*Теоретические вопросы:*

1. **Методы обеспечения технологической безопасности Баз данных**

**1. Шифрование данных:** Шифрование данных помогает защитить данные от несанкционированного доступа.

**2. Резервное копирование данных:** Резервное копирование данных помогает защитить информацию от случайных ошибок или сбоев системы.

**3. Контроль доступа:** Контроль доступа позволяет ограничить доступ к информации только авторизованным пользователям.

**4. Защита от вредоносных программ:** Защита от вредоносных программ помогает предотвратить несанкционированный доступ к данным.

**5. Физическая безопасность:** Физическая безопасность помогает предотвратить несанкционированный доступ к данным.

**6. Аутентификация:** Аутентификация помогает предотвратить несанкционированный доступ к данным.

**7. Мониторинг безопасности:** Мониторинг безопасности помогает защитить данные от несанкционированного доступа.

1. **Способы Восстановление базы данных в критических ситуациях.**

* Резервное копирование и восстановление.
* Использование журналов транзакций.
* Использование систем управления базами данных (СУБД).
* Использование специализированных инструментов.
* Перенос данных на другую систему.

1. **Управление буферами базы данных. Механизм резервного копирования.**

**Управление буферами базы данных** - это процесс управления памятью, используемой для хранения данных и индексов базы данных. Буферы базы данных используются для временного хранения данных, которые будут записаны на диск. Это позволяет ускорить работу с базой данных, так как данные не нужно каждый раз считывать с диска.

**Механизм резервного копирования** - это процесс создания копий данных для восстановления в случае их потери или повреждения. Резервные копии могут быть созданы как вручную, так и автоматически с помощью специальных программ.

1. **Дать описание Технологии полного резервного копирования**

**Технология полного резервного копирования** - это процесс создания полной копии всех данных и настроек базы данных. Это означает, что все данные, включая таблицы, индексы, хранимые процедуры, функции и другие объекты базы данных, будут скопированы и сохранены в безопасном месте.

**Технология полного резервного копирования позволяет** восстановить базу данных в случае ее потери или повреждения. Это может произойти из-за различных причин, таких как аппаратные сбои, ошибки пользователей, атаки хакеров и другие факторы.

1. **Дать описание Технологии разностного резервного копирования**

**Технология разностного резервного копирования** - это процесс создания копии только тех данных, которые были изменены с момента последнего полного резервного копирования. Это означает, что при каждом новом резервном копировании создается только разница между текущим состоянием базы данных и предыдущим полным резервным копированием.

**Технология разностного резервного копирования позволяет** значительно сократить время и ресурсы, необходимые для создания резервных копий. Вместо того чтобы создавать полную копию всех данных каждый раз, создается только разница между текущим состоянием базы данных и предыдущим полным резервным копированием.

1. **Дать описание Технологии Инкрементального резервного копирования**

**Технология инкрементального резервного копирования** - это процесс создания копии только тех данных, которые были изменены с момента последнего полного или инкрементального резервного копирования. Это означает, что при каждом новом резервном копировании создается только разница между текущим состоянием базы данных и предыдущим инкрементальным резервным копированием.

**Технология инкрементального резервного копирования позволяет** значительно сократить время и ресурсы, необходимые для создания резервных копий. Вместо того чтобы создавать полную копию всех данных каждый раз, создается только разница между текущим состоянием базы данных и предыдущим инкрементальным резервным копированием.

1. **Способы защиты информации в СУБД MS SQL Server**

**Существует несколько способов защиты информации в СУБД MS SQL Server:**

1. **Шифрование данных:** MS SQL Server предоставляет возможность шифрования данных, что позволяет защитить информацию от несанкционированного доступа.

2. **Контроль доступа:** MS SQL Server позволяет установить различные уровни доступа к данным, что позволяет ограничить доступ только авторизованным пользователям.

3. **Резервное копирование данных:** MS SQL Server предоставляет возможность создания резервных копий данных, что позволяет восстановить данные в случае их потери или повреждения.

4. **Защита от вредоносных программ:** MS SQL Server предоставляет защиту от вредоносных программ, что позволяет предотвратить несанкционированный доступ к данным.

5. **Физическая безопасность:** MS SQL Server предоставляет возможность установки физической защиты на серверы и хранилища данных, что позволяет предотвратить несанкционированный доступ к данным.

6. **Аутентификация:** MS SQL Server предоставляет возможность аутентификации пользователей, что позволяет предотвратить несанкционированный доступ к данным.

7. **Мониторинг безопасности:** MS SQL Server предоставляет возможность мониторинга безопасности, что позволяет обнаруживать и предотвращать угрозы безопасности.

1. **Какие существуют типы ролей в СУБД MS SQL Server. Приведите пример.**

В СУБД MS SQL Server существуют различные типы ролей, которые определяют права и возможности пользователей при работе с базой данных. Ниже приведены некоторые из них:

1. **Администратор базы данных (DBA):** Имеет полный доступ к базе данных, включая создание, изменение и удаление объектов базы данных, а также управление правами доступа.

2. **Пользователь базы данных (DB User):** Имеет ограниченный доступ к базе данных, может выполнять только определенные операции, такие как чтение, запись или удаление данных.

3. **Аудитор:** Имеет доступ к информации о действиях, выполняемых другими пользователями в базе данных.

4. **Серверный администратор (SA):** Имеет полный доступ к серверу, включая управление правами доступа, создание и удаление баз данных, а также управление резервными копиями.

5. **Пользователь приложения:** Имеет доступ к базе данных только через приложение, которое использует эту базу данных.

**Пример роли - Администратор базы данных (DBA).** Эта роль имеет полный доступ к базе данных, включая создание, изменение и удаление объектов базы данных, а также управление правами доступа.

1. **Перечислить основные задачи, которые выполняет администратор баз данных**

**Основные задачи, которые выполняет администратор баз данных, включают:**

1. Установка, настройка и обновление СУБД и связанных с ней приложений.

2. Создание, изменение и удаление баз данных, таблиц, индексов, представлений, хранимых процедур и функций.

3. Управление правами доступа пользователей к базам данных и объектам внутри них.

4. Резервное копирование и восстановление баз данных.

5. Мониторинг производительности баз данных и оптимизация их работы.

6. Решение проблем, связанных с работой баз данных, таких как ошибки, сбои и отказы.

7. Обеспечение безопасности баз данных, включая защиту от несанкционированного доступа, вирусов и других угроз.

8. Поддержка пользователей баз данных, включая обучение, консультации и помощь в решении проблем.

9. Разработка и внедрение политик и процедур, связанных с управлением базами данных.

10. Управление инфраструктурой баз данных, включая серверы, хранилища данных и сетевые ресурсы.

1. **Назовите отличия двух системных ролей Server Admin и Security Admin. Какого уровня эти роли.**

**Server Admin и Security Admin** - это две системные роли в СУБД MS SQL Server. Они имеют разные уровни доступа и ответственности.

**Server Admin** - это роль, которая имеет полный доступ к серверу, включая управление правами доступа, создание и удаление баз данных, а также управление резервными копиями. Эта роль обычно предоставляется администраторам сервера или другим высокоуровневым сотрудникам, которые имеют широкий доступ к системе.

**Security Admin** - это роль, которая имеет доступ к информации о действиях, выполняемых другими пользователями в базе данных. Эта роль обычно предоставляется аудиторам или другим сотрудникам, которые отвечают за безопасность данных.

Таким образом, Server Admin имеет более высокий уровень доступа и ответственности, чем Security Admin.

1. **Как максимально быстро добиться целостности базы данных. Какие элементы буду использованы для этого?**

Для достижения максимальной целостности базы данных можно использовать следующие элементы:

1. **Правильная структура базы данных:** База данных должна быть правильно структурирована, чтобы обеспечить целостность данных. Это включает в себя создание правильных таблиц, полей и связей между ними.

2. **Правильные правила проверки данных:** Необходимо установить правила проверки данных, чтобы предотвратить ввод неправильных или неполных данных. Например, можно установить ограничения на длину полей, типы данных и т.д.

3. **Правильные правила обновления данных:** Необходимо установить правила обновления данных, чтобы предотвратить несанкционированное изменение данных. Например, можно установить ограничения на доступ к определенным полям или таблицам.

4. **Правильные правила удаления данных:** Необходимо установить правила удаления данных, чтобы предотвратить несанкционированное удаление данных. Например, можно установить ограничения на удаление определенных полей или таблиц.

5. **Правильные правила резервного копирования данных:** Необходимо установить правила резервного копирования данных, чтобы предотвратить потерю данных в случае сбоя системы или других проблем. Например, можно установить регулярное создание резервных копий базы данных.

6. **Правильные правила безопасности данных:** Необходимо установить правила безопасности данных, чтобы предотвратить несанкционированный доступ к данным. Например, можно установить пароли для доступа к базе данных, шифрование данных и т.д.

1. **Что такое ссылочная целостность между таблицами? На какие операция она распространяется.**

**Ссылочная целостность между таблицами** - это механизм, который обеспечивает корректность данных в базе данных. Он гарантирует, что данные в одной таблице соответствуют данным в другой таблице.

**Ссылочная целостность распространяется** на операции вставки, обновления и удаления данных. Когда данные вставляются, обновляются или удаляются в одной таблице, ссылочная целостность гарантирует, что соответствующие данные в другой таблице также будут изменены или удалены

1. **Опишите механизм действия свойства межтабличной связи «Каскадное обновление»**

**Свойство межтабличной связи "Каскадное обновление"** в базе данных позволяет автоматически обновлять связанные данные в одной таблице при изменении данных в другой таблице.

**Когда устанавливается связь "Каскадное обновление",** при изменении данных в одной таблице, связанные данные в другой таблице также автоматически обновляются. Это происходит благодаря тому, что при установке связи "Каскадное обновление" база данных автоматически создает триггеры, которые отслеживают изменения в одной таблице и обновляют соответствующие данные в другой таблице.

1. **Опишите механизм действия свойства межтабличной связи «Каскадное удаление»**

**Свойство межтабличной связи "Каскадное удаление"** в базе данных позволяет автоматически удалять связанные данные в одной таблице при удалении данных в другой таблице.

**Когда устанавливается связь "Каскадное удаление",** при удалении данных в одной таблице, связанные данные в другой таблице также автоматически удаляются. Это происходит благодаря тому, что при установке связи "Каскадное удаление" база данных автоматически создает триггеры, которые отслеживают удаление данных в одной таблице и удаляют соответствующие данные в другой таблице.

1. **Дайте определение Индекса. Какие существуют индексы.**

**Индекс** — это объект базы данных, который создаётся для повышения производительности поиска данных. Индекс формируется из значений одного или нескольких столбцов таблицы и указателей на соответствующие строки таблицы. Он позволяет искать строки, удовлетворяющие критерию поиска.

**Кластеризованный индекс** определяет физический порядок данных в таблице. Он обеспечивает быстрый доступ к данным при поиске в диапазоне значений.

**Некластеризованный индекс** не изменяет физическое упорядочивание строк таблицы. Он состоит из ключей индекса и закладок.

1. **Опишите алгоритм установки политики паролей в MS SQL Server.**

**Алгоритм установки политики паролей в MS SQL Server включает следующие шаги:**

1. Войдите в SQL Server Management Studio (SSMS) и подключитесь к серверу SQL Server.

2. В SSMS выберите "Политики безопасности" в левом меню.

3. В разделе "Политики безопасности" выберите "Политика паролей".

4. В правой части окна вы увидите настройки политики паролей.

5. В поле "Минимальная длина пароля" введите минимальную длину пароля, которую вы хотите установить.

6. В поле "Минимальное количество символов" введите минимальное количество различных символов, которые должны быть использованы в пароле.

7. В поле "Минимальное количество специальных символов" введите минимальное количество специальных символов, которые должны быть использованы в пароле.

8. В поле "Минимальное количество цифр" введите минимальное количество цифр, которые должны быть использованы в пароле.

9. В поле "Максимальное количество неудачных попыток входа" введите максимальное количество неудачных попыток входа, после которых учетная запись будет заблокирована.

10. В поле "Время блокировки после неудачных попыток входа" введите время, на которое учетная запись будет заблокирована после неудачных попыток входа.

11. Нажмите кнопку "Применить", чтобы применить изменения.

12. После применения изменений, проверьте, что политика паролей была успешно установлена.

1. **Инструкция GRANT. Определение, за что отвечает, где применяется.**

**Инструкция GRANT в SQL используется для** предоставления прав доступа к объектам базы данных. Она позволяет администратору базы данных (DBA) назначать определенные права доступа к таблицам, представлениям, хранимым процедурам, функциям и другим объектам базы данных.

**Инструкция GRANT отвечает за** предоставление прав доступа к объектам базы данных. Она позволяет DBA определить, какие пользователи или роли могут иметь доступ к определенным объектам базы данных и какие действия они могут выполнять с этими объектами.

**Инструкция GRANT применяется в различных сценариях**, когда необходимо предоставить доступ к объектам базы данных. Например, она может быть использована для предоставления доступа к таблице только для чтения, или для предоставления полного доступа к хранимой процедуре.

1. **Реляционная модель данных: элементы структуры, основные операции над данными и ограничения целостности. Виды отношений и ключей.**

**Реляционная модель данных** - это способ организации данных в базе данных, при котором данные хранятся в таблицах, связанных между собой. В реляционной модели данных элементы структуры включают в себя таблицы, столбцы и строки.

Основные операции над данными в реляционной модели данных включают в себя вставку, обновление, удаление и выборку данных.

**Ограничения целостности в реляционной модели данных включают в себя следующие:**

1. Ограничение целостности сущности: каждое значение в таблице должно быть уникальным.

2. Ограничение целостности полей: каждое поле в таблице должно иметь уникальное имя.

3. Ограничение целостности отношений: каждая таблица должна иметь первичный ключ, который является уникальным идентификатором каждой строки в таблице.

4. Ограничение целостности ссылок: каждая ссылка на другую таблицу должна быть уникальной и соответствовать первичному ключу этой таблицы.

**Виды отношений в реляционной модели данных включают в себя:**

1. Один-к-одному (1:1): каждая строка в одной таблице связана с одной строкой в другой таблице.

2. Один-ко-многим (1:М): каждая строка в одной таблице связана с несколькими строками в другой таблице.

3. Многие-ко-многим (М:М): каждая строка в одной таблице связана с несколькими строками в другой таблице, и каждая строка в другой таблице связана с несколькими строками в первой таблице.

**Ключи в реляционной модели данных включают в себя:**

1. Первичный ключ: уникальный идентификатор каждой строки в таблице.

2. Вторичный ключ: уникальный идентификатор каждой строки в таблице, который не является первичным ключом.

3. Внешний ключ: уникальный идентификатор каждой строки в таблице, который ссылается на первичный ключ другой таблицы.

1. **Как работает шифрование данных с использованием сертификаты в MS SQL Server**

**Шифрование данных с использованием сертификатов в MS SQL Server** - это процесс защиты данных путем их преобразования в непонятный для посторонних вид. Сертификаты используются для обеспечения безопасности и конфиденциальности данных, передаваемых между клиентом и сервером.

**Шифрование данных с использованием сертификатов в MS SQL Server работает следующим образом:**

1. Сервер генерирует сертификат, который содержит информацию о домене и пользователях, имеющих доступ к базе данных.

2. Клиент получает сертификат от сервера и использует его для аутентификации.

3. Клиент и сервер обмениваются ключами шифрования, которые используются для защиты данных.

4. Данные, передаваемые между клиентом и сервером, шифруются с использованием ключей шифрования.

5. После получения данных сервер расшифровывает их с использованием ключа шифрования.

6. Сервер проверяет сертификат клиента и, если он действителен, предоставляет доступ к базе данных.

1. **Инструкция REVOKE. Определение, за что отвечает, где применяется.**

**Инструкция REVOKE в SQL используется для** отмены предоставленных прав доступа к объектам базы данных. Она позволяет администратору базы данных (DBA) отменить права доступа, которые были предоставлены с помощью инструкции GRANT.

**Инструкция REVOKE отвечает за** отмену прав доступа к объектам базы данных. Она позволяет DBA определить, какие пользователи или роли больше не должны иметь доступ к определенным объектам базы данных и какие действия они больше не могут выполнять с этими объектами.

**Инструкция REVOKE применяется** в различных сценариях, когда необходимо отозвать доступ к объектам базы данных. Например, она может быть использована для отмены полного доступа к хранимой процедуре или для отмены доступа к таблице только для чтения.

**Инструкция REVOKE является** важной частью управления доступом в базе данных и помогает обеспечить безопасность данных, ограничивая доступ только авторизованным пользователям.

1. **Физическое проектирование. Особенности, влияющие на организацию внешней памяти. Технологии хранения данных.**

**Физическое проектирование базы данных** - это процесс организации и размещения данных на физическом уровне, то есть на уровне внешней памяти. Этот процесс включает в себя выбор технологий хранения данных и оптимизацию структуры базы данных для обеспечения эффективного доступа к данным.

**Особенности, влияющие на организацию внешней памяти, включают в себя:**

1. Размер базы данных: Чем больше размер базы данных, тем больше требуется памяти для ее хранения.

2. Тип данных: Различные типы данных требуют различного объема памяти для хранения. Например, текстовые данные занимают больше места, чем числовые данные.

3. Частота доступа: Чем чаще данные используются, тем больше требуется памяти для их хранения.

4. Требования к производительности: Если требуется высокая производительность, то необходимо выбирать технологии хранения данных, которые обеспечивают быстрый доступ к данным.

**Технологии хранения данных, которые могут быть использованы при физическом проектировании базы данных, включают в себя:**

1. Диски: Диски являются наиболее распространенным способом хранения данных. Они обеспечивают большой объем памяти и относительно низкую стоимость. Однако диски имеют низкую скорость доступа к данным.

2. Твердотельные накопители (SSD): SSD обеспечивают более быстрый доступ к данным по сравнению с дисками, но имеют меньший объем памяти и более высокую стоимость.

3. Флэш-память: Флэш-память обеспечивает очень быстрый доступ к данным и имеет большой объем памяти. Однако она имеет высокую стоимость и ограничена по количеству циклов записи.

4. RAM: RAM обеспечивает очень быстрый доступ к данным, но имеет ограниченный объем памяти и высокую стоимость.

1. **Дайте определение триггера. Свойства триггера. В каких случаях применяют INSTEAD OF, AFTER.**

**Триггеры используются** для автоматического выполнения определенных действий при изменении данных в базе данных.

**Свойства триггера:**

1. Событие: Триггеры могут быть установлены на различные события, такие как вставка, обновление или удаление данных.

2. Условия: Триггеры могут быть установлены с определенными условиями, которые должны быть выполнены для запуска триггера.

3. Действия: Триггеры могут выполнять различные действия, такие как вставка, обновление или удаление данных, а также выполнение хранимых процедур или функций.

4. Назначение: Триггеры могут быть назначены на определенные таблицы или представления.

5. Идентификатор: Каждому триггеру присваивается уникальный идентификатор.

**Триггеры могут быть использованы в различных случаях, например:**

- Для автоматического обновления связанных данных при изменении первичного ключа.

- Для автоматического удаления связанных данных при удалении первичного ключа.

- Для автоматического создания резервных копий данных при их изменении.

- Для автоматического выполнения определенных действий при выполнении определенных условий.

**Триггеры INSTEAD OF и AFTER** - это два типа триггеров, которые могут быть использованы в базе данных. Триггер INSTEAD OF используется для замены операции, которую он отслеживает, а триггер AFTER используется для выполнения действий после выполнения операции.

1. **Для чего нужен журнал Аудита MS SQL Server, какую информацию он содержит.**

**Журнал аудита MS SQL Server** - это специальный объект базы данных, который используется для отслеживания и записи всех действий, выполняемых в базе данных. Журнал аудита содержит информацию о том, кто, когда и какие действия выполнял в базе данных.

**Журнал аудита MS SQL Server содержит следующую информацию:**

1. Имя пользователя или роли, которые выполняли действие.

2. Дата и время выполнения действия.

3. Тип действия (например, вставка, обновление или удаление данных).

4. Имя объекта, на который было направлено действие (например, таблица или представление).

5. Информация о выполненном действии (например, строка, которая была удалена или изменена).

**Журнал аудита MS SQL Server используется для следующих целей:**

1. Мониторинг безопасности: Журнал аудита позволяет отслеживать все действия, выполняемые в базе данных, и выявлять любые несанкционированные действия.

2. Анализ производительности: Журнал аудита может быть использован для анализа производительности базы данных, чтобы определить, какие действия выполняются чаще всего и какие из них могут быть оптимизированы.

3. Восстановление данных: Журнал аудита может быть использован для восстановления данных, которые были случайно удалены или изменены.

4. Соблюдение нормативных требований: Журнал аудита может быть использован для соблюдения нормативных требований, таких как PCI DSS или HIPAA, которые требуют отслеживания всех действий, выполняемых в базе данных.

1. **Перечислите пути по уменьшению размера баз данных.**

**Существует несколько способов уменьшения размера баз данных:**

1. Оптимизация запросов: Оптимизация запросов может помочь уменьшить размер базы данных, так как оптимизированные запросы выполняются быстрее и используют меньше ресурсов.

2. Удаление неиспользуемых данных: Удаление неиспользуемых данных может помочь уменьшить размер базы данных, так как они занимают место на диске и увеличивают время выполнения запросов.

3. Архивация данных: Архивация данных может помочь уменьшить размер базы данных, так как архивные данные не используются регулярно и могут быть перемещены на менее дорогие носители данных.

4. Использование сжатия данных: Использование сжатия данных может помочь уменьшить размер базы данных, так как сжатые данные занимают меньше места на диске.

5. Регулярное резервное копирование: Регулярное резервное копирование может помочь уменьшить размер базы данных, так как резервные копии занимают меньше места на диске и могут быть удалены после успешного восстановления.

6. Использование индексов: Использование индексов может помочь уменьшить размер базы данных, так как индексы ускоряют выполнение запросов и уменьшают количество данных, которые необходимо прочитать.

7. Использование транзакций: Использование транзакций может помочь уменьшить размер базы данных, так как транзакции позволяют выполнять несколько операций как единое целое и уменьшают количество записей, которые необходимо сохранить.

8. Использование кэширования: Использование кэширования может помочь уменьшить размер базы данных, так как кэшированные данные занимают меньше места на диске и могут быть использованы быстрее.

1. **Обеспечение функционирования БД. Транзакции: понятия, модели завершения, свойства. Управление транзакциями.**

**Транзакции** - это набор операций, которые выполняются как единое целое. Они используются для обеспечения целостности данных в базе данных.

Модели завершения транзакций:

**1. Модель ACID (Atomicity, Consistency, Isolation, Durability):**

- Атомарность: Транзакция должна быть выполнена полностью или не выполнена вообще.

- Согласованность: Транзакция должна оставлять базу данных в согласованном состоянии.

- Изолированность: Транзакции не должны влиять на другие транзакции, выполняемые одновременно.

- Надежность: Изменения, внесенные транзакцией, должны быть постоянными и не могут быть отменены.

**2. Модель BASE (Basically Available, Soft state, Eventual consistency):**

- Базово доступная: Данные доступны для чтения и записи.

- Мягкое состояние: Данные могут быть изменены и не всегда согласованы.

- Согласованность в конечном итоге: Данные становятся согласованными со временем.

**Свойства транзакций:**

1. Атомарность: Транзакция должна быть выполнена полностью или не выполнена вообще.

2. Согласованность: Транзакция должна оставлять базу данных в согласованном состоянии.

3. Изолированность: Транзакции не должны влиять на другие транзакции, выполняемые одновременно.

4. Надежность: Изменения, внесенные транзакцией, должны быть постоянными и не могут быть отменены.

**Управление транзакциями:**

1. Уровни изоляции: Уровни изоляции определяют, как транзакции взаимодействуют друг с другом.

2. Журнал транзакций: Журнал транзакций используется для записи всех изменений, внесенных транзакцией.

3. Управление блокировками: Управление блокировками используется для предотвращения конфликтов между транзакциями.

4. Коммитирование и откат транзакций: Коммитирование и откат транзакций используются для завершения транзакции.

1. **Обеспечение функционирования БД. Журнализация: откат транзакции, восстановление данных в результате сбоев.**

**Обеспечение функционирования базы данных включает в себя** множество аспектов, одним из которых является журнализация. Журнализация — это процесс записи всех изменений, которые происходят в базе данных, в специальный журнал. Этот журнал позволяет восстановить данные в случае сбоя или ошибки.

**Откат транзакции** — это процесс отмены всех изменений, которые были сделаны в рамках транзакции. Это может быть необходимо, если транзакция была выполнена неправильно или если возникли проблемы с базой данных.

**Восстановление данных в результате сбоев** — это процесс восстановления данных после сбоя системы. Журнал изменений позволяет восстановить данные до момента сбоя и продолжить работу с базой данных.

1. **Обеспечение функционирования БД. Проблемы многопользовательских систем. Конфликты между транзакциями.**

**Обеспечение функционирования базы данных** в многопользовательской системе может столкнуться с рядом проблем. Одной из основных проблем является конфликт между транзакциями.

**Транзакция** — это набор операций, которые выполняются как единое целое. Если одна транзакция изменяет данные, которые затем пытается изменить другая транзакция, возникает конфликт.

**Существует несколько типов конфликтов между транзакциями:**

1. Конфликт чтения-записи: это происходит, когда одна транзакция пытается прочитать данные, которые были изменены другой транзакцией, но еще не были зафиксированы.

2. Конфликт записи-записи: это происходит, когда две транзакции пытаются изменить одни и те же данные.

3. Конфликт записи-отката: это происходит, когда одна транзакция пытается изменить данные, которые были изменены другой транзакцией, но затем были откачены.

1. **Обеспечение функционирования БД. Триггеры: понятие, правила создания. Хранимые процедуры: понятие, виды, преимущества использования.**

**Триггеры** — это специальные функции, которые автоматически вызываются при выполнении определенных событий в базе данных. Триггеры могут быть созданы для различных событий, таких как вставка, обновление или удаление данных.

Правила создания триггеров зависят от конкретной системы управления базами данных. Однако, в общем случае, триггеры создаются с помощью специального синтаксиса, который определяет, когда и как должен вызываться триггер.

**Хранимые процедуры** — это специальные функции, которые хранятся в базе данных и могут быть вызваны извне. Хранимые процедуры могут выполнять различные операции, такие как вставка, обновление или удаление данных.

**Существует несколько видов хранимых процедур:**

1. Простые хранимые процедуры: это процедуры, которые выполняют одну или несколько операций.

2. Многоступенчатые хранимые процедуры: это процедуры, которые выполняют несколько операций в определенной последовательности.

3. Хранимые функции: это функции, которые возвращают значение и могут быть вызваны из других хранимых процедур или из SQL-запросов.

**Преимущества использования хранимых процедур включают:**

1. Увеличение производительности: хранимые процедуры могут быть оптимизированы для выполнения определенных операций, что может увеличить производительность.

2. Упрощение кода: хранимые процедуры позволяют объединить несколько SQL-запросов в одну операцию, что упрощает код и делает его более читаемым.

3. Повторное использование кода: хранимые процедуры могут быть вызваны из разных мест базы данных, что позволяет повторно использовать код.

4. Защита данных: хранимые процедуры могут быть защищены паролем, что предотвращает несанкционированный доступ к данным.

1. **Администрирование БД: понятие, особенности.**

**Администрирование базы данных** — это процесс управления и обслуживания базы данных. Этот процесс включает в себя множество задач, таких как создание и удаление таблиц, индексов, пользователей, резервное копирование и восстановление данных, а также обеспечение безопасности и целостности данных.

Особенности администрирования базы данных зависят от конкретной системы управления базами данных. Однако, в общем случае, администратор базы данных должен обладать знаниями о структуре базы данных, ее функциях и возможностях.

**Администратор базы данных должен также иметь навыки** работы с инструментами администрирования, такими как SQL-запросы, хранимые процедуры и триггеры. Он должен уметь создавать и удалять таблицы, индексы, пользователей, а также настраивать права доступа к данным.

**Одной из важных задач администратора базы данных является** обеспечение безопасности данных. Это включает в себя установку паролей для пользователей, создание и управление ролями и группами пользователей, а также настройку прав доступа к данным.

**Администратор базы данных также должен уметь** выполнять резервное копирование и восстановление данных. Резервное копирование позволяет сохранить данные в случае сбоя системы или ошибки, а восстановление данных позволяет восстановить данные до момента сбоя.

1. **Хранилище данных: понятие, особенности.**

**Хранилище данных (Data Warehouse)** — это система, предназначенная для хранения больших объемов данных, которые используются для принятия решений в бизнесе. Хранилище данных представляет собой централизованное место, где хранятся данные из различных источников, таких как базы данных, файлы, приложения и т.д.

**Особенности хранилища данных:**

1. Хранилище данных предназначено для хранения больших объемов информации. Оно может содержать миллионы записей и терабайты данных.

2. Данные в хранилище данных обычно организованы в виде таблиц, которые могут быть связаны между собой.

3. Хранилище данных используется для анализа данных и принятия решений на основе этих данных.

4. Хранилище данных может быть использовано для создания отчетов, графиков и других инструментов визуализации данных.

5. Хранилище данных может быть использовано для интеграции данных из различных источников, что позволяет получить более полную картину о бизнесе.

6. Хранилище данных может быть использовано для создания систем управления базами данных (СУБД), которые позволяют управлять данными в хранилище данных.

7. Хранилище данных может быть использовано для создания систем управления бизнес-процессами (BPM), которые позволяют автоматизировать процессы в бизнесе.

1. **Что такое хранимая процедура в MS SQL Server. Для чего используется.**

**Хранимая процедура в MS SQL Server** - это подпрограмма, которая написана на языке Transact-SQL и хранится в базе данных. Она может быть вызвана из любого места, где используется SQL Server.

**Хранимые процедуры используются для** выполнения повторяющихся задач, таких как обновление таблиц, создание отчетов и т.д. Они позволяют упростить и ускорить выполнение этих задач, а также уменьшить количество ошибок при написании SQL-запросов.

**Хранимые процедуры также могут быть использованы для** обеспечения безопасности данных, так как они могут быть защищены паролем и иметь ограниченный доступ к данным. Кроме того, хранимые процедуры могут быть использованы для оптимизации производительности базы данных, так как они могут быть оптимизированы для выполнения определенных задач.

1. **В чёт отличии симметричного ключа шифрования от асимметричного ключа шифрования.**

**Симметричное и асимметричное шифрование** - это два разных метода шифрования данных.

**Симметричное шифрование использует** один и тот же ключ для шифрования и расшифровки данных. Это означает, что если вы хотите отправить зашифрованное сообщение кому-то, вы должны сначала обменяться ключом с этим человеком. Это может быть проблемой, если вы хотите отправить зашифрованное сообщение кому-то, кого вы не знаете или не можете обменяться ключом.

**Асимметричное шифрование** использует два разных ключа: открытый и закрытый. Открытый ключ может быть распространен публично, а закрытый ключ должен быть сохранен в секрете. Когда вы хотите отправить зашифрованное сообщение кому-то, вы используете его открытый ключ для шифрования сообщения. Только человек, имеющий соответствующий закрытый ключ, может расшифровать сообщение.

**Основное преимущество асимметричного шифрования заключается в том,** что оно позволяет безопасно обмениваться ключами без необходимости предварительного обмена ключами. Это делает его идеальным для передачи зашифрованных сообщений через Интернет или другие открытые сети.

1. **В чём отличие Уникального ключа (UNIQUE) от первичного ключа (PRIMARY KEY)**

**Уникальный ключ (UNIQUE) и первичный ключ (PRIMARY KEY)** - это два разных типа ключей в базах данных, которые используются для идентификации записей в таблице.

**Уникальный ключ (UNIQUE)** - это набор полей, которые гарантируют, что каждая запись в таблице имеет уникальное значение. Это означает, что в таблице не может быть двух записей с одинаковыми значениями в этих полях. Уникальный ключ может быть составным, то есть состоять из нескольких полей.

**Первичный ключ (PRIMARY KEY)** - это уникальный идентификатор каждой записи в таблице. Он также гарантирует, что каждая запись имеет уникальное значение. Однако, в отличие от уникального ключа, первичный ключ может быть только одним полем или составным ключом.

**Основное отличие между уникальным ключом и первичным ключом заключается в том,** что первичный ключ является обязательным для каждой записи в таблице, в то время как уникальный ключ может быть необязательным. Если вы хотите добавить запись в таблицу без уникального ключа, вы можете это сделать. Однако, если вы хотите добавить запись в таблицу без первичного ключа, это невозможно.

1. **Дайте определение инструкции CHECK CONSTRAINT. Для чего в MS SQL Server используется данная инстркция.**

**Инструкция CHECK CONSTRAINT** в MS SQL Server используется для проверки условий на значения в столбцах таблицы. Эта инструкция позволяет определить, какие значения могут быть введены в столбец таблицы.

**Инструкция CHECK CONSTRAINT** может быть использована для проверки различных условий, таких как:

- Значение должно быть больше или равно определенному значению.

- Значение должно быть меньше или равно определенному значению.

- Значение должно быть больше определенного значения.

- Значение должно быть меньше определенного значения.

- Значение должно быть равно определенному значению.

- Значение должно быть не равно определенному значению.

- Значение должно быть в определенном диапазоне.

- Значение должно быть не в определенном диапазоне.

**Инструкция CHECK CONSTRAINT может быть использована для** обеспечения целостности данных в таблице, предотвращая ввод некорректных данных. Она также может быть использована для упрощения работы с данными, так как позволяет избежать необходимости проверки условий вручную при вводе данных.

1. **Понятие первичного ключа, зачем он нужен в таблице.**

**Первичный ключ (Primary Key)** - это уникальный идентификатор каждой записи в таблице базы данных. Он используется для однозначной идентификации каждой записи в таблице.

**Первичный ключ необходим для следующих целей:**

1. Уникальность: каждый первичный ключ должен быть уникальным, то есть не может быть двух записей с одинаковым первичным ключом. Это гарантирует, что каждая запись в таблице имеет уникальный идентификатор.

2. Идентификация: первичный ключ используется для идентификации каждой записи в таблице. Это позволяет быстро и точно находить нужную запись в таблице.

3. Связь: первичный ключ может быть использован для связи с другими таблицами в базе данных через внешние ключи. Это позволяет создавать связи между различными таблицами и обеспечивает целостность данных.

4. Ограничение: первичный ключ может быть использован для ограничения ввода данных в таблицу. Например, если первичный ключ является обязательным полем, то пользователь не сможет добавить новую запись без указания значения для этого поля.

1. **Понятие составного ключа, когда применяется составной первичный ключ, какие данные хранит**

**Составной ключ** - это ключ, который состоит из нескольких полей таблицы. Он используется, когда одно поле не может однозначно идентифицировать запись в таблице.

Составной первичный ключ применяется, когда несколько полей вместе могут однозначно идентифицировать запись в таблице.

**Например**, если у вас есть таблица с информацией о клиентах, где каждый клиент имеет уникальный номер клиента и уникальный адрес, то вы можете использовать составной первичный ключ, состоящий из этих двух полей.

**Составной ключ хранит несколько полей**, которые вместе могут однозначно идентифицировать запись в таблице. Например, если у вас есть таблица с информацией о клиентах, где каждый клиент имеет уникальный номер клиента и уникальный адрес, то составной ключ будет состоять из этих двух полей.

1. **Дать определение связи один – к – одному**

**Связь один-к-одному (one-to-one relationship)** - это тип связи между двумя таблицами в реляционной базе данных, при котором каждой записи в одной таблице соответствует только одна запись в другой таблице.

**Например**, у вас есть две таблицы: "Студенты" и "Учебные группы". Каждый студент может быть членом только одной учебной группы, и каждая учебная группа может иметь только одного студента. В этом случае между таблицами "Студенты" и "Учебные группы" существует связь один-к-одному.

**Связь один-к-одному используется для** хранения связанных данных в разных таблицах, но при этом обеспечивает уникальную связь между этими данными. Это позволяет сохранять данные более структурированно и эффективно использовать пространство на диске.

1. **Дать определение связи один – к – многим**

**Связь один-к-многим (one-to-many relationship)** - это тип связи между двумя таблицами в реляционной базе данных, при котором каждой записи в одной таблице может соответствовать несколько записей в другой таблице.

**Например**, у вас есть две таблицы: "Студенты" и "Учебные предметы". Каждый студент может изучать несколько учебных предметов, но каждый учебный предмет может изучать только один студент. В этом случае между таблицами "Студенты" и "Учебные предметы" существует связь один-к-многим.

**Связь один-к-многим используется для** хранения связанных данных в разных таблицах, но при этом обеспечивает уникальную связь между этими данными. Это позволяет сохранять данные более структурированно и эффективно использовать пространство на диске.

1. **Дать определение связи многие – к – многим**

**Связь многие-к-многим (many-to-many relationship)** - это тип связи между двумя таблицами в реляционной базе данных, при котором каждой записи в одной таблице может соответствовать несколько записей в другой таблице, и наоборот.

**Например,** у вас есть две таблицы: "Студенты" и "Учебные предметы". Каждый студент может изучать несколько учебных предметов, и каждый учебный предмет может изучать несколько студентов. В этом случае между таблицами "Студенты" и "Учебные предметы" существует связь многие-к-многим.

**Связь многие-к-многим используется для** хранения связанных данных в разных таблицах, но при этом обеспечивает уникальную связь между этими данными. Это позволяет сохранять данные более структурированно и эффективно использовать пространство на диске.

**Связь многие-к-многим может быть реализована** с помощью третьей таблицы, которая содержит первичный ключ из каждой из двух таблиц. Эта третья таблица называется ассоциативной таблицей или таблицей связи.

1. **Создание и модификация базы данных. Создание, модификация и удаление таблиц.**

**Создание и модификация базы данных** - это процесс, который включает в себя создание новой базы данных или изменение существующей. Это может включать в себя добавление новых таблиц, полей, индексов и других элементов.

**Создание таблицы (create table)** - это процесс создания новой таблицы в базе данных. Для этого нужно определить имя таблицы, а также определить поля, которые будут содержаться в таблице. Поля могут быть различных типов данных, таких как строки, числа, даты и т.д.

**Модификация таблицы (update table)** - это процесс изменения существующей таблицы. Это может включать в себя добавление новых полей, удаление существующих полей или изменение их типов данных.

**Удаление таблицы (drop table)** - это процесс удаления таблицы из базы данных. Это может быть полезно, если таблица больше не нужна или содержит устаревшую информацию.

1. **Обеспечение сохранности базы данных. Создание полной резервной копии базы данных. Восстановление базы данных из резервных копий.**

**Обеспечение сохранности базы данных** - это процесс, который включает в себя создание резервных копий базы данных и восстановление данных из этих копий в случае необходимости.

**Создание полной резервной копии базы данных** - это процесс создания копии всех данных, которые содержатся в базе данных. Это может быть полезно в случае, если база данных будет повреждена или утеряна.

**Восстановление базы данных из резервных копий** - это процесс восстановления данных из резервной копии базы данных. Это может быть полезно в случае, если база данных была повреждена или утеряна.

1. **Какая резервная копия будет использоваться в базе данных, с небольшим набором данных, но в которой в день будет минимум 300 транзакций. Объясните почему.**

В случае с базой данных, в которой в день будет минимум 300 транзакций, рекомендуется использовать полную резервную копию базы данных.

**Полная резервная копия базы данных включает в себя** все данные, которые содержатся в базе данных на момент создания копии. Это означает, что она будет содержать все изменения, которые были сделаны в базе данных с момента последнего создания полной резервной копии.

**При использовании инкрементальной резервной копии**, которая включает только изменения, сделанные с момента последнего создания полной резервной копии, может возникнуть проблема с восстановлением данных в случае, если база данных была повреждена или утеряна. В этом случае, восстановление данных может быть затруднено или невозможно, если не будет доступна полная резервная копия.

1. **Где администратор базы данных может сохранять резервные копии.**

**Администратор базы данных может сохранять резервные копии на различных носителях информации**, таких как жесткие диски, сетевые хранилища (NAS), ленточные библиотеки и облачные хранилища.

**Жесткие диски** - это один из наиболее распространенных способов хранения резервных копий. Они обеспечивают быстрый доступ к данным и могут быть легко перенесены на другой компьютер. Однако жесткие диски могут быть подвержены повреждениям и потере данных.

**Сетевые хранилища (NAS)** - это устройства, которые позволяют хранить данные на нескольких жестких дисках и обеспечивают доступ к данным через сеть. Они обеспечивают высокую скорость доступа к данным и могут быть использованы для хранения резервных копий.

**Ленточные библиотеки** - это устройства, которые используют ленты для хранения данных. Они обеспечивают высокую емкость хранения и могут быть использованы для создания резервных копий больших объемов данных.

**Облачные хранилища** - это сервисы, которые предоставляют хранение данных в интернете. Они обеспечивают высокую доступность и масштабируемость, а также могут быть использованы для хранения резервных копий.

1. **Какие права доступа необходим выдать начальнику конкретного отдела. Какого уровня доступа будут эти права. Объясните почему.**

**Права доступа, которые необходимо выдать начальнику конкретного отдела, зависят от его роли и обязанностей в организации**. Обычно начальнику отдела предоставляются права доступа, которые позволяют ему выполнять свои обязанности и контролировать работу своих подчиненных.

***Начальнику отдела могут быть предоставлены следующие права доступа:***

1. **Право на чтение данных** - это право позволяет начальнику отдела просматривать данные в базе данных. Это может быть полезно для контроля за работой подчиненных и анализа данных.

2. **Право на изменение данных** - это право позволяет начальнику отдела изменять данные в базе данных. Это может быть полезно для внесения изменений в данные, которые были неправильно введены или для исправления ошибок.

3. **Право на удаление данных** - это право позволяет начальнику отдела удалять данные из базы данных. Это может быть полезно для удаления устаревших или ненужных данных.

4. **Право на создание новых записей** - это право позволяет начальнику отдела создавать новые записи в базе данных. Это может быть полезно для добавления новых сотрудников или клиентов.

5. **Право на выполнение административных задач** - это право позволяет начальнику отдела выполнять административные задачи, такие как создание новых пользователей, изменение прав доступа и т.д.

1. **СУБД MS SQL Server, использует стандартную политику защиты паролей Windows. Почему именно её. Какие основные правила этой политики защиты паролей.**

**СУБД MS SQL Server использует стандартную политику защиты паролей Windows**, потому что это обеспечивает совместимость с другими продуктами Microsoft и упрощает управление паролями.

***Основные правила этой политики защиты паролей включают:***

1. **Минимальная длина пароля** - пароль должен содержать не менее 8 символов.

2. **Минимальное количество различных типов символов** - пароль должен содержать не менее 3 различных типов символов, таких как буквы верхнего и нижнего регистра, цифры и специальные символы.

3. **Срок действия пароля** - пароль должен меняться каждые 90 дней.

4. **Запрет на использование предыдущих паролей** - пользователь не может использовать свой предыдущий пароль при смене пароля.

5. **Запрет на использование общих слов или фраз** - пользователь не может использовать общие слова или фразы в своем пароле.

6. **Запрет на использование личных данных** - пользователь не может использовать свои личные данные, такие как имя, фамилия, дата рождения и т.д. в своем пароле.

1. **В чём отличие инструкции GRANT от инструкции WITH GRANT OPTION**

**Инструкция GRANT используется** для предоставления прав доступа к объектам базы данных, таким как таблицы, представления, процедуры и т.д. Она позволяет предоставить пользователю или роли определенные права доступа, такие как право на чтение, изменение или удаление данных.

**Инструкция WITH GRANT OPTION используется** для предоставления прав доступа с возможностью передачи этих прав другим пользователям или ролям. Это означает, что пользователь, которому были предоставлены права доступа с использованием этой инструкции, может в свою очередь предоставить эти же права другим пользователям или ролям.

Например, если администратор базы данных предоставляет пользователю право на изменение данных в определенной таблице с использованием инструкции GRANT, то этот пользователь сможет изменять данные в этой таблице. Однако, если администратор базы данных предоставляет пользователю право на изменение данных в той же таблице с использованием инструкции GRANT WITH GRANT OPTION, то этот пользователь сможет не только изменять данные в таблице, но и предоставить право на изменение данных другим пользователям или ролям.

1. **За что в триггере отвечает инструкция ROLLBACK. Как часто она используется.**

**Инструкция ROLLBACK в триггере отвечает за отмену всех изменений**, которые были сделаны в транзакции. Она используется в случае, если в процессе выполнения транзакции возникла ошибка или если пользователь решил отменить все изменения, которые были сделаны в этой транзакции.

**ROLLBACK используется** не очень часто, так как обычно пользователи хотят сохранить изменения, которые они сделали в транзакции. Однако, в некоторых случаях, когда возникают ошибки или пользователь хочет отменить все изменения, ROLLBACK может быть полезным инструментом.

Например, если в процессе выполнения транзакции возникла ошибка, то ROLLBACK может быть использован для отмены всех изменений, которые были сделаны в этой транзакции. Это поможет избежать потери данных и сохранить целостность базы данных.

**Также ROLLBACK может быть использован в случае**, если пользователь решил отменить все изменения, которые были сделаны в транзакции. Например, если пользователь случайно удалил данные, которые он не хотел удалять, он может использовать ROLLBACK для отмены этого действия.

1. **Зачем необходимо в плане резервного копирования, ставить ограничение на хранение инкрементального BackUp.**

**Ограничение на хранение инкрементального BackUp в плане резервного копирования необходимо для того**, чтобы избежать накопления большого количества данных, которые могут занимать много места на диске или другом носителе информации.

**Инкрементальное резервное копирование** - это процесс создания резервной копии, которая содержит только изменения, сделанные с момента последнего создания полной резервной копии. Это позволяет экономить время и ресурсы, так как для создания инкрементальной резервной копии требуется меньше времени и места на диске, чем для создания полной резервной копии.

Однако, если не установить ограничение на хранение инкрементального BackUp, то со временем может накопиться большое количество данных, которые занимают много места на диске или другом носителе информации. Это может привести к проблемам с производительностью системы и затруднить доступ к данным.

1. **Что такое хранимая процедура в MS SQL Server. Для чего используется.**

**Хранимая процедура в MS SQL Server** - это набор инструкций SQL, которые можно вызывать из других запросов SQL. Она позволяет автоматизировать выполнение повторяющихся задач и упростить работу с базой данных.

**Хранимая процедура может содержать** любой набор инструкций SQL, включая создание, изменение и удаление таблиц, вставку, обновление и удаление данных, а также выполнение других операций с базой данных.

***Хранимые процедуры используются для следующих целей:***

1. **Упрощение работы с базой данных**: Хранимая процедура позволяет объединить несколько инструкций SQL в одну, что упрощает работу с базой данных и уменьшает вероятность ошибок.

2. **Автоматизация повторяющихся задач**: Хранимая процедура может быть использована для автоматизации повторяющихся задач, таких как вставка, обновление или удаление данных.

3. **Обеспечение безопасности**: Хранимая процедура может быть использована для ограничения доступа к базе данных, предоставляя доступ только к определенным хранимым процедурам.

4. **Ускорение работы:** Хранимая процедура может быть оптимизирована для ускорения работы, что может быть полезно при работе с большими объемами данных.

1. **Как работает шифрование данных с использованием сертификаты в MS SQL Server.**

***Шифрование данных с использованием сертификатов в MS SQL Server работает следующим образом:***

1. **Создание сертификата:** Сначала необходимо создать сертификат, который будет использоваться для шифрования данных. Сертификат может быть создан с помощью команды CREATE CERTIFICATE в SQL Server Management Studio.

2. **Шифрование данных:** После создания сертификата можно начать шифрование данных. Для этого необходимо использовать команду ENCRYPTBYCERT с указанием имени сертификата и данных, которые нужно зашифровать.

3. **Дешифрование данных:** Чтобы дешифровать данные, необходимо использовать команду DECRYPTBYCERT с указанием имени сертификата и зашифрованных данных.

4. **Управление сертификатами:** Сертификаты могут быть удалены, изменены или переименованы с помощью команд DROP CERTIFICATE, ALTER CERTIFICATE и RENAME CERTIFICATE соответственно.

5. **Безопасность:** Шифрование данных с использованием сертификатов обеспечивает высокий уровень безопасности, так как только владелец сертификата может дешифровать данные.

*Практические задания:*

**1. К тестовой базе данных создать полное резервное копирование. Сделать анализ увеличения размера файла резервного копирования.**

1. Откройте SQL Server Management Studio (SSMS) и подключитесь к вашему экземпляру SQL Server.

2. Выберите вашу тестовую базу данных в дереве объектов и щелкните правой кнопкой мыши, затем выберите "Tasks" -> "Back Up..." для создания резервной копии.

3. В окне "Back Up Database" выберите "Full" в качестве типа резервного копирования, укажите путь для сохранения файла резервной копии и нажмите "OK", чтобы начать процесс создания резервной копии.

4. После завершения создания резервной копии запомните размер файла резервного копирования.

5. Внесите изменения в вашу тестовую базу данных, например, добавьте новые данные или обновите существующие.

6. Повторите шаги 2-3 для создания нового полного резервного копирования после внесения изменений.

7. Сравните размер нового файла резервного копирования с исходным размером, чтобы проанализировать увеличение.

**2. К тестовой базе данных создать разностное резервное копирование. Провести анализ увеличения размера файла резервного копирования.**

1. Откройте SQL Server Management Studio (SSMS) и подключитесь к вашему экземпляру SQL Server.

2. Выберите вашу тестовую базу данных в дереве объектов и щелкните правой кнопкой мыши, затем выберите "Tasks" -> "Back Up..." для создания резервной копии.

3. В окне "Back Up Database" выберите "Full" в качестве типа резервного копирования, укажите путь для сохранения файла резервной копии и нажмите "OK", чтобы начать процесс создания полной резервной копии.

4. После завершения создания полной резервной копии восстановите базу данных из этой резервной копии.

5. Внесите изменения в вашу тестовую базу данных, например, добавьте новые данные или обновите существующие.

6. Повторите шаги 2-3 для создания разностной резервной копии. В окне "Back Up Database" выберите "Differential" в качестве типа резервного копирования и укажите путь для сохранения файла разностной резервной копии.

7. Сравните размер файла разностной резервной копии с размером файла полной резервной копии, чтобы проанализировать увеличение размера файла разностной копии относительно полной.

**3. К тестовой базе данных создать инкрементальное резервное копирование. Провести анализ увеличение размера файла резервного копирования.**

1. Откройте SQL Server Management Studio (SSMS) и подключитесь к вашему экземпляру SQL Server.

2. Выберите вашу тестовую базу данных в дереве объектов и щелкните правой кнопкой мыши, затем выберите "Tasks" -> "Back Up..." для создания резервной копии.

3. В окне "Back Up Database" выберите "Full" в качестве типа резервного копирования и укажите путь для сохранения файла резервной копии. Нажмите "OK", чтобы начать процесс создания полной резервной копии.

4. После завершения создания полной резервной копии, повторите шаги 2-3 для создания инкрементальной резервной копии. В окне "Back Up Database" выберите "Transaction Log" в качестве типа резервного копирования и укажите путь для сохранения файла инкрементальной резервной копии.

5. Внесите изменения в вашу тестовую базу данных, например, добавьте новые данные или обновите существующие.

6. Повторите шаги 2-4 для создания дополнительных инкрементальных резервных копий. Каждая последующая инкрементальная копия будет содержать только изменения, произошедшие с момента предыдущей инкрементальной или полной резервной копии.

7. Сравните размеры файлов инкрементальных резервных копий с размером файла полной резервной копии и анализируйте увеличение размера каждой последующей инкрементальной копии относительно предыдущей. Это поможет вам понять, какие изменения в базе данных приводят к увеличению размера инкрементальных копий и как можно оптимизировать процесс создания и хранения инкрементальных копий.

**4. К тестовой базе данных создать полное резервное копирование. Выполнить инструкцию DROP DATABASE, а затем восстановить базу данных из файла резервного копирования**

1. Создание полного резервного копирования:

- Откройте SQL Server Management Studio (SSMS) и подключитесь к вашему экземпляру SQL Server.

- Выберите вашу тестовую базу данных в дереве объектов и щелкните правой кнопкой мыши, затем выберите "Tasks" -> "Back Up..." для создания резервной копии.

- В окне "Back Up Database" выберите "Full" в качестве типа резервного копирования и укажите путь для сохранения файла резервной копии. Нажмите "OK", чтобы начать процесс создания полной резервной копии.

2. Выполнение инструкции DROP DATABASE:

- В новом запросе SQL выполните следующую инструкцию:

DROP DATABASE [название\_вашей\_тестовой\_базы\_данных];

Пожалуйста, замените [название\_вашей\_тестовой\_базы\_данных] на фактическое название вашей тестовой базы данных.

3. Восстановление базы данных из файла резервного копирования:

- Чтобы восстановить базу данных из файла резервного копирования, щелкните правой кнопкой мыши на "Databases" в дереве объектов SSMS и выберите "Restore Database...".

- В окне "Restore Database" выберите "Device" и укажите путь к файлу резервного копирования, который вы создали на первом шаге. Нажмите "OK", чтобы начать процесс восстановления базы данных из файла резервного копирования.

5. К тестовой базе данных создать разностное резервное копирование. Выполнить инструкцию DROP DATABASE, а затем восстановить базу данных из файла резервного копирования

6. К тестовой базе данных создать инкрементальное резервное копирование. Выполнить инструкцию DROP DATABASE, а затем восстановить базу данных из файла резервного копирования

**7. В тестовой базе данных создать пользователя с Именем «TestUser» и паролем 12345. При первом входе пароль должен быть изменён пользователем. Серверная роль Public**

1. Откройте SQL Server Management Studio (SSMS) и подключитесь к вашему экземпляру SQL Server.

2. Выполните следующий запрос SQL для создания пользователя с указанными параметрами:

USE [название\_вашей\_тестовой\_базы\_данных]

GO

CREATE LOGIN TestUser WITH PASSWORD = '12345', CHECK\_POLICY = OFF;

CREATE USER TestUser FOR LOGIN TestUser;

ALTER LOGIN TestUser WITH PASSWORD = '12345' UNLOCK;

ALTER LOGIN TestUser WITH PASSWORD = 'новый\_пароль' MUST\_CHANGE;

ALTER ROLE Public ADD MEMBER TestUser;

Пожалуйста, замените [название\_вашей\_тестовой\_базы\_данных] на фактическое название вашей тестовой базы данных и 'новый\_пароль' на новый пароль, который пользователь должен будет установить при первом входе.

3. После выполнения этого запроса, пользователь "TestUser" будет создан в вашей тестовой базе данных с паролем "12345", который он должен изменить при первом входе. Пользователь также будет иметь серверную роль Public.

8. В тестовой базе данных создать пользователя с Именем «TestAdministrator» и паролем admin123. При первом входе пароль должен быть изменён пользователем. Серверная роль ServerAdmin

9. В тестовой базе данных создать пользователя с Именем «TestAdministrator» и паролем admin123. При первом входе пароль должен быть изменён пользователем. Серверная роль ServerAdmin

10. В тестовой базе данных создать пользователя с Именем «TestCreator» и паролем creator123. При первом входе пароль должен быть изменён пользователем. Серверная роль dbCreator

**11. В тестовой базе данных необходимо создать пользователя с Именем «AuditUser». Далее создать таблицу с именем Audit. Данному пользователю дать возможность добавлять значения в эту таблицу.**

1. Создание пользователя "AuditUser":

USE [название\_вашей\_тестовой\_базы\_данных]

GO

CREATE LOGIN AuditUser WITH PASSWORD = 'новый\_пароль', CHECK\_POLICY = OFF;

CREATE USER AuditUser FOR LOGIN AuditUser;

Пожалуйста, замените [название\_вашей\_тестовой\_базы\_данных] на фактическое название вашей тестовой базы данных и 'новый\_пароль' на желаемый пароль для пользователя "AuditUser".

2. Создание таблицы "Audit":

CREATE TABLE Audit (

ID INT PRIMARY KEY,

Description NVARCHAR(255)

);

3. Предоставление прав на добавление значений в таблицу "Audit" пользователю "AuditUser":

GRANT INSERT ON Audit TO AuditUser;

12. В тестовой базе данных необходимо создать пользователя с Именем «AuditUser». Далее создать таблицу с именем Audit. Данному пользователю дать возможность изменять значения в этой таблицу.

13. В тестовой базе данных необходимо создать пользователя с Именем «AuditUser». Далее создать таблицу с именем Audit. Данному пользователю дать возможность удалять значения в этой таблице.

**14. В тестовой базе данных необходимо создать пользователя с Именем «Manager». Далее создать таблицу с именем Test. Данному пользователю дать возможность просматривать значения в этой таблице, и наделять других пользователей в этой базе данных такой же возможностью.**

1. Создание пользователя "Manager":

USE [название\_вашей\_тестовой\_базы\_данных]

GO

CREATE LOGIN Manager WITH PASSWORD = 'новый\_пароль', CHECK\_POLICY = OFF;

CREATE USER Manager FOR LOGIN Manager;

2. Создание таблицы "Test":

CREATE TABLE Test (

ID INT PRIMARY KEY,

Description NVARCHAR(255)

);

3. Предоставление прав на просмотр значений в таблице "Test" пользователю "Manager":

GRANT SELECT ON Test TO Manager;

4. Наделение других пользователей той же возможностью просматривать значения в таблице "Test":

GRANT SELECT ON Test TO [имя\_другого\_пользователя];

15. В тестовой базе данных необходимо создать пользователя с Именем «Manager». Далее создать таблицу с именем Test. Данному пользователю дать возможность добавлять значения в этой таблице, и наделять других пользователей в этой базе данных такой же возможностью.

16. В тестовой базе данных необходимо создать пользователя с Именем «Manager». Далее создать таблицу с именем Test. Данному пользователю дать возможность удалять значения в этой таблице, и наделять других пользователей в этой базе данных такой же возможностью.

**17. В тестовой базе данных необходимо создать таблицу с именем Clients. Атрибуты таблицы: Fname, LName, MName, DateOfBirth.Заполнить её тестовыми данными. Создать таблицы с именем History. Написать триггер, который будет добавлять в таблицу History значения: какие данные добавлены в таблицу Clients и в какое время.**

1. Создание таблицы "Clients" и заполнение её тестовыми данными:

CREATE TABLE Clients (

ID INT PRIMARY KEY IDENTITY(1,1),

Fname NVARCHAR(50),

LName NVARCHAR(50),

MName NVARCHAR(50),

DateOfBirth DATE

);

INSERT INTO Clients (Fname, LName, MName, DateOfBirth)

VALUES ('Иван', 'Иванов', 'Иванович', '1990-01-15'),

('Петр', 'Петров', 'Петрович', '1985-07-20'),

('Мария', 'Сидорова', 'Петровна', '1978-03-10');

2. Создание таблицы "History":

CREATE TABLE History (

ID INT PRIMARY KEY IDENTITY(1,1),

Action NVARCHAR(100),

TimeStamp DATETIME

);

3. Написание триггера для отслеживания добавления данных в таблицу "Clients" и записи изменений в таблицу "History":

CREATE TRIGGER trg\_ClientInsert

ON Clients

AFTER INSERT

AS

BEGIN

DECLARE @Action NVARCHAR(100);

SET @Action = 'Добавлены данные: ';

SELECT @Action = @Action + 'ID=' + CAST(ID AS NVARCHAR) + ', Fname=' + Fname + ', LName=' + LName + ', MName=' + MName + ', DateOfBirth=' + CONVERT(NVARCHAR, DateOfBirth, 120)

FROM inserted;

INSERT INTO History (Action, TimeStamp)

VALUES (@Action, GETDATE());

END;

18. В тестовой базе данных необходимо создать таблицу с именем Users. Атрибуты таблицы: Fname, LName, MName, DateOfBirth.Заполнить её тестовыми данными. Создать таблицы с именем History. Написать триггер, который будет добавлять в таблицу History значения: какие данные добавлены в таблицу Users и в какое время.

19. В тестовой базе данных необходимо создать таблицу с именем Managers. Атрибуты таблицы: Fname, LName, MName, DateOfBirth.Заполнить её тестовыми данными. Создать таблицы с именем History. Написать триггер, который будет добавлять в таблицу History значения: какие данные добавлены в таблицу Managers и в какое время.

20. В тестовой базе данных необходимо создать таблицу с именем Clients. Атрибуты таблицы: Fname, LName, MName, DateOfBirth.Заполнить её тестовыми данными. Создать таблицы с именем History. Написать триггер, который будет добавлять в таблицу History значения: какие данные изменены в таблице Clients и в какое время.

21. В тестовой базе данных необходимо создать таблицу с именем Users. Атрибуты таблицы: Fname, LName, MName, DateOfBirth.Заполнить её тестовыми данными. Создать таблицы с именем History. Написать триггер, который будет добавлять в таблицу History значения: какие данные изменены в таблице Users и в какое время.

22. В тестовой базе данных необходимо создать таблице с именем Managers. Атрибуты таблицы: Fname, LName, MName, DateOfBirth.Заполнить её тестовыми данными. Создать таблицы с именем History. Написать триггер, который будет добавлять в таблицу History значения: какие данные изменены в таблице Managers и в какое время.

23. В тестовой базе данных необходимо создать таблицу с именем Clients. Атрибуты таблицы: Fname, LName, MName, DateOfBirth.Заполнить её тестовыми данными. Создать таблицы с именем History. Написать триггер, который будет добавлять в таблицу History значения: какие данные удалены в таблице Clients и в какое время.

24. В тестовой базе данных необходимо создать таблицу с именем Users. Атрибуты таблицы: Fname, LName, MName, DateOfBirth.Заполнить её тестовыми данными. Создать таблицы с именем History. Написать триггер, который будет добавлять в таблицу History значения: какие данные удалены в таблице Users и в какое время.

25. В тестовой базе данных необходимо создать таблицу с именем Managers. Атрибуты таблицы: Fname, LName, MName, DateOfBirth.Заполнить её тестовыми данными. Создать таблицы с именем History. Написать триггер, который будет добавлять в таблицу History значения: какие данные удалены в таблице Managers и в какое время.