МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«КУЗБАССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ Т. Ф. ГОРБАЧЕВА»
Филиал КузГТУ в г. Белово

Кафедра инженерно-экономическая

УП.07.01 «Соадминистрирование баз данных и серверов»»

Методические рекомендации
по выполнению учебной практики
для специальности
09.02.07 «Информационные системы и программирование»

Составитель: Витвицкий М.Н.
Рассмотрены и утверждены на заседании кафедры
Протокол № 5 от 17.01.2026 г.
Рекомендовано учебно-методической комиссией специальностей СПО в качестве электронного издания для использования в учебном процессе
Протокол № 5 от 20.01.2026 г.

Белово 2026

СОДЕРЖАНИЕ

ОРГАНИЗАЦИЯ УЧЕБНОЙ ПРАКТИКИ

Первоначальные профессиональные навыки обучающиеся по основным профессиональным образовательным программам получают во время прохождения учебных и производственных практик. Практика имеет целью комплексное освоение обучающимися всех видов профессиональной деятельности по специальности (профессии) среднего профессионального образования, формирование общих и профессиональных компетенций, а также приобретение необходимых умений и опыта практической работы по специальности (профессии).

Практика по профилю специальности направлена на формирование у обучающегося общих и профессиональных компетенций, приобретение практического опыта и реализуется в рамках профессиональных модулей ОП СПО по каждому из видов профессиональной деятельности, предусмотренных ФГОС СПО по специальности.

Учебная практика проводится в филиале КузГТУ в г.Белово в специально оборудованных аудиториях. Организацию и руководство практикой по профилю специальности (профессии) осуществляют руководители практики из числа профессорско-преподавательского состава филиала КузГТУ в г.Белово.

Руководитель практики из числа лиц, относящихся к профессорско-преподавательскому составу филиала:

- разрабатывает содержание и планируемые результаты практики;
- осуществляет руководство практикой;
- контролирует реализацию программы практики и условия проведения практики, в том числе требования охраны труда, безопасности жизнедеятельности и пожарной безопасности в соответствии с правилами и нормами, в том числе отраслевыми;
- формирует группы в случае применения групповых форм проведения практики;
- определяет процедуру оценки общих и профессиональных компетенций обучающегося, освоенных им в ходе прохождения практики;
- разрабатывает формы отчетности и оценочный материал прохождения практики.

Обучающиеся в период прохождения практики обязаны:

- выполнять задания, предусмотренные программами практики;

- соблюдать действующие в филиале КузГТУ в г.Белово правила внутреннего трудового распорядка;

- соблюдать требования охраны труда и пожарной безопасности.

Результаты практики определяются программами практики, разрабатываемыми руководителями практики из числа профессорско-преподавательского состава филиала КузГТУ в г.Белово. По результатам практики руководителем практики формируется аттестационный лист, содержащий сведения об уровне освоения обучающимся профессиональных компетенций, а также характеристика на обучающегося по освоению профессиональных компетенций в период прохождения практики.

В период прохождения практики обучающимся ведется дневник практики. По результатам практики обучающимся составляется отчет. В качестве приложения к дневнику практики обучающийся оформляет графические, аудио-, фото-, видеоматериалы, подтверждающие практический опыт, полученный на практике.

Аттестация по итогам учебной практики проводится с учетом (или на основании) результатов ее прохождения, подтверждаемых аттестационным листом.

Отчет по практике является основным документом, характеризующим работу обучающегося во время практики. Отчет составляется в соответствии с программой практики и содержит следующие разделы:

1. Титульный лист
2. Задание на учебную практику
3. Введение
4. Теоретические основы в соответствии с темами практики
5. Реализация поставленной задачи
6. Выводы
7. Список используемой литературы

Комплект документов, оформляемых при прохождении учебной практики, а также титульный лист отчета по учебной практике приведены в Приложении А-Д.

КОНТРОЛЬ РЕЗУЛЬТАТОВ ВЫПОЛНЕНИЯ ЗАДНИЙ ПО УЧЕБНОЙ ПРАКТИКЕ

Контроль результатов по учебной практике студентов осуществляться в пределах времени, отведенного на обязательные учебные занятия, может проходить в письменной, устной или смешанной форме, с представлением продукта деятельности учащегося.

В качестве форм и методов контроля работы обучающихся, могут быть использованы, зачеты, тестирование, самоотчеты по этапам практики, практические задания, и др., которые могут осуществляться на учебном занятии или вне его.

Общими критериями оценки результатов работы обучающегося являются:

- уровень освоения учащимся учебного материала;
- умение обучающегося использовать теоретические знания при выполнении практических задач;
- сформированность общих и профессиональных компетенций;
- обоснованность и четкость изложения ответа;
- оформление материала в соответствии с требованиями.

МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ

УЧЕБНАЯ ПРАКТИКА ПМ.07 «Соадминистрирование баз данных и серверов».

Учебная практика — это форма организации учебного процесса, нацеленная на формирование и закрепление практических умений и навыков через непосредственное выполнение профессиональных или учебно-профессиональных действий. Её суть — в переносе теоретических знаний в плоскость реального (или моделируемого) практического действия.

Цель учебной практики 07.01 «Соадминистрирование баз данных и серверов».

Комплексное освоение обучающимися вида профессиональной деятельности «Соадминистрирование баз данных и серверов», формирование общих и профессиональных компетенций, приобретение необходимого опыта практической работы по специальности.

Задачи учебной практики 07.01 «Соадминистрирование баз данных и серверов».

Выявление технических проблем, возникающих в процессе эксплуатации баз данных и серверов

Осуществление администрирования отдельных компонентов серверов.

Формирование требований к конфигурации локальных компьютерных сетей и серверного оборудования, необходимого для работы баз данных и серверов; осуществление администрирования баз данных в рамках своей компетенции; проведения аудита баз данных и серверов с использованием регламентов по защите информации.

Выполнение работ по вводу и представлению данных, формированию входящей и исходящей документации.

Учебная практика осуществляется в двух формах:

1. Проработка теоретической части рассматриваемой задачи (теоретическая часть работы);
2. Проработка практической части рассматриваемой задачи (выдается преподавателем индивидуально согласно перечню тем заданий).

Задания по учебной практике

№ раздела (темы)	Вопросы, выносимые на изучение	Количество часов
Тема 1. Принципы построения и администрирования баз данных	Задание 1.1. Рассмотрение обязанностей администратора баз данных. Основные утилиты администратора баз данных. Режимы запуска и останова базы данных.	4
	Задание 1.2. Рассмотрение схемы базы данных, привилегий. Управление пользователями баз данных.	4
	Задание 1.3. Рассмотрение вопросов, связанных с табличными пространствами и файлами данных. Модели и типы данных.	4
	Задание 1.4. Рассмотрение вопросов, связанных со схемой и объектами схемы данных. Блоки данных, экстенты сегменты.	4
	Задание 1.5. Рассмотрение вопросов, связанных с структурой памяти. Однопроцессорные и многопроцессорные базы данных.	4
	Задание 1.6. Рассмотрение вопросов, связанных с транзакциями, блокировкой и согласованностью данных.	4
	Задание 1.7. Журнал базы данных: структура и назначение файлов журнала, управление переключениями и контрольными точками.	4
	Задание 1.8. Словарь данных: назначение, структура, префиксы.	4
	Задание 1.9. Рассмотрение распределённых баз данных.	4
	Задание 1.10. Выбор среды для реализации создаваемой учебной базы данных.	4
	Задание 1.11. Определение обязанностей администратора применительно к создаваемой	4

	учебной базы данных.	
	Задание 1.12. Построение схемы создаваемой учебной базы данных.	4
	Задание 1.13. Построение словаря создаваемой учебной базы данных.	4
	Задание 1.14. Изучение команд администрирования данных, применительно к создаваемой учебной базе данных.	4
	Задание 1.15. Разработка сценариев работы с данными в создаваемой учебной базе данных.	4
Тема 2. Серверы баз данных	Задание 2.1. Рассмотрение вопросов, связанных с классификацией серверов, принципами разделения между клиентскими и серверными частями. Типовое разделение функций.	4
	Задание 2.2. Рассмотрение вопросов, связанных с протоколами удаленного вызова процедур. Требования к аппаратным возможностям и базовому программному обеспечению клиентов и серверов.	4
	Задание 2.3. Рассмотрение вопросов, связанных с хранимыми процедурами и триггерами.	4
	Задание 2.4. Рассмотрение вопросов, связанных с характеристиками серверов баз данных и механизмами доступа к базам данных.	4
	Задание 2.5. Рассмотрение вопросов, связанных с аппаратным обеспечением и развертыванием серверов баз данных.	4
	Задание 2.6. Рассмотрение состава и схемы банка данных	4
	Задание 2.7. Разработка технических требований к серверу баз данных, применительно к разрабатываемому заданию.	4

	Задание 2.8. Разработка требований к корпоративной сети, применительно к разрабатываемому заданию.	4
	Задание 2.9. Решение вопросов, связанных с конфигурированием сети, применительно к разрабатываемому заданию.	6
	Задание 2.10. Разработка хранимых процедур, применительно к создаваемой учебной базе данных.	6
	Задание 2.11. Разработка триггеров, применительно к создаваемой учебной базе данных.	6
Тема 3. Администрирование баз данных и серверов	Задание 3.1. Рассмотрение вопросов, связанных с технологией установки и настройка сервера MySQL в операционной системе Windows. Клиентские настойки, протоколирование, безопасность.	6
	Задание 3.2. Рассмотрение вопросов, связанных с технологией установки и настройка сервера MySQL в операционных системах Linux.	4
	Задание 3.3. Рассмотрение вопросов, связанных с удаленным администрированием.	4
	Задание 3.4. Рассмотрение вопросов, связанных с аудитом базы данных, аудиторским журналом. Установка опций, включение и отключение аудита. Очистка и уменьшение размеров журнала.	4
	Задание 3.5. Рассмотрение вопросов, связанных с технологиями создания базы данных с применением языка SQL. Добавление, удаление данных и таблиц.	6
	Задание 3.6. Рассмотрение вопросов, связанных с создание запросов, процедур и триггеров. Создание запросов и процедур на изменение структуры базы данных.	6
	Задание 3.7. Рассмотрение вопросов, связанных с динамическим SQL и его операторами.	4
Промежуточная аттестация в форме: дифференцированного зачета		

Типовые вопросы на защиту отчета

1. Особенности реализации поставленной задачи.

2. Что является целью выполнения задания по практике?

3. Поясните схему реализованной базы данных.

4. Особенности, выявленные при анализе предметной области.

5. Сформулируйте требования к базе данных.

6. Концептуальная модель базы данных.

7. Инфологическая модель базы данных.

8. Даталогическое проектирование базы данных.

9. Определение информационных единиц и связей между ними.

10. Преобразование исходной инфологической модели в модель данных, поддерживаемую конкретной СУБД.

11. Проверка адекватности полученной даталогической модели предметной области.

12.Сценарии работы с данными.

13. Архитектуры системы.

14. Технические требования к серверу баз данных.

15. Требования к корпоративной сети.

16. Серверные компоненты базы данных.

17. Клиентские компоненты базы данных.

18. Установка и настройка сервера баз данных.

19. Создание механизмов сервера для обслуживания базы данных.

20. Установка и развѐртывание системы.

21. Работа с журналом аудита базы данных.

22. Мониторинг нагрузки сервера.

23. Настройка политики безопасности для базы данных.

24. Создание резервных копий базы данных.

25. Восстановление базы данных.

26. Мониторинг активности портов.

27. Оформление требований и разработка технического задания по сертификации информационной системы (базы данных).

28. Выбор сертификатов.

29. Сроки действия сертификатов.

Тема 1. Принципы построения и администрирования баз данных.

В современных условиях развития информационных технологий базы данных стали ключевым элементом практически любой информационной системы. Эффективное управление данными и их безопасное хранение определяют успешность работы организаций различных масштабов и направлений деятельности.

База данных представляет собой структурированное хранилище информации, организованное таким образом, чтобы обеспечить:

Надёжное хранение данных.

Быстрый доступ к информации.

Возможность многопользовательской работы.

Целостность и безопасность данных.

Администрирование баз данных включает комплекс мероприятий по:

Созданию и настройке БД.

Обеспечению безопасности.

Оптимизации производительности.

Резервному копированию.

Мониторингу состояния системы.

Основные цели администрирования БД:

Обеспечение бесперебойной работы системы.

Поддержание целостности данных.

Защита от несанкционированного доступа.

Оптимизация производительности.

Масштабирование системы.

Ключевые задачи:

Проектирование структуры БД.

Настройка параметров производительности.

Управление пользователями и правами доступа.

Мониторинг работоспособности.

Восстановление после сбоев.

Фундаментальные принципы создания БД:

Целостность данных — обеспечение корректности и непротиворечивости информации.

Безопасность — защита от несанкционированного доступа.

Производительность — оптимизация скорости обработки запросов.

Масштабируемость — возможность расширения системы.

Отказоустойчивость — способность к восстановлению после сбоев.

Задание 1.1. Рассмотрение обязанностей администратора баз данных. Основные утилиты администратора баз данных. Режимы запуска и останова базы данных.

Теоретическая часть: Администратор баз данных (DBA) — специалист, который отвечает за планирование, проектирование, установку, настройку, поддержку, обслуживание и безопасность систем управления базами данных (СУБД). От его работы зависит надёжность, доступность, производительность и безопасность данных, лежащих в основе бизнеса.

Основные обязанности администратора баз данных

Планирование и проектирование:

определение потребностей в хранении данных;

выбор подходящей СУБД и создание архитектуры базы данных;

разработка стандартов и процедур для управления данными, включая политики резервного копирования и восстановления;

создание моделей данных и оптимизация их структуры для повышения производительности.

Установка, настройка и конфигурация СУБД:

установка и настройка серверов, параметров конфигурации;

настройка репликации, кластеров и других механизмов для обеспечения высокой доступности;

обновление СУБД и её компонентов.

Управление пользователями и доступом:

создание учётных записей пользователей и управление их правами доступа;

настройка ролей и привилегий;

обеспечение безопасности и конфиденциальности данных.

Мониторинг и оптимизация:

отслеживание производительности СУБД;

оптимизация запросов и индексов;

мониторинг использования ресурсов (CPU, память, дисковое пространство);

анализ нагрузки и прогнозирование будущих потребностей.

Резервное копирование и восстановление:

регулярное создание резервных копий данных;

разработка и внедрение стратегий резервного копирования;

тестирование процедур восстановления;

восстановление данных в случае аварии или сбоя.

Обеспечение безопасности:

защита от несанкционированного доступа и угроз;

реализация политики шифрования данных и паролей;

аудит действий пользователей и мониторинг подозрительной активности;

соответствие требованиям законодательства и нормативных актов по защите данных (например, GDPR, PCI DSS).

Обслуживание и поддержка:

исправление ошибок и уязвимостей в СУБД;

управление хранилищами данных;

предоставление технической поддержки пользователям;

обучение пользователей работе с СУБД.

Документирование:

ведение документации по СУБД, процедурам управления данными, политике резервного копирования и восстановления;

описание архитектуры базы данных, её компонентов и конфигурации.

Анализ и отчётность:

анализ данных, выявление тенденций и проблем;

создание инструментов для анализа и визуализации данных;

подготовка отчётов о состоянии базы данных и её производительности.

Основные утилиты администратора баз данных

MySQL Workbench — инструмент визуального моделирования и проектирования баз данных для MySQL. Позволяет проектировать, создавать, разрабатывать, сопровождать и администрировать SQL.

phpMyAdmin — менеджер баз данных, часто используемый для веб-приложений. Позволяет копировать данные и таблицы, удалять содержимое, создавать запросы и выполнять другие операции.

HeidiSQL — инструмент управления базами данных с открытым исходным кодом. Поддерживает MySQL, SQL Server, PostgreSQL. Позволяет просматривать и редактировать данные, создавать и редактировать таблицы, представления и базы данных.

Prometheus + Grafana — система для мониторинга и визуализации метрик производительности. Prometheus собирает данные о работе СУБД, а Grafana превращает их в информативные дашборды.

Percona Monitoring and Management (PMM) — открытая платформа для мониторинга и управления MySQL, PostgreSQL и MongoDB. Идентифицирует проблемные запросы и даёт рекомендации по оптимизации.

Liquibase — решение для контроля версий схемы базы данных. Позволяет документировать изменения, автоматизировать миграции и синхронизировать схемы между различными средами.

Redgate SQL Toolbelt — комплексный набор инструментов для MS SQL Server, включающий средства для сравнения и синхронизации схем и данных, поиска узких мест производительности и т. д..

isql (Интерактивный SQL) — утилита для выполнения операторов SQL в режиме диалога, анализа схемы БД.

gbak — утилита для резервного копирования и восстановления баз данных.

nbackup — утилита для создания быстрых и инкрементных резервных копий всей базы данных.

gfix — утилита для управления режимом чистки базы данных, перевода её в эксклюзивное/оперативное состояние, изменения параметров БД.

mysqladmin — стандартный клиент для получения общей информации о работе MySQL.

mysqldumpslow — утилита для анализа данных slow.log и вывода самых частых медленных запросов.

Percona Toolkit for MySQL — пакет скриптов для рутинных операций администрирования: проверки состояния репликации, сбора информации,

оптимизации запросов и т. д..

MySQLTuner — Perl-скрипт, который читает текущие настройки сервера и установки MySQL и выдаёт рекомендации по их изменению.

Режимы запуска базы данных

Холодный запуск (Cold Startup). Полный запуск базы данных с нуля, включая загрузку всех системных файлов, инициализацию всех служб и компонентов. Используется при первом запуске после установки, восстановлении из резервной копии или перезагрузке системы. Требует больше времени и ресурсов, но обеспечивает полную инициализацию всех компонентов.

Тёплый запуск (Warm Startup). Быстрый запуск, при котором база данных загружает только необходимую информацию из системных файлов, используя данные о предыдущем состоянии. Применяется при перезапуске после кратковременного сбоя или для быстрого возобновления работы. Использует меньше ресурсов, чем холодный запуск, но не всегда обеспечивает полную инициализацию всех компонентов.

Горячий запуск (Hot Startup). Запуск без отключения базы данных, позволяющий выполнять определённые действия (например, обновление конфигурации или изменение параметров) без прерывания операций. Используется для обслуживания без остановки работы. Самый быстрый режим, не прерывает работу пользователей, но требует осторожности, так как изменения могут повлиять на работу системы.

Режимы остановки базы данных

Нормальная остановка (NORMAL). Используется по умолчанию. Новые соединения не разрешаются, сервер ожидает отсоединения всех пользователей, затем записывает изменённые буферы на диск, завершает фоновые процессы и удаляет SGA из памяти. Перед остановкой база данных закрывается и демонтируется. При следующем запуске восстановление не требуется.

Транзакционная остановка (TRANSACTIONAL). Обеспечивает сохранность данных клиентов. Ни один клиент не может начать новую транзакцию, пользователи принудительно отсоединяются после завершения текущих транзакций. После завершения всех транзакций выполняется остановка. При следующем запуске восстановление не требуется.

Немедленная остановка (IMMEDIATE). Обработка текущих команд SQL не завершается, сервер не ожидает отсоединения пользователей. Происходит откат всех активных транзакций, принудительное отсоединение пользователей, закрытие и демонтаж базы данных. При следующем запуске восстановление не требуется.

Аварийная остановка (ABORT). Применяется, если другие режимы остановки не сработали. Немедленно отменяет все обрабатываемые команды SQL, не записывает буферы на диск, не выполняет откат незафиксированных транзакций. База данных не закрывается и не демонтируется, экземпляр удаляется без закрытия файлов. При следующем запуске потребуется автоматическое восстановление.

Рекомендации:

планировать остановку базы данных для технических работ или обновления;

выбирать режим запуска/остановки в зависимости от ситуации;

регулярно создавать резервные копии перед запуском или остановкой;

отслеживать состояние базы данных во время этих процессов;

документировать все действия по запуску и остановке, включая использованные параметры и команды.

Режимы запуска и остановки могут различаться в зависимости от конкретной СУБД. Важно внимательно изучать документацию к используемой системе

Практическая часть: Разработать типовую инструкцию администратора баз данных по действующему ГОСТу и стандарту.

Задание 1.2. Рассмотрение схемы базы данных, привилегий. Управление пользователями баз данных.

Теоретическая часть: Схема базы данных — это структура и организация базы данных, которая определяет её таблицы, связи, ограничения, поля и типы данных. Она служит образцом для хранения, доступа и управления данными, обеспечивая согласованность и целостность системы.

Основные типы схем базы данных:

Концептуальная схема — высший уровень абстракции, представляющий логическое представление всей базы данных. Включает унифицированный набор сущностей, атрибутов и связей, отражающих информационные требования пользователей или приложений. Не зависит от технологий реализации.

Логическая схема — описывает организацию данных в базе данных с точки

зрения таблиц, полей, связей и ограничений. Не зависит от физической реализации и фокусируется на логической организации данных.

Физическая схема — определяет, как данные хранятся на диске, включая распределение дискового пространства, методы сжатия и шифрования.

Компоненты схемы базы данных:

Таблицы — основные структуры для хранения данных, состоящие из строк и столбцов (полей). Каждая таблица предназначена для хранения определённого типа сущности (например, клиентов, продуктов).

Поля (столбцы) — отдельные элементы данных в таблице, каждый из которых имеет имя, тип данных (текст, целое число, дата и т. д.) и дополнительные ограничения (например, «не пустое» или «уникальное»).

Отношения — определяют взаимосвязи между таблицами. Основные типы отношений: «один к одному», «один ко многим» и «многие ко многим».

Ограничения — правила, обеспечивающие целостность данных. Например, первичные и внешние ключи, уникальные ограничения.

В Oracle Database схема привязывается к одному пользователю и является логическим набором объектов базы данных (таблиц, последовательностей, хранимых процедур и т. д.), созданных этим пользователем. В PostgreSQL концепция пользователей и групп объединена в понятие «роли», которые могут владеть объектами и управлять доступом к ним.

Привилегии в базах данных

Привилегия — это разрешение на выполнение определённого типа операции SQL или доступ к объекту базы данных (например, таблице). Привилегии управляют использованием вычислительных ресурсов и доступом к данным.

Типы привилегий:

Системные привилегии — позволяют выполнять действия с определённым типом объектов на уровне всей системы. Например, привилегия удалять строки любой таблицы в базе данных. Дают возможность создавать, изменять или удалять объекты базы данных.

Объектные привилегии — разрешают выполнять определённые действия с конкретным объектом (таблицей, функцией, пакетом). Например, право удалять строки из таблицы.

Некоторые стандартные привилегии в SQL:

SELECT — позволяет выполнять запросы к данным;

INSERT — разрешает добавлять строки в таблицу;

UPDATE — позволяет изменять значения в таблице (может быть ограничена отдельными столбцами);

DELETE — разрешает удалять строки из таблицы;

ALTER — позволяет изменять структуру таблицы или последовательности;

REFERENCES — разрешает устанавливать внешний ключ с использованием столбцов таблицы;

INDEX — позволяет создавать индексы для таблицы;

EXECUTE — разрешает выполнять хранимые процедуры или функции.

Управление привилегиями осуществляется с помощью SQL-операторов:

GRANT — предоставляет привилегии. Синтаксис: GRANT привилегии ON объект TO пользователь.

REVOKE — отменяет привилегии. Синтаксис: REVOKE привилегии ON объект FROM пользователь.

Фраза WITH GRANT OPTION в операторе GRANT позволяет получателю привилегии передавать её другим пользователям.

Управление пользователями баз данных

Управление пользователями — ключевой аспект обеспечения безопасности данных в СУБД. Оно включает создание учётных записей, назначение прав доступа, контроль и аудит действий пользователей.

Основные понятия:

Учётная запись пользователя (login) — позволяет подключиться к серверу, но не даёт автоматического доступа к базам данных.

Пользователь базы данных (user) — создаётся на основе учётной записи для конкретной базы данных и получает права доступа в рамках этой базы.

Роль — механизм для группировки пользователей с одинаковыми правами доступа. Упрощает управление привилегиями, так как позволяет назначать права группе, а не отдельным пользователям.

В MS SQL Server существует двухуровневая система доступа:

На первом уровне создаётся учётная запись пользователя (login) для

подключения к серверу.

На втором уровне для каждой базы данных на основании login создаётся запись пользователя (user) с определёнными правами доступа.

В PostgreSQL роли могут быть:

Суперпользователями (SUPERUSER) — имеют неограниченные права, проверки разграничения доступа для них не выполняются.

Владельцами объектов — получают полный набор привилегий для созданного объекта. Могут передавать права другим ролям.

Обычными ролями — имеют доступ к объектам только в рамках выданных им привилегий.

Псевдороль public включает все остальные роли. Привилегии, выданные для public, получают все пользователи.

Для автоматизации управления пользователями используются системы класса IdM/IGA (Identity and Access Management / Identity Governance and Administration). Они позволяют централизованно управлять учётными записями, назначать и отзывать права, проводить аудит доступа.

Аудит действий пользователей — важный аспект управления. Он позволяет отслеживать, кто и когда производил изменения в базе данных, что помогает выявлять потенциальные угрозы.

Эффективное управление пользователями и привилегиями снижает риски несанкционированного доступа, обеспечивает целостность данных и упрощает администрирование системы.

Практическая часть: Разработать логическую ER-модель базы данных по варианту. Продумать роли в ИС и БД.

Задание 1.3. Рассмотрение вопросов, связанных с табличными пространствами и файлами данных. Модели и типы данных.

Теоретическая часть: Табличные пространства и файлы данных — ключевые концепции в управлении базами данных, связанные с физическим и логическим размещением данных.

Табличные пространства

Табличное пространство — логическая единица в базе данных, которая определяет физическое расположение данных. Оно представляет собой контейнер

для хранения объектов базы данных (таблиц, индексов, последовательностей и т. д.) и состоит из одного или нескольких физических файлов данных. Табличные пространства позволяют администратору базы данных управлять распределением памяти, устанавливать квоты для пользователей, контролировать доступность данных и распределять их по устройствам для повышения производительности.

Некоторые особенности табличных пространств:

У каждой базы данных есть табличное пространство по умолчанию, в котором создаются все объекты, если явно не указано иное.

Одно и то же табличное пространство может использоваться разными базами данных, а одна база данных может хранить данные в нескольких табличных пространствах.

Логическая и физическая структуры не зависят друг от друга: объекты одной схемы могут находиться в разных табличных пространствах, а одно табличное пространство может содержать объекты из разных схем.

Табличные пространства можно переводить в состояния ONLINE (доступно) и OFFLINE (недоступно).

В Oracle Database каждое табличное пространство содержит словарь данных, который включает определения таблиц, представлений и хранимых процедур, необходимых базе данных. В PostgreSQL при инициализации кластера создаются два табличных пространства по умолчанию:

pg_default — используется как табличное пространство по умолчанию, если явно не выбрано другое пространство. Располагается в каталоге PGDATA/base.

pg_global — хранит общие для всего кластера объекты системного каталога. Располагается в каталоге PGDATA/global.

Для создания табличного пространства в PostgreSQL используется команда CREATE TABLESPACE.

Файлы данных

Файл данных — физический файл в постоянном хранилище, созданный СУБД и содержащий структуры данных, такие как таблицы и индексы. Файлы данных ассоциированы с табличным пространством и хранят все данные этого табличного пространства.

Особенности файлов данных:

Любой файл данных может ассоциироваться только с одним табличным пространством и только с одной базой данных.

При создании файла данных СУБД распределяет ему указанное количество дисковой памяти.

Содержимое файла данных после его создания ещё не содержит данных, но зарезервировано за будущими сегментами табличного пространства.

Индивидуальные файлы данных можно переводить в состояние OFFLINE, но это обычно делается лишь при некоторых процедурах восстановления базы данных.

В SQL Server выделяют два типа файлов данных:

Первичные. Содержат информацию, необходимую для запуска базы данных, и ссылки на другие файлы в базе данных. В каждой базе данных есть ровно один первичный файл данных. Рекомендуется использовать расширение .mdf.

Вторичные. Необязательные файлы, которые могут хранить данные и объекты, отсутствующие в первичном файле. База данных может не иметь вторичных файлов, если все её данные хранятся в первичном файле.

Модели данных

Модель данных — структурное представление элементов данных, их отношений и ограничений в системе управления базами данных (СУБД). Она определяет способ организации и хранения данных.

Основные типы моделей данных:

Иерархическая. Данные организованы в виде древовидной структуры, где каждый узел имеет одного родителя и может иметь несколько дочерних узлов. Подходит для отношений «один ко многим». Пример: организационные структуры, файловые системы.

Сетевая. Позволяет каждому элементу иметь несколько родительских связей. Подходит для представления сложных сетей и отношений.

Реляционная. Данные организованы в виде таблиц, взаимосвязь между которыми осуществляется через ключи. Наиболее распространённая модель, используется в СУБД вроде PostgreSQL, MySQL, Oracle.

Объектно-ориентированная. Данные представляются в виде объектов, которые содержат атрибуты и методы. Поддерживает наследование, инкапсуляцию и полиморфизм. Подходит для работы со сложными, неструктурированными

данными.

Документная. Позволяет хранить данные в формах документов, часто в формате JSON или XML. Удобна для веб-приложений и служб, где структура данных может часто меняться.

Типы данных

Типы данных определяют вид информации, которая может храниться в столбцах таблиц, и операции, которые можно с ней выполнять.

В реляционных базах данных используются базовые типы данных, такие как:

CHAR — строковый тип фиксированной длины;

DATE — дата;

NUMBER — числовой тип.

В объектно-ориентированных базах данных поддерживаются сложные типы данных, определённые пользователем (классы).

Некоторые СУБД (например, PostgreSQL) поддерживают пользовательские типы данных и расширения, что позволяет работать с географическими данными, сложными структурами и т. д..

Выбор модели данных и типов данных зависит от конкретных требований к системе, сложности взаимосвязей данных и других факторов.

Задание 1.4. Рассмотрение вопросов, связанных со схемой и объектами схемы данных. Блоки данных, экстенты сегменты.

Теоретическая часть: Схема базы данных — это логическая структура, которая определяет организацию данных, включая таблицы, поля, связи и ограничения целостности. Она служит шаблоном для хранения, доступа и управления данными, обеспечивая согласованность и целостность системы. В реляционных базах данных схема описывает таблицы, их атрибуты и связи между ними.

Объекты схемы — это управляемые элементы внутри схемы, которые определяют структуру и расположение хранимых данных. Они включают таблицы, представления, индексы, последовательности, триггеры, хранимые процедуры, функции, синонимы, кластеры и связи баз данных. Каждый объект схемы служит определённой цели: организует данные, повышает производительность, обеспечивает целостность данных или инкапсулирует бизнес-логику.

Блоки данных, экстенты и сегменты — логические единицы управления пространством в базе данных, которые определяют, как используется физическое пространство. Эти понятия тесно связаны с управлением данными на уровне хранения.

Схема и объекты схемы

Схема — это коллекция объектов логической структуры базы данных. В Oracle схема привязывается к одному пользователю: при создании пользователем первого объекта автоматически формируется схема с таким же именем, и все последующие объекты этого пользователя становятся её частью. В PostgreSQL схемы представляют собой пространства имён для объектов базы данных (таблиц, типов данных, функций и т. д.). Они позволяют объединять объекты в логические группы, избегать конфликтов имён и упрощать управление данными.

Некоторые типы объектов схемы:

Таблицы — основные структуры для хранения данных, состоящие из строк и столбцов.

Представления — виртуальные таблицы, основанные на результатах запроса SELECT. Не хранят данные, а предоставляют альтернативный доступ к базовым таблицам.

Индексы — объекты, которые ускоряют поиск данных по определённым столбцам.

Последовательности — объекты, генерирующие уникальные порядковые номера, часто используемые для реализации полей-счётчиков в таблицах.

Триггеры — процедуры, выполняемые автоматически при возникновении определённых событий (например, вставка, обновление или удаление данных).

Хранимые процедуры и функции — программные модули, хранящиеся в базе данных и выполняемые по запросу.

Синонимы (синонимы) — альтернативные имена объектов базы данных, упрощающие доступ к ним.

Кластеры — объекты, задающие способ совместного хранения данных в нескольких таблицах, если у них есть общие ключевые поля.

Блоки данных, экстенты и сегменты

Блок данных (data block, Oracle block, страница) — наименьшая единица

логического хранения данных в базе данных. Один блок данных соответствует фиксированному количеству байт физического пространства на диске. Размер блока устанавливается при создании базы данных и кратен размеру блока операционной системы. База данных использует и распределяет свободное пространство блоками.

Экстент (extent) — единица хранения, состоящая из одного или нескольких смежных блоков данных, выделенных для хранения определённого типа информации. Когда требуется больше места, выделяется новый экстент, который может быть не смежен с предыдущими.

Сегмент (segment) — набор экстентов, выделенных для конкретного объекта базы данных (таблицы, индекса, временного объекта и т. д.). Все экстенты сегмента хранятся в одном табличном пространстве, но могут распределяться по разным файлам данных. Например, данные каждой таблицы хранятся в её собственном сегменте данных, а данные каждого индекса — в сегменте индекса.

Типы сегментов в Oracle:

сегмент данных (для таблиц);

сегмент индекса;

сегмент отката;

временный сегмент.

Взаимосвязь понятий:

Данные хранятся в блоках данных.

Несколько блоков данных образуют экстент.

Несколько экстентов составляют сегмент, который относится к конкретному объекту схемы.

Сегменты размещаются в табличных пространствах.

Дополнительные сведения

Табличное пространство (tablespace) — логическая единица хранения, контейнер для сегментов. Оно состоит из одного или нескольких файлов данных, где хранятся экстенты и сегменты.

При удалении объекта базы данных (например, таблицы) все выделенные ему экстенты освобождаются и становятся доступными для других объектов в том же табличном пространстве.

Управление пространством может осуществляться разными способами в

зависимости от типа табличного пространства (например, управляемое словарём или локально управляемое).

Эти концепции важны для администрирования баз данных, так как позволяют эффективно управлять дисковым пространством, оптимизировать производительность и обеспечивать целостность данных.

Задание 1.5. Рассмотрение вопросов, связанных с структурой памяти. Однопроцессорные и многопроцессорные базы данных.

Теоретическая часть: Структура памяти в базах данных — это организация данных и управление ресурсами для эффективного доступа и обработки информации. В зависимости от архитектуры системы (однопроцессорная или многопроцессорная) подходы к управлению памятью различаются.

Однопроцессорные базы данных

Однопроцессорные системы — это вычислительные системы с одним процессором, памятью и внешними устройствами, соединёнными через общую шину. В таких системах используется иерархическая структура памяти, которая включает несколько уровней: регистры процессора, кэш-память, основную память (ОЗУ) и внешние устройства хранения данных (диски).

Особенности управления памятью в однопроцессорных базах данных:

Память обычно не разделяется между несколькими процессорами, так как система однопроцессорная.

Все операции с данными выполняются последовательно, что ограничивает параллелизм.

Для оптимизации доступа к данным может использоваться кэш-память. Алгоритмы управления кэшем (например, сквозная запись (Write Through), обратная запись (Write Back)) влияют на производительность системы.

В некоторых СУБД память может выделяться отдельно для каждого пользователя, и данные, считанные в кэш одного пользователя, недоступны другим до повторного считывания с диска.

Ограничение по масштабируемости: при увеличении нагрузки единственный процессор может не справляться с обработкой запросов.

Многопроцессорные базы данных

Многопроцессорные системы используют несколько процессоров для

повышения вычислительной мощности. В таких системах ключевое значение имеет архитектура доступа к памяти.

Основные архитектуры доступа к памяти в многопроцессорных системах:

UMA (Uniform Memory Access — однородный доступ к памяти). Все процессоры имеют равный доступ к общей памяти. Время доступа к памяти не зависит от того, какой процессор запрашивает данные, за исключением случаев, когда данные находятся в кэше процессора. Такая архитектура также называется симметричной многопроцессорной (SMP). Процессоры совместно используют оперативную память и дисковое пространство. Пример: Oracle RAC, PostgreSQL на многоядерных серверах.

NUMA (Non-Uniform Memory Access — неоднородный доступ к памяти). Память разделена на узлы, и доступ к локальной памяти быстрее, чем к удаленной. Процессоры имеют предпочтительный доступ к определённой части памяти, но могут обращаться и к другим узлам.

NORMA (No Remote Memory Access — без доступа к удаленной памяти). В таких системах нет общей памяти, каждый узел имеет только локальную память. Для обмена данными используется сетевой интерфейс или общая виртуальная память (SVM).

Особенности многопроцессорных баз данных:

Требуется синхронизация доступа к данным, чтобы избежать конфликтов при одновременном обращении нескольких процессоров к одним и тем же данным.

Необходимо эффективное управление общей памятью, особенно в архитектурах UMA/SMP. Это включает динамическое распределение буферов, кэширование и механизмы блокировок.

Возможность параллельной обработки запросов и операций (например, загрузка данных, резервное копирование, создание индексов).

В системах с разделяемыми ресурсами (например, SMP) может возникать конкуренция за доступ к памяти и другим ресурсам. Для снижения конкуренции применяется логическое разделение (logical partitioning), при котором каждый раздел базы данных работает со своим участком памяти.

Сравнение однопроцессорных и многопроцессорных систем

Критерий Однопроцессорные системы Многопроцессорные системы

Количество процессоров Один Несколько

Доступ к памяти Единый процессор управляет всей памятью В зависимости от архитектуры (UMA, NUMA, NORMA)

Параллелизм Ограничен одним процессором Возможность параллельной обработки запросов и операций

Масштабируемость Ограничена производительностью процессора Выше за счёт добавления процессоров, но может ограничиваться конкуренцией за ресурсы

Сложность управления Ниже Выше из-за необходимости синхронизации и управления доступом к общим ресурсам

Дополнительные аспекты

В СУБД, таких как Oracle, память организуется в структуры вроде SGA (System Global Area) — общая область памяти, содержащая данные и управляющие структуры для экземпляра базы данных, и PGA (Program Global Area) — неразделяемая область памяти для каждого процесса.

В многопроцессорных системах важно эффективное планирование процессов и распределение нагрузки между процессорами, чтобы избежать простаивания ресурсов.

Таким образом, выбор между однопроцессорной и многопроцессорной архитектурой зависит от требований к производительности, масштабируемости и сложности системы. Многопроцессорные системы подходят для высоконагруженных и сложных задач, но требуют более сложного управления ресурсами.

Задание 1.6. Рассмотрение вопросов, связанных с транзакциями, блокировкой и согласованностью данных.

Теоретическая часть: Транзакции, блокировки и согласованность данных — ключевые концепции в управлении базами данных, обеспечивающие целостность, надёжность и корректную работу с данными в многопользовательской среде.

Транзакции

Транзакция — это последовательность операций с данными, которая выполняется как единое целое. Либо все операции транзакции выполняются успешно, либо ни одна из них не применяется. Цель транзакции — гарантировать, что база данных перейдёт из одного согласованного состояния в другое.

Пример транзакции: перевод денег с одного счёта на другой. Транзакция включает два действия: уменьшение баланса на счёте-отправителе и увеличение баланса на счёте-получателе. Если одно из действий не выполнится, вся транзакция откатится, и данные останутся в исходном состоянии.

Свойства транзакций (ACID)

Транзакции описываются набором требований ACID:

Атомарность (Atomicity). Транзакция выполняется полностью или не выполняется вовсе. Нет промежуточных состояний.

Согласованность (Consistency). После завершения транзакции база данных должна находиться в допустимом состоянии, соответствующем всем правилам целостности (ограничениям, ссылкам и т. д.). Например, если в таблице есть внешний ключ, запись, на которую он указывает, должна существовать после транзакции.

Изолированность (Isolation). Параллельные транзакции не влияют на результат друг друга. Каждая транзакция выполняется так, как будто других транзакций нет.

Долговечность (Durability). Изменения, сделанные успешной транзакцией, сохраняются навсегда, даже при сбое системы.

Блокировки

Блокировка — это временное ограничение доступа к данным для обеспечения корректной обработки транзакций. Она предотвращает конфликты при одновременном доступе нескольких транзакций к одним и тем же данным.

Виды блокировок:

Эксклюзивная (Exclusive Lock). Даёт одной транзакции исключительный доступ к данным. Другие транзакции не могут получить доступ к заблокированным данным до снятия блокировки.

Совместная (Shared Lock). Позволяет нескольким транзакциям одновременно читать данные, но не изменять их.

По области действия блокировки бывают:

строчные (на одну строку таблицы);

гранулярные (на всю таблицу или страницу);

предикатные (на область, ограниченную предикатом, например, диапазон ключей).

По логике реализации:

пессимистические (накладываются перед предполагаемой модификацией данных);

оптимистические (не ограничивают модификацию данных, но перед фиксацией изменений проверяют, не изменились ли данные другими транзакциями).

Проблемы параллельного выполнения транзакций

При одновременном выполнении нескольких транзакций могут возникать следующие проблемы:

Потерянное обновление (Lost Update). Если две транзакции изменяют одни и те же данные, изменения одной из них могут быть перезаписаны другой.

Грязное чтение (Dirty Read). Транзакция читает данные, которые ещё не были зафиксированы другой транзакцией. Если эта транзакция откатится, прочитанные данные окажутся некорректными.

gb.ru +1

Неповторяющееся чтение (Non-Repeatable Read). При повторном чтении одних и тех же данных в рамках транзакции значения могут измениться из-за действий других транзакций.

Фантомное чтение (Phantom Read). В ходе выполнения транзакции появляются или исчезают строки, которые соответствуют условиям выборки. Например, одна транзакция несколько раз выбирает данные, а другая в это время вставляет или удаляет строки.

Уровни изоляции транзакций

Уровень изоляции определяет, какие проблемы параллелизма могут возникать при выполнении транзакций. Стандарт SQL-92 определяет четыре уровня:

Уровень Описание

Read Uncommitted (чтение незафиксированных данных) Самый слабый уровень. Допускает все типы проблем с согласованностью.

Read Committed (чтение зафиксированных данных) Исключает грязное чтение, но допускает неповторяющееся и фантомное чтение.

Repeatable Read (повторяемое чтение) Исключает грязное и неповторяющееся чтение, но допускает фантомное чтение.

Serializable (сериализуемый) Самый строгий уровень. Исключает все

проблемы параллелизма, транзакции выполняются так, как если бы они были последовательными.

Повышение уровня изоляции увеличивает согласованность данных, но может снижать производительность из-за роста числа блокировок.

Согласованность данных

Согласованность данных — это состояние, при котором данные в базе непротиворечивы, соответствуют внутренней логике, структуре и всем заданным правилам. Это включает корректность типов данных, соблюдение ограничений целостности (уникальности, ссылочной целостности и т. д.).

Для обеспечения согласованности используются:

ограничения целостности (уникальность, внешние ключи и т. д.);

механизмы блокировок;

журналирование транзакций (для отката при сбоях).

Дополнительные аспекты

Откат транзакции (Rollback). При ошибке или явном указании система отменяет все изменения, сделанные в рамках транзакции, возвращая данные в исходное состояние.

Журнал транзакций. Записи о всех изменениях сохраняются в журнале, что позволяет восстановить состояние базы данных после сбоя.

Распределённые транзакции. Охватывают несколько баз данных или систем. Для их согласования используются протоколы, например, двухфазный коммит (2PC), который гарантирует, что транзакция либо завершится успешно на всех узлах, либо будет откачена.

Таким образом, транзакции, блокировки и уровни изоляции работают вместе, обеспечивая целостность, согласованность и надёжность данных в базах данных.

Практическая часть: Продумать ограничения для БД с блокировкой при их нарушении.

Задание 1.7. Журнал базы данных: структура и назначение файлов журнала, управление переключениями и контрольными точками.

Теоретическая часть: Журнал базы данных — это файл или набор файлов, в которых фиксируются все изменения, внесённые в базу данных. Он играет

ключевую роль в обеспечении надёжности, согласованности данных и возможности восстановления после сбоев.

Структура файлов журнала

Структура файлов журнала может различаться в зависимости от СУБД. Например, в PostgreSQL используются файлы с префиксом pg_ и последовательным номером. Некоторые из них:

pg_xlog (в новых версиях — pg_wal) — основной файл журнала, хранящий информацию обо всех изменениях в таблицах, индексах и других объектах базы данных. Файлы pg_xlog хранятся в каталоге pg_xlog и называются по принципу 000000010000000000000001.00000002.

pg_clog — файл, хранящий информацию о блокировках транзакций.

pg_subtrans — файл с информацией о подтранзакциях.

pg_multixact — файл с информацией о многоверсионных транзакциях.

В SQL Server файлы журнала транзакций имеют рекомендуемое расширение .ldf. Каждый раз, когда данные меняются, SQL Server сохраняет информацию об операции в журнал транзакций, чтобы отменить (откатить) или повторить (воспроизвести) эти действия.

interface.ru +2

Назначение файлов журнала

Восстановление после сбоя. В случае сбоя системы СУБД может использовать файлы журнала для восстановления базы данных до последнего согласованного состояния.

Обеспечение ACID-свойств. Журнал транзакций гарантирует, что все изменения выполняются атомарно, согласованно, изолированно и долговечно.

Репликация. Журнал транзакций используется для репликации данных на другие серверы, что позволяет создать отказоустойчивую систему.

Точечное восстановление. Журнал позволяет восстановить базу данных до определённого момента времени, что может быть полезно для восстановления удалённых данных или отката изменений.

Управление переключениями журнала

Переключение журнала — это процесс создания новых файлов журнала для записи изменений. Когда текущий файл журнала достигает определённого размера,

он становится активным, а старый — доступен для архивирования. Это предотвращает переполнение диска и улучшает производительность восстановления.

В Oracle процесс LGWR (Log Writer) выполняет переключение журнала, выполняя следующие шаги:

Выполняет транзакцию управляющего файла (controlfile transaction) для выбора следующего используемого файла журнала и очистки соответствующей записи в управляющем файле.

Сбрасывает любую оставшуюся в буфере журнала информацию на диск, а затем записывает номер системного изменения (SCN) последней записи в журнальном файле.

Закрывает текущий журнальный файл.

Увеличивает значение SCN и выполняет вторую транзакцию управляющего файла, чтобы пометить следующий файл журнала как текущий (CURRENT), а прежний — как активный (ACTIVE).

Переключение журнала можно принудительно вызвать вручную с помощью оператора ALTER SYSTEM SWITCH LOGFILE или ALTER SYSTEM ARCHIVE LOG CURRENT.

Контрольные точки

Контрольная точка — это момент времени в журнале, в котором установлено известное и непротиворечивое состояние системы базы данных. Во время выполнения контрольной точки все «грязные» страницы данных, находящиеся в памяти, сбрасываются на диск, а в файле журнала создаётся специальная запись контрольной точки. После завершения контрольной точки журнальные записи, предшествующие её началу, больше не нужны для восстановления.

Типы контрольных точек:

Автоматические. Возникают, когда количество записей журнала достигает значения, которое СУБД может обработать в течение времени, указанного в параметре конфигурации сервера (например, recovery interval в SQL Server).

Косвенные. Позволяют управлять временем восстановления базы данных за счёт постоянной записи грязных страниц на диск в фоновом режиме.

Ручные. Инициируются вручную с помощью команд (например,

CHECKPOINT в SQL Server).

Внутренние. Создаются различными серверными компонентами в ответ на определённые события (например, резервное копирование, создание моментального снимка базы данных).

За работу контрольных точек в PostgreSQL отвечает фоновый процесс checkpointer. В SQL Server ядро СУБД поддерживает несколько типов контрольных точек, и их поведение можно настраивать с помощью параметров конфигурации.

Назначение контрольных точек:

Сокращение времени восстановления, так как при сбое система начинает восстановление с последней контрольной точки, а не с начала журнала.

Уменьшение размера журнала, так как после контрольной точки можно удалить журнальные записи, предшествующие её началу.

Обеспечение согласованности данных на диске, так как все изменения, зафиксированные до контрольной точки, гарантированно записаны на физический носитель.

Параметры управления контрольными точками могут включать:

wal_level — определяет уровень записи в журнал (PostgreSQL).

checkpoint_segments — определяет количество сегментов журнала, которые будут записаны в базу данных при выполнении checkpoint (PostgreSQL).

wal_buffers — определяет количество буферов для записи в журнал (PostgreSQL).

wal_writer_delay — определяет задержку между записью в журнал и записью в базу данных (PostgreSQL).

recovery interval — параметр конфигурации сервера, определяющий максимальное время, которое экземпляр сервера должен использовать для восстановления базы данных при перезагрузке системы (SQL Server).

Правильное управление журналом транзакций, включая переключения и контрольные точки, является ключевым фактором для надёжной работы СУБД.

Практическая часть: Создать контрольную точку сохранения БД.

Задание 1.8. Словарь данных: назначение, структура, префиксы.

Теоретическая часть: Словарь данных в системах управления базами данных

Словарь данных (Data Dictionary) — это централизованное хранилище

метаданных, содержащее информацию о структуре базы данных, её объектах и их характеристиках.

Основные функции:

Управление данными: проверка корректности запросов, создание и модификация объектов

Оптимизация: планирование запросов и выбор эффективных планов выполнения

Контроль доступа: управление правами пользователей

Документация: поддержка инструментов разработки и администрирования

Структура словаря данных

Основные компоненты:

Системные таблицы с префиксом pg_:

pg_database — информация о базах данных

pg_namespace — данные о пространствах имён

pg_class — описание таблиц, представлений, индексов

pg_attribute — характеристики столбцов

pg_type — информация о типах данных

pg_proc — данные о функциях

pg_constraint — описание ограничений

pg_user и pg_group — учётные данные пользователей

Пространства имён:

pg_catalog — системное пространство для таблиц словаря

information_schema — стандартное пространство для SQL-совместимости

Типы словарей данных

Активный словарь — автоматически обновляется при изменениях в структуре БД

Пассивный словарь — требует ручного обновления документации

Префиксы и их назначение

Основные префиксы в системах управления базами данных:

pg_ — системные таблицы словаря данных

pg_catalog — пространство имён для системных объектов

information_schema — стандартное пространство для SQL-совместимости

USER — представления для текущего пользователя

ALL — расширенные представления с правами доступа

DBA — представления для администраторов

V$ — динамические представления состояния БД

Практическое применение

Основные сценарии использования:

Анализ структуры базы данных

Проверка зависимостей между объектами

Мониторинг прав доступа

Оптимизация производительности

Документирование изменений

Правила работы со словарём данных

Важные принципы:

Словарь доступен только для чтения

Изменения в словаре производятся через специальные команды

Необходимо соблюдать правила нормализации данных

Требуется контроль целостности метаинформации

Примеры запросов к словарю данных

sql

```
-- Получение информации о таблицах
SELECT * FROM pg_catalog.pg_tables
WHERE schemaname = 'public';
-- Данные о столбцах таблицы
SELECT * FROM pg_catalog.pg_attribute
WHERE attrelid = 'users'::regclass AND attnum > 0;
-- Информация о типах данных
SELECT * FROM pg_catalog.pg_type;
```

Словарь данных является критически важным компонентом любой СУБД, обеспечивающим:

Целостность и корректность данных

Эффективность работы системы

Безопасность доступа

Возможность масштабирования

Задание 1.9. Рассмотрение распределённых баз данных.

Теоретическая часть: Распределённая база данных — это совокупность логически взаимосвязанных данных, которые физически распределены в компьютерной сети на нескольких узлах (компьютерах, серверах). При этом данные могут пересекаться или дублироваться, а система управления обеспечивает их согласованность и доступность как единой базы.

Система управления распределённой базой данных (РСУБД) — программная система, которая управляет распределёнными данными и делает их распределённость прозрачной для пользователя. Каждый фрагмент базы данных работает под управлением отдельной СУБД, которая осуществляет доступ к данным этого фрагмента.

Основные принципы распределённых баз данных

К. Дж. Дейт сформулировал фундаментальный принцип: для пользователя распределённая система должна выглядеть так же, как нераспределённая. Из этого следуют ряд правил:

Локальная автономия. Узел должен функционировать даже в изоляции, отвечая на запросы на чтение и изменение данных.

Отсутствие центрального узла. Нет единой точки отказа.

Непрерывное функционирование. Система должна работать без плановых отключений.

Независимость от расположения. Пользователь не должен знать, где физически находятся данные.

Независимость от фрагментации. Программы не должны «знать», на каком узле находятся данные.

Независимость от репликации. Автоматическая поддержка копий данных на разных узлах.

Обработка распределённых запросов. Один запрос может получать данные с разных узлов.

Управление распределёнными транзакциями. Транзакция, затрагивающая несколько узлов, должна быть либо зафиксирована везде, либо отменена везде.

Независимость от аппаратного обеспечения, ОС, сети и типа СУБД.

Типы распределённых СУБД

Тип Описание

Однородная Все узлы используют одинаковое ПО и архитектуру СУБД.

Гетерогенная Узлы могут использовать разное ПО, модели данных или архитектуры.

Федеративная Локальные базы данных поддерживаются отдельными сайтами или федерациями и объединяются через промежуточное ПО.

Реплицированная Поддерживает несколько копий одного и того же фрагмента данных на разных узлах для обеспечения доступности и отказоустойчивости.

Секционированная База данных разделена на разделы, каждый из которых назначен определённому узлу.

Гибридная Комбинация нескольких типов для удовлетворения конкретных требований.

Фрагментация данных

Фрагментация — разделение базы данных на более мелкие подмножества (фрагменты). Виды фрагментации:

Горизонтальная — разделение по строкам в зависимости от условий.

Вертикальная — разделение по столбцам.

Гибридная — сочетание горизонтальной и вертикальной фрагментации.

Репликация данных

Репликация — создание нескольких копий одних и тех же данных на разных узлах. Используется для обеспечения доступности, отказоустойчивости и бесперебойной работы. Требует качественной синхронизации изменений между репликами.

Распределённые транзакции

Распределённая транзакция — транзакция, которая затрагивает несколько узлов. Для её обработки часто используется протокол двухфазной фиксации (2PC):

Подготовка: все узлы блокируют ресурсы и подтверждают готовность к фиксации.

Фиксация или откат: координатор отправляет команду на фиксацию или откат всем узлам.

Проблемы распределённых баз данных

Синхронизация данных. Необходимо согласовывать состояние данных на разных узлах, особенно при репликации и распределённых транзакциях.

Сетевые задержки и сбои. Передача данных через сеть может вызывать задержки, а сбои связи — нарушать работу системы.

Масштабируемость. С ростом объёмов данных и нагрузки на систему возникает необходимость горизонтального масштабирования (добавления новых узлов).

Интеграция и совместимость. В гетерогенных системах сложно обеспечить совместимость разных СУБД, форматов данных и схем.

Управление каталогом. Требуется глобальный каталог, содержащий информацию о каждом фрагменте БД и его местоположении в сети.

Безопасность. Распределённые системы подвержены угрозам утечки данных, атакам и взломам.

CAP-теорема

Согласно CAP-теореме, в распределённой системе можно обеспечить только два из трёх свойств одновременно:

Consistency — согласованность данных на всех узлах.

Availability — доступность системы для ответов на запросы.

Partition tolerance — устойчивость к разрывам связи между узлами.

При разрыве связи система может частично отказаться от доступности (функционирует только одна часть системы) или от согласованности (не все узлы будут в согласованном состоянии).

Преимущества распределённых баз данных

Приближение данных к месту их порождения повышает достоверность данных.

Отказоустойчивость за счёт репликации и распределения данных по разным узлам.

Масштабируемость — возможность добавлять новые узлы для обработки растущей нагрузки.

Гибкость — поддержка разных моделей данных и СУБД в гетерогенных системах.

Распределённые базы данных широко применяются в крупных организациях,

облачных сервисах, финансовых системах, e-commerce и других сферах, где требуется обработка больших объёмов данных, высокая доступность и масштабируемость.

Практическая часть: Разработать распределенную БД.

Задание 1.10. Выбор среды для реализации создаваемой учебной базы данных.

Практическая часть: Описать выбранную вами среду разработки БД.

Задание 1.11. Определение обязанностей администратора применительно к создаваемой учебной базы данных.

Практическая часть: Расписать обязанности администратора вашей БД.

Задание 1.12. Построение схемы создаваемой учебной базы данных.

Практическая часть: Разработать физическую ER-модель вашей БД.

Задание 1.13. Построение словаря создаваемой учебной базы данных.

Практическая часть: Разработать словарь вашей БД.

Задание 1.14. Изучение команд администрирования данных, применительно к создаваемой учебной базе данных.

Практическая часть: Рассмотреть команды администрирования реализуемые вашей СУБД.

Задание 1.15. Разработка сценариев работы с данными в создаваемой учебной базе данных.

Теоретическая часть: Сценарий работы с данными учебной базы данных «Библиотека»

Описание системы

Учебная база данных предназначена для автоматизации работы библиотеки. Основные сущности: книги, читатели, выдачи.

Сценарий 1: Добавление новой книги

Цель сценария: внесение информации о новой книге в базу данных

Участники: библиотекарь

Предварительные условия:

Пользователь авторизован как библиотекарь

Книга физически поступила в библиотеку

Основной поток событий:

Библиотекарь открывает форму добавления книги

Заполняет обязательные поля:

ISBN

Название

Автор

Год издания

Издательство

Количество экземпляров

Добавляет дополнительную информацию:

Жанр

Язык

Аннотация

Сохраняет запись

Система:

Проверяет уникальность ISBN

Создает уникальный идентификатор книги

Сохраняет данные в таблицу книг

Создает записи в таблице экземпляров

Поток исключений:

При совпадении ISBN — сообщение об ошибке

При незаполнении обязательных полей — запрос на заполнение

Сценарий 2: Выдача книги читателю

Цель сценария: оформление выдачи книги читателю

Участники: библиотекарь, читатель

Предварительные условия:

Книга доступна (не выдана другим читателям)

Читатель имеет действующий читательский билет

Основной поток событий:

Библиотекарь сканирует штрих-код книги или вводит её идентификатор

Система:

Проверяет доступность книги

Показывает информацию о книге

Библиотекарь вводит данные читателя

Система:

Проверяет наличие задолженностей

Определяет срок возврата

Оформляет выдачу:

Создает запись в журнале выдач

Изменяет статус книги на “Выдана”

Печатает читательский билет

Поток исключений:

Книга отсутствует — сообщение об ошибке

Задолженности у читателя — предложение погасить

Технические проблемы — отмена операции

Сценарий 3: Поиск книги

Цель сценария: поиск книги по различным критериям

Участники: читатель, библиотекарь

Предварительные условия:

Пользователь авторизован

База данных доступна

Основной поток событий:

Пользователь открывает форму поиска

Задает параметры поиска:

Название

Автор

ISBN

Жанр

Система:

Выполняет поиск

Показывает результаты

Позволяет уточнить критерии

Поток исключений:

Нет результатов — сообщение об отсутствии

Ошибка поиска — повтор запроса

Сценарий 4: Возврат книги

Цель сценария: оформление возврата книги

Участники: библиотекарь, читатель

Предварительные условия:

Книга выдана данному читателю

Прошел срок возврата

Основной поток событий:

Библиотекарь сканирует штрих-код книги

Система:

Проверяет статус книги

Показывает информацию о выдаче

Библиотекарь подтверждает возврат

Система:

Обновляет статус книги

Создает запись о возврате

Рассчитывает штрафы (если есть)

Поток исключений:

Повреждение книги — оформление акта

Потеря книги — оформление заявления

Общие требования к сценариям

Валидация данных:

Проверка обязательных полей

Контроль форматов

Уникальность значений

Логирование:

Запись всех операций

Сохранение времени и пользователя

Фиксация изменений

Безопасность:

Контроль прав доступа

Аутентификация пользователей

Шифрование конфиденциальных данных

Практическая часть: Разработать сценарии работы с вашей БД.

Тема 2. Серверы баз данных.

В современном мире информационных технологий серверы баз данных являются критически важным компонентом любой организации, работающей с данными. Они обеспечивают надежное хранение, обработку и предоставление доступа к информации для множества пользователей одновременно.

Сервер баз данных представляет собой специализированную программно-аппаратную платформу, предназначенную для управления структурированными данными. Он включает в себя:

Аппаратную часть (серверное оборудование).

Программное обеспечение СУБД.

Системы хранения данных.

Механизмы обеспечения безопасности.

Основные функции серверов баз данных.

Хранение и защита данных.

Обработка запросов пользователей.

Обеспечение многопользовательского доступа.

Управление транзакциями.

Резервное копирование и восстановление.

Оптимизация производительности.

Задание 2.1. Рассмотрение вопросов, связанных с классификацией серверов, принципами разделения между клиентскими и серверными частями. Типовое разделение функций.

Теоретическая часть: Классификация серверов может осуществляться по разным критериям, например:

По форме представления (физической реализации):

Tower-серверы — вертикальные шкафы, напоминающие башни. Компоненты расположены на расстоянии друг от друга, что снижает нагрев. Несколько таких

серверов можно объединить в сеть.

Rack-серверы (стоечные) — автономные компьютеры, которые устанавливают в специальные вертикальные шкафы с полками-стойками.

Блейд-серверы — компактные серверы, которые монтируют в специальные шасси (блейд-корзины).

По классу (в зависимости от производительности, надёжности и масштабируемости):

Начального уровня — для небольших компаний или отделов, с ограниченной производительностью и масштабируемостью.

Для рабочих групп — обслуживают небольшие рабочие группы, имеют больший уровень производительности, чем серверы начального уровня.

Уровень департамента — для департаментов в компании, обеспечивают высокий уровень производительности и масштабируемости.

Уровень предприятия — самые мощные серверы для крупных корпораций, поддерживают работу с большим количеством пользователей.

По назначению:

Файловые — обеспечивают общий доступ к документам и файлам.

Базы данных — управляют базами данных, обрабатывают запросы, структурируют данные.

Веб-серверы — обрабатывают запросы от клиентов через протокол HTTP, доставляют веб-страницы и другой контент.

Почтовые — управляют отправкой, приёмом и хранением электронных писем.

Прокси-серверы — обрабатывают запросы от клиентов перед тем, как они достигнут конечного сервера.

DNS-серверы — преобразуют доменные имена в IP-адреса.

Серверы приложений — предоставляют инфраструктуру для работы приложений.

Терминальные — позволяют нескольким пользователям работать на одной физической машине одновременно.

Кластерные — объединяют несколько физических или виртуальных серверов в единую систему.

Медиа-серверы — предназначены для потоковой передачи мультимедийного

контента.

Игровые — обеспечивают инфраструктуру для многопользовательских онлайн-игр.

Бекап-серверы (системы резервного копирования) — копируют и клонируют данные, файлы, приложения и информационные базы.

Принципы разделения между клиентскими и серверными частями в клиент-серверной архитектуре основаны на принципе разделения ответственности. Клиент и сервер имеют чёткое разграничение функций:

Клиент инициирует запросы к серверу, обычно в ответ на действия пользователя или входные данные. Он отвечает за пользовательский интерфейс, формирование запросов и обработку ответов от сервера.

Сервер пассивно ожидает входящих запросов, обрабатывает их, применяя бизнес-логику и взаимодействуя с хранилищем данных, а затем формирует и отправляет ответы клиентам.

Некоторые фундаментальные принципы клиент-серверной архитектуры:

Модульность — система структурируется на независимые компоненты, каждый из которых выполняет конкретную задачу.

Масштабируемость — система может адаптироваться к растущей нагрузке за счёт добавления серверов или обновления аппаратных ресурсов.

Надёжность и доступность — резервирование и механизмы отказоустойчивости обеспечивают непрерывную работу при сбоях.

Централизация ресурсов — серверы консолидируют данные, бизнес-логику и вычислительные ресурсы, что упрощает управление и обеспечивает централизованные политики безопасности и доступа к данным.

Стандартизированные протоколы — обеспечивают согласованное взаимодействие между клиентом и сервером (например, HTTP/HTTPS, TCP/IP, WebSocket).

Типовое разделение функций может варьироваться в зависимости от модели архитектуры:

Модель Клиентская часть Серверная часть

Тонкий клиент Минимальная логика, преимущественно UI Вся бизнес-логика и обработка данных

Толстый клиент Значительная часть бизнес-логики Управление данными и критичная логика

Примеры моделей разделения функций:

Модель удалённого доступа к данным (RDA). Программы, реализующие функции представления информации и логику прикладной обработки, совмещены и выполняются на клиенте. Обращение за сервисом управления данными происходит через среду передачи с помощью операторов языка SQL или вызова функций специальной библиотеки API.

Модель распределённого представления. Основную часть функций представления информации реализуют СУБД, а окончательное построение изображения на терминалах пользователя выполняется на оконечных устройствах.

Модель распределённой функции. Логика обработки данных распределена по двум узлам: общая часть прикладных функций реализована на сервере, а специфические функции обработки информации выполняются на клиенте.

Трёхзвенная модель (AS-модель, Application Server). Каждая из трёх функций приложения (представление информации, обработка бизнес-логики, управление данными) реализуется на отдельном компьютере или сервере.

Клиент-серверная архитектура позволяет эффективно распределять нагрузку, улучшать масштабируемость и обеспечивать более эффективную обработку запросов.

Задание 2.2. Рассмотрение вопросов, связанных с протоколами удаленного вызова процедур. Требования к аппаратным возможностям и базовому программному обеспечению клиентов и серверов.

Теоретическая часть: Удалённый вызов процедур (RPC, Remote Procedure Call) — это технология, которая позволяет программе вызывать функции или процедуры в другом адресном пространстве (на удалённом узле или в независимой системе) так, словно они являются частью локальной программы. RPC упрощает создание распределённых систем, скрывая детали сетевого взаимодействия от разработчика.

Принципы работы RPC

Механизм RPC включает несколько ключевых этапов:

Клиент вызывает «заглушку» (stub) — локальную функцию, которая на самом деле является прокси для удалённого вызова.

Клиентская заглушка сериализует параметры процедуры и отправляет их на сервер через сетевой протокол.

Серверная заглушка получает сообщение, десериализует параметры и вызывает реальную реализацию удалённой процедуры.

После выполнения процедуры сервер сериализует результат и отправляет его клиенту.

Клиентская заглушка десериализует ответ и возвращает результат вызывающей программе.

Характерные черты RPC:

Асимметричность. Одна из взаимодействующих сторон (клиент) является инициатором вызова.

Синхронность. Выполнение вызывающей процедуры приостанавливается до получения ответа от удалённой процедуры.

Прозрачность — ключевой принцип RPC: программисту кажется, что он вызывает локальную функцию, а все детали сетевого взаимодействия скрыты.

Протоколы и технологии RPC

На транспортном уровне RPC чаще всего используют протоколы TCP и UDP, но некоторые реализации базируются на HTTP. Для сериализации данных применяются различные форматы, например:

Protocol Buffers (protobuf) — бинарный формат, используемый в gRPC.

JSON — используется в JSON-RPC.

XML — применяется в XML-RPC и SOAP.

Примеры технологий RPC:

gRPC — современный фреймворк от Google, использует protobuf и HTTP/2, поддерживает стриминг и аутентификацию.

Apache Thrift — фреймворк для создания кросс-языковых сервисов, имеет собственный IDL.

JSON-RPC — простой текстовый протокол на базе HTTP.

XML-RPC — использует XML для кодирования сообщений.

SOAP — текстовый протокол на базе HTTP, изначально ориентирован на RPC,

но позже эволюционировал в общий формат обмена сообщениями.

DCE/RPC (Distributed Computing Environment RPC) — стандарт, разработанный консорциумом Open Software Foundation, поддерживается IBM, HP и другими.

ONC RPC (Open Network Computing RPC) — разработан Sun Microsystems, используется в NFS.

Требования к аппаратным возможностям

Аппаратные требования зависят от масштаба и нагрузки системы. Например, для RPC-узлов в сети NEAR рекомендуются:

Процессор: 8 ядер (16 потоков) Intel i7/Xeon или эквивалент с поддержкой инструкций CMPXCHG16B, POPCNT, SSE4.1, SSE4.2, AVX, SHA-NI.

Оперативная память: 32 ГБ DDR4 (минимально — 16 ГБ).

Хранилище: 4 ТБ NVMe SSD (минимально — 2,5 ТБ SATA3-class SSD с поддержкой не менее 15 тыс. IOPS и 800 MiBps).

Для высоконагруженных систем могут потребоваться более мощные процессоры, увеличенный объём ОЗУ и высокоскоростные накопители.

Требования к базовому программному обеспечению

Для серверов и клиентов необходимы:

Операционная система, поддерживающая RPC. Например, для работы с MSMQ (Message Queuing) в Windows требуются Windows Server 2003, Windows XP, Windows 2000 или более поздние версии.

Библиотеки и компоненты для работы с RPC. Например, для MSMQ нужны серверная (RpcMqSvr.dll) и клиентская (RpcMqCl.dll) транспортные DLL.

Компилятор IDL (Interface Definition Language), если используется описание интерфейсов. IDL позволяет определить структуру вызовов процедур, а компилятор генерирует код заглушек для клиента и сервера.

Поддержка сериализации/десериализации данных в соответствии с выбранным форматом (protobuf, JSON, XML и т. д.).

Транспортные протоколы (TCP, HTTP/2 и т. п.) и сетевые стеки, обеспечивающие доставку сообщений.

Для некоторых технологий могут потребоваться дополнительные компоненты, например, балансировщики нагрузки, системы аутентификации или мониторинга.

Важно учитывать:

При работе с разнородными системами (разные ОС, архитектуры процессоров) необходимо решать проблемы совместимости форматов данных и порядков байтов.

Для обеспечения безопасности могут потребоваться реализации шифрования, аутентификации и авторизации.

Выбор конкретных технологий и требований зависит от специфики проекта, требований к производительности, масштабируемости и совместимости.

Задание 2.3. Рассмотрение вопросов, связанных с хранимыми процедурами и триггерами.

Теоретическая часть: Хранимая процедура — это набор SQL-инструкций, который компилируется один раз и хранится на сервере базы данных. Она может принимать входные параметры, выполнять различные операции с данными (включая запросы, вставку, обновление и удаление), а также возвращать результаты. Хранимые процедуры похожи на обычные процедуры языков высокого уровня: в них могут использоваться числовые вычисления, операции над символьными данными, циклы и ветвления.

Триггер — это особый тип хранимой процедуры, которая автоматически выполняется в ответ на определённое событие в базе данных. Обычно такие события связаны с операциями вставки, обновления или удаления данных в определённой таблице. Триггеры используются для автоматизации задач, обеспечения целостности данных, аудита изменений и других целей.

Хранимые процедуры

Некоторые преимущества хранимых процедур:

Повышение производительности. Хранимые процедуры компилируются и сохраняются в базе данных, что ускоряет их выполнение, особенно при частом использовании.

Безопасность. Позволяют ограничить доступ к базовым таблицам и данным, предоставляя пользователям доступ только к определённым процедурам.

Модульность и упрощение поддержки. Приложения, работающие с одной и той же базой данных, могут использовать одну и ту же хранимую процедуру, что уменьшает размер кода и упрощает его поддержку.

Согласованность. Позволяют обеспечивать согласованность операций с базой данных, инкапсулируя сложную логику и применяя бизнес-правила.

Пример хранимой процедуры в MySQL, которая вставляет нового клиента в базу данных:

sql

```
CREATE PROCEDURE insert_customer
    @name VARCHAR(50),
    @email VARCHAR(50),
    @phone VARCHAR(20)
AS
BEGIN
    INSERT INTO customers (name, email, phone)
    VALUES (@name, @email, @phone);
END
```

Вызов процедуры: EXEC insert_customer 'Иван', 'ivan@example.com', '123-456-7890'.

Триггеры

Триггеры могут использоваться для:

проверки целостности данных;

отката ошибочных транзакций;

ограничения реализации значений внутри базы данных;

сбора статистики;

оптимизации запросов.

Ключевые слова, определяющие момент запуска триггера:

BEFORE — триггер запускается до выполнения события;

AFTER — после события.

blog.skillfactory.ru +1

Типы триггеров по уровню действия:

FOR EACH ROW. Триггер выполняется для каждой строки, затронутой операцией.

FOR EACH STATEMENT. Триггер выполняется один раз для каждого оператора, независимо от количества изменяемых строк.

Пример триггера в PostgreSQL, который обновляет поле LastUpdated перед каждым обновлением записи в таблице Customers:

sql

```
DELIMITER //
CREATE TRIGGER UpdateCustomersLastUpdated
BEFORE UPDATE ON Customers
FOR EACH ROW
BEGIN
   SET NEW.LastUpdated = NOW();
END //
DELIMITER ;
```

Ещё один пример — триггер, который создаёт запись в таблице AuditLog после удаления записи из таблицы Customers:

sql

```
DELIMITER //
CREATE TRIGGER LogCustomerDeletion
AFTER DELETE ON Customers
FOR EACH ROW
BEGIN
   INSERT INTO AuditLog (TableName, RecordID, Action, Timestamp)
   VALUES ('Customers', OLD.CustomerID, 'DELETE', NOW());
END //
DELIMITER ;
```

Сравнение хранимых процедур и триггеров

Критерий Хранимая процедура Триггер

Вызов Вызывается явно пользователем или приложением с помощью команды CALL или EXECUTE Автоматически выполняется в ответ на определённое событие (INSERT, UPDATE, DELETE)

Цель Автоматизация сложных операций, инкапсуляция бизнес-логики Обеспечение целостности данных, аудит, реализация простых действий при изменении данных

УправлениеМожет управлять транзакциями (COMMIT, ROLLBACK)

Выполняется в контексте транзакции, вызвавшей событие. Если в триггере возникает ошибка, транзакция может быть откачена

Важно: триггеры следует использовать с осторожностью, так как они могут влиять на производительность базы данных. Их нужно тщательно тестировать, чтобы избежать ошибок и непредвиденных последствий.

Задание 2.4. Рассмотрение вопросов, связанных с характеристиками серверов баз данных и механизмами доступа к базам данных.

Теоретическая часть: Сервер баз данных — это специализированное программное обеспечение, которое управляет базами данных, обеспечивает их хранение, обработку запросов, целостность, безопасность и доступ к информации. Он выступает в качестве централизованного хранилища, предоставляя авторизованным пользователям и приложениям доступ к данным.

Характеристики серверов баз данных

При выборе сервера баз данных необходимо учитывать ряд ключевых характеристик, которые влияют на его производительность, масштабируемость и надёжность.

Процессор (CPU). Определяет скорость обработки запросов к БД. Для больших объёмов данных и высокой нагрузки требуются многоядерные процессоры с высокой тактовой частотой.

Оперативная память (RAM). Используется для кэширования часто используемых данных и ускорения обработки запросов. Недостаток оперативной памяти может значительно снизить производительность.

Хранилище данных. SSD-накопители обеспечивают более высокую скорость чтения и записи по сравнению с HDD, что критично для производительности. Для больших объёмов данных могут применяться RAID-массивы.

Сетевая карта. Определяет скорость обмена данными. Для высоконагруженных систем необходимы сетевые карты с высокой пропускной способностью.

Операционная система. Часто используются Linux (например, Ubuntu, CentOS) из-за стабильности и производительности, а также Windows Server для некоторых специфических задач и приложений.

Масштабируемость. Возможность увеличивать ресурсы сервера (вертикальное масштабирование) или добавлять новые серверы в кластер (горизонтальное масштабирование) для обработки растущих объёмов данных и нагрузки.

Отказоустойчивость. Механизмы репликации данных, резервного копирования и восстановления после сбоев.

Тип базы данных. Реляционные (SQL) и нереляционные (NoSQL) базы данных имеют разные требования к ресурсам и архитектуре. SQL-СУБД (MySQL, PostgreSQL, Microsoft SQL Server) хорошо подходят для структурированных данных и транзакционных операций, а NoSQL (MongoDB, Cassandra, Redis) — для работы с большими объёмами неструктурированных данных и обеспечивают высокую масштабируемость.

Количество одновременных подключений. Чем больше пользователей одновременно работают с сервером, тем больше ресурсов требуется для обработки их запросов.

Частота и тип запросов. Сложные запросы, требующие обработки больших объёмов данных, создают большую нагрузку на сервер, чем простые запросы.

Механизмы доступа к базам данных

Механизмы доступа к данным определяют, как клиентские приложения взаимодействуют с базой данных. Они могут быть реализованы через API СУБД или универсальные механизмы.

API СУБД. Большинство СУБД предоставляют специальные библиотеки с API для доступа к данным. В случае серверных СУБД API инициирует передачу запросов на сервер и получение результатов.

Универсальные механизмы доступа к данным. Реализованы в виде библиотек и драйверов, которые позволяют работать с разными СУБД через единый интерфейс. Примеры:

ODBC (Open Database Connectivity). Стандартный интерфейс для доступа к реляционным данным. Требует ODBC-драйвер для конкретной СУБД.

OLE DB. Программный интерфейс для доступа к различным источникам данных (реляционным и нереляционным). Использует провайдеры данных, которые реализуют взаимодействие с конкретной СУБД.

ADO (ActiveX Data Objects). Набор библиотек от Microsoft, использующий

OLE DB. Позволяет работать с данными через объекты.

BDE (Borland Database Engine). Механизм доступа к данным от Borland, поддерживающий работу с различными БД. Отличается трудоёмкостью развёртывания.

dbExpress. Технология от Borland для однонаправленного доступа к БД с высокой скоростью работы. Использует легковесные драйверы.

Клиент-серверная архитектура. В этом случае клиентское приложение отправляет запросы на сервер баз данных, а сервер обрабатывает их и возвращает результаты. Клиент может использовать API СУБД или универсальные механизмы доступа.

Прямые вызовы функций клиентского API. Приложение напрямую использует функции клиентской библиотеки СУБД. Это обеспечивает высокую производительность, но привязывает приложение к конкретной СУБД.

Использование ORM (Object-Relational Mapping). Технологии ORM (например, Hibernate, Entity Framework) позволяют работать с данными через объекты, автоматически преобразуя их в SQL-запросы.

Выбор механизма доступа зависит от требований к проекту, типа СУБД, необходимости интеграции с разными системами и других факторов. Универсальные механизмы упрощают смену СУБД, но могут снижать производительность и не поддерживать всю специфику конкретной системы.

Задание 2.5. Рассмотрение вопросов, связанных с аппаратным обеспечением и развертыванием серверов баз данных.

Теоретическая часть: Аппаратное обеспечение и развёртывание серверов баз данных — ключевые аспекты, влияющие на производительность, надёжность и масштабируемость системы. Рассмотрим основные компоненты и процессы.

Аппаратное обеспечение

Требования к аппаратному обеспечению зависят от типа СУБД, объёма данных, нагрузки и других факторов. Общие рекомендации включают:

Процессор (CPU). Количество ядер и тактовая частота должны соответствовать ожидаемой нагрузке. Для сложных запросов и высокой нагрузки требуются многоядерные процессоры с высокой тактовой частотой. Например, для SQL Server рекомендуется использовать процессоры Intel Xeon Scalable. При оценке

нагрузки можно ориентироваться на правило: от 10 до 30 соединений на одно ядро процессора (в зависимости от сложности запросов).

Оперативная память (RAM). Достаточный объём ОЗУ позволяет кэшировать данные и уменьшить нагрузку на дисковую подсистему. Идеальным считается вариант, когда вся база данных помещается в оперативную память. Для промышленных систем рекомендуется использовать память с коррекцией ошибок (ECC RAM).

Дисковая подсистема. SSD-накопители обеспечивают более высокую скорость чтения и записи по сравнению с HDD. Для баз данных и бэкапов рекомендуется использовать RAID-массивы:

RAID 10 — для рабочих данных (минимум 4 диска);

RAID 1 — для SSD-дисков (зеркальное копирование для повышения надёжности). Важно оставлять до 30% свободного места на SSD-дисках из-за повышенного износа.

Сетевые карты. Пропускная способность должна соответствовать нагрузке. Рекомендуется использовать сетевые карты с высокой пропускной способностью, например 1 Гбит/с или выше.

Блоки питания. Желательно использовать дублированные блоки питания с возможностью горячей замены.

Операционная система. Часто используются Linux (например, Ubuntu, CentOS) и Windows Server.

Развёртывание серверов баз данных

Развёртывание включает несколько этапов и аспектов:

Выбор конфигурации. Можно развернуть базу данных на одном сервере или распределить компоненты (например, базу данных и приложения) по разным серверам для повышения надёжности. Для высоконагруженных систем может потребоваться кластерная конфигурация с балансировкой нагрузки.

Настройка параметров. После установки СУБД необходимо сконфигурировать параметры производительности (например, размер буфера, параметры кэширования), безопасности (аутентификация, шифрование) и резервного копирования. Например, в PostgreSQL важно правильно настроить параметр shared_buffers и механизм автовакуумирования.

Создание пользователей и ролей. Определяются учётные записи с соответствующими правами доступа. Рекомендуется придерживаться принципа минимальных привилегий.

Тестирование. Перед вводом в эксплуатацию система тестируется под нагрузкой. Проверяется производительность, надёжность резервного копирования и восстановления, отказоустойчивость. Для нагрузочного тестирования можно использовать инструменты вроде pgbench для PostgreSQL.

Резервное копирование и восстановление. Разрабатывается стратегия резервного копирования (например, еженедельные полные бэкапы плюс непрерывное архивирование WAL-логов в PostgreSQL) и регулярно тестируется процесс восстановления.

Мониторинг. Настраиваются системы мониторинга для отслеживания ключевых метрик (загрузка CPU, использование памяти, дисковый ввод-вывод, количество активных соединений и т. д.). Это позволяет своевременно выявлять проблемы.

Безопасность. Настраиваются файрволы, ограничиваются доступы с недоверенных IP-адресов, используются SSL/TLS для защищённых соединений.

Дополнительные аспекты

Отказоустойчивость. Для повышения надёжности можно развернуть резервный сервер базы данных с синхронизацией через репликацию. Возможны многоуровневые структуры: основной сервер, горячий и холодный резерв.

Масштабируемость. При росте нагрузки система должна допускать вертикальное (увеличение ресурсов на существующем сервере) или горизонтальное (добавление серверов в кластер) масштабирование.

Синхронизация времени. Если база данных развёрнута на нескольких серверах, необходимо настроить синхронизацию времени между ними.

Обновление ПО. Регулярное обновление СУБД и операционной системы важно для устранения уязвимостей и повышения производительности.

При планировании развёртывания рекомендуется обращаться к документации производителя СУБД и проводить нагрузочное тестирование для точной оценки требований к аппаратному обеспечению.

Задание 2.6. Рассмотрение состава и схемы банка данных

Теоретическая часть: Банк данных — это сложная человеко-машинная система, которая включает в себя взаимосвязанные компоненты для централизованного хранения, обработки и многоцелевого использования данных. Он обеспечивает доступ к информации для множества пользователей и приложений, поддерживая целостность, безопасность и актуальность данных.

Состав банка данных

В структуру банка данных входят следующие компоненты:

База данных (БД). Ядро системы, представляющее собой совокупность взаимосвязанных данных из определённой предметной области, организованных специальным образом и хранимых во внешней памяти (файлах). Данные имеют минимальную избыточность, что позволяет их многоцелевое использование.

Система управления базами данных (СУБД). Программное обеспечение, которое управляет базой данных: создаёт, модифицирует, извлекает данные, обеспечивает целостность, контроль доступа и другие функции. СУБД служит посредником между пользователем и БД, скрывая внутреннюю сложность системы.

Вычислительная система. Включает технические средства (серверы, хранилища данных, сетевое оборудование) и операционную систему, обеспечивающие работу БД и СУБД.

Словарь данных. Централизованное хранилище сведений об объектах БД, их элементах, взаимосвязях, источниках, форматах представления, а также о группах элементов данных и базах данных. Совокупность базы данных и словаря данных составляет информационную базу.

Администратор базы данных (АБД). Группа специалистов, отвечающая за создание, функционирование и развитие банка данных. В их обязанности входит анализ предметной области, проектирование структуры БД, обеспечение целостности данных, защита от несанкционированного доступа, восстановление БД после сбоев и другие задачи.

Обслуживающий персонал. Специалисты, которые поддерживают технические и программные средства в работоспособном состоянии: проводят профилактические, регламентные и восстановительные работы.

Организационно-методические средства. Инструкции, методические и

регламентирующие материалы для пользователей разных категорий.

Также в состав банка данных могут входить прикладные программы для доступа к данным и управления ими.

Схема банка данных

Схема — это концептуальный проект всей БД, который определяет структуру хранения данных, связи между ними и правила обработки. Она включает несколько уровней абстракции:

Концептуальная схема. Описывает все сущности, атрибуты и связи между ними с указанием ограничений для поддержки целостности данных. Это высокоуровневое представление структуры БД, независимое от конкретной модели данных или СУБД.

Внутренняя схема. Полное описание внутренней модели данных, включающее определения хранимых записей, методов представления, полей данных, индексов и схем хеширования. Это низкоуровневое представление, зависящее от конкретной СУБД и способа хранения данных.

Внешние схемы (подсхемы). Соответствуют разным представлениям данных для конкретных пользователей или приложений. Позволяют разным группам пользователей видеть данные в удобном для них виде.

При проектировании БД сначала разрабатывается концептуальная модель, затем строится внутренняя модель (определяется структура хранения), а потом — внешняя модель для пользователей.

Дополнительные аспекты

Модели данных. Данные в БД могут организовываться по различным моделям: реляционной, иерархической, сетевой, объектно-ориентированной и др. Реляционная модель, например, использует таблицы для представления данных и связей между ними.

Метаданные. Это данные о данных, которые хранятся в словаре данных. Они включают информацию о структуре БД, пользователях, правах доступа, источниках данных и т. д..

Безопасность и контроль доступа. СУБД и организационно-методические средства обеспечивают защиту данных от несанкционированного доступа, а также разграничение прав пользователей.

Резервное копирование и восстановление. Механизмы для создания резервных копий данных и их восстановления в случае сбоев.

Банк данных должен соответствовать требованиям к надёжности, масштабируемости, производительности, удобству использования и другим параметрам. Его проектирование и эксплуатация требуют координации технических, программных и организационных ресурсов.

Задание 2.7. Разработка технических требований к серверу баз данных, применительно к разрабатываемому заданию.

Теоретическая часть: Разработка технических требований к серверу баз данных — комплексный процесс, который зависит от множества факторов: типа СУБД, объёма данных, нагрузки, требований к производительности, надёжности, масштабируемости и бюджета. Эти требования определяют конфигурацию аппаратного и программного обеспечения, а также параметры сетевой инфраструктуры.

Основные компоненты технических требований

Процессор (CPU). Количество ядер и тактовая частота должны соответствовать ожидаемой нагрузке. Для сложных запросов и высокой нагрузки требуются многоядерные процессоры с высокой тактовой частотой. Например, для проектов с большим количеством пользователей или интенсивными операциями может потребоваться 8–16 ядерных процессоров или больше. При оценке нагрузки можно ориентироваться на правило: от 10 до 30 соединений на одно ядро процессора (в зависимости от сложности запросов).

Оперативная память (RAM). Достаточный объём ОЗУ позволяет кэшировать данные и уменьшить нагрузку на дисковую подсистему. Идеальным считается вариант, когда вся база данных помещается в оперативную память. Для промышленных систем рекомендуется использовать память с коррекцией ошибок (ECC RAM). Объём памяти зависит от размера базы данных и нагрузки: например, для SQL Server минимальный объём для выпусков, кроме Express, — 1 ГБ, но для оптимальной производительности рекомендуется не менее 4 ГБ, и объём должен увеличиваться по мере роста базы данных.

Дисковая подсистема. Для баз данных предпочтительны SSD-накопители из-за высокой скорости чтения и записи по сравнению с HDD. Рекомендуется

использовать RAID-массивы:

RAID 10 — для рабочих данных (минимум 4 диска);

RAID 1 — для SSD-дисков (зеркальное копирование для повышения надёжности). Важно оставлять до 30% свободного места на SSD-дисках из-за повышенного износа. Для высоконагруженных систем могут использоваться NVMe-накопители с высокой скоростью чтения/записи.

Сетевая инфраструктура. Пропускная способность сетевых интерфейсов должна соответствовать нагрузке. Рекомендуется использовать сетевые карты с высокой пропускной способностью, например 1 Гбит/с или выше. Для отказоустойчивых систем часто требуется несколько независимых сетей: публичная для взаимодействия с клиентами и частная для служебного трафика между узлами кластера.

Блоки питания. Желательно использовать дублированные блоки питания с возможностью горячей замены для повышения надёжности.

Система охлаждения. Эффективная система охлаждения необходима для предотвращения перегрева компонентов.

Операционная система. Часто используются Linux (например, Ubuntu Server, CentOS) и Windows Server. Выбор зависит от требований к СУБД и корпоративной политики.

Резервирование и отказоустойчивость. Могут включать дублирование компонентов, использование кластерных конфигураций, репликации данных и систем резервного копирования.

Дополнительные параметры

IOPS (операций ввода-вывода в секунду). Критично для дисковой подсистемы. Например, для SQL Server и PostgreSQL могут требоваться минимальные значения IOPS для SSD-накопителей.

Резервное копирование и восстановление. Необходимо определить требования к частоте резервного копирования, хранению копий, методам восстановления данных.

Безопасность. Шифрование данных, контроль доступа, защита от вредоносного ПО.

Масштабируемость. Требования к возможности увеличения мощности

системы в будущем (добавление ядер, памяти, дисков).

Мониторинг и управление. Инструменты для отслеживания состояния сервера и СУБД, логирования событий.

Примеры расчётов

В некоторых источниках приводятся формулы для приблизительного расчёта требований на основе количества пользователей:

Процессор: количество одновременно работающих пользователей / 100 ядер (с тактовой частотой 3,2 ГГц и выше).

Оперативная память: количество одновременно работающих пользователей / 50.

Однако такие расчёты являются упрощёнными и требуют корректировки с учётом специфики проекта, типа СУБД, сложности запросов и других факторов.

Рекомендации

Обратиться к разработчикам СУБД. Они предоставляют официальные рекомендации по аппаратным требованиям для своих продуктов.

Провести нагрузочное тестирование. Это поможет уточнить требования к производительности и масштабируемости.

Учесть будущее развитие системы. Заложить запас мощности для роста объёма данных и нагрузки.

Рассмотреть виртуализацию или контейнеризацию, если это допустимо для проекта. Это может упростить управление и масштабирование.

Обратиться к документации и лучшим практикам для конкретных СУБД (например, требования Microsoft для SQL Server или PostgreSQL).

Технические требования должны быть документированы и регулярно пересматриваться с учётом изменений в системе и бизнес-требованиях.

Практическая часть: Разработать типовую конфигурацию сервера БД.

Задание 2.8. Разработка требований к корпоративной сети, применительно к разрабатываемому заданию.

Теоретическая часть: Разработка требований к корпоративной сети — комплексный процесс, который включает анализ бизнес-задач, определение технических параметров, учёт масштабируемости, безопасности, производительности и управляемости. Корпоративная сеть должна обеспечивать

эффективное взаимодействие подразделений, доступ к ресурсам, поддержку бизнес-приложений и защиту данных.

Основные принципы построения корпоративных сетей

Мультисервисность. Передача всех типов трафика (голос, видео, данные) по единым каналам связи.

Открытые стандарты и интерфейсы. Обеспечивают масштабируемость сети и возможность интеграции с другими системами.

Коммутационная технология. Использование сетей с коммутацией пакетов для эффективного использования каналов связи и минимизации затрат.

Модульность. Разделение сети на отдельные модули (например, модуль внешних сервисов, модуль глобальной сети WAN, модуль локальной вычислительной сети ЛВС) для предсказуемого развития и упрощения поиска неисправностей.

Ключевые требования к корпоративной сети

Параметр Описание

Расширяемость Возможность простой интеграции новых компонентов: пользователей, компьютеров, приложений, служб.

Масштабируемость Возможность увеличения количества узлов, протяжённости связей и производительности сетевого оборудования.

Производительность Обеспечение требуемых значений параметров: время реакции, скорость передачи данных, задержка передачи и вариация задержки передачи сетевых узлов и каналов связи.

Управляемость Централизованное управление, мониторинг состояния сети и планирование её развития.

Надёжность Безотказная работа узлов сети и каналов связи, сохранность, согласованность и доставка данных без искажений узлу назначения.

Безопасность Защита данных от несанкционированного доступа, вирусов, хакерских атак и других угроз.

Компоненты, которые учитываются при разработке требований

Топология сети. Например, «звезда», «шина», «кольцо» для LAN или Remote Access, Intranet, Extranet, Client-Server для VPN.

Типы сетей. Локальная сеть (LAN) для объединения устройств в пределах

одного офиса, глобальная сеть (WAN) для связи филиалов, виртуальная частная сеть (VPN) для защищённого удалённого доступа.

Сетевое оборудование. Коммутаторы (Switches), маршрутизаторы (Routers), точки доступа (Access Points), межсетевые экраны (Firewalls).

Серверы. Файловые, терминальные, почтовые, базы данных, веб-серверы, серверы резервного копирования и другие.

Кабельная инфраструктура. Структурированные кабельные системы для минимизации рисков отказов.

Протоколы и технологии. TCP/IP, DHCP, DNS, VLAN, VPN (IPsec, SSL, GRE), OSPF, BGP и другие.

Безопасность. Межсетевые экраны, антивирусные программы, системы обнаружения и предотвращения вторжений (IDS/IPS), шифрование данных, аутентификация пользователей.

Дополнительные аспекты

Резервирование и отказоустойчивость. Использование дублированных компонентов, кластерных конфигураций, балансировки нагрузки.

Сегментация сети. Разделение сети на логические или физические сегменты для оптимизации трафика и повышения безопасности.

Управление доступом. Разграничение доступа по ролям, использование паролей, смарт-карт, биометрии.

Мониторинг и журналирование. Сбор данных о трафике, активности пользователей, состоянии оборудования для выявления аномалий и инцидентов.

Резервное копирование и восстановление. Регулярное создание копий данных и планирование действий при сбоях.

Соответствие стандартам и нормативам. Например, ГОСТ Р ИСО/МЭК 27033-1-2011, ФЗ о защите персональных данных (№152-ФЗ) и другие.

Этапы разработки требований

Анализ требований. Определение проблем и деловых целей предприятия, формулировка задач проектирования.

Сбор информации о текущей и планируемой сети. Изучение архитектуры, приложений, услуг, видов соединений и других характеристик.

Оценка рисков. Идентификация потенциальных угроз и уязвимостей.

Формулировка технических параметров. Определение требований к пропускной способности, задержке, количеству узлов, уровню безопасности и другим параметрам.

Документирование требований. Создание технического задания, спецификаций, политик безопасности и других документов.

При разработке требований важно учитывать не только текущие потребности организации, но и перспективы её развития, а также бюджет и доступные технологии. Для сложных проектов рекомендуется привлекать опытных сетевых интеграторов и использовать отраслевые стандарты и лучшие практики.

Практическая часть: Разработать типовые требования к вашей корпоративной сети.

Задание 2.9. Решение вопросов, связанных с конфигурированием сети, применительно к разрабатываемому заданию.

Теоретическая часть: Конфигурирование сети — это процесс настройки параметров сетевого оборудования и программного обеспечения для обеспечения корректной работы, безопасности, производительности и масштабируемости сети. Он включает настройку IP-адресов, масок подсети, шлюзов, DNS-серверов, параметров безопасности, а также конфигурацию активного сетевого оборудования (маршрутизаторов, коммутаторов, точек доступа).

Основные этапы конфигурирования сети

Планирование и анализ требований. Перед началом настройки необходимо определить цели сети, её масштаб, количество устройств, требования к производительности, безопасности и масштабируемости. Создаётся сетевой план, учитывающий текущие и будущие потребности.

Выбор оборудования. Подбираются маршрутизаторы, коммутаторы, точки доступа и другое оборудование с учётом типа сети (домашняя, офисная, корпоративная), скорости передачи данных, необходимости поддержки VLAN, VPN, PoE и других функций.

Физическая установка оборудования. Маршрутизаторы и коммутаторы обычно размещают в центральной части сети для равномерного распределения нагрузки. Важно правильно проложить кабели, минимизируя их длину и избегая помех.

Настройка IP-адресов и подсетей. Каждому устройству присваивается уникальный IP-адрес. Определяется маска подсети, которая разделяет сетевую и хостовую части адреса. Настраивается шлюз по умолчанию — адрес маршрутизатора для доступа к внешним сетям.

Настройка DNS-серверов. DNS-серверы преобразуют доменные имена в IP-адреса. В настройках сети указывается адрес локального или внешнего DNS-сервера.

Конфигурация активного сетевого оборудования:

Маршрутизаторы. Настраиваются таблицы маршрутизации (статические или динамические), параметры VPN, NAT, правила фильтрации трафика.

Коммутаторы. Настраиваются VLAN для сегментации сети, QoS (качество обслуживания) для приоритизации трафика, агрегация каналов для увеличения пропускной способности, параметры PoE.

Точки доступа. Настраиваются параметры беспроводной сети: каналы, мощность сигнала, методы шифрования (WPA3, WPA2), интеграция с контроллерами.

Настройка безопасности:

активация брандмауэра для контроля трафика;

настройка правил доступа для предотвращения несанкционированных подключений;

внедрение систем обнаружения и предотвращения вторжений (IDS/IPS);

настройка VPN для защищённого удалённого доступа;

активация шифрования трафика, аутентификации пользователей.

Настройка служб DHCP. DHCP-сервер автоматически раздаёт IP-адреса, маски подсети, адреса шлюза и DNS-серверов подключённым устройствам.

Тестирование и мониторинг. После настройки проводится проверка работоспособности сети с помощью команд ping, tracert и других инструментов. Настраивается мониторинг состояния сети, трафика и производительности оборудования с помощью специализированных систем (например, SolarWinds, PRTG Network Monitor).

Ключевые параметры конфигурации

Параметр Описание

IP-адрес Уникальный адрес устройства в сети.

Маска подсети Определяет размер и структуру сети, разделяет сетевую и хостовую части IP-адреса.

Шлюз по умолчанию Адрес маршрутизатора, через который устройство получает доступ к внешним сетям.

DNS-серверПреобразует доменные имена в IP-адреса.

VLAN Виртуальные локальные сети для логического разделения трафика между группами устройств.

QoS Настройка приоритетов трафика для критически важных приложений.

БрандмауэрКонтролирует сетевой трафик, блокирует несанкционированный доступ.

Распространённые ошибки при конфигурировании

конфликты IP-адресов;

неправильная настройка масок подсети, приводящая к проблемам с маршрутизацией;

отключённый или неправильно настроенный DHCP, из-за чего устройства не получают IP-адреса автоматически;

недостаточные меры безопасности, оставляющие сеть уязвимой для атак;

не обновлённая прошивка оборудования, что может привести к ошибкам и уязвимостям.

Рекомендации

Регулярно обновляйте прошивку сетевого оборудования и ПО для устранения уязвимостей.

Используйте централизованное управление сетью (например, через SNMP или веб-интерфейсы) для удобства мониторинга и настройки.

Разрабатывайте и внедряйте политику паролей и доступа к сетевому оборудованию.

Проводите аудит безопасности и тестирование на проникновение для выявления слабых мест.

Конфигурирование сети — сложный и ответственный процесс, требующий знаний сетевых технологий, протоколов и опыта работы с оборудованием. Для сложных задач рекомендуется привлекать квалифицированных специалистов.

Практическая часть: Опишите конфигурацию вашей корпоративной сети.

Задание 2.10. Разработка хранимых процедур, применительно к создаваемой учебной базе данных.

Практическая часть: Разработать не менее 3 хранимых процедур в вашей БД.

Задание 2.11. Разработка триггеров, применительно к создаваемой учебной базе данных.

Практическая часть: Разработать не менее 3 триггеров в вашей БД.

Тема 3. Администрирование баз данных и серверов.

В современных условиях развития информационных технологий эффективное администрирование баз данных и серверов становится критически важным фактором успешного функционирования любой организации. Администрирование информационных систем представляет собой комплекс мероприятий по управлению, поддержке и развитию инфраструктуры хранения и обработки данных.

Администрирование баз данных — это процесс управления, настройки и поддержки баз данных, включающий в себя:

Обеспечение безопасности данных.

Оптимизацию производительности.

Резервное копирование.

Мониторинг состояния системы.

Управление доступом пользователей.

Администрирование серверов охватывает:

Настройку серверного оборудования.

Конфигурацию программного обеспечения.

Обеспечение отказоустойчивости.

Масштабирование ресурсов.

Техническую поддержку.

Основные цели:

Обеспечение бесперебойной работы информационных систем.

Защита данных от несанкционированного доступа.

Оптимизация производительности системы.

Своевременное резервное копирование.

Масштабирование ресурсов под растущие потребности.

Ключевые задачи:

Установка и настройка программного обеспечения.

Мониторинг производительности системы.

Обеспечение безопасности данных.

Управление пользовательскими правами.

Восстановление данных при сбоях.

Оптимизация работы системы.

Задание 3.1. Рассмотрение вопросов, связанных с технологией установки и настройка сервера MySQL в операционной системе Windows. Клиентские настойки, протоколирование, безопасность.

Практическая часть: Создать сервер MySQL на ОС Windows в виртуальной машине Oracle VM.

Задание 3.2. Рассмотрение вопросов, связанных с технологией установки и настройка сервера MySQL в операционных системах Linux.

Практическая часть: Создать сервер MySQL на ОС RedOS в виртуальной машине Oracle VM.

Задание 3.3. Рассмотрение вопросов, связанных с удаленным администрированием.

Практическая часть: Установить необходимое ПО, реализовать удаленное администрирование.

Задание 3.4. Рассмотрение вопросов, связанных с аудитом базы данных, аудиторским журналом. Установка опций, включение и отключение аудита. Очистка и уменьшение размеров журнала.

Теоретическая часть: Определение и назначение аудита

Аудит базы данных — это контролируемая запись событий, происходящих в системе управления базами данных (СУБД). Основная цель аудита — обеспечение безопасности, соответствия нормативным требованиям и контроль действий пользователей.

Основные задачи аудита:

Мониторинг доступа к данным.

Отслеживание изменений в базе данных.

Регистрация административных действий.

Выявление нарушений безопасности.

Создание и анализ отчетов.

Хранение данных аудита.

Система аудита включает следующие основные элементы:

Аудируемые события — действия, подлежащие отслеживанию.

Механизм аудита — компонент, перехватывающий и записывающий события.

Журнал аудита — хранилище данных об отслеживаемых событиях.

Инструменты анализа — средства для обработки и анализа данных аудита.

Встроенный аудит:

Использование встроенных возможностей СУБД.

Простая настройка и интеграция.

Ограниченная функциональность.

Триггерный аудит:

Гибкая настройка правил отслеживания.

Возможность детального контроля.

Возможное влияние на производительность.

Сторонние инструменты.

Расширенная функциональность.

Минимальное влияние на производительность.

Интеграция с системами безопасности.

Журнал аудита должен соответствовать следующим требованиям:

Безопасность — защита от несанкционированного доступа.

Надежность — гарантия сохранности данных.

Емкость — достаточный объем для хранения.

Производительность — минимальное влияние на работу системы.

Поисковые возможности — эффективный поиск по данным.

Настройка аудита включает:

Определение объектов аудита.

Установка политик аудита.

Настройка уровня детализации.

Определение периода хранения данных.

Настройка автоматического очищения.

Практические аспекты.

Важные рекомендации по работе с аудитом:

Регулярный анализ журналов.

Оптимизация настроек под конкретные задачи.

Контроль влияния на производительность.

Планирование хранения данных.

Разработка процедур очистки.

Управление размером журнала.

Методы контроля размера журнала аудита:

Регулярная очистка устаревших записей.

Архивация данных.

Настройка политик хранения.

Оптимизация параметров аудита.

Мониторинг роста журнала.

Безопасность аудита.

Ключевые аспекты безопасности:

Защита от несанкционированного доступа.

Шифрование данных аудита.

Контроль целостности записей.

Разграничение прав доступа.

Регулярное резервное копирование.

Эффективное управление аудитом требует баланса между полнотой отслеживания событий и влиянием на производительность системы. Необходимо регулярно анализировать настройки аудита и корректировать их в соответствии с изменяющимися требованиями безопасности и бизнес-потребностями.

Пример настройки аудита для Windows Windows

Создание аудита SQL Server:

Настройка через SQL Server Management Studio:

sql

```
-- Создание аудита сервера
CREATE SERVER AUDIT HIPAA_Audit
TO FILE ( FILEPATH ='E:\SQLAudit\' );
-- Создание спецификации аудита
CREATE SERVER AUDIT SPECIFICATION HIPAA_Audit_Specification
FOR SERVER AUDIT HIPAA_Audit
ADD (FAILED_LOGIN_GROUP);
-- Включение аудита
ALTER SERVER AUDIT HIPAA_Audit WITH (STATE = ON);
```

Основные параметры аудита:

Путь к файлу — указывается папка для хранения логов

Задержка очереди — время в миллисекундах до принудительной обработки

Действие при сбое — продолжение работы или остановка сервера

Управление журналом:

Просмотр через SSMS

Очистка через команды SQL

Настройка ротации файлов

Linux (PostgreSQL)

Настройка аудита через pgAudit:

Базовая конфигурация:

sql

```
-- Добавление расширения
CREATE EXTENSION pgaudit;
-- Настройка параметров
SET pgaudit.log = 'read, write, ddl';
SET pgaudit.log_relation = on;
SET pgaudit.log_level = 'log';
```

Конфигурация аудита объектов:

sql

```
-- Настройка роли аудита
SET pgaudit.role = 'auditor';
```

-- Предоставление прав

GRANT SELECT, INSERT, UPDATE, DELETE ON table_name TO auditor;

Общие настройки для обеих систем

Управление размером журнала:

Настройка максимального размера файлов

Автоматическое архивирование

Ротация логов

Очистка устаревших записей

Мониторинг аудита:

Проверка состояния аудита

Анализ производительности

Отслеживание ошибок

Включение и отключение аудита

Windows:

sql

-- Включение

ALTER SERVER AUDIT HIPAA_Audit WITH (STATE = ON);

-- Отключение

ALTER SERVER AUDIT HIPAA_Audit WITH (STATE = OFF);

Linux:

sql

-- Включение

ALTER SYSTEM SET pgaudit.log = 'all';

SELECT pg_reload_conf();

-- Отключение

ALTER SYSTEM SET pgaudit.log = 'none';

SELECT pg_reload_conf();

Очистка журнала

Windows:

Через SSMS — очистка через графический интерфейс

Через T-SQL:

sql

ALTER SERVER AUDIT HIPAA_Audit WITH (STATE = OFF);

ALTER SERVER AUDIT HIPAA_Audit WITH (STATE = ON);

Linux:

Ручная очистка файлов лога

Настройка автоматического удаления старых записей

Использование системных утилит для сжатия и архивации

Рекомендации по безопасности

Регулярно проверяйте права доступа к файлам аудита

Настройте резервное копирование журналов

Используйте шифрование для важных данных аудита

Ограничьте доступ к настройкам аудита только для администраторов

Практическая часть: Реализовать включение и отключение аудиторского журнала.

Задание 3.5. Рассмотрение вопросов, связанных с технологиями создания базы данных с применением языка SQL. Добавление, удаление данных и таблиц.

Практическая часть: Создать интерфейс для добавление, удаление данных и таблиц.

Задание 3.6. Рассмотрение вопросов, связанных с создание запросов, процедур и триггеров. Создание запросов и процедур на изменение структуры базы данных.

Практическая часть: Создать запрос или процедуру на изменение структуры вашей БД.

Задание 3.7. Рассмотрение вопросов, связанных с динамическим SQL и его операторами.

Теоретическая часть: Динамический SQL — это техника создания SQL-запросов во время выполнения программы, а не на этапе разработки. В отличие от статического SQL, где запросы фиксированы в коде, динамический SQL позволяет генерировать запросы на основе различных условий, пользовательского ввода, данных из других таблиц или логики приложения. Это обеспечивает гибкость и адаптивность запросов к изменяющимся условиям.

Преимущества динамического SQL

Гибкость. Возможность создавать запросы, адаптированные к конкретным потребностям, например, работать с различными таблицами, столбцами и условиями фильтрации.

Автоматизация. Упрощение создания сложных запросов, особенно при работе с большими объёмами данных или сложными условиями.

Создание мета-запросов. Возможность манипулировать структурой базы данных (создавать таблицы, добавлять столбцы, изменять схемы).

Динамическая фильтрация и сортировка. Возможность формировать условия WHERE или ORDER BY на лету в зависимости от пользовательского ввода или логики приложения.

Недостатки динамического SQL

Риск SQL-инъекций. При неправильном использовании динамический SQL уязвим для атак SQL-инъекциями, когда злоумышленник внедряет вредоносный код через пользовательский ввод.

Сложность отладки. Отладка динамических запросов может быть сложнее, чем отладка статических, так как код запроса формируется во время выполнения.

Снижение производительности. Создание и компиляция динамических запросов может занимать больше времени, чем выполнение статических запросов, так как требуется дополнительный этап синтаксического анализа.

Проблемы с кешированием планов выполнения. Каждый уникальный запрос заставляет СУБД строить новый план выполнения, что может негативно сказаться на производительности.

Операторы динамического SQL

Операторы динамического SQL зависят от конкретной СУБД, но в общем случае они позволяют формировать, подготавливать и выполнять SQL-запросы в виде строк.

В MySQL

Динамический SQL реализуется с помощью хранимых процедур и операторов:

PREPARE statement_name FROM query_string;. Подготавливает SQL-запрос, хранящийся в строковой переменной query_string, и присваивает ему имя statement_name. На этом этапе запрос не выполняется, а только анализируется и

компилируется.

EXECUTE statement_name USING @variable1, @variable2, ...;. Выполняет подготовленный SQL-запрос statement_name, передавая ему значения параметров, указанных в списке @variable1, @variable2, ...

DEALLOCATE PREPARE statement_name;. Освобождает ресурсы, связанные с подготовленным SQL-запросом statement_name.

В SQL Server

Для выполнения динамического SQL используются:

EXEC или EXECUTE. Команды для выполнения строки с SQL-запросом.

sp_executesql. Хранимая процедура, которая позволяет выполнять динамический SQL с параметрами. Имеет синтаксис: EXEC sp_executesql N'SQL query', N'@param1 data_type', @param1 = 'value1'. Префикс N указывает, что оператор SQL обрабатывается как строка в Юникоде.

В PostgreSQL

Основные инструменты:

EXECUTE. Выполняет строку как SQL-запрос.

FORMAT. Помогает безопасно собирать строку запроса. Например, %I экранирует идентификаторы (имена таблиц, столбцов), а %L — литералы.

RAISE NOTICE. Используется для вывода текста запроса перед выполнением, что полезно для отладки.

Пример использования динамического SQL в SQL Server

sql

```
DECLARE @customerStatus INT = 1;
DECLARE @statement VARCHAR(4000);
BEGIN
    SET @statement = 'SELECT id, customer_name FROM customer WHERE status = @customerStatus';
    EXECUTE sp_executesql @statement, N'@customerStatus INT', @customerStatus = @customerStatus;
END;
```

В этом примере переменная @customerStatus используется в динамическом запросе, который выполняется с помощью sp_executesql.

Пример использования динамического SQL в PostgreSQL

sql

```
CREATE OR REPLACE FUNCTION get_filtered_data(start_date DATE, end_date DATE, status TEXT)
RETURNS TABLE(id INT, created_at DATE, status TEXT) AS
$$
BEGIN    RETURN QUERY EXECUTE FORMAT(        'SELECT id, created_at, status FROM orders WHERE created_at BETWEEN %L AND %L AND status = %L', start_date, end_date, status    );END
$$
 LANGUAGE plpgsql;
```

Здесь функция FORMAT используется для безопасного построения запроса с параметрами.

Рекомендации по использованию динамического SQL

Используйте параметризованные запросы. Это поможет избежать SQL-инъекций. В SQL Server для этого применяется sp_executesql, в PostgreSQL — предложение USING с EXECUTE.

Экранируйте идентификаторы и литералы. В PostgreSQL для этого используются %I и %L в функции FORMAT.

Проверяйте вводимые данные. Всегда валидируйте пользовательский ввод перед использованием в динамическом SQL.

Логируйте запросы. Это упростит отладку и анализ проблем.

Избегайте конкатенации строк без необходимости. Склеивание строк может привести к ошибкам и уязвимостям.

Динамический SQL — мощный инструмент, но его использование требует осторожности и соблюдения лучших практик для обеспечения безопасности и производительности.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

Основная литература

1. Перлова О. Н. Соадминистрирование баз данных и серверов: учебное издание / Перлова О. Н., Ляпина О. П. - Москва : Академия, 2023. - 304 с. (Специальности среднего профессионального образования). - URL: https://academia-moscow.ru - Режим доступа: Электронная библиотека «Academia-moscow». - Текст : электронный.

2. Федорова, Г. Н. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности : Учебное пособие / Г. Н. Федорова. – Москва : КУРС : ИНФРА-М, 2022. – 336 с. (Среднее профессиональное образование). – ISBN 9785906818416. – Текст : непосредственный.

Дополнительная литература

1. Компьютерные сети : учебник для среднего профессионального образования по специальностям 09.02.06 "Сетевое и системное администрирование", 09.02.07 "Информационные системы и программирование" / В. В. Баринов, И. В. Баринов, А. В. Пролетарский, А. Н. Пылькин ; В. В. Баринов [и др.]. – 4-е изд., испр. и доп.. - Москва : Академия, 2021. – 192 с. – ISBN 9785446899258. – URL: https://academia-moscow.ru/catalogue/4831/551458/. – Текст : электронный.

2. Гохберг Г.С. Информационные технологии: ЭУМК: учебное издание / Гохберг Г.С., Зафиевский А.В., Короткин А.А. - Москва : Академия, 2024. - 0 с. (Специальности среднего профессионального образования). - URL: https://academia-moscow.ru - Режим доступа: Электронная библиотека «Academia-moscow». - Текст : электронный.

3. Казанский, А. А. Программирование на C# : учебное пособие для среднего профессионального образования / А. А. Казанский. — 3-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2025. — 181 с. — (Профессиональное образование). — ISBN 978-5-534-21380-5. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/569863.

4. Семакин И.Г. Основы алгоритмизации и программирования: ЭУМК: учебное издание / Семакин И.Г., Шестаков А. П. - Москва : Академия, 2025. - 0 с. (Специальности среднего профессионального образования). - URL: https://academia-

moscow.ru - Режим доступа: Электронная библиотека «Academia-moscow». - Текст : электронный

5. Куприянов, Д. В. Информационное обеспечение профессиональной деятельности : учебник и практикум для среднего профессионального образования / Д. В. Куприянов. — 3-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2025. — 236 с. — (Профессиональное образование). — ISBN 978-5-534-20826-9. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/558828.

6. Грекул, В. И. Проектирование информационных систем : учебник и практикум для среднего профессионального образования / В. И. Грекул, Н. Л. Коровкина, Г. А. Левочкина. — 2-е изд. — Москва : Издательство Юрайт, 2025. — 404 с. — (Профессиональное образование). — ISBN 978-5-534-19506-4. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/566739.

7. Проектирование информационных систем : учебник и практикум для среднего профессионального образования / Д. В. Чистов, П. П. Мельников, А. В. Золотарюк, Н. Б. Ничепорук. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2025. — 273 с. — (Профессиональное образование). — ISBN 978-5-534-20362-2. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/562355.

Приложение А. Бланк титульного листа

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение \

высшего образования

«КУЗБАССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ ИМЕНИ Т.Ф.ГОРБАЧЕВА»

Филиал КузГТУ в г.Белово

ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ

ПМ.07 «Соадминистрирование баз данных и серверов»

Выполнил:

Студент группы _____

/_____/_____

подпись

Руководитель:

/_____/_____

подпись

Оценка_____

«___»_____20___ г.

Белово

20__г.

ПРИЛОЖЕНИЕ Б. Бланк задания по УП

ЗАДАНИЕ

на учебную практику

по профессионального модуля **ПМ.07 «Соадминистрирование баз данных и серверов»**

студент _____

группы_____

специальности «09.02.07 Информационные системы и программирование»

Дата начала практики «___» _____ 20___г.

Дата окончания практики «___» _____ 20__г.

Дата сдачи практики «___» _____ 20__г.

Виды работ, обязательных для выполнения:

1. _____
2. _____
3. _____
4. _____

Задание выдал руководитель учебной практики от

филиала КузГТУ в г.Белово /_____/_____

подпись

ПРИЛОЖЕНИЕ В. Дневник по УП

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«КУЗБАССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ Т.Ф.ГОРБАЧЕВА»

Филиал КузГТУ в г.Белово

ДНЕВНИК УЧЕБНОЙ ПРАКТИКИ

профессионального модуля ПМ.07 «Соадминистрирование баз данных и серверов»

студент _____

группы_____

специальности «09.02.07 Информационные системы и программирование»

Период практики с «___» _____ 20__г. по «___» _____ 20__.

База практики_____

М.П.

Руководитель учебной практики от

филиала КузГТУ в г.Белово

/_____/_____

подпись

Закончил практику «___» _____ 20__.

Дата	Содержание работы	Оценка	Подпись

ПРИЛОЖЕНИЕ Г. Бланк аттестационного листа по УП

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение \

высшего образования

«КУЗБАССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ ИМЕНИ Т.Ф.ГОРБАЧЕВА»

Филиал КузГТУ в г.Белово

АТТЕСТАЦИОННЫЙ ЛИСТ

по учебной практике

по профессиональному модулю _____

(индекс и наименование профессионального модуля)

Обучающийся	
Институт/факультет	
Специальность	
(код специальности)	
Курс	Группа
Вид практики	
Способ прохождения практики	
Период прохождения практики с	по
Профильная организация	

(наименование, местонахождение)

Во время прохождения практики обучающимся были освоены следующие профессиональные и общие компетенции

Наименование компетенции	Оценка	
	Освоена	Не освоена

Руководитель учебной практики от

филиала КузГТУ в г.Белово

/_____/_____

подпись

ПРИЛОЖЕНИЕ Д. Бланк характеристики

на обучающегося в период прохождения УП

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение \

высшего образования

«КУЗБАССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ ИМЕНИ Т.Ф.ГОРБАЧЕВА»

Филиал КузГТУ в г.Белово

ХАРАКТЕРИСТИКА

на обучающегося по освоению общих и профессиональных компетенций

в период прохождения учебной практике

по профессиональному модулю_____

(индекс и наименование профессионального модуля)

Обучающийся

Институт/факультет

Специальность

(код специальности)

Курс	Группа
Вид практики	
Способ прохождения практики	
Период прохождения практики с	по
Профильная организация	

(наименование, местонахождение)

Виды и качество выполненных работ:

Виды работ	Критерии выполнения работ		
	Выполнены полностью	Выполнены с незначительной помощью	Выполнены с помощью наставника

Руководитель учебной практики от

филиала КузГТУ в г.Белово

/_____/_____

подпись

Составитель
Витвицкий Максим Николаевич

Методические указания по учебной практике УП.07.01
для студентов очной формы обучения
по направлению специальности
09.02.07 «Информационные системы и программирование»

Публикуется в авторской редакции