по умолчанию службы SQL Server является MSSQLSERVER.

Именованные экземпляры

Именованные экземпляры идентифицируются сетевым именем компьютера, к которому добавляется имя экземпляра с использованием формата *имя_компьютера\имя_экземпляра*, например MOSCOW\SQLINSTANCE2 для экземпляра SQLINSTANCE2 на компьютере MOSCOW. Имя нового экземпляра должно начинаться с буквы или со знака подчеркивания и может содержать цифры, буквы и другие символы. Именованные экземпляры содержат разные наборы служб и могут иметь различные настройки порядка сортировки, безопасности и других параметров. В структуре каталогов, структуре реестра и именах служб отражаются заданные имена экземпляров. В частности, имя службы SQL Server для именованного экземпляра имеет вид MSSQL$имя_экземпляра. Например, служба для экземпляра с именем SQLINSTANCE2 называется MSSQL$SQLINSTANCE2.

Примечание. Для поддержки системы «1С:Предприятие 8» именованный экземпляр может быть необходим для экспериментов или предоставления отдельной базы tempdb. Максимальное количество лицензионных экземпляров на одном компьютере равно 50.



Для установок SQL Server, используемых при выполнении повседневных операций, требуются производственные лицензии. В настоящее время имеется три варианта лицензирования SQL Server: *лицензия на ядра, серверная лицензия плюс клиентские лицензии на устройство или на пользователя*.

Лицензия на ядра

Эта модель лицензирования предусматривает необходимость одной лицензии для каждого ядра, доступного для операционной системы, в которой выполняется экземпляр SQL Server. Минимальное число ядер 4, далее пакеты по 2. При использовании этой лицензии не требуются клиентские лицензии ни на устройство, ни на пользователя. Такая модель лицензирования лучше всего подходит для приложений, доступ к которым осуществляется через Интернет, или для приложений внутреннего применения с высоким отношением количества клиентов к количеству серверов. Для SQL Server Enterprise это единственный способ лицензирования.

Серверная лицензия плюс клиентские лицензии на устройство или пользователя

При применении этой модели лицензирования лицензия для компьютера, на котором запущен SQL Server, а также клиентская лицензия для каждого клиентского устройства или пользователя. В соответствии с лицензионным соглашением, если подключающиеся устройства или программы играют роль мультиплексора, то необходимое количество лицензий увеличивается. Количество лицензий, необходимых системе «1С:Предприятие 8», определяется количеством машин с установкой 1С, использующих базы на сервере предприятия.

Виртуализация

Лицензировать можно отдельную виртуальную машину по любой модели и приобрести SA (Software Assurance), что позволяет легко переносить виртуальные машины на новый сервер виртуализации.

Лицензировать можно все ядра основного сервера виртуализации для SQL Server Enterprise Edition и приобрести покрытие SA, что позволяет не ограничивать количество виртуальных машин.

Программа Software Assurance для корпоративного лицензирования помогает повысить продуктивность ИТ, предоставляя клиентам возможность извлечь максимум пользы из программных продуктов Майкрософт.

SA позволяет устанавливать и запускать пассивные экземпляры SQL Server 2017 в отдельной среде ОС или на отдельном сервере, чтобы обеспечить высокий уровень доступности в случае события отказа.

Службы SQL Server работают в контексте безопасности назначенной учетной записи Windows. Учетная запись Windows, указанная для служб, может быть локальной учетной записью пользователя, учетной записью пользователя домена или локальной системной учетной записью. В зависимость от потребностей доступа рекомендуется применять или учетную запись пользователя домена, или локальную системную учетную запись. Можно назначать одну и ту же учетную запись Windows для всех служб SQL Server или настраивать учетную запись каждой службы индивидуально.

Службы

Службы SQL Server 2017, настраиваемые во время установки, описаны в следующей таблице.

Служба	**Описание**
SQL Server	Ядро СУБД SQL Server
Агент SQL Server	Выполняет задания, контролирует SQL Server и позволяет автоматизировать административные задачи.
Браузер SQL Server	Браузер SQL Server — это служба разрешения имен, которая предоставляет клиентским компьютерам данные подключения SQL Server. К этой службе предоставляется общий доступ для нескольких экземпляров служб SQL Server и Integration Services.

Учетные записи служб SQL Server

Каждая служба работает в контексте безопасности, определяемом учетной записью Windows, под которой она запущена. Учетная запись, используемая службой, называется учетной записью службы.

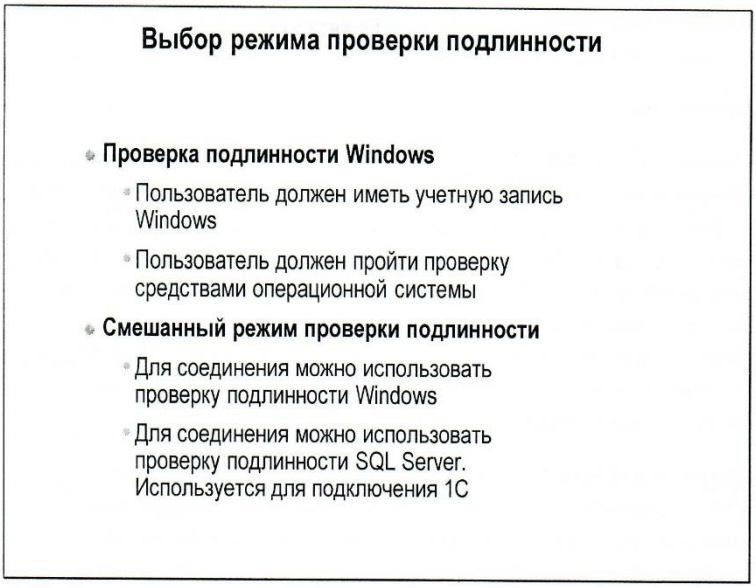
Windows 8/10 и Windows Server 2012 и выше содержат два новых типа учетных записей служб, называемых управляемыми учетными записями служб (MSA) и виртуальными учетными записями. Управляемые учетные записи служб и виртуальные учетные записи разработаны для обеспечения важных приложений, таких как SQL Server, изоляцией собственных учетных записей, устраняя необходимость в ручном администрировании имени участника-службы (SPN) и учетных данных для этих учетных записей. Это намного упрощает долгосрочное управление пользователями учетных записей служб, паролями и именами SPN.

- **Управляемые учетные записи служб**

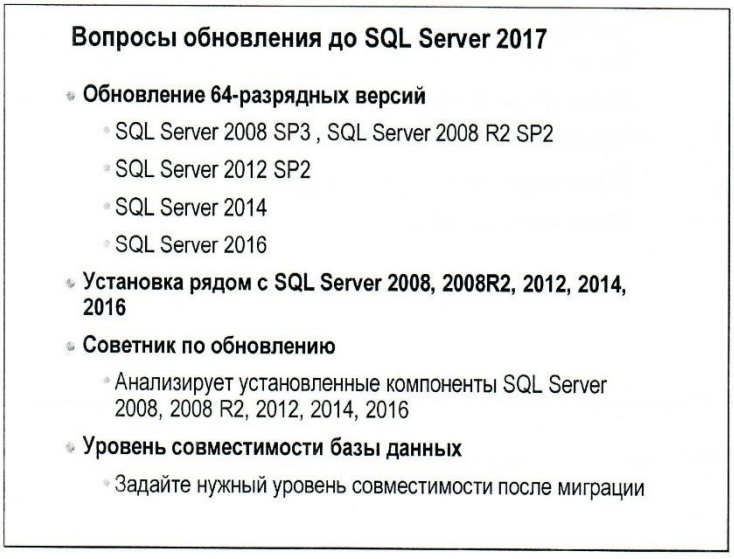
Управляемая учетная запись службы (MSA) — это тип учетной записи домена, создаваемый и управляемый контроллером домена. Она назначается отдельному компьютеру-участнику для использования при запуске службы. Управление паролем осуществляет автоматически контроллер домена. MSA нельзя использовать для входа на компьютер, но компьютер может использовать MSA для запуска службы Windows. MSA имеет возможность регистрировать имя участника-службы (SPN) с помощью Active Directory. MSA присваивается имя с суффиксом **$**, например **DOMAIN\ACCOUNTNAME$**. При указании MSA следует оставить поле пароля пустым. Поскольку учетная запись MSA назначается одному компьютеру, ее нельзя использовать на разных узлах кластера Windows. Учетная запись MSA должна быть создана в Active Directory администратором домена до того, как ее сможет использовать программа установки SQL Server для служб SQL Server.

- **Виртуальные учетные записи**

Виртуальные учетные записи в Windows Server 2012, 2012 R2, 2016 и Windows 8, 8.1, 10 представляют собой *управляемые локальные учетные записи*, которые предоставляют следующие возможности для упрощения администрирования служб. Управление виртуальной учетной записью осуществляется автоматически, и она может получать доступ к сети в среде домена. Если во время установки SQL Server используется значение по умолчанию для учетных записей служб, создается виртуальная учетная запись с именем, соответствующим имени экземпляра, в формате **NT SERVICE**<*SERVICENAME*>. Службы, запускаемые от имени виртуальных учетных записей, получают доступ к сетевым ресурсам с использованием учетных данных учетной записи компьютера в формате <*domain_name*>****<*computer_name*>**$**. При указании виртуальной учетной записи для запуска SQL Server оставьте поле пароля пустым.



При установке SQL Server назначаются параметры и выбираются правила, определяющие порядок сортировки. Термин *параметры сортировки* относится к набору правил, которые определяют, как сравниваются и разбираются данные. Символьные данные сортируются с использованием правил, которые определяют должную последовательность символов.



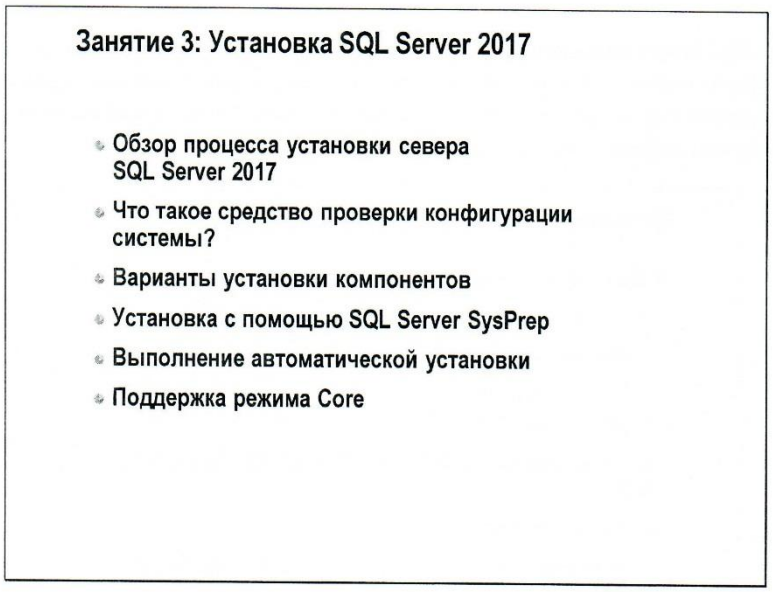
Можно непосредственно обновить экземпляры SQL Server 2008, SQL Server 2008 R2, SQL Server 2012, SQL Server 2014 и SQL Server 2016 до SQL Server 2017, если позволяет разрядность. Другой путь внедрения основан на миграции баз данных в новую установку SQL Server 2017 с помощью восстановления

резервных копий или отсоединения и присоединения баз данных. После миграции возврат к старой версии SQL Server возможен через механизм выгрузки системы «1С:Предприятие 8».

Советник по обновлению

Советник по обновлению Microsoft SQL Server 2017, который можно вызвать из Центра установки SQL Server 2017, - это программное средство, которое можно использовать для подготовки к обновлению до SQL Server 2017. Советник по обновлению анализирует установленные компоненты предыдущих версий, после чего создает отчет, в котором указываются проблемы, которые следует разрешить до или после обновления до SQL Server 2017. Отчет Советника содержит ссылки на информацию, которая поможет устранить или обойти известные неполадки. Советник по обновлению следует установить с носителя установки продукта SQL Server 2017. После того как советник по обновлению установлен, его можно запустить из меню «Пуск». При использовании системы «1С:Предприятие 8» советнику по обновлению нужно дать для анализа трассировку SQL Profiler.

Примечание. При миграции базы данных на SQL Server 2017 ее параметр Уровень совместимости не повышается до 140, а сохраняется прежним. Для гарантированного использования всех возможностей новой версии этот параметр нужно установить вручную. Новая возможность SQL Server 2017 **Адаптивная обработка запросов в базах данных**, которая позволяет повысить производительность запросов, работает только при уровне совместимости 140.



На этом занятии рассматривается процедура установки SQL Server 2017.

Процесс установки сервера SQL Server 2017 состоит из двух основных этапов: обновления компонентов и установки пакета SQL Setup MSI.

Обновление компонентов

На этапе обновления компонентов программа установки SQL Server 2017 проверяет наличие следующих компонентов и при необходимости выполняет их установку: Платформа .NET Framework 4.6

Файлы поддержки для установки SQL Server

Набор пакетов MSI

На этом этапе установки выполняются следующие задачи:

1. Анализирует компьютер с помощью средства проверки конфигурации системы.

2. Определяет, какие должны быть установлены функции.

3. Определяет подходящий тип установки (экземпляр по умолчанию или именованный экземпляр).

4. Устанавливает выбранные функции.

В процессе установки сервера SQL Server используется средство проверки конфигурации системы (SCC). С его помощью выполняется множество проверок системы и проводится сравнение выявленных им настроек с теми, которые требуются для успешной установки сервера SQL Server 2017. При обнаружении каких-либо несоответствий SCC предлагает выполнить определенные действия для разрешения выявленных проблем. Проверки, выполняемые SCC, подразделяются на пять категорий: конфигурация системы, доступность системы, настройка безопасности, конфигурация версий и настройка удаленного доступа и кластеров.

Проверки конфигурации системы

Средство SCC выполняет следующие типы проверок:

• **Требования к программному обеспечению**. Средство SCC проверяет совместимость операционной системы с устанавливаемым выпуском SQL Server и примененным пакетом обновления. Также проверяется наличие необходимых программных компонентов, таких как службы Microsoft XML Core Services (MSXML) и служба Windows Management Interface (WMI).

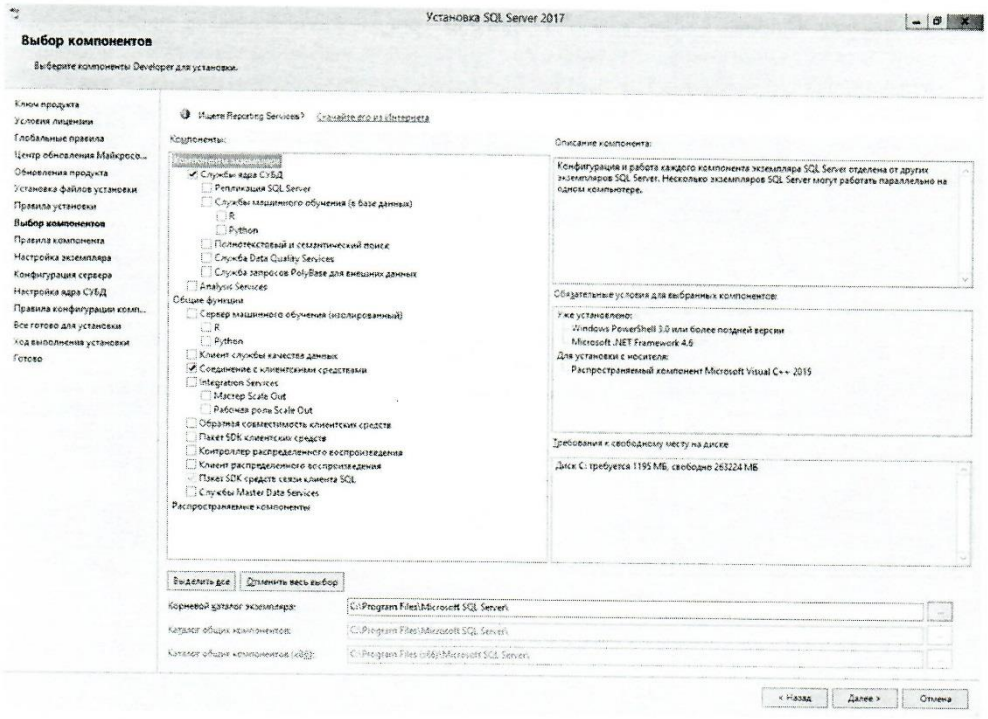
• **Требования к оборудованию**. Средство SCC проверяет, что сервер отвечает минимальным требованиям к процессору и памяти.

• **Требования безопасности**. Средство SCC проверяет, что пользователь, выполняющий установку, обладает правами, достаточными для установки сервера SQL Server, и имеет разрешения файловой системы на заданный по умолчанию каталог установки.

• **Требования к состоянию системы**. Средство SCC проверяет, что нет файлов, заблокированных для ожидающих перезагрузок, и конфигурация каталога COM+ подходит для установки SQL Server. Также проверяется, что общий ресурс Admin$ должным образом настроен для установки на кластер.

Отчет средства SCC

После окончания проверок средство SCC формирует отчет, который можно просмотреть и сохранить. В этом отчете содержатся сведения о проблемах, которые могут помешать установке, и даются рекомендации по их разрешению. В нем также содержатся предупреждения и рекомендации (например, рекомендованные исправления или настройки безопасности), относящиеся к проблемам, которые не будут препятствовать установке, но могут привести к неполадкам в работе. В большинстве случаев следует разрешить эти проблемы и повторно выполнить программу установки, а не пытаться разрешить их после того, как установка закончится.



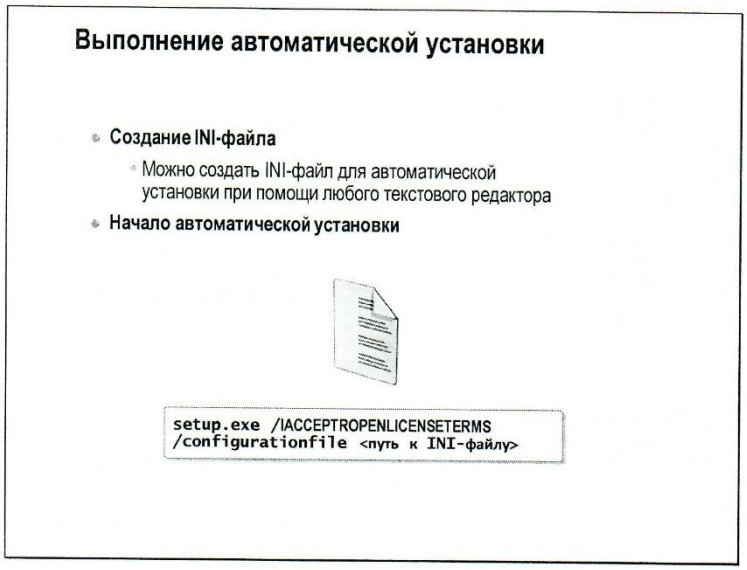
При установке SQL Server 2017 используется интерактивная программа установки. Важно понимать, какие действия следует предпринимать при выполнении программы установки, чтобы сделать выбор, соответствующий вашим конкретным потребностям. На экране «Выбор компонентов» программы установки отображается полный перечень компонентов. Ядро СУБД SQL Server будет установлено при выборе **Службы компонента Database Engine**. SQL Server Management Studio 2017 устанавливается отдельно как из центра установки, так и из дистрибутива в интернете. На рисунке выше отмечены компоненты, достаточные для поддержки системы «1С:Предприятие 8». Для SQL Server 2017, помимо встроенной установки изолированного экземпляра SQL Server за одно действие, доступна также установка за два действия. Эти действия включают следующие шаги.

- Подготовка образа. На этом шаге выполняется установка двоичных файлов продукта без настройки компьютера, сети или сведений, относящихся к определенной учетной записи, для подготавливаемого экземпляра SQL Server.
- Завершение создания образа. На этом шаге завершается настройка подготовленного экземпляра SQL Server. При его выполнении можно задать информацию, связанную только с определенным компьютером, сетью или учетной записью.

Функцию SQL Server SysPrep можно использовать одним из следующих способов.

- На шаге подготовки образа можно подготовить один или несколько ненастроенных экземпляров SQL Server на одном компьютере. Эти подготовленные экземпляры можно настроить на шаге завершения создания образа на том же компьютере. Можно захватить файл конфигурации программы установки подготавливаемого экземпляра SQL Server и использовать его для последующей настройки при подготовке других ненастроенных экземпляров SQL Server на нескольких компьютерах.

- Совместно с Windows SysPrep можно создать образ операционной системы, включая ненастроенный подготовленный экземпляр SQL Server, на первоначальном компьютере. Затем можно выполнить развертывание этого образа операционной системы на нескольких компьютерах и применить «Завершение создания образа» в программе установки SQL Server.



Можно выполнить автоматическую установку SQL Server 2017, создав INI-файл, содержащий необходимую для программы установки информацию, и запустив из командной строки программу setup.exe. Знание того, как выполняется автоматическая установка, может помочь при развертывании нескольких одинаковых установок SQL Server в рамках организации или делегировании обязанностей по установке другим техническим специалистам.

Создание INI-файла

Для создания INI-файла для автоматической установки можно воспользоваться любым текстовым редактором, например Блокнотом. Этот INI-файл состоит из одного раздела **[Options]**, содержащего несколько параметров, каждый из которых относится к определенным функциям или параметрам настройки.

Запуск автоматической установки

Для запуска автоматической установки используйте следующий синтаксис командной строки.

setup.exe /configurationfile <путь к .ini файлу>

Например, чтобы выполнить автоматическую установку с помощью INI-файла installsettings.ini, находящегося в папке C:\setup, используется следующая команда.

setup.exe /configurationfile c:\setup\installsettings.ini

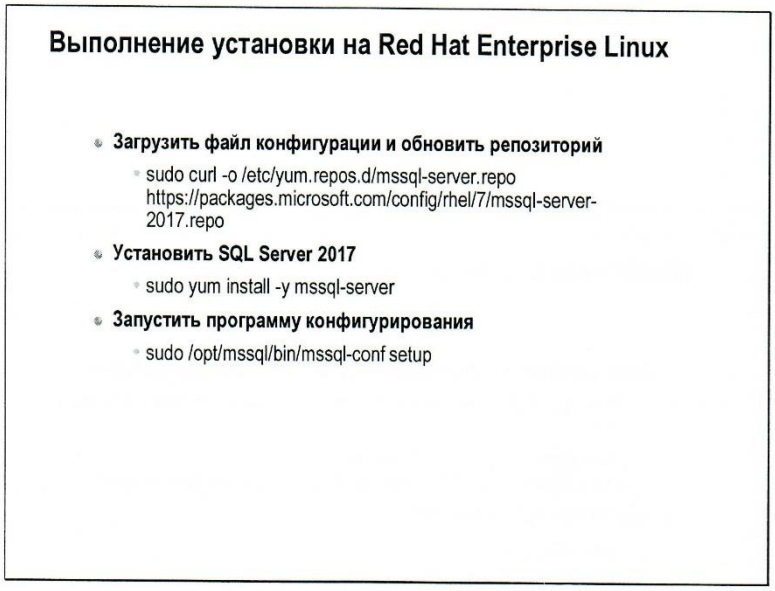
Кроме того, можно указать QUIET="True" для выполнения автоматической установки без отображения диалоговых окон.

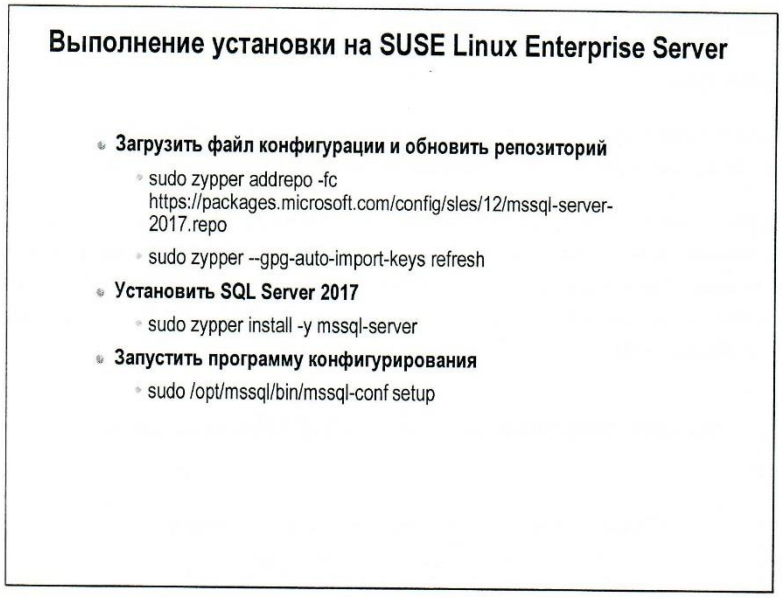
Параметр IACCEPTROPENLICENSETERMS="True" может находиться внутри файла.

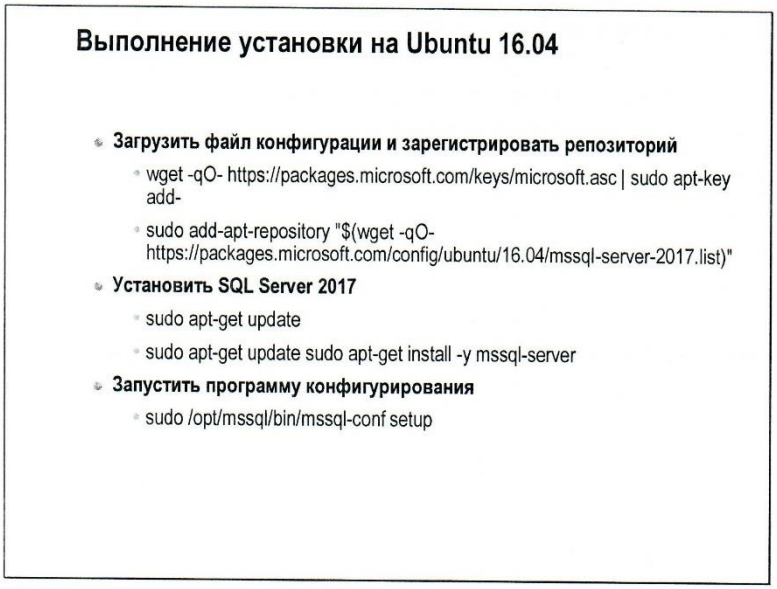
Поддержка Server Core:

Установка SQL Server 2017 поддерживается в режиме Server Core не ниже следующих выпусков Windows Server 2012 R2: Windows Server 2012 R2 с пакетом обновления 1 (SP1), Datacenter

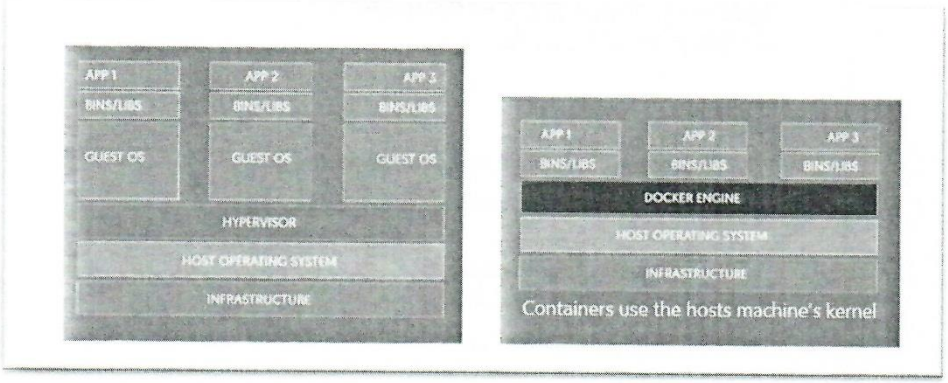
Установка SQL Server 2017 в различных реализациях Linux содержит схожие шаги и команды. На первом шаге загружается файл конфигурации и регистрируется репозиторий. Вторым шагом выполняется запуск программы установки. Третий шаг необходим для конфигурирования установленного экземпляра. Ниже приведены необходимые команды в Red Hat Enterprise Linux (RHEL) 7.3, SUSE Linux Enterprise Server (SLES) V12 SP2 и Ubuntu 16.04.







Установка SQL Server 2017 в качестве контейнера возможна при поддержке подсистемы docker 1.8. Платформой поддержки контейнеров с SQL Server 2017 могут быть следующие операционные системы: Windows Server 2016, Windows 10, Red Hat Enterprise Linux 7.2, SUSE Enterprise Linux v12 SP2, Ubuntu 16.04. Контейнеры не требуют создания и настройки виртуальных машин. Они используют ядро основной машины. Ниже на рисунке схематично для сравнения представлены виртуальные машины и контейнеры.



Контейнеры создаются под определенный тип среды, которая их поддреживает, то есть зависят от движка docker. Контейнеры содержат готовый к работе экземпляр SQL Server 2017. Таблица ниже содержит сведения об известных вариантах docker 1.8

Тип docker	Где устанавливается	Что использует	Какие контейнеры поддерживает
Docker Engine	Linux	Linux OS	Linux контейнеры
Docker for Windows containers	Windows	Windows	Windows контейнеры
Docker for Mac and Windows	Linux VM on Mac or Windows	Linux VM OS	Linux контейнеры

Контейнер сначала устанавливается, а затем запускается. Ниже приведен один из вариантов команд.

> docker pull microsoft/mssql-server-windows-express

> docker images

> docker run –e ACCEPT_EULA=Y –e SA_PASSWORD=SQLpwd2017 –p 1433:1433 –d microsoft/mssql-server-windows-express

Занятие 4: Управление установкой и конфигурирование SQL Server

- Управление с помощью Windows PowerShell
- Что представляет собой диспетчер конфигурации SQL Server
- Что представляет собой среда SQL Server Management Studio
- Программа управления установками
- Что представляет собой sqlcmd
- Что представляют собой параметры уровня сервера

В этом занятии описывается, как управлять установкой SQL Server, даются начальные сведения об административных средствах.

Управление SQL Server 2017 с помощью Windows PowerShell 4.0 и 5.0

PowerShell:

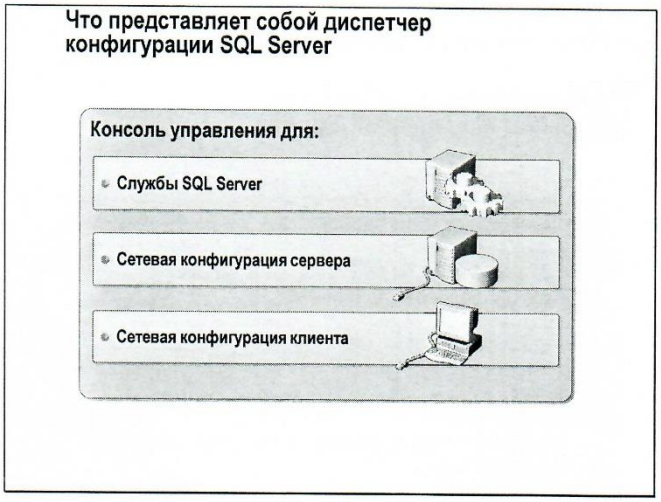
- Структурированные команды и сценарии
- Доступ к структуре и параметрам конфигурации
- Просмотр баз и таблиц как файловой системы

Команды	Описание
Stop-DbaProcess –Sqlserver "MSSQLSERVER" -Programs "1CV83 Server"	• Удаление всех клиентских соединений приложения "1CV83 Server"
Backup-SqlDatabase -ServerInstance "MSSQLSERVER" -Database "DB1C"	• Резервное копирование базы данных
Invoke-Sqlcmd	• Выполнение сценариев Transact-SQL

Язык PowerShell поддерживает более сложную логику, чем Transact-SQL, что дает возможность администраторам создавать мощные сценарии управления. Команда Invoke-Sqlcmd обеспечивает запуск сценариев на Transact-SQL. Для выхода в среду PowerShell нужно запустить утилиту SQLPS, которая начинает сессию с поставщиком SQL Server PowerShell, загружает и регистрирует команды. Некоторые команды приведены на рисунке. Install-Module SqlServer

Иерархия SQL Server PowerShell представлена диском и путями, аналогичными путям файловой системы. Корневой узел иерархии SQL Server в PowerShell представляет собой диск SQLSERVER:. Диск SQLSERVER: имеет вложенные папки. К объектам внутри папок и подпапок можно применять методы SQL Server SMO. Перемещение по иерархии возможно с помощью команды CD. PowerShell позволяет пользователям определять виртуальные диски, PSDrives, используемые для сокращения путей.

Оболочка Windows PowerShell 4.0 не устанавливается программой установки SQL Server 2017, но является обязательной для установки SQL Server 2017. Из среды SQL Server Management Studio 2017 можно использовать отдельно устанавливаемый в PowerShell модуль SQLServer.



Диспетчер конфигурации SQL Server — это средство, которое можно применять для управления службами, связанными с SQL Server, настраивать сетевые протоколы, используемые сервером SQL Server, и управлять конфигурацией сетевых подключений с клиентских компьютеров.

Службы SQL Server

Диспетчер конфигурации SQL Server можно использовать для запуска, остановки, приостановки и восстановления работы служб Windows, связанных с SQL Server. Кроме того, можно настроить эти службы для управления режимами запуска и учетными записями служб, а также задать дополнительные свойства этих служб, например параметры запуска.

Примечание. Вносить изменения в учетные записи служб следует с помощью диспетчера конфигурации SQL Server, а не консоли управления службами Windows, поскольку диспетчер конфигурации SQL Server автоматически применяет необходимые разрешения на доступ к реестру для указанных учетных записей.

Параметры запуска компонента Компонент Database Engine настраиваются с помощью новой вкладки **Параметры запуска в** свойствах службы.

Сетевая конфигурация сервера

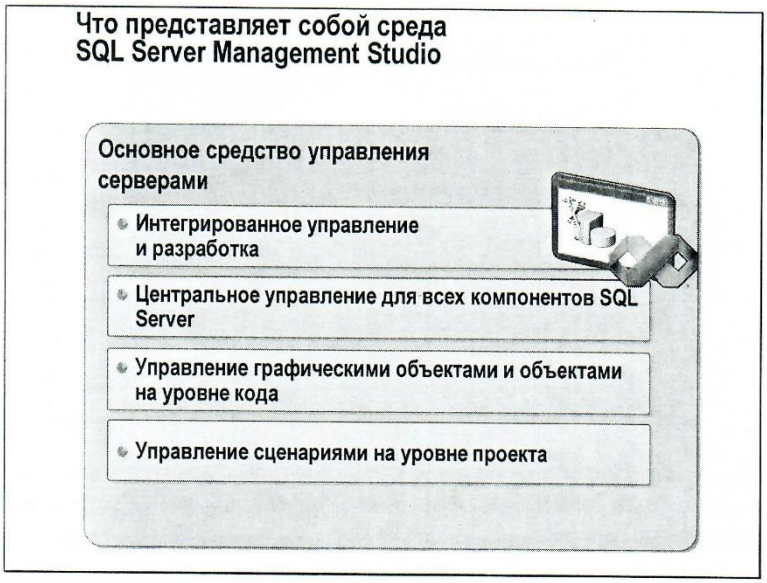
Диспетчер конфигурации SQL Server можно применять для настройки сетевых протоколов, используемых экземпляром SQL Server. Можно включать и отключать отдельные протоколы и управлять

специфичными для протоколов параметрами, такими как номер TCP-порта, используемого протоколом TCP/IP.

Сетевая конфигурация клиента

Когда диспетчер конфигурации SQL Server установлен на клиентском компьютере, его можно использовать для управления библиотекой собственного клиента SQL с помощью указания приоритетов сетевых протоколов и создания псевдонимов серверов.

Примечание. Для поддержки системы «1С:Предприятие 8» на клиенте используется SNAC11.dll. Если рабочий сервер «1С:Предприятие 8» и SQL Server установлены на одном компьютере, может используется протокол Shared Memory собственного клиента.



Большая часть административных задач для SQL Server 2017 выполняется в среде SQL Server Management Studio, которая устанавливается отдельно. При установке SQL Server Management Studio также устанавливаются служебная программа sqlcmd, SQL Server Profiler и Помощник по настройке ядра.

Функции среды SQL Server Management Studio

Среда SQL Server Management Studio предоставляет администраторам следующие возможности:

• Интегрированное средство управления и разработки, основанное на среде разработки Microsoft Visual Studio.

• Полное управление реляционными базами данных, базами данных Analysis Services, службами Reporting Services, службами SQL Server Integration Services (SSIS) и базами данных Compact SQL Server.

• Обозреватель объектов — графическая область в SQL Server Management Studio, которая может быть использована для настройки сервера, а также для управления и разработки баз данных.

• Редакторы запросов для управления и разработки на базе сценариев. Предусмотрены редакторы для Transact-SQL-, MDX-, DMX- и XMLA-запросов.

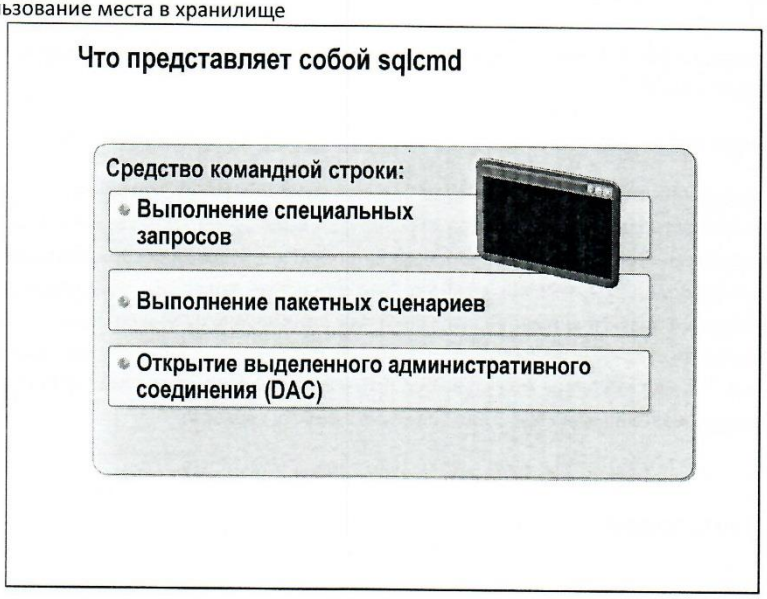
• Управление сценариями на основе проектов, при котором сценарии создания и управления базами данных хранятся как один проект и управление осуществляется в области обозревателя решений в среде SQL Server Management Studio.

• Среда SQL Server Management Studio использует платформу Visual Studio Framework и включает функции Visual Studio для создания запросов и сценариев, поддержки работы с исходным кодом при хранении и обслуживании копий сценариев, в которые со временем вносятся изменения, и для предоставления доступа к интерактивной справочной системе.

Примечание. Среда SQL Server Management Studio не требует активного подключения к базе данных при написании сценариев и запросов.

Служебная программа SQL Server. Это средство используется в SQL Server для управления несколькими экземплярами. С помощью точек обзора программы SQL Server и обозревателя программ в среде Среда SQL Server Management Studio(SSMS) администраторы получают целостное представление об исправности ресурсов SQL Server. Для этого используется экземпляр SQL Server, служащий точкой управления служебной программой (UCP). В точке управления служебной программой SQL Server отображаются объемы использования следующих ресурсов отдельными экземплярами.

- Загрузка ЦП
- Использование места в хранилище



Применение графических средств, таких как среда SQL Server Management Studio, для выполнения инструкций на языке Transact-SQL не всегда возможно или желательно. Служебная программа sqlcmd позволяет выполнить инструкции и сценарии на языке Transact-SQL из командной строки и планировать пакетные задания. Для запуска пакетов Transact-SQL эта служебная программа использует ODBC. Для выполнения нерегламентированных запросов и команд можно пользоваться служебной программой sqlcmd интерактивно. Запуск sqlcmd без указания сервера или учетных данных для проверки

подлинности приводит к подключению к локальному экземпляру, заданному по умолчанию, с применением проверки подлинности Windows. Можно подключиться к удаленному серверу или именованному экземпляру с помощью параметра **-S,** как показано в следующем примере, где выполняется подключение к именованному экземпляру SQLINSTANCE1 на сервере DBSERVER1.

sqlcmd –S DBSERVER1\SQLINSTANCE1

В служебной программе sqlcmd предусмотрено большое количество параметров командной строки. Для получения полного списка введите **sqlcmd -?** в командной строке. Чтобы выполнить запрос в служебной программе sqlcmd, введите этот запрос на языке Transact-SQL, а затем на новой строке введите команду **GO** и нажмите клавишу ВВОД. Результаты запроса отображаются в окне консоли sqlcmd. Например, чтобы извлечь данные из таблицы dbo.V8users базы данных DB1C, можно в программе sqlcmd выполнить приведенные ниже инструкции. (Обратите внимание, что программа sqlcmd автоматически добавляет номера строк.)

1> USE DB1C

2> SELECT name FROM dbo.V8Users

3> GO

Сценарии

Так же, как при интерактивном выполнении служебной программы sqlcmd, можно сохранить команды и инструкции языка Transact-SQL в файле сценария и вызвать программу sqlcmd для выполнения этого сценария. Можно создать сценарии с применением переменных и ввести значения переменных в командной строке sqlcmd.

Выделенное административное соединение

Выделенное административное соединение (DAC) — это функция, предусмотренная в SQL Server, которая предоставляет возможность доступа к серверу, даже когда он зависает или становится недоступным по какой-либо другой причине. У функции DAC предусмотрен собственный планировщик SQL Server. Поэтому подключение невозможно только в то время, когда служба SQL Server остановлена или приостановлена. Обратите внимание, что SQL Server поддерживает только один экземпляр DAC; попытка использовать второй экземпляр DAC в то время, когда первый еще активен, приведет к отказу. После получения доступа к серверу можно выполнить команды для диагностики неполадок, закрыть неработающие подключения или корректно завершить работу сервера.

SQLCMD –A

1> SHUTDOWN WITH NOWAIT

2> GO

Подключение средствами DAC

По умолчанию соединение разрешено только из клиента, запущенного на сервере. Сетевые соединения не разрешаются, если они не настроены с помощью хранимой процедуры **sp_configure** с параметром remote admin connections Option. Только члены роли SQL Server **sysadmin** могут подключаться с использованием соединения DAC. Соединение DAC доступно и поддерживается через программу

командной строки **sqlcmd** со специальным ключом (**-A**). Можно также подключиться, подставляя префикс **ADMIN:** к имени экземпляра в формате **sqlcmd –S ADMIN:***<instance_name>*. Подключение DAC можно также запустить через редактор запросов среды SQL Server Management Studio, подключившись к **ADMIN:***<instance_name>*.

Чтобы гарантировать, что для соединения есть доступные ресурсы, на один экземпляр SQL Server разрешено только одно соединение DAC. Если соединение DAC уже активно, любой новый запрос на соединение через DAC отклоняется с ошибкой 17810.

Для экономии ресурсов SQL Server Express Edition не прослушивает порт DAC без запуска с флагом трассировки 7806.

Рекомендуется подключаться к базе данных master через соединение DAC, так как база данных master будет в любом случае доступна, если запущен экземпляр компонента Database Engine. SQL Server запрещает выполнение параллельных запросов или команд через соединение DAC. Через соединение DAC гарантированно доступны только ограниченные ресурсы. DAC используется для запроса динамических административных представлений (DMV) для базовой диагностики, таких как sys.dm_tran_locks для статуса блокировки, sys.dm_os_memory_cache_counters для проверки состояния кэша, а sys.dm_exec_requests и sys.dm_exec_sessions — для активных сессий и запросов. Избегайте динамических административных представлений DMV, потребляющих много ресурсов (например, представление sys.dm_tran_version_store просматривает хранилище полных версий, что может привести к резкому увеличению объема входящих/выходящих данных) или использующих сложные соединения.

SQL Server слушает DAC на выделенном порту TCP, динамически назначенном при запуске Database Engine. Журнал ошибок SQL Server приводит номер порта для подключения DAC; по умолчанию он равен 1434. Можно разрешить средству прослушивания соединений DAC прием удаленных соединений, даже если SQL Server не отвечает. Это можно сделать, сначала подключившись к SQL Server с локальным использованием соединения DAC, а затем выполнив хранимую процедуру **sp_configure** для разрешения приема удаленных соединений.

Что представляют собой параметры уровня сервера

Параметр	Описание
Показать дополнительные параметры **show advanced options**	Определяет, отображает ли **sp_configure** дополнительные параметры настройки или нет
Расширение буферного пула	ALTER SERVER CONFIGURATION SET BUFFER POOL EXTENSION ON (FILENAME = 'E:\SSDCACHE\MYCACHE.BPE', SIZE = 50 GB);
Мин. память сервера **и макс. память сервера**	Управляет объемом памяти, используемым экземпляром SQL Server
cost threshold for parallelism	Указывает порог стоимости запроса для параллельного выполнения
Порог заблокированных процессов	Определяет интервал в секундах для выдачи отчетов о заблокированных процессах

Параметры уровня сервера управляют поведением экземпляра SQL Server.

sp_configure

Параметры уровня сервера могут настраиваться с помощью хранимой процедуры sp_configure. Многие параметры можно также устанавливать с помощью SQL Server Management Studio и средства настройки контактной зоны SQL Server. Когда используется хранимая процедура sp_configure, после установки параметра настройки необходимо выполнить инструкцию RECONFIGURE или RECONFIGURE WITH OVERRIDE. Инструкция RECONFIGURE WITH OVERRIDE обычно зарезервирована для параметров настройки, которые должны применяться с предельной осторожностью. Однако инструкция RECONFIGURE WITH OVERRIDE работает для всех параметров настройки, и ее можно использовать вместо инструкции RECONFIGURE. Текущее значение для каждого параметра можно определить с помощью следующей инструкции.

```
SELECT * FROM sys.configurations ORDER BY name;
GO
```

Некоторые глобальные параметры конфигурации текущего сервера в SQL Server 2017 (привязку к процессорам, ведение диагностического журнала, параметры кластера отработки отказа) можно также изменять с помощью команды

```
ALTER SERVER CONFIGURATION
```

В SQL Server 2017 существует настройка файла расширения буферного пула. Расширение буферного даст повышение производительности, если справедливы определенные условия. SQL Server 2017 поддерживает высокопроизводительные устройства хранения, такие как SSD диски, для расширения буферного пула. Добавление дисковой памяти часто проще чем добавление оперативной. Когда расширение буферного пула настроено, SQL Server использует диск для страниц данных буферного пула. Только чистые страницы, содержащие зафиксированные данные, хранятся в расширении буферного

пула, гарантируя, что не будет потери данных в случае отказа диска. Дополнительно, если отказывает диск, расширение буферного пула автоматически отключается. После замены диска можно снова включить расширение буферного пула.

Расширение буферного пула дает следующие преимущества

- Повышение производительности для OLTP приложений с большим числом операций чтения
- Поскольку SSD диски дешевле оперативной памяти расширение буферного пула представляет собой недорогое решение повышения производительности баз с интенсивным вводом-выводом
- Расширение буферного пула легко настраивается и не требует внесения изменений в приложения
- В нагрузке ввода-вывода OLTP приложений наблюдается большое число операций чтения
- Сервер содержит до 32Гб физической памяти
- Расширение буферного пула настроено на использование файла, размер которого в 4-10 превышает объем физической памяти

Расширение буферного пула не даст существенного повышения производительности, если справедливы следующие условия

- Имеется нагрузка характерная для хранилищ данных
- В OLTP нагрузке наблюдается большое число операций записи
- Более 64Гб физической оперативной памяти доступно SQL Server.

Для изменения размера файла под расширение буферного пула нужно сначала отключить расширение, а затем включить с новыми параметрами. При отключении расширения буферного пула, в SQL Server уменьшится размер буферного пула, что приведет к немедленной нагрузке на память и ввод-вывод и падению производительности. Изменение конфигурации буферного пула нужно внимательно планировать для уменьшения влияния на работу пользователей.

Для просмотра статуса расширения буферного пула используется представление динамического управления sys.dm_os_buffer_pool_extension_configuration. Для мониторинга использования расширения буферного пула используется представление динамического управления sys.dm_os_buffer_descriptors.

Для включения и отключения расширения буферного пула используется команда

ALTER SERVER CONFIGURATION

Пример настройки расширения буферного пула на 50Гб

```
ALTER SERVER CONFIGURATION
SET BUFFER POOL EXTENSION ON
(FILENAME = 'E:\SSDCACHE\MYCACHE.BPE', SIZE =
```

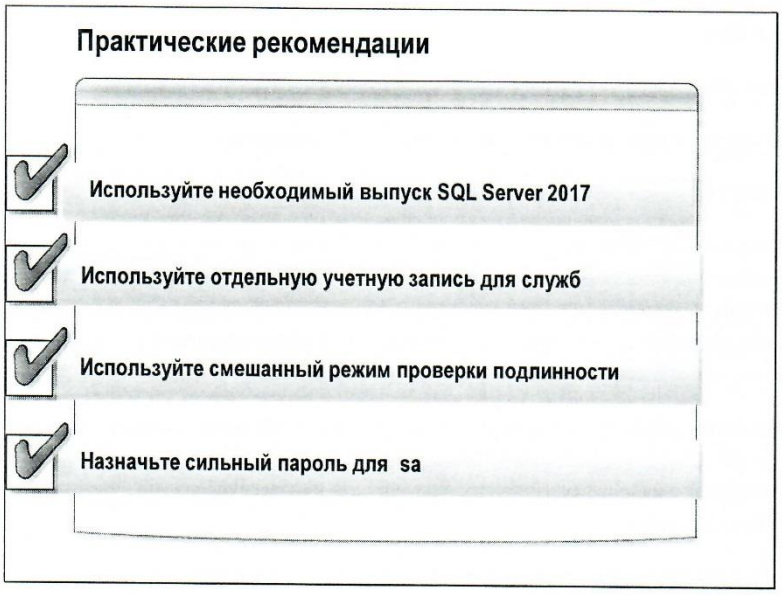
50 GB);

Пример отключения расширения буферного пула

```
ALTER SERVER CONFIGURATION
SET BUFFER POOL EXTENSION OFF
```

Расширение буферного пула будет использоваться только в случае нехватки оперативной памяти. В этом случае SQL Server освобождает страницы буферного пула в соответствии с алгоритмом LRU (Least Recently Used). Страницы сбрасываются в файл расширения буферного пула, а не в файл базы данных. Если на странице были сделаны изменения, то она параллельно записывается и в файл данных. Если файл расширения буферного пула заполнился, SQL Server освобождает место в файле расширения в соответствии с алгоритмом LRU и записывает эти страницы в файл базы данных. Файл расширения буферного пула представляет собой дополнительный слой между буферным пулом и хранилищем базы данных.

Важно! Если после изменения параметров не удается запустить SQL Server, запустите службу сервера с помощью параметра запуска «–f» с конфигурацией по умолчанию. При этом SQL Server переходит в однопользовательский режим.



Задание 1. Установка SQL Server для использования системой «1С:Предприятие 8».

1. Установите на рабочее место экземпляр по умолчанию пробного выпуска SQL Server 2017 Enterprise Edition.
    - Перейдите в папку с дистрибутивом Server 2017 Enterprise Edition
    - Запустите программу установки SQL Server 2017 **Setup.exe**
    - Выберите параметры установки, необходимые и достаточные для поддержки сервера предприятия системы «1С:Предприятие 8»

- Для служб СУБД и агента используйте доменную учетную запись, **с которой вы вошли в систему**.
- Установите для всех служб автоматический режим запуска
- Выберите смешанный режим проверки подлинности и установите пароль для sa
- Во время установки добавьте запись текущего пользователя к администраторам SQL Server
- Посмотрите возможность настроить файлы базы tempdb во время установки

2. Выполните проверку и настройку параметров SQL Server.
   - В Центре установки просмотрите отчет об установленных компонентах
3. Запустите **Диспетчер конфигурации SQL Server**
   - Проверьте настройки протокола TCP/IP
4. В папке **C:\Program Files\Microsoft SQL Server\140\Setup Bootstrap\Log** просмотрите файл Summary.txt
5. Установите на рабочее место **SQL Server Management Studio 2017**.
   - Запустите на своей машине **SQL Server Management Studio 2017**, откройте свойства локального сервера
   - Проверьте параметры сортировки.
   - Проверьте режим проверки подлинности

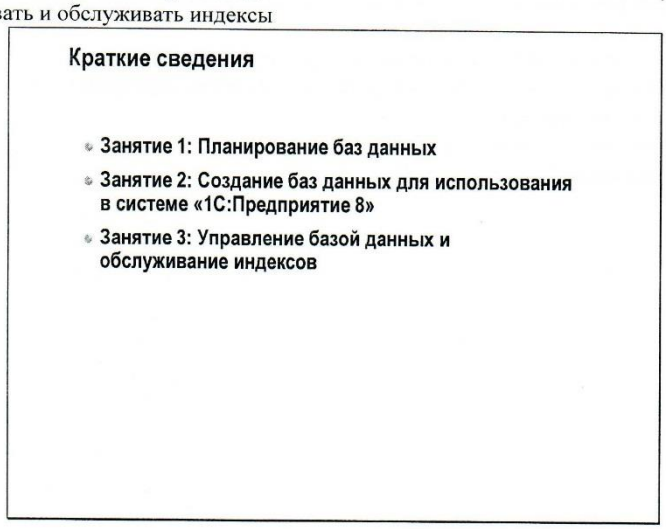
Раздел 2: Управление файлами базы данных

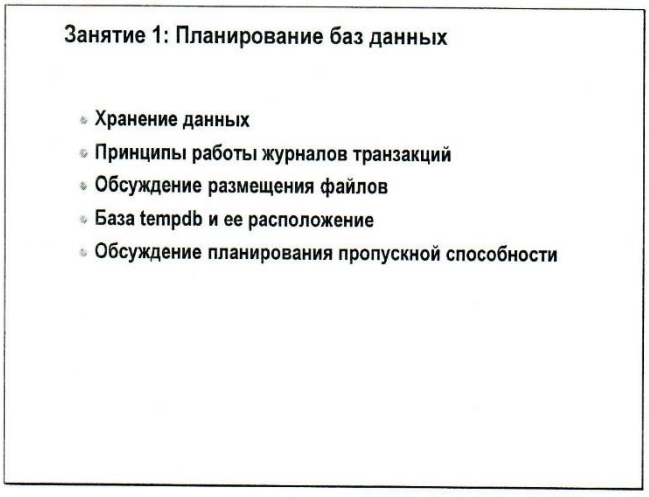
В этом разделе содержатся инструкции по планированию и созданию баз данных, извлечению сведений о базах данных, а также об использовании параметров баз данных для управления различными аспектами баз данных в разных ситуациях.

Цели

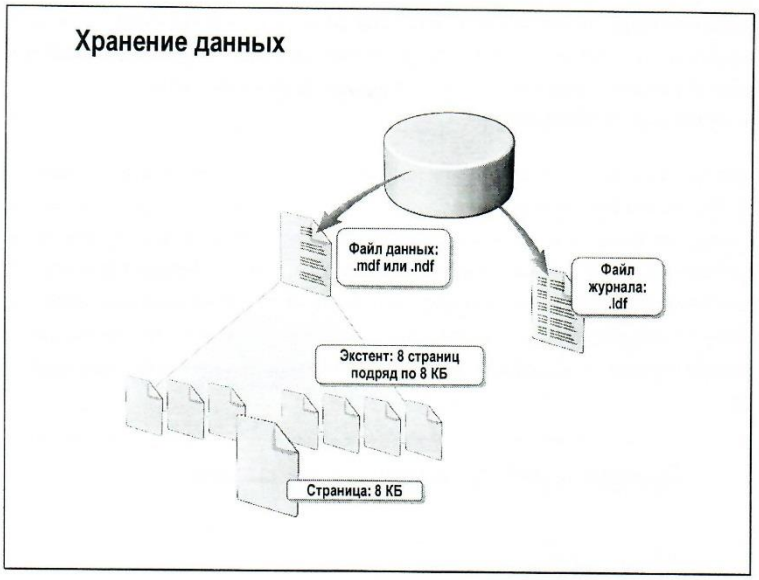
После изучения данного раздела вы сможете:

- планировать реализацию баз данных, соответствующих необходимым требованиям.
- создавать базы данных SQL Server.
- управлять базами данных SQL Server.
- анализировать и обслуживать индексы





На этом занятии вы узнаете, каким образом в SQL Server 2017 осуществляется хранение данных в базе данных, получите инструкции по использованию файлов и файловых групп для баз данных, а также узнаете, как можно оценить объем дискового пространства, необходимый для создания новой базы данных.



Для всех баз данных существует первичный файл данных (MDF-файл) и один или несколько файлов журналов транзакций (LDF-файл). В базе данных могут также существовать вторичные файлы данных (NDF-файлы). Данные хранятся в блоках, представляющих собой непрерывные участки дискового пространства объемом 8 КБ, которые называются страницами. Это значит, что база данных может содержать 128 страниц на 1 мегабайт (МБ) своего объема:

• При создании базы данных копия базы данных model, содержащая системные таблицы, копируется в базу данных, а остальная часть базы данных заполняется пустыми страницами.

• Строка не может переходить на другую страницу. Таким образом, максимальный объем данных в отдельной строке за вычетом служебной информации составляет 8060 байт. Существует два исключения из этого правила.

• Функция «Строка-переполнение» в SQL Server разрешает использование строк, содержащих столбцы, определенные как **varchar**, **nvarchar**, **varbinary**, **sql_variant** или определяемые пользователем среды CLR типы, размер которых может превысить размер страницы, если размер столбца не превышает 8000 байт.

• Столбцы, определяемые как **varchar**, **nvarchar** и столбцы **varbinary**, определенные с помощью спецификатора **max**, хранятся с использованием указателя страницы данных, содержащего ссылку на дополнительный набор страниц, где хранится фактическое значение столбца.

• Таблицы и индексы хранятся в экстентах. Экстент — это восемь страниц, не разделенных физически, общий объем которых равен 64 КБ. Таким образом, в базе данных на один мегабайт пространства приходится 16 экстентов. Для маленьких таблиц экстенты могут использоваться совместно с другими объектами базы данных. Размер 64 КБ можно учесть в настройке RAID массивов и форматировании разделов NTFS.

• Файлы журналов транзакций содержат сведения, необходимые для восстановления базы данных в случае сбоя в системе, но не содержат страницы с данными.

Для физических файлов существуют как имена файлов операционной системы, так и логические имена файлов, которые могут использоваться в инструкциях Transact-SQL. По умолчанию все файлы данных и журналы транзакций расположены по адресу C:\Program Files\Microsoft SQL Server\MSSQL14.MSSQLSERVER\MSSQL\Data.

В SQL Server действует возможность, которая называется немедленной инициализацией файлов (*instant file initialization*). Она позволяет не заполнять файлы данных нулями, что резко сокращает время, требуемое для создания файлов баз данных или их увеличения. Однако эта возможность используется при условии: учетная запись, от имени которой работает SQL Server, обладает специальным правом операционной системы **Выполнение задач обслуживания тома**. По умолчанию такое право есть у встроенной группы локальных администраторов. Эту привилегию можно задать во время установки SQL Server. Немедленная инициализация файлов при создании и приращении файлов журналов транзакций не используется.

Транзакция — это набор, состоящий из одной или нескольких инструкций Transact- SQL, которые рассматриваются как отдельная единица работы и восстановления. Инструкции Transact-SQL в транзакции должны выполняться полностью или не выполняться совсем.

Выполнение транзакций в SQL Server

SQL Server выполняет явную транзакцию, если явно определяются начало и завершение транзакции. Можно определить начало и завершение транзакции в Transact-SQL с помощью инструкций BEGIN TRANSACTION и COMMIT TRANSACTION. SQL Server может также функционировать в режиме неявных транзакций. При установке этого режима SQL Server выполняет неявную транзакцию, когда в качестве транзакции выполняется любая из следующих инструкций Transact-SQL: ALTER TABLE, CREATE, DELETE, DROP, FETCH, GRANT, INSERT, OPEN, REVOKE, SELECT, TRUNCATE, TABLE, UPDATE. По умолчанию SQL Server работает в режиме автоматического завершения транзакций. Неявная транзакция завершается после выполнения без использования инструкции COMMIT TRANSACTION. Это значит, что в этом режиме когда

инструкция Transact-SQL начинает транзакцию (BEGIN TRANSACTION), транзакция должна содержать инструкцию COMMIT TRANSACTION для завершения.

Использование журналов транзакций

SQL Server записывает каждую транзакцию в журнал транзакций для обеспечения согласованности базы данных и для использования при восстановлении базы данных. Журнал — это область хранения, которая позволяет автоматически отследить изменения базы данных. SQL Server записывает изменения в журнале на диск по мере их выполнения, но до того, как они будут записаны в базу данных.

Процесс ведения журнала транзакций

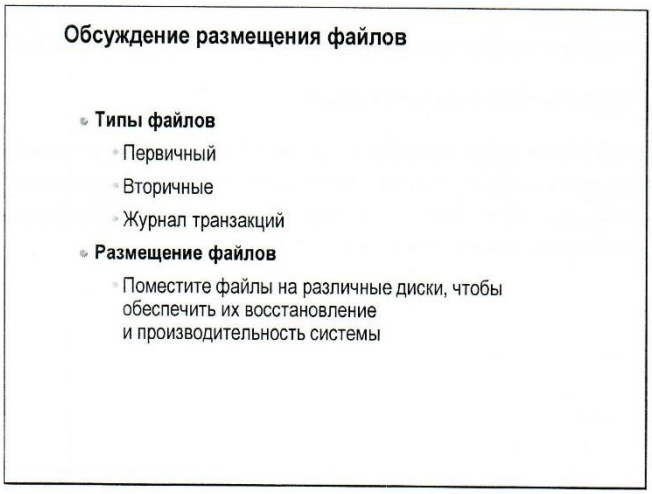
Изменения данных записываются в журнал транзакций по мере их осуществления. Процесс ведения журнала состоит из таких шагов:

1. Приложение отправляет измененные данные.

2. После выполнения изменения SQL Server загружает страницу данных, для которой осуществляется изменение, с диска в память (буферный кэш), если страница еще не находится в кэше с момента предыдущего запроса.

3. SQL Server записывает все инструкции по изменению данных в журнал по мере их выполнения. Изменение всегда записывается в журнал на диск перед тем, как выполняется в базе данных. Такой тип журнала называется журналом упреждающей записи.

4. В процессе установки контрольных точек измененные данные и страницы индекса периодически записываются в базу данных на диске.



При создании базы данных SQL Server создает файл данных и журнал транзакций для этой базы данных и позволяет определять расположение этих файлов.

Типы файлов

Каждая база данных должна содержать первичный файл и файл журнала транзакций. Она также может содержать один или несколько вторичных файлов. Первичный файл содержит данные запуска для базы данных и указывает на другие файлы в базе данных. Объекты и данные пользователя могут храниться в этом первичном файле или во вторичных файлах данных. В каждой базе данных есть один первичных файл. Рекомендуемое расширение имени файла — MDF. Для хранения данных пользователя можно создавать вторичные файлы. Рекомендуемое расширение имени файла для вторичного файла — NDF. Все базы данных должны содержать журнал транзакций. Если не указано другое, файл журнала транзакций создается автоматически с именем, сформированным системой. Рекомендуемое расширение для файла журнала транзакций — LDF. Дополнительные файлы журнала транзакций не повышают производительность.

Размещение файлов

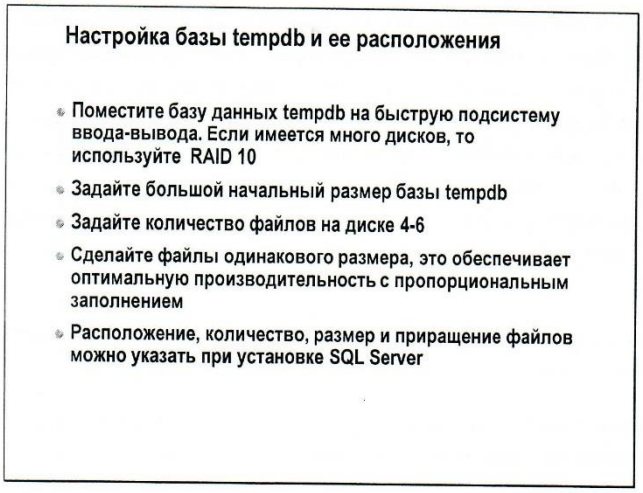
Управляя размещением файлов данных и журналов транзакций на дисках, можно повысить производительность и реализовать отказоустойчивость. SQL Server использует запросы на ввод-вывод в Microsoft Windows для осуществления считывания с диска и записи на диск. SQL Server управляет временем и способом выполнения ввода-вывода, но основные операции ввода-вывода выполняет Microsoft Windows Server.

При работе с большими базами данных следует распределять максимальный возможный объем данных по максимальному возможному количеству физических дисков. Это позволяет повысить пропускную способность путем параллельного доступа к данным, используя несколько файлов. Следует создать один файл для каждого физического диска и сгруппировать файлы в одну или несколько файловых групп.

Чтобы равномерно распределить данные по всем дискам, используйте технологии избыточных массивов независимых дисков (RAID), а затем используйте определяемые пользователями файловые группы для распределения данных по группам дорожек жесткого диска, если это необходимо.

Создание журналов транзакций на отдельных дисках

Следует создать журнал транзакций на отдельном диске вне файлов базы данных или использовать RAID. Так как файл журнала транзакций записывается последовательно, использование отдельного выделенного диска позволяет головкам диска оставаться на месте для следующей операции записи. Использование технологии RAID также обеспечивает отказоустойчивость.



Использование базы данных tempdb

База tempdb активно используется системой «1С:Предприятие 8». В процессе работы системы «1С:Предприятие 8» возможно значительное увеличение размера базы данных tempdb.

Системная база данных tempdb является глобальным ресурсом, доступным всем пользователям, которые подключены к экземпляру SQL Server. База данных tempdb служит для хранения следующих объектов: пользовательские объекты, внутренние объекты и хранилища версий. Пользовательские объекты явно создаются пользователями. Внутренние объекты создаются ядром СУБД SQL Server при необходимости для обработки инструкций SQL Server. Они создаются и удаляются в области действия инструкции. Хранилище версий — это коллекция страниц данных, содержащих строки данных, которые необходимы для поддержки возможностей, применяющих управление версиями строк. В SQL Server предусмотрено два хранилища версий: общее хранилище версий и хранилище версий оперативного построения индексов.

Модель восстановления базы данных tempdb имеет значение SIMPLE. Эта модель автоматически контролирует место под журнал, уменьшая требования к месту на диске. Автоматическое расширение файлов tempdb включено. Это разрешает файлу расти вплоть до заполнения диска. Установите шаг роста файлов на приемлемую величину (по умолчанию 64 МБ), чтобы избежать слишком маленького увеличения размера файлов базы данных tempdb. Если увеличение файлов будет идти слишком медленно по сравнению с объемом записываемых в базу tempdb данных, база данных tempdb может требовать постоянного расширения. Это повлияет на производительность.

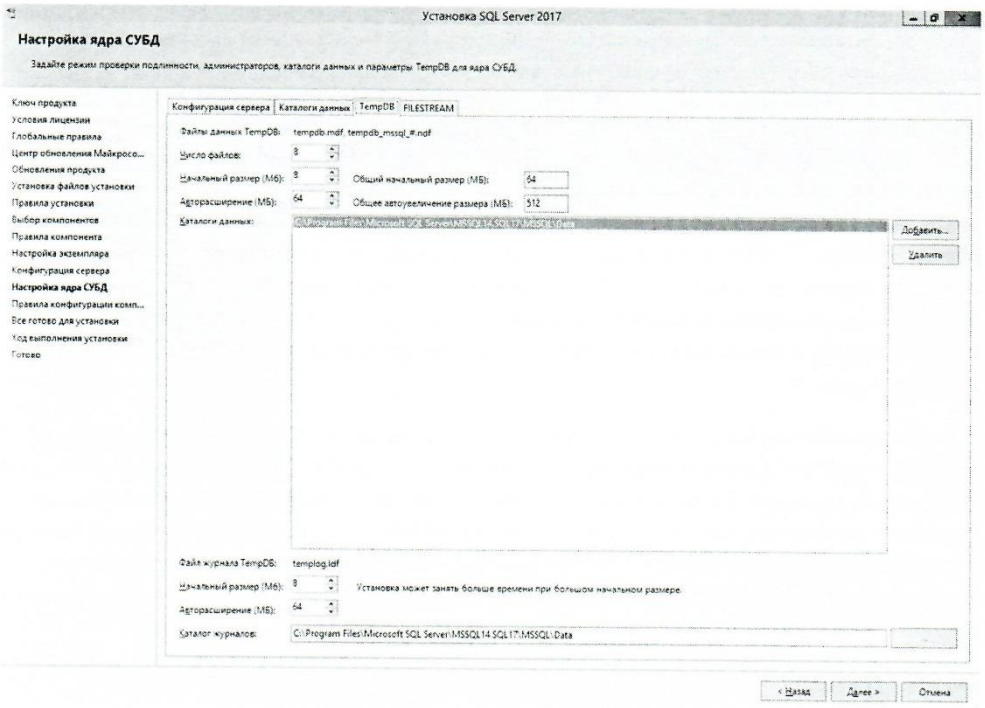
Можно установить процент, основываясь на скорости подсистемы ввода-вывода, на которой расположены файлы tempdb. Чтобы избежать таймаутов возможных кратковременных блокировок, рекомендуется ограничить продолжительность операции авторасширения приблизительно двумя минутами. Например, если подсистема ввода-вывода может заполнять файл со скоростью 50 МБ в секунду, то шаг роста FILEGROWTH следует установить в значение не более 6 ГБ, вне зависимости от размера файла tempdb. Если возможно, используйте мгновенную инициализацию файла базы данных, чтобы улучшить производительность операций авторасширения. Поведение tempdb в SQL Server 2017 по умолчанию соответствует механизму, который ранее обеспечивался трассировочным флагом T1118.

Размещение базы данных tempdb

Заранее выделите место для всех файлов tempdb, установив размер файла в значение, достаточное, чтобы гарантировать обычную рабочую нагрузку в среде. Это предотвращает слишком частое расширение tempdb, которое может повлиять на производительность. Создайте столько файлов, сколько требуется, чтобы максимально увеличить пропускную способность диска. Использование нескольких файлов сокращает конфликты хранилищ базы данных tempdb и обеспечивает гораздо лучшую масштабируемость. Общая рекомендация состоит в том, чтобы создать от 4 файлов данных. Сделайте файлы одинакового размера, это обеспечивает оптимальную производительность с пропорциональным заполнением. Количество файлов больше 8 практически не повышает производительность.

Поместите базу данных tempdb на быструю подсистему ввода-вывода. Если имеется много дисков, то используйте чередование дисков.

Расположите базу данных tempdb на дисках, отличных от используемых пользовательскими базами данных. Все настройки можно сделать при установке SQL Server 2017.



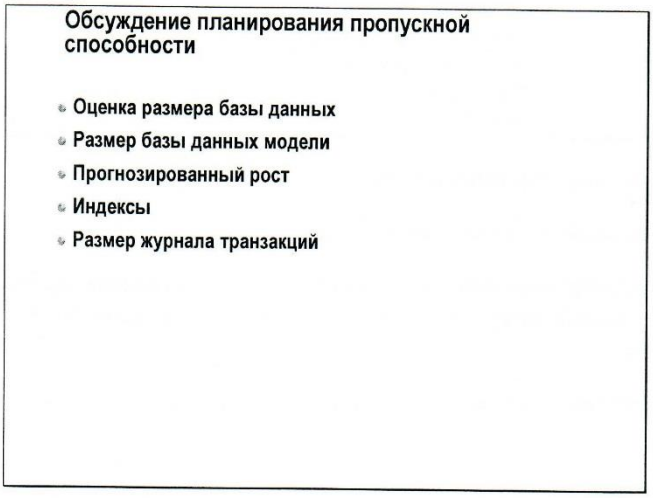
После перезапуска службы SQL Server tempdb создается заново с существующими размерами. После установки можно переместить tempdb, выполнив следующие действия:

В SQL Server Management Studio посмотреть свойства базы tempdb или отчет о занимаемом дисковом пространстве.

Изменить месторасположение файлов базы данных tempdb с помощью команды ALTER DATABASE. Для этого в SQL Server Management Studio выполнить следующую последовательность команд:

```
USE master
GO
ALTER DATABASE tempdb MODIFY FILE (NAME = tempdev, FILENAME =
'Новый_Диск:\Новый_Каталог\tempdb.mdf')
GO
ALTER DATABASE tempdb MODIFY FILE (NAME = templog, FILENAME =
'Новый_Диск:\Новый_Каталог\templog.ldf')
GO
```

После этого надо перезапустить Microsoft SQL Server.



При оценке объема, который будет занимать база данных, примите во внимание такие коэффициенты:

• Размер объектов в базе данных model и системных таблицах, учитывая предполагаемое расширение.

• Объем данных в таблицах, учитывая предполагаемое расширение.

• Количество и размер индексов, в частности размер значения ключа, количество строк и значение коэффициента заполнения. Значение коэффициента заполнения позволяет резервировать места в таблице для потенциального расширения таблицы в будущем.

• Размер журнала транзакций, который зависит от объема и частоты изменений, размер каждой транзакции и частота создания резервных копий или вывода содержимого журнала.

• Размер системных таблиц, например количество пользователей, объекты и т. д., которые обычно не составляют большую часть объема базы данных.

Примечание. Обычно объем файла журнала транзакций составляет 25% от объема данных. Для поддержки системы «1С:Предприятие 8» начальный размер журнала может составлять 50% от объема

данных. Меньшую часть объема можно распределить для баз данных, которые используются в основном для запросов.

При создании базы данных в SQL Server выполняются такие действия:

• Создается файл с данными и журнал транзакций для базы данных.

• В SQL Server необходимо, чтобы владелец и создатель новой базы данных имел разрешение на использование базы данных master, так как сведения обо всех базах данных в SQL Server записываются в базе данных master.

• SQL Server позволяет определять имя базы данных, свойства базы данных и местоположение файлов базы данных.

• Используется копия объектов в базе данных **model** для инициализации базы данных и ее метаданных. Все параметры и настройки, применяемые в базе данных **model**, копируются в новую базу данных.

• Оставшаяся часть базы данных заполняется пустыми страницами, за исключением страниц, содержащих внутренние данные об использовании пространства в базе данных.

Определение параметров при создании базы данных

При создании базы данных можно указать такие параметры:

• **Первичный файл**. Имя первичного файла по умолчанию — <имя базы данных>.mdf, он размещается в папке Data экземпляра SQL Server. Для установки по умолчанию файл размещен по адресу \Program Files\Microsoft SQL Server\MSSQL14.MSSQLSERVER\MSSQL\Data на системном диске.

• **Вторичные файлы**. Дополнительные вторичные файлы могут быть созданы и распределены в первичную файловую группу или в любую существующую файловую группу, определенную пользователем. При необходимости можно создать новые файловые группы. Рекомендуемое расширение имени файла для вторичного файла — ndf.

• **Журнал транзакций**. Имя журнала транзакций — <имя базы данных>.ldf, файл размещается в папке Data экземпляра SQL Server.

Примечание. Как правило, для достижения высокой производительности и избыточности файлы следует распределить по нескольким дискам.

• **Размер**. Можно указать размеры для каждого файла данных и файла журнала. Значение начального размера совпадает со значением, используемым в базе данных model. Размер, указанный для первичного файла данных, не должен быть меньше размера базы данных модели.

• **Увеличение размера файла**. Если необходимо, можно указать, будет ли файл увеличиваться в размере. Этот параметр часто называют автоматическое расширение. По умолчанию возможность увеличения размера файла включена. Можно указать увеличение размера файла в мегабайтах или в виде процента от начального размера файла. Принятое по умолчанию значение увеличения равно 64 МБ.

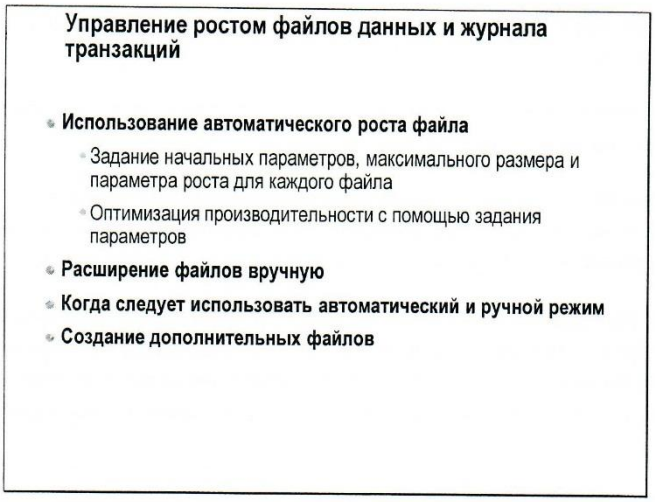
• **Максимальный размер файлов**. Можно указать максимальный размер файла в мегабайтах. Рекомендуется указать максимальный размер, до которого файл может увеличиваться. Если не указать размер, а возможность увеличения размера файла будет включена, по умолчанию файл будет увеличиваться до тех пор, пока диск не переполнится.

Параметры сортировки. Эта функция определяет способ сортировки данных, указывающий, в каком порядке перечислены значения, когда данные отсортированы соответствующим образом. Существуют различные варианты сортировки данных, порядок сортировки зависит от того, выбрана ли сортировка с учетом регистра или без учета регистра, от правил сортировки для диакритических символов и специальных символов, а также от других факторов. По умолчанию для базы данных применяется та же сортировка, что и для экземпляра SQL Server, в котором база данных была создана, но можно указать другой тип сортировки, если это необходимо.

В SQL Server необходимо дополнительное место на диске для файлов журналов транзакций. В ходе стадии отката при восстановлении после сбоя SQL Server позволяет пользователям получить доступ к базе данных. Это возможно потому, что транзакции, не зафиксированные на момент возникновения сбоя, повторно получают все блокировки, которыми они владели перед сбоем. При откате транзакций их блокировки помогают защитить эти транзакции от вмешательства пользователей. Эти дополнительные данные о блокировках должны сохраняться в журнале транзакций.

Лучше создать достаточно большой журнал транзакций, чтобы избежать частого увеличения размера. Если требуется автоматическое увеличение, лучше задать большее значение приращения размера файла.

Примечание. С целью повышения производительности создавайте базу данных на SQL Server для системы «1С:Предприятие 8» заранее. Используйте эффективное расположение на разных физических дисках. Задавайте большие начальные размеры файлов.



Увеличение файла базы данных

Можно использовать инструкцию ALTER DATABASE с помощью предложения MODIFY FILE, чтобы изменить размер файла базы данных. Обратите внимание, что вы можете использовать эту команду, чтобы увеличить файл базы данных, то есть новый размер должен быть больше текущего размера файла. Следующий пример кода демонстрирует часть синтаксиса инструкции ALTER DATABASE с предложением MODIFY FILE. Следующий пример кода показывает, как с помощью Transact-SQL увеличить размер файла базы данных.

```
ALTER DATABASE DB1C

MODIFY FILE (NAME = N'DB1CData1', SIZE = 100 MB)
```

Графический интерфейс также позволяет выполнять такие операции.

Сжатие файла базы данных

В зависимости от того, что содержится в файле базы данных в настоящий момент, у вас может быть возможность уменьшить его размер на диске с помощью команды DBCC SHRINKFILE. Следующий пример кода показывает, как сжать файл базы данных с помощью Transact-SQL.

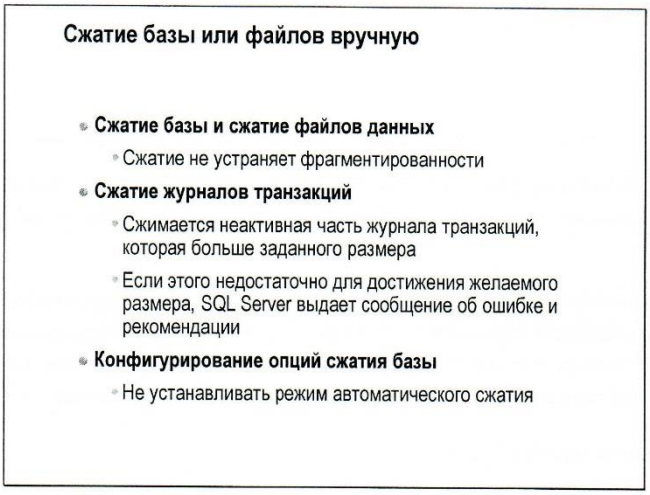
```
USE DB1C

DBCC SHRINKFILE (N'DB1CData1', 8)
```

Наибольший эффект от операции сжатия достигается при ее применении после операции, создающей много неиспользуемого пространства, например, после усечения или удаления таблицы. Большинству баз данных требуется некоторое свободное пространство для выполнения обычных ежедневных операций. Если сжатие базы данных производится регулярно, но она снова увеличивается в размерах, это означает, что место, освобожденное при сжатии, необходимо для нормальной работы. В этом случае регулярное сжатие базы данных не принесет результата. Операция сжатия не уменьшает фрагментацию индексов в базе данных и, как правило, приводит к большей фрагментации. Это еще одна причина, почему не стоит производить сжатие базы данных регулярно.

Перемещение файлов данных и журналов транзакций

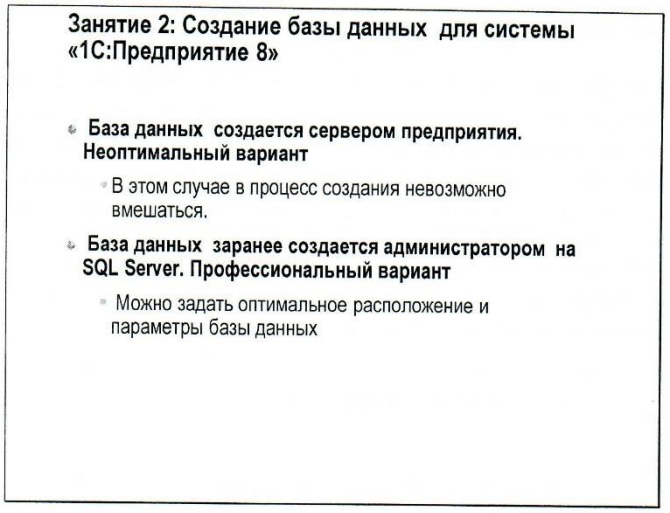
Вы можете отсоединять файлы данных и журналы транзакций от экземпляра SQL Server и снова присоединять их к тому же самому или к другому экземпляру. Отсоединение используется при перемещении БД на другой экземпляр SQL Server или на другой сервер. Оно также используется для перемещения данных и журналов транзакций на другие физические диски. Отсоединить и снова присоединить БД и связанные с ней физические файлы можно в SQL Server Management Studio.

Примечание. При перемещении или размещении файлов данных и журналов транзакций на разделе диска с файловой системой NTFS проверьте права доступа учетной записи, используемой службой SQL Server. Она должна иметь полный доступ к этим файлам.



Сжатие базы данных лучше выполнять на уровне отдельных файлов. Эту операцию обычно выполняют после удаления большого объема данных из базы. Однако сжатие файла не устраняет фрагментации индексов и таблиц, а только устраняет пустые промежутки внутри файла.

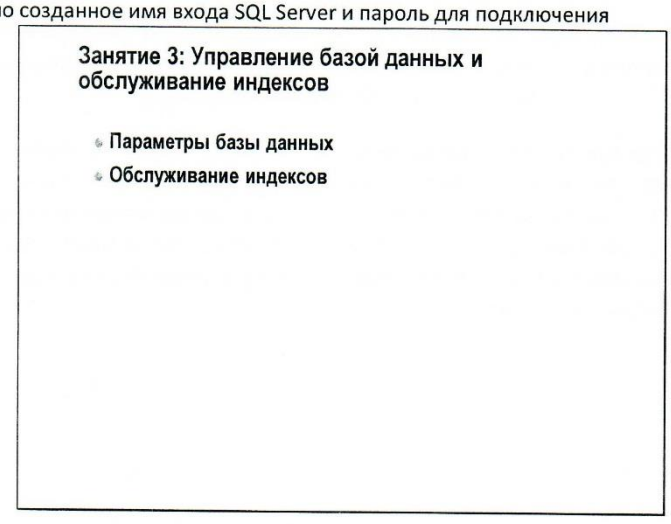
Сжатие файла журнала транзакций не всегда выполняется с первой попытки, поскольку сокращение выполняется отдельными модулями, основной единицей которых является виртуальный файл журнала. В этом случае SQL Server отправляет сообщение о необходимости заархивировать журнал транзакций, чтобы удалить виртуальные файлы журнала в конце файла. Если сокращаемый журнал транзакций содержит неактивные виртуальные файлы в конце, он будет уменьшен на их размер, чтобы размер был максимально приближен к требуемому.

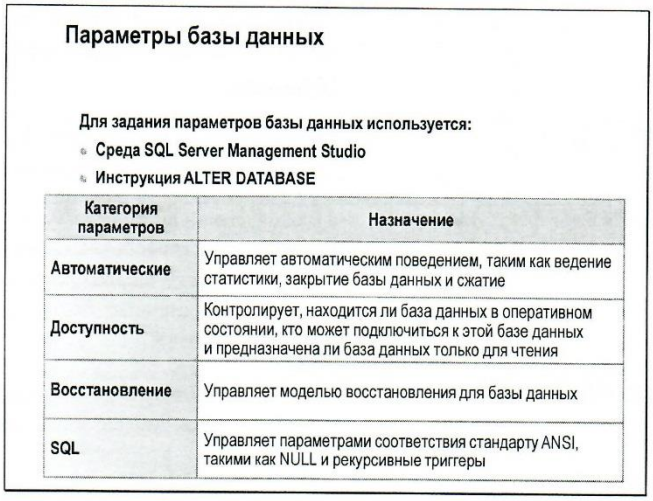


Если сервер системы «1С:Предприятие 8» выполняет соединение с SQL Server и создает базу, все параметры базы принимают значения по умолчанию, которые, как правило, не являются оптимальными.

Предпочтительными являются создание и подготовка базы администратором на SQL Server заранее с оптимальным расположением, количеством файлов и их параметрами. Подключение к существующей базе данных на SQL Server выполняется средствами кластера. При этом администратор кластера системы «1С:Предприятие 8 получает от администратора SQL Server следующие сведения:

- имя или IP-адрес сервера SQL Server
- имя базы данных
- специально созданное имя входа SQL Server и пароль для подключения





Параметры базы данных

Для задания параметров базы данных используется:

- **Среда SQL Server Management Studio**
- **Инструкция ALTER DATABASE**

Категория параметров	Назначение
Автоматические	Управляет автоматическим поведением, таким как ведение статистики, закрытие базы данных и сжатие
Доступность	Контролирует, находится ли база данных в оперативном состоянии, кто может подключиться к этой базе данных и предназначена ли база данных только для чтения
Восстановление	Управляет моделью восстановления для базы данных
SQL	Управляет параметрами соответствия стандарту ANSI, такими как NULL и рекурсивные триггеры

После создания базы данных можно задать параметры баз данных с помощью среды SQL Server Management Studio или с помощью инструкции Transact-SQL ALTER DATABASE.

Следующая таблица содержит сведения о некоторых параметрах.

Параметр	Описание
AUTO_CREATE_STATISTICS	Автоматическое создание отсутствующей статистики, необходимой для оптимизации запроса. Значение по умолчанию — **True**.
AUTO_UPDATE_STATISTICS	Автоматическое обновление устаревших статистических данных. Выполняется по внутреннему алгоритму сервера. Значение по умолчанию — **True**.
Изоляция моментальных снимков read committed включена	Параметр базы для управления режимом изоляции Read committed snapshot. Значение по умолчанию - **False**. Устанавливается и проверяется сервером системы «1С:Предприятие 8» и влияет на работу с базой
LEGACY_CARDINALITY_ESTIMATION = { ON | OFF | PRIMARY } Оценка унаследованной кратности	Указание использовать независимо от уровня совместимости базы версию Cardinality Estimator 70
MAXDOP Максимальная DOP	Уровень параллелизма по умолчанию. 0 – будут использоваться настройки сервера. Эта настройка перекрывает соответствующую на уровне сервера, однако, подсказки на уровне

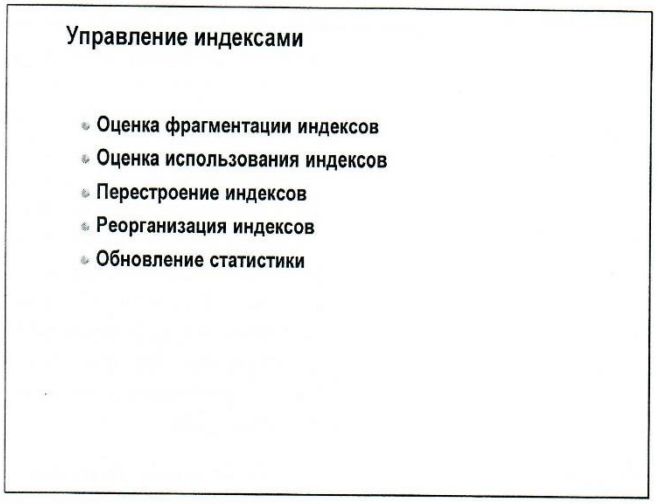
	запросов обладают высшим приоритетом. При этом **Регулятор Ресурсов** применяет свои ограничения
CLEAR PROCEDURE_CACHE	Возможность очистить процедурный кэш только для конкретной базы данных
PARAMETER_SNIFFING = { ON | OFF | PRIMARY}	Включает или выключает Parameter Sniffing. Выключение аналогично флагу трассировки 4136. На уровне запроса этим можно управлять с помощью подсказки OPTIMIZE FOR UNKNOWN.
QUERY_OPTIMIZER_HOTFIXES = { ON | OFF | PRIMARY } Исправления оптимизатора запросов	Оптимизатор запросов со всеми последними обновлениями. Аналог флага трассировки 4199.
SINGLE_USER | RESTRICTED_USER | MULTI_USER	Определяет, кто из пользователей может подключаться к базе данных. SINGLE_USER позволяет подключаться только одному пользователю. RESTRICTED_USER разрешает подключение для участника роли базы данных db_owner и ролей сервера dbcreator и sysadmin. Параметр MULTI_USER позволяет подключаться любому пользователю с соответствующими правами доступа. По умолчанию указывается параметр MULTI_USER.
Уровень совместимости 100 |110 |120| 130|140	Значение 130 и 140 обеспечивает работу нового алгоритма автоматического обновления статистики
Модель восстановления Полная (FULL)| Простая (SIMPLE)| С неполным протоколированием (BULK_LOGGED)	Значение FULL указывает на возможность полного восстановления в случае сбоя носителя, оно является значением по умолчанию. Значение BULK_LOGGED — используется меньшая часть пространства журнала, так как в журнал заносится минимальная часть данных, но уровень защищенности системы снижается. Значение SIMPLE восстанавливает базу данных только до последней полной резервной копии базы данных или до последней разностной резервной копии.
Тип включения	PARTIAL используется в автономных базах
PAGE_VERIFY	Позволяет SQL Server определять неполные операции ввода-вывода, причиной которых стал сбой питания или другие системные сбои.

	Параметр CHECKSUM позволяет сохранить в заголовке страницы значение, рассчитанное на основе содержимого страницы. Это значение рассчитывается повторно и сравнивается с сохраненной версией при считывании страниц с данными с диска. Это значение задается по умолчанию. TORN_PAGE_DETECTION сохраняет определенный бит для каждого 512-байтного сектора в странице данных, объемом 8 КБ, как часть заголовка страницы. Эти биты, сохраненные в заголовке страницы, сравниваются с фактическими данными сектора страницы при считывании страниц с данными с диска.
TARGET_RECOVERY_TIME	Время восстановления целевого объекта в секундах. Влияет на частоту контрольных точек.

Задание 2. Создание базы данных системы «1С:Предприятие 8»

Параметры файлов и свойства базы

1. Запустите «1С:Предприятие 8» на своей машине и создайте информационную базу на сервере приложений, используя следующие данные.
   - Кластер серверов «1С:Предприятие 8» – имя предоставит инструктор
   - Имя информационной базы в кластере - **DB1C**X, где X соответствует двум последним цифрам вашего IP-адреса.
   - Тип СУБД - SQL Server
   - SQL Server - IP адрес вашей машины.
   - Имя SQL базы **DB1C**.
   - Имя пользователя **sa**.
   - Пароль, заданный во время установки SQL Server.
   - Отметьте опцию создания SQL базы.
   - Установите блокировку регламентных заданий
2. Запустите **SQL Server Management Studio**
   - Выберите базу **DB1C**.
3. Просмотрите свойства базы в ее контекстном меню и ответьте на вопросы:
   - Почему параметры имеют такие значения?
   - Какие параметры можно изменить и для чего?
4. Выполните настройку параметров файлов базы данных и журнала транзакций, увеличив размер файла данных на **2мб**.
5. Добавьте новый файл данных **NewData** размером **10мб** к базе.
6. Просмотрите для базы отчет о занимаемом дисковом пространстве
7. Удалите файл **NewData**



Ряд динамических административных представлений (DMV) и функций (DMF), входящих в состав SQL Server, может помочь администраторам баз данных оценить эффективность индексов и определить проблемы с производительностью.

Две функции, а именно: sys.dm_db_index_physical_stats, и sys.dm_db_index_operational_stats и представление sys.dm_db_index_usage_stats, позволяют понять, работают ли индексы так, как планировалось. С их помощью можно посмотреть, как ведут себя индексы в ходе операций ввода-вывода и при блокировках, а также определить, действительно ли оптимизатор запросов применяет индексы так, что это не приводит к ненужному состязанию в базе данных.

Оценка фрагментация индексов

Для просмотра сведений о фрагментации индекса в среде SQL Server Management Studio, откройте окно **Свойства** для нужного индекса, а затем выберите страницу **Фрагментация**. Кроме ряда основных свойств страниц индекса, окно **Свойства** показывает среднее заполнение страниц и общую фрагментацию по индексу в виде процента. Чем больше это значение, тем больше фрагментирован индекс. Можно просматривать графический отчет **Физическая статистика индексов** Функция DMF sys.dm_db_index_physical_stats показывает фрагментацию индекса. Функция sys.dm_db_index_physical_stats устанавливает только намеренную общую блокировку (IS), что позволяет значительно уменьшить блокирование таблицы во время выполнения функции. Чтобы определить уровень фрагментации индекса с помощью функции sys.dm_db_index_physical_stats, нужно изучить значение некоторых столбцов результатов выполнения функции. Логическую фрагментацию индексов (фрагментацию экстентов куч) можно определить по значению в столбце avg_fragmentation_in_percent. Это то же значение, что и в окне свойств индекса в среде SQL Server Management Studio. Логическая фрагментация — это процент неупорядоченных страниц на конечном уровне индекса, а фрагментация экстентов — это процент неупорядоченных экстентов на конечном уровне индекса. Нужно стараться, чтобы уровни как логической фрагментации, так и фрагментации экстентов были настолько близок к нулю, насколько это возможно.

Внутренняя фрагментация индекса — это процент заполненности страниц. Конечно, хотелось бы, чтобы страница индекса была заполнена настолько, насколько возможно, но нужно еще и соблюдать баланс

между заполненностью и числом вставок в страницы индекса, чтобы число разбиений страниц было минимальным.

Узнать заполненность страниц индекса можно с помощью аргумента avg_page_space_used_in_percent функции sys.dm_db_index_physical_stats. Чтобы правильно определить, насколько близко это число должно быть к 100 процентам, нужно настроить коэффициент заполнения индекса, одновременно наблюдая за числом происходящих разбиений страниц. Начиная с некоторого момента число разбиений страниц начнет расти очень быстро. Это означает, что для коэффициента заполнения индекса было задано слишком высокое значение.

Например, чтобы определить уровень фрагментации всех индексов в базе DB1C, можно воспользоваться такой инструкцией:

SELECT * FROM sys.dm_db_index_physical_stats (DB_ID ('DB1C'),NULL,NULL,NULL, 'LIMITED')

С помощью данной функции DMF можно автоматически определить, какие индексы должны быть перестроены, какие нуждаются в реорганизации, а какие можно не трогать. Выявление значений столбцов avg_page_space_used_in_percent и avg_fragmentation_in_percent этой функции DMF, превышающих некоторый логический порог и порог плотности, поможет определить, какие операции необходимо выполнить с этим индексом. Результаты запросов можно записать в табличную переменную, а затем просмотреть эту переменную, чтобы построить динамическую строку для правильной инструкции ALTER INDEX

Оценка использования индексов

На практике часто возникает задача: определить, какие индексы действительно используются при выполнении запросов данных таблицы. Часто разработчики или администраторы баз данных создают для таблицы индексы, которые, как они думают, оптимизатор запросов будет использовать при выполнении запроса. Динамическое административное представление, sys.dm_db_index_usage_stats, — это простой способ определить, как индексы используются оптимизатором запросов и запросами данных из таблицы. Изучив результаты этого представления DMV на предмет индексов с нулевым числом операций поиска и просмотра, можно определить, использовался ли индекс с момента последнего запуска сервера SQL Server. Впрочем, необходимо помнить, что результаты многих динамических административных представлений и функций не сохраняются и сбрасываются обратно в ноль после перезапуска сервера SQL Server. Не следует забывать об этом, применяя представление DMV или функцию DMF для оценки использования индекса. Возможно, индекс просто ни разу не понадобился с момента последнего перезапуска службы, но он потребуется для запросов при составлении недельных, месячных или квартальных отчетов. Чтобы просмотреть все индексы, которые не использовались на данном сервере с момента последнего перезапуска службы сервера SQL Server, можно воспользоваться следующей инструкцией:

SELECT DB_NAME (database_id), OBJECT_NAME ([object_id]) FROM sys.dm_db_index_usage_stats WHERE user_seeks = 0 AND user_scans = 0 AND user_lookups = 0 AND system_seeks = 0 AND system_scans = 0 AND system_lookups = 0

Можно выставить все параметры по умолчанию и не накладывать фильтр на столбцы и строки, кроме идентификатора базы, например:

SELECT * FROM sys.dm_db_index_usage_stats WHERE database_id = DB_ID ('DB1C')

Активность операций индексов

Для определения активности операций индексов очень полезной может оказаться функция DMF sys.dm_db_index_operational_stats. Ее можно использовать для просмотра активности операций ввода-вывода, блокировок, кратковременных блокировок и метода доступа для каждого индекса в базе данных. Такая информация помогает понять, как используются индексы, и отметить случаи блокировки индексов из-за высокой активности операций ввода-вывода или из-за существования в индексе проблемной области. С помощью столбцов latch wait данной функции DMF можно определить, сколько времени требуется операциям READ и WRITE для получения доступа к ресурсам индекса. Это позволяет понять, соответствует ли дисковая подсистема, которая используется для хранения индекса, активности его операций ввода-вывода. Кроме того, если неудачная структура или неправильное использование индекса привели к появлению проблемной области, в которой высокая активность на одной или нескольких страницах индекса вызывает состязание для данных этих страниц, это тоже будет видно из полученных результатов. Такое состязание часто приводит к избыточному блокированию операций READ или WRITE для данной области.

Представление sys.dm_db_missing_index_details возвращает подробные сведения о желательных, но отсутствующих индексах.

Существует два способа дефрагментации индекса: реорганизация и перестроение. Реорганизация индекса дефрагментирует конечный уровень кластеризованных и некластеризованных индексов, физически изменяя порядок страниц конечного уровня для соответствия логическому порядку (слева направо) узлов конечного уровня. Упорядочивание страниц улучшает производительность просмотра индексов. Индекс реорганизуется внутри существующих страниц, выделенных для индекса, новые страницы не выделяются. Реорганизация индекса также сжимает страницы индекса. Все пустые страницы, созданные этим сжатием, удаляются, высвобождая дисковое пространство. Сжатие основано на значении коэффициента заполнения в представлении каталога sys.indexes. Реорганизация индекса выполняется в оперативном режиме и может быть прервана с сохранением результатов. Перестроение индекса удаляет индекс и создает новый. Перестроение реализовано как одна транзакция. При этом фрагментация исчезает, а дисковое пространство освобождается с помощью сжатия страниц, используя заданное или существующее значение коэффициента заполнения, строки индекса упорядочиваются заново в смежных страницах (при необходимости выделяются новые страницы). Это может повысить быстродействие диска, уменьшая число чтений страниц, необходимое для получения запрошенных данных. В результате перестроения индекса сохраняется обновленное значение статистики.

Реорганизация индекса статистику не обновляет.

Перестроение всех индексов в таблице и указание параметров

В данном примере указывается ключевое слово ALL. Так можно перестроить все индексы, связанные с таблицей.

```
USE DB1C;

GO

ALTER INDEX ALL ON dbo.v8users REBUILD;

GO
```

Реорганизация индексов

ALTER INDEX ALL ON dbo.v8users REORGANIZE;

GO

Сравнение реорганизации и перестроения индексов

Решение о том, реорганизовывать или перестраивать индекс для устранения дефрагментации, должно основываться на существующем уровне фрагментации индекса, сообщаемого средой SQL Server Management Studio в отчете о физической статистике индексов или функцией **sys.dm_db_index_physical_stats**. Рекомендации по оптимальному подходу к устранению дефрагментации различной степени приведены в следующей таблице. Фрагментация меньше 5% не рассматривается

avg_fragmentation_in_percent	Действие
< 30%	Реорганизовать
>=30%	Перестроить

Редакция SQL Server 2017 Enterprise допускает онлайновое перестроение индексов (Online Index Rebuild).

При онлайновом перестроении индексов SQL Server накладывает Shared Table Lock (S) в начале операции и Schema Modification Lock (Sch-M) на соответствующую таблицу в конце операции. SQL Server позволяет контролировать блокировки в начале и в конце операции Online Index Rebuild. Пример.

ALTER INDEX ALL ON **dbo.v8users** REBUILD WITH (ONLINE = ON (WAIT_AT_LOW_PRIORITY (MAX_DURATION = 1, ABORT_AFTER_WAIT = SELF)))

WAIT_AT_LOW_PRIORITY определяет, что произойдет в случае блокирования MAX_DURATION указывает, сколько минут

ABORT_AFTER_WAIT указывает, какую сессию откатывать. SELF означает что откатится команда ALTER INDEX REBUILD

Если указать BLOCKERS, то откатится блокирующая сессия.

В SQL Server 2017 Enterprise появилась возможность использовать возобновляемые операции онлайнового перестроения индекса. Перестроение можно приостановить и потом продолжить. Таким образом стало выполнимым обслуживание индекса по частям. Возможно также возобновление перестроения после сбоя. Параметр RESUMABLE = ON в команде ALTER INDEX управляет возможностью возобновления операции. Представление sys.index_resumable_operations позволяет отслеживать онлайновое перестроение индексов. Также допускается очистка журнала транзакций во время онлайнового перестроения индекса.

Обновление статистики

В SQL Server 2017 при автоматическом обновлении статистики для уровня совместимости 130 и 140 используется пороговое значение, которое изменяется в зависимости от числа строк в таблице. Благодаря этому изменению статистика для больших таблиц будет обновляться чаще. Хранимая процедура sp_updatestats обновляет в базе данных все статистики, требующие обновления в соответствии с внутренним критерием SQL Server (изменение одной записи). Для определения размера выборки используется нелинейный алгоритм. Выборка данных для формирования статистики выполняется в параллельном режиме (на уровне совместимости 130 и 140), что повышает производительность сбора статистики.

Асинхронная статистика рекомендуется для достижения более прогнозируемого времени ответа на запросы в следующих сценариях.

•Приложение часто выполняет один и тот же запрос, схожие запросы или схожие кэшированные планы запроса. Асинхронное обновление статистики может обеспечить более прогнозируемое время ответа на запрос по сравнению с синхронным обновлением статистики, поскольку оптимизатор запросов может выполнять входящие запросы, не ожидая появления актуальной статистики. Это устраняет задержку в некоторых запросах, но не влияет на другие запросы.

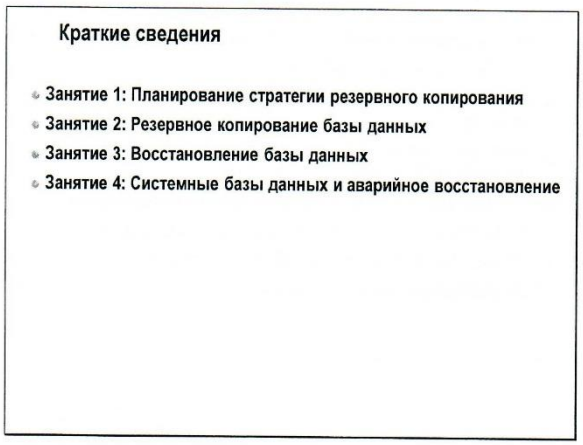
•Были случаи, когда в приложении истекало время ожидания клиентских запросов в результате ожидания обновленной статистики. В некоторых случаях ожидание синхронной статистики может вызвать аварийное завершение приложений, в которых задано малое время ожидания.

Задание 3. Обслуживание индексов базы данных системы «1С:Предприятие 8 и обновление статистики.

Работа по сопровождению индексов

1. Запустите на своей машине **SQL Server Management Studio** зарегистрируйте **Database Engine** локального сервера, используя проверку подлинности Windows
2. Выберите базу **DB1C**.
3. В обозревателе объектов в базе **DB1C** выберите любую таблицу, содержащую индексы.
4. В разделе **Индексы** для таблицы выберите индекс и в его свойствах просмотрите параметры и оцените фрагментацию
5. В разделе **Статистики** просмотрите параметры статистик и дату обновления
6. Оцените фрагментацию индексов во всех таблица базы с помощью отчета **Физическая статистика индексов**
7. Оцените использование индексов в базе с помощью отчета **Статистика использования индексов**
8. Найдите в интернете на сайте MSDN описание функции динамического управления **sys.dm_db_index_physical_stats**. Найдите в примерах использования функции пример Г., содержащий сценарий использования результатов выполнения функции **sys.dm_db_index_physical_stats** для перестроения и реорганизации индексов. Скопируйте скрипт в окно нового запроса. Сохраните скрипт в файл с именем **dbreindex.sql**. Выполните сценарий.
9. Выполните необходимые обновления статистик в базе с помощью вызова хранимой процедуры **exec sp_updatestats**

Раздел 3: Резервное копирование и восстановление баз данных

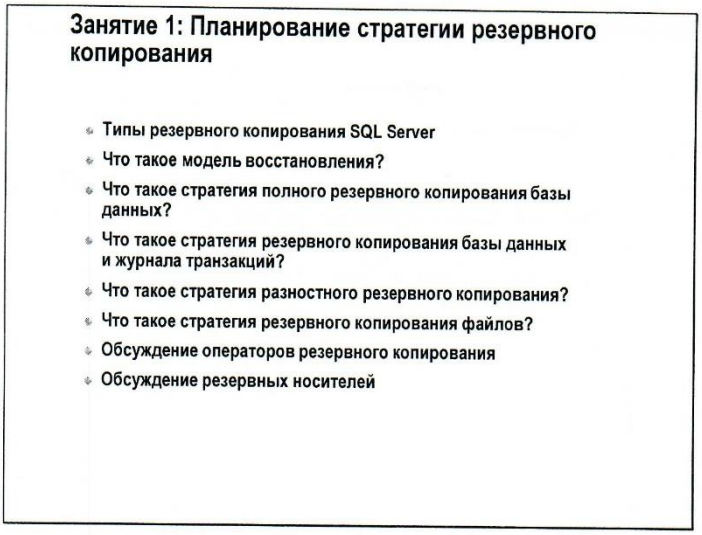


В каждой системе управления базой данных должны быть соответствующие процедуры аварийного восстановления. Операции резервного копирования и восстановления составляют жизненно важную часть управления данными. Поэтому одной из главных обязанностей администратора базы данных является обеспечение резервного копирования данных и их быстрого восстановления в случае возникновения аварии. Расписание резервного копирования и типы резервных копий подбирают исходя из требований к точке восстановления и к времени для восстановления данных. К регламентным работам относится не только создание резервных копий, но регулярное тестирование восстановления. Резервные копии базы данных содержат ценную информацию и требуют надежного хранения.

Цели

После изучения данного раздела вы сможете:

- планировать стратегию резервного копирования базы данных;
- выполнять резервное копирование пользовательских баз данных;
- восстанавливать пользовательские базы данных из резервных копий;
- откатить состояние базы данных к моменту создания моментального снимка;



Типы резервного копирования SQL Server

Тип резервной копии	Описание
Полная	Все файлы данных и часть журнала транзакций
Журнал транзакций	Любые изменения базы данных, записанные в файл журнала
Заключительные фрагменты журнала	Активная часть журнала
Разностная	Части базы данных, которые изменились с момента выполнения полного резервного копирования базы данных
Файл / файловая группа	Указанные файлы или файловые группы
Доступная только для копирования	База данных или журнал (не оказывается влияния на последовательность резервного копирования)

В SQL Server предоставляется несколько методов резервного копирования для удовлетворения требований всевозможных сфер бизнеса и разнообразных применений баз данных.

Полные резервные копии

Полная резервная копия базы данных содержит файлы данных и часть журнала транзакций. Полная резервная копия представляет базу данных на момент создания резервной копии и служит основным источником данных в случае сбоя системы. При осуществлении полного резервного копирования базы данных сервером SQL Server выполняются следующие действия:

• резервное копирование всех данных в базе данных;

• резервное копирование всех изменений, которые возникают во время выполнения резервного копирования;

• резервное копирование всех транзакций, не зафиксированных в журнале транзакций.

Сервером SQL Server используются части журнала транзакций, которые были записаны в файл резервной копии для обеспечения согласованности данных при восстановлении резервной копии. Восстановленная база данных совпадает с состоянием базы данных на момент завершения резервного копирования за исключением всех незафиксированных транзакций. При восстановлении базы данных производится откат незафиксированных транзакций.

Если база данных доступна только для чтения, возможно, полных резервных копий будет достаточно для предотвращения потери данных.

Резервные копии журнала транзакций

В резервные копии журнала транзакций записываются все изменения базы данных. Резервное копирование журналов транзакций обычно выполняется при создании полных резервных копий базы данных. Обратите внимание на следующие факты, касающиеся резервных копий журналов транзакций:

• не следует выполнять резервное копирование журнала, если хотя бы раз не создавалась полная резервная копия базы данных;

• журналы транзакций невозможно восстановить без соответствующей резервной копии базы данных;

• при использовании простой модели восстановления невозможно создать резервные копии журналов транзакций.

При резервном копировании журнала транзакций сервером SQL Server выполняется следующее:

• Создаются резервные копии журнала транзакций от последней успешно выполненной инструкции BACKUP LOG до конца текущего журнала транзакций.

• Очищаются (отмечаются как неактивные) части журнала транзакций до начала активной части и отбрасываются сведения в неактивной части.

Активная часть журнала транзакций начинается с момента самой последней открытой транзакции и продолжается до конца журнала транзакций.

Резервные копии заключительных фрагментов журнала

Резервная копия заключительных фрагментов журнала — это резервная копия журнала транзакций, включающая часть журнала, которая ранее не подвергалась резервному копированию (известна как активная часть журнала). Резервное копирование заключительных фрагментов журнала осуществляется без усечения журнала и обычно используется, когда файлы данных становятся недоступными для базы данных, но файл журнала не поврежден.

Разностные резервные копии

Разностное резервное копирование следует выполнять для минимизации времени, которое необходимо для восстановления часто изменяемой базы данных. Разностное резервное копирование возможно только в том случае, когда создана полная резервная копия базы данных. Когда создаются разностные резервные копии, сервером SQL Server выполняются следующие действия:

• Создаются резервные копии частей базы данных, которые изменились с момента выполнения полного резервного копирования базы данных.

• Создаются резервные копии всех операций, происходивших во время разностного резервного копирования, а также всех транзакций, не зафиксированных в журнале транзакций.

Резервные копии файлов и файловых групп

Если выполнение полного резервного копирования очень больших баз данных нецелесообразно с практической точки зрения, можно создать резервные копии файлов и файловых групп базы данных. Когда создаются резервные копии файлов и файловых групп, сервером SQL Server выполняются следующие действия:

• Создаются резервные копии только файлов базы данных, которые указаны в параметре FILE или FILEGROUP.

• Разрешается резервное копирование конкретных файлов базы данных вместо всей базы данных.

При создании резервных копий файлов и файловых групп необходимо:

• указать логические файлы и файловые группы;

• создать резервные копии журнала транзакций, чтобы восстанавливаемые файлы согласовывались с остальной базой данных;

• создать план резервного копирования каждого файла на циклической основе, чтобы обеспечить регулярное резервное копирование всех файлов и файловых групп базы данных.

Резервные копии данных, доступных только для копирования

В SQL Server поддерживается создание резервных копий данных, доступных только для копирования. В отличие от других резервных копий резервная копия данных, доступных только для копирования, не влияет на общие процедуры резервного копирования и восстановления, которые выполняются для базы данных. Резервные копии данных, доступных только для копирования, могут использоваться для создания копии архива с целью его хранения в надежном помещении вне рабочего места. Резервные копии данных, доступных только для копирования, также удобны, когда необходимо выполнить некоторые операции восстановления в интерактивном режиме. Резервные копии данных, доступных только для копирования, поддерживаются всеми моделями восстановления. Резервную копию данных, доступных только для копирования, можно создать для любого типа резервного копирования. Резервная копия данных, доступных только для копирования, не может использоваться как базовая резервная копия и не влияет на любые существующие разностные резервные копии. Разностные резервные копии данных, доступных только для копирования, идентичны обычным разностным резервным копиям.

Что такое модели восстановления?

Модель восстановления	Описание
Простая	Использует полные или разностные резервные копии базы данных. Усекает журналы транзакций (освобождает место)
Полная	Включает резервные копии как базы данных, так и журнала транзакций
С неполным протоколированием	Включает резервные копии как базы данных, так и журнала транзакций, но использует меньше пространства журнала для некоторых операций

В SQL Server имеется три модели восстановления базы данных: *простая*, *полная* и *с неполным протоколированием*. Каждая из моделей сохраняет данные в случае сбоя сервера, но между моделями существуют основные различия в восстановлении данных сервером SQL Server. Модель восстановления можно установить или изменить в любой момент, однако модель восстановления следует планировать при создании базы данных.

Простая модель восстановления

Простая модель восстановления обычно используется для малых баз данных или баз данных, в которых данные изменяются редко. В этой модели используются полные или разностные копии базы данных, и восстановление ограничивается восстановлением базы данных до момента, когда была создана последняя резервная копия. Все изменения, внесенные после создания резервной копии, утрачиваются. Основное преимущество этой модели заключается в том, что для хранения журналов требуется меньше места и это самая простая модель для реализации. Ведение журнала в этой модели упрощенное, записываются только сведения, необходимые для отката транзакции.

Полная модель восстановления

Полную модель восстановления можно использовать, когда наивысший приоритет имеет полное восстановление с поврежденного носителя. В этой модели для восстановления базы данных используются копии базы данных и все сведения журнала. Сервером SQL Server заносятся в журнал все изменения базы данных, включая массовые операции и операции создания индексов. Если сами журналы не повреждены, сервером SQL Server могут быть восстановлены все данные за исключением транзакций, которые обрабатывались на момент сбоя. Поскольку все транзакции записаны в журнал, восстановление может быть выполнено до любого момента времени. Сервером SQL Server поддерживается вставка именованных меток в журнал транзакций, что позволяет осуществлять восстановление до конкретной метки. Так как метки транзакций занимают место в журнале, их следует использовать только для транзакций, которые играют важную роль в стратегии восстановления базы данных. Основное ограничение этой модели — большой размер файлов журналов и итоговые затраты памяти и процессорного времени. До выполнения первого резервного копирования базы данных полная модель работает как простая, контрольная точка очищает журнал транзакций.

Модель восстановления с неполным протоколированием

В модели восстановления с неполным протоколированием для восстановления базы данных используются резервные копии как базы данных, так и журнала. Однако в модели восстановления с неполным протоколированием требуется меньше места для следующих операций: CREATE INDEX, операции массовой загрузки, SELECT INTO, WRITETEXT и UPDATETEXT. Вместо хранения в журнале сведений об операциях в нем отмечается только наличие этих операций в виде разрядов в экстентах.

Стратегия полного резервного копирования базы данных — это метод восстановления, включающий в себя создание регулярных полных резервных копий базы данных. Если база данных повреждена, можно воспользоваться самой последней полной резервной копией, чтобы восстановить базу данных до состояния, в котором она находилась на момент создания резервной копии. Время и ресурсы, необходимые для реализации стратегии полного резервного копирования базы данных, определяются размером базы данных и частотой изменения данных.

Когда следует применять стратегию полного резервного копирования базы данных?

Применяйте стратегию полного резервного копирования базы данных в следующих случаях:

• База данных имеет небольшой размер. Резервное копирование небольшой базы данных выполняется в течение приемлемого времени.

• База данных подвергается незначительным изменениям или доступна только для чтения. При выполнении полного резервного копирования фиксируется достаточно полный набор данных. Возможно, придется смириться с небольшими потерями данных, если база данных повредится между резервными копированиями и ее потребуется восстановить.

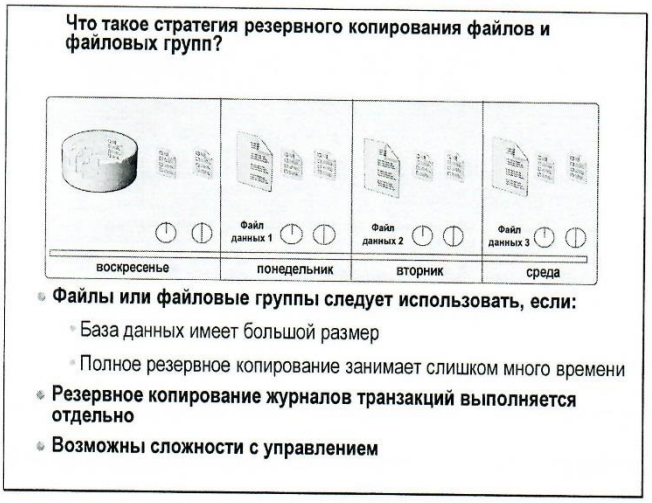
Когда используется простая модель восстановления, все зафиксированные транзакции записываются в базу данных при достижении контрольной точки, а журнал транзакций автоматически усекается. В журнале транзакций не содержатся изменения, которые вносились в базу данных с момента создания последней полной резервной копии базы данных.

При реализации стратегии резервного копирования базы данных и журнала транзакций можно восстановить базу данных из самой последней полной резервной копии базы данных, а затем применить все резервные копии журнала транзакций, которые были созданы с момента последнего полного резервного копирования.

Применяйте стратегию полного резервного копирования базы данных и журнала транзакций для часто изменяемых баз данных. Следует также проанализировать, можно ли выполнить резервное копирование базы данных и журналов транзакций за приемлемое время.

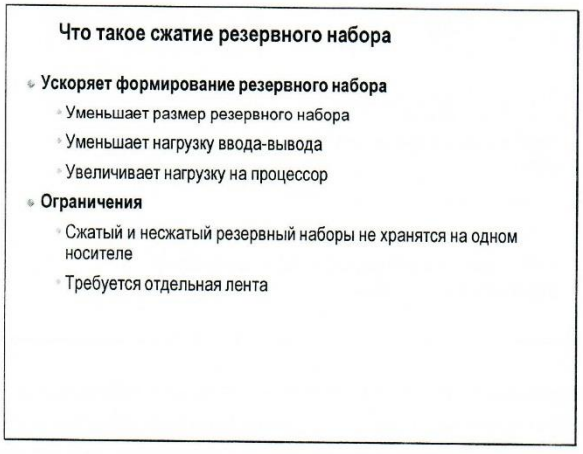
Стратегия разностного резервного копирования включает создание регулярных полных резервных копий базы данных с промежуточными разностными резервными копиями. Между полными и разностными резервными копированиями можно также дополнительно выполнять резервные копирования журнала транзакций. Чтобы восстановить базу данных в случае аварии, необходимо восстановить самую последнюю полную резервную копию базы данных, после этого самую последнюю разностную резервную копию и затем в порядке очередности восстановить каждый журнал транзакций

с момента создания последней разностной резервной копии. Используйте эту стратегию для уменьшения времени восстановления, если база данных повреждена.



Стратегия резервного копирования файлов и файловых групп включает резервное копирование отдельных файлов и файловых групп, выполняемое на регулярной основе. Обычно эта стратегия реализуется путем поочередного резервного копирования всех файлов и файловых групп, доступных для чтения и записи. Кроме того, обычно между резервными копированиями файлов и файловых групп выполняется резервное копирование журнала транзакций. Однако эта стратегия сложна и автоматически не поддерживает целостность ссылок.

Используйте эту стратегию для очень большой базы данных, которая секционирована на множество файлов. При объединении с регулярными резервными копированиями журналов транзакций этот метод представляет впечатляющую по времени альтернативу полным резервным копированиям базы данных.



После установки сжатие резервных наборов по умолчанию отключено. Изменить настройку можно как для всего сервера, так и для отдельной команды резервного копирования. Коэффициент сжатия можно получить следующей командой

SELECT backup_size/compressed_backup_size FROM msdb.dbo.backupset

Для резервного копирования базы данных SQL Server требуются специальные права, отображенные на рисунке.

Носители, поддерживаемые SQL Server

Резервное копирование может выполняться сервером SQL Server в файл на жестком диске или на ленту, или в облако (управляемое резервное копирование). Дисковые файлы (локальные или сетевые) являются наиболее распространенными носителями, используемыми для хранения резервных копий. Прямое резервное копирование на ленту не поддерживается.

Что такое устройство резервного копирования?

Файл резервной копии, определяемый до того, как он будет использоваться для операции резервного копирования, называется устройством резервного копирования. Устройства резервного копирования можно создавать с помощью SQL Server Management Studio или путем выполнения системной хранимой процедуры **sp_addumpdevice**.

Хранение резервных копий в нескольких файлах

Сервером SQL Server может одновременно (параллельно) вестись запись в несколько файлов резервных копий. Когда имеется несколько файлов резервных копий, данные распределены по всем файлам, которые используются для создания резервной копии. В этих файлах хранится разбитый на части резервный набор данных. Резервный набор данных является результатом одиночной операции резервного копирования, выполняемой над одним или несколькими файлами. Резервное копирование можно выполнять на несколько лент или контроллеров дисков, чтобы уменьшить общее время резервного копирования базы данных. При использовании нескольких файлов для хранения резервных копий примите во внимание следующие сведения:

• Все устройства, используемые в одиночной операции резервного копирования, должны относиться к одному и тому же типу носителей (диск). Набор носителей — это коллекция файлов, используемых для хранения одного или нескольких резервных наборов данных.

• При создании резервного набора данных можно использовать комбинацию постоянных и временных файлов.

• Не допускается использовать только один элемент резервного набора данных для операции резервного копирования, если файлы не переформатированы.

• Если переформатировать один элемент резервного набора данных, данные, содержащиеся в других элементах резервного набора данных, станут недействительными и непригодными для использования.

Шифрование резервной копии

Резервное копирование представляет собой фундаментальное требование защиты данных. Резервные копии необходимо обезопасить от несанкционированного доступа. Для этого можно хранить резервные копии в безопасном месте на файловой системе.

Для использования шифрования резервной копии необходимо:

1.Создать главный ключ базы данных в базе данных master. Это симметричный ключ, который используется для защиты всех ключей шифрования и сертификатов в базе. Пример

```
USE master;
GO
CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'My BAK Pa$$w0rd';
GO
```

2. Создать сертификат или несимметричный ключ, которым будет зашифрован резервный набор. В экземпляре SQL Server для этого используются команды CREATE CERTIFICATE или CREATE ASYMMETRIC KEY. Несимметричный ключ должен храниться внешним поставщиком шифрования (EKM).

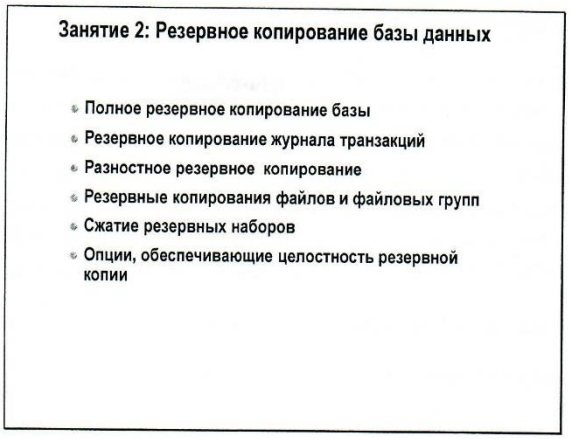
```
CREATE CERTIFICATE MyServerCert WITH SUBJECT = 'My BAK Certificate'
GO
BACKUP certificate MyServerCert to file = 'C:\MyServerCert.cer'
WITH PRIVATE KEY (FILE='C:\keys\SuperSAfeBackupCertificate.ppk',
ENCRYPTION BY PASSWORD ='PasswordToEncryptPrivateKey123'
```

3. Создать резервную копию с опцией ENCRYPTION, указав алгоритм и сертификат или несимметричный ключ и использовать новый набор носителей.

Необходимо создать резервную копию главного ключа базы и ключей шифрования на отдельный носитель, чтобы можно было восстановить резервную копию базы на другом экземпляре SQL Server

```
BACKUP DATABASE [DB1C] TO DISK = N'C:\Backup\db1c.bak' WITH FORMAT, INIT,  NAME = N'db1c-Full Database Backup', SKIP, NOREWIND, NOUNLOAD, ENCRYPTION(ALGORITHM = AES_256, SERVER CERTIFICATE = [MyServerCert]),  STATS = 10
```

Зашифрованный резервный набор должен быть единственным на носителе.

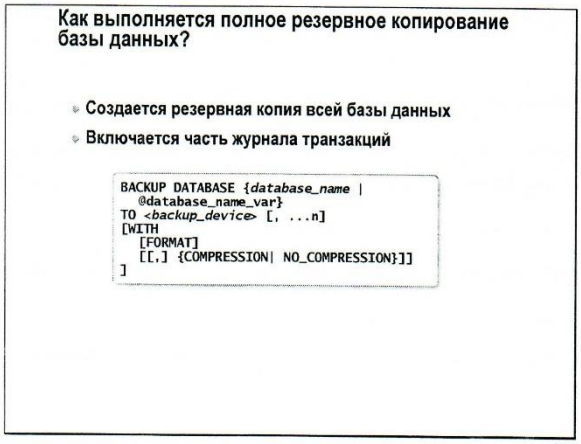


Операции резервного копирования можно выполнять с помощью SQL Server Management Studio или языка программирования Transact-SQL. На этом занятии будет рассмотрен синтаксис Transact-SQL для разных методов резервного копирования, включая параметры для проверки достоверности резервной копии. Знание этих инструкций Transact-SQL позволяет выполнять резервное копирование баз данных с большей гибкостью, чем при использовании только среды SQL Server Management Studio.

Для выполнения резервного копирования в облако нужно иметь подписку Windows Azure.

Затем нужно создать Azure Storage Account, который будет использован для хранения резервных копий. Далее создаются учетные данные (Credential) на основе Azure Storage Account. Резервное копирование можно выполнить с помощью графического интерфейса или командой следующего вида.

BACKUP DATABASE DB1C TO URL = 'AzStoreAct.blob.core.windows.net/backups/db1c.bak' WITH CREDENTIAL = 'AzureStore';



Большинство стратегий резервного копирования включают полное резервное копирование базы данных. Полное резервное копирование можно выполнять с помощью обозревателя объектов в среде SQL Server Management Studio или с помощью инструкции BACKUP DATABASE языка программирования Transact-SQL.

Выполнение полного резервного копирования базы данных

Чтобы выполнить полное резервное копирование базы данных, в обозревателе объектов щелкните базу данных правой кнопкой мыши, в контекстном меню наведите указатель на пункт **Задачи** и выберите пункт **Резервное копирование.** В итоговом диалоговом окне **Резервное копирование базы данных** выберите **Полное** в качестве значения параметра **Тип резервного копирования**, и укажите, что должно выполняться резервное копирование базы данных. В альтернативном варианте полное резервное копирование базы данных можно выполнить с помощью инструкции BACKUP DATABASE. Частичный синтаксис инструкции BACKUP DATABASE показан в следующем программном коде Transact-SQL.

```
BACKUP DATABASE {database_name | @database_name_var}

TO <backup_device> [, ...n] [WITH [FORMAT] [[,] {INIT | NOINIT}]]
```

При резервном копировании базы данных определите, должен ли файл резервной копии перезаписываться или новые данные следует добавлять в этот файл. Сервер SQL Server настроен по умолчанию на добавление (NOINIT) резервных копий в файл. Если используется параметр NOINIT, резервная копия добавляется сервером SQL Server в существующий файл резервной копии или в резервный набор данных. Если используется параметр INIT, сервером SQL Server перезаписываются все существующие данные на наборе резервных носителей, но сохраняются сведения заголовка. Если первый файл резервного набора данных в устройстве имеет метку стандарта ANSI, сервером SQL Server определяется возможность перезаписи предыдущего набора резервных данных. Операция резервного копирования завершается сбоем, и данные не перезаписываются в следующих случаях:

• Срок действия параметра EXPIREDATE, заданного для устройства резервного копирования, еще не истек.

• Параметры *backup_set_name,* заданные в настройке NAME, не соответствуют параметрам *backup_set_name* в устройстве резервного копирования.

• Предпринимается попытка перезаписи одного элемента ранее именованного резервного набора данных.

• Сервером SQL Server обнаружено, что файл является элементом резервного набора данных.

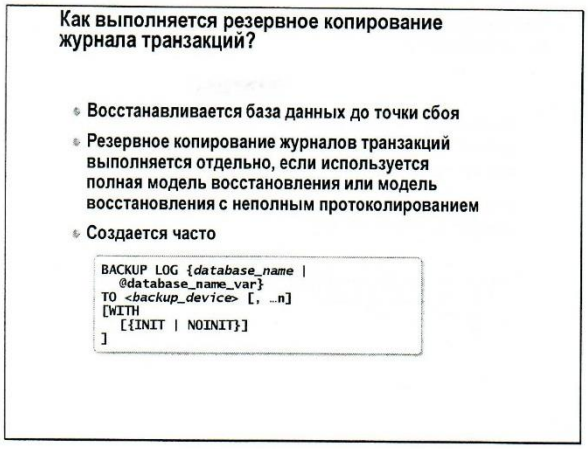
Чтобы перезаписать содержимое файла резервной копии и разделить резервный набор данных, воспользуйтесь параметром FORMAT. Когда указан параметр FORMAT, выполняются следующие действия:

• Во все файлы, используемые для этой операции резервного копирования, записывается заголовок нового носителя.

• Сервером SQL Server перезаписываются существующие носители и содержимое файла резервной копии.

Пользуйтесь параметром FORMAT аккуратно. При форматировании только одного файла резервной копии из набора носителей весь набор резервных данных становится непригодным для использования.

Например, если одна лента, содержащая часть существующего резервного набора данных, разделенных на части, подвергается переформатированию, весь резервный набор данных становится непригодным для использования.



В полной модели восстановления и в модели восстановления с неполным протоколированием необходимо регулярно создавать резервные копии журналов транзакций для восстановления данных. С помощью резервных копий журнала транзакций базу данных можно восстановить до точки сбоя или до определенного момента времени.

Резервными копиями журналов транзакций обычно используется меньше ресурсов, чем полными резервными копиями. В результате, их можно создавать чаще, чем полные резервные копии, при этом уменьшается риск потери данных.

Выполнение резервного копирования журналов транзакций

Чтобы выполнить резервное копирование журнала транзакций, в обозревателе объектов щелкните базу данных правой кнопкой мыши, в контекстном меню наведите указатель мыши на пункт **Задачи** и выберите пункт **Резервное копирование.** В итоговом диалоговом окне **Резервное копирование базы данных** выберите **Журнал транзакций** в качестве значения параметра **Тип резервного копирования** и укажите, что должно выполняться резервное копирование базы данных.

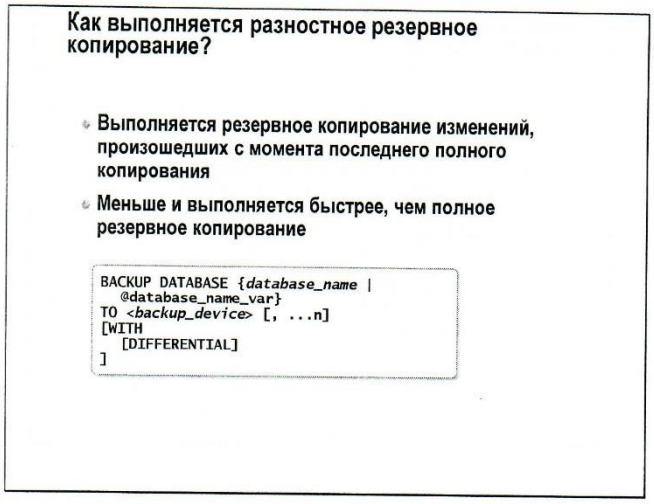
В альтернативном варианте резервное копирование журнала транзакций можно выполнить с помощью инструкции BACKUP LOG. Частичный синтаксис инструкции BACKUP LOG показан в следующем программном коде Transact-SQL.

BACKUP LOG {database_name | @database_name_var} TO <backup_device> [, ...n] [WITH [{INIT | NOINIT}]. Функция sys.dm_db_log_stats (DB_ID (DB1C)) возвращает в поле log_since_last_log_backup_mb размер журнала с момента последнего резервного копирования.

Выполнение резервного копирования заключительных фрагментов журнала

Если файлы данных, относящиеся к базе данных, недоступны, а файл журнала не поврежден, можно выполнить резервное копирование заключительных фрагментов журнала, чтобы записать действия, совершавшиеся в базе данных с момента последнего резервного копирования, и использовать их для восстановления базы данных до момента сбоя. Резервное копирование заключительных фрагментов журнала можно выполнить в SQL Server Management Studio из контекстного меню устройства резервного копирования или с помощью следующего образца кода Transact-SQL.

BACKUP LOG DB1C TO DISK = 'C:\Backup\DB1CTail.bak' WITH NORECOVERY, NO_TRUNCATE

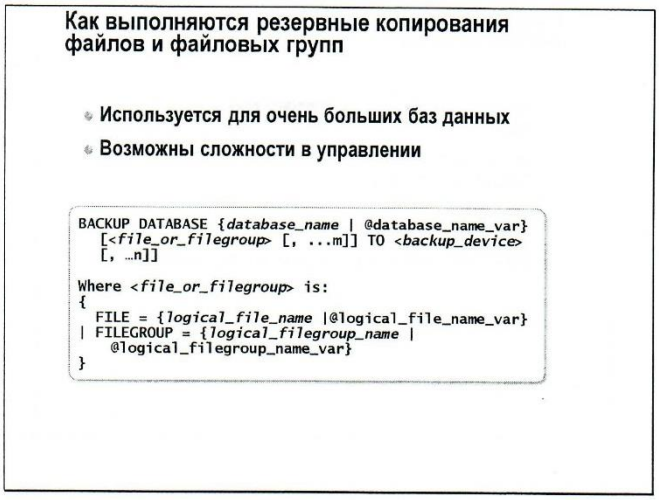


Резервная копия, на которой основана разностная резервная копия, называется базовой резервной копией. Базовая резервная копия для файла может содержаться в полной резервной копии или в резервной копии файла. При разностном резервном копировании записываются только данные, которые изменились с момента создания последней базовой резервной копии. Разностные резервные копии меньше по размеру и создаются быстрее базовых, что позволяет чаще создавать резервные копии, уменьшая риск потери данных.

Выполнение разностного резервного копирования

Чтобы выполнить разностное резервное копирование базы данных, в обозревателе объектов щелкните базу данных правой кнопкой мыши, в контекстном меню наведите указатель на пункт **Задачи** и выберите пункт **Резервное копирование**. В итоговом диалоговом окне **Резервное копирование базы данных** выберите **Разностное** в качестве значения параметра **Тип резервного копирования** и укажите, что должно выполняться резервное копирование базы данных. В альтернативном варианте разностное резервное копирование базы данных можно выполнить с помощью инструкции BACKUP DATABASE. Частичный синтаксис инструкции BACKUP DATABASE для разностного резервного копирования показан в следующем программном коде Transact-SQL.

BACKUP DATABASE {database_name | @database_name_var} TO <backup_device> [, ...n] [WITH [DIFFERENTIAL]]. В SQL Server 2017 представление sys.dm_db_file_space_usage возвращает в поле modified_extent_page_count количество экстентов, имеющих изменения после полного резервного копирования.

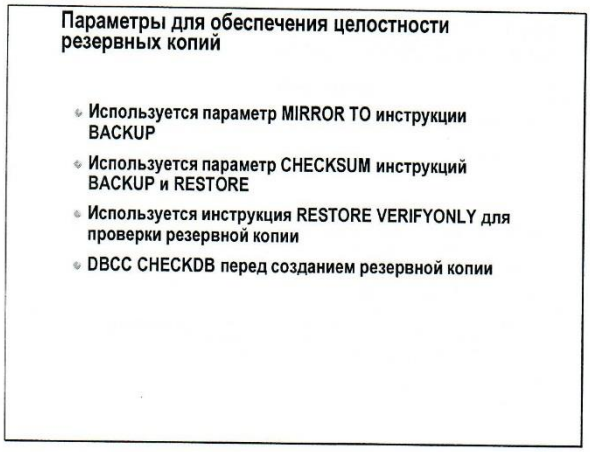


Файлы и файловые группы в базе данных могут подвергаться резервному копированию и восстановлению по отдельности. Использование этого типа резервного копирования позволяет повысить скорость восстановления за счет возможности восстанавливать только поврежденные файлы без восстановления остальной базы данных. Например, если база данных состоит из нескольких файлов, расположенных на разных дисках, и отказал один диск, потребуется восстановить только файл на отказавшем диске. В общем случае, указание файловой группы во время операций резервного копирования и восстановления эквивалентно перечислению всех файлов, содержащихся в файловой группе.

Выполнение резервных копирований файлов и файловых групп

Чтобы выполнить резервное копирование файла или файловой группы, в обозревателе объектов щелкните базу данных правой кнопкой мыши, в контекстном меню наведите указатель мыши на пункт **Задачи** и выберите пункт **Резервное копирование**. В итоговом диалоговом окне **Резервное копирование базы данных** выберите **Полное** или **Разностное** в качестве значения параметра **Тип резервного копирования**, укажите, что должно выполняться резервное копирование файлов и файловых групп, а затем выберите файлы и файловые группы, которые хотите включить в резервную копию. В альтернативном варианте разностное резервное копирование базы данных можно выполнить с помощью инструкции BACKUP DATABASE. Частичный синтаксис инструкции BACKUP DATABASE для резервного копирования файлов и файловых групп показан в следующем программном коде Transact-SQL.

BACKUP DATABASE {database_name | @database_name_var} {FILE = {logical_file_name | @logical_file_name_var}| FILEGROUP = {logical_filegroup_name | @logical_filegroup_name_var}} [, ...n] TO <backup_device> [, ...n] [WITH DIFFERENTIAL]



SQL Server позволяет зеркально отображать резервный носитель, уменьшая таким образом отрицательные эффекты сбоев устройства резервного копирования. Резервное копирование завершится сбоем, если какое-либо устройство в зеркальном наборе недоступно или отсутствует. Однако для успешного выполнения операций восстановления достаточно одного устройство в каждом зеркальном наборе. Во время резервного копирования при необходимости генерируются контрольные суммы, которые могут проверяться при восстановлении данных. Команда RESTORE VERIFYONLY расширена для включения сведений о контрольных суммах, используемых при анализе резервного набора данных.

Создание зеркальных резервных копий

Сервером SQL Server поддерживается создание зеркальных резервных носителей, повышающих надежность резервных копий за счет обеспечения избыточности данных. Все устройства резервного копирования для одиночной операции резервного копирования или восстановления должны быть одного типа — дисковыми или ленточными. В пределах этих более широких классов необходимо использовать сходные устройства, обладающие одинаковыми свойствами, например дисководы с одинаковыми номерами моделей, изготовленные одним и тем же производителем. При недостаточном сходстве устройств выводится сообщение об ошибке (3212). Эти резервные копии не поддерживаются средой SQL Server Management Studio.

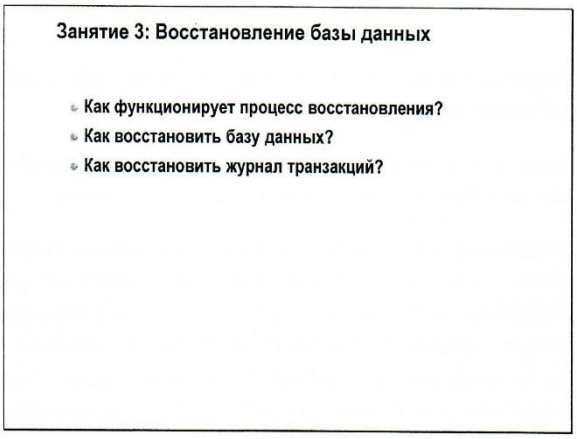
Контрольная сумма резервной копии

Важным механизмом обнаружения ошибок является используемая по желанию контрольная сумма резервной копии, которая может создаваться операцией резервного копирования и проверяться операцией восстановления. Имеется возможность управления поведением операции: будет ли операцией выполняться контроль ошибок, и будет ли операция останавливаться или продолжаться при возникновении ошибки. В среде SQL Server Management Studio можно задать создание контрольной суммы резервной копии с помощью параметра **Рассчитать контрольную сумму перед записью на носитель** в диалоговом окне **Резервное копирование базы данных**.

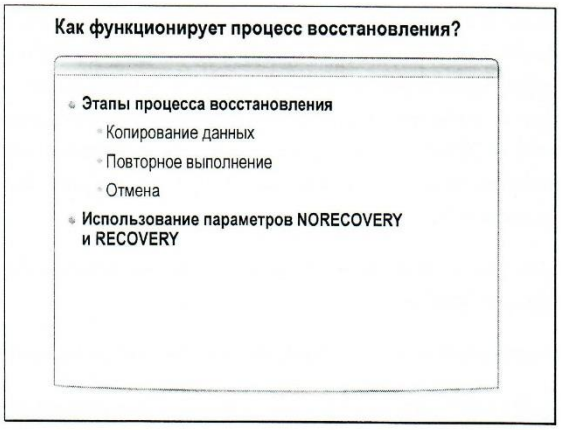
После выполнения резервного копирования можно воспользоваться инструкцией RESTORE VERIFYONLY, чтобы проверить резервную копию без ее восстановления. Инструкция RESTORE VERIFYONLY позволяет убедиться в полноте резервного набора данных и читаемости всей резервной копии. В SQL Server команда проверки целостности базы данных DBCC CHECKDB проверяет логическую и физическую целостность всех объектов в базе.

Задание 4. Выполнение резервного копирования базы данных

1. Запустите на своей машине **SQL Server Management Studio** и подключитесь к локальному экземпляру **SQL Server**, используя проверку подлинности Windows
2. Создайте постоянное устройство резервного копирования на файловой системе.
3. Выберите базу **DB1C**.
4. В контекстном меню базы выберите команду **Задачи->Создать резервную копию.**
5. Сделайте полную копию базы в файл, расположенный в папке, предлагаемой по умолчанию.
6. Выполните изменение или добавление данных в системе «1С:Предприятие 8»
7. Сделайте резервную копию журнала транзакций
8. При желании повторите пункты 6 и 7 несколько раз
9. Просмотрите отчет о событиях резервного копирования в базе данных.



На этом занятии предоставляются сведения и отрабатываются навыки, необходимые для восстановления базы данных и журналов транзакций. Рассматриваются процесс восстановления SQL Server и использование инструкции RESTORE для получения сведений и выполнения операций восстановления. Здесь также объясняется, как определять порядок выполнения операций восстановления на основе конкретного метода резервного копирования.



Восстановление базы данных — это процесс копирования данных из резервной копии и последующее применение к данным зарегистрированных в журнале транзакций с целью наката базы данных до целевой точки восстановления. Восстановление — это полный набор операций, с помощью которого обеспечивается согласованность базы данных и ее работоспособное состояние. Обычно в базе данных имеются незафиксированные транзакции в точке восстановления, а сама база данных находится в противоречивом, неработоспособном состоянии. В таких случаях восстановление включает откат незафиксированных транзакций. Набор всех восстанавливаемых данных называется набором *наката*. Набор наката определяется путем восстановления последовательности из одной или нескольких резервных копий данных (полных, частичных или файла). Если резервная копия данных содержит записи журнала, восстанавливаемые данные будут накатываться с использованием этих записей журнала.

Этапы восстановления

Восстановление является многоэтапным процессом. Возможные этапы восстановления включают *копирование данных, повторное выполнение (накат), отмену (откат)*:

• **Этап копирования данных**. Этап копирования данных включает копирование всех данных, журнала и страниц индекса с резервного носителя базы данных в файлы базы данных.

• **Этап повторного выполнения.** На этапе повторного выполнения зарегистрированные в журнале транзакции применяются к данным, скопированным из резервной копии, для наката этих данных до точки восстановления. В этой точке в базе данных обычно имеются незафиксированные транзакции, а сама база данных находится в противоречивом, неработоспособном состоянии, поэтому необходим этап отмены, являющийся частью процедуры восстановления базы данных. Чтобы выполнить накат, ядром СУБД обрабатываются резервные копии журнала по мере их восстановления, начиная с журнала, содержащегося в резервных копиях данных.

• **Этап отмены**. На этапе отмены выполняется откат всех незафиксированных транзакций, и база данных делается доступной для пользователей. После этапа отката последующие резервные копии не могут быть восстановлены. Затем в процессе восстановления база данных переводится в оперативный режим.

Использование параметров NORECOVERY, RECOVERY и STANDBY

Особенности процесса восстановления определяются использованием параметра RECOVERY или NORECOVERY в инструкции RESTORE. Следует всегда указывать параметр RECOVERY или NORECOVERY, чтобы предотвратить ошибки управления во время процесса восстановления и сделать инструкцию RESTORE более простой для понимания. Параметр RECOVERY используется сервером SQL Server по умолчанию. Когда параметр RECOVERY используется для восстановления последнего журнала транзакций или для полного восстановления базы данных с целью возврата базы данных в согласованное состояние, выполняются следующие действия:

• Сервером SQL Server выполняется откат всех не зафиксированных транзакций в журнале транзакций и накат всех зафиксированных транзакций.

• После завершения процесса восстановления база данных становится доступной для использования.

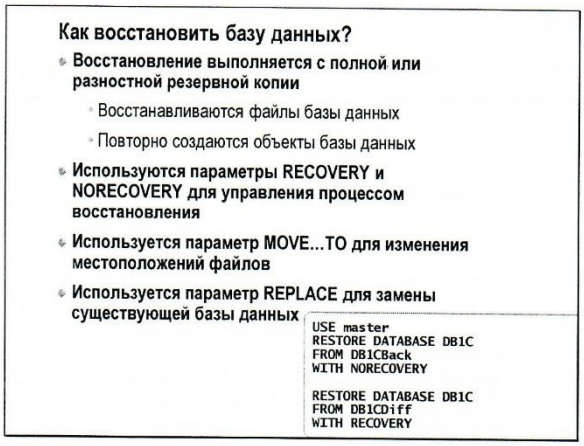
Примечание. Не используйте этот параметр, если имеются дополнительные журналы транзакций или разностные резервные копии, которые должны быть восстановлены. Параметр NORECOVERY используется, когда имеется несколько восстанавливаемых резервных копий. Параметр NORECOVERY

следует указывать для всех резервных копий за исключением последней восстанавливаемой резервной копии. При использовании параметра NORECOVERY примите во внимание следующие основные соображения:

• Сервером SQL Server не выполняется откат каких-либо не зафиксированных транзакций в журнале транзакций, а также не выполняется накат каких-либо зафиксированных транзакций.

• Пока база данных не будет восстановлена, она недоступна для использования.

Параметр STANDBY не запрещает дальнейшее восстановление и позволяет использовать чтение данных.



Когда база данных восстанавливается из ее резервной копии, сервером SQL Server воссоздаются база данных и все связанные с ней файлы, которые затем помещаются в их исходное местоположение. Все объекты базы данных воссоздаются автоматически.

Обычно восстановление из полной резервной копии базы данных выполняется в следующих случаях: поврежден физический диск базы данных; повреждена, испорчена или удалена вся база данных; идентичная копия базы данных восстанавливается на другой экземпляр SQL Server.

Параметры восстановления базы данных

Существует ряд параметров, которые можно указывать при восстановлении базы данных, включая RECOVERY или NORECOVERY, MOVE TO и REPLACE:

• **RECOVERY** и **NORECOVERY**. Параметром RECOVERY запускается процесс восстановления, в результате выполнения которого база данных возвращается в согласованное состояние. Указывайте параметр RECOVERY, если применяется стратегия полного резервного копирования базы данных, и отсутствуют резервные копии журналов транзакций и разностные резервные копии. Если существуют какие-либо резервные копии журнала транзакций или разностные резервные копии, указывайте параметр NORECOVERY, чтобы отложить процесс восстановления до тех пор, пока не будет восстановлена последняя резервная копия.

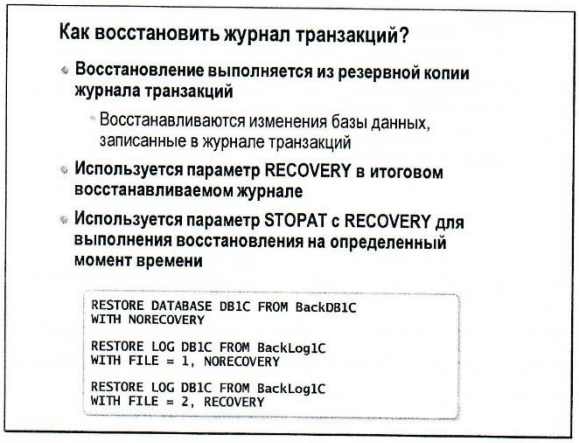
• **MOVE TO**. Используйте параметр MOVE TO, чтобы указать, куда следует восстанавливать файлы резервной копии, если планируется восстанавливать файлы в другое место, например на другой диск, сервер или резервный сервер.

• **REPLACE**. Используйте параметр REPLACE только в том случае, если хотите заменить существующую базу данных данными из резервной копии другой базы данных. Если используется параметр REPLACE, сервером SQL Server не выполняется проверка безопасности. По умолчанию сервером SQL Server выполняется проверка безопасности, гарантирующая, что существующая база данных не заменена, если справедливо одно из следующих условий:

• База данных уже существует на целевом сервере, и имя базы данных отличается от имени, которое записано в резервном наборе данных.

• Набор файлов в базе данных отличается от файлов, содержащихся в резервном наборе данных. Сервером SQL Server игнорируются различия в размерах файлов.

Сервером SQL Server ведется журнал резервного копирования для всех баз данных, автоматически определяются самые последние резервные копии и надлежащий порядок, в котором выполняется их восстановление.



Когда восстанавливают журнал транзакций, изменения в журнале применяются сервером SQL Server к базе данных. Обычно журналы транзакций восстанавливают для применения изменений, которые были внесены в базу данных с момента создания последней полной резервной копии или разностной резервной копии базы данных. Кроме того, можно восстановить журналы транзакций для воссоздания базы данных, существовавшей на определенный момент времени.

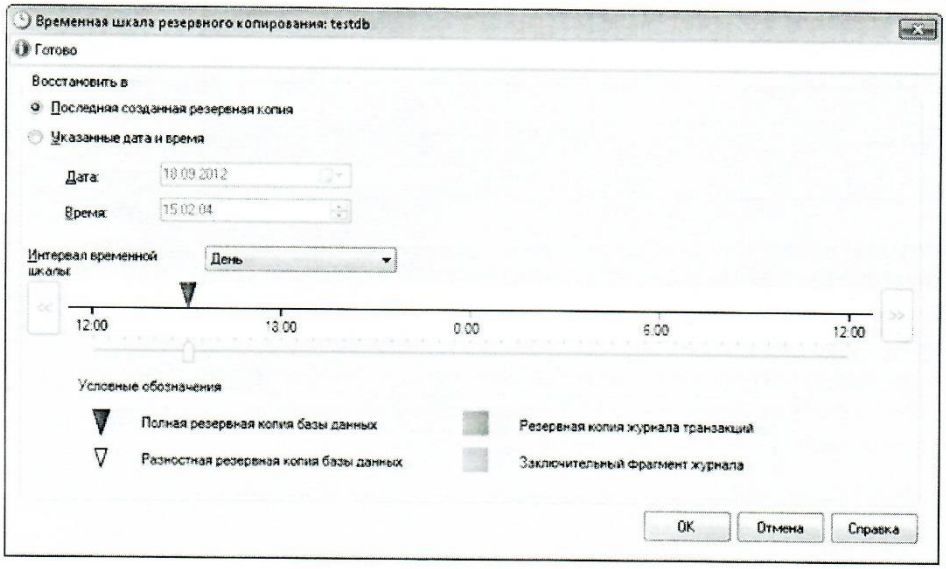
Обсуждение процесса восстановления журналов транзакций

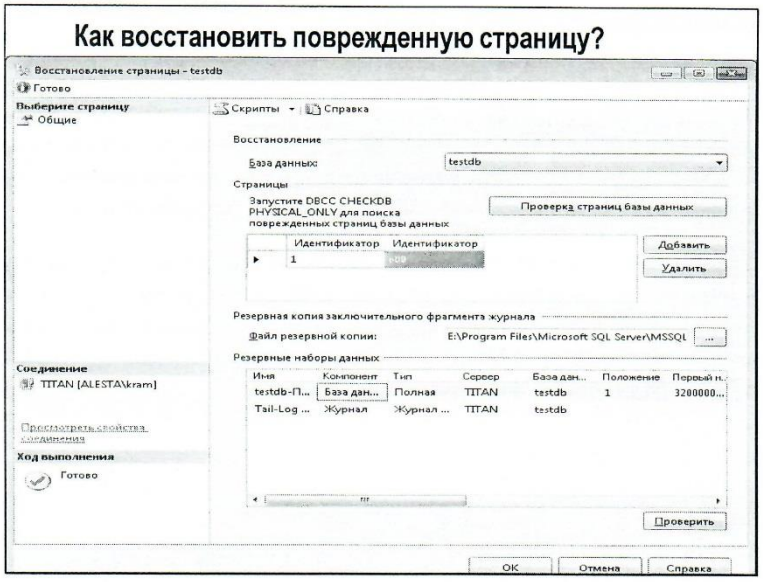
Хотя восстановление разностной резервной копии может ускорить процесс восстановления, для обеспечения согласованности данных, возможно, потребуется восстановить дополнительные резервные копии журналов транзакций, которые были созданы после разностной резервной копии. Прежде чем восстанавливать какие-либо журналы транзакций, следует сначала восстановить полную резервную копию базы данных, а затем самую последнюю разностную резервную копию, если такая существует. Далее необходимо восстановить по порядку все резервные копии журналов транзакций, записанные с момента создания последней полной или разностной резервной копии. Когда для применения имеется несколько журналов транзакций, укажите параметр NORECOVERY для всех журналов транзакций за исключением последнего.

Восстановление на определенный момент времени

Инструкции RESTORE LOG и RESTORE DATABASE позволяют также восстанавливать базу данных в состояние, в котором она находилась на определенный момент времени. Для всех транзакций, которые не были зафиксированы на этот момент времени, будет выполнен откат, а все транзакции, выполнявшиеся после этого момента времени, не будут применяться к базе данных. Для этого можно использовать новые графические возможности среды SQL Server Management Studio.

Важно! Все транзакции, выполнявшиеся после этого времени и записанные в журнал транзакций, отбрасываются, и все резервные копии журналов транзакций, созданные после этого момента времени, становятся неработоспособными; не пытайтесь применять их к базе данных. Новые резервные копии содержат изменения, внесенные после момента времени, указанного в операторе STOPAT.





SQL Server 2017 Enterprise Edition позволяет восстанавливать одиночные страницы, отдельные файлы и полные файловые группы, в то время как остальная часть базы данных находится в оперативном режиме. В SQL Server поврежденные страницы обнаруживаются автоматически при попытке прочитать данные, обычно как результат выполнения пользователем или приложением запроса Transact-SQL. Типовое обнаруживаемое повреждение включает оборванную страницу или страницу с неверной контрольной суммой, страницу с недостоверными данными заголовка, такими как неправильный идентификатор страницы, или с непредвиденно усеченными данными.

Как в SQL Server обрабатываются поврежденные страницы?

Когда встречается поврежденная страница, сервером SQL Server выполняется откат текущей транзакции и выводится сообщение об ошибке (823 или 824), а сама база данных остается в оперативном режиме, и другие пользователи могут продолжать работать с ней. Однако, если сервером SQL Server обнаруживается ошибка страницы во время отката транзакции, база данных переводится в автономный режим и ее потребуется восстановить. Когда база данных возвращается в оперативный режим, сервером SQL Server повторно получаются блокировки, установленные ранее на страницах (даже на дефектных), в качестве части процесса восстановления, и выполняется откат транзакций, для которых сервером может быть осуществлен откат. Транзакции, которые невозможно откатить, помещаются в режим DEFERRED, и они не откатываются, и с них не снимаются блокировки до тех пор, пока не устранят повреждения. Сервером SQL Server записываются в журнал ошибок SQL Server все случаи доступа к поврежденным страницам и сохраняются дополнительные сведения в таблице suspect_pages базы данных msdb. Используя эти сведения, можно идентифицировать поврежденные страницы, восстановить их и исправить данные, в то время как база данных будет находиться в оперативном режиме. Во время восстановления недоступен только файл, содержащий поврежденную страницу; все другие файлы в файловой группе доступны, если только файл не является частью первичной файловой группы, в последнем случае база данных переводится в автономный режим.

Примечание. Оперативное восстановление страницы можно выполнить только в том случае, если для базы данных используется модель полного восстановления или модель восстановления с неполным

протоколированием. Базы данных, для которых используется простая модель восстановления, должны восстанавливаться в автономном режиме.

Восстановление страницы

Чтобы восстановить отдельную страницу из резервной копии, используйте новые возможности графического интерфейса. Определите идентификатор страницы, которую хотите восстановить. В зависимости от типа ошибки существует несколько источников, которые можно использовать для получения этих сведений.

• В таблицу **suspect_pages** базы данных **msdb** записываются сведения о страницах, которые привели к ошибкам ввода-вывода, являются оборванными или имеют контрольную сумму, не соответствующую базе данных. Однако существует предел на размер этой таблицы, в ней может содержаться максимум 1000 строк. Эта таблица действует как журнал поврежденных страниц, и очистка данной таблицы входит в обязанности администратора базы данных. Если эта таблица заполнена, новые записи не добавляются.

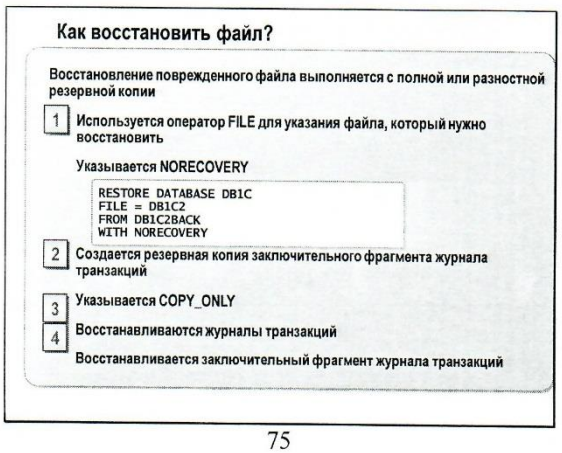
• Всякий раз, когда обращаются к поврежденным страницам, их идентификаторы записываются в журнал ошибок SQL Server.

• Событие ErrorLog в трассировке событий, создаваемой приложением SQL Server Profiler.

• Другие источники включают команды DBCC, такие как DBCC CHECKTABLE, а также поставщика инструментальных средств управления средой Windows (WMI) для SQL Server.

В файле на поврежденную страницу указывает номер файла и номер страницы. Какой файл соответствует заданному номеру файла можно определить, запросив системное представление **sys.database_files** в базе данных, содержащей поврежденную страницу. Восстановите поврежденную страницу из резервной копии. Восстановите страницу, используя самую последнюю полную или разностную резервную копию, которая содержит поврежденную страницу. Используйте команду RESTORE DATABASE с оператором PAGE, чтобы указать идентификатор(ы) страницы (или страниц), которую (которые) необходимо восстановить. С помощью этой команды можно восстановить до 1000 отдельных страниц

В контекстном меню базы данных выберите команду Восстановить-> Страницу. Введите номер файла и номер страницы и выберите резервную копию. После нажатия кнопки OK страница восстановится.



С помощью команды RESTORE DATABASE можно восстановить до 1000 отдельных страниц. Однако если в одном файле имеется более пяти поврежденных страниц, следует рассмотреть возможность восстановления всего файла.

Восстановление файла

Чтобы восстановить отдельный файл, выполните следующие действия:

1. Восстановите поврежденный файл из самой последней резервной копии этого файла. Используйте команду RESTORE DATABASE с оператором FILE, указывающим имя восстанавливаемого файла. Задайте параметр NORECOVERY.

```
USE master

RESTORE DATABASE DB1C FILE = DB1C2 FROM DB1C2Back WITH NORECOVERY
```

2. Выполните резервное копирование заключительного фрагмента журнала транзакций. Используйте команду BACKUP LOG с параметром COPY_ONLY.

```
BACKUP LOG DB1C TO DISK = 'C:\Backups\TempLogBackup.bak' WITH COPY_ONLY
```

Эту резервную копию следует использовать только для восстановления после завершения операции оперативного восстановления, и затем следует отказаться от ее использования.

3. Чтобы перевести базу данных в согласованное состояние, восстановите резервные копии журналов транзакций, которые были записаны после создания резервной копии файла. Используйте команду RESTORE LOG. Один за другим восстановите все файлы журнала транзакций и задайте параметр NORECOVERY.

4. Восстановите версию COPY_ONLY резервной копии заключительного фрагмента журнала транзакций, которая была создана на шаге 3. Используйте параметр RECOVERY команды RESTORE LOG. После завершения данной операции отбросьте копию COPY_ONLY журнала транзакций.

Определение

Моментальный снимок базы данных — это доступное только для чтения статическое представление базы данных в определенный момент времени, которое не изменяется после создания моментального снимка. База данных, для которой делается моментальный снимок, называется базой данных-источником. Моментальные снимки баз данных могут быть полезны в качестве точки быстрого восстановления при случайном или злонамеренном повреждении информации в базе данных. Однако их нельзя использовать в качестве замены резервных копий, так как моментальный снимок базы данных не содержит всех записей базы данных. Поддержка моментальных снимков есть во всех выпусках SQL Server 2017.

Ограничения на создание моментальных снимков

Одно из ограничений моментальных снимков баз данных заключается в том, что моментальный снимок должен находиться на том же сервере, что и база данных-источник. В отношении моментальных снимков баз данных действуют также следующие ограничения:

• Моментальные снимки не могут создаваться для баз данных **model**, **master** и **tempdb**.

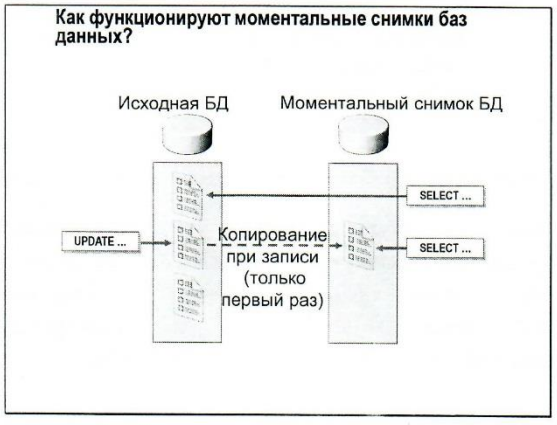
•Для моментальных снимков баз данных не могут быть выполнены резервное копирование и восстановление.

• Моментальные снимки невозможно присоединить или отсоединить.

• Моментальные снимки невозможно создать на разделах FAT32 и на неформатированных разделах.

• Прежде чем удалить саму базу данных, необходимо удалить все моментальные снимки, созданные для базы данных.

• Средой SQL Server Management Studio не предоставляется графический пользовательский интерфейс для создания моментальных снимков. Поэтому моментальные снимки баз данных могут быть созданы только с помощью Transact-SQL.



Когда происходят обновления базы данных-источника, моментальные снимки базы данных сохраняют статическое представление базы данных-источника путем хранения копий данных до их изменения.

Позже эти скопированные сведения возвращаются, когда запрашиваются в виде части обычного запроса.

Создание моментального снимка базы данных

Для создания моментального снимка базы данных используется оператор AS SNAPSHOT OF инструкции CREATE DATABASE.

Извлечение данных из моментального снимка базы данных

В SQL Server для реализации моментальных снимков баз данных без затраты вычислительных ресурсов на создание полной копии базы данных используется технология «копирование при записи». Моментальный снимок базы данных первоначально пуст и физически реализуется в виде разреженных файлов NTFS, являющихся файлами, для которых пространство на физическом диске выделяется только по запросу. Когда страница в базе данных-источнике обновляется в первый раз, исходное изображение этой страницы копируется в моментальный снимок базы данных. Если страница никогда не изменяется, она никогда не копируется. Если страница данных в базе данных-источнике не изменялась с момента создания моментального снимка базы данных, запросы, запрашивающие данные из моментального снимка базы данных, извлекают данные с исходной страницы данных в базе данных-источнике. Если в базе данных-источнике обновляется какая-либо строка на странице, вся страница сначала копируется в файл данных моментального снимка базы данных, и последующие запросы данных на этой странице используют скопированный файл в моментальном снимке базы данных.

Моментальный снимок базы данных можно использовать для восстановления случайно измененной базы данных. С этой целью данные из моментального снимка применяются к базе данных-источнику. Однако следует осознавать, что моментальный снимок базы данных представляет собой весьма упрощенный механизм восстановления, который не может служить заменой реализации всеобъемлющей стратегии резервного копирования и восстановления.

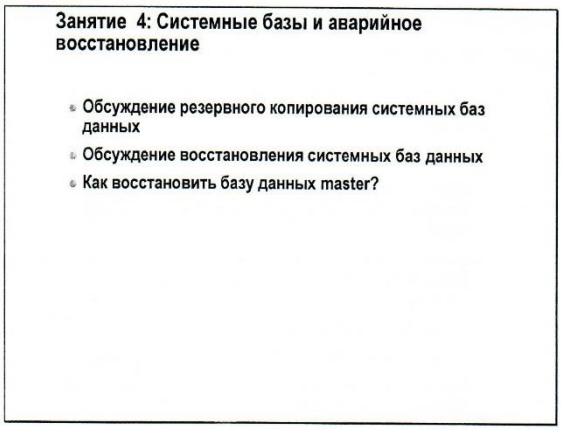
Применимые сценарии

Существуют различные причины потери данных, начиная от случайного удаления таблицы или изменения одиночной строки до повреждения или утраты файла базы данных. Природа моментального снимка базы данных делает его идеальным средством исправления ошибок приложений и

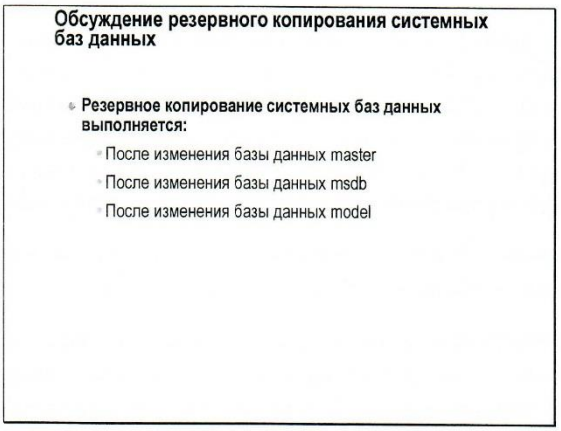
пользователей, которые приводят к случайному удалению или обновлению строк либо к удалению таблиц. Восстановление данных из моментального снимка базы данных происходит быстрее и проще, чем выполнение операции восстановления из резервной копии базы данных. Однако механизм «копирование при записи» не позволяет использовать моментальные снимки базы данных для восстановления подозрительной базы данных, содержащей испорченные файлы — в этом сценарии требуемые файлы необходимо восстановить из резервной копии базы данных. Для системы «1С: Предприятие 8» всегда применим сценарий возвращения всей базы к моментальному снимку. Моментальный снимок базы данных можно использовать для формирования файла выгрузки.

Возвращение к моментальному снимку

Если в базе данных значительный объем данных случайно или умышленно потерян, базу данных можно восстановить из самого последнего моментального снимка и вернуть ее в состояние, в котором она находилась на момент, когда был создан моментальный снимок.



Следует регулярно выполнять резервное копирование системных баз данных и в особенности после каждого их изменения. На этом занятии объясняется, когда следует выполнять резервное копирование системных баз данных на сервере, включая главную базу данных, а также рассматривается порядок восстановления системных баз данных.

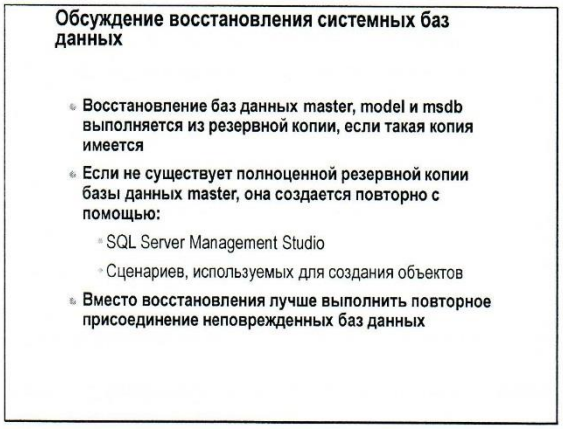


В главной базе данных содержатся сведения обо всех базах данных на SQL Server. Выполняйте резервное копирование главной базы данных всякий раз, когда создаются, изменяются или удаляются любые пользовательские базы данных.

Осуществляйте резервное копирование главной базы данных всякий раз, когда выполняете: инструкцию CREATE DATABASE, ALTER DATABASE или DROP DATABASE, с помощью которой создается, изменяется или удаляется база данных; либо хранимые процедуры **sp_addserver, sp_dropserver** и **sp_addlinkedserver**, с помощью которых добавляются и удаляются серверы; либо системную хранимую процедуру **sp_addmessage**, которая используется для добавления в SQL Server специальных сообщений об ошибках.

Выполняйте резервное копирование базы данных **msdb** после ее изменения, поскольку в базе данных **msdb** содержатся сведения о заданиях, предупреждениях и операторах, которые используются службой агента SQL Server.

Если базу данных **model** изменили, выполняйте ее резервное копирование, чтобы включить конфигурацию по умолчанию для всех новых пользовательских баз данных.



Когда следует восстанавливать системные базы данных?

Решение о восстановлении или создании заново системных баз данных должно приниматься в зависимости от ответа на вопрос, можно ли запустить службу SQL Server. Если службу можно запустить, следует восстановить системные базы данных из самой последней резервной копии, чтобы как можно меньше данных было потеряно. Если службу невозможно запустить, необходимо заново создать системные базы данных, добавляя утерянные сведения путем присоединения существующих таблиц и повторного создания объектов. После того, как системные базы данных созданы заново и служба SQL Server запускается, следует восстановить системные базы данных в следующем порядке:

1. Восстановите из резервной копии главную базу данных. Если полноценная резервная копия главной базы данных не существует, необходимо вручную заново создать данные.

2. Восстановите базу данных msdb из резервной копии. Базу данных msdb необходимо восстановить, когда заново создается главная база данных. Когда главная база данных создается вновь, база данных msdb удаляется и затем создается повторно. Поэтому все сведения утрачиваются. Базу msdb можно восстановить, если остановлена служба агента.

3. Восстановите из резервной копии базу данных model, запустив службу экземпляра с флагом -T3608.

Когда следует восстанавливать пользовательские базы данных?

Пользовательские базы данных присоединяют или восстанавливают в зависимости от того, была ли восстановлена главная база данных из резервной копии:

• Если главная база данных была восстановлена из полноценной резервной копии, в ней будут содержаться ссылки на все пользовательские базы данных. Никаких дополнительных действий выполнять не требуется.

• Если master база данных была создана заново, и полноценная резервная копия не применялась, необходимо восстановить пользовательские базы данных из резервной копии или присоединить файлы существующих пользовательских баз данных к новой master базе данных. Если файлы пользовательских баз данных не повреждены, присоедините их к новой master базе данных с помощью команды CREATE DATABASE с оператором FOR ATTACH. При присоединении файлов существующей базы данных сведения о пользовательской базе данных добавляются в master базу данных. Для присоединения базы данных к master базе данных резервная копия этой базы данных не требуется.

Примечание. Присоединение пользовательской базы данных более эффективно, чем восстановление из резервной копии.

Восстановление базы master

- **Если доступен экземпляр SQL Server**
  1. Запустить SQL Server в однопользовательском режиме
  2. Восстановить последнюю резервную копию базы данных master из SQLCMD
  3. Перезапустить сервер
- **Если экземпляр SQL Server недоступен**
  1. Перестроить базу данных master с помощью
  2. Setup.exe /ACTION= REBUILDDATABASE /INSTANCENAME=instance_name /SQLSYSADMINACCOUNTS=accounts [/SAPWD=password] [/SQLCOLLATION=collation_name]
  3. Восстановить базу данных master
  4. Восстановить базы данных msdb и model

Восстановление главной базы данных, когда служба SQL Server доступна

Если база данных **master** все еще доступна, можно будет запустить экземпляр SQL Server. В этом сценарии следует запустить SQL Server в однопользовательском режиме и затем восстановить копию master базы данных из самой последней полной резервной копии базы данных обычным способом, как описано ниже.

1. Запустите SQL Server в однопользовательском режиме администрирования. В командной строке перейдите в папку для установки SQL Server, а затем введите следующую команду **sqlservr.exe -c –m**

2. Восстановите базу данных master из самой последней резервной копии, запустив sqlcmd и выполнив команду RESTORE DATABASE master FROM masterbackup

Если в **master** базу данных вносились какие-либо изменения с момента ее последнего резервного копирования, необходимо вручную повторно применить эти изменения, после того как база данных будет восстановлена и переведена в оперативный режим. Когда процесс восстановления завершен, служба SQL Server автоматически останавливается. На этом этапе или можно запустить SQL Server в однопользовательском режиме администрирования, чтобы внести изменения вручную до переключения базы данных в оперативный режим, или можно запустить SQL Server для непосредственного использования клиентом.

Восстановление главной базы данных, когда служба SQL Server недоступна

Если **master** база данных серьезно повреждена, возможно, не удастся запустить экземпляр SQL Server. В этой ситуации следует создать заново полностью новую версию **master** базы данных. Чтобы создать заново **master** базу данных, следует запустить программу установки SQL Server со следующими параметрами:

Setup.exe /ACTION= REBUILDDATABASE /INSTANCENAME=instance_name
/SQLSYSADMINACCOUNTS=accounts [/SAPWD=password] [/SQLCOLLATION=collation_name]

Когда процесс повторного создания завершен, то можно восстановить исходную версию на сервер, выполнив предыдущие шаги. Перепостроение системных баз данных включает перепостроение баз данных **msdb** и **model**, поэтому необходимо убедиться, что имеются резервные копии требуемых версий для выполнения процедуры их восстановления.

Задание 5. Выполнение восстановления базы данных

1. Имитируйте поломку, переименовав файл данных и журнал транзакций.
    - Остановите SQL Server в диспетчере конфигурации
    - Откройте папку **C:\Program Files\Microsoft SQL Server\MSSQL14.MSSQLSERVER\MSSQL\DATA**
    - Переименуйте **DB1C.mdf** в **DB1C_old.mdf.**
    - Переименуйте файл **DB1C_log.ldf** в **DB1C_log_old.ldf.**
    - Запустите SQL Server.
2. Восстановите базу данных **DB1C** сервера «1С: Предприятие 8» из резервной копии. Для восстановления нужной копии используйте временную шкалу восстановления с графическим интерфейсом.

Создание и использование моментального снимка базы данных

1. Создайте базу **DB1CSnapshot** с моментальным снимком базы **DB1C**, используя следующий текст запроса:
   CREATE DATABASE DB1Csnapshot ON (NAME= 'DB1C', FILENAME='C:\Program Files\Microsoft SQL Server\MSSQL14.MSSQLSERVER\MSSQL\DATA\DB1C.ss') AS SNAPSHOT OF DB1C

2. Выполните транзакции в 1С
3. Закройте все соединения с базой **DB1C**
4. Сделайте возврат к моментальному снимку для DB**1C** с помощью команды
   RESTORE DATABASE **DB1C** FROM DATABASE_SNAPSHOT = **'DB1Csnapshot'**

5. Зайдите в «1С: Предприятие 8» и проверьте результат

Раздел 4: Управление безопасностью данных

Обеспечение безопасности является основной задачей при разработке и управлении средой баз данных. В этом разделе вы узнаете о модели безопасности в Microsoft SQL Server 2017 и ее использовании при поддержке системы «1С:Предприятие 8»

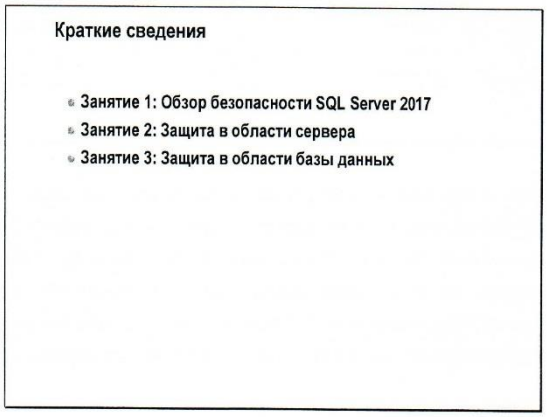
Цели

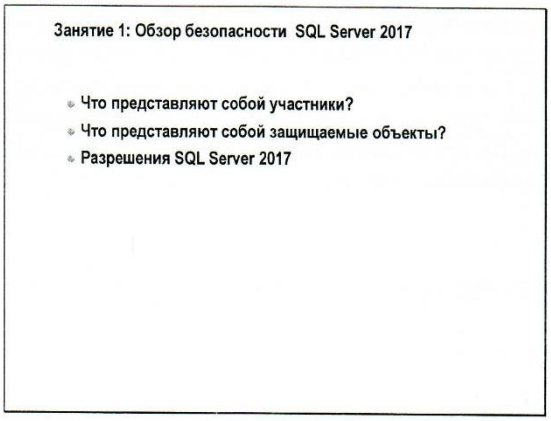
После изучения данного раздела вы сможете:

• объяснить, как в SQL Server осуществляется управление безопасностью;

• обеспечить защиту SQL Server на уровне сервера;

• защитить базы данных SQL Server





В этом занятии содержится описание основных компонентов безопасности SQL Server: участников и защищаемых объектов. В нем также описываются разрешения, которые могут применяться к любым защищаемым объектам в модели безопасности SQL Server.

В SQL Server понятие *участник* используется по отношению к пользователям, которые прошли проверку подлинности в системе SQL Server. Участником является любой пользователь, прошедший проверку подлинности, которому может быть предоставлено разрешение на доступ к объекту в системе базы данных. В SQL Server существуют отличия между *неделимыми участниками*, которые являются отдельными удостоверениями (например, имена входа) и *коллективными участниками*, которые являются коллекциями удостоверений (например, фиксированные серверные роли).

Уровни участников

Участники существуют на трех уровнях: Microsoft Windows, SQL Server и база данных. Типы участников, допустимые на каждом из этих уровней, показаны на рисунке. Объекты, доступ к которым регулируется в системе авторизации SQL Server, называются *защищаемыми объектами*. Защищаемые объекты организованы во вложенные иерархии, которые называются областями и тоже могут быть защищены. Три защищаемые области в SQL Server: *сервер, база данных* и *схема*. Защищаемые объекты на уровне Windows включают файлы и разделы реестра.

Область сервера

К защищаемым объектам в области сервера относятся:

- Имена входа
- Конечные точки
- Базы данных

Область базы данных

К защищаемым объектам в области базы данных относятся:

- Пользователи
- Роли
- Роли приложений
- Сертификаты

- Симметричные ключи
- Асимметричные ключи
- Сборки
- Полнотекстовые каталоги
- События DDL
- Схемы

SQL Server использует разрешения для управления доступом участников к защищенным объектам. Разрешения — это правила, которые управляют уровнем доступа участников к защищаемым объектам. SQL Server определяет собственные наборы разрешений, которые можно применить по отношению к защищаемым объектам SQL Server.

Примеры разрешений

Единственным способом доступа участника к ресурсу в системе SQL Server является предоставление ему разрешения на доступ, напрямую или посредством членства вторичного участника, например роли. Управлять разрешениями можно, используя обозреватель объектов в SQL Server Management Studio или выполнив инструкции GRANT, REVOKE или DENY. Определенные разрешения, связанные с отдельными защищаемыми объектами, отличаются в зависимости от типов действий, поддерживаемых защищаемыми объектами.

Наследуемые разрешения

Определенные разрешения в SQL Server могут быть унаследованы через разрешения, предоставленные на более высоком уровне в иерархии защищаемой области. Например:

• участник, получивший разрешение SELECT для схемы, автоматически наследует разрешение SELECT для всех объектов схемы.

• Участник, получивший разрешение CONTROL для объекта базы данных, автоматически наследует разрешение CONTROL для всех защищаемых объектов, содержащихся в этой базе данных, и всех защищаемых объектов, содержащихся в схемах базы данных.

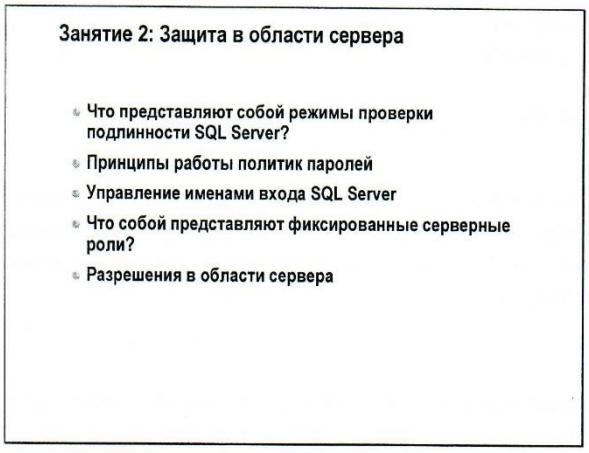
Действующие разрешения

Действующие разрешения для участника оцениваются тем же способом, что и в предыдущих выпусках SQL Server. Участник может выполнить определенное действие, если выполняются оба условия:

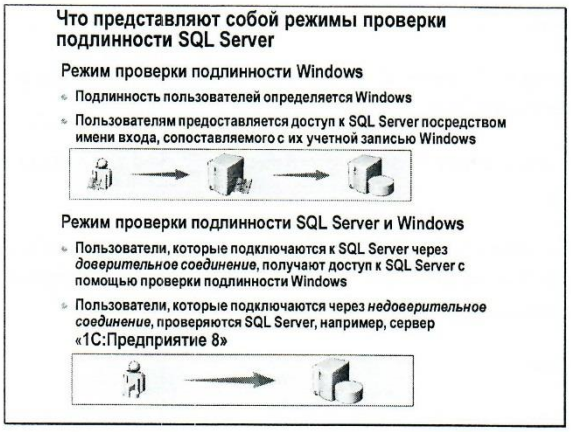
• Разрешение было предоставлено явным образом участнику или коллекции, в которую входит участник.

• Разрешение не было запрещено явным образом участнику или коллекции, в которую входит участник.

Примечание. Явная инструкция DENY всегда имеет преимущество перед инструкцией GRANT. Например, если пользователю было явным образом предоставлено разрешение SELECT для определенной таблицы, но он является участником роли, которой было явным образом запрещено разрешение SELECT для доступа к таблице, пользователь не сможет выполнить инструкцию SELECT по отношению к таблице.



На этом занятии рассматривается проверка безопасности на уровне объекта сервера в SQL Server 2016 Важно понимать, каким образом выполняется защита области сервера, чтобы снизить возможность доступа неавторизованных пользователей к экземплярам SQL Server.



SQL Server можно настроить для использования одного из двух следующих режимов проверки подлинности:

• **Режим проверки подлинности Windows**. Пользователи определяются Windows, им предоставляется доступ к SQL Server посредством имени входа, сопоставляемого с их учетной записью Windows (или группой Winows, участниками которой они являются). При первом запросе связи маркер доступа пользователя, созданный при попытке входа пользователя в Windows, предоставляется для SQL Server.

• **Режим проверки подлинности Windows и SQL Server**. Пользователи, которые подключаются к SQL Server через *доверительное* соединение, получают доступ к SQL Server с помощью проверки подлинности Windows. Кроме того, SQL Server поддерживает имена входа, которые не сопоставляются с пользователями Windows, и проверяются SQL Server отдельно от Windows. Режим проверки подлинности Windows и SQL Server иногда называют *смешанным* режимом.

Когда следует использовать режим проверки подлинности Windows и SQL Server

Режим проверки подлинности SQL Server и Windows устанавливается для поддержки системы «1С:Предприятие 8». Для подключения сервера приложения используются имена входа, которые не зависят от имен входа Windows. Подключение с именами Windows при этом остается возможным и может использоваться для административных задач.

Использование Active Directory аутентификации с SQL Server в Linux

Для такой аутентификации необходимо включить узел Linux в домен Active Directory с помощью команды **realmd**, отредактировать файл **/etc/network/interfaces**, создать в Active Directory пользователя и SPN для SQL Server.

Примечание. На SQL Server по умолчанию отсутствует имя входа для подключения группы локальных администраторов. Ей может быть предоставлен доступ во время установки сервера. Но при запуске экземпляра SQL Server с параметром –m доступ для группы локальных администраторов открывается. После параметра –m можно указать имя единственного приложения для подключения. Ниже даны примеры запуска.

sqlservr.exe -m"SQLCMD"

net start mssqlserver -m"SQLCMD"

Для подключения сервера системы «1С:Предприятие 8» не рекомендуется использовать имя входа sa. Для имени входа sa установите соответствующий политике безопасности пароль. Можно также сделать имя входа sa неактивным и не использовать его.

В Windows Server можно использовать групповую политику, определяя конфигурации компьютеров и пользователей для групп компьютеров и пользователей. Групповую политику можно использовать для настройки множества параметров, в том числе политик учетных записей. Политики паролей можно использовать для обеспечения достаточного уровня сложности паролей, а также для регулярного изменения паролей с целью обеспечения максимального уровня безопасности и предупреждения доступа неавторизованных пользователей. Политики паролей можно применять и для имен входа SQL, если SQL Server установлен на компьютер, работающий под управлением Windows Server.

Политики сложности паролей

Политики сложности паролей разработаны для предупреждения атак перебора паролей путем увеличения количества возможных паролей. Если включена политика сложности паролей, новые пароли должны соответствовать требованиям политики, установленным политикой паролей Windows. Пример такой политики:

• Пароль не должен содержать все имя или часть имени учетной записи пользователя. Часть учетного имени определяется как три и более последовательных буквенно-цифровых символа, отделенных пробелами с каждой стороны (пробел, табуляция, возврат и т. д.) или любым из следующих символов: , . - _ #

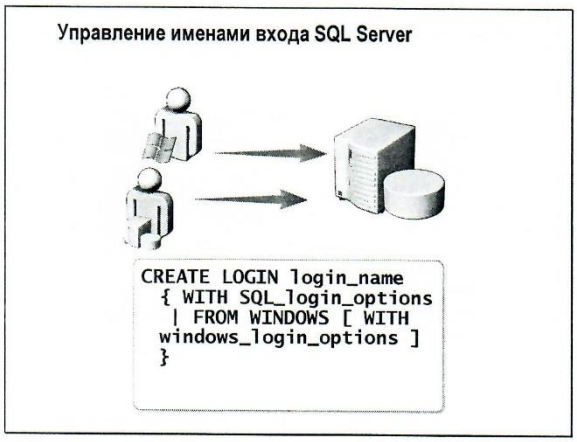
• Длина пароля не должна быть менее семи символов.

• Пароль содержит символы трех из четырех следующих категорий:

- латинские буквы верхнего регистра (буквы от A до Z)
- латинские буквы нижнего регистра (буквы от a до z)
- цифры от 0 до 9
- не буквенно-цифровые символы, например, !, $, # или %

Политики окончания срока действия паролей

Политики окончания срока действия паролей используются для управления сроком действия паролей. При использовании политики окончания срока действия пароля пользователи получают напоминания о необходимости изменить старые пароли, а учетные записи с истекшим сроком действия паролей деактивируются.



Управлять именами входа можно с помощью обозревателя объектов в SQL Server Management Studio, или выполнив инструкции Transact-SQL CREATE LOGIN, ALTER LOGIN и DROP LOGIN.

Создание имен входа

Можно использовать инструкцию CREATE LOGIN, чтобы создать имена входа Windows или SQL Server. Определенные параметры, которые можно использовать в инструкции CREATE LOGIN, зависят от того, создается имя входа как имя входа Windows или SQL Server.

В следующем примере продемонстрирована инструкция CREATE LOGIN, с помощью которой создается имя входа Windows для локальной группы Windows DB1CAdmin.

```
CREATE LOGIN [SERVERX\DB1CAdmin]

FROM WINDOWS WITH DEFAULT_DATABASE = DB1C
```

Если политика пароля включена для сервера, SQL Server активизирует ее для новых имен входа SQL по умолчанию, но эти действия можно изменить, используя параметры в инструкции CREATE LOGIN, отображенные в следующей таблице.

Параметр	**Описание**
HASHED.	Указывает, что пароль уже хеширован Если это не указано, строка будет сначала хеширована и только затем сохранена.
MUST_CHANGE	Требует изменения пароля для имени входа при первом подключении. Для поддержки сервера предприятия 1С не используется. Если этот параметр указан, для параметров CHECK_EXPIRATION и CHECK_POLICY следует задать значение ON.
CHECK_EXPIRATION	Если для этого параметра задано значение ON (по умолчанию), этот параметр указывает, что политика окончания срока действия Windows Server должна применяться для имени входа SQL Server. Если для этого параметра задано значение ON, для параметра CHECK_POLICY также должно быть задано значение ON. Иначе инструкцию не удастся выполнить
CHECK_POLICY	Если для этого параметра задано значение ON (по умолчанию), эта политика указывает, что политика сложности Windows Server должна применяться для имени входа SQL Server.

В следующем примере продемонстрирована инструкция CREATE LOGIN, с помощью которой создается имя входа SQL, которое управляется политикой пароля, определенной для сервера.

```
CREATE LOGIN Server1C WITH PASSWORD = 'password', DEFAULT_DATABASE = DB1C, CHECK_EXPIRATION = ON, CHECK_POLICY = ON
```

Важно! Использование параметров CHECK_EXPIRATION и CHECK_POLICY возможно только для Windows Server. Кроме того, в Windows существует проблема — не выполняется сброс счетчика неверных паролей по достижении значения **LockoutThreshold**. Это может привести к немедленной блокировке при последующих неудачных попытках входа. Вы можете вручную сбросить счетчик неправильных паролей, быстро указав для параметра CHECK_POLICY значение OFF, а затем значение ON.

Изменение имен входа

Вы можете изменить имя входа, просмотрев его свойства в обозревателе объектов или выполнив инструкцию ALTER LOGIN. Инструкция ALTER LOGIN часто используется, чтобы разблокировать имя входа, которое было заблокировано в связи с истекшим сроком действия пароля. В следующем примере демонстрируется, как разблокировать заблокированную учетную запись.

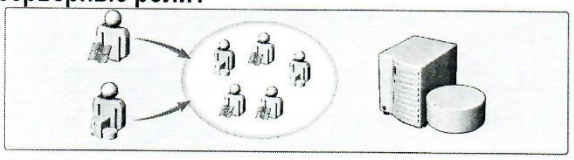
ALTER LOGIN Server1C WITH PASSWORD = 'NewPa$$w0rd' UNLOCK

Удаление имен входа

Можно удалить имя входа, щелкнув его правой кнопкой мыши в обозревателе объектов, а затем нажав кнопку **Удалить** или выполнив инструкцию DROP LOGIN, как это показано в следующем примере.

DROP LOGIN Server1C

Что собой представляют фиксированные серверные роли?



Роль	Описание
sysadmin	Выполнение любых действий
dbcreator	*Создание и изменение баз данных*
diskadmin	Управление файлами дисков
serveradmin	Настройка параметров безопасности по всему серверу
securityadmin	Управление и аудит имен входа сервера
processadmin	**Управление процессами SQL Server**
bulkadmin	Выполнение инструкции BULK INSERT
setupadmin	Настройка репликации и связанных серверов

SQL Server обеспечивает определенные заранее серверные роли для общих административных функций, чтобы определенному пользователю можно было легко предоставить набор административных разрешений. Фиксированные серверные роли обеспечивают группирования административных привилегий на уровне сервера. Они управляются независимо от баз данных пользователя на уровне сервера.

Фиксированные серверные роли

Фиксированные серверные роли описаны в следующей таблице.

Роль	**Описание**
sysadmin	Выполнение любых действий
dbcreator	Создание, изменение и удаление баз данных
diskadmin	Управление файлами дисков
serveradmin	Настройка параметров безопасности по всему серверу
securityadmin	Управление и аудит имен входа сервера

processadmin	***Управление процессами SQL Server***
bulkadmin	Выполнение инструкция BULK INSERT
setupadmin	Настройка репликации и связанных серверов

В SQL Server 2017 можно создавать дополнительные серверные роли, если функций встроенных ролей недостаточно.

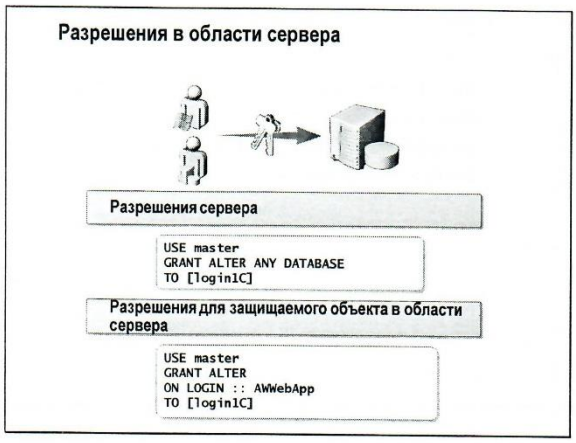
Роль ***processadmin*** для имени входа позволяет серверу системы «1С:Предприятие 8» управлять процессами.

Назначение учетной записи входа для фиксированной роли сервера

Можно использовать свойства имен входа SQL Server в SQL Server Management Studio или команду ***ALTER SERVER ROLE***, чтобы добавить учетную запись входа в качестве участника фиксированной серверной роли. При добавлении учетной записи входа к серверной роли, учетной записи предоставляются разрешения, связанные с серверной ролью. При назначении учетных записей входа фиксированным серверным ролям, помните о следующих указаниях:

• Нельзя изменять или удалять фиксированные серверные роли.

• Любой участник фиксированной серверной роли может добавлять учетные записи входа к этой роли.



К защищаемым объектам в области сервера относятся такие объекты как сам сервер, имена входа и базы данных.

Разрешения на уровне сервера

В приведенной ниже таблице содержатся некоторые примеры разрешений.

Защищаемый объект	**Разрешение**	**Описание**
Сервер	CONNECT_SQL	Соединение с сервером
	CREATE LOGIN	Создание имени входа

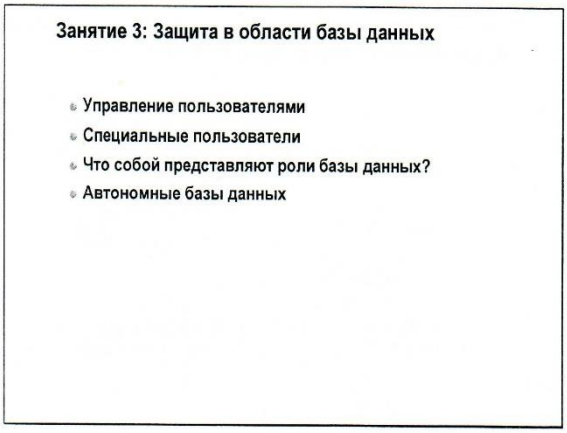
	ALTER ANY LOGIN	Изменение имени входа в области сервера
	CONTROLSERVER	Полное административное управление
Имя входа	ALTER	Изменение имени входа
	IMPERSONATE	Олицетворение имени входа
База данных	CREATE TABLE	Создание таблицы в базе данных
	ALTER ANY USER	Изменение любого пользователя в базе данных
	CONTROL	Полное управление базой данных

Предоставление разрешений для защищаемых объектов в области сервера

Чтобы предоставить разрешения для защищаемых объектов в области сервера:

1. Используйте обозреватель объектов в SQL Server Management Studio для просмотра свойств имен входа, для которых следует предоставить разрешения.

2. На вкладке **Защищаемые объекты** добавьте необходимые защищаемые объекты и задайте нужные разрешения.



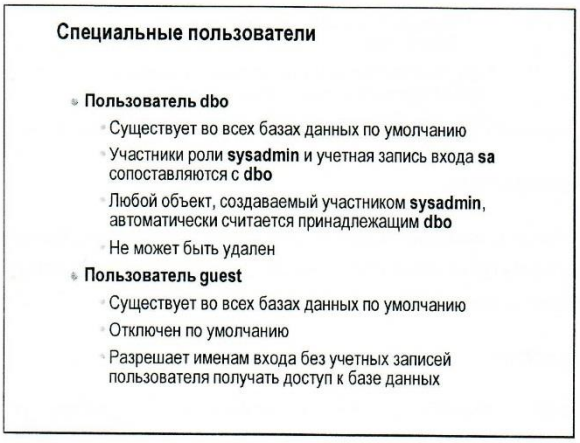
На этом занятии рассматривается настройка безопасности на уровне базы данных в SQL Server 2016. Важно понимать, каким образом выполняется обеспечение безопасности на уровне базы данных, чтобы управлять доступом к базам данных в экземпляре SQL Server и регулировать действия, которые пользователи могут выполнять в базе данных.

Имена входа используются для предоставления доступа в систему SQL Server. Однако доступ к отдельным базам данных осуществляется путем создания пользователей в этих базах данных. Создавать пользователей можно с помощью обозревателя объектов в SQL Server Management Studio или выполнив инструкцию CREATE USER в соответствующей базе данных.

Сопоставление пользователей с именами входа

В большинстве случаев пользователи баз данных сопоставляются с именами входа. Например, можно создать пользователя с именем **Server1C** в базе данных DB1C чтобы сделать эту базу доступной при использовании имени входа **Server1C.** По умолчанию все имена входа, которые являются участниками фиксированной серверной роли sysadmin, сопоставляются с пользователем dbo во всех базах данных.

Пользователя можно создать в SQL Server Management Studio, используя обозреватель объектов или инструкцию CREATE USER Transact-SQL.



Специальные пользователи в базе данных являются заранее определенными пользователи, у которых есть специальные функции, например возможность предоставления административного или гостевого доступа.

dbo

Имя входа sa и участники роли sysadmin сопоставляются со специальной учетной записью пользователя во всех базах данных с именем dbo. Любой объект, создаваемый системным администратором, автоматически считается принадлежащим dbo. Пользователь dbo является учетной записью по умолчанию и не может быть удален.

guest

Учетная запись пользователя guest разрешает доступ к базе данных именам пользователя без учетных записей пользователя. Учетная запись входа принимает идентификатор пользователя guest, если выполняются такие условия:

• Учетная запись входа обладает правом доступа к SQL Server, но не обладает доступом к базе данных через свою учетную запись пользователя.

• Учетная запись пользователя включена.

Учетную запись гостя можно активизировать в любой базе данных, кроме баз **master** и **tempdb**, где она всегда активна.

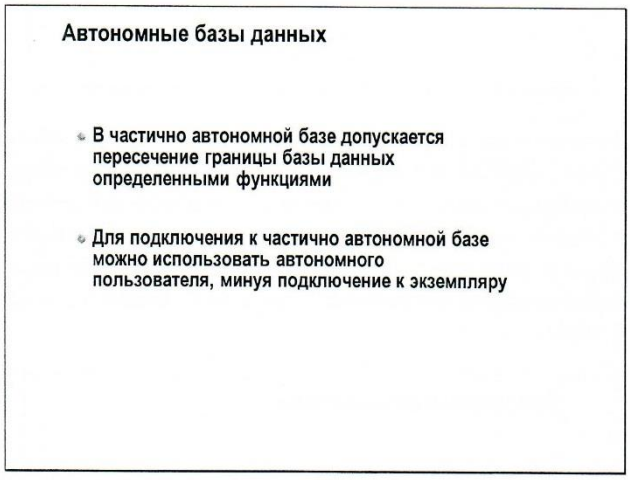
Фиксированные роли базы данных

Фиксированные роли базы данных представляют группирования административных привилегий на уровне базы данных, которым были предоставлены разрешения на стандартные задачи базы данных. Фиксированные роли в базе данных описаны в следующей таблице.

Роль	Описание
db_accessadmin	Добавление или удаление пользователей базы данных, групп и ролей
db_backupoperator	Резервное копирование базы данных
db_datareader	Считывание данных из любой таблицы
db_datawriter	Добавление, изменение или удаление данных из любой таблицы

db_ddladmin	Добавление, изменение и удаление объектов базы данных
db_denydatareader	Невозможность считывания данных из любой таблицы
db_denydatareader	Невозможность изменения данных в любой таблице
db_owner	**Выполнение любых действий роли базы данных**
db_securityadmin	Изменение ролей базы данных, изменение ролей приложений, создание схем
public	Обеспечение разрешений по умолчанию

Роль db_owner достаточна для использования в системе «1С:Предприятие 8» базы данных, созданной заранее на SQL Server

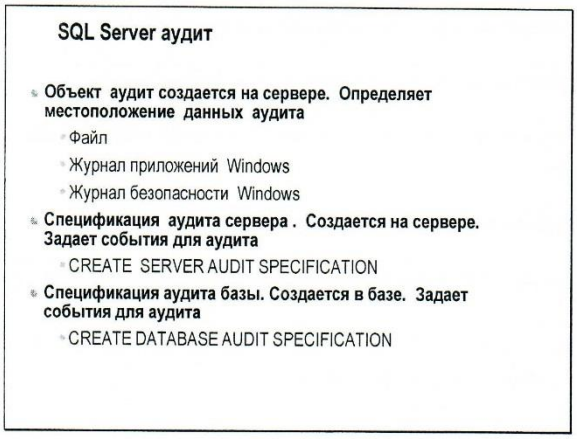


Для баз данных SQL Server 2017 в настоящее время доступна только частичная автономия. Частично автономной базой данных является автономная база данных, которая разрешает использование неавтономных функций. В частично автономных базах данных важная информация может храниться в базе данных, в результате чего она останется в базе данных и после ее переноса.

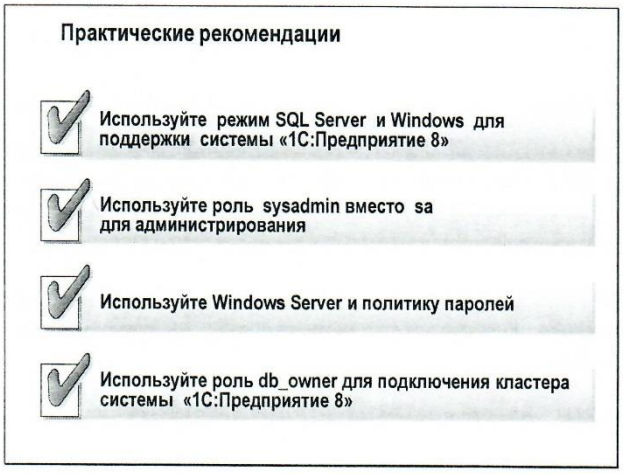
Приложение 1CV83 Server может подключиться и использовать частично автономную базу, но не будет обладать правами серверной роли processadmin. Ниже приведены команды настройки сервера и автономности базы данных.

```
EXEC SYS.SP_CONFIGURE N'CONTAINED DATABASE AUTHENTICATION', N'1'

GO

RECONFIGURE WITH OVERRIDE

GO

USE [master]

GO
```

```
ALTER DATABASE [DB1C] SET CONTAINMENT = PARTIAL WITH NO_WAIT

GO
```



В SQL Server можно обеспечить аудит безопасности с помощью специальных объектов аудита. Аудит создается командой CREATE SERVER AUDIT или через графический интерфейс. Включается командой ALTER SERVER AUDIT. Аудит на уровне сервера поддерживается во всех выпусках SQL Server. Аудит на уровне базы данных доступен только в выпусках Enterprise Edition, Developer Edition и Evaluation Edition. В SQL Server 2017 с помощью хранимой процедуры sp_audit_write можно регистрировать в аудите пользовательские события аудита, предварительно определив в спецификации аудита класс событий USER_DEFINED_AUDIT_GROUP

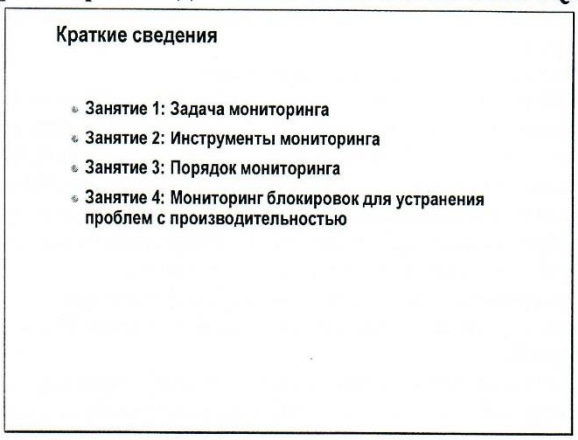


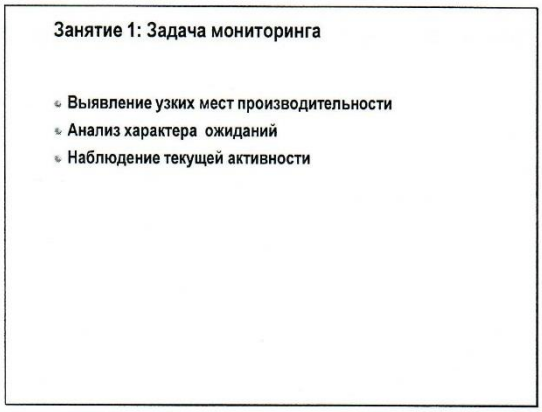
Задание 6. Настройка подключения сервера системы «1С:Предприятие 8» к готовой базе SQL Server.

1. Запустите на своей машине **SQL Server Management Studio** зарегистрируйте **Database Engine** локального сервера, используя проверку подлинности Windows
2. В SQL Server создайте базу данных **DBOwner1CX**, где **X** – ваш номер
3. Перейдите в контейнер **Безопасность.**

4. Создайте имя входа **Login1C** типа SQL Server
5. В свойствах сервера SQL Server в разделе **Разрешения** посмотрите действующие разрешения для имени входа **Login1C**
6. Добавьте имя входа в серверную роль **ProcessAdmin**
7. В базе данных **DBOwner1CX** создайте пользователя **Login1C** для имени входа **Login1C**. Добавьте созданного пользователя в роль **db_owner.**Это предоставит необходимые и достаточные права для подключения сервера «1С: Предприятие 8» и использования этой базы.
8. В контекстном меню базы выберите команду **Свойства** и в разделе **Разрешения** установите для этого пользователя запрет на выполнение резервного копирование базы и журнала транзакций.
9. Проверьте настройки, запустив 1С и создав на кластере пустую базу под загрузку с именем **DBOwner1CX**, где **X -** две последние цифры вашего IP-адреса
10. Запустите «1С:Предприятие 8» и убедитесь, что база работоспособна.
11. Проверьте возможность создания резервной копии базы данных **DBOwner1C**, подключившись к SQL Server как **Login1C**
12. Создайте аудит в разделе **Безопасность** сервера. Настройте аудит в файл.
13. Создайте спецификацию аудита в разделе **Безопасность** базы DB1C для отслеживания выборки (**Select**) данных из таблицы **V8USERS**
14. Запустите «1С:Предприятие 8»
15. Проверьте работу аудита, просмотрев его журнал через контекстное меню

Раздел 5: Мониторинг производительности и активности SQL Server 2017



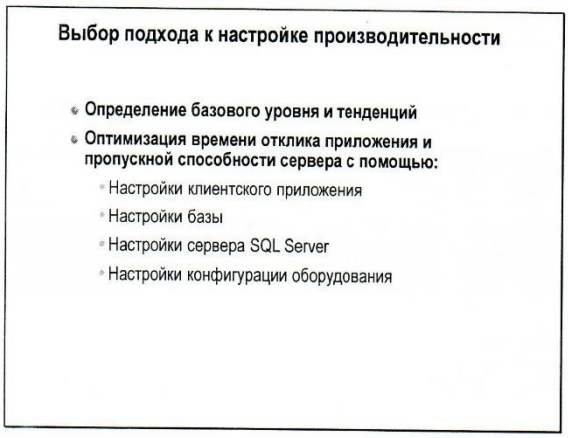


Первый этап наблюдения за производительностью — понимание основных целей мониторинга. Ими являются:

- с точки зрения пользователя — снижение времени реакции на передаваемые серверу запросы (время на возвращение пользователю первого ряда набора результатов). Таким образом, пользователь получает визуальное подтверждение того, что его запрос обрабатывается.
- с точки зрения сервера — максимизация общей пропускной способности (число запросов, обрабатываемое им за определенный период времени). Чтобы достичь этого, потребуется предпринять множество действий — подобрать подходящее оборудование, спроектировать базу так, чтобы не налагалось чрезмерно большое число блокировок, и создать приложения, генерирующие эффективные запросы.

Администратор обычно не может управлять всеми факторами производительности. Однако доступные вам средства мониторинга позволяют выявить и изолировать источник проблем производительности. Например, если проблема связана с некорректной структурой базы, вызывающей очень большое число блокировок, попытка решить ее при помощи новых аппаратных ресурсов даст минимальный выигрыш в производительности. Для повышения производительности SQL Server необходимо предварительно выявить его узкие места — ограничивающее производительность условия, вызванные интенсивным использованием системного ресурса или объекта базы. Возникновение узких мест также приводит к неполному использованию других системных ресурсов и объектов базы. Узким местом

производительности может быть оборудование, например память или процессор. Обычно для устранения таких узких мест можно нарастить аппаратные ресурсы или перенести часть нагрузки на другие серверы. На то, что узким местом является оборудование, зачастую указывает интенсивное использование одного или нескольких устройств. Повышенная нагрузка на процессор не всегда означает, что его нужно заменить более мощным. Источником проблемы могут быть и другие факторы, включая недостаточный объем памяти, который приводит к записи страниц памяти на жесткий диск, и неэффективные запросы, повышающие нагрузку на процессор. Мониторинг производительности — это процесс выявления ограничивающих ее факторов с целью их последующего устранения. Устранение одного узкого места иногда помогает выявить и другие. С увеличением числа обращений к базе, возможно, потребуется оптимизировать запросы, которые при небольшом количестве пользователей и избыточном объеме аппаратных ресурсов считались эффективными.



Определение базового уровня и тенденций

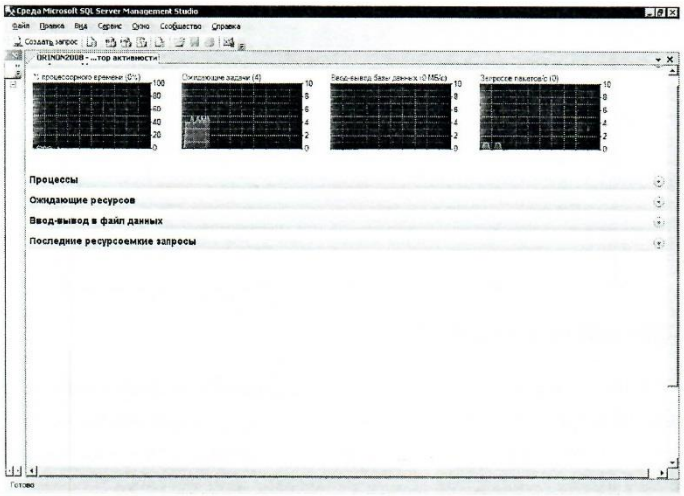
В процессе мониторинга SQL Server следует уяснить нормальный диапазон значений различных счетчиков. Благодаря этому вы сможете обнаружить проблему в самом начале и предпринять необходимые действия по ее устранению. С помощью различных средств мониторинга определите базовый уровень производительности SQL Server. Это позволит вам понять, как работают различные компоненты системы в обычных условиях и перед возникновением проблем. Периодически корректируйте базовый уровень производительности при помощи тех же средств и методов мониторинга. Анализируйте любые значительные изменения этого уровня. Записывая и наблюдая однотипные значения, вы со временем сможете выявлять значения, сильно отличающиеся от нормальных. Как правило, их появление означает скорое возникновение проблемы, требующей дополнительного изучения. Корректируемый базовый уровень позволяет администратору определить, когда необходимы дополнительные аппаратные ресурсы, индексы или оптимизация часто выполняемых запросов. Регулярное наблюдение за тенденциями использования базы позволит вам выявить области, требующие повышенного внимания и дополнительных ресурсов.

Обычно, чтобы выявить источник проблемы, с помощью одной или нескольких утилит мониторинга сначала определяют ее симптомы. Затем проводят дополнительные наблюдения и собирают специфическую информацию, которая поможет изолировать источник проблемы.

Занятие 2: Инструменты мониторинга

- Монитор активности
- Системный монитор
- Представления динамического управления
- Приложение SQL Profiler
- Утилита SQLDiag
- Сбор данных
- Мониторинг расширенных событий
- Хранилище запросов

Что представляет собой монитор активности



Монитор активности представляет собой графическое средство среды SQL Server Management Studio, в котором отображается информация о текущих процессах, ожиданиях ресурсов, вводе-выводе и ресурсоемких запросах.

В виде графиков отображаются: % процессорного времени, ожидающие задачи, ввод-вывод базы данных (мб/с), запросов пакетов /с.

Отдельные группы содержат подробную информацию по разделам

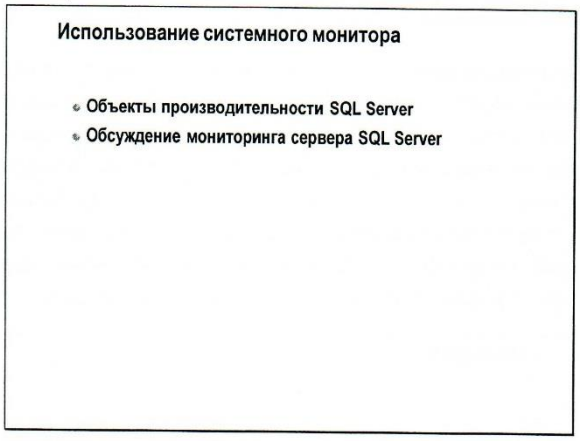
• **Процессы**. В таблице **Процессы** монитор активности отображает список всех процессов подключения в системе. Каждый процесс идентифицируется уникальным идентификатором процесса, и монитор активности отображает информацию о каждом процессе, таком как доступ к базе данных, имя клиентского приложения и имя для входа, использованное процессом. Администратор может уничтожить любой процесс или запустить приложение SQL Profiler.

• **Ожидающие ресурсов.** В таблице **Ожидающие ресурсов** монитор активности отображает все блокировки и ожиджания, относящиеся к пользовательским процессам.

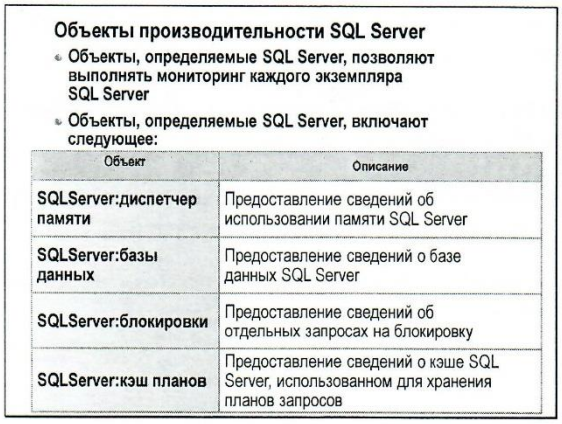
• **Ввод-вывод в файл данных**. В таблице **Ввод-вывод в файл данных** отображаются операции чтения и записи во все файлы всех баз данных.

• **Последние ресурсоемкие запросы**. В таблице **Последние ресурсоемкие запросы** монитор активности отображает сведения о затратах в последних ресурсоемких запросах.

• **Активные ресурсоемкие запросы**. В таблице **Активные ресурсоемкие запросы** монитор активности отображает сведения о затратах в активных ресурсоемких запросах.



Системный монитор может использоваться для получения полной информации о вашем компьютере и экземплярах SQL Server, работающих на этом компьютере. Эта информация может применяться для диагностики проблем производительности и выявления узких мест в системе. В этом уроке вы узнаете о том, каким образом средство «Системный монитор» в консоли управления (MMC) «Производительность» операционной системы Microsoft Windows используется для сбора и просмотра данных в журнале или в режиме реального времени, относящихся к памяти, диску, процессору и активности SQL Server.



SQL Server предоставляет объекты и счетчики, которые могут использоваться системным монитором для отслеживания активности на компьютерах, на которых выполняется экземпляр SQL Server. Объектом является любой ресурс SQL Server, например диспетчер блокировок SQL Server. Каждый объект содержит один или более счетчиков, определяющих различные аспекты объектов, за которыми осуществляется наблюдение. Например, объект **SQLServer: блокировки** содержит счетчики **Количество взаимоблокировок/сек** и **Превышений времени ожидания блокировки/сек.** В следующей таблице описываются несколько наиболее часто используемых объектов SQL Server. Показания счетчиков можно снимать внутри SQL Server с помощью DMV sys.dm_os_performance_counters.

Объект производительности	**Описание**
SQLServer:методы доступа	Осуществляет поиск объектов базы данных SQL Server и измеряет их распределение (например, количество поисков по индексу или количество страниц, которые распределены для индексов и данных).
SQLServer:базы данных	Предоставляет сведения о базе данных SQL Server, например о доступном объеме свободного места или количестве активных транзакций в базе данных. В системе может быть несколько экземпляров этого объекта
SQLServer:блокировки	Предоставляет сведения об индивидуальных запросах блокировки, сделанных сервером SQL Server, например о времени ожидания блокировки и взаимоблокировках. В системе может быть несколько экземпляров этого объекта.
SQLServer:диспетчер памяти SQLServer:Memory Manager	Предоставляет сведения об использовании памяти SQL Server, например об общем количестве распределенных в данный момент структур блокировки.
SQLServer:кэш планов	Предоставляет сведения о кэше SQL Server, использованном для хранения объектов, таких как хранимые процедуры, триггеры и планы

	запросов.
SQLServer:транзакции	Предоставляет сведения об активных транзакциях в SQL Server.

Мониторинг использования памяти

Объект: Счетчик	Рекомендации
SQL Server:Memory Manager: Total Server Memory/SQL ServerMemory Manager: Target Server Memory	~1
SQL Server: Memory Manager: Lock Memory	< 24% SQL Server: Memory Manager: Total Server Memory
SQL Server: Memory Manager : Memory Grants Pending	~0

Мониторинг использования памяти

Чтобы отслеживать условие, связанное с нехваткой памяти, используйте счетчики объектов, описанные в следующей таблице.

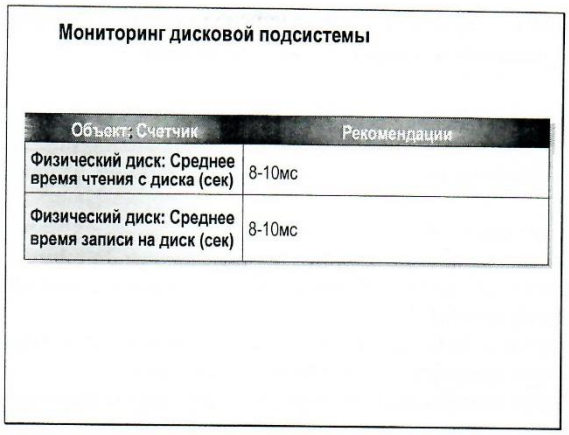
SQLServer: Memory Manager – счетчик	Описание	Рекомендации
Target Server Memory (KB) Память целевого сервера	Идеальное количество памяти, которое может использовать SQL Server	Total Server Memory/Target Server Memory ~ 1
Total Server Memory (KB) Общая память сервера	Количество памяти, которое взял диспетчер памяти	Total Server Memory/ Target Server Memory ~ 1
Memory Grants Pending Ожидающие получения памяти	Текущее количество процессов, ожидающих получения памяти в рабочем пространстве	Среднее значение должно быть приблизительно равно 0
Lock Memory Память блокировок	Объем оперативной памяти, занимаемый блокировками	< 24% SQL Server:Memory Manager:Total Server Memory

Мониторинг использования процессора	
Объект: Счетчик	**Рекомендации**
Процессор: % загруженности процессора	< 80%
Система: Длина очереди команд процессора	< (2 * число процессоров)
Процесс: % загруженности процессора (экземпляр sqlservr)	Оценка вклада SQL Server

Мониторинг загрузки процессора

Осуществляйте периодический мониторинг экземпляра SQL Server, чтобы определить, находится ли значение показателя загрузки процессора в нормальном диапазоне. Альтернативным образом высокий показатель загрузки процессора может свидетельствовать о плохой настройке соответствующего приложения или о том, что при разработке этого приложения были допущены ошибки. Оптимизация приложения может привести к снижению загрузки процессора. Используйте счетчики, описанные в следующей таблице, для наблюдения за загрузкой процессора.

Объект – Счетчик	Описание	Рекомендации
Процессор -% загруженности процессора	Осуществляет наблюдение за общим временем, которое затрачивается процессором на выполнение потока, отличного от простоя.	Если значение счетчика устойчиво составляет больше 80%, это может свидетельствовать о необходимости обновления процессора или добавления нескольких дополнительных процессоров. Для многопроцессорных систем следует осуществлять наблюдение за отдельным экземпляром этого счетчика для каждого процессора.
Процесс - % загруженности процессора (экземпляр sqlservr)	Осуществляет наблюдение за общим временем, которое затрачивается процессором на выполнение потока в процессе SQL Server.	Используйте этот счетчик для оценки вклада SQL Server в общую загрузку процессора.



Мониторинг дисковой подсистемы

Объект: Счетчик	Рекомендации
Физический диск: Среднее время чтения с диска (сек)	8-10мс
Физический диск: Среднее время записи на диск (сек)	8-10мс

Мониторинг активности дисков

SQL Server 2017 использует запросы на ввод-вывод в операционной системе Windows для осуществления операций считывания с диска и записи в дисковых подсистемах. SQL Server управляет временем и способом выполнения дисковых операций ввода-вывода, но соответствующие операции ввода-вывода выполняет операционная система Windows. Дисковые операции ввода-вывода часто приводят к возникновению узких мест в системе. Мониторинг активности дисков охватывает следующие две основных области:

• Мониторинг дисковых операций ввода-вывода и обнаружение излишней подкачки.

• Изоляция активности диска, созданной SQL Server. Для определения дисковых операций ввода-вывода и обнаружение излишней подкачки можно отслеживать следующие счетчики в объекте **Физический диск**.

Объект:Счетчик	**Описание**	**Рекомендации**
Физический диск: Среднее время чтения с диска (сек)	Отслеживает среднее время в секундах, требуемое для чтения данных с диска.	8 – 10мс
Физический диск: Среднее время записи на диск (сек)	Отслеживает среднее время в секундах, требуемое для записи данных на диска.	8 – 10мс

Если на одном жестком диске существует больше одного логического раздела, используйте счетчики **Логический диск**, а не счетчики **Физический диск**. Наблюдайте за счетчиками логических дисков, чтобы определить, к каким файлам часто обращаются. Если значения счетчиков приближаются к пределу пропускной способности оборудования подсистемы ввода-вывода, попытайтесь уменьшить значения путем настройки приложения или базы данных, чтобы уменьшить объем операций ввода-вывода (например: увеличением объема индексированных данных, улучшением индексов или нормализацией), увеличивая пропускную способность оборудования ввода-вывода или добавляя память. Например, можно использовать помощник по настройке ядра СУБД для анализа обычных

рабочих нагрузок SQL Server и получения рекомендаций для индексов, индексированных представлений и секционирования для улучшения работы сервера

Задание 7. Мониторинг с помощью утилиты System Monitor

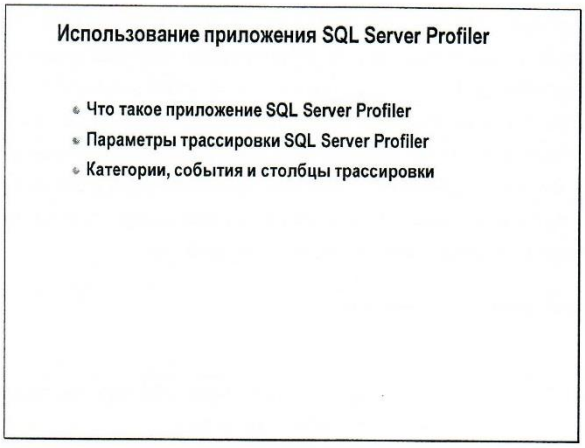
1. Запустите на своей машине **Системный монитор,** вызвав оснастку **Производительность** в средствах администрирования или выполнив команду **perfmon**, и подключитесь к локальному компьютеру.
2. Перейдите в раздел **Группа сборщиков данных**
3. Создайте новый пользовательский набор сборщиков данных. Назовите его **SQL.** Выберите вариант **Создать вручную (для опытных).** Выберите **Создать журнал** и отметьте галочкой **Счетчик производительности**
4. Настройте мониторинг работы SQL Server с оперативной памятью и процессором. Используйте следующие счетчики, предварительно прочитав объяснения:
    - **SQLServer: Memory Manager\Lock Memory (Память блокировок) (KB).**
    - **SQLServer: Memory Manager \Memory Grants Pending (Ожидающие получения памяти).**
    - **SQLServer: Memory Manager \Total Server Memory (Общая память сервера) (KB)**
    - **SQLServer: Memory Manager \Target Server Memory (Память целевого сервера) (KB)**
    - **Процесс: %загруженности процессора\sqlservr**
5. Запустите сбор показаний счетчиков
6. Запустите приложение «1С:Предприятие 8».
7. Выполните любые с вашей точки зрения ресурсоемкие операции.
8. Переключитесь в окно **Производительность**
9. Остановите сбор показаний счетчиков
10. Просмотрите отчет сборщика данных SQL
11. Оцените значения счетчиков.
12. Соответствуют ли значения, которые вы наблюдали, рекомендованным значениям счетчиков?

Представления динамического управления и функции динамического управления могут использоваться для запроса динамических метаданных в SQL Server 2016. Они предоставляют информацию о текущем состоянии SQL Server, запрашиваются с помощью стандартной инструкции SELECT.

Представления динамического управления

Представления динамического управления перечислены вместе с представлениями каталогов в папке **Системные представления**, которая находится в обозревателе объектов в среде SQL Server Management Studio. В отличие от представлений каталогов, которые могут использоваться для представления статистической информации о конфигурации, представления динамического управления возвращают сведения о текущем состоянии активности в SQL Server. Представления динамического управления, как и представления каталогов, определяются в схеме **sys,** однако их имена обычно содержат префикс **dm**, позволяющий отличить их от представлений каталогов. В следующей таблице перечислены некоторые из часто используемых представлений динамического управления.

Представления динамического управления	**Описание**
sys.dm_exec_sessions	Возвращает информацию обо всех текущих сеансах, подключенных к серверу
sys.dm_io_pending_io_requests	Возвращает информацию об ожидающих запросах ввода-вывода
sys.dm_io_virtual_file_stats	Возвращает информацию о статистике ввода-вывода
sys.dm_os_threads	Возвращает информацию о потоках в системе
sys.dm_tran_locks	Возвращает информацию о каждой предоставленной в данный момент или запрошенной блокировке в системе
sys.dm_exec_requests	Возвращает одну строку о каждом запросе, существующем в SQL Server
sys.dm_os_wait_stats sys.dm_exec_session_wait_stats	Возвращает агрегированные данные о числе ожиданий, зафиксированных потоками, выполняющимися в данный момент.
sys.dm_os_waiting_tasks	Возвращает сведения об очереди задач, ожидающих освобождения определенного ресурса.



SQL Server Profiler предоставляет возможность выполнять трассировку активности сервера и базы данных, например активности, связанной с выполнением входа, действиями пользователя и приложения. Соответствующие данные могут записываться в таблицу, файл или сценарий Transact-SQL для последующего анализа.

Microsoft SQL Server Profiler представляет собой графическое средство интерфейса пользователя, используемое для мониторинга экземпляра ядра СУБД SQL или служб Analysis Services. Данные о каждом событии можно записывать и сохранять в файле или таблице для последующего анализа. Например, может осуществляться наблюдение рабочей среды с целью выявить хранимые процедуры, которые влияют на производительность в результате их слишком медленного выполнения.

Функции SQL Server Profiler

SQL Server Profiler показывает, как SQL Server разрешает запросы внутренним образом, позволяя администраторам просматривать, какие точно инструкции Transact-SQL передаются на сервер и как сервер осуществляет доступ к базе данных, чтобы возвратить результирующий набор. Использование SQL Server Profiler позволяет:

• создавать трассировку, основанную на шаблоне, который может использоваться многократно;

• просматривать результаты, получаемые при запуске трассировки;

• сохранять результаты трассировки в таблицу или файл для последующего анализа;

• запускать, останавливать, приостанавливать и модифицировать трассировку в случае необходимости;

• воспроизводить результаты трассировки.

Используйте SQL Server Profiler для просмотра только тех событий, которые представляют для вас интерес. Если активность слишком велика, что затрудняет ее анализ, можно отфильтровать события на основе нужной вам информации, чтобы осуществлялся сбор только некоторого подмножества данных о событиях. Мониторинг слишком большого числа событий усиливает нагрузку на сервер и процесс мониторинга. Это может привести к слишком быстрому росту файла или таблицы трассировки, особенно в тех случаях, когда процесс мониторинга осуществляется в течение длительного периода.

Трассировка SQL Server с помощью SQL Server Profiler

При использовании SQL Server Profiler необходимо сначала решить, трассировку каких процессов требуется выполнять, а затем выбрать соответствующие критерии. К видам активности, которые может потребоваться отслеживать, относятся:

• плохо выполняемые запросы;

• запросы, вызывающие просмотр таблицы;

• действия отдельных пользователей или приложений;

• производительность базы данных **tempdb**;

• проблемы взаимоблокировки;

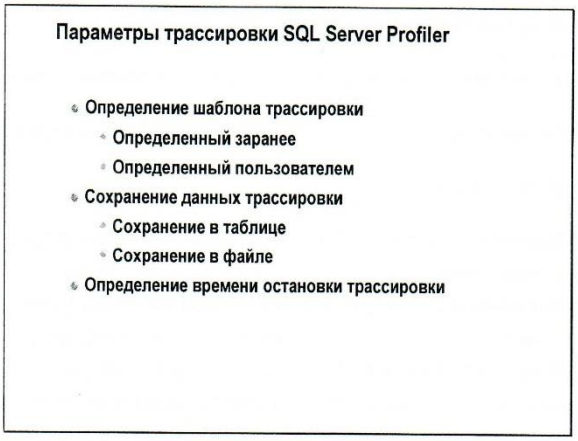
• попытки входа, сбои, подключения и отключения;

• дисковые операции чтения-записи;

• загрузка процессора на уровне инструкций;

• время ожидания для всех событий, следующих после выполнения.

Можно задать системные хранимые процедуры в SQL Server Profiler для трассировки определенного набора событий и фильтрации уровня информации, собираемой об этих событии. С помощью системной хранимой процедуры **sp_trace_create** может осуществляться трассировка определенных событий на сервере.



При использовании SQL Server Profiler для создания трассировки может использоваться ряд параметров, определяющих, какая активность будет записываться, и где будет храниться журнал трассировки активности.

Задание шаблона трассировки

События, включенные в трассировку, определяются посредством задания классов событий, которые требуется отслеживать, и отдельных значений данных (столбцов), которые требуется записывать. Это выполняется с помощью выбора шаблона, на котором будет основываться трассировка. Затем осуществляется добавление или удаление отдельных классов событий или столбцов и применение фильтров, позволяющих ограничить собираемые данные, основываясь на конкретных критериях. SQL Server Profiler предоставляет ряд определенных заранее шаблонов, позволяющих легко настроить события, которые требуется отслеживать для конкретных видов активности. Например, шаблон **Standard** помогает создать стандартную трассировку для записи информации о входах, выходах, завершенных пакетах и подключении. Этот шаблон может без изменения применяться для запуска трассировки или же использоваться в качестве основы для создания дополнительных шаблонов с другими настройками событий. Можно также создать свои собственные шаблоны или изменить уже существующие шаблоны.

Сохранение данных трассировки

Данные трассировки событий следует записывать в файл или таблицу SQL Server, если необходимы для их анализа и воспроизведения в дальнейшем. Сохраняя трассировку, можно:

• использовать файл трассировки или таблицу трассировки для создания рабочей нагрузки, используемой в качестве входа входных данных для помощника по настройке ядра СУБД;

• использовать файл трассировки для записи событий и отправлять его для анализа поставщику услуг технической поддержки;

• использовать средства обработки запросов в SQL Server для доступа к данным или для просмотра данных в SQL Server Profiler. Прямой доступ к таблице трассировки имеют только члены фиксированной серверной роли **sysadmin** или же создатели этой таблицы. При сохранении трассировки в таблицу доступны следующие параметры:

• Местоположение и имя таблицы.

• Максимальное количество строк, которое может храниться в таблице (необязательно).

При сохранении трассировки в файл доступны следующие параметры:

• Местоположение и имя файла.

• Максимальный размер файла.

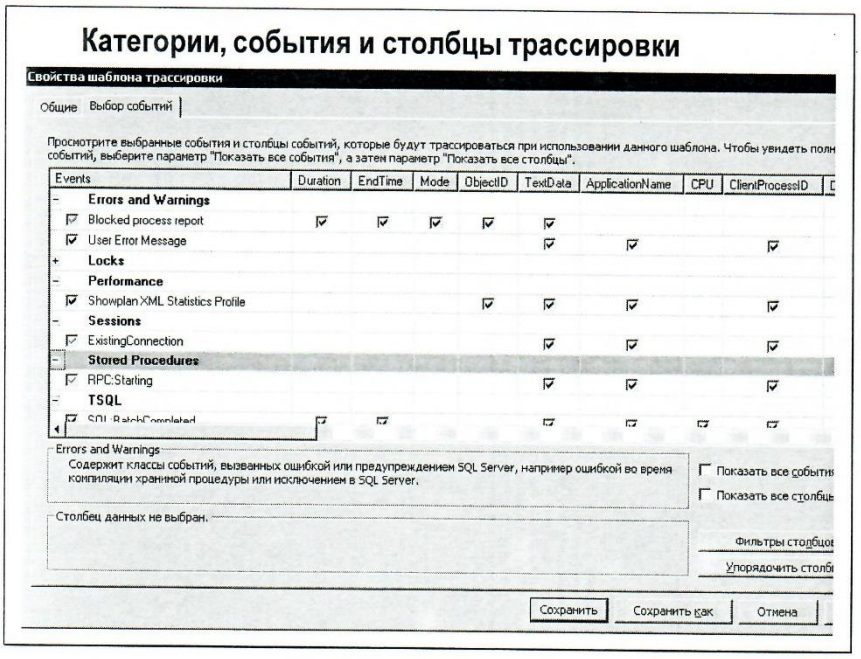
• Порядок записи данных трассировки при заполнении файла (повторная запись в начало этого же файла или создание нового файла).

• Обработка трассировки сервером или приложением SQL Server Profiler.

Настройка сервера для обработки трассировки может снизить воздействие трассировки на производительность.

Задание времени остановки трассировки

Время остановки трассировки может быть задано, что позволяет запускать трассировку и выполнять ее вплоть до указанной даты и времени. Возможность задавать время остановки трассировки является полезной, когда требуется записать сведения об активности SQL Server для заранее определенного периода.



Сведения, записываемые в трассировку, делятся на категории. Категории содержат события, каждое из которых имеет атрибуты, определяемые столбцами.

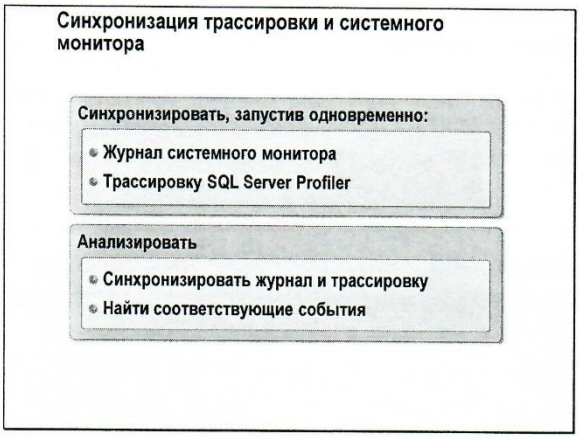
Категории трассировки

В SQL Server Profiler категория представляет собой группу взаимосвязанных классов событий. Классы события состоят из типов событий, трассировка которых может выполняться. Класс событий содержит все столбцы данных, относящихся к событию.

События

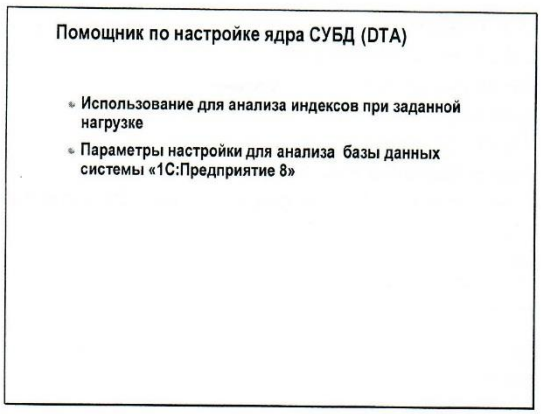
Событие определяется как проявление действия в экземпляре ядра СУБД SQL Server. События определяются также своими атрибутами, перечисленными в столбцах данных.

Столбцы данных содержат атрибуты событий. SQL Server Profiler использует столбцы данных в выходных данных трассировки для описания событий, записанных при запуске трассировки. Управление столбцами может осуществляться с помощью фильтров столбцов, определяющих, какие данные будут собираться. Например, использование фильтра **Имя приложения** позволяет исключить все данные, генерируемые самим приложением SQL Server Profiler. Столбцы можно также организовать в связанные группы с помощью функции **Упорядочение столбцов**.



В SQL Server можно синхронизировать трассировку SQL Server Profiler и журнал системного монитора, запустив их одновременно и параллельно.

После сбора данных за определенный период можно синхронизировать результаты с помощью команды **Импорт данных о производительности** в SQL Server Profiler. В задаче мониторинга блокировок полезно синхронизировать журнал системного монитора счетчика SQLServer: статистика ожиданий с трассировкой блокировок.



Помощник по настройке ядра СУБД - это инструмент для анализа влияния *рабочей нагрузки* на производительность в одной или нескольких базах данных. Рабочая нагрузка представляет собой набор инструкций Transact-SQL, которые выполняются в отношении баз данных, нуждающихся в настройке. После анализа влияния рабочей нагрузки на базы данных помощник по настройке ядра СУБД рекомендует добавить, удалить или изменить физическую структуру в базах данных Microsoft SQL Server. К структурам физической производительности относятся кластеризованные и некластеризованные индексы, индексированные представления, а также секционирование.

Помощник по настройке ядра СУБД располагает двумя интерфейсами:

- Автономным графическим пользовательским интерфейсом для настройки баз данных и просмотра рекомендаций и отчетов по настройке.
- Консольной программой **dta.exe** для обеспечения деятельности помощника по настройке ядра СУБД в программах и сценариях.

Для настройки базы данных системы «1С:Предприятие 8» более всего подходит следующий параметр помощника: **Некластеризованные индексы.** Установка этого параметра приводит к тому, что помощник по настройке ядра СУБД рассматривает возможность добавления только некластеризованных индексов. Рекомендованные помощником индексы нужно создавать на уровне структур 1С, установив соответствие таблиц базы SQL Server и 1С через глобальный контекст в 1С.

Примечание. Рекомендации помощника по настройке ядра основаны на анализе той нагрузки, которую ему предоставили. Поэтому помощник может предложить уничтожить индексы, которые были бы полезны при другой нагрузке.

Задание 8. Мониторинг с помощью утилиты SQL Profiler

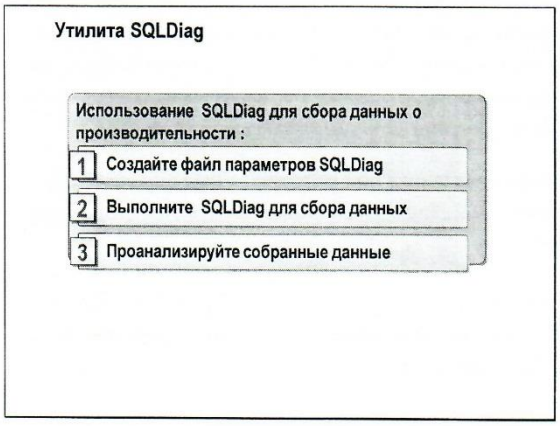
В свойствах сервера в разделе *Дополнительно->Разное* установите значение параметра *Порог заблокированных процессов* равным 5

Создайте в SQL Server Profiler шаблон и используйте его для мониторинга блокировок в базе DB1C, возникающих при работе сервера предприятия. В качестве исходного шаблона выберите шаблон TSQL_Locks

1. Установите в шаблоне отслеживание следующих событий.
    - **Errors and Warnings: Blocked process report**
    - **Errors and Warnings: User Error Message**
    - **Locks:Deadlock graph**

- **Locks:Lock: Deadlock**
- **Locks:Lock: Timeout (Timeout >0)**
- **Stored Procedures:RPC:Completed**
- **TSQL:SQL:BatchCompleted**
- **Session:Existing Connections**
- **Performance: Showplan XML Statistics Profile**

2. Проверьте, указан ли в шаблоне сбор следующих данных (**Data Columns.**)
   - **SPID**
   - **ObjectID**
   - **Mode**
   - **Error**
   - **TextData**
   - **StartTime**
   - **BinaryData**
3. Установите в шаблоне следующие фильтры.
   - **ApplicationName ->Похоже на->1CV83 Server**
4. Сохраните шаблон. Запустите трассировку по этому шаблону с сохранением результатов в файле.
5. Запустите журнал счетчиков системного монитора из предыдущего задания
6. Постарайтесь одновременно выполнить проведение документов одного типа в нескольких клиентских соединениях системы «1С:Предприятие 8»
7. Остановите трассировку.
8. Откройте файл трассировки
9. Проанализируйте результат трассировки.
10. Найдите длительные запросы и проанализируйте планы их выполнения
11. Синхронизируйте журнал счетчиков монитора и файл трассировки, используя команду **Импорт данных о производительности** из меню **Файл**.
12. Полученный файл трассировки используйте для анализа с помощью Помощника по настройке ядра



Программа SQLdiag может собирать следующие типы диагностических сведений: журналы производительности Windows; журналы событий Windows; трассировки SQL Server Profiler; сведения о блокировках SQL Server; сведения о конфигурации SQL Server. Можно запустить утилиту SQLDiag из командной строки или как службу Windows.

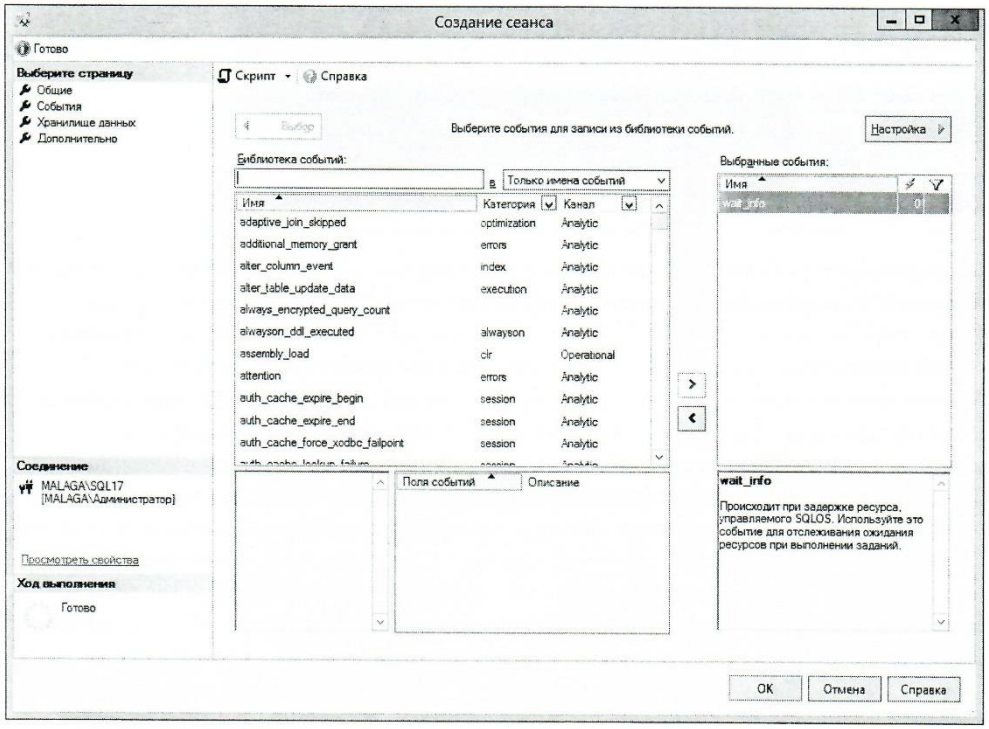
Запустите SQLDiag из командной строки укажите параметры, такие как директория для результатов, время начала сбора данных, время продолжительности сбора данных, файл параметров конфигурации. По умолчанию утилита SQLDiag использует файл параметров SQLdiag.xml. В этом файле можно указать экземпляр SQL Server для мониторинга, счетчики системного монитора для измерений и события для трассировки. Можно использовать параметр /? для отображения синтаксиса.

Для запуска SQLDiag как службы запустите SQLDiag с параметром /R. SQLDiag при этом зарегистрируется как служба Windows с именем SQLDIAG.

Поставщик данных является внешним для сбора данных и собирает информацию из SQL Server, которую потом можно использовать. Элемент коллекции представляет собой экземпляр типа коллекции с входными параметрами и частотой сбора данных. Тип коллекции это логическая обертка для пакета SSIS, обеспечивающего механизм сбора данных и загрузки в хранилище. Наборы коллекций развертываются на экземпляре SQL Server и работают независимо друг от друга благодаря заданиям SQL Agent. При определении набора коллекции можно задать режим сбора и загрузки данных: с кэшированием или без кэширования. В случае кэширования одно задание агента собирает данные, а другое задание по расписанию, заданному администратором, загружает данные в хранилище. После включения набора коллекции начинается сбор данных. Агент запускает задания, выполняющие пакеты SSIS. Когда цикл сбора заканчивается, данные загружаются в хранилище. База данных хранилища может быть расположена на отдельном экземпляре SQL Server. По данным из хранилища можно строить отчеты в SQL Server Management Studio, для отображения различных типов ожиданий и использования ресурсов.

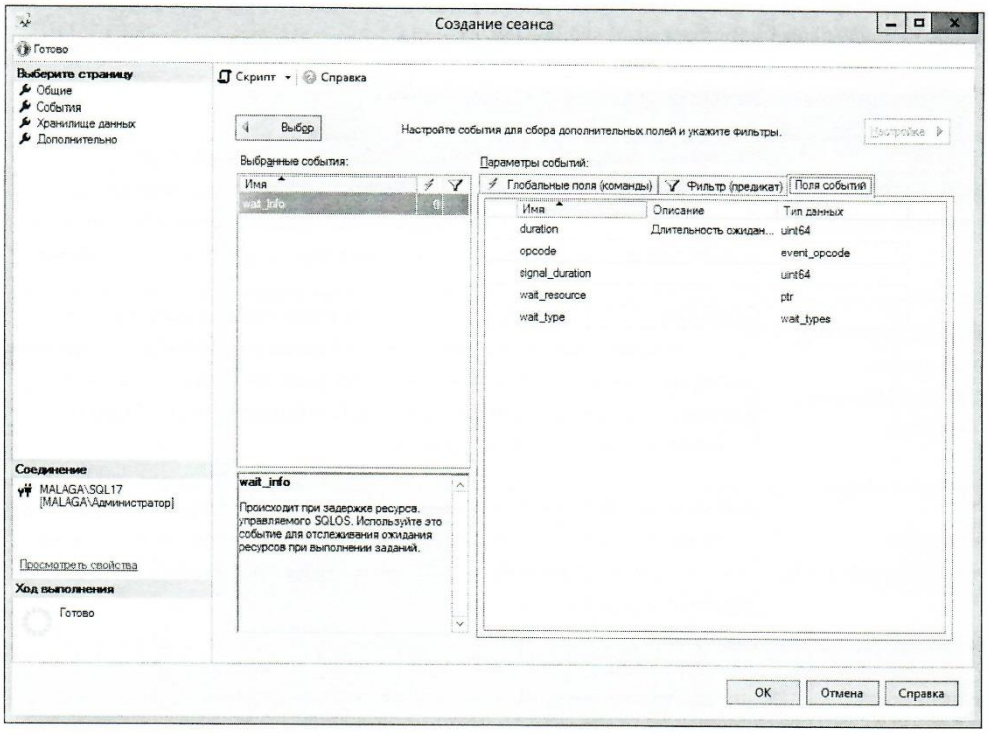
Расширенные события

- Отмечены в коде SQL Server (пакеты)
- Собирают информацию только когда событие происходит
- Минимальная нагрузка на систему
- Графический интерфейс для настройки и анализа

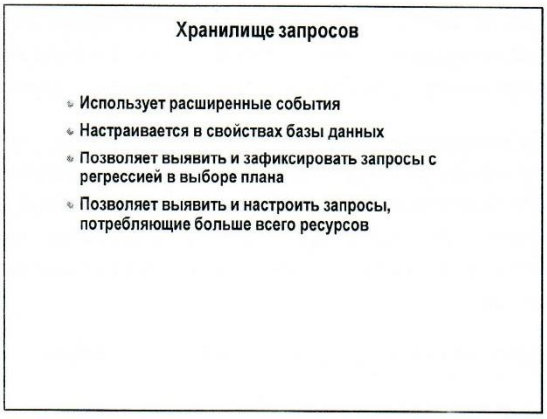


Возможности системы расширенных событий выходят далеко за пределы возможностей любого другого механизма отслеживания событий и устранения неполадок, предоставленного SQL Server. SQL Server 2017 поставляется с заранее определенным сеансом, который установлен на выполнение по умолчанию и именуется **system_health**. Этот сеанс отслеживает информацию, обычно используемую для отладки клиентских систем, например случаи взаимоблокировки или серьезные ошибки. Этот сеанс создается и запускается как часть процесса установки для экземпляра SQL Server. Он записывает события в кольцевой буфер и в файл.

Для настройки нового сеанса удобно использовать SQL Server Management Studio.



Настройку созданного сеанса можно сохранить в виде файла в формате XML для повторного использования.



Хранилище запросов представляет собой новый инструмент мониторинга на базе расширенных событий, появившийся в SQL Server 2016. Хранилище запросов настраивается в свойствах базы данных. Хранилище запросов следует настраивать в соответствии с рабочей нагрузкой и требованиями к устранению проблем производительности. Параметры по умолчанию подходят для быстрого запуска, но необходимо наблюдать за поведением хранилища запросов с течением времени и соответствующим образом настраивать его конфигурацию.

В узле базы данных в обозревателе объектов расположена вложенная папка Хранилище запросов. В контекстном меню Хранилища запросов имеются представления, каждое из которых выражается в виде какой-либо из следующих статистических функций.

Представление SSMS	Сценарий
Regressed Queries (регрессивные запросы)	Выявление запросов, метрики выполнения для которых недавно регрессировали (т. е. стали хуже). Используйте это представление для сопоставления наблюдаемых проблем производительности в приложении с фактическими запросами, которые необходимо улучшить или исправить.
Top Resource Consuming Queries (запросы, потребляющие больше всего ресурсов)	Выберите интересующую метрику выполнения и определите запросы, которые имели максимальные значения в указанном промежутке времени. Используйте это представление, чтобы сосредоточиться на наиболее важных запросах, которые оказывают самое большое воздействие на потребление ресурсов базы данных.
Tracked Queries (отслеживаемые запросы)	Отслеживайте выполнение наиболее важных запросов в режиме реального времени. Как правило, эти представления используются, когда имеются запросы с принудительными планами и требуется убедиться в стабильной производительности запросов.
Overall Resource Consumption (общее потребление ресурсов)	Анализируйте общее потребление ресурсов базы данных для любой из метрик выполнения. Используйте это представление для определения шаблонов ресурсов (дневная и ночная рабочие нагрузки) и оптимизации общего потребления для базы данных. SQL Server 2017 позволяет в этом представлении анализировать статистику ожиданий.

Хранилище запросов может без предупреждения изменять режим работы. Необходимо постоянно наблюдать за состоянием хранилища запросов, чтобы знать, что хранилище запросов работает, и предпринимать действия для исключения сбоев по предотвращаемым причинам.

Сбор данных в **Хранилище запросов** позволяет для выпуска SQL Server 2017 Enterprise использовать настройку **Автоматическая коррекция регрессии планов.** SQL Server должен выбирать наилучший план для текущего запроса. Но выбранный план может быть неоптимальным, если одни и те же запросы выполняются с разными значениями параметров. В этом случае SQL Server повторно использует из кэша ранее созданный план, который не будет оптимальным для текущего значения параметров. Это называется регрессией планов.

Настройка коррекции включается с помощью приведенной ниже команды.

```
ALTER DATABASE DB1C

SET AUTOMATIC_TUNING ( FORCE_LAST_GOOD_PLAN = ON );

-- Проверка установки параметра

SELECT * FROM  sys.database_automatic_tuning_options;
```

GO

При включенном **Хранилище запросов** и отключенной автоматической коррекции регрессии планов представление **sys.dm_db_tuning_recommendations** позволяет получить сведения о деградации и рекомендации по устранению. Сведения сохраняются до перезапуска службы.

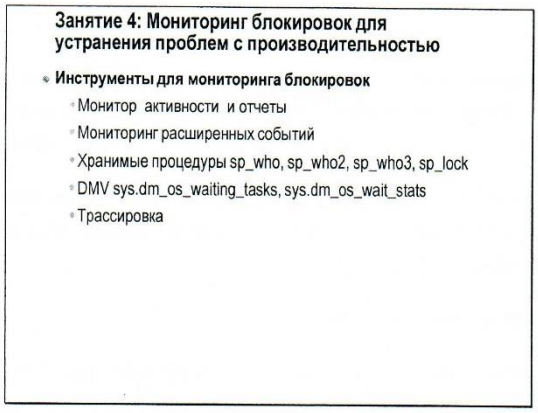
Расширенные события **automatic_tuning_plan_regression_detection_check_completed** и **automatic_tuning_plan_regression_verification_check_completed** можно использовать для отслеживания автоматической коррекции регрессии планов.

Сценарии использования хранилища запросов

- Выявление и фиксация запросов с регрессией в выборе плана
- Выявление и настройка запросов, потребляющих больше всего ресурсов
- Тестирование А/Б. Использовать хранилище запросов для сравнения производительности рабочей нагрузки до и после того, как приложение изменит план.
- Определение и улучшение нерегламентированных рабочих нагрузок

После отслеживания симптомов неоптимальной работы, например, медленного выполнения определенных бизнес операций, переходят к поиску причин и их устранению. Мониторинг и поиск причин неоптимальной работы состоит из нескольких этапов, направленных на сужение области поиска.

- Этап 1. Мониторинг окружения базы. На этом этапе можно измерить использование памяти, процессора, подсистемы ввода/вывода, факты взаимных блокировок.
- Этап 2 Сужение мониторинга до отдельной области окружения, например, блокировок.
- Этап 3 Сужение мониторинга до отдельного объекта базы, например, отдельных таблиц и запросов к ним.
- Этап 4 Выявление отдельных неисправностей, например, отсутствие индексов.
- Этап 5 Внедрение решения.



Общие причины блокировоания:

- Неуместные длительные транзакции
- Неподходящие уровни изоляции
- Операции INSERT в последовательный кластеризованный индекс
- Транзакции, использующие неодинаковый порядок обращений к нескольким таблицам
- Неподходящее использование подсказок по блокировкам в запросах

В таблице ниже приведены характеристики различных уровней изоляции транзакций

Уровень изоляции	**Грязное чтение**	**Неповторяемое чтение**	**Фантомы**
Read uncommitted	Да	Да	Да
Read committed	Нет	Да	Да
Repeatable read	Нет	Нет	Да
Snapshot	Нет	Нет	Нет
Serializable	Нет	Нет	Нет

Типы ожиданий процессов.

Когда процесс на SQL Server пытается получить доступ к ресурсу, который недоступен, процесс помещается в список ожидания ресурса. Когда вы исследуете проблемы с производительностью и параллельной работой, список ожиданий даст четкое представление об ожиданиях. SQL Server 2017 предоставляет данные об ожиданиях через представление sys.dm_os_wait_stats. Ниже в таблице приведено описание некоторых столбцов

Столбец	**Описание**
wait_type	Имя типа ожидания.
waiting_tasks_count	Число ожиданий данного типа. Этот счетчик наращивается каждый раз при начале ожидания.

wait_time_ms	Общее время ожидания данного типа в миллисекундах.
max_wait_time_ms	Максимальное время ожидания данного типа.

Для мониторинга блокировок можно использовать несколько инструментов:

- Монитор активности и отчеты в SQL Server Management Studio;
- Системные хранимые процедуры: sp_who, sp_who2, sp_who3, sp_lock;
- Представления динамического управления sys.dm_os_waiting_tasks, sys.dm_os_wait_stats, sys.dm_tran_locks, sys.query_store_runtime_stats.
- Трассировку событий блокировок с помощью SQL Server Profiler
- Мониторинг расширенных событий.

Можно также использовать счетчик SQL Server: Блокировки: количество взаимоблокировок в секунду для выявления фактов взаимных блокировок. Обычно ненулевое значение обнаруживается при анализе журнала системного монитора.

Монитор активности. Монитор активности не позволяет накапливать информацию о блокировках, но позволяет оперативно вмешаться в ситуацию

Мониторинг расширенных событий. Мониторинг расширенных событий позволяет исследовать ожидания на блокировках и взаимоблокировки.

Системные хранимые процедуры.

Вызываемые в цикле системные хранимые процедуры sp_who, sp_who2, sp_who3, sp_lock также возвращают данные о текущих процессах и блокировках.

Представления динамического управления

Ниже приведен пример выбора данных из sys.dm_os_wait_stats

```
USE master

SELECT * FROM sys.dm_os_wait_stats

WHERE wait_type <>'LAZYWRITER_SLEEP' and wait_type <>'WAITFOR'

ORDER BY wait_time_ms DESC
```

Типы ожиданий нужно изучить по документации. Например,

LCK_M_U - задача ожидает получения блокировки на обновление

LCK_M_S - задача ожидает получения коллективной блокировки.

Трассировка. При использовании SQL Server Profiler для создания трассировки по взаимным блокировкам удобно использовать шаблон TSQL-Locks или события

- Errors and Warning: Blocked process report
- Locks: Deadlock graph
- Locks: Lock: Deadlock
- Locks: Lock: Timeout (Timeout > 0)
- Stored Procedures:RPC:Completed
- TSQL:SQL:BatchCompleted

- Session:Existing Connection
- Performance: Showplan XML Statistics Profile

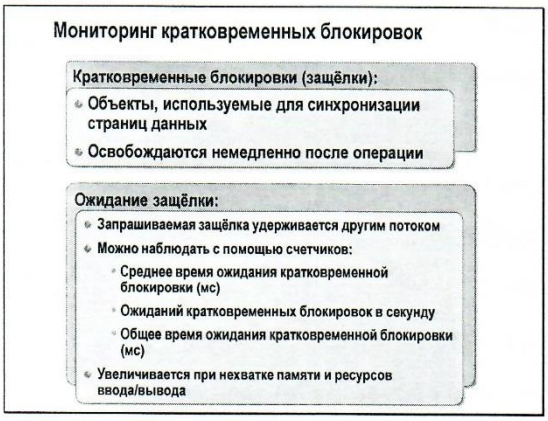
По указанным событиям необходимо исследовать данные в колонках: SPID, ObjectID, Mode, TextData, StartTime, BinaryData и т.д.

После завершения трассировки во время анализа результатов выберите событие Deadlock Graph, чтобы увидеть графическое представление события. Графическое представление можно через контекстное меню экспортировать в отдельный файл в формате XML, который можно просматривать в SQL Server Management Studio. Можно при настройке трассировки заранее указать файл для экспорта данных о взаимной блокировке в XML файл.

Если в конфигурации SQL Server установить нужное значение параметра BlockedProcessThreshold, можно в трассировке отслеживать ожидания блокировок, превысившие заданный порог. Для этого нужно выбрать событие трассировки Errors and Warnings: Blocked process report.

Независимо от используемых инструментов мониторинга нужно найти ID корневой сессии, вызвавшей проблему. После этого нужно отследить всю активность корневой сессии

После исследования блокировок средствами SQL Server или подбора индексов необходимо перейти в среду конфигуратора «1С:Предприятие 8» и там настроить индексирование. Для установления контекста блокировок в системе «1С:Предприятие 8» можно использовать технологический журнал. Сопоставление объектов системы «1С:Предприятие 8» и объектов базы на SQL Server можно получить из глобального контекста средствами языка программирования 1С



Кратковременные блокировки (защёлки)

Кратковременная блокировка - это облегченный объект синхронизации, используемый различными компонентами SQL Server. Кратковременные блокировки используются прежде всего для синхронизации страниц баз данных. Каждая кратковременная блокировка ассоциируется с одной единицей размещения. Ожидание кратковременной блокировки происходит в случаях, когда запрос на кратковременную блокировку не может быть удовлетворен немедленно, поскольку эта кратковременная блокировка удерживается другим потоком в конфликтующем режиме. В отличие от обычной блокировки, кратковременная блокировка высвобождается немедленно по завершении

операции, даже если это операция записи. Кратковременные блокировки группируются в классы по компонентам и по способам использования. В любой момент времени в том или ином экземпляре SQL Server может существовать ноль или большее число кратковременных блокировок определенного класса.

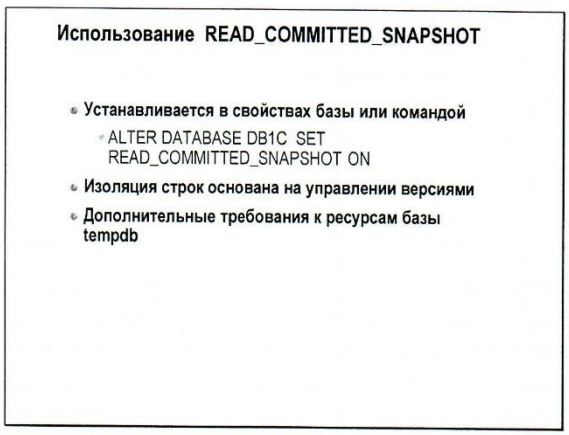
Как использовать защелки для устранения проблем с ожиданиями

SQL Server 2017 предоставляет три счетчика для измерения активности защелок:

- SQL Server: Latches: Average Latch Wait Time (ms.).- Среднее время ожидания кратковременной блокировки (мс)
- SQL Server: Latches: Latch Waits/sec. - Ожиданий кратковременных блокировок в секунду
- SQL Server: Latches: Total Latch Wait Time (ms.). - Общее время ожидания кратковременной блокировки за последнюю секунду

Повышение активности защелок часто говорит об одной из двух потенциальных проблем: нехватке памяти или ресурсов подсистемы ввода/вывода. Для поиска процесса, вызвавшего проблемы, нужно использовать следующие представления динамического управления:

- sys.dm_os_latch_stats – возвращает информацию об ожиданиях защелок по классам.
- sys.dm_os_wait_stats. – возвращает информацию об ожиданиях по потокам



Использование уровня изоляции строк, основанного на управлении версиями строк

Если параметр базы данных READ_COMMITTED_SNAPSHOT установлен в ON, то транзакция, запущенная с уровнем изоляции READ_COMMITTED, использует контроль версий строк вместо блокировки. Если транзакция выполняется с уровнем изоляции READ_COMMITTED, все инструкции видят моментальный снимок данных в состоянии, в котором он находился при запуске инструкции.

Применение этого уровня изоляции приводит к минимизации взаимоблокировок, возникающих между операциями считывания и записи.

Для установки параметра READ_COMMITTED_SNAPSHOT в состояние ON или OFF к базе данных не должно быть активных подключений, за исключением подключения, выполняющего команду ALTER DATABASE. Изменить состояние этого параметра невозможно, если база данных находится в режиме OFFLINE.

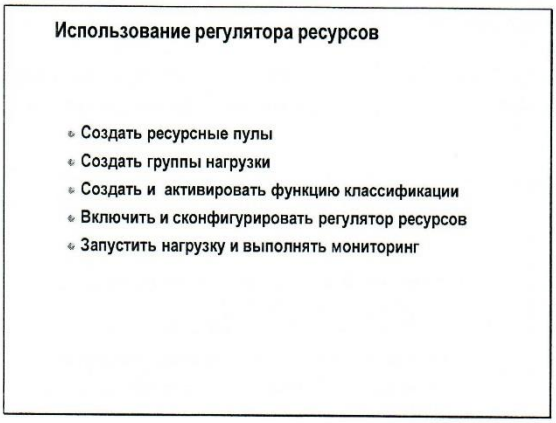
Параметр READ_COMMITTED_SNAPSHOT не может быть установлен в ON для системных баз данных master, tempdb или msdb. При изменении настройки для базы данных model эта настройка становится значением по умолчанию для любых вновь создаваемых баз данных, за исключением tempdb.

Текущее состояние этого параметра можно определить с помощью проверки значения столбца is_read_committed_snapshot_on в представлении каталога sys.databases.

При спользовании уровня изоляции строк, основанного на управлении версиями строк транзакция будет считывать версии строк даже после того, как другая транзакция изменила данные. Однако в отличие от транзакции моментального снимка, она:

- считывает измененные данные после того, как другая транзакция фиксирует изменения;
- может обновлять данные, измененные другой транзакцией.

Приложение «1С:Предприятие 8» устанавливает значание этого параметра в зависимости от режима совместимости конфигурации с целью уменьшения ожиданий на блокировках.



Регулятор ресурсов выполняет для каждой сессии классификацию на основе определяемой администратором функции. Сессия направляется в соответствующую группу нагрузки. Группа нагрузки использует соответствующий пул ресурсов. Пул ресурсов в случае конкуренции обеспечивает граничные значения ресурсов: минимальную и максимальную загрузку процессора и объем используемой оперативной памяти. Ниже приведен текст функции классификации, которая выделает приложение 1CV83 Server и направляет его в соответствующую группу нагрузки. В SQL Server есть возможность с помощью команд настроить регулятор ресурсов для управления ресурсами ввода-вывода.

```
USE master;

GO

CREATE RESOURCE POOL RestrictedIOPool WITH (MAX_IOPS_PER_VOLUME = 30, MIN_IOPS_PER_VOLUME =
1);

GO

USE master;

GO
```

```
CREATE WORKLOAD GROUP RestrictedIOGroup
USING RestrictedIOPool;
GO
USE master;
GO
CREATE FUNCTION dbo.RestrictedIO()
RETURNS SYSNAME WITH SCHEMABINDING
AS
BEGIN
    DECLARE @GroupName SYSNAME
    IF SUSER_NAME() = 'RestrictMyIO'
    BEGIN
        SET @GroupName = 'RestrictedIOGroup'
    END
    ELSE
    BEGIN
        SET @GroupName = 'default'
    END
    RETURN @GroupName;
END
GO
```

Задание 9. Мониторинг с помощью сеанса расширенных событий

Отслеживание планов длительных запросов в сеансе расширенных событий

В этом задании вы проанализируете планы выполнения длительных запросов

1. Настройте сеанс расширенных событий для сбора данных о **query_post_execution_showplan**
2. Отметьте галочками глобальные поля **client_app_name** и **sql_text**
3. Для события **query_post_execution_showplan** установите фильтр на поле **duration greater_then_equal_uint64 1000.** Для уменьшения объема собираемой информации можно увеличить значение порога фильтра в зависимости от ваших интересов. Порог задается в микросекундах.
4. Для события **query_post_execution_showplan** установите фильтр **sqlserver.client_app_name like_i_sql_unicode_string 1CV83 Server**

5. Объедините фильтры условием **AND** (**И**)
6. В качестве целевого объекта используйте файл
7. Получите и сохраните скрипт сеанса
8. Запустите сеанс
9. В приложении «1С: Предприятие 8» выполните интересующие вас операции
10. Остановите сеанс.
11. Откройте файл расширенных событий
12. Включите отображение поля **duration** в верхней части таблицы.
13. Отсортируйте события по полю **duration**
14. Найдите самое длительное событие
15. Исследуйте план запроса и поле **sql_text**.

Раздел 6: Автоматизация задач обслуживания базы данных

Существует множество ежедневных административных задач, выполнение которых необходимо для управления базами данных. Автоматизация этих задач способствует уменьшению административных нагрузок, связанных с управлением базами данных, а также обнаружению и устранению неполадок, прежде чем последние смогут повлиять на доступность базы данных. В этом разделе подробно описано выполнение ежедневных и конфигурационных задач. А также указывается, как автоматизировать задачи путем создания заданий, операторов и предупреждений.

Цели

Изучив данный раздел, вы сможете:

• Определять задачи администрирования Microsoft SQL Server и планировать автоматическое выполнение этих задач.

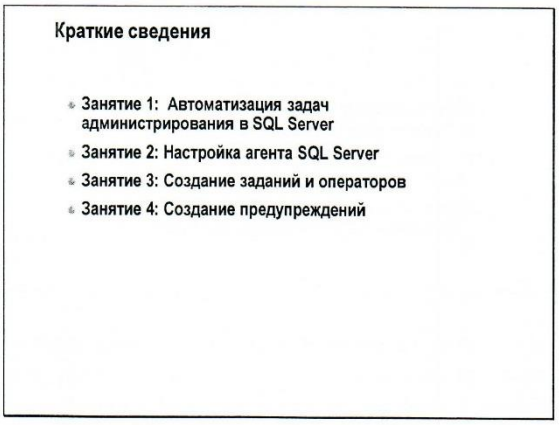
• Настраивать агент SQL Server для поддержки автоматического планирования задач.

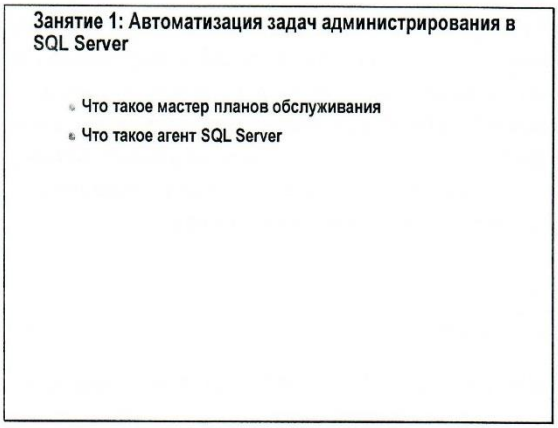
• Создавать сценарии для задач с использованием заданий SQL Server и определять операторов для управления этими заданиями.

• Определять оповещения для уведомления операторов о тех или иных событиях SQL Server.

• Определять задачи администрирования для нескольких серверов и управлять такими задачами.

• Настраивать параметры безопасности агента SQL Server.





Одной из основных функций администратора базы данных является обслуживание сервера SQL Server и его баз данных. Работа администратора обычно связана с выполнением различных ежедневных административных задач. Можно автоматизировать эти повседневные задачи и настроить SQL Server на заблаговременное отслеживание определенных типов проблем.

Можно использовать мастер планов обслуживания для планирования задач обслуживания, что гарантирует регулярное создание резервных копий баз данных, хорошую производительность баз данных и проверку на несогласованность. С помощью мастера планов обслуживания можно создать несколько заданий агента SQL Server Agent, которые автоматически будут выполнять эти задачи обслуживания через заданные промежутки времени.

Использование мастера планов обслуживания для автоматизации задач

Можно запланировать автоматическое выполнение нескольких задач обслуживания, включая следующие:

• Резервное копирование базы данных и файлов журнала транзакций. Резервные копии базы данных и журналов могут храниться в течение указанного периода времени.

• Выполнение заданий агента SQL Server, ответственных за различные действия.

• Сжатие файлов данных путем удаления пустых страниц баз данных.

• Выполнение проверок внутренней несогласованности данных и страниц данных в базе данных для поиска неполадок программного обеспечения и поврежденных данных.

• Реорганизация сведений на страницах данных и индексов путем перестроения индексов.

• Обновление статистики индексов для получения оптимизатором запросов самых последних сведений о распределении значений данных в таблицах.

Результаты выполнения задач обслуживания могут быть записаны в виде отчета в текстовый файл или занесены в таблицы плана обслуживания — **sysmaintplan_log** и **sysmaintplan_log_detail** — в базе данных **msdb**.

Запуск мастера планов обслуживания

Мастер планов обслуживания можно запустить в среде SQL Server Management Studio. В обозревателе объектов разверните сервер и папку управления, щелкните правой кнопкой мыши «Планы обслуживания» и выберите «Мастер планов обслуживания». Затем можно использовать этот мастер для создания плана, предназначенного для определенных требований обслуживания.

Создание и изменение планов обслуживания вручную

Можно изменить существующий план обслуживания с помощью конструктора планов обслуживания, который также доступен в среде SQL Server Management Studio. С помощью этого инструмента с графическим интерфейсом пользователя можно упорядочить задачи, добавить новые задачи и организовать простой поток операций, который будет определять способы обработки задач, завершившихся успешно или со сбоем.

Агент SQL Server — это компонент SQL Server, ответственный за автоматизацию административных задач SQL Server. Чтобы агент SQL Server мог выполнять задачи и отображать предупреждения, он должен постоянно работать и обладать достаточными разрешениями.

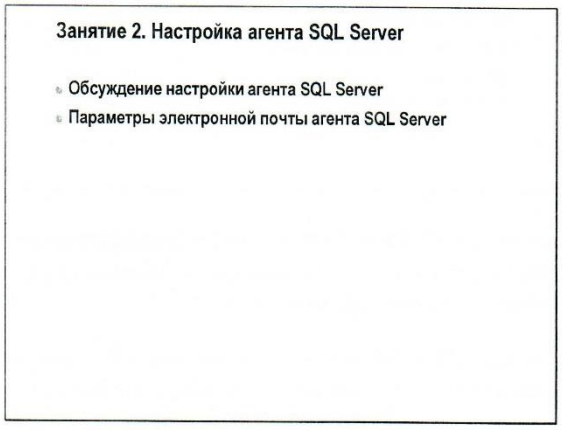
Необходимо настроить службу «SQL Server агент» на автоматический запуск при загрузке Windows Server. Кроме того, с помощью диспетчера конфигурации SQL Server можно настроить службу агента SQL

Server на автоматический перезапуск при неожиданной остановке. Чтобы происходил автоматический перезапуск, учетная запись службы агента SQL Server должна быть членом локальной группы администраторов.

Задание 10. Создание плана обслуживания базы данных 1С

1. Запустите на своей машине **SQL Server Management Studio** зарегистрируйте **Database Engine** локального сервера, используя проверку подлинности Windows
2. Перейдите в контейнер **Управление**, далее в **Планы обслуживания**.
3. В контекстном меню запустите **Мастер планов обслуживания.**
4. На шаге **Выбор свойств плана** щелкните на кнопке **Изменить** и создайте расписание для выполнения каждую ночь
5. На шаге Выбор **задач по обслуживанию** отметьте следующие задачи:
    - **Проверка целостности базы данных**
    - **Восстановить индекс**
    - **Обновить статистику**
6. На шаге **Выбор порядка задач по обслуживанию** установите следующий порядок
    - **Восстановить индекс**
    - **Обновить статистику**
    - **Проверка целостности базы данных**
7. На шаге Задача **«Перестроение индексов»** установите следующие параметры.
    - **Базы данных:** В выпадающем списке выберите **DB1C**
    - **Объект: Таблицы и представления**
    - **Сохранять индекс в состоянии «в сети» в процессе переиндексирования**
8. На шаге Задача **«Обновление статистики»** установите следующие параметры.
    - **Базы данных:** В выпадающем списке выберите **DB1C**
    - **Объект: Таблицы и представления**
    - **Вся собранная статистика**
9. На шаге Задача **«Проверка целостности базы данных»** установите следующие параметры.
    - **Базы данных:** В выпадающем списке выберите **DB1C**
    - **Включить индексы**
10. В оставшихся шагах мастера планов обслуживания оставьте параметры по умолчанию

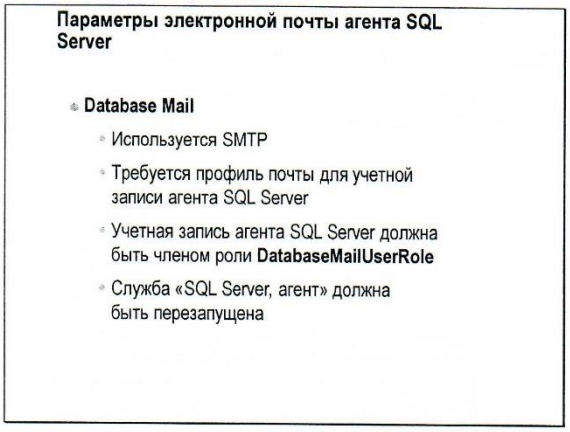
Откройте план с помощью команды **Изменить**. Удалите задачу **«Восстановление индексов».** Вместо нее добавьте задачу **«Выполнение инструкции T-SQL».** В качестве команд сценария используйте сохраненный ранее сценарий DBReindex.sql. Удалите задачу **«Обновление статистики».** Вместо нее добавьте задачу **«Выполнение инструкции T-SQL».** В качестве команд сценария используйте **Exec sp_updatestats**



Необходимо корректно настроить агент SQL Server, чтобы он поддерживал автоматические задачи обслуживания базы данных. В этом занятии описываются параметры настройки агента SQL Server. Агент SQL Server работает в качестве службы Windows. Каждый экземпляр SQL Server имеет собственную службу агента SQL Server, которая называется **SQL Server агент (SQLServerAgent)** для экземпляров по умолчанию и **SQLAgent$<название_экземпляра>** для именованных экземпляров.

Конфигурация службы «SQL Server агент» при загрузке

Как и любая другая служба Windows, служба «SQL Server агент» может быть настроена на автоматический и ручной запуски или может быть отключена. По умолчанию служба «SQL Server агент» запускается вручную при установке SQL Server. Можно изменить конфигурацию загрузки службы «SQL Server агент» с помощью диспетчера конфигурации или административного средства «Службы» в Windows. Если планируется внедрение автоматизированных задач и предупреждений, как правило, необходимо настроить службу агента SQL Server на автоматический запуск при загрузке Windows, если нет необходимости в ручном запуске этой службы. Служба «SQL Server агент» зависит от службы SQL Server того экземпляра, которому она принадлежит.



Использование компонента Database Mail вместе с агентом SQL Server

Компонент Database Mail — это функциональная возможность SQL Server 2017, которая позволяет SQL Server отправлять электронную почту через SMTP-сервер. Чтобы использовать компонент Database Mail вместе с агентом SQL Server, необходимо выполнить следующие задачи по настройке:

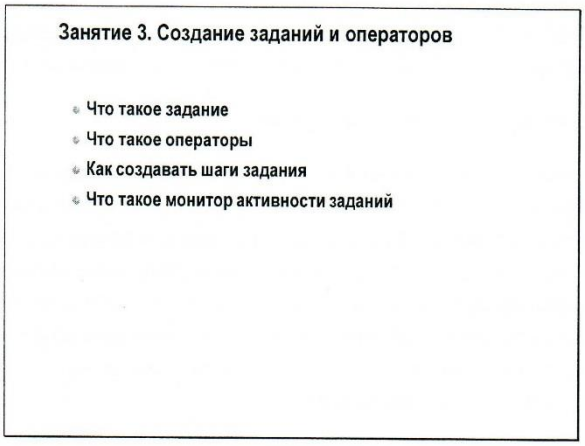
1. Добавьте пользователя для имени входа службы SQLServerAgent в роль базы данных **DatabaseMailUserRole** в базе данных **msdb**.

2. Включите компонент Database Mail.

3. Создайте профиль Database Mail, который содержит учетную запись электронной почты для использования агентом SQL Server. Создайте этот профиль по умолчанию для пользователя в базе данных **msdb**, сопоставленного имени входа для учетной записи службы SQLServerAgent.

4. Настройте свойства системы предупреждений агента SQL Server на использование компонента Database Mail и укажите профиль, созданный ранее.

5. Перезапустите службу SQLServerAgent. При настроенном компоненте Database Mail можно сделать отправку ответов по электронной почте одним из заданий агента SQL Server.



На этом занятии объясняется, как создавать задания и операторы.

Задание — это определенная последовательность действий, выполняемая агентом SQL Server. В задании могут быть выполнены различные действия, включая запуск сценариев Transact-SQL, приложения командной строки, сценарии и команды PowerShell, пакеты служб Integration Services, команды и запросы служб Analysis Services и задачи репликации. Задания могут выполнять повторяющиеся или запланированные задачи, а также уведомлять определенных пользователей (называемых операторами) о состоянии задания путем создания предупреждений, этим самым, упрощая администрирование SQL Server. Можно выполнять задания вручную или настроить их на автоматический запуск в соответствии с расписанием или в ответ на предупреждения.

Создание заданий

Чтобы определить новое задание, можно использовать среду SQL Server Management Studio или выполнить системную хранимую процедуру **sp_add_job**. Определение задания хранится в системной

таблице **sysjobs** в базе данных **msdb.** Эта таблица удерживается в кэше для улучшения производительности. При определении заданий необходимо выполнить следующее:

• Убедиться, что задание включено. Задания включены по умолчанию. Если задание отключено, его невозможно запустить по расписанию. Однако пользователь может выполнить отключенное задание вручную, запустив его в среде SQL Server Management Studio.

• Указать владельца, ответственного за выполнение задания. По умолчанию владельцем является учетная запись пользователя Windows или SQL Server, с помощью которой было создано задание.

• Определить, выполняется ли задание на локальном сервере или на нескольких удаленных серверах.

• Создать категории заданий для организации, фильтрации и управления несколькими заданиями. Например, можно создать категории заданий, которые будут соответствовать подразделениям организации.

Операторы — это псевдонимы пользователей или групп, которые получают электронные уведомления о завершении заданий. При завершении задания или сбое на каком-либо этапе задания, можно уведомить оператора по пейджеру, электронной почте или с помощью команды **net send**.

Рекомендации по созданию операторов

Чтобы создать нового оператора, можно использовать среду SQL Server Management Studio или выполнить системную хранимую процедуру **sp_add_operator** Определение оператора хранится в системной таблице **sysoperators** в базе данных **msdb.** При создании операторов необходимо выполнить следующее:

• Использовать псевдоним электронной почты группы для уведомления нескольких лиц о возможных неполадках.

• Проверить каждый метод уведомления, используемый для оповещения оператора, чтобы убедиться в возможности оператора получать сообщения.

• Указать рабочее расписание для каждого оператора, которого следует уведомлять по пейджеру. Уведомление не будет работать, если расписание уведомлений оператора по пейджеру не согласуется с расписанием самого оператора.

Определение резервного оператора

Можно определить резервного оператора, который будет отвечать на предупреждение, если отправление уведомлений указанным операторам завершается сбоем. Например, если все операторы недоступны при появлении предупреждения, будет произведен вызов резервного оператора.

Резервный оператор уведомляется в следующих случаях:

• Предупреждение содержит уведомления на пейджер, заданные на ответ.

• Никто из операторов, которых необходимо уведомить по пейджеру, не находится на службе.

• Определен резервный оператор.

При назначении резервного оператора учтите следующее:

• Сведения о резервном операторе хранятся в кэше, поэтому они не зависят от подключения к базе данных **msdb**.

• Можно создать только одного резервного оператора.

• Невозможно удалить оператора, который был назначен в качестве резервного. Однако можно удалить назначение резервного оператора, а потом удалить самого оператора.

Чтобы определить шаг задания, можно использовать среду SQL Server Management Studio или выполнить системную хранимую процедуру **sp_add_jobstep**. Определения шагов задания хранятся в системной таблице **sysjobsteps** в базе данных **msdb**. Можно определить шаги задания для выполнения инструкций Transact-SQL, системных команд, сценариев ActiveX или задач репликации SQL Server. Однако можно указать только один тип запуска для каждого шага задания.

Определение шагов задания, содержащих Transact-SQL

При определении шагов задания, на которых будут выполняться инструкции Transact-SQL, хранимые процедуры или расширенные хранимые процедуры, учтите следующие рекомендации:

• Необходимо указать используемую базу данных.

• Необходимо указать требуемые для этого шага задания переменные и параметры.

• Можно отправить результирующий набор шага задания в выходной файл. Выходные файлы часто используются при устранении неполадок для просмотра сообщений об ошибках, которые могут произойти во время выполнения процедуры. Выходной файл шага работы невозможно использовать в качестве входного файла для следующего шага.

Определение шагов задания, содержащих системные команды

При определении шага задания для выполнения системных команд или команд приложения (определяемых по расширениям файла EXE, BAT, CMD или COM) необходимо выполнить следующее:

• Определить код выхода процесса, указывающего, что команда была выполнена успешно.

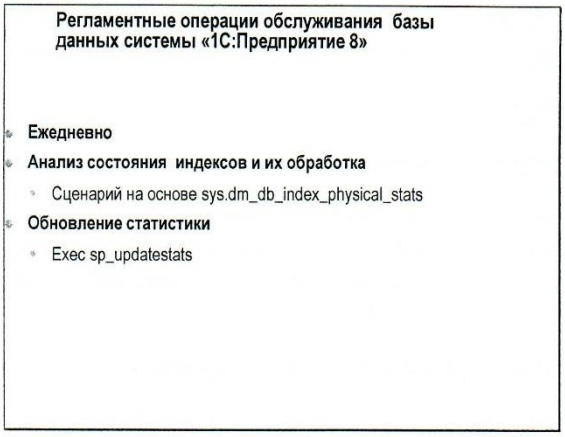
• Включить полный путь к выполняемому приложению. Этот путь требуется агентом SQL Server для нахождения источника приложения.

Определение шагов задания, содержащих сценарии PowerShell

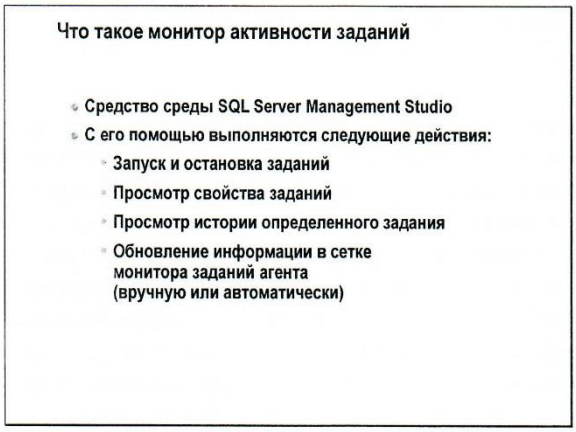
Можно создавать шаги задания, основанные на сценариях PowerShell.

Поток шагов задания

При создании заданий необходимо указать действие, которое должно быть выполнено SQL Server при успешном или неуспешном выполнении каждого шага задания. По умолчанию SQL Server переходит к новому шагу задания при каждом успешном выполнении шага задания и останавливается при завершении выполнения шага задания со сбоем. Однако можно определить любой шаг задания, на который будет выполнен переход при каждом успешном или неуспешном завершении шага задания. Можно указать количество попыток повтора выполнения шага задания при сбое, которое будет предпринимать SQL Server. Также можно указать интервалы повтора (в минутах). Например, если для выполнения шага задания требуется подключение к удаленному серверу, можно определить несколько повторных попыток на тот случай, если подключение завершится со сбоем. Кроме того, если задание должно быть выполнено только один раз, можно указать необходимость удаления задания после его завершения.



Для обслуживания базы данных системы «1С:Предприятие 8» необходимо создать задание, которое выполняет задачи, приведенные на рисунке, и настроить для него соответствующее расписание. Для анализа состояния индексов и их обработки можно выполнять готовый сценарий примера использования DMF sys.dm_db_index_physical_stats из электронной документации на сайте MSDN. Системная хранимая процедура sp_updatestats обновляет только ту статистику, которая по внутренним оценкам сервера считается устаревшей.



Монитор активности заданий — это средство, используемое в среде SQL Server Management Studio, которое позволяет просматривать сведения в таблице **sysjobactivity** в виде диаграмм. Можно просматривать все задания на сервере или определить фильтры для ограничения количества отображаемых заданий. Также можно упорядочить сведения о заданиях, щелкнув заголовок столбца в сетке **Активность заданий агента**. Например, если выбран заголовок столбца **Последний запуск**, можно просмотреть задания в порядке их последнего запуска. Повторный щелчок заголовка столбца приведет к включению упорядочивания заданий по возрастающему или убывающему принципу в зависимости от даты последнего запуска.

Когда следует использовать монитор активности заданий

Монитор активности заданий следует использовать при необходимости определения заданий, которые должны быть выполнены; получения выходных данных заданий, которые выполнялись в течение текущего сеанса; и получения сведений о работающих или бездействующих заданиях. Если служба «SQL Server, агент» неожиданно завершается со сбоем, можно определить, какое задание выполнялось, просмотрев предыдущий сеанс в мониторе активности заданий. С помощью монитора активности заданий можно выполнить следующие задачи:

• Запуск и остановка заданий.

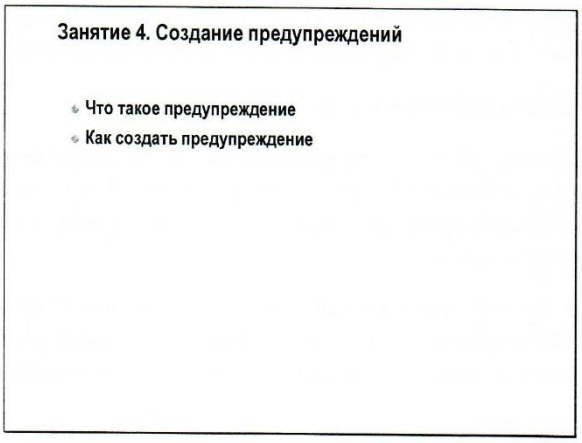
• Просмотр свойства заданий.

• Просмотр истории определенного задания.

• Обновите сведения в сетке монитора заданий агента вручную или настройте интервал автоматического обновления с помощью параметра

Просмотреть настройки обновления.

Чтобы открыть монитор активности заданий в среде SQL Server Management Studio, откройте обозреватель объектов, разверните узел агента SQL Server, правой кнопкой мыши щелкните пункт «Задания» и выберите «Просмотр активности заданий». Также можно просмотреть активность заданий для текущего сеанса с помощью хранимой процедуры **sp_help_jobactivity** в базе данных **msdb**.



SQL Server позволяет создавать предупреждения в ответ на ошибки SQL Server, ошибки, определенные пользователем, или условия производительности. Также можно создать резервного оператора на тот случай, если уведомление по пейджеру не достигло оператора. В этом занятии описываются параметры конфигурации для предупреждений в SQL Server.

Предупреждения — это предварительно определенные ответы на события, которые могут произойти в решении SQL Server. Можно настроить предупреждения, которые будут выполнять задание или уведомлять оператора при возникновении определенного события или превышении порогового значения производительности. События создаются в SQL Server и записываются в журнал приложений Windows. Агент SQL Server считывает журнал приложений и сравнивает зарегистрированные события с определенными предупреждениями. Если агент SQL Server обнаружит совпадение, вызывается предупреждение, которое является автоматическим ответом на событие. Кроме наблюдения за событиями агент SQL Server может отслеживать условия производительности и события WMI.

Определение предупреждений для ошибок SQL Server

При создании предупреждения для ответного действия при ошибке SQL Server можно указать отдельный номер ошибки, например 9002 или все ошибки определенного уровня критичности, например 17. Можно определить оповещение по номеру ошибки или уровню критичности для всех баз данных или для определенной базы.

Примечание. Для настройки записи возникающих при работе системы «1С:Предприятие:8» событий с номерами ошибок 1205 (взаимоблокировка) и 1222 (превышение времени ожидания) в журнал приложений Windows необходимо выполнить хранимую процедуру **sp_altermessage,** например,

EXEC sp_altermessage 1205, 'with_log', true

Определение предупреждений для условий производительности

Кроме использования предупреждений для ответа на ошибки SQL Server, можно использовать их для ответа на условия производительности SQL Server, например на те, которые можно просматривать с помощью системного монитора Windows. Если значение условия превышено, отображается предупреждение. Например, можно создать предупреждение при условии производительности, которое будет отображаться при превышении журналом транзакций в базе данных **DB1C** 75% объема. Ответом на предупреждение может служить выполнение задания по резервному копированию журнала транзакций и уведомление администратора базы данных.

Примечание. Отработка предупреждения по условиям производительности не зависит от запуска системного монитора. Данные о производительности записываются периодически (несколько раз в

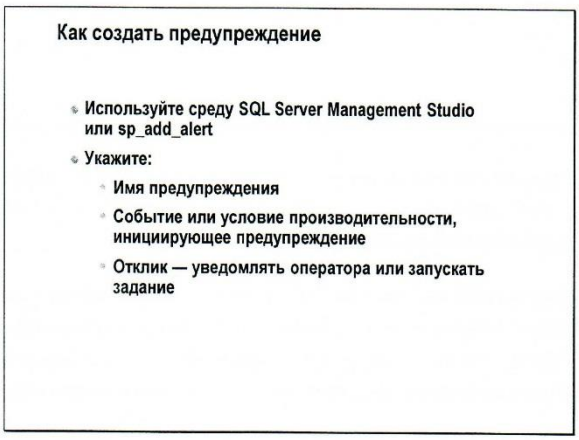
минуту), что может привести к задержкам, длящимся до нескольких секунд, между достижением порогового значения и отображением предупреждения. Поэтому может понадобиться уменьшить время задержки между ответами или изменить пороговое значение для условия производительности, если необходимо, чтобы ответ на предупреждение привел к быстрому изменению условия.

Определение предупреждений для событий WMI

Можно указать, что предупреждение должно отображаться в качестве ответа на определенное событие WMI. При определении предупреждения для события WMI агент SQL Server выполняет следующие задачи:

• Регистрируется в качестве клиента WMI в пространстве имен WMI, предоставленном для запроса событий.

• Выполняет инструкцию WQL, указанную для идентификации определенного события.



Можно создать предупреждение с помощью среды SQL Server Management Studio или хранимой процедуры sp_add_alert. Чтобы создать предупреждение, следует указать:

• Имя предупреждения.

• Событие или условие производительности, приводящее предупреждение в действие.

• Действие, выполняемое агентом SQL Server в ответ на событие или условие производительности.

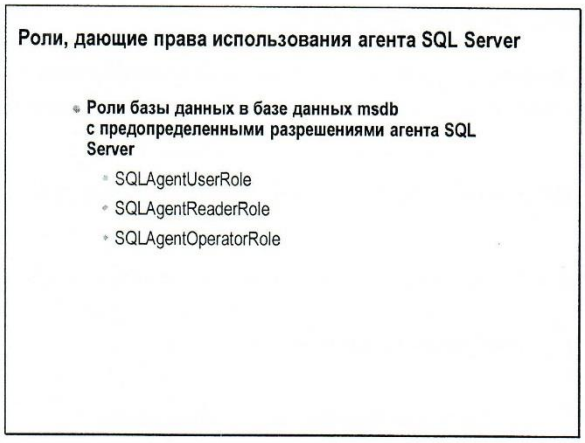
Тип события определяет параметры, которые будут использоваться для указания конкретного события.

Уведомление оператора

Действие, предпринимаемое агентом SQL Server в ответ на событие или условие производительности, может включать и уведомление оператора. Чтобы связаться с оператором, необходимо указать контактные сведения этого оператора и определить тип уведомления. Операторов можно уведомлять по электронной почте.

Выполнение задания

Действие, предпринимаемое агентом SQL Server в ответ на событие или условие производительности, может включать и выполнение задания. Чтобы предупреждение привело к запуску задания, необходимо указать имя этого задания на странице ответов выбранного предупреждения. При определении предупреждений можно использовать текущее задание или создать новое.



SQL Server содержит следующие фиксированные роли базы данных в базе данных msdb для предоставления администраторам более точного средства управления доступом к агенту SQL Server: **SQLAgentUserRole, SQLAgentReaderRole, SQLAgentOperatorRole**

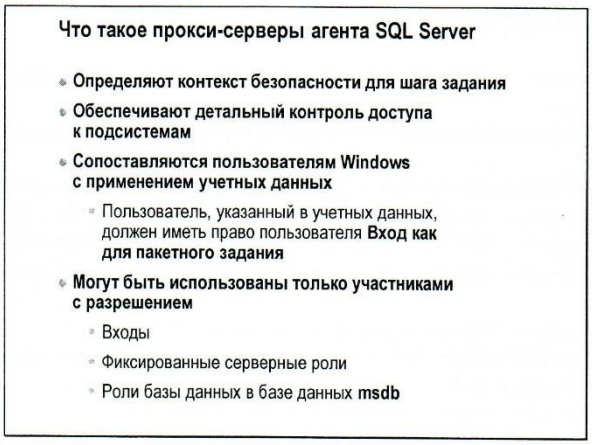
Если пользователи, не являющиеся членами одной из этих ролей, пытаются подключиться к SQL Server в среде SQL Server Management Studio, узел агента SQL Server в обозревателе объектов не отображается. Чтобы использовать агент SQL Server, пользователь должен являться членом одной из этих фиксированных ролей или быть членом фиксированной роли сервера sysadmin.

SQLAgentUserRole — это наименее привилегированная из всех фиксированных ролей базы данных агента SQL Server. Члены роли **SQLAgentUserRole** имеют разрешения только на локальные задания и на расписания заданий, которыми они владеют. Они не могут изменять владельца задания для получения доступа к заданиям, которыми они не владеют. Члены роли **SQLAgentUserRole** могут просматривать список доступных учетных записей-посредников в диалоговом окне «Свойства шага задания» среды SQL Server Management Studio.

SQLAgentReaderRole включает все разрешения **SQLAgentUserRole**. Члены этой роли также могут просматривать список всех доступных заданий, расписания заданий и их свойства, а не только задания и расписания заданий, которыми владеют. Члены роли **SQLAgentReaderRole** не могут изменять владельца задания для получения доступа к заданиям, владельцами которых они не являются.

SQLAgentOperatorRole — это наиболее привилегированная из всех фиксированных ролей базы данных агента SQL Server. Она включает в себя все разрешения ролей **SQLAgentUserRole** и **SQLAgentReaderRole**. Члены этой роли также могут просматривать свойства операторов и учетных записей-посредников, перечислять доступные учетные записи-посредники и предупреждения на сервере. Члены роли **SQLAgentOperatorRole** имеют дополнительные разрешения для локальных заданий и расписаний. Они могут выполнять, останавливать или запускать все локальные задания, а также удалять журнал любого локального задания на сервере. Также они могут включать или отключать все локальные задания и расписания на сервере. Чтобы включить или отключить локальные задания или расписания, члены этой

роли должны использовать хранимые процедуры **sp_update_job** и **sp_update_schedule**, указывая параметр имени задания или идентификатора расписания и параметр **enabled**. Если указаны другие параметры, выполнение этих хранимых процедур заканчивается со сбоем. Члены роли **SQLAgentOperatorRole** не могут изменять владельца задания для получения доступа к заданиям, владельцами которых они не являются.



Учетная запись-посредник агента SQL Server определяет контекст безопасности для шага задания, не содержащего Transact-SQL, который имеет доступ к определенной подсистеме (например, сценарии PowerShell). Учетная запись-посредник предоставляет агенту SQL Server доступ к учетным данным безопасности пользователя Windows. Каждая учетная запись-посредник может быть сопоставлена с несколькими подсистемами. На шаге задания, на котором используется учетная запись-посредник, можно получить доступ к указанным подсистемам с помощью контекста безопасности пользователя Windows. Перед выполнением агентом SQL Server шага задания, на котором используется учетная запись-посредник, агент SQL Server олицетворяет учетные данные, определенные в учетной записи-посреднике, а затем выполняет шаг задания с помощью контекста безопасности.

Использование прокси-серверов агента SQL Server

Прокси-серверы агента SQL Server используют учетные данные для хранения сведений об учетных записях пользователей Windows. Пользователь, указанный в учетных данных, должен иметь разрешение **Вход как для пакетного задания** на компьютере, где запущен SQL Server. Агент SQL Server проверяет доступ к подсистеме для прокси-серверов и предоставляет доступ каждый раз при выполнении шага задания. Если прокси-сервер больше не получает доступа к подсистеме, шаг задания заканчивается со сбоем. В противном случае, агент SQL Server олицетворяет пользователя, указанного в прокси-сервере, и выполняет шаг задания. Создание прокси-серверов не приводит к изменению разрешений пользователя, указанных в учетных данных. Доступ может быть предоставлен трем типам участников безопасности:

- Имена входа SQL Server
- Фиксированные серверные роли
- Роли базы данных в базе данных **msdb**

Если имени входа пользователя предоставлен доступ к прокси-серверу или пользователь принадлежит любой роли, имеющей право доступа к прокси-серверу, пользователь может использовать прокси-сервер на шаге задания.

Задание 11. Создание заданий и предупреждений

Создайте задание, которое по расписанию вызывает созданный ранее сценарий обслуживания индексов DBReindex.sql в базе DB1C и обновляет статистику. Шаги задания:

1. Выполнение сценария **DBReindex.sql** для анализа и обработки индексов
2. Exec sp_updatestats

Создайте профиль электронной почты Database Mail для использования службой Агент SQL Server

1. В контейнере Безопасность сервера в свойствах имени входа, соответствующего учетной записи Windows для SQL Server настройте сопоставление с базой **msdb** и роль **DatabaseMailUser.**
2. Настройте профиль электронной почты Database Mail для пользователя базы **msdb,** созданного на предыдущем шаге, с использованием доступного вам сервера SMTP и адреса электронной почты.
3. Отправьте тестовое сообщение и проверьте его получение
4. В свойствах **Агент SQL Server** в разделе **Предупреждения** выберите созданный профиль и перезапустите службу агента.

Создайте оператор для получения оповещений

1. Перейдите к контейнеру **Операторы**
2. Создайте оператор с именем **SQLAdmin**.
3. В поле **Адрес электронной почты** укажите используемый вами SMTP адрес электронной почты
4. Используйте этот оператор для уведомления в задании, созданном ранее и в задании с вложенным планом обслуживания базы.
5. Выполните имеющиеся задания, проверьте историю выполнения и получение уведомлений оператором

Создайте предупреждение о производительности SQL Server

1. Запустите на своей машине **SQL Server Management Studio** зарегистрируйте **Database Engine** локального сервера, используя проверку подлинности Windows
2. Перейдите в базу данных **DB1C**.
3. Выполните запрос, содержащий команду **DBCC sqlperf (logspace)**
4. Запомните процент заполнения журнала в базе **DB1C**
5. Перейдите к контейнеру **Предупреждения.**
6. Создайте новое предупреждение с именем **PerformanceAlert**.
7. В выпадающем списке **Тип**: выберите **Предупреждение о производительности SQL Server**
8. В выпадающем списке **Объект:** выберите **databases**
9. В выпадающем списке **Счетчик:** выберите **PercentLogUsed**
10. В выпадающем списке **Экземпляр**: выберите **DB1C**
11. В выпадающем списке **Создать предупреждение, если счетчик**: выберите **больше**.
12. В поле **Значение** установите целое значение меньше значения, полученного в отчете.
13. В группе **Ответ** отметьте оператора **SQLAdmin** для отправки ему сообщения с помощью электронной почты.
14. В группе **Параметры** можете указать дополнительное сообщение для уведомления и настроить задержку между ответами
15. Проверьте в свойствах предупреждения PerformanceAlert журнал.

Раздел 7: Поддержание высокой доступности данных

Многие системы управления базами данных являются крайне важными для обеспечения нормальной деятельности компаний. Если система становится недоступной, пользователи, возможно, не смогут выполнять свои функции. Нарушение электропитания, отказы системы, неполадки сети и даже применение пакетов обновления могут быть причинами прерывания обслуживания. Требования по обеспечению доступности различаются для разных систем. Некоторые системы должны обеспечивать непрерывный доступ, тогда как для других систем существуют периоды в течение дня или недели, когда систему можно переключить в автономный режим. Допустимое время восстановления системы также варьируется в зависимости от обстоятельств. Хорошо разработанная стратегия резервного копирования необходима для любой системы с высоким уровнем доступности, и в некоторых случаях системой может предоставляться подходящее встроенное решение, однако в этом модуле рассматриваются методы, которые обеспечивают более быстрое и более автоматизированное решение.

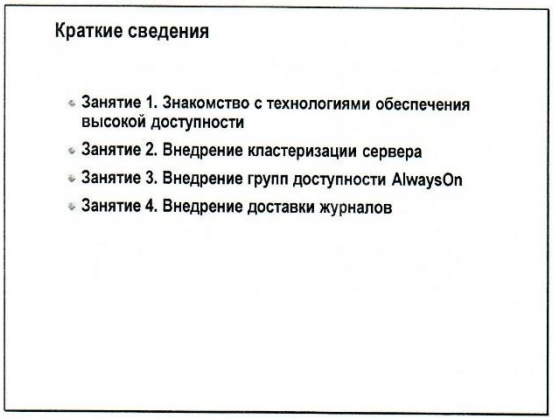
Цели

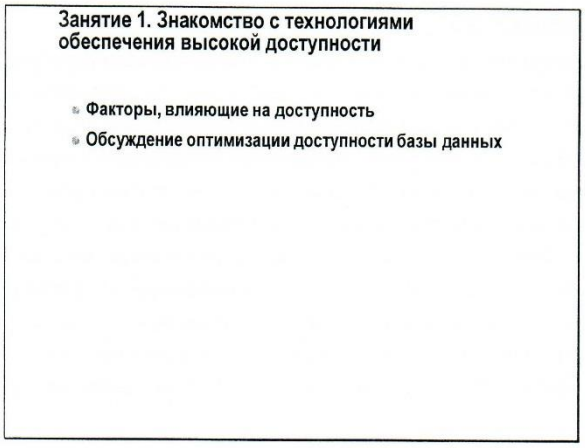
Изучив данный раздел, вы сможете:

• объяснить, как внедрить кластеризацию для поддержки быстрого перехода на другой ресурс при сбое компьютеров, на которых работают экземпляры Microsoft SQL Server;

• объяснить, как внедрить группы доступности AlwaysOn;

• объяснить, как внедрить доставку журналов для поддержки быстрого аварийного восстановления базы данных SQL Server в режиме резервирования.





На этом занятии рассматриваются факторы, влияющие на доступность, и обсуждаются вопросы оптимизации доступности баз данных. На доступность оказывают влияние различные факторы, такие как ошибки программного обеспечения, неисправность оборудования, неполадки сети, отсутствие электропитания и природные катаклизмы. Противостоять этим проблемам могут разнообразные решения, такие как резервирование компонентов оборудования, сети, серверных баз данных и массивы RAID.

Резервирование помогает устранить большинство факторов, негативно влияющих на доступность. Резервирование может применяться для дублирования баз данных, оборудования, сетевых компонентов, целых серверов или даже всего узла.

Резервирование компонентов оборудования

Для улучшения доступности может дублироваться большинство компонентов современных серверов. Резервные источники питания, вентиляторы, блоки памяти и сетевые интерфейсные платы служат для предоставления вспомогательных компонентов в случае отказа.

Резервирование сети

Несколько сетевых интерфейсных плат можно присоединить к разным подсетям, обеспечив резервирование при сбое одной из подсетей. Сетевые интерфейсные платы можно объединять с помощью программного обеспечения, предоставляемого поставщиками плат, используя, так называемую процедуру *группирования сетевых интерфейсных плат*. Каждой сетевой интерфейсной платой используется общий виртуальный IP-адрес, и, когда работают все платы, полоса пропускания увеличивается.

Технология RAID

Массивы независимых дисков с избыточностью (RAID) — это решение для одиночного сервера, предоставляющее резервирование жестких дисков и обеспечивающее улучшение производительности системы дисковой памяти. RAID-массивы могут быть программным решением Microsoft Windows или аппаратным решением с аппаратными реализациями, предоставляющими улучшенную производительность и защиту данных, но по более высокой цене. Наиболее популярными разновидностями RAID-массивов для обеспечения доступности данных являются RAID 1 (зеркальные диски), RAID 5 (чередование дисков с распределенной четностью) и RAID 10 (зеркальные диски с чередованием, также известные как RAID 1+0).

Резервирование серверов и баз данных

Приложением SQL Server 2017 предоставляется несколько вариантов обеспечения высокого уровня доступности для серверов и баз данных. Возможные варианты достижения высокого уровня доступности:

• **Кластеризация сервера.** Кластеризация предоставляет решение, обеспечивающее высокий уровень доступности в масштабе сервера. В случае отказа операционная система и службы действуют совместно, чтобы выполнить автоматический переход на другой ресурс менее чем за одну минуту. При отказоустойчивой кластеризации ни на сервере, ни на клиентах не требуется ручное вмешательство во время перехода на другой ресурс.

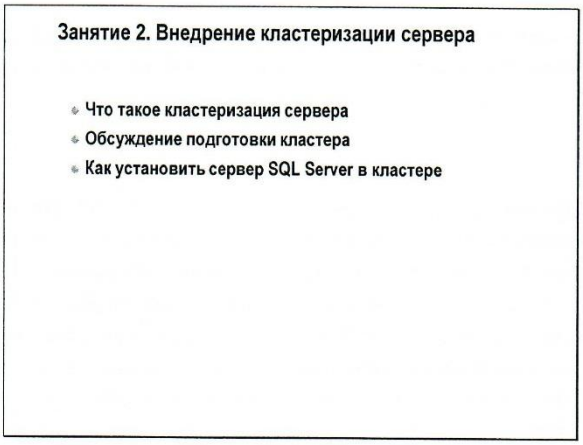
• **Группы доступности AlwaysOn.** Группы доступности объединяют достоинства других методов поддержания высокой доступности. Для синхронизации баз используется механизм зеркальных сеансов с несколькими вторичными репликами, среди которых могут быть реплики для чтения. Экземпляры-участники могут устанавливаться как на узлах кластера отработки отказа Windows, так и на узлах, не входящих в кластер WSFC.

Группы доступности могут быть реализованы как в среде Windows, так и в среде Linux.

• **Базовые группы доступности AlwaysOn.** Базовые группы доступности AlwaysOn заменяют нерекомендуемую функцию зеркального отображения базы данных и предоставляют такой же уровень поддержки. Базовые группы доступности позволяют поддерживать одну реплику базы данных-источника. Эта реплика может использовать режим синхронной или асинхронной фиксации. Реплика не может находиться в состоянии **Для чтения**.

• **Доставка журналов.** Доставка журналов — это дешевый способ создания резервного сервера с использованием стандартного оборудования. Первоначально полная резервная копия базы данных, находящейся на сервере-источнике, восстанавливается в резервную систему, которая затем периодически обновляется путем применения к резервной системе журналов транзакций с сервера-источника. Доставка журналов доступна для пользовательских баз данных, однако для системных баз

данных операции резервного копирования и восстановления требуется выполнять вручную. Если отказывает сервер-источник, необходимо вручную перевести резервный сервер в оперативный режим.



Кластеры серверов имеют один или более серверов (называемых *узлами*), которые используют общие жесткие диски. Кластер серверов обеспечивает высокий уровень доступности всего экземпляра SQL Server, но не защищает против отказов дисков. База данных хранится на общем жестком диске, чтобы в случае отказа узла другой узел начал действовать вместо него. При использовании SQL Server 2017 Enterprise Edition в кластере количество узлов ограничивается операционной системой.

Кластеризация — это стратегия обеспечения высокого уровня доступности, в которой несколько физических серверов настроены так, что ведут себя как единый виртуальный сервер. Каждый физический сервер упоминается как *узел*, а каждое приложение базы данных, расположенное в кластере, имеет назначенный *активный узел*, которым обслуживаются запросы от клиентов. Узлы в кластере совместно пользуются массивом дисков хранения, и в случае отказа активного узла приложения другой узел в кластере автоматически принимает на себя роль активного узла. Это автоматическое переназначение активного узла известно, как *автоматический переход на другой ресурс*, а конфигурация кластеризации, поддерживающая автоматический переход на другой ресурс, называется *отказоустойчивой кластеризацией.*

Поддержка отказоустойчивой кластеризации существует в SQL Server 2017 Enterprise Edition, Developer Edition и с некоторыми ограничениями в выпуске Standard Edition.

Выпуски SQL Server и кластеризация

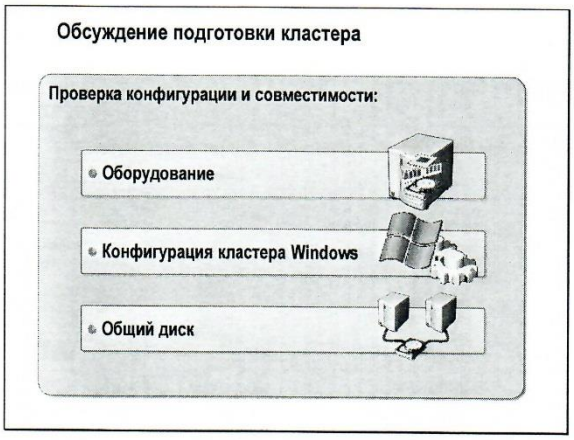
Выпуски SQL Server Enterprise Edition и SQL Server Developer Edition в полном объеме поддерживают кластеризацию на всех узлах, поддерживаемых операционной системой. Выпуск Standard Edition поддерживает кластеризацию лишь двух узлов. В других выпусках возможность создания кластеров отсутствует.

Используйте отказоустойчивую кластеризацию в следующих случаях:

• Требуется автоматический переход на другой ресурс в случае отказа сервера.

• Существует необходимость в автоматическом переходе на ресурсы уровня сервера, такие как имена для входа, конечные точки, задания и конфигурация агента SQL Server.

• Имеется оборудование, позволяющее создавать кластеры.



Кластеризацией предъявляются особые требования к оборудованию и программному обеспечению.

Оборудование

Для Windows Server 2012/2016 используемое оборудование должно отображаться в каталоге и списке совместимого оборудования Microsoft Windows. Аппаратная система должна отображаться в категории кластерного решения. При использовании сети хранения данных (SAN) все аппаратное решение должно принадлежать категории кластерных/многокластерных устройств в каталоге и списке совместимого оборудования Microsoft Windows. В Windows Server 2012 все компоненты оборудования в кластере должны иметь сертификацию "Certified for Windows Server 2012". Конфигурация должна пройти все тесты мастера Validate a Configuration Wizard. Если кластерное решение использует географически распределенные узлы, требуется дополнительная проверка сети и общего дискового ресурса. Список оборудования в этом случае называется Geographic Cluster Hardware Compatibility List.

Конфигурация кластера Windows

Убедитесь, что используемая операционная система поддерживает отказоустойчивую кластеризацию. Активизируйте поставщик службы криптографии Windows (CSP — Cryptographic Service Provider) в Microsoft Windows Server. Если служба CSP не выполняется на каком-либо узле кластера, работа программы установки SQL Server завершится сбоем с выводом диалогового окна, содержащего требования совместимости с Windows. Активизируйте службу планировщика задач во всех операционных системах для удаленной и кластерной установки. Если служба планировщика задач отключена, работа программы установки SQL Server завершится ошибкой 1058.

Общий диск

Кластером используются общие диски, чтобы в случае возникновения отказа другой узел мог стать владельцем дисков. Приложением SQL Server 2017 поддерживаются точки подключения. Общий диск может принадлежать сети хранения данных или быть доступен по протоколу SMB. Кластеризованные установки SQL Server ограничиваются числом доступных букв дисков. Если предположить, что используется только одна буква диска для операционной системы, а буквы всех других дисков доступны как обычные диски кластера или диски кластера, на которых размещаются точки подключения, тогда на каждый сервер может приходиться максимум 25 экземпляров SQL Server.

Служба кластера

Служба кластера MSCS должна быть установлена хотя бы на одном из узлов кластера.

Для установки, конфигурирования и обслуживания кластера SQL Server 2017 используется программа установки. Программа установки может выполнить:

1. Установку и создание отказоустойчивого кластера.
2. Добавление и удаление узлов в конфигурацию кластера, не затрагивая другие узлы кластера
3. Назначение нескольких IP-адресов каждому кластеризованному экземпляру, по одному адресу на подсеть

Настройка виртуального сервера

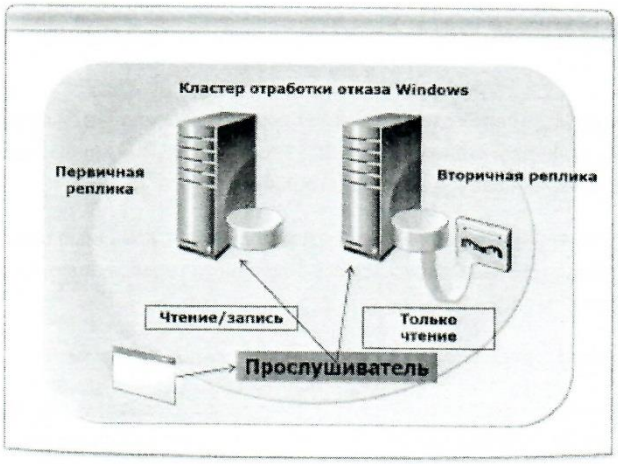
Предположив, что отказоустойчивый кластер уже существует, выберите кластерный диск, где хотите расположить файлы данных SQL Server, а затем запустите программу установки SQL Server на узле, который управляет этим диском. Необходимо также указать дополнительные узлы кластера, которые должны быть включены в виртуальный сервер. Программой установки автоматически установятся

требуемые компоненты SQL Server на каждом узле в виртуальном сервере. Создайте экземпляр SQL Server, используемый по умолчанию, а затем укажите сетевое имя виртуального сервера. При каждом подключении к SQL Server следует использовать имя этого виртуального сервера. Виртуальный сервер в каждый данный момент времени работает на одном из узлов. Остальные узлы играют пассивную роль. Балансировка нагрузки не выполняется.

Установка экземпляров SQL Server в кластере

В каждой группе ресурсов может содержаться максимум один экземпляр SQL Server. Чтобы установить другой экземпляр, запустите программу установки на узле кластера, управляющем кластерным диском, на котором будут находиться файлы данных SQL Server. Создайте именованный экземпляр с именем нового виртуального сервера в другой группе ресурсов кластера службы кластеризации Windows. Каждый виртуальный сервер находится в своей группе ресурсов службы кластеризации Windows, причем каждый виртуальный сервер имеет уникальный набор IP-адресов, индивидуальное сетевое имя и файлы данных, которые размещаются на отдельном наборе общих кластерных дисков. Когда для какого-либо ресурса в группе ресурсов службы кластеризации Windows выполняется переход на другой ресурс, для всех ресурсов, являющихся членами этой группы, также выполняется переход на другие ресурсы.

Занятие 3. Внедрение групп доступности AlwaysOn



Работа групп доступности AlwaysOn основана на синхронизации реплик баз данных. Экземпляры SQL Server должны быть установлены на серверах, входящих в отказоустойчивый кластер Windows.

Когда следует использовать группы доступности

Группы доступности AlwaysOn используется в следующих случаях:

• Существует необходимость в резервировании на уровне отдельной базы данных, например, базы данных системы «1С:Предприятие 8»

• Нежелательны инвестиции в оборудование, обладающее возможностями создания кластера SQL Server.

• Требуется сократить расходы на администрирование по сравнению с отказоустойчивой кластеризацией.

Прежде чем можно будет активировать группы доступности, необходимо выполнить ряд действий.

Подготовительная работа для настройки групп доступности AlwaysOn

• Построить отказоустойчивый кластер Windows (WSFC), если он необходим.

• В оснастке **Диспетчер конфигурации** в свойствах службы MSSQLSERVER разрешить группы доступности AlwaysOn

• **Установка модели восстановления**. Для баз данных в группе доступности необходимо установить для модели восстановления значение FULL(полная).

• **Резервное копирование основной базы данных и ее восстановление на сервере реплики.** При использовании мастера для настройки группы доступности AlwaysOn для начальной синхронизации можно выбрать полное резервное копирование основной базы данных и ее восстановление на реплике. В альтернативном варианте можно использовать последнюю полную резервную копию. Можно указать, что реплики уже синхронизированы вручную. При использовании команды CREATE AVAILABILITY GROUP можно указать параметр **SEEDING_MODE=AUTOMATIC** для автоматизированной инициализации без резервной копии.

• **Копирование ресурсов на уровне сервера**. Следует вручную скопировать все ресурсы на уровне сервера, такие как имена входа и задания агента SQL Server, которые потребуются в случае перехода ресурса на реплику при возникновении аварийной ситуации.

• **Создание конечных точек.** Мастер создаст в каждом экземпляре конечные точки и регистрационное имя для всякого экземпляра сервера, запускаемого с основного сервера под отличной от других пользовательской учетной записью домена.

Перенаправление клиента в решении, использующем группы доступности

Когда в сеансе синхронизации реплик базы данных происходит переход на другой ресурс, все клиентские приложения должны подключиться к новому серверу, содержащему реплику. Клиентские приложения, использующие собственный клиент SQL (SNAC) или поставщик данных Microsoft .NET Framework версии 2.0 или 3.5 для Microsoft SQL Server, поддерживают автоматическое перенаправление клиента и могут обрабатывать переход ресурса на реплику в явном виде. Клиентские приложения, использующие другие технологии доступа к данным, должны настраиваться для перенаправления запросов на реплику при отработке отказа.

Группы доступности AlwaysOn используют конечные точки SQL Server для взаимодействия экземпляров. Можно также создать службу прослушивателя для группы AlwaysOn, которая будет принимать входящие соединения для группы. Прослушиватель состоит из уникального IP-адреса и уникального сетевого имени. Это является одним из наиболее существенных качеств, которые делают группу баз высоко доступной. Создание централизованной точки доступа к группе баз позволяет клиентским приложениям избежать проблем при отработке отказа. В группе доступности AlwaysOn используются традиционные синхронный и асинхронный режимы зеркальных сеансов. Асинхронные реплики

поддерживают ручную отработку отказа, а синхронные реплики поддерживают как ручную, так и автоматическую отработку отказа.

Служба прослушивателя AlwaysOn принимает централизованные запросы к группе доступности AlwaysOn.

Существует два режима синхронизации реплик базы данных, которые отличаются друг от друга по уровню производительности и защиты.

Синхронный режим

В этом режиме транзакции применяются к основной и зеркальной репликам баз данных синхронно. Когда основным сервером фиксируется транзакция, сервером реплики также фиксируется транзакция. Фиксация совершается основным сервером только тогда, когда сервер реплики присылает подтверждение, что им сохранена транзакция на диск. Режим допускает автоматический или ручной переход ресурса с основного сервера на сервер реплики.

Асинхронный режим

В асинхронном режиме основной сервер продолжает работу не дожидаясь получение подтверждения от сервера реплики. Режим допускает ручной переход ресурса с основного сервера на сервер реплики.

Конечная точка — это объект SQL Server, который предоставляет для сервера возможность сетевых подключений. Для синхронизации реплик базы данных конечная точка определяет TCP-порт (по умолчанию 5022), на котором экземпляром прослушиваются сообщения. Для каждого экземпляра требуется выделенная конечная точка.

Полные группы доступности AlwaysOn с максимальными возможностями, которые можно построить с использованием SQL Server 2017 Enterprise Edition, имеют следующие свойства.

- **Возможность отработки отказа при неисправностях на уровне базы при указании определенных метрик.**
- **Поддержка распределенных транзакций DTC для баз, входящих в группу доступности**. Необходим Windows Server 2016
- **Специальные управляемые учетные записи для групп доступности AlwaysOn**
- **Балансировка нагрузки для вторичных реплик для чтения**. Возможно циклическое использование на чтение вторичных реплик с помощью прослушивателя.
- **Дополнительные целевые реплики для автоматической отработки отказа**. Можно указать до трех реплик для автоматический отработки отказа. Это соответствует допустимому числу синхронных реплик
- **Минимальное число реплик для фиксации** (параметр REQUIRED_SYNCHRONIZED_SECONDARIES_TO_COMMIT). Гарантирует, что транзакции SQL Server будут ожидать, пока журналы транзакций не будут обновлены минимальным числом вторичных реплик. Значение по умолчанию — 0, что дает ту же реакцию на событие, как и в SQL Server 2016. Минимальное значение — 0. Максимальное значение — число реплик минус 1. Если SQL Server, на котором размещена вторичная синхронная реплика, перестает отвечать, SQL Server, на котором размещена первичная реплика, отметит вторичную реплику как NOT SYNCHRONIZED и продолжит. Когда недоступная база данных снова подключается к сети, она находится в состоянии "не синхронизировано", и реплика отмечена как неработоспособная, пока снова не синхронизируется с первичной репликой. Данный параметр гарантирует, что первичная реплика будет ждать, пока минимальное число реплик не зафиксирует каждую транзакцию. Если минимальное число реплик недоступно, фиксация в первичной реплике завершается ошибкой. В

кластере типа EXTERNAL этот параметр меняется при добавлении группы доступности в кластерный ресурс.

- При использовании Windows Server 2016 группы доступности AlwaysOn могут быть настроены вне домена.
- **Группы доступности AlwaysOn включают поддержку групп без кластеров**. В значении параметра CLUSTER_TYPE указывается WSFC, если группа доступности находится на экземпляре отказоустойчивого кластера в отказоустойчивом кластере Windows. EXTERNAL указывается, если кластер управляется диспетчером кластера, но не отказоустойчивым кластером Windows Server, например Linux Pacemaker. NONE указывается, если группа доступности не использует WSFC для координации кластера. Например, если группа доступности включает серверы Linux без диспетчера кластеров.
- **Есть возможность миграции между Windows и Linux**, что обеспечивает кросс платформенную миграцию и тестирование

Поддерживаются до 8 реплик, в том числе одна первичная реплика и две вторичные реплики для чтения с синхронной фиксацией.

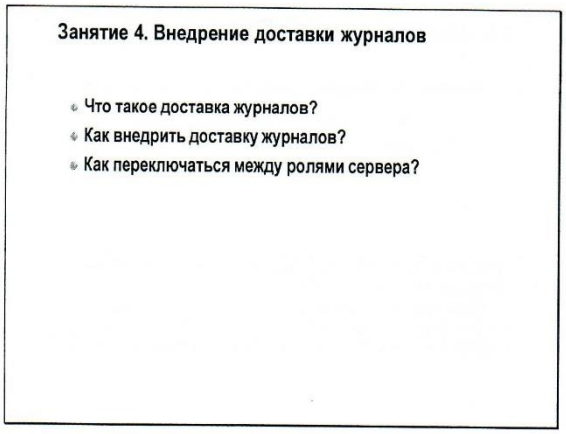
Создание группы доступности упрощается при использовании мастера настройки. Для устранения неполадок групп доступности AlwaysOn можно использовать специальную панель управления (Always On Availability Group Dashboard), а также журналы SQL Server и Windows. Панель управления позволяет просмотреть информацию о состоянии реплик, информацию о кворуме кластера, а также выполнить вручную отработку отказа.

Базовые группы доступности

Набор возможностей базовых групп доступности включает часть возможностей полных групп доступности из SQL Server 2017 Enterprise Edition. Для базовых групп доступности действуют следующие ограничения.

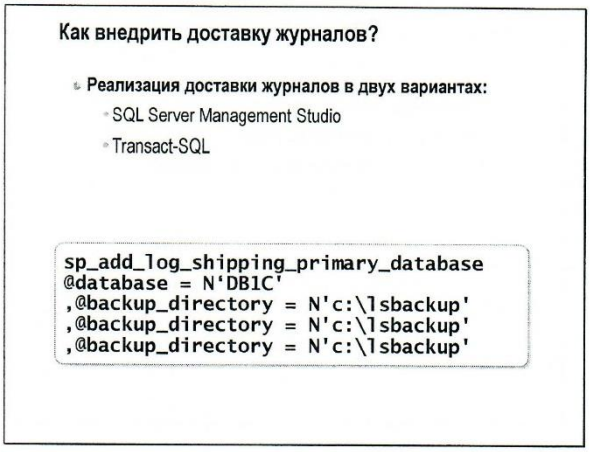
- Ограничение двух реплик (первичная и вторичная).
- Отсутствует доступ для чтения вторичной реплики.
- Отсутствует возможность резервного копирования вторичной реплики.
- Отсутствует поддержка добавления или удаления реплик в существующих основных группах доступности.
- Поддержка одной базы данных в группе доступности.
- Базовые группы доступности не могут быть обновлены до полных групп доступности. В этом случае группу необходимо удалить и повторно добавить в группу, содержащую серверы под управлением SQL Server 2017 Enterprise Edition.
- Базовые группы доступности поддерживаются только для серверов с установленным **SQL Server 2017 Standard Edition**.

Для базовых групп доступности AlwaysOn допускается создание прослушивателя. В процессе создания базовой группы доступности вам потребуется указать обе реплики.



Доставка журналов — это дешевый способ создания резервного сервера с использованием стандартного оборудования. Доставка журналов действует путем первоначального восстановления полной резервной копии базы данных с сервера-источника на сервер-получатель и последующего периодического применения журналов транзакций с сервера-источника к резервной системе. Доставка журналов доступна для пользовательских баз данных, и недоступна для системных баз данных.

Можно планировать резервные копирования журнала с частотой, которая наилучшим образом соответствует требованиям, предъявляемым к доступности и производительности. Помимо обеспечения избыточности резервный сервер может использоваться для запросов, доступных только для чтения, чтобы снять часть нагрузки с сервера-источника. В случае отказа сервера-источника автоматический переход на другой ресурс не выполняется. Необходимо вручную назначить резервному серверу другую роль и перенастроить все клиенты для подключения к этому серверу. Дополнительно можно создать сервер мониторинга. Сервер мониторинга регистрирует в журнале все проблемы с доставкой журналов, а также ведет запись последних операций резервного копирования и восстановления. Серверы мониторинга должны отличаться от сервера-источника и резервного серверов на случай, если один из серверов откажет.



Доставку журналов можно настроить с помощью среды SQL Server Management Studio или с помощью языка программирования Transact-SQL. Однако прежде чем настраивать доставку журналов, следует выполнить следующие задания:

• Создайте общую папку для резервных копий журнала транзакций, желательно на отказоустойчивом сервере, который не является частью конфигурации доставки журналов

• Создайте папку для каждого сервера-получателя, в которую при доставке журналов копируются файлы резервных копий журнала транзакций. Обычно эти папки находятся на серверах-получателях.

SQL Server Management Studio

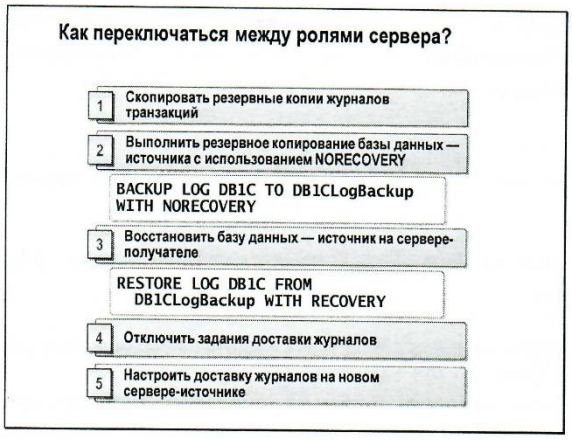
Доставку журналов можно настроить с помощью страницы доставки журналов транзакций диалогового окна «Свойства: База данных» в среде SQL Server Management Studio. Эта страница позволяет указать расписание резервного копирования базы данных — источника, а также экземпляр SQL Server и базу данных, куда должны восстанавливаться записанные резервные копии.

Transact-SQL

Доставку журналов можно также настроить вручную, используя следующие хранимые процедуры:

- master.dbo.sp_add_log_shipping_primary_database
- msdb.dbo.sp_add_schedule
- msdb.dbo.sp_attach_schedule
- msdb.dbo.sp_update_job
- master.dbo.sp_add_log_shipping_primary_secondary
- master.dbo.sp_add_log_shipping_alert_job
- master.dbo.sp_add_log_shipping_secondary_primary
- master.dbo.sp_add_log_shipping_secondary_database

При использовании SQL Server 2017 в доставке журналов можно применить сжатие резервных наборов данных.



С помощью переключения ролей резервный сервер становится сервером-источником. При первоначальном переключении ролей следует настроить доставку журналов для базы данных получателя. В этом нет необходимости при последующих изменениях ролей, и потом переключение назад и вперед осуществляется проще.

Переключение ролей

Чтобы переключить роли или назначить резервный сервер сервером-источником, выполните следующие действия:

1. Скопируйте все резервные копии журнала транзакций из общей папки резервных копий в целевую папку копирования и восстановите эти и все остальные резервные копии, содержащиеся в папке, на резервный сервер.

2. Если сервер-источник доступен, выполните резервное копирование журнала с параметром NORECOVERY, как показано в следующем примере кода Transact-SQL. BACKUP LOG DB1C TO DB1CLogBackup WITH NORECOVERY

3. Резервную копию, полученную на предыдущем шаге, восстановите на резервном сервере с использованием параметра RECOVERY, как показано в следующем примере кода Transact-SQL. RESTORE LOG DB1C FROM DB1CLogBackup WITH RECOVERY

В альтернативном варианте, если резервная копия недоступна, выполните восстановление с параметром RECOVERY без указания файла резервной копии, как показано в следующем примере кода Transact-SQL. RESTORE LOG DB1C WITH RECOVERY

4. Отключите задания доставки журналов на исходном сервере-источнике, а также отключите задания копирования и восстановления на сервере-получателе.

5. Если серверные роли изменяются в первый раз, потребуется настроить доставку журналов для базы данных — получателя. Теперь она должна обрабатываться как база данных — источник. Используйте для создания резервных копий ту же общую папку, которая была создана для исходного сервера-источника.

Когда добавляется база данных — получатель, в диалоговом окне **Настройки базы данных — получателя** введите имя исходной базы данных — источника в поле **База данных — получатель** и установите флажок **Нет, база данных — получатель инициализирована.**

Задание 12A. Настройка доставки журналов

Сформируйте группы из двух компьютеров и определите сервер-источник и сервер-получатель в доставке журналов

На сервере-получателе удалите базу DB1C. Установите на сервере получателе пароль для sa такой же как на сервере-источнике

1. Создайте папку **C:\Src** на сервере-источнике и дайте к ней сетевой доступ
2. Создайте папку **C:\Dest** на сервере-получателе

Задайте базу данных — источник для доставки журналов на сервере-источнике

1. В обозревателе объектов щелкните базу данных **DB1C** правой кнопкой мыши и выберите в контекстном меню пункт **Свойства**.
2. В диалоговом окне **Свойства Базы данных - DB1C** щелкните страницу **Доставка журналов транзакций.**
3. Установите флажок **Включить эту базу данных в качестве источника в конфигурацию доставки журналов**.

Настройте параметры резервного копирования для базы данных — источника

3. На странице **Доставка журналов транзакций** в разделе **Резервные копии журналов транзакций** щелкните **Параметры копирования**.
4. В диалоговом окне **Параметры резервного копирования журналов транзакций** в поле **Сетевой путь к папке резервного копирования** введите **\\<IP сервера -источника>\Src.**
5. В текстовом поле **Если папка резервного копирования находится на сервере-источнике, укажите локальный путь к папке** введите **C:\Src.**
6. В разделе **Задание резервного копирования** щелкните **Расписание**.
7. В диалоговом окне **Свойства расписания задания** в разделе **Сколько раз в день** установите повторяемость задания каждую 1 минуту и нажмите кнопку **OK**.
8. В диалоговом окне **Параметры резервного копирования журналов транзакций** нажмите кнопку **OK**.

Настройте сервер-получатель на сервере-источнике

1. На странице **Доставка журналов транзакций** в разделе **Экземпляры сервера-получателя и базы данных** щелкните **Добавить**.
2. В диалоговом окне **Параметры базы данных — получателя** щелкните **Соединить**.
3. В диалоговом окне **Соединение с сервером** подключитесь к серверу-получателю, используя проверку подлинности Windows.
4. В диалоговом окне **Настройки базы данных — получателя** в списке **База данных — получатель** введите **DB1C**.
5. На вкладке **Параметры базы данных — получателя** щелкните **Да, создать полную резервную копию базы данных — источника и выполнить восстановление из нее в базу данных — получатель**.
6. В диалоговом окне **Параметры базы данных — получателя** откройте вкладку **Копирование файлов**.
7. На вкладке **Копирование файлов** в поле **Папка назначения для копирования файлов** введите **C:\Dest**
8. В разделе **Задание копирования** щелкните **Расписание**.
9. В диалоговом окне **Свойства расписания задания** в разделе **Сколько раз в день** установите повторяемость задания каждую 1 минуту и нажмите кнопку **OK**.

10. В диалоговом окне **Параметры базы данных — получателя** откройте вкладку Восстановление журнала транзакций.
11. На вкладке **Восстановление журнала транзакций** в разделе **Состояние базы данных во время восстановления резервных копий** щелкните **Режим без восстановления**.
12. В разделе **Задание восстановления** щелкните **Расписание**.
13. В диалоговом окне **Свойства расписания задания** в разделе **Сколько раз в день** установите повторяемость задания каждую 1 минуту и нажмите кнопку **ОК**.
14. В диалоговом окне **Параметры базы данных — получателя** нажмите кнопку **ОК**.
15. В диалоговом окне **Свойства Базы данных – DB1C** нажмите кнопку **ОК**.
16. В диалоговом окне **Сохранение конфигурации доставки журналов** убедитесь, что все задания выполнены успешно и нажмите кнопку **Закрыть**.

Проверьте работоспособность доставки журналов

1. В обозревателе объектов щелкните **Соединить**, а затем щелкните **Компонент Database Engine.**
2. В диалоговом окне **Соединение с сервером** подключитесь к серверу-получателю, используя проверку подлинности Windows.
3. В обозревателе объектов раскройте папку **Базы данных** и убедитесь, что база данных **DB1C** находится в состоянии **Восстановления**.
4. Пользуясь проводником Windows на сервере-источнике, перейдите в папку **C:\Src**. Подождите минуту, а затем убедитесь, что резервные копии журналов созданы в этой папке.
5. Перейдите на сервере-получателе в папку **C:\Dest**. Подождите минуту, а затем убедитесь, что резервные копии журналов скопированы в эту папку.
6. Вернитесь в среду SQL Server Management Studio.

Остановите сервер-источник. На сервере-получателе приведите базу данных в рабочее состояние командой

RESTORE DATABASE DB1C WITH RECOVERY

Задание 12Б. Внедрение групп доступности AlwaysOn

Сформируйте группы из двух компьютеров для создания группы доступности AlwaysOn

Через панель управления компьютером установите средство администрирования кластера системы «1С:Предприятие 8»

Установите на сервере реплики пароль для sa такой же как на основном сервере

Создайте папку **C:\Share** на основном сервере и дайте сетевой доступ

На сервере-получателе удалите базу DB1C.

Проверьте кластер и конфигурацию AlwaysOn на базе двух экземпляров.

1. Откройте оснастку управления кластеризацией и проверьте, что узлы кластера находятся в состоянии **Up**
2. В диспетчере конфигурации в свойствах службы MSSQLSERVER разрешите группы доступности AlwaysOn и перезапустите службу MSSQLSERVER.
3. Сделайте полную резервную копию и копию журнала транзакций базы **DB1CX** в папку **C:\Share**
4. На сервере реплики восстановите базу в состояние NORECOVERY

Создайте группу доступности AlwaysOn на базе двух экземпляров.

1. На основном экземпляре создайте группу доступности **GRP1CX** с помощью мастера
2. Добавьте базу DB1CX в группу доступности

3. Добавьте реплику базы DB1CX в группу доступности
4. Настройте автоматическую отработку отказа.
5. Вторичную реплику настройте только для чтения
6. Основную реплику настройте для чтения и записи
7. Создайте прослушивателя группы доступности с параметрами
   - DNS имя - Lst1CX
   - Порт – 1433
   - IP-адрес – DHCP
   - Пропишите имя прослушивателя в параметрах информационной базы в консоли администрирования кластера системы «1С:Предприятие 8» в строке **Сервер баз данных**
8. На шаге **Выбор начальной синхронизации** выберите **Только соединение**
9. Просмотрите информацию о статусе группы доступности
10. Проверьте работоспособность приложения «1С:Предприятие 8»

Проверьте автоматическую отработку отказа

1. Остановите основной экземпляр или запустите отработку отказа с помощью мастера
2. Проверьте работоспособность приложения «1С:Предприятие 8»
3. Просмотрите информацию о статусе группы доступности