***Теоретические вопросы:***

1. Законодательство Российской̆ Федерации в области защиты информации.
2. Требования безопасности к серверам баз данных.
3. Основные группы методов противодействия угрозам безопасности в корпоративных сетях. Описание и примеры.
4. Программно-аппаратные методы защиты процесса обработки и передачи информации.
5. Политика безопасности: определение и инструменты.
6. Процесс настройки политики безопасности.
7. Резервное копирование. Цели. Методы.
8. Концепции резервного копирования: описание методов.
9. Этапы: наименование и краткая характеристика планирования методов резервного копирования.
10. Резервное копирование. Роль журнала транзакций.
11. Виды резервных копий: описание и примеры.
12. Утилиты резервного копирования: назначение и примеры.
13. Воссоздание утраченных файлов. Описание процесса и инструменты.
14. Процесс мониторинга активности. Результаты мониторинга и блокирование угроз.
15. Автоматизированные средства аудита: определение, этапы.
16. Создание резервных копий базы данных: цель и инструменты.
17. Уровни качества программной̆ продукции: этапы, требования к качеству.
18. Характеристика объектов информатизации, требующих обязательной̆ сертификации программных средств и обеспечения.
19. Сертификаты безопасности: виды, функции, срок действия.
20. SSL сертификат. Содержание, формирование запроса.
21. SSL сертификат. Проверка данных с помощью сервисов.
22. Процесс подписи и проверки программного кода.
23. Сертификация соответствия программных средств: требования, этапы.
24. Особенности методов противодействия угрозам безопасности в корпоративных сетях.
25. Задачи при проведении сертификационных испытаний программных продуктов.
26. Обязанности администратора баз данных.

Администратор баз данных (DBA) выполняет важные обязанности, связанные с управлением и обслуживанием баз данных. Вот некоторые из них:

1. **Установка и настройка баз данных**: DBA устанавливает и настраивает системы управления базами данных (СУБД), такие как MySQL, PostgreSQL, Oracle или Microsoft SQL Server.
2. **Мониторинг и оптимизация производительности**: DBA следит за производительностью баз данных, мониторит запросы, оптимизирует индексы и настраивает параметры СУБД для обеспечения эффективной работы.
3. **Резервное копирование и восстановление**: DBA регулярно создает резервные копии данных и разрабатывает планы восстановления в случае сбоев или потери данных.
4. **Безопасность данных**: DBA обеспечивает безопасность данных, управляя доступом пользователей, настройкой прав доступа и шифрованием данных.
5. **Миграция данных**: При необходимости DBA переносит данные между разными СУБД или версиями.
6. **Управление схемой данных**: DBA разрабатывает и поддерживает структуру баз данных, создает таблицы, индексы и связи между ними.
7. **Разрешение проблем**: DBA анализирует и решает проблемы, связанные с базами данных, такие как блокировки, сбои или ошибки.
8. **Обучение и документирование**: DBA обучает пользователей, разрабатывает документацию и поддерживает справочные материалы.
9. Основные функции утилит администратора баз данных.

**Утилиты администрирования баз данных** выполняют важные задачи, связанные с управлением и обслуживанием баз данных. Вот некоторые из них:

1. **SQL Manager for PostgreSQL**: Это программа для разработки и администрирования баз данных сервера PostgreSQL. [Она позволяет управлять таблицами, индексами, пользователями и другими аспектами базы данных1](https://www.sqlmanager.ru/tools/free).
2. **HeidiSQL**: Утилита для управления базами данных, такими как MySQL, MariaDB и PostgreSQL. Она предоставляет графический интерфейс для выполнения запросов, создания таблиц и управления данными.
3. **MySQL Workbench**: Инструмент для администрирования баз данных MySQL. Он включает в себя функции моделирования, мониторинга, резервного копирования и восстановления данных.
4. **Shorewall**: Утилита для управления безопасностью баз данных. Она помогает настроить права доступа, аутентификацию пользователей и обеспечить безопасность данных.
5. **Webmin**: Веб-интерфейс для администрирования различных серверов, включая базы данных. Позволяет управлять пользователями, настройками и другими параметрами.
6. **EMS SQL Manager**: Это мощный инструмент для администрирования баз данных, поддерживающий различные СУБД, такие как MySQL, PostgreSQL, SQL Server и другие.
7. **SQLiteManager**: Утилита для управления базами данных SQLite. Она предоставляет возможности для создания, редактирования и просмотра данных в SQLite-файлах.
8. Краткое описание каждого из этапов режимов запуска и остановы баз данных.

**Режим NOMOUNT**:

* + В этом режиме база данных еще не подключена к физическим файлам данных.
  + Здесь выполняется инициализация экземпляра, но не загружаются данные.
  + [Этот режим используется, например, при создании новой базы данных или восстановлении из резервной копии1](http://www.olap.ru/home.asp?artId=708).

**Режим MOUNT**:

* + В этом режиме база данных подключается к физическим файлам данных, но они еще не открыты для пользователей.
  + Здесь выполняется проверка целостности данных и подготовка к открытию.
  + [Этот режим используется, например, при восстановлении базы данных из резервной копии или при изменении параметров экземпляра1](http://www.olap.ru/home.asp?artId=708).

**Режим OPEN**:

* + В этом режиме база данных полностью доступна для пользователей.
  + Файлы данных открыты, и пользователи могут выполнять запросы и изменять данные.
  + [Этот режим используется в обычной работе с базой данных1](http://www.olap.ru/home.asp?artId=708).

1. Роли пользователей и схемы базы данных.

**Схемы баз данных**:

* + **Схема** — это именованное пространство, в котором хранятся объекты базы данных, такие как таблицы, представления, индексы и процедуры.
  + Схемы позволяют организовать объекты логически и упростить управление ими.
  + Пример: В базе данных “AdventureWorks” может существовать схема “Production” для таблиц, связанных с производством, и схема “Sales” для данных о продажах.

**Владельцы схем**:

* + Каждая схема имеет владельца.
  + Владелец схемы может создавать, изменять и удалять объекты в этой схеме.
  + По умолчанию, владельцем схемы является пользователь, который ее создал.

**Роли пользователей**:

* + Роли определяют набор прав доступа к объектам базы данных.
  + Примеры ролей: “db\_owner” (владелец базы данных), “db\_datareader” (только чтение данных), “db\_datawriter” (только запись данных).
  + Роли позволяют управлять доступом пользователей к объектам.

**Разделение схемы пользователей**:

* + Пользователи могут быть назначены владельцами схем или иметь доступ к схемам.
  + Это позволяет разделить обязанности между пользователями и обеспечить безопасность данных.
  + Например, один пользователь может управлять схемой “HR”, а другой — схемой “Finance”.

**Встроенные схемы**:

* + Некоторые базы данных имеют встроенные схемы, такие как “dbo” (по умолчанию в SQL Server).
  + Встроенные схемы содержат системные объекты, такие как системные таблицы и представления.

1. Привилегии, назначение привилегий пользователей базы данных.

  Системные привилегии (System Privileges):

Системные привилегии контролируют общий доступ к базе данных.

Примеры системных привилегий:

ALL PRIVILEGES: Полный доступ к указанным базам данных.

CREATE: Создание новых баз данных или таблиц.

DROP: Удаление баз данных или таблиц.

DELETE: Удаление строк из таблиц.

INSERT: Вставка строк в таблицы.

Объектные привилегии (Object Privileges):

Объектные привилегии контролируют доступ к конкретным объектам базы данных.

Примеры объектных привилегий:

SELECT: Чтение данных из таблиц.

UPDATE: Изменение данных в таблицах.

REFERENCES: Создание внешних ключей.

INDEX: Создание индексов.

Идентификация и сеансы:

Пользователь базы данных идентифицируется именем пользователя (идентификатором авторизации).

После успешной идентификации открывается сеанс пользователя, и устанавливается соединение с базой данных.

Схемы и владельцы:

Все объекты пользователя входят в его схему.

Владелец схемы может изменять привилегии своих объектов.

Стандарт SQL/92 подразумевает, что одному пользователю может принадлежать несколько схем.

Синтаксис и управление привилегиями:

Синтаксис для работы с привилегиями зависит от конкретной СУБД.

Для управления привилегиями используются команды GRANT (предоставление) и REVOKE (отмена).

1. Схемы и объекты схемы данных. Блоки данных, экстенты сегменты.

**Схемы**:

* + **Схема** представляет собой именованное пространство, в котором хранятся объекты базы данных.
  + В схеме могут содержаться таблицы, представления, индексы и другие объекты.
  + Схемы помогают организовать данные логически и упростить управление ими.

**Объекты схемы**:

* + Объекты схемы — это логические структуры, прямо связанные с данными в базе данных.
  + Примеры объектов схемы:
    - **Таблицы**: Хранят фактические данные, такие как информация о заказах, клиентах и продуктах.
    - **Представления**: Виртуальные таблицы, которые предоставляют удобный доступ к данным из нескольких таблиц.
    - **Индексы**: Ускоряют поиск данных в таблицах.
    - **Процедуры**: Хранят логику обработки данных.

**Блоки данных, экстенты и сегменты**:

* + **Блок данных (Data Block)**: Мельчайший строительный блок базы данных, состоящий из определенного количества байт на диске.
  + **Экстенты (Extent)**: Это два или более последовательных блока данных, представляющих собой единицу выделения места на диске.
  + **Сегменты (Segment)**: Набор экстентов, которые выделены для конкретной логической структуры, например, для таблицы или индекса.

[Сегменты объединяют экстенты, а экстенты — блоки данных, образуя иерархию хранения данных в базе данных](https://www.opennet.ru/docs/RUS/db_admin/)

1. Структуры памяти. Однопроцессорные и многопроцессорные базы данных.

**Структуры памяти базы данных** — это важные компоненты, которые обеспечивают хранение и управление данными в базах данных. Давайте рассмотрим их более подробно:

1. **SGA (System Global Area)**:
   * Глобальная область системы (SGA) — это совместно используемый сегмент памяти, к которому обращаются все процессы базы данных.
   * В SGA хранятся данные, такие как буферы данных, словари, планы выполнения запросов и другие системные структуры.
2. **PGA (Process Global Area)**:
   * Глобальная область процесса (PGA) — это приватная область памяти процесса или потока, недоступная другим процессам.
   * В PGA хранятся данные, специфичные для каждого процесса, например, переменные сессии и результаты выполнения запросов.
3. **UGA (User Global Area)**:
   * Глобальная область пользователя (UGA) — это область памяти сессии пользователя.
   * UGA может находиться либо в SGA, либо в PGA, в зависимости от режима работы сервера (распределенный или выделенный).

[Структуры памяти обеспечивают эффективное управление данными и обеспечивают быстрый доступ к информации в базах данных](https://practicum.yandex.ru/blog/10-osnovnyh-struktur-dannyh/)

**Однопроцессорные базы данных**:

* + В однопроцессорных системах используется **один физический процессор** для выполнения задач.
  + Примером однопроцессорной архитектуры является **архитектура фон Неймана**, где процессор последовательно выбирает и обрабатывает команды программы и данные.
  + Современные ЭВМ, такие как компьютеры IBM PC, относятся к **CISC (Complex Instruction Set Computer)** архитектуре, где используется сложная система команд.
  + Еще одной разновидностью однопроцессорной архитектуры является **суперскалярная обработка**, позволяющая одновременно выполнять несколько операций.

**Многопроцессорные базы данных**:

* + В многопроцессорных системах используется **несколько физических процессоров** для обработки данных.
  + Преимущество многопроцессорных систем — возможность параллельной обработки задач.
  + Многопроцессорные системы могут быть симметричными (SMP), где все процессоры равноправны, или асимметричными (AMP), где есть главный процессор и вспомогательные.
  + Суперкомпьютеры и серверы с множеством процессоров — примеры многопроцессорных систем.

1. Транзакции, блокировки и согласованность данных: описание и назначение.

**Транзакции** в базах данных представляют собой неделимые последовательности операций, которые либо выполняются полностью, либо откатываются целиком. Они обеспечивают целостность данных и гарантируют, что база данных остается в согласованном состоянии.

Вот некоторые ключевые аспекты транзакций:

1. **Свойства транзакций (ACID)**:
   * **Атомарность (Atomicity)**: Транзакция либо выполняется полностью, либо не выполняется вообще.
   * **Согласованность (Consistency)**: Транзакция переводит базу данных из одного целостного состояния в другое.
   * **Изолированность (Isolation)**: Транзакции выполняются параллельно, но изолированно друг от друга.
   * **Устойчивость (Durability)**: Результаты успешно завершенных транзакций сохраняются даже после сбоев.
2. **Блокировки**:
   * Транзакции накладывают блокировки на данные, чтобы обеспечить выполнение требований ACID.
   * Блокировки предотвращают конфликты при одновременном доступе к данным.
3. **Уровни изоляции**:
   * Определяют, как транзакции видят изменения, внесенные другими транзакциями.
   * Примеры уровней изоляции: **READ COMMITTED**, **REPEATABLE READ**, **SERIALIZABLE**.
4. Журнал базы данных: структура и назначение файлов журнала, управление переключениями и контрольными точками.

**Журнал транзакций** в базе данных SQL Server играет критически важную роль, записывая все транзакции и изменения, внесенные каждой транзакцией. Давайте рассмотрим его архитектуру и функции более подробно:

1. **Логическая архитектура журнала транзакций**:
   * Журнал транзакций SQL Server работает логически, как если бы он был строкой записей журналов.
   * Каждая запись журнала определяется порядковым номером журнала (LSN). Новые записи добавляются в логический конец журнала с LSN, который больше номера LSN предыдущей записи.
   * Записи журнала связаны в цепочку по обратным указателям, ускоряя откат транзакции.
   * Каждая запись журнала содержит идентификатор транзакции, к которой она относится.
   * Пример LSN: 00000031:00000da0:0001, где 0x31 - идентификатор VLF, 0xda0 - идентификатор блока журнала, 0x1 - первая запись журнала в этом блоке.
   * Записи журнала для изменений данных описывают либо выполненную логическую операцию, либо изображения измененных данных до и после операции.
2. **Физическая архитектура журнала транзакций**:
   * Журнал транзакций выполняется как отдельный файл или набор файлов в базе данных.
   * Кэш журналов управляется отдельно от кэша буфера для страниц данных.
   * Журналы обеспечивают простоту, быстроту и надежность восстановления базы данных.
3. **Управление журналом транзакций**:
   * Журнал транзакций можно управлять с помощью следующих операций:
     1. **Резервное копирование журнала**: Регулярное резервное копирование журнала позволяет сохранять его состояние. Это важно для восстановления базы данных после сбоя.
     2. **Переключение журнала**: Переключение журнала происходит при переходе к новому журналу после резервного копирования. Это обеспечивает непрерывность записи транзакций.
     3. **Контрольные точки**: Контрольные точки создаются для уменьшения времени восстановления. Они записывают текущее состояние базы данных и журнала.
     4. **Усечение журнала**: Усечение удаляет старые записи журнала, освобождая место. Это важно для управления размером журнала.
4. **Роли журнала транзакций**:
   * **Журнал восстановления**: Записывает все изменения данных, чтобы обеспечить восстановление базы данных.
   * **Журнал репликации**: Используется для репликации данных между серверами.
   * **Журнал регистрации**: Содержит информацию о событиях, таких как входы и ошибки.
5. **Применение журнала транзакций**:
   * Восстановление базы данных после сбоя.
   * Восстановление до определенной точки во времени.
   * Восстановление на другой сервер с помощью репликации.
6. Словарь данных: назначение, структура, префиксы.

Словарь данных (или data dictionary) - это структурированное хранилище информации о данных, используемых в базе данных или другом информационном системе. Он описывает схему базы данных, определяет свойства и характеристики данных, и предоставляет метаданные, необходимые для правильного использования и управления данными. Вот некоторые ключевые аспекты словаря данных:

Назначение:

Описывает структуру базы данных: таблицы, поля, индексы, связи между таблицами и другие элементы.

Содержит описание типов данных, диапазонов значений, правил проверки и других ограничений.

Обеспечивает документацию по данным, что упрощает понимание структуры и использование базы данных.

Структура:

Таблицы и поля: Описывает все таблицы базы данных и их поля, включая типы данных, размеры и связи между таблицами.

Ограничения: Включает ограничения целостности данных, такие как первичные ключи, внешние ключи, уникальные ограничения и правила проверки.

Индексы: Содержит информацию о созданных индексах для ускорения запросов к данным.

Префиксы:

Префикс данных: Обозначает первые несколько символов или байт в поле, который может использоваться для категоризации или идентификации типа данных.

Префикс таблицы: Используется для добавления префикса к именам таблиц, чтобы избежать конфликтов имен в случае использования нескольких баз данных или схем в одной системе.

Словарь данных играет важную роль в поддержке и управлении базами данных, обеспечивая структурированную документацию и средства контроля за данными.

1. Правила Дейта для оценки СУБД.

Правила Дейта (или Codd's Rules) — это набор принципов, предложенных Эдгаром Коддом, который считается одним из основателей реляционных баз данных. Эти правила определяют стандарты, которым должны соответствовать реляционные системы управления базами данных (СУБД). Вот основные правила:

Информационная правила (Information Rule):

Вся информация в базе данных должна быть представлена в виде значений в ячейках таблицы, а не в виде адресов файлов или указателей.

Правило гарантированного доступа (Guaranteed Access Rule):

Каждый элемент данных должен быть доступен по использованию комбинации значения первичного ключа (Primary Key) и имени таблицы.

Система каталогов (Systematic Treatment of Null Values):

База данных должна поддерживать представление отсутствующих данных (NULL), и это должно быть сделано систематически.

Динамическая онлайн-каталогизация на основе отношений (Dynamic Online Catalog Based on The Relational Model):

Метаданные о базе данных, такие как схема, должны храниться в виде отношений и могут быть изменены таким же образом, как и обычные данные.

Язык данных высокого уровня (Comprehensive Data Sublanguage Rule):

Должен существовать язык запросов, который является полным, позволяет выразить любой запрос, который можно выразить в реляционной алгебре, и который может быть использован как для определения данных, так и для их манипуляций.

Поддержка представления (View Updating Rule):

Все представления, которые могут быть определены на основе базы данных, могут быть обновлены и модифицированы с использованием тех же средств, что и основная таблица.

Физическая независимость данных (Physical Data Independence):

Изменения в физической организации данных (например, индексы или порядок хранения) не должны влиять на приложения, использующие базу данных.

Логическая независимость данных (Logical Data Independence):

Изменения в логической структуре данных, такие как добавление или удаление таблиц, не должны влиять на приложения, использующие базу данных.

Целостность (Integrity Independence):

Вся целостность данных, включая ограничения целостности, должна быть определена независимо от приложений, использующих базу данных, и должна быть поддержана самой системой управления базой данных.

Соблюдение этих правил обеспечивает эффективность и надежность работы реляционных СУБД, а также упрощает разработку и поддержку баз данных.

1. Классификация серверов: описание и свойства.

Серверы могут быть классифицированы по различным критериям, включая их назначение, функциональность, характеристики обработки данных, архитектуру и другие параметры. Вот общая классификация серверов с описанием и основными свойствами:

Веб-серверы:

Назначение: Обслуживание запросов клиентов по протоколу HTTP/HTTPS, обеспечение доступа к веб-ресурсам (веб-сайтам).

Свойства: Обработка статических и динамических контентов, поддержка SSL/TLS для безопасной передачи данных.

Файловые серверы:

Назначение: Хранение и предоставление доступа к файлам, обмен файлами между пользователями в локальной сети.

Свойства: Обеспечение доступа к файлам по различным протоколам (FTP, SMB/CIFS), управление правами доступа.

Базы данных серверы:

Назначение: Хранение, управление и предоставление доступа к данным баз данных.

Свойства: Поддержка SQL-запросов, обеспечение целостности данных, масштабируемость.

Почтовые серверы:

Назначение: Передача, прием, хранение и доставка электронной почты.

Свойства: Поддержка протоколов SMTP, POP3, IMAP, антиспам и антивирусные меры.

Приложенческие серверы:

Назначение: Выполнение специфических функций для поддержки приложений.

Свойства: Поддержка определенных приложений, например, приложенческие серверы для обработки транзакций или серверы для обработки мультимедийных данных.

DNS-серверы:

Назначение: Преобразование доменных имен в IP-адреса и наоборот.

Свойства: Обеспечение распределенной иерархической структуры для разрешения доменных имен.

Прокси-серверы:

Назначение: Посредничество при передаче запросов и ответов между клиентами и другими серверами.

Свойства: Кэширование данных, фильтрация трафика, повышение безопасности.

Печатные серверы:

Назначение: Управление и обеспечение доступа к сетевым принтерам.

Свойства: Распределение печатных задач, мониторинг статуса принтеров.

Виртуальные серверы:

Назначение: Запуск и управление виртуальными машинами на физическом сервере.

Свойства: Эффективное использование ресурсов, возможность масштабирования и изоляции приложений.

Кластерные серверы:

Назначение: Объединение нескольких серверов в кластер для обеспечения отказоустойчивости и балансировки нагрузки.

Свойства: Автоматическое восстановление после сбоев, распределение нагрузки между узлами.

1. Описание протоколов удаленного вызова процедур.

В контексте баз данных, протоколы удаленного вызова процедур (Remote Procedure Call, RPC) обеспечивают возможность вызова хранимых процедур или функций на удаленных серверах баз данных. Это позволяет клиентским приложениям выполнять операции на базе данных без необходимости наличия кода процедур на клиентской стороне. Вот несколько примеров протоколов RPC, которые могут использоваться в контексте баз данных:

ODBC (Open Database Connectivity):

Описание: ODBC представляет собой стандартный интерфейс доступа к базам данных. Он может включать в себя возможность удаленного вызова процедур через спецификации SQL, такие как EXECUTE PROCEDURE или эквивалентные.

JDBC (Java Database Connectivity):

Описание: Аналогично ODBC, но предназначен для языка программирования Java. Включает в себя возможности для вызова процедур на сервере базы данных.

ADO.NET:

Описание: Используется в Microsoft .NET Framework. ADO.NET позволяет клиентским приложениям вызывать хранимые процедуры на сервере баз данных.

PL/SQL (Procedural Language/Structured Query Language):

Описание: Протокол для вызова процедур в базах данных Oracle. PL/SQL может использоваться для определения и вызова хранимых процедур на сервере Oracle.

TDS (Tabular Data Stream):

Описание: Протокол, используемый в Microsoft SQL Server. Позволяет клиентам SQL Server вызывать хранимые процедуры на сервере.

DBRPC (Database Remote Procedure Call):

Описание: Некоторые базы данных могут предоставлять свои собственные протоколы RPC для удаленного вызова процедур. Например, MySQL может использовать свой собственный протокол для этой цели.

SOAP и RESTful веб-службы:

Описание: Веб-службы, построенные на протоколах SOAP или REST, могут предоставлять интерфейс для удаленного вызова процедур в базах данных. Обычно, это реализуется с использованием HTTP-протокола.

Каждая база данных или базовый протокол может иметь свои собственные спецификации для удаленного вызова процедур. Важно учитывать безопасность и производительность при использовании RPC в базах данных, особенно в распределенных средах.

1. Требования к аппаратным возможностям и базовому программному обеспечению клиентов и серверов.

Требования к аппаратным возможностям и базовому программному обеспечению клиентов и серверов базы данных могут варьироваться в зависимости от конкретной системы управления базами данных (СУБД), приложения и потребностей организации. Ниже приведены общие рекомендации для клиентов и серверов:

Требования к клиентским компьютерам:

Процессор:

Минимальные: Двухъядерный процессор с тактовой частотой не менее 2 ГГц.

Рекомендуемые: Четырехъядерный процессор с тактовой частотой не менее 2.5 ГГц.

Оперативная память (RAM):

Минимальные: 4 ГБ оперативной памяти.

Рекомендуемые: 8 ГБ и более.

Хранение данных:

Доступное место на жестком диске для установки клиентского программного обеспечения и временного хранения данных.

Сетевая карта:

100 Мбит/с или выше для эффективного взаимодействия с сервером базы данных.

Операционная система:

Поддерживаемая версия операционной системы (Windows, Linux, macOS и т. д.).

Программное обеспечение:

Версия и настройки браузера для веб-интерфейсов.

Клиенты для доступа к конкретным СУБД (если требуется).

Требования к серверам баз данных:

Процессор:

Минимальные: Четырехъядерный процессор с тактовой частотой не менее 2.5 ГГц.

Рекомендуемые: Восьмиядерный процессор с тактовой частотой не менее 3 ГГц.

Оперативная память (RAM):

Минимальные: 16 ГБ оперативной памяти.

Рекомендуемые: 32 ГБ и более.

Хранение данных:

Достаточное пространство на жестком диске для хранения баз данных и их резервных копий.

Поддержка RAID для повышения отказоустойчивости и производительности.

Сетевая карта:

Гигабитная сетевая карта для обеспечения высокой пропускной способности при обращении к базе данных из различных клиентов.

Операционная система:

Поддерживаемая и оптимизированная для конкретной СУБД (Windows Server, Linux, UNIX и т. д.).

Программное обеспечение:

Установленная и настроенная СУБД с учетом требований приложения.

Системы резервного копирования и восстановления:

Механизмы резервного копирования и восстановления для обеспечения безопасности данных.

Мониторинг и управление ресурсами:

Инструменты для мониторинга и управления ресурсами сервера (например, системы мониторинга производительности, системы резервирования).

Защитные меры:

Антивирусное программное обеспечение, брандмауэр и другие меры безопасности для защиты от угроз и несанкционированного доступа.

Эти требования являются общими и могут быть адаптированы в соответствии с конкретными потребностями и характеристиками каждого проекта и организации.

1. Хранимые процедуры и триггеры: определение и применение.

Хранимые процедуры и триггеры - это объекты базы данных, предназначенные для хранения и выполнения программного кода внутри самой базы данных. Они предоставляют средства для обработки данных и автоматизации определенных операций. Вот их определение и применение:

1. Хранимые процедуры:

Определение:

Хранимая процедура (Stored Procedure) - это подготовленный заранее и сохраненный в базе данных набор инструкций SQL, который можно вызвать и выполнить в любое время.

Применение:

Оптимизация и централизация логики: Хранимые процедуры позволяют централизованно хранить и управлять логикой обработки данных внутри базы данных.

Уменьшение трафика по сети: Выполнение операций на стороне сервера может сократить объем передаваемых данных по сети.

Безопасность: Хранимые процедуры могут быть предоставлены пользователям с ограниченными правами доступа, что повышает безопасность данных.

Повторное использование кода: Одна и та же логика может быть использована в различных частях приложения.

2. Триггеры:

Определение:

Триггер (Trigger) - это объект базы данных, который автоматически выполняет определенные действия (например, выполнение хранимой процедуры) при возникновении определенного события (например, вставка, обновление или удаление записи).

Применение:

Соблюдение целостности данных: Триггеры могут использоваться для проверки или модификации данных, обеспечивая соблюдение целостности данных.

Логирование изменений: Триггеры могут фиксировать изменения в данных для целей аудита и восстановления.

Реализация бизнес-логики: Триггеры могут быть использованы для автоматической обработки данных в ответ на определенные события.

Автоматизация задач: Триггеры могут автоматически выполнять определенные задачи при изменениях в данных, что уменьшает необходимость вручную управлять процессами.

Оба этих объекта базы данных предоставляют мощные инструменты для оптимизации работы с данными, повышения безопасности и обеспечения более эффективного управления базой данных в целом.

1. Понятия и основные характеристики серверов баз данных.

Сервер баз данных (DBMS Server): Это специализированное программное обеспечение, предназначенное для управления базами данных (БД). Сервер баз данных обеспечивает доступ к данным, их хранение, обработку, безопасность и другие функции для клиентских приложений.

Основные понятия и характеристики серверов баз данных:

Хранение данных:

Определение: Сервер баз данных отвечает за организацию и хранение данных в структурированной форме.

Характеристики: Эффективное управление хранилищем данных, поддержка различных типов данных и индексов.

Язык запросов:

Определение: Предоставляет средства для выполнения запросов к данным. SQL (Structured Query Language) часто используется для этого.

Характеристики: Поддержка языков запросов, оптимизация выполнения запросов.

Транзакции:

Определение: Гарантируют целостность данных в базе при одновременном доступе нескольких пользователей.

Характеристики: ACID-свойства (Атомарность, Согласованность, Изолированность, Долговечность).

Безопасность:

Определение: Обеспечивает контроль доступа к данным и защиту от несанкционированного доступа.

Характеристики: Аутентификация, авторизация, аудит, шифрование данных.

Оптимизация запросов:

Определение: Процесс оптимизации выполнения запросов для повышения производительности.

Характеристики: Планы выполнения запросов, индексы, статистика.

Резервное копирование и восстановление:

Определение: Гарантирует безопасность данных с помощью регулярного создания резервных копий и возможности восстановления.

Характеристики: Механизмы создания резервных копий, стратегии восстановления.

Масштабируемость:

Определение: Способность системы масштабироваться для обработки роста объема данных и запросов.

Характеристики: Вертикальное и горизонтальное масштабирование.

Системы управления версиями данных:

Определение: Возможность отслеживать и управлять изменениями данных с течением времени.

Характеристики: Журнализация изменений, системы версионирования.

Работа с многопользовательским доступом:

Определение: Поддержка одновременного доступа нескольких пользователей к данным.

Характеристики: Управление транзакциями, изоляция данных.

Контроль целостности:

Определение: Обеспечивает целостность данных, предотвращает нарушения правил и ограничений.

Характеристики: Ограничения целостности, проверки целостности.

Управление ресурсами:

Определение: Эффективное управление памятью, процессорными ресурсами и сетевым взаимодействием.

Характеристики: Мониторинг производительности, оптимизация ресурсов.

Серверы баз данных являются критическими компонентами информационных систем и предоставляют базовую инфраструктуру для работы приложений и хранения данных в организации.

1. Механизмы доступа к базам данных.

Механизмы доступа к базам данных определяют, как приложения и пользователи могут получать доступ к данным в базе данных. Существует несколько основных механизмов доступа к базам данных:

Язык SQL (Structured Query Language):

Описание: SQL является стандартным языком запросов для работы с реляционными базами данных. С его помощью пользователи и приложения могут выполнять запросы, добавлять, обновлять и удалять данные, создавать и изменять структуру базы данных.

Характеристики: Простота в использовании, стандартизированный интерфейс, мощные возможности для манипулирования данными.

ODBC (Open Database Connectivity):

Описание: ODBC предоставляет стандартный интерфейс для взаимодействия приложений с различными СУБД. Он позволяет приложениям использовать SQL для доступа к данным независимо от конкретной базы данных.

Характеристики: Универсальность, возможность использования в различных ОС и языках программирования.

JDBC (Java Database Connectivity):

Описание: Аналогично ODBC, но предназначен для языка программирования Java. JDBC обеспечивает стандартизированный способ взаимодействия Java-приложений с базами данных.

Характеристики: Поддержка Java-приложений, использование в многих СУБД.

ORM (Object-Relational Mapping):

Описание: ORM-фреймворки, такие как Hibernate, Entity Framework, предоставляют абстракцию от реляционной модели данных, позволяя взаимодействовать с базой данных через объекты в коде.

Характеристики: Упрощение работы с базой данных, объектно-ориентированный подход.

RESTful API (Representational State Transfer):

Описание: RESTful API позволяет взаимодействовать с базой данных через HTTP-протокол, используя различные методы запросов (GET, POST, PUT, DELETE).

Характеристики: Простота в использовании, поддержка множества языков программирования, архитектурная гибкость.

GraphQL:

Описание: GraphQL - это язык запросов и среда выполнения для API, который позволяет клиентам запрашивать только те данные, которые им нужны.

Характеристики: Гибкость запросов, уменьшение объема передаваемых данных, использование в современных веб-приложениях.

Программные библиотеки и драйверы:

Описание: Многие СУБД предоставляют официальные или сторонние программные библиотеки и драйверы для взаимодействия с базой данных из кода приложения.

Характеристики: Специфичные возможности, оптимизированные для конкретной СУБД.

Выбор конкретного механизма доступа зависит от требований приложения, особенностей базы данных, языка программирования и архитектуры системы.

1. Банк данных: состав, схема.

Термин "банк данных" не имеет устоявшегося стандартного значения и может использоваться в различных контекстах. Однако, в общем смысле, он обычно означает организованное хранилище данных или базу данных, которое используется для хранения, управления и обработки информации. Давайте рассмотрим основные компоненты и структуру банка данных:

Состав банка данных:

Таблицы (Tables):

Описание: Таблицы представляют собой основной способ организации данных в базе данных. Они состоят из строк (записей) и столбцов (полей), где каждая строка представляет собой отдельную запись, а каждый столбец - поле с определенным типом данных.

Пример: Таблица "Сотрудники" с полями "Имя", "Фамилия", "Зарплата", и т. д.

Отношения (Relationships):

Описание: Отношения устанавливают связи между таблицами в базе данных. Они определяют, как данные в одной таблице связаны с данными в другой таблице.

Пример: Отношение между таблицей "Сотрудники" и таблицей "Отделы" на основе поля "Отдел\_ID".

Индексы (Indexes):

Описание: Индексы ускоряют поиск данных в таблицах, обеспечивая быстрый доступ к определенным значениям.

Пример: Индекс на столбце "ИД\_Сотрудника" в таблице "Сотрудники".

Хранимые процедуры (Stored Procedures) и Триггеры (Triggers):

Описание: Хранимые процедуры - это предварительно скомпилированные блоки кода SQL, которые можно вызвать для выполнения определенных операций. Триггеры - это автоматически выполняемые действия, связанные с изменением данных в таблицах.

Пример: Хранимая процедура для расчета средней зарплаты. Триггер для автоматического обновления даты последнего изменения записи.

Представления (Views):

Описание: Представления представляют собой виртуальные таблицы, основанные на результатах запросов. Они обеспечивают удобный способ просмотра и использования данных без изменения базовой структуры таблиц.

Пример: Представление, отображающее информацию о сотрудниках и их зарплате.

Схема банка данных:

Схема базы данных - это графическое представление структуры и организации данных в базе данных. Она включает в себя таблицы, связи, индексы и другие объекты. Схема предоставляет визуальное представление того, как данные организованы и как они взаимодействуют друг с другом.

Пример схемы банка данных:

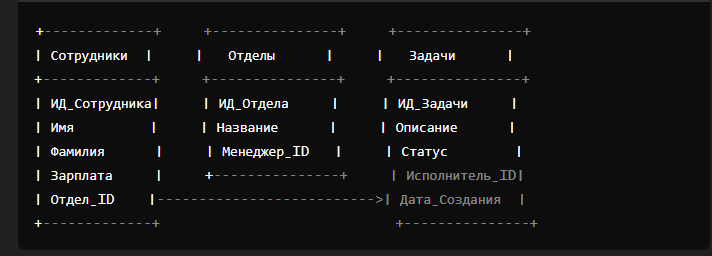
В этом примере:

Таблица "Сотрудники" связана с таблицей "Отделы" через поле "Отдел\_ID".

Таблица "Задачи" содержит внешние ключи (ИД\_Сотрудника, ИД\_Отдела, Исполнитель\_ID), связанные с таблицей "Сотрудники".

Схема банка данных помогает лучше понять структуру данных и их взаимосвязи, что является важным при проектировании и поддержке баз данных.

1. Технология установки и настройка сервера PostgreSQL в операционной системе Windows.

Установка и настройка сервера PostgreSQL в операционной системе Windows — это относительно простая процедура, осуществляемая с использованием официального установщика. Вот пошаговая инструкция:

Установка PostgreSQL на Windows:

Скачивание установщика:

Перейдите на официальный сайт PostgreSQL: https://www.postgresql.org/download/windows/.

Выберите версию PostgreSQL, подходящую для вашей операционной системы (32-бит или 64-бит).

Скачайте установщик.

Запуск установщика:

Запустите скачанный установщик PostgreSQL.

Выбор компонентов:

Выберите компоненты для установки. Обычно это включает в себя PostgreSQL Database Server и утилиты командной строки.

Выбор директории установки:

Укажите директорию, в которую будет установлен PostgreSQL.

Выбор данных пользователя:

Выберите пользователя и пароль для суперпользователя PostgreSQL (по умолчанию - postgres).

Выбор порта:

Выберите порт, на котором PostgreSQL будет слушать. По умолчанию, это 5432.

Выбор локали:

Выберите локаль (язык, формат даты и времени) для вашей базы данных.

Выбор службы:

Выберите, как служба PostgreSQL будет запускаться: автоматически при старте системы или вручную.

Установка дополнительных компонентов:

Выберите, установить ли pgAdmin (графический интерфейс администрирования для PostgreSQL) и Stack Builder (инструмент для установки дополнительных компонентов).

Завершение установки:

Дождитесь завершения установки.

Настройка PostgreSQL на Windows:

Настройка переменной PATH:

Добавьте путь к бинарным файлам PostgreSQL в переменную PATH. Обычно это C:\Program Files\PostgreSQL\<version>\bin.

Запуск службы:

После установки PostgreSQL служба должна запуститься автоматически. Если это не произошло, запустите службу вручную. Можно сделать это через службы Windows (напишите services.msc в командной строке).

Проверка подключения:

Откройте командную строку или PowerShell и выполните команду psql -U postgres. Введите пароль, который вы установили во время установки.

Если вы видите приглашение PostgreSQL (postgres=#), значит, подключение прошло успешно.

Настройка pgAdmin:

Если вы установили pgAdmin, запустите его и добавьте новый сервер. Введите имя сервера, хост (обычно localhost), порт (по умолчанию 5432), имя пользователя и пароль.

Создание базы данных:

Используя pgAdmin или командную строку, создайте базу данных и пользователя по вашему усмотрению.

Это базовая инструкция, и в зависимости от конкретных требований вашего проекта могут потребоваться дополнительные настройки. Будьте внимательны к документации PostgreSQL и рекомендациям по безопасности.

1. Клиентские настройки, протоколирование, безопасность.

 Настройка клиентских параметров, протоколирование и обеспечение безопасности баз данных играют критическую роль в обеспечении надежности и безопасности вашей системы управления базами данных (СУБД). Вот некоторые основные аспекты этих процессов:

1. Клиентские настройки:

Настройка подключения:

Клиенты могут настраиваться для подключения к базе данных с использованием определенных параметров, таких как имя хоста, порт, имя пользователя и пароль.

Настройка клиентских приложений:

Клиентские приложения должны быть настроены таким образом, чтобы использовать правильные учетные данные и параметры подключения к базе данных.

Кеширование:

Некоторые клиентские настройки могут включать параметры кеширования, которые позволяют клиентам кэшировать запросы или данные для улучшения производительности.

2. Протоколирование:

Журналирование событий:

Системы управления базами данных обычно предоставляют возможность протоколирования различных событий, таких как успешные и неудачные попытки входа, выполнение SQL-запросов, изменения данных и т. д.

Анализ журналов:

Журналы событий могут использоваться для анализа действий пользователей, обнаружения проблем, аудита и соблюдения требований безопасности.

3. Безопасность баз данных:

Аутентификация:

СУБД обычно предоставляют механизмы аутентификации пользователей, такие как пароли, сертификаты, аутентификация на основе токенов и т. д.

Авторизация:

Настройка прав доступа пользователей к объектам базы данных (таблицам, представлениям, процедурам и т. д.) и определение, какие действия они могут выполнять.

Шифрование:

Шифрование данных в пути и в состоянии покоя для защиты от несанкционированного доступа к конфиденциальной информации.

Управление ролями:

Использование ролей для управления набором привилегий и разграничения доступа пользователей к данным в базе данных.

Мониторинг и обнаружение угроз:

Регулярное мониторинг и обнаружение потенциальных угроз безопасности, таких как атаки внутри или снаружи сети, внутренние угрозы и т. д.

Резервное копирование и восстановление:

Регулярное создание резервных копий баз данных для обеспечения возможности восстановления данных в случае чрезвычайных ситуаций или инцидентов безопасности.

Обеспечение безопасности баз данных — это комплексный процесс, требующий не только правильной конфигурации СУБД, но и осознания угроз и мер по их предотвращению и обнаружению.

1. Технология установки и настройка сервера PostgreSQL в операционных системах Linux.

Установка и настройка сервера PostgreSQL в операционных системах Linux может быть выполнена с использованием пакетных менеджеров, что облегчает процесс. Вот шаги для установки и настройки PostgreSQL на Linux:

Для Debian/Ubuntu:

Обновление пакетов:

bash

Copy code

sudo apt-get update

Установка PostgreSQL:

bash

Copy code

sudo apt-get install postgresql postgresql-contrib

Настройка пользователя postgres:

bash

Copy code

sudo -u postgres psql

Создание пользователя и базы данных:

sql

Copy code

CREATE USER your\_username WITH PASSWORD 'your\_password';

CREATE DATABASE your\_database;

ALTER DATABASE your\_database OWNER TO your\_username;

Разрешение удаленных подключений (опционально):

Редактируйте файл postgresql.conf и pg\_hba.conf для разрешения удаленных подключений (если это необходимо).

Перезапуск PostgreSQL:

bash

Copy code

sudo service postgresql restart

Для Red Hat/CentOS:

Обновление пакетов:

bash

Copy code

sudo yum update

Установка PostgreSQL:

bash

Copy code

sudo yum install postgresql-server postgresql-contrib

Инициализация кластера и запуск службы:

bash

Copy code

sudo postgresql-setup initdb

sudo systemctl start postgresql

Настройка пользователя postgres:

bash

Copy code

sudo -u postgres psql

Создание пользователя и базы данных:

sql

Copy code

CREATE USER your\_username WITH PASSWORD 'your\_password';

CREATE DATABASE your\_database;

ALTER DATABASE your\_database OWNER TO your\_username;

Разрешение удаленных подключений (опционально):

Редактируйте файл postgresql.conf и pg\_hba.conf для разрешения удаленных подключений (если это необходимо).

Перезапуск PostgreSQL:

bash

Copy code

sudo systemctl restart postgresql

Общие дополнительные шаги:

Изменение пароля пользователя postgres (если необходимо):

bash

Copy code

sudo -u postgres psql -c "ALTER USER postgres WITH PASSWORD 'new\_password';"

Настройка доступа к PostgreSQL извне (опционально):

Редактирование файла pg\_hba.conf и postgresql.conf для указания IP-адресов и методов аутентификации.

Установка клиента psql (опционально):

bash

Copy code

sudo apt-get install postgresql-client # Для Debian/Ubuntu

sudo yum install postgresql # Для Red Hat/CentOS

Эти инструкции предоставляют базовую установку и настройку PostgreSQL. Убедитесь в том, что вы также выполняете шаги по обеспечению безопасности, такие как настройка брандмауэра, шифрование соединений и управление пользователями и правами доступа в зависимости от ваших конкретных требований.

1. Аудит базы данных. Аудиторский журнал. Установка опций, включение и отключение аудита. Очистка и уменьшение размеров журнала.

Аудит баз данных представляет собой процесс отслеживания и регистрации событий, происходящих в базе данных. Это важная часть обеспечения безопасности, а также соблюдения требований законодательства и стандартов. Аудит позволяет мониторить активность пользователей, выявлять потенциальные угрозы безопасности и следить за соответствием правилам доступа. Вот некоторые ключевые аспекты аудита баз данных:

1. Виды аудита:

Аудит доступа:

Отслеживание попыток входа в систему, изменений прав доступа и выполнения привилегированных операций.

Аудит изменений данных:

Регистрация изменений в данных, таких как вставка, обновление и удаление записей.

Аудит структуры базы данных:

Отслеживание изменений в схеме базы данных, таких как создание, изменение или удаление таблиц, индексов и представлений.

Аудит сетевой активности:

Мониторинг сетевого трафика, связанного с базой данных, для выявления потенциальных атак или несанкционированных подключений.

Аудит выполнения SQL-запросов:

Отслеживание выполнения SQL-запросов для выявления необычной или подозрительной активности.

2. Инструменты аудита:

Журналирование (Logging):

Система журналирования событий базы данных регистрирует различные события в специальных журналах (логах), которые затем могут быть проанализированы.

Системы мониторинга и управления (Database Management Systems - DBMS):

Многие DBMS предоставляют встроенные инструменты аудита, такие как Oracle Audit Vault, Microsoft SQL Server Audit, и другие.

Внешние инструменты аудита:

Существуют также сторонние инструменты аудита, которые могут предоставить дополнительные функции и аналитику.

3. Журналы аудита:

Журнал безопасности (Security Log):

Регистрирует события, связанные с безопасностью, такие как попытки входа в систему, изменения привилегий и т. д.

Журнал изменений данных (Transaction Log):

Содержит информацию об изменениях в данных, выполняемых транзакциями.

Журнал структурных изменений (DDL Log):

Фиксирует изменения в структуре базы данных, такие как создание, изменение или удаление объектов.

4. Обработка событий:

Оповещения и тревоги (Alerts):

Настройка оповещений для мгновенного уведомления о важных событиях или потенциальных угрозах.

Автоматизированные реакции (Automated Responses):

Создание автоматизированных процессов реагирования на определенные события, такие как блокировка учетных записей при неудачных попытках входа.

5. Соблюдение требований:

Соответствие законодательству и стандартам:

Аудит баз данных может быть необходим для соблюдения законодательства, такого как GDPR, HIPAA, PCI DSS и др.

Аудит для внутреннего контроля:

Обеспечение соблюдения внутренних политик и процедур организации.

6. Управление аудитом:

Настройка параметров аудита:

Определение, какие события следует аудитировать, и какие данные включать в журналы аудита.

Регулярный мониторинг и анализ:

Регулярное просмотр и анализ журналов аудита для выявления аномалий и потенциальных угроз.

Резервное копирование данных аудита:

Обеспечение сохранности данных аудита для последующего анализа и аудита.

Эффективный аудит баз данных содействует обеспечению безопасности, выявлению угроз и соблюдению требований законодательства, что важно для бизнеса и защиты конфиденциальных данных.

Аудиторский журнал базы данных (БД) представляет собой документацию о действиях и событиях, происходящих в системе управления базами данных (СУБД). Этот журнал используется для отслеживания и регистрации различных событий в целях обеспечения безопасности, мониторинга активности пользователей, а также соблюдения требований законодательства и политик безопасности организации.

Установка, включение и отключение аудита в базе данных зависят от используемой системы управления базами данных (СУБД). Рассмотрим примеры для двух популярных СУБД: PostgreSQL и Microsoft SQL Server.

1. PostgreSQL:

Установка опций аудита:

В PostgreSQL аудит может быть реализован с использованием расширения, такого как "pg\_audit". Давайте предположим, что вы уже установили это расширение:

Установка расширения:

sql

Copy code

CREATE EXTENSION IF NOT EXISTS pg\_audit;

Создание таблицы для хранения аудиторских записей:

sql

Copy code

CREATE TABLE IF NOT EXISTS audit\_log (

audit\_event jsonb,

audit\_time timestamp

);

Включение и отключение аудита:

Включение аудита для конкретной таблицы:

sql

Copy code

SELECT pg\_audit.table\_set\_audit('your\_table\_name', true);

Отключение аудита для конкретной таблицы:

sql

Copy code

SELECT pg\_audit.table\_set\_audit('your\_table\_name', false);

Включение аудита для всех таблиц в схеме:

sql

Copy code

SELECT pg\_audit.schema\_set\_audit('your\_schema\_name', true);

Отключение аудита для всех таблиц в схеме:

sql

Copy code

SELECT pg\_audit.schema\_set\_audit('your\_schema\_name', false);

2. Microsoft SQL Server:

Включение и отключение аудита:

Включение аудита для базы данных:

sql

Copy code

USE your\_database;

CREATE DATABASE AUDIT SPECIFICATION YourAuditSpecification

FOR SERVER AUDIT YourServerAudit

ADD (DATABASE\_OBJECT\_CHANGE\_GROUP);

ALTER DATABASE AUDIT SPECIFICATION YourAuditSpecification WITH (STATE = ON);

Отключение аудита для базы данных:

sql

Copy code

USE your\_database;

ALTER DATABASE AUDIT SPECIFICATION YourAuditSpecification WITH (STATE = OFF);

Включение аудита для объекта (например, таблицы):

sql

Copy code

USE your\_database;

CREATE DATABASE AUDIT SPECIFICATION YourAuditSpecification

FOR SERVER AUDIT YourServerAudit

ADD (SCHEMA\_OBJECT\_CHANGE\_GROUP)

WITH (OBJECT\_NAME = 'your\_table\_name');

ALTER DATABASE AUDIT SPECIFICATION YourAuditSpecification WITH (STATE = ON);

Отключение аудита для объекта (например, таблицы):

sql

Copy code

USE your\_database;

ALTER DATABASE AUDIT SPECIFICATION YourAuditSpecification WITH (STATE = OFF);

Очистка и уменьшение размеров журнала базы данных включают в себя ряд шагов, которые могут зависеть от конкретной системы управления базами данных (СУБД). Ниже приведены общие рекомендации для нескольких популярных СУБД: PostgreSQL и Microsoft SQL Server.

PostgreSQL:

Очистка журнала:

Очистка журнала транзакций:

sql

Copy code

SELECT pg\_switch\_xlog(); -- Производит переключение журнала и запускает архивацию устаревших журналов.

Автоматическое удаление старых журналов (опционально):

Настроить параметры archive\_command в postgresql.conf для отправки архивов на удаленное хранилище.

Уменьшение размеров журнала:

Уменьшение размеров таблицы журнала транзакций (опционально):

sql

Copy code

VACUUM FULL pg\_xlog;

Настройка параметров хранения журнала:

Пересмотрите параметры max\_wal\_size и min\_wal\_size в postgresql.conf для более эффективного управления размерами журнала.

Microsoft SQL Server:

Очистка журнала:

Резервное копирование и восстановление базы данных:

Выполните резервное копирование базы данных и восстановление из этого резервного копирования. Это автоматически очистит транзакционный журнал.

Требование выполнения резервного копирования (опционально):

Убедитесь, что в базе данных включена опция SIMPLE или BULK-LOGGED вместо FULL, если нет необходимости в точных восстановлениях.

Уменьшение размеров журнала:

Изменение размера файла журнала:

Вы можете изменить размер файла журнала с помощью команды:

sql

Copy code

ALTER DATABASE YourDatabaseName MODIFY FILE (NAME = 'YourLogFileName', SIZE = NewSize);

Освобождение неиспользуемого пространства:

Используйте команду DBCC SHRINKFILE для уменьшения размера файла журнала (лог-файла):

sql

Copy code

DBCC SHRINKFILE('YourLogFileName', NewSize);

Настройка автоматического управления размерами файла журнала (опционально):

Измените настройки роста файла журнала для более эффективного управления его размерами.

Обратите внимание, что изменение размеров и очистка журнала должны выполняться осторожно, чтобы избежать потери данных или нарушения целостности базы данных. Важно проводить эти операции в соответствии с лучшими практиками и учитывать потребности конкретной среды базы данных.

1. Технологии создания базы данных с применением языка SQL. Добавление, удаление данных и таблиц.

Язык SQL (Structured Query Language) широко используется для создания, изменения и управления базами данных. Вот основные операции создания базы данных с использованием SQL:

Создание таблицы:

sql

Copy code

CREATE TABLE YourTableName (

column1 datatype1,

column2 datatype2,

column3 datatype3,

-- ...

PRIMARY KEY (column1)

);

Пример:

sql

Copy code

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50),

BirthDate DATE,

Salary DECIMAL(10, 2)

);

Добавление данных:

sql

Copy code

INSERT INTO YourTableName (column1, column2, column3, ...)

VALUES (value1, value2, value3, ...);

Пример:

sql

Copy code

INSERT INTO Employees (EmployeeID, FirstName, LastName, BirthDate, Salary)

VALUES (1, 'John', 'Doe', '1990-01-01', 50000.00);

Обновление данных:

sql

Copy code

UPDATE YourTableName

SET column1 = value1, column2 = value2, ...

WHERE condition;

Пример:

sql

Copy code

UPDATE Employees

SET Salary = 55000.00

WHERE EmployeeID = 1;

Удаление данных:

sql

Copy code

DELETE FROM YourTableName WHERE condition;

Пример:

sql

Copy code

DELETE FROM Employees WHERE EmployeeID = 1;

Удаление таблицы:

sql

Copy code

DROP TABLE YourTableName;

Пример:

sql

Copy code

DROP TABLE Employees;

Эти команды предоставляют основные инструменты для создания, изменения и удаления данных и таблиц в базе данных с использованием SQL. Обратите внимание, что выполнение операций удаления данных (DELETE) и удаления таблицы (DROP TABLE) следует выполнять осторожно, так как они могут привести к потере данных. Важно иметь резервные копии данных перед выполнением подобных операций.

1. Динамический SQL и его операторы.

Динамический SQL — это концепция, при которой SQL-запросы формируются и выполняются во время выполнения программы, а не на этапе компиляции. Это позволяет строить и изменять запросы на лету в зависимости от условий или ввода пользователя. Для работы с динамическим SQL используются специальные операторы и конструкции языка программирования. Ниже представлены примеры динамических SQL-операторов в различных языках программирования.

Примеры встроенных операторов:

Вот пример использования динамического SQL на PHP с использованием расширения PDO для работы с базой данных MySQL:

php

<?php

$servername = "localhost";

$username = "root";

$password = "";

$dbname = "example";

try {

$conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);

$conn->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

$tableName = 'Employees';

$columnName = 'Salary';

$value = 60000;

// Динамический SQL-запрос

$sqlQuery = "UPDATE $tableName SET $columnName = :value WHERE EmployeeID = 1";

// Подготовка и выполнение запроса с использованием параметра

$stmt = $conn->prepare($sqlQuery);

$stmt->bindParam(':value', $value, PDO::PARAM\_INT);

$stmt->execute();

echo "Запись успешно обновлена";

} catch (PDOException $e) {

echo "Ошибка: " . $e->getMessage();

}

$conn = null;

?>

В данном примере мы используем PDO (PHP Data Objects) для соединения с базой данных MySQL. Динамический SQL-запрос формируется с использованием переменных, а параметризованный запрос готовится с использованием prepare() и выполняется с использованием execute(). Это помогает предотвратить SQL-инъекции, так как значения передаются как параметры, а не конкатенируются в строке запроса.

1. Цель и задачи инструментов мониторинга нагрузки сервера.

Инструменты мониторинга нагрузки сервера используются для наблюдения и анализа активности сервера в реальном времени или в определенные периоды времени. Они играют ключевую роль в обеспечении стабильной работы серверов, предотвращении проблем и оптимизации производительности. Вот цели и задачи, которые решают инструменты мониторинга нагрузки сервера:

Цели мониторинга нагрузки сервера:

Обеспечение стабильности:

Главная цель - поддерживать стабильную работу серверов, предотвращая сбои и снижая риск возникновения неполадок.

Оптимизация производительности:

Мониторинг помогает выявить слабые места в работе сервера и оптимизировать его производительность.

Предупреждение о проблемах:

Инструменты мониторинга позволяют выявлять потенциальные проблемы до того, как они повлияют на работу сервера.

Планирование ресурсов:

Мониторинг предоставляет данные о загрузке ресурсов, что помогает планировать и масштабировать серверную инфраструктуру.

Соблюдение SLA (уровня обслуживания):

Мониторинг помогает обеспечивать соответствие уровню обслуживания, определенному в SLA, и быстро реагировать на любые отклонения.

Задачи инструментов мониторинга нагрузки сервера:

Измерение использования ресурсов:

Слежение за использованием CPU, памяти, дискового пространства, сетевого трафика и других ресурсов.

Анализ производительности:

Оценка скорости выполнения запросов, времени отклика сервера и других параметров производительности.

Обнаружение узких мест:

Выявление узких мест в инфраструктуре, таких как неэффективные запросы, сетевые узкие места и т.д.

Мониторинг событий и журналирование:

Регистрация событий, ошибок и предупреждений для анализа и реагирования на них.

Мониторинг приложений:

Оценка производительности приложений, выявление ошибок и проблем в коде.

Масштабирование и балансировка нагрузки:

Планирование масштабирования ресурсов и балансировка нагрузки для эффективного распределения запросов.

Определение аномалий:

Обнаружение аномального поведения сервера, что может указывать на проблемы или атаки.

Генерация отчетов и уведомлений:

Создание отчетов о состоянии сервера и отправка уведомлений при превышении заданных пороговых значений.

Слежение за изменениями в конфигурации:

Мониторинг изменений в конфигурации сервера для предотвращения несанкционированных изменений.

Соответствие стандартам безопасности:

Проверка соответствия сервера стандартам безопасности и реагирование на уязвимости.

Прогнозирование нагрузки:

Анализ исторических данных для прогнозирования будущей нагрузки и предотвращения проблем.

Интеграция инструментов мониторинга в инфраструктуру предприятия помогает поддерживать надежную и эффективную работу серверов, что является ключевым элементом в современных ИТ-системах.

***Практические задания:***

1. Разработать схему Базы данных «Колледж».
2. Разработать схему Базы данных «Абитуриенты».
3. Разработать схему Базы данных «Цветочный магазин».
4. Разработать схему Базы данных «Интернет-магазин».
5. Разработать словарь данных Базы данных «Колледж».
6. Разработать словарь данных Базы данных «Абитуриенты».
7. Разработать словарь данных Базы данных «Студенты».
8. Разработать словарь данных Базы данных «Цветочный магазин».
9. Разработать словарь данных Базы данных «Интернет-магазин».
10. Разработка технических требований к серверу баз данных на 800 зарегистрированных пользователей.
11. Разработка технических требований к серверу баз данных на 1500 зарегистрированных пользователей.
12. Разработка технических требований к серверу баз данных на 430 зарегистрированных пользователей.
13. Разработка технических требований к серверу баз данных на 2700 зарегистрированных пользователей.
14. Разработка технических требований к серверу баз данных на 150 зарегистрированных пользователей.
15. Создать хранимые процедуры.
16. Написать SQL-запрос для создания 4 таблиц, содержащих не менее 5 атрибутов.
17. Разработать схему Базы данных «Студенты».
18. Описать SSL сертификат WEB-ресурса https://ya.ru/
19. Описать SSL сертификат WEB-ресурса <https://mail.ru/>
20. Описать SSL сертификат WEB-ресурса https://ru.wikipedia.org/
21. Описать SSL сертификат WEB-ресурса https://school.mos.ru/
22. Описать SSL сертификат WEB-ресурса https://www.gosuslugi.ru/
23. Описать SSL сертификат WEB-ресурса https://www.mos.ru/uslugi/
24. Описать SSL сертификат WEB-ресурса https://vk.com/
25. Описать SSL сертификат WEB-ресурса https://dnevnik.mos.ru/