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

ПП.07.01 «Соадминистрирование баз данных и серверов»

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

Составитель: Витвицкий М.Н.
Рассмотрены и утверждены на заседании кафедры
Протокол № 6 от 14.02.2026 г.
Рекомендовано учебно-методической комиссией специальностей СПО в качестве электронного издания для использования в учебном процессе
Протокол № 6 от 17.02.2026 г.

Белово 2026

СОДЕРЖАНИЕ

ОРГАНИЗАЦИЯ ПРОИЗВОДСТВЕННОЙ 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.

Производственная практика проводится в организациях и учреждениях в отделах и подразделениях IT. Организацию и руководство практикой по профилю специальности (профессии) осуществляют руководители практики из числа профессорско-преподавательского состава филиала КузГТУ в г.Белово и руководитель практики от организации.

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

- разрабатывает содержание и планируемые результаты практики;
- осуществляет общее 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.

Руководитель практики из числа лиц, относящихся к сотрудникам IT-отдела в организации где проходит практика:

- согласовывает содержание и планируемые результаты практики;

- осуществляет руководство практикой в организации;

- организует 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;

- оценивает применение общих и профессиональных компетенций обучающегося, используемых им в ходе прохождения практики.

Обучающиеся в период прохождения практики обязаны:

- выполнять задания, предусмотренные программами практики;

- соблюдать действующие в организации для прохождения практики правила его 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 производственную 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 производственной практики, а также титульный лист отчета по производственной практике приведены в Приложении А-Д.

КОНТРОЛЬ РЕЗУЛЬТАТОВ ВЫПОЛНЕНИЯ ЗАДНИЙ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ.

Контроль результатов по производственной 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 применения студентом навыков и компетенций;
- умение обучающегося использовать теоретические знания при выполнении практических задач;
- полнота сформированных общих и профессиональных компетенций;
- обоснованность и четкость выполнения производственных задач;
- оформление необходимого материала в соответствии с требованиями.

МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ.

ПРОИЗВОДСТВЕННАЯ ПРАКТИКА 07.01 «Соадминистрирование баз данных и серверов».

Производственная практика — это метод обучения студентов в высших учебных заведениях, направленный на приобретение практических навыков и опыта работы в определенной отрасли. Она является важной составляющей образовательного процесса и представляет собой переход от теоретических знаний к их практическому применению.

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

Производственной практики является комплексное освоение обучающимися основного вида профессиональной деятельности «Соадминистрирование баз данных и серверов», закрепление теоретических знаний, полученных при изучении профессионального модуля ПМ.07 «Соадминистрирование баз данных и серверов» и на основе практического участия в деятельности организаций (предприятий) различных форм собственности.

Задачи производственной 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; получение практического опыта сбора, анализа, систематизации и подготовки контента данных для обработки отраслевой направленности средствами автоматизированных информационных систем предприятия; выполнение работ по вводу и представлению данных, формированию входящей и исходящей документации на рабочем месте практики.

Производственная практика осуществляется в форме:

1. Выполнения поставленных учебных и производственных задач

Задания по производственной практике

№ раздела (темы)	Вопросы, выносимые на изучение	Количество часов
Тема 1. Принципы построения и администрирования баз данных	Задание 1.1. Анализ предметной области и формирование требований к базе данных в соответствии с поставленной задачей.	6
	Задание 1.1. Разработка концептуальной модели базы данных в соответствии с поставленной задачей.	6
	Задание 1.3. Даталогическое проектирование базы данных. Определение всех информационных единиц и связей между ними, задание их имен и типов, а также некоторых количественных характеристик. преобразование исходной инфологической модели в модель данных, поддерживаемую конкретной СУБД, и проверка адекватности полученной даталогической модели отображаемой предметной области.	18
	Задание 1.4. Разработка сценариев работы с данными. Определение архитектуры системы.	12
Тема 2. Серверы баз данных	Задание 2.1. Разработка технических требований к серверу баз данных. Разработка требований к корпоративной сети.	6
	Задание 2.2. Разработка серверной и клиентских компонент базы данных.	12
	Задание 2.3. Установка и настройка сервера баз данных	6

Тема 3. Администрирование баз данных и серверов	Задание 3.1. Создание механизмов сервера для обслуживания базы данных. Установка и развёртывание системы.	8
	Задание 3.2. Работа с журналом аудита базы данных. Мониторинг нагрузки сервера.	6
Тема 4. Защита и сохранность информации баз данных	Задание 4.1. Настройка политики безопасности для созданной базы данных.	8
	Задание 4.2. Создание резервных копий базы данных. Восстановление базы данных. Мониторинг активности портов.	6
Тема 5. Сертификация информационных систем	Задание 5.1. Оформление требований и разработка технического задания по сертификации информационной системы (базы данных).	8
	Задание 5.2. Выбор сертификатов. Сроки их действия.	6
Промежуточная аттестация в форме: 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.

Принципы построения и администрирования баз данных — ключевые аспекты работы с системами хранения данных, которые включают проектирование структуры БД, обеспечение её надёжности, безопасности, производительности и поддержки. Эти знания критически важны для производственной практики, так как позволяют эффективно создавать, настраивать и обслуживать базы данных в реальных проектах.

Принципы построения баз данных

База данных (БД) — именованная совокупность данных, отражающая состояние объектов и их отношений в определённой предметной области. **Система управления базами данных (СУБД)** — совокупность языковых и программных средств для создания, наполнения, обновления и удаления БД.

К современным БД и СУБД предъявляются следующие основные требования:

- **Высокое быстродействие** (малое время отклика на запрос — промежуток от момента запроса до получения данных).
- **Простота обновления данных**.
- **Независимость данных** — возможность изменения логической и физической структуры БД без изменения представлений пользователей.
- **Совместное использование данных многими пользователями**.
- **Безопасность данных** — защита от преднамеренного или непреднамеренного нарушения секретности, искажения или разрушения.
- **Стандартизация построения и эксплуатации БД**.
- **Адекватность отображения данных соответствующей предметной области**.
- **Простой интерфейс пользователя**.

Важнейшими считаются первые два требования: повышение быстродействия часто требует упрощения структуры БД, что затрудняет обновление данных и может увеличивать их избыточность.

Целостность данных — устойчивость хранимых данных к разрушению и уничтожению, связанному с неисправностями технических средств, системными ошибками и ошибочными действиями пользователей. Она предполагает:

- отсутствие неточно введённых данных или двух одинаковых записей об одном и том же факте;
- защиту от ошибок при обновлении БД;
- невозможность удаления (или каскадное удаление) связанных данных разных таблиц;
- неискажение данных при работе в многопользовательском режиме и в распределённых БД;
- сохранность данных при сбоях техники (восстановление данных).

Целостность обеспечивается **триггерами целостности** — специальными приложениями-программами, работающими при определённых условиях.

Этапы проектирования баз данных

Проектирование БД включает три основных этапа:

1. **Концептуальное (инфологическое) проектирование**. Создание высокоуровневой модели данных на основе представлений пользователей о предметной области. Используются стандартные языковые средства, например ER-диаграммы (диаграммы «сущность-связь»). На этом этапе определяют объекты предметной области, их атрибуты, связи между ними и ограничения целостности. nsportal.ru +2

2. **Логическое (даталогическое) проектирование**. Преобразование концептуальной модели в структуру, совместимую с конкретной СУБД (например, реляционную модель). Определяются таблицы, поля, ключи, проводятся нормализация данных для устранения избыточности. nsportal.ru +1

3. **Физическое проектирование**. Выбор СУБД, определение реальных структур хранения данных на диске, параметров таблиц и индексов, методов доступа к данным. Учитываются требования к производительности и безопасности.

Нормализация — процесс разбиения таблиц на более мелкие для устранения избыточности и повышения целостности данных. Выделяют несколько нормальных форм (1NF, 2NF, 3NF и др.).

Администрирование баз данных

Администрирование БД — комплекс мероприятий по проектированию, разработке, поддержке и оптимизации баз данных организации. **Администратор баз данных (DBA)** отвечает за эффективность, надёжность и безопасность БД, предотвращение потери данных.

Ключевые задачи администрирования: it-aurora.ru +1

- **Проектирование и создание БД**. Разработка структуры таблиц, индексов и связей, соответствующей бизнес-требованиям.
- **Обеспечение доступности и целостности данных**. Настройка систем репликации, кластеризации и резервного копирования.
- **Управление производительностью**. Мониторинг, диагностика и оптимизация запросов, индексов и конфигурационных параметров.
- **Обеспечение безопасности**. Разграничение доступа, аудит действий пользователей, защита от внешних и внутренних угроз, шифрование данных.

- **Планирование мощностей**. Анализ трендов роста данных и нагрузки, прогнозирование потребностей в ресурсах.
- **Миграция и обновление СУБД**. Планирование и проведение обновлений ПО, миграция данных между системами.
- **Документирование**. Создание и поддержание технической документации по структуре БД, процедурам администрирования и политикам безопасности.

Инструменты и технологии

Для администрирования используются:

- интегрированные среды администрирования (графические интерфейсы для управления СУБД);
- системы мониторинга и диагностики (для отслеживания производительности и выявления узких мест);
- решения для резервного копирования и восстановления;
- средства оптимизации запросов (анализаторы SQL);
- инструменты для управления схемой БД (версионирование структуры данных);
- средства защиты данных (шифрование, маскирование, аудит доступа).

Практические аспекты для производственной практики

В рамках производственной практики обучающийся должен:

- получить практический опыт работы с объектами БД в конкретной СУБД, использования средств заполнения БД и стандартных методов защиты данных;
- научиться создавать объекты БД, управлять доступом к ним, работать с CASE-средствами проектирования, формировать и настраивать схему БД, разрабатывать прикладные программы с использованием SQL, создавать хранимые процедуры и триггеры;
- знать основные положения теории БД, принципы построения концептуальной, логической и физической модели данных, современные инструментальные средства разработки схемы БД, методы организации целостности данных, способы контроля доступа к данным и управления привилегиями, основные методы и средства защиты данных.

Примеры задач на практике:

- проектирование структуры БД для конкретной предметной области;
- настройка параметров СУБД для оптимизации производительности;
- реализация резервного копирования и плана восстановления данных;
- настройка прав доступа пользователей;
- анализ медленных запросов и их оптимизация;
- документирование процедур администрирования.

Таким образом, принципы построения и администрирования БД охватывают как теоретические основы проектирования, так и практические навыки работы с СУБД, обеспечения безопасности и оптимизации производительности, что является ключевым для успешной работы в сфере информационных технологий.

Задание 1.1. Анализ предметной области и формирование требований к базе данных в соответствии с поставленной задачей.

Анализ предметной области и формирование требований к базе данных (БД) — ключевые этапы проектирования информационных систем. Эти процессы включают изучение бизнес-процессов, определение сущностей, связей и атрибутов, а также учёт особенностей платформ (Windows и Linux). Практические рекомендации помогут систематизировать работу и избежать ошибок.

Анализ предметной области

1. **Изучение бизнес-процессов**. Определите, какие операции и данные задействованы в предметной области. Проведите интервью с экспертами, изучите документацию, регламенты и существующие системы. Например, для системы учёта заявок на ремонт нужно описать этапы обработки заявок, роли пользователей, типы оборудования и т. д..

2. **Идентификация сущностей и атрибутов**. Выявите объекты, которые будут храниться в БД (например, «Клиент», «Заказ», «Товар»). Для каждой сущности определите атрибуты (поля): название, тип данных, ограничения (NOT NULL, уникальные значения и т. д.).

3. **Определение связей между сущностями**. Установите типы связей: один ко многим (1:N), многие ко многим (N:M) и т. п. Например, связь между «Клиентом» и «Заказом» — один ко многим, если один клиент может сделать несколько заказов.

4. **Анализ бизнес-правил и ограничений**. Выявите правила, которые должны соблюдаться в системе. Например, ограничение на минимальную сумму заказа, требование уникальности email для пользователей, условия для выполнения определённых действий (например, подтверждение оплаты перед отправкой товара).

5. **Моделирование данных**. Используйте диаграммы «сущность-связь» (ERD) для визуализации модели данных. Это поможет наглядно представить структуру БД и упростить коммуникацию с командой. Можно применять нотации П. Чена или другие стандарты.

6. **Учёт масштабируемости и производительности**. Оцените, как система будет работать при росте объёма данных. Например, если ожидается большой поток данных, стоит предусмотреть партиционирование таблиц или использование распределённых СУБД.

Формирование требований к базе данных.

Критерий	Описание
Целостность данных	Требования к ограничениям (NOT NULL, уникальные ключи, внешние ключи), бизнес-правилам и ссылочной целостности.
Производительность	Требования к скорости выполнения запросов, времени отклика системы, оптимизации индексов и т. д.
Безопасность	Доступ к данным на уровне пользователей/ролей, шифрование чувствительных данных, аудит изменений.
Резервное копирование и восстановление	Частота бэкапов, требования к хранению копий, процедуры восстановления после сбоев.
Интеграция с другими системами	Требования к API, форматам данных для обмена с внешними системами.

Критерий	Описание
Масштабируемость	Возможность горизонтального или вертикального масштабирования при росте нагрузки.

Особенности для Windows и Linux.

Критерий	Windows	Linux
Инструменты разработки и администрирования	Графические интерфейсы (SQL Server Management Studio, PowerShell для SQL Server), интеграция с экосистемой Microsoft.	Командная строка, инструменты вроде `psql` для PostgreSQL, поддержка Docker и оркестраторов (Kubernetes, Docker Swarm).
СУБД	Часто используется Microsoft SQL Server, который тесно интегрирован с экосистемой Windows.	Широкая поддержка открытых СУБД (MySQL, PostgreSQL, MongoDB). Для некоторых задач могут применяться дистрибутивы с оптимизированными версиями СУБД (например, Postgres Pro для 1С).
Лицензирование	Лицензирование SQL Server основано на количестве ядер процессора.	Для открытых СУБД лицензирование обычно не требуется, но могут быть ограничения для коммерческих

Критерий	Windows	Linux
		дистрибутивов Linux (например, Red Hat Enterprise Linux).
Настройка и администрирование	Более простой интерфейс для новичков, меньше требований к навыкам работы с командной строкой.	Требует навыков работы с командной строкой, написания скриптов. Больше гибкости для тонкой настройки и автоматизации.
Совместимость с ПО	Лучше подходит для проектов на ASP.NET, интеграции с Microsoft 365, Exchange и т. д..	Оптимален для веб-приложений на PHP, открытых стандартов и технологий. Поддержка широкого спектра языков программирования.

Практические рекомендации

1. **Используйте шаблоны проектирования БД**. Например, шаблон «Звезда» для хранилищ данных или нормализацию для реляционных БД.
2. **Тестируйте на ранних этапах**. Создавайте прототипы и проводите нагрузочное тестирование, чтобы выявить узкие места.
3. **Учитывайте особенности СУБД**. Например, в PostgreSQL есть расширения для полнотекстового поиска и JSONB, а в SQL Server — инструменты для анализа планов запросов.
4. **Документируйте требования и модель данных**. Используйте спецификации, ER-диаграммы, словари данных. Это упростит поддержку и развитие системы.

5. **Для Linux-среды освойте инструменты автоматизации**. Например, Ansible или Terraform для развёртывания инфраструктуры, а также CI/CD-пайплайны для непрерывной интеграции и доставки.

6. **При миграции между платформами (например, с Windows на Linux) учитывайте различия в синтаксисе и поведении СУБД**. Например, в PostgreSQL и SQL Server могут отличаться функции работы с датами или строками.

7. **Для Windows используйте встроенные инструменты мониторинга** (Performance Monitor, SQL Server Profiler), а для Linux — `top`, `htop`, `vmstat` и т. п.

Анализ предметной области и формирование требований к БД — итеративный процесс, который требует учёта как бизнес-требований, так и технических особенностей платформы. Для Windows характерна более простая интеграция с экосистемой Microsoft, а Linux предлагает гибкость и оптимизацию для открытых технологий. Ключевое — чётко формулировать требования, использовать проверенные подходы и инструменты, а также регулярно тестировать систему на соответствие заданным параметрам.

Задание 1.2. Разработка концептуальной модели базы данных в соответствии с поставленной задачей.

Разработка концептуальной модели базы данных — это первый этап проектирования, который включает высокоуровневое описание предметной области: сущностей, их атрибутов и связей между ними без привязки к конкретной СУБД или технологиям реализации. Этот этап одинаково важен как для Windows, так и для Linux, хотя некоторые инструменты и подходы могут различаться.

Основные этапы разработки концептуальной модели

1. **Анализ предметной области и сбор требований**. Определите, какие данные должны храниться в базе, кто будет её использовать и какие задачи она должна решать. Изучите бизнес-процессы, документы, регламенты, проведите интервью с экспертами. Важно понять, какие запросы будут предъявляться к базе данных, и учесть требования всех категорий пользователей.

2. **Определение сущностей**. Сущности — это объекты реального мира, информация о которых должна храниться в базе данных (например, «Клиент», «Заказ», «Товар»). Каждая сущность должна иметь чёткое название (в

единственном числе) и текстовое описание. Избегайте технических наименований и расплывчатых формулировок.

3. **Определение атрибутов**. Атрибуты — это свойства сущностей (например, «Имя клиента», «Дата заказа», «Цена товара»). Для каждого атрибута укажите название, тип данных и ограничения (например, NOT NULL). Атрибуты должны иметь чёткое смысловое значение и именоваться в единственном числе.

4. **Определение связей между сущностями**. Проанализируйте, как сущности взаимодействуют друг с другом. Основные типы связей: «один к одному» (1:1), «один ко многим» (1:N) и «многие ко многим» (N:N). Например, связь между «Клиентом» и «Заказом» часто бывает 1:N, если один клиент может сделать несколько заказов.1

5. **Определение ограничений целостности**. Укажите правила, которые должны соблюдаться в базе данных (например, уникальность значений, обязательные поля, ограничения на диапазоны значений).

6. **Построение ER-диаграммы**. Визуализируйте модель с помощью диаграммы «сущность-связь» (Entity-Relationship, ER-диаграммы). Сущности изображаются прямоугольниками, атрибуты — овалами, связи — ромбами или линиями с указанием типа связи. Можно использовать нотации Чена, Мартина (Crow’s Foot) или IDEF1X.

7. **Обзор и уточнение модели**. Проверьте модель на предмет несоответствий, избыточности и недостающей информации. При необходимости внесите коррективы.

Рекомендации для Windows.

- **Инструменты**. В Windows часто используются графические инструменты для моделирования данных, например, Microsoft Visio, ERWin Data Modeler или PowerDesigner. Они позволяют создавать и редактировать ER-диаграммы с визуальным интерфейсом.
- **Интеграция с экосистемой Microsoft**. Если планируется использование SQL Server, можно применять инструменты, интегрированные с этой СУБД, например, SQL Server Management Studio для дальнейшего логического и физического проектирования.

- **Обучение и ресурсы**. Для пользователей Windows могут быть доступны больше материалов и курсов, ориентированных на экосистему Microsoft.

Рекомендации для Linux

- **Инструменты**. В Linux популярны открытые инструменты, такие как pgAdmin (для PostgreSQL), DBeaver или онлайн-сервисы вроде ERBuilder. Также можно использовать командные инструменты и скрипты для автоматизации части работы.
- **Гибкость и настройка**. Linux позволяет глубже настраивать среду разработки, использовать скрипты для генерации моделей или интеграции с системами контроля версий (например, Git).
- **Работа с контейнеризациями**. Если модель будет реализовываться в контейнеризированной среде (например, с использованием Docker), в Linux это будет более естественно и удобно.

Общие рекомендации

- **Нормализация**. Хотя нормализация относится к логическому проектированию, уже на этапе концептуального моделирования стоит стремиться к минимизации избыточности данных.
- **Учёт масштабируемости**. При проектировании модели думайте о том, как система будет работать при росте объёма данных. Это повлияет на выбор структуры и связей.
- **Документирование**. Ведите подробную документацию по модели: описания сущностей, атрибутов, связей и ограничений. Это упростит дальнейшую разработку и поддержку.
- **Согласование с заинтересованными сторонами**. Покажите модель пользователям, бизнес-аналитикам и другим участникам проекта для получения обратной связи и уточнения требований.
- **Использование стандартов и методологий**. Применяйте устоявшиеся методологии, такие как IDEF1X, чтобы обеспечить единообразие и понятность модели.

Концептуальная модель служит основой для дальнейшего логического и физического проектирования базы данных, поэтому её качество напрямую влияет на успех всего проекта.

Задание 1.3. Даталогическое проектирование базы данных. Определение всех информационных единиц и связей между ними, задание их имен и типов, а также некоторых количественных характеристик. преобразование исходной инфологической модели в модель данных, поддерживаемую конкретной СУБД, и проверка адекватности полученной даталогической модели отображаемой предметной области.

Основные этапы проектирования.

1. **Анализ инфологической модели.**

- Изучение ER-диаграмм.
- Определение сущностей и их атрибутов.
- Анализ связей между сущностями.
- Выявление ограничений целостности.

2. **Преобразование в даталогическую модель.**

- Создание таблиц.
- Определение типов данных.
- Установка связей между таблицами.
- Добавление ограничений.

Определение информационных единиц.

Основные компоненты:

- Таблицы (entities).
- Поля (attributes).
- Ключи (primary, foreign, unique).
- Индексы.
- Ограничения целостности.

Задание имен и типов данных.

Рекомендации по именованию:

- Таблицы: множественное число, префиксы (например, tbl_).
- Поля: camelCase или snake_case.
- Первичные ключи: id или table_id.
- Внешние ключи: table_id.

Типы данных:

- Числовые: INT, BIGINT, DECIMAL.

- Строковые: VARCHAR, TEXT, CHAR.
- Дата/время: DATE, DATETIME, TIMESTAMP.
- Логические: BOOLEAN.
- Двоичные: BLOB.

Количественные характеристики.

Параметры для определения:

- Максимальная длина строк.
- Диапазон значений для числовых полей.
- Точность и масштаб для десятичных чисел.
- Размер индексов.
- Ограничения на количество записей.

Преобразование модели.

Шаги преобразования:

1. Создание таблиц на основе сущностей.
2. Определение связей через внешние ключи.
3. Добавление индексов.
4. Реализация ограничений целостности.
5. Нормализация структуры.

Проверка адекватности модели.

Методы проверки:

- Тестирование типовых запросов.
- Проверка производительности.
- Анализ целостности данных.
- Стресс-тестирование.
- Проверка соответствия бизнес-требованиям.

Особенности для Windows.

Инструменты:

- SQL Server Management Studio.
- Visual Studio с Entity Framework.
- Microsoft Visio.
- PowerDesigner.

Рекомендации:

- Использование T-SQL специфичных функций.
- Интеграция с Active Directory.
- Применение встроенных инструментов мониторинга.
- Использование SQL Server Profiler.

Особенности для Linux.

Инструменты:

- pgAdmin для PostgreSQL.
- MySQL Workbench.
- DBeaver.
- phpMyAdmin.

Рекомендации:

- Использование Docker для развертывания.
- Применение инструментов мониторинга (Prometheus, Grafana).
- Настройка репликации и кластеризации.
- Оптимизация под Linux-системы.

Практические рекомендации.

1. **Нормализация данных.**

- Приведение к 3NF.
- Проверка на избыточность.
- Оптимизация запросов.

2. **Оптимизация производительности.**

- Создание индексов.
- Партиционирование таблиц.
- Оптимизация запросов.
- Кэширование данных.

3. **Обеспечение безопасности.**

- Разграничение доступа.
- Шифрование данных.
- Журналирование изменений.
- Регулярное резервное копирование.

Документирование модели.

Обязательные элементы:

- Схема базы данных.
- Описание таблиц и полей.
- Список ограничений.
- Документация по индексам.
- Инструкции по использованию.

Тестирование модели.

Виды тестирования:

- Функциональное.
- Нагрузочное.
- Стресс-тестирование.
- Тестирование безопасности.
- Тестирование восстановления.

При даталогическом проектировании важно:

- Учитывать особенности выбранной СУБД.
- Следовать стандартам именования.
- Обеспечивать нормализацию данных.
- Проводить тщательное тестирование.
- Документировать все изменения.

Успешное даталогическое проектирование обеспечивает:

- Эффективность работы с данными.
- Безопасность хранения информации.
- Масштабируемость системы.
- Простоту поддержки и развития.

Задание 1.4. Разработка сценариев работы с данными. Определение архитектуры системы.

Сценарий работы с данными — это последовательность действий пользователей и системы при выполнении определенных операций с данными. **Архитектура системы** определяет структуру, компоненты и их взаимодействие.

Этапы разработки сценариев.

1. **Анализ требований.**

- Определение целевой аудитории.
- Выявление основных операций с данными.

- Оценка нагрузки на систему.
- Определение требований к безопасности.

2. **Проектирование сценариев.**

- Создание пользовательских сценариев (use cases).
- Описание системных сценариев.
- Определение потоков данных.
- Документирование требований.

3. **Валидация сценариев.**

- Тестирование на реальных пользователях.
- Проверка полноты сценариев.
- Оценка производительности.

Определение архитектуры системы.

Основные компоненты архитектуры:

- Слой представления (UI/UX).
- Прикладной слой (бизнес-логика).
- Слой доступа к данным.
- Инфраструктурный слой.

Рекомендации для Windows.

Особенности архитектуры:

- Использование .NET Framework/Core.
- Интеграция с Active Directory.
- Работа с Windows-специфичными сервисами.
- Применение MS SQL Server.

Инструменты разработки:

- Visual Studio.
- SQL Server Management Studio.
- Windows Performance Monitor.
- IIS для веб-приложений.

Рекомендации для Linux.

Особенности архитектуры:

- Использование контейнеров (Docker).
- Работа с Linux-специфичными сервисами.

- Применение PostgreSQL/MySQL.
- Горизонтальное масштабирование.

Инструменты разработки:

- Docker/Kubernetes.
- PostgreSQL/MySQL Workbench.
- Prometheus для мониторинга.
- Nginx/Apache для веб-серверов.

Практические рекомендации по разработке сценариев.

1. **Сбор требований.**
- Проведение интервью с пользователями.
- Анализ существующих процессов.
- Документирование бизнес-правил.
2. **Проектирование.**
- Создание диаграмм последовательностей.
- Разработка схем данных.
- Определение API-интерфейсов.
3. **Тестирование.**
- Проверка корректности сценариев.
- Оценка производительности.
- Тестирование безопасности.

Архитектурные паттерны.

Популярные паттерны:

- MVC (Model-View-Controller).
- Microservices.
- Event-Driven Architecture.
- Serverless Architecture.

Документация архитектуры.

Обязательные разделы:

- Концептуальная схема архитектуры.
- Диаграммы компонентов.
- Описание интерфейсов.
- Требования к производительности.

- План развертывания.

Мониторинг и оптимизация.

Ключевые метрики:

- Время отклика системы.
- Загрузка CPU/RAM.
- Количество одновременных подключений.
- Скорость обработки данных.

Безопасность системы.

Основные меры:

- Аутентификация и авторизация.
- Шифрование данных.
- Журналирование действий.
- Защита от DDoS-атак.

Масштабируемость.

Рекомендации:

- Проектирование с учетом роста нагрузки.
- Использование кэширования.
- Оптимизация запросов к базе данных.
- Балансировка нагрузки.

Практические советы.

1. **Планирование ресурсов.**

- Оценка необходимого оборудования.
- Расчет требуемой памяти.
- Определение пропускной способности сети.

2. **Тестирование.**

- Модульное тестирование.
- Интеграционное тестирование.
- Нагрузочное тестирование.

3. **Документация.**

- Создание технической документации.
- Описание API.
- Руководство администратора.

При разработке сценариев работы с данными и определении архитектуры системы важно учитывать:

- Специфику платформы (Windows/Linux).
- Требования к производительности.
- Безопасность данных.
- Масштабируемость решения.
- Удобство поддержки и развития.

Грамотно спроектированная архитектура и проработанные сценарии работы с данными обеспечат надежность, производительность и удобство использования системы.

ТЕМА 2. СЕРВЕРЫ БАЗ ДАННЫХ.

Сервер баз данных — это специализированное техническое оборудование с программным обеспечением (ПО), предназначенное для хранения, обработки и управления структурированными данными. Он обеспечивает централизованный доступ к базам данных (БД) через систему управления базами данных (СУБД), позволяя пользователям и приложениям выполнять запросы, вносить изменения и получать информацию. В производственной практике работа с серверами баз данных включает их установку, настройку, администрирование, обеспечение безопасности и мониторинг.

Основные функции серверов баз данных

- **Хранение данных.** Серверы обеспечивают централизованное хранение структурированных данных в виде таблиц, таблиц, представлений и других объектов БД.
- **Обработка запросов.** Выполняют операции чтения, записи, обновления и удаления данных по запросам от клиентов (приложений, веб-сервисов и т. д.).
- **Управление доступом.** Контролируют права пользователей, обеспечивают аутентификацию и авторизацию.
- **Обеспечение целостности данных.** Поддерживают ограничения целостности, транзакции (ACID-свойства — атомарность, согласованность, изоляция, долговечность) и ссылочную целостность.

- **Резервное копирование и восстановление.** Позволяют создавать копии данных и восстанавливать их после сбоев.
- **Масштабирование.** Поддерживают увеличение нагрузки за счёт добавления ресурсов (памяти, процессоров, дисков) или кластеризации.

Популярные СУБД

К ним относятся:

- MySQL;
- PostgreSQL;
- Oracle;
- Microsoft SQL Server;
- MongoDB.

Аппаратные требования

Конфигурация сервера зависит от типа СУБД, объёма данных и нагрузки. Некоторые общие рекомендации:

- **Процессор.** Многопоточные модели с высокой тактовой частотой. Для крупных систем — несколько процессоров.
- **Оперативная память (ОЗУ).** Достаточный объём для обработки больших объёмов информации. Для высоконагруженных систем — от 32 ГБ и выше.
- **Дисковая подсистема.** Использование SSD (особенно NVMe) для быстрого доступа к данным. Часто применяется RAID-массив (например, RAID 10) для повышения надёжности и производительности.
- **Сетевые интерфейсы.** Минимум две сетевые карты Ethernet (рекомендуется 1 Гбит/с или выше).
- **Блоки питания.** Резервированные модульные БП с горячей заменой.

Программные требования

- **Операционная система.** Windows Server, Linux (например, Ubuntu, CentOS).
- **СУБД.** Выбор зависит от требований проекта (например, MySQL, PostgreSQL, SQL Server).
- **Дополнительные инструменты.** ПО для мониторинга, резервного копирования, управления доступом.

Администрирование серверов баз данных

Включает:

- **Установку и настройку СУБД.** Конфигурацию параметров памяти, кэширования, параллелизма запросов и других аспектов работы сервера.
- **Управление пользователями и правами доступа.** Создание учётных записей, назначение привилегий, настройка аутентификации.
- **Мониторинг и оптимизацию.** Отслеживание производительности, нагрузки, выявление узких мест и их устранение (например, оптимизация запросов, настройка индексов).
- **Резервное копирование и восстановление.** Регулярное создание копий данных, тестирование процедур восстановления.
- **Обновление ПО.** Установка патчей и новых версий СУБД для устранения уязвимостей и повышения производительности.
- **Обеспечение безопасности.** Настройка файрвола, шифрование данных, аудит доступа.

Безопасность серверов баз данных

Ключевые меры:

- изменение стандартного пароля для суперпользователя (например, root в MySQL);
- создание пользователей с ограниченными правами доступа;
- настройка файрвола для ограничения доступа к серверу с определённых IP-адресов;
- использование шифрования данных (например, Transparent Data Encryption в SQL Server);
- регулярный аудит безопасности и мониторинг подозрительной активности.

Инструменты для администрирования

- **SQL Server Management Studio (SSMS)** — для управления SQL Server.
- **phpMyAdmin, MySQL Workbench** — для работы с MySQL.
- Утилиты для резервного копирования (например, RMAN для Oracle, pg_dump для PostgreSQL).

Навыки, необходимые для производственной практики

- установка и настройка серверов баз данных;
- работа с SQL для создания, изменения и удаления объектов БД;
- администрирование пользователей и управление правами доступа;
- оптимизация производительности серверов;
- настройка резервного копирования и восстановления;
- мониторинг состояния сервера и анализ логов;
- обеспечение безопасности данных и серверов.

В рамках производственной практики студенты обычно выполняют задания, связанные с этими навыками, например, устанавливают и настраивают сервер MySQL, создают резервные копии БД, проводят аудит безопасности.

Если вам нужны дополнительные детали по конкретным аспектам (например, настройка определённого типа СУБД или инструменты мониторинга), уточните запрос.

Задание 2.1. Разработка технических требований к серверу баз данных. Разработка требований к корпоративной сети.

Разработка технических требований к серверу баз данных и корпоративной сети требует учёта множества факторов, включая тип СУБД, ожидаемую нагрузку, требования к надёжности, безопасности и масштабируемости. Рассмотрим ключевые аспекты для Windows и Linux.

Требования к серверу баз данных.

- **Анализ нагрузки**: определение количества пользователей, запросов в секунду, объёма данных и частоты их обновления.
- **Нормализация данных**: минимизация избыточности для оптимизации производительности.
- **Резервирование и отказоустойчивость**: использование RAID-массивов, кластеризации, репликации данных.
- **Мониторинг и логирование**: внедрение систем для отслеживания состояния сервера и аудита операций.

Для Windows.

Аппаратные требования:

- Процессор: многоядерный (например, Intel Xeon или AMD EPYC) с тактовой частотой от 2 ГГц.
- Оперативная память: минимум 16 ГБ, для крупных нагрузок — 64 ГБ и выше.
- Диск: SSD с минимальным IOPS (например, для SQL Server рекомендуется от 12 000 IOPS на чтение и 4 000 на запись).
- Сетевой интерфейс: минимум 1 Гбит/с.

Программные требования:

- Поддержка конкретной СУБД (например, Microsoft SQL Server, Oracle).
- Актуальная версия Windows Server (например, Windows Server 2016 или новее).
- Необходимые компоненты: .NET Framework (для SQL Server), службы агента SQL Server и др..
- Настройка брандмауэра для разрешения доступа к СУБД.

Дополнительные рекомендации:

- Разделение ролей: не размещать базу данных и контроллер домена на одном сервере.
- Использование Server Core для минимизации поверхности атаки.

Для Linux.

Аппаратные требования:

- Процессор: многоядерный (например, Intel Xeon, AMD EPYC) с тактовой частотой от 2 ГГц.
- Оперативная память: минимум 8 ГБ, для крупных нагрузок — 32 ГБ и выше.
- Диск: SSD с минимальным IOPS (например, для PostgreSQL рекомендуется RAID 1).
- Сетевой интерфейс: минимум 1 Гбит/с.

Программные требования:

- Поддержка конкретной СУБД (например, PostgreSQL, MySQL, MongoDB).

- Дистрибутив Linux: Red Hat Enterprise Linux, SUSE Linux Enterprise Server, Ubuntu LTS и др.
- Файловая система: XFS или ext4 (другие, например BTRFS, не поддерживаются для некоторых СУБД).
- Необходимые пакеты и зависимости для выбранной СУБД.

Дополнительные рекомендации:

- Использование контейнеров (Docker, Kubernetes) для изоляции и упрощения управления.
- Настройка swap-файла при объёме ОЗУ менее 4 ГБ.

Требования к корпоративной сети.

Общие принципы.

- **Высокая производительность**: обеспечение достаточной пропускной способности для обработки трафика.
- **Доступность**: минимизация простоев, использование резервных каналов связи.
- **Безопасность**: сегментация сети, межсетевые экраны, шифрование трафика, контроль доступа.
- **Масштабируемость**: возможность расширения сети при росте компании.

Для Windows.

- **DNS и DHCP**: централизованное управление именами и IP-адресами.
- **Active Directory**: для централизованного управления пользователями, компьютерами и групповыми политиками.
- **Брандмауэр и антивирус**: настройка правил безопасности, использование Microsoft Defender.
- **VPN и RADIUS-аутентификация** для удалённого доступа.
- **WSUS или SCCM** для централизованного обновления ПО.

Для Linux.

- **Поддержка протоколов**: TCP/IP, IPv6, DHCP, PPP, SLIP и др..
- **Сегментация сети**: разделение на VLAN для повышения безопасности (например, отдельная подсеть для серверов).

- **Межсетевые экраны**: использование iptables или более современных решений (например, nftables).
- **Шифрование трафика**: VPN, SSL/TLS для защиты данных.
- **Централизованное управление**: использование инструментов вроде Ansible, Puppet или Chef для управления конфигурациями.

Дополнительные рекомендации для обеих платформ

- **QoS (Quality of Service)**: приоритизация трафика для критически важных приложений.
- **Резервирование каналов связи**: использование нескольких интернет-провайдеров или резервных линий.
- **Мониторинг сети**: внедрение систем для отслеживания состояния сети, трафика, доступности сервисов (например, Zabbix, Nagios).
- **Регулярное тестирование на уязвимости**: пентесты, сканирование портов.
- **Документация**: создание схемы сети, списка IP-адресов, настроек маршрутизации и безопасности.

Сравнение подходов

Критерий	Windows	Linux
Интеграция с Active Directory	Нативная поддержка	Требуется интеграция через LDAP или другие механизмы
Гибкость настройки	Ограничена стандартными инструментами Microsoft	Высокая благодаря открытому ПО и разнообразию инструментов
Стоимость	Обычно выше из-за лицензионных требований	Чаще ниже благодаря бесплатным дистрибутивам и ПО
Сообщество и поддержка	Большая база знаний и поддержка Microsoft	Активное сообщество и множество открытых

Критерий	Windows	Linux
		ресурсов

При разработке требований важно учитывать специфику конкретной организации: размер, отрасль, бюджет, уровень подготовки персонала и существующие инфраструктурные решения. Для сложных проектов рекомендуется привлекать специалистов в области системного администрирования и сетевой безопасности.

Задание 2.2. Разработка серверной и клиентских компонент базы данных.

Разработка серверной и клиентской компонент базы данных включает создание и настройку сервера для хранения и обработки данных, а также разработку клиентского приложения для взаимодействия с этим сервером. Подходы могут различаться в зависимости от операционной системы (Windows и Linux), используемой СУБД и технологий разработки.

Серверная компонента.

Общие принципы:

- выбор СУБД (SQL Server, PostgreSQL, MySQL и др.);
- настройка аппаратных и программных ресурсов;
- создание и конфигурация базы данных;
- обеспечение безопасности и резервного копирования.

Для Windows.

1. **Выбор СУБД.** Например, Microsoft SQL Server — популярный выбор для Windows-среды. Также можно использовать SQL Server Express — бесплатную версию с базовым набором функций.

2. **Установка и настройка.** Используйте SQL Server Management Studio (SSMS) для управления сервером и базами данных. Настройте параметры сервера, включая память, сетевые протоколы (например, TCP/IP) и аутентификацию (Windows Authentication или смешанный режим).

3. **Создание базы данных и таблиц.** Используйте SSMS или скрипты T-SQL для создания баз данных, таблиц, индексов и ограничений. Можно применять конструктор таблиц в Visual Studio для визуального проектирования.

4. **Резервное копирование и восстановление.** Настройте регулярное резервное копирование баз данных через SSMS или планировщик задач Windows.

5. **Мониторинг и оптимизация.** Используйте инструменты вроде SQL Server Profiler для отслеживания запросов и производительности, а также настройте индексы и статистику для оптимизации запросов.

Для Linux.

1. **Выбор СУБД.** Часто используются PostgreSQL, MySQL или MongoDB. Например, PostgreSQL можно установить через пакетные менеджеры (apt, yum).

2. **Установка и настройка.** Для PostgreSQL потребуется настроить параметры в файле `postgresql.conf`, включая кодировку (обычно UTF-8), параметры подключения и локаль. Используйте `pgAdmin` для управления базами данных (для Linux его нужно устанавливать отдельно).

3. **Создание базы данных и таблиц.** Используйте `psql` (консольный клиент) или `pgAdmin` для создания баз данных, таблиц и пользователей. Например, для создания пользователя выполните скрипт:

```
sql
CREATE ROLE "USER_NAME" LOGIN ENCRYPTED PASSWORD 'USER_PASSWORD' NOSUPERUSER INHERIT NOCREATEDB NOCREATEROLE NOREPLICATION;
```

Для создания базы данных:

```
sql
CREATE DATABASE "DATABASE_NAME" WITH OWNER = "USER_NAME" ENCODING = 'UTF8' ...;
```

4. **Резервное копирование.** Используйте `pg_dump` для резервного копирования PostgreSQL или соответствующие инструменты для других СУБД.

5. **Мониторинг.** Применяйте инструменты вроде `Prometheus` или `Zabbix` для мониторинга состояния сервера и нагрузки на базу данных.

Клиентская компонента

Клиентское приложение отвечает за взаимодействие пользователя с базой данных через интерфейс, отправку запросов и отображение результатов.

Общие принципы

- выбор языка программирования и фреймворков;
- использование API или библиотек для работы с СУБД;
- обработка ошибок и управление соединениями;
- обеспечение безопасности (например, проверка ввода данных для предотвращения SQL-инъекций).

Для Windows.

1. **Языки и технологии.** Часто используются C# с .NET Framework или .NET Core, Visual Basic. Для работы с SQL Server можно применять ADO.NET, Entity Framework или LINQ to SQL.

2. **Создание приложения.** Например, в Visual Studio можно создать проект Windows Forms или WPF, добавить источник данных (через мастер настройки источника данных), связать его с базой данных и использовать компоненты вроде `DataGridView` для отображения данных.

3. **Работа с данными.** Используйте классы `SqlConnection`, `SqlCommand` и `SqlDataReader` (для ADO.NET) для подключения к базе данных, выполнения запросов и обработки результатов.

4. **Обработка ошибок.** Реализуйте обработку исключений, чтобы гарантировать закрытие соединений даже при ошибках.

Для Linux

1. **Языки и технологии.** Популярны Python с библиотеками `psycopg2` (для PostgreSQL), `mysql-connector-python` (для MySQL), C/C++ с использованием ODBC или специфических API СУБД.

2. **Создание приложения.** Например, на Python можно использовать фреймворки вроде Django или Flask для веб-приложений с доступом к базе данных. Для настольных приложений подойдут библиотеки вроде Tkinter или PyQt.

3. **Работа с данными.** Используйте ORM (Object-Relational Mapping), например SQLAlchemy для Python, чтобы упростить взаимодействие с базой данных.

4. **Безопасность.** Применяйте параметризованные запросы или ORM-методы для предотвращения SQL-инъекций.

Общие рекомендации.

- **Нормализация данных.** Спроектируйте структуру базы данных с учётом нормализации, чтобы избежать избыточности и упростить обновления.
- **Индексация.** Создавайте индексы для столбцов, которые часто используются в запросах, чтобы ускорить поиск данных.
- **Тестирование.** Проводите юнит-тесты и интеграционные тесты для проверки корректности работы как серверной, так и клиентской части.
- **Масштабируемость.** При проектировании учитывайте возможность роста нагрузки и добавляйте возможности горизонтального или вертикального масштабирования.
- **Документация.** Ведите документацию по структуре базы данных, API и клиентскому приложению для упрощения поддержки и развития.

При разработке важно учитывать специфику выбранной СУБД и языка программирования, а также требования к проекту по производительности, безопасности и удобству использования.

Задание 2.3. Установка и настройка сервера баз данных.

Установка и настройка сервера баз данных зависит от выбранной СУБД и операционной системы. Рассмотрим ключевые шаги для Windows и Linux на примере MySQL и PostgreSQL.

Установка и настройка MySQL.

В Windows.

1. **Скачивание дистрибутива**. Загрузите MySQL Installer for Windows с официального сайта MySQL. Рекомендуется выбирать последнюю стабильную версию.

2. **Запуск установщика**. Дважды кликните по загруженному файлу, чтобы запустить мастер. При необходимости система может запросить права администратора.

3. **Выбор типа установки**. В мастере доступны варианты:

- **Developer Default** — установка всех необходимых компонентов для разработки.
- **Server Only** — только сервер и базовые инструменты.
- **Custom** — выбор компонентов вручную. Для начинающих чаще всего достаточно Developer Default.

4. **Проверка зависимостей**. Установщик проверит наличие нужных компонентов, например Visual C++ Redistributable. Если чего-то не хватает, мастер предложит доустановить.

5. **Процесс установки**. Нажмите Execute, чтобы начать установку. По завершении каждого шага мастер покажет статус и наличие ошибок или предупреждений.

6. **Первичная настройка**. В конце установки автоматически запускается мастер настройки (Configuration). Он позволит выбрать тип конфигурации, задать порт (стандартный — 3306), включить или отключить сетевой доступ и установить пароль для root-пользователя.

7. **Проверка работы**. После завершения установки и конфигурации MySQL сервер должен автоматически запуститься как служба Windows. Проверить это можно в «Службах» (Services) через меню «Пуск» или поисковую строку Windows. Найдите службу с названием вроде MySQL80 или MySQL57 (зависит от версии) и убедитесь, что её статус — Работает (Running).

В Linux (на примере Ubuntu).

1. **Обновление пакетов и системы**:

```
bash
sudo apt update && sudo apt upgrade -y
```

2. **Установка MySQL**:

```
bash
sudo apt install mysql-server -y
```

3. **Проверка статуса службы**:

```
bash
sudo systemctl status mysql
```

Если сервис не активен, запустите его командой:

```
bash
systemctl start mysql
```

4. **Начальная настройка безопасности**. Запустите интерактивный скрипт `mysql_secure_installation`:

```
bash
sudo mysql_secure_installation
```

Скрипт предложит установить плагин валидации паролей, задать пароль для root-пользователя, удалить анонимные учётные записи и отключить удалённый доступ для root.

5. **Создание пользователя и базы данных**. Войдите в командную строку MySQL:

```
bash
mysql
```

Создайте базу данных:

```
sql
CREATE DATABASE db;
```

Создайте пользователя:

```
sql
CREATE USER 'db_user'@'localhost' IDENTIFIED BY 'pswrd';
```

Предоставьте пользователю права на базу данных:

```
sql
GRANT ALL PRIVILEGES ON db.* TO 'db_user'@'localhost';
FLUSH PRIVILEGES;
```

Выйдите из командной строки:

```
sql
mysql quit
```

Установка и настройка PostgreSQL.

В Windows.

1. **Скачивание дистрибутива**. Загрузите установочный файл PostgreSQL с официального сайта.

2. **Запуск установщика**. Дважды кликните по загруженному файлу.

3. **Выбор каталога для установки**. По умолчанию предлагается `C:\Program Files\PostgreSQL\15`, его можно оставить.

4. **Выбор дополнительных компонентов**. Вместе с СУБД можно установить:

- PostgreSQL Server — сам сервер баз данных;
- pgAdmin 4 — приложение с графическим интерфейсом для администрирования СУБД;
- Stack Builder — утилита для установки дополнительных библиотек и инструментов;
- Command Line Tools — инструменты для работы через командную строку.

5. **Указание каталога для файлов баз данных**. Можно оставить путь по умолчанию (`C:\Program Files\PostgreSQL\15\data`) или выбрать свой.

6. **Установка пароля для системного пользователя postgres**. Рекомендуется выбирать пароль из 12 и более символов, со строчными и прописными буквами, цифрами и символами.

7. **Выбор порта и кодировки**. Можно оставить порт по умолчанию (5432) или указать свой.

8. **Проверка параметров и завершение установки**. В финальном окне выводится сводная информация об установке. Если параметры указаны верно, нажимайте Next.

В Linux.

1. **Установка PostgreSQL**. Например, в Ubuntu:

```
bash
sudo apt install postgresql
```

2. **Проверка статуса службы**:

```
bash
sudo systemctl status postgresql
```

3. **Вход в систему как пользователь postgres**:

```
bash
sudo -u postgres psql
```

4. **Создание пользователя и базы данных**. Например:

```
sql
CREATE USER myuser WITH ENCRYPTED PASSWORD 'mypassword';
CREATE DATABASE mydb OWNER myuser;
```

5. **Настройка доступа**. Если требуется удалённый доступ, отредактируйте файл `pg_hba.conf` (обычно находится в `/etc/postgresql//main/`) и файл `postgresql.conf`, указав нужный IP-адрес или диапазон. После изменений перезапустите службу:

```
bash
sudo systemctl restart postgresql
```

Общие рекомендации.

- **Резервное копирование**. Настройте регулярное резервное копирование баз данных. Для MySQL можно использовать `mysqldump`, для PostgreSQL — `pg_dump`.
- **Мониторинг**. Используйте инструменты для мониторинга состояния сервера и нагрузки на базу данных (например, Prometheus, Zabbix).
- **Обновления**. Регулярно обновляйте СУБД и операционную систему для устранения уязвимостей.
- **Безопасность**. Ограничьте доступ к серверу баз данных, используйте сложные пароли, отключайте ненужные учётные записи и удалённый доступ для администраторов, если это не требуется.
- **Тестирование**. После настройки проведите тестирование, создав тестовую базу данных и выполнив базовые операции (создание таблиц, вставка данных, запросы).

Если вы используете другую СУБД (например, SQL Server), уточните архитектуру, методики работы и типовые настройки у руководителя производственной практики от учебной организации или предприятия.

ТЕМА 3. АДМИНИСТРИРОВАНИЕ БАЗ ДАННЫХ И СЕРВЕРОВ.

Цели и задачи практики

Основная цель — получение практических навыков администрирования баз данных и серверных систем в реальных условиях эксплуатации.

Ключевые задачи:

- Освоение технологий установки и настройки серверного оборудования.
- Изучение методов обеспечения безопасности данных.
- Приобретение навыков мониторинга и оптимизации производительности.
- Освоение процедур резервного копирования и восстановления.
- Изучение принципов сертификации программных средств.

Основные компетенции.

Профессиональные компетенции:

- Выявление технических проблем в процессе эксплуатации.
- Администрирование компонентов серверов.
- Формирование требований к конфигурации серверного оборудования.
- Разработка политик безопасности.
- Проведение аудита систем безопасности.

Практические навыки.

Базовые навыки:

- Установка и настройка серверного ПО.
- Работа с системами управления базами данных.
- Создание и оптимизация баз данных.
- Выполнение запросов на языке SQL.
- Мониторинг производительности систем.

Основные направления работы.

Ключевые области:

- Администрирование серверных операционных систем.
- Управление базами данных.
- Обеспечение информационной безопасности.
- Техническая поддержка пользователей.
- Документирование процессов.

Технологический стек.

Используемое ПО:

- Системы управления базами данных (MS SQL Server, MySQL, PostgreSQL).
- Серверное программное обеспечение.
- Средства мониторинга и анализа.
- Инструменты резервного копирования.
- Системы безопасности.

Процессы администрирования.

Основные процессы:

- Установка и конфигурация серверного оборудования.
- Настройка и оптимизация баз данных.
- Обеспечение резервного копирования.
- Мониторинг производительности.
- Устранение технических неполадок.
- Обновление программного обеспечения.

Требования к знаниям и умениям.

Необходимые знания:

- Модели данных и основные операции.
- Технологии установки и настройки серверов.
- Принципы работы систем безопасности.
- Стандарты обслуживания баз данных.
- Основы программирования на SQL.

Практические умения:

- Проектирование и создание баз данных.
- Выполнение SQL-запросов.
- Администрирование серверных систем.
- Разработка политик безопасности.
- Проведение сертификации ПО.

Организация работы.

Основные этапы практики:

- Изучение документации и регламентов.
- Ознакомление с инфраструктурой.
- Выполнение практических заданий.

- Документирование результатов.
- Подготовка отчетной документации.

Контроль качества.

Методы оценки:

- Экспертное наблюдение за выполнением работ.
- Анализ результатов деятельности.
- Проверка отчетной документации.
- Оценка практических навыков.
- Тестирование теоретических знаний.

Документационное обеспечение.

Необходимая документация:

- Регламенты администрирования.
- Политики безопасности.
- Инструкции по эксплуатации.
- Журналы мониторинга.
- Отчеты о проделанной работе.

В рамках производственной практики студенты должны освоить все аспекты администрирования баз данных и серверов, получить практические навыки работы с современным оборудованием и программным обеспечением, а также научиться применять теоретические знания в реальных условиях эксплуатации информационных систем.

Задание 3.1. Создание механизмов сервера для обслуживания базы данных. Установка и развёртывание системы.

Создание механизмов сервера для обслуживания базы данных включает установку СУБД, настройку параметров, обеспечение безопасности, а также развёртывание системы в Windows и Linux. Рассмотрим ключевые этапы для обеих операционных систем.

Установка СУБД.

В Windows.

Например, для установки **Microsoft SQL Server**:

1. **Скачайте дистрибутив** с официального сайта Microsoft. Выберите нужную версию и архитектуру (x64).

2. **Запустите установщик** от имени администратора. В разделе «Установка» выберите «Новая установка изолированного экземпляра SQL Server».

3. **Выберите редакцию** (например, Developer для разработки или Enterprise для продакшена). При наличии лицензии введите ключ продукта.

4. **Примите условия лицензии** и перейдите к выбору компонентов. Обычно устанавливают «Службы ядра СУБД», «Репликация SQL Server» и «Полнотекстовый и семантический поиск».

5. **Настройте аутентификацию**: можно выбрать режим Windows Authentication (проверка через учётные записи Windows) или смешанный режим (с отдельными учётными записями SQL Server).

6. **Укажите каталоги для данных**, журналов, TempDB и резервных копий. Рекомендуется размещать их на разных дисках для оптимизации производительности.

7. **Запустите установку** и дождитесь завершения. После этого можно установить SQL Server Management Studio (SSMS) для управления сервером.

Для **MySQL** процесс похож: после скачивания дистрибутива запустите установщик, выберите тип установки (например, Full для всех компонентов) и следуйте инструкциям.

В Linux.

Для установки **SQL Server**:

1. **Настройте репозиторий Microsoft**. Например, для Ubuntu выполните:

```
bash
sudo apt update
sudo apt install mssql-server
```

2. **Настройте сервер** с помощью `mssql-conf setup`, указав пароль для учётной записи `sa` и другие параметры.

Для **PostgreSQL**:

1. **Добавьте репозиторий PostgreSQL** (если требуется). Например:

```
bash
sudo wget https://repo.postgrespro.ru/1c-15/keys/pgpro-repo-add.sh
sudo sh pgpro-repo-add.sh
```

2. **Обновите пакеты и установите PostgreSQL**:

```
bash
sudo apt update
sudo apt install postgresql postgresql-contrib
```

3. **Запустите сервис**: `sudo systemctl start postgresql`. AdminVPS.ru +1

Настройка параметров сервера.

Общие параметры.

- **Память**. Выделите достаточный объём ОЗУ для буфера кэша и других компонентов СУБД.
- **Процессор**. Убедитесь, что количество ядер соответствует нагрузке.
- **Файловая система**. Для Linux используйте XFS или ext4 (другие файловые системы, например BTRFS, не поддерживаются для SQL Server).

Безопасность.

- **Пароли**. Установите сложные пароли для административных учётных записей (минимум 12 символов, включая буквы, цифры и спецсимволы).
- **Доступ**. Ограничьте удалённый доступ для административных учётных записей (например, root в MySQL).
- **Брандмауэр**. Настройте правила для разрешения необходимого трафика (например, порт 1433 для SQL Server, 5432 для PostgreSQL).

Сетевые настройки.

- В PostgreSQL отредактируйте `postgresql.conf`, чтобы разрешить подключения с нужных IP-адресов (параметр `listen_addresses`). В `pg_hba.conf` укажите правила доступа для пользователей и хостов. AdminVPS.ru +1
- В SQL Server настройте сетевые протоколы (TCP/IP) через SQL Server Configuration Manager.

Развёртывание системы

В Windows

1. **Добавьте сервер в домен** (если требуется).
2. **Настройте планировщик задач** для регулярного резервного копирования, обслуживания индексов и других регламентных операций.

3. **Используйте SSMS** (для SQL Server) или MySQL Workbench для управления базами данных, создания пользователей и настройки прав доступа.

В Linux

1. **Настройте автозапуск сервиса**:

```
bash
sudo systemctl enable postgresql
```

2. **Используйте cron или systemd** для автоматизации задач обслуживания.

3. Для управления PostgreSQL можно установить pgAdmin4 или использовать командную строку psql. softonit.ru +1

Дополнительные рекомендации

- **Резервное копирование**. Настройте регулярное резервное копирование баз данных и журналов транзакций. Для SQL Server используйте планы обслуживания в SSMS, для PostgreSQL — скрипты или сторонние инструменты.
- **Мониторинг**. Используйте инструменты для отслеживания производительности (например, SQL Server Profiler для SQL Server, pg_stat_statements для PostgreSQL).
- **Обновления**. Регулярно обновляйте СУБД и операционную систему для устранения уязвимостей и улучшения производительности.
- **Тестирование**. После развёртывания проведите тестирование на нагрузку и проверьте корректность работы всех компонентов.

Сравнение ключевых аспектов.

Аспект	Windows	Linux
Установка	Через графический установщик	Через пакетные менеджеры (apt, yum и др.)
Управление	SSMS, PowerShell	Командная строка, графические инструменты (pgAdmin, phpMyAdmin)

Аспект	Windows	Linux
Интеграция с ОС	Тесная интеграция с Windows	Требует настройки прав доступа и окружения
Поддержка	Официальная поддержка Microsoft	Поддержка сообществом, требует больше ручной настройки

При выборе СУБД и стратегии развёртывания учитывайте специфику проекта, требования к производительности, бюджет и уровень квалификации администраторов.

Задание 3.2. Работа с журналом аудита базы данных. Мониторинг нагрузки сервера.

Работа с журналом аудита базы данных и мониторинг нагрузки сервера — ключевые задачи для обеспечения безопасности, стабильности и производительности системы. Подходы различаются в зависимости от операционной системы и используемой СУБД.

Работа с журналом аудита базы данных

В Windows (SQL Server)

Для аудита в SQL Server используется встроенная подсистема, которая позволяет отслеживать различные события, связанные с доступом к данным, изменениями в структуре базы и другими операциями.

Настройка аудита:

1. В SQL Server Management Studio (SSMS) перейдите в раздел «Безопасность» → «Аудиты».
2. Создайте новый аудит, указав путь к файлу журнала и другие параметры.
3. Настройте спецификации аудита для конкретных серверов, баз данных или объектов, выбрав события, которые нужно отслеживать (например, изменения разрешений, доступ к объектам, операции с пользователями). learn.microsoft.com +1

Просмотр журнала:

- В SSMS: в обозревателе объектов раскройте папку «Безопасность» → «Аудиты», щёлкните правой кнопкой мыши по нужному журналу и выберите «Просмотр журналов аудита».
- С помощью функции `sys.fn_get_audit_file` в T-SQL для чтения файлов аудита в необработанном формате.

Рекомендации:

- Фильтруйте события по типам, временным диапазонам и другим параметрам для упрощения анализа.
- Настройте ротацию журналов и ограничение их размера, чтобы избежать переполнения диска.
- Используйте централизованное хранение журналов для удобства анализа и защиты от удаления злоумышленником.

В Linux (PostgreSQL, MySQL)

В Linux для аудита системных событий, включая доступ к базам данных, часто используется демон **auditd**. docs.oracle.com +2

Настройка auditd:

1. Установите пакет `audit` (например, через `sudo dnf install audit` для дистрибутивов на базе RPM).
2. Настройте правила в файле `/etc/audit/audit.rules` или с помощью утилиты `auditctl`. Например, чтобы отслеживать доступ к файлу `/etc/passwd`, выполните:

```
bash
sudo auditctl -w /etc/passwd -p wra -k passwd
```

где `-w` — путь к файлу, `-p` — права доступа (w — запись, r — чтение, a — изменение атрибутов), `-k` — ключ для маркировки событий. docs.oracle.com +2

Просмотр журналов:

- Журналы по умолчанию хранятся в `/var/log/audit/audit.log`.
- Для поиска событий используйте `ausearch` (например, `ausearch -k passwd` для событий с ключом `passwd`).

- Для создания сводных отчётов применяйте `aureport`.

Для PostgreSQL и MySQL можно также настраивать внутреннее журналирование событий через параметры конфигурации баз данных (например, логирование запросов, изменений в структуре БД).

Мониторинг нагрузки сервера.

В Windows.

Инструменты и метрики:

- **Системный монитор (Performance Monitor, Perfmon.exe)** — основное средство для отслеживания метрик: использование CPU, памяти, дискового ввода-вывода, сетевых ресурсов. Можно создавать пользовательские счётчики и графики.
- **Диспетчер задач** — для быстрого просмотра загрузки CPU и памяти.
- **Монитор активности в SSMS** — показывает текущие процессы, блокировки, дорогие запросы в SQL Server.
- **Динамические административные представления (DMV) в SQL Server** — для детального анализа производительности, например, потребления памяти, операций ввода-вывода.

Рекомендации:

- Настройте оповещения при достижении критических значений метрик (например, через SQL Server Alerts & Notifications или System Monitor).
- Используйте инструменты вроде dbForge Monitor или скрипты Brentozar (sp_blitz) для упрощённого анализа.

В Linux

Консольные утилиты:

- `top`/`htop` — мониторинг процессов и загрузки CPU.
- `free -m` — использование памяти.
- `vmstat` — сводка по ресурсам.
- `iostat` — дисковая активность.
- `netstat`/`ss` — сетевые соединения.

Графические и продвинутые системы:

- **Zabbix, Prometheus + Grafana, Nagios** — для централизованного мониторинга с графиками, алертами и историей данных.
- **Netdata** — быстрый мониторинг в реальном времени с современным интерфейсом.
- **Cacti** — система для построения графиков состояния параметров.

Для баз данных:

- В MySQL используйте `SHOW PROCESSLIST`, `SHOW STATUS`, активируйте логирование медленных запросов (`SET GLOBAL slow_query_log = 1`).
- Для PostgreSQL применяйте `pg_stat_activity` и другие системные представления.
- Инструменты вроде Percona Toolkit для анализа и оптимизации MySQL.

Общие рекомендации

1. **Базовые линии (baseline)**. Зафиксируйте нормальные значения метрик (загрузка CPU, память, дисковый ввод-вывод и т. д.) в штатных условиях. Это поможет оперативно выявлять отклонения.
2. **Алерты и уведомления**. Настройте оповещения при превышении пороговых значений или возникновении критических событий.
3. **Автоматизация**. Используйте скрипты или готовые решения для регулярного сбора данных и генерации отчётов.
4. **Централизация данных**. Собирайте логи и метрики с нескольких серверов в единую систему для упрощения анализа.
5. **Анализ трендов**. Отслеживайте изменения метрик во времени, чтобы прогнозировать нагрузку и планировать масштабирование.

Выбор инструментов и подходов зависит от конкретных требований, масштаба инфраструктуры и используемых технологий.

ТЕМА 4. ЗАЩИТА И СОХРАННОСТЬ ИНФОРМАЦИИ БАЗ ДАННЫХ.

Информационная безопасность БД — комплекс мер, направленных на обеспечение конфиденциальности, целостности и доступности данных.

Сохранность информации — совокупность мероприятий по защите данных от несанкционированного доступа, разрушения и модификации.

Цели и задачи защиты.

Основные цели:

- Обеспечение конфиденциальности данных.
- Поддержание целостности информации.
- Гарантирование доступности данных.
- Защита от несанкционированного доступа.

Ключевые задачи:

- Предотвращение несанкционированного доступа.
- Обеспечение резервного копирования.
- Защита от вирусов и вредоносного ПО.
- Мониторинг безопасности.
- Восстановление данных после сбоев.

Методы защиты информации.

Технические методы:

- Шифрование данных.
- Аутентификация пользователей.
- Контроль доступа.
- Резервное копирование.
- Системы обнаружения вторжений.

Организационные методы:

- Разработка политик безопасности.
- Обучение персонала.
- Регламентация доступа.
- Документирование процедур.

Средства обеспечения безопасности.

Программные средства:

- Системы управления доступом.
- Антивирусное ПО.
- Средства шифрования.
- Системы мониторинга.
- Инструменты резервного копирования.

Аппаратные средства:

- Файрволы.
- Системы резервного хранения.
- Средства биометрической защиты.
- Аппаратные модули шифрования.

Процедуры обеспечения сохранности.

Основные процедуры:

- Регулярное резервное копирование.
- Тестирование процедур восстановления.
- Мониторинг целостности данных.
- Аудит безопасности.
- Обновление защитных механизмов.

Практические аспекты для производственной практики.

Необходимые навыки:

- Настройка систем защиты.
- Работа с системами резервного копирования.
- Проведение аудита безопасности.
- Восстановление данных.
- Мониторинг угроз.

Этапы работы с системой защиты.

1. **Анализ угроз.**

- Выявление потенциальных рисков.
- Оценка уязвимостей.
- Определение критичности данных.

2. **Проектирование системы защиты.**

- Выбор методов и средств защиты.
- Разработка политик безопасности.

- Планирование процедур резервного копирования.

3. **Реализация защитных мер.**

- Установка и настройка средств защиты.
- Внедрение процедур безопасности.
- Обучение персонала.

4. **Мониторинг и поддержка.**

- Регулярный аудит безопасности.
- Обновление защитных механизмов.
- Анализ инцидентов.

Документационное обеспечение.

Необходимая документация:

- Политики безопасности.
- Инструкции по работе с данными.
- Регламенты резервного копирования.
- Журналы мониторинга.
- Отчеты об инцидентах.

Оценка эффективности защиты.

Критерии оценки:

- Время восстановления после сбоев.
- Процент успешных операций резервного копирования.
- Количество предотвращенных инцидентов.
- Скорость реагирования на угрозы.
- Удовлетворенность пользователей уровнем защиты.

В рамках производственной практики студенты должны освоить все аспекты защиты и сохранности информации, получить практические навыки работы с современными средствами защиты, научиться применять теоретические знания в реальных условиях эксплуатации информационных систем.

Задание 4.1. Настройка политики безопасности для созданной базы данных.

Настройка политики безопасности базы данных включает управление доступом, аутентификацией, шифрованием, сетевыми ограничениями и регулярный аудит. Подходы различаются в зависимости от СУБД и операционной системы.

В Windows (SQL Server).

1. Аутентификация и управление учётными записями:

- Используйте **аутентификацию Windows** (Windows Authentication), если это возможно. Это безопаснее, чем смешанный режим, так как данные пользователя проверяются на уровне ОС.
- Для учётных записей служб SQL Server не используйте доменные аккаунты с привилегиями администратора домена. Лучше применять **групповые управляемые учётные записи службы (gMSA)**, которые автоматически управляются Active Directory.
- Настройте политику паролей для логинов SQL Server, чтобы принудительно применять политики Windows (например, срок действия пароля, сложность). Это можно сделать при создании логина в SQL Server Management Studio (SSMS), отметив опции *Enforce Password Policy* и *Enforce Password Expiration*.

2. Управление доступом (принцип наименьших привилегий):

- Используйте **безопасность на основе ролей**. Назначайте разрешения ролям, а не отдельным пользователям. В SQL Server есть встроенные роли сервера и баз данных с предопределёнными правами (например, `sysadmin`, `securityadmin`).
- Ограничьте права пользователей. Например, для приложений создавайте учётные записи с минимальными необходимыми привилегиями. Избегайте предоставления прав системного администратора (`sysadmin`) обычным пользователям.
- Не допускайте, чтобы пользователи без прав системного администратора владели схемой `dbo` или объектами в ней.

3. Сетевая безопасность:

- Настройте **брандмауэр Windows**, чтобы разрешить доступ только с доверенных IP-адресов. Для SQL Server обычно требуется открыть порт 1433 (по умолчанию) или пользовательский порт, если он изменён.
- Отключите ненужные сетевые протоколы (например, Named Pipes, если используется TCP/IP).

- Измените порт по умолчанию для SQL Server, чтобы снизить риск атак, нацеленных на известные порты.

4. Шифрование и аудит:

- Включите **шифрование трафика** с помощью SSL/TLS, если требуется удалённый доступ.
- Настройте **аудит** для отслеживания критических событий (например, доступа к данным, изменений в структуре БД). В SSMS это делается через раздел *Security → Audits*.

5. Дополнительные меры:

- Регулярно обновляйте SQL Server и операционную систему.
- Используйте функцию **оценки уязвимостей** в SSMS (*Tasks → Vulnerability Assessment → Scan For Vulnerabilities*) для выявления проблем в конфигурации безопасности.
- Ограничьте доступ к физическим ресурсам: убедитесь, что только учётная запись службы SQL Server имеет права на доступ к каталогам с данными.

В Linux (PostgreSQL, MySQL/MariaDB).

1. Аутентификация и управление учётными записями:

- Для PostgreSQL используйте **SCRAM-SHA-256** вместо устаревшего MD5 для хэширования паролей.
- В MySQL/MariaDB после установки запустите скрипт `mysql_secure_installation`, чтобы установить пароль для root, удалить анонимных пользователей, отключить удалённый доступ для root и удалить тестовую базу данных.
- Создавайте отдельных пользователей для приложений с минимальными привилегиями. Например, в MySQL:

```
sql
CREATE USER 'appuser'@'localhost' IDENTIFIED BY 'SecurePassword';
GRANT SELECT, INSERT ON mydb.* TO 'appuser'@'localhost';
FLUSH PRIVILEGES;
```

2. Управление доступом:

- В PostgreSQL используйте **роли** для управления правами. Роли могут включать пользователей и другие роли, что упрощает управление доступом.

- Ограничьте владение объектами базы данных. Не допускайте, чтобы пользователи без административных прав владели критическими объектами.

3. Сетевая безопасность:

- Для PostgreSQL в файле `postgresql.conf` установите `listen_addresses = 'localhost'`, если удалённый доступ не требуется. Для разрешения доступа с конкретных IP-адресов настройте `pg_hba.conf`.
- В MySQL измените `bind-address` в `mysqld.cnf` на `127.0.0.1`, чтобы ограничить доступ локально. Для удалённого доступа используйте правила брандмауэра, чтобы разрешить доступ только с доверенных IP.
- Используйте `iptables` или `ufw` для ограничения доступа к портам (по умолчанию для PostgreSQL — 5432, для MySQL — 3306).

4. Шифрование и аудит:

- Включите **SSL/TLS** для шифрования трафика. Для PostgreSQL требуется настройка сертификатов и параметров в `postgresql.conf`. В MySQL используйте опцию `REQUIRE SSL` при создании пользователя.
- Настройте **аудит**. В MariaDB можно установить плагин `server_audit` и настроить параметры логирования.

5. Дополнительные меры:

- Регулярно обновляйте СУБД и операционную систему.
- Для PostgreSQL при изменении каталога данных смените контекст безопасности с помощью `semanage fcontext` и `restorecon`, если используется SELinux.
- Ограничьте права на файлы и каталоги. Например, в MySQL убедитесь, что только учётная запись службы имеет права на доступ к данным.
- Регулярно проверяйте журналы логов на предмет подозрительной активности.

Общие рекомендации

- **Регулярно аудитируйте** учётные записи пользователей и привилегии. Удаляйте неиспользуемые учётные записи.

- **Резервное копирование**: регулярно создавайте резервные копии данных и проверяйте их целостность.
- **Обновления**: следите за выходом патчей безопасности и своевременно обновляйте СУБД и ОС.
- **Мониторинг**: настройте систему оповещений для подозрительной активности (например, неудачных попыток входа, аномального использования ресурсов).
- **Документация**: ведите документацию по всем изменениям в конфигурации безопасности.

Безопасность базы данных — это непрерывный процесс, требующий постоянного внимания и адаптации к новым угрозам.

Задание 4.2. Создание резервных копий базы данных. Восстановление базы данных. Мониторинг активности портов.

Создание резервных копий базы данных, восстановление и мониторинг активности портов — ключевые задачи для обеспечения целостности данных и безопасности системы. Подходы различаются в зависимости от операционной системы и используемой СУБД.

Создание резервных копий базы данных.

В Windows (SQL Server).

Способы создания резервных копий:

1. **Через SQL Server Management Studio (SSMS).** Подключитесь к серверу, щёлкните правой кнопкой мыши по базе данных, выберите «Задачи» → «Создать резервную копию». Укажите тип резервной копии (полная, дифференциальная, резервная копия журнала транзакций), путь к файлу (.bak) и другие параметры.
2. **С помощью командной строки (sqlcmd).** Пример команды:

bash

```
sqlcmd -S <server> -U <user> -P <password> -Q "BACKUP DATABASE [<database>] TO DISK = N'<file path>' <options>"
```

Для автоматизации используйте планировщик заданий.

3. **Через PowerShell.** Импортируйте модуль `sqlps` и используйте команду:

powershell

```
Backup-SqlDatabase -ServerInstance <имя сервера> -Database <имя базы> -BackupFile <путь к файлу>
```

Типы резервных копий:

- **Полная** — копирует всю базу данных.
- **Дифференциальная (разностная)** — копирует изменения с момента последней полной резервной копии.
- **Резервная копия журнала транзакций** — фиксирует изменения в журнале транзакций.

Рекомендации:

- Регулярно создавайте резервные копии по расписанию.
- Сохраняйте копии на отдельные физические диски, не используемые для данных или журналов транзакций.
- Используйте сжатие резервных копий (в SQL Server 2008 и выше) для экономии места.
- Проверяйте целостность копий с помощью опции `WITH CHECKSUM`.

В Linux (PostgreSQL).

Утилиты для создания резервных копий:

1. **pg_dump** — создаёт дамп отдельной базы данных. Поддерживает разные форматы: plain (текстовый SQL), custom (сжатый архив), tar, directory. Пример:

bash

```
pg_dump -U postgres -h localhost -p 5432 -d mydb > mydb.sql
```

2. **pg_dumpall** — создаёт дамп всех баз данных, включая глобальные объекты (роли, табличные пространства).
3. **pg_basebackup** — создаёт резервную копию всего кластера PostgreSQL, включая данные и конфигурационные файлы.

Рекомендации:

- Используйте специального пользователя с правами только на чтение для создания резервных копий.
- Настройте файл `.pgpass` для безопасного хранения паролей.

- Для автоматизации используйте cron или другие планировщики.
- Рассмотрите многопоточное восстановление для форматов custom или directory с помощью `pg_restore -j N`.

Восстановление базы данных

В Windows (SQL Server)

1. В SSMS щёлкните правой кнопкой мыши по разделу «Базы данных» и выберите «Восстановить базу данных».
2. На странице «Общие» укажите имя восстанавливаемой базы, выберите источник (устройство) и путь к файлу резервной копии.
3. На странице «Параметры» можно настроить перезапись существующей базы данных, пути к файлам и другие опции.

Важно: если восстанавливаете на другой версии SQL Server, после восстановления рекомендуется изменить параметр «Уровень совместимости» на последнюю версию.

В Linux (PostgreSQL)

- Для восстановления из дампа, созданного `pg_dump`, используйте `psql`:

```
bash
psql -U postgres -h localhost -d mydb < mydb.sql
```

- Для восстановления из архива (форматы custom, tar, directory) используйте `pg_restore`:

```
bash
pg_restore -U postgres -h localhost -d mydb -j 4 mydb.dump
```

Ключ `-j 4` задаёт количество потоков восстановления.

Если база уже существует, перед восстановлением удалите её командой `DROP DATABASE`.

Мониторинг активности портов.

В Windows.

Чтобы узнать, на каком порту слушает SQL Server:

1. **Через SQL Server Configuration Manager.** Откройте его, перейдите в «SQL Server Network Configuration» → «Протоколы для <имя экземпляра>». Щёлкните правой кнопкой мыши на TCP/IP, выберите «Свойства». В разделе «IP Addresses» найдите группу `IPAll` — там будет указан порт в поле `TCP Port` или `TCP Dynamic Ports`.

2. **Через Event Viewer.** Фильтруйте события по ID 26022 в журнале Application — там будет сообщение с номером порта.

3. **С помощью T-SQL.** Выполните запрос:

```
sql
SELECT local_tcp_port FROM sys.dm_exec_connections WHERE local_tcp_port IS NOT NULL;
```

В Linux

Чтобы узнать порт PostgreSQL:

1. **Проверьте файл конфигурации** `postgresql.conf`**.** Ищите строку `port = ...`.

2. **Посмотрите логи PostgreSQL.** Например, в `/var/log/postgresql/postgresql-14-main.log` можно найти сообщения вида `LOG: listening on IPv4 address "127.0.0.1", port 5432`.

3. **Используйте команду** `pg_lsclusters` (в Debian/Ubuntu). Она покажет информацию о кластерах, включая порты.

4. **С помощью** `netstat` **или** `ss`**.** Например:

```
bash
netstat -anv | grep postgres
```

Для мониторинга активности портов в реальном времени можно использовать инструменты вроде `nethogs`, `iftop` или `tcpdump` в Linux, а в Windows — Resource Monitor или сторонние утилиты.

Важно: регулярно проверяйте открытые порты и закрывайте ненужные для повышения безопасности системы.

ТЕМА 5. СЕРТИФИКАЦИЯ ИНФОРМАЦИОННЫХ СИСТЕМ.

Сертификация информационных систем — это процедура подтверждения соответствия системы установленным требованиям безопасности, качества и функциональности. Она позволяет убедиться, что информационная система (ИС) отвечает определённым стандартам, что особенно важно для защиты данных, обеспечения надёжности работы и повышения доверия пользователей. В рамках производственной практики изучение этой темы включает освоение нормативно-правовой базы, этапов процесса, видов сертификации, а также практических навыков работы с документами и процедурами.

Сертификация — процедура подтверждения соответствия продукции, услуг или системы установленным требованиям. В случае информационных систем это означает проверку их соответствия стандартам безопасности, функциональности и качества.

Сертификат соответствия — документ, выдаваемый по результатам успешной сертификации, который подтверждает, что система соответствует установленным требованиям.

Орган по сертификации — организация, уполномоченная проводить сертификацию. Может самостоятельно проводить испытания или осуществлять надзор за этой деятельностью, выполняемой по его поручению другими органами.

Испытательная лаборатория — организация, проводящая сертификационные испытания и оформляющая технические заключения и протоколы.

Цели:

- подтверждение соответствия системы установленным стандартам;
- обеспечение безопасности информации;
- повышение доверия пользователей и заказчиков;
- формализация качества предоставляемых услуг;
- создание условий для деятельности организации на едином товарном рынке и в международном сотрудничестве.

Задачи:

- оценка соответствия системы требованиям;
- выявление уязвимостей;

- проверка работоспособности и надёжности;
- документирование результатов;
- обеспечение возможности контроля за соблюдением требований в будущем.

Виды сертификации.

1. **Обязательная.** Проводится для систем, обрабатывающих конфиденциальную информацию, государственных информационных систем, объектов критической информационной инфраструктуры (КИИ), информационных систем персональных данных и в других случаях, предусмотренных законодательством. Например, обязательна сертификация средств защиты информации, используемых для защиты сведений, составляющих государственную тайну или относимых к охраняемой информации ограниченного доступа.
2. **Добровольная.** Применяется для повышения конкурентоспособности системы, расширения сферы использования и получения экономических преимуществ. Может проводиться по отраслевым или фирменным стандартам.
3. **Сертификация по международным стандартам** (например, ISO 27001 для систем менеджмента информационной безопасности). Подтверждает соответствие системы международным требованиям к защите информации.

Нормативная база

В России сертификация регулируется:

- Федеральным законом «О техническом регулировании» (№184-ФЗ);
- Федеральным законом «Об информации, информационных технологиях и о защите информации» (№149-ФЗ);
- приказами ФСТЭК России (например, Приказ от 3 апреля 2018 г. №55 «Об утверждении Положения о системе сертификации средств защиты информации»);
- ГОСТами и другими нормативными документами.

Также применяются международные стандарты (ISO, IEC).

Этапы процесса сертификации

1. **Подготовка.** Сбор документации, анализ требований, оценка готовности системы к сертификации.

2. **Оценка соответствия.** Тестирование системы, анализ документации, проверка на соответствие стандартам.

3. **Оформление результатов.** Подготовка протоколов испытаний, принятие решения о выдаче сертификата.

4. **Выдача сертификата и регистрация.** При положительном результате заявитель получает сертификат соответствия, который регистрируется в соответствующем реестре.

5. **Инспекционный контроль.** Периодическая проверка сертифицированной системы на соответствие требованиям в течение срока действия сертификата.

Объекты сертификации

К объектам сертификации информационных систем относятся:

- программное обеспечение;
- аппаратные средства;
- документация;
- процессы разработки и эксплуатации;
- системы защиты информации;
- инфраструктура (серверы, сети и т. д.).

Практические аспекты для производственной практики

В рамках производственной практики студенты могут освоить:

- работу с нормативной документацией (законы, приказы, ГОСТы, международные стандарты);
- проведение тестирования информационных систем на соответствие требованиям;
- анализ результатов испытаний и подготовку отчётов;
- оформление документов для сертификации (заявки, протоколы, сертификаты);
- взаимодействие с органами по сертификации и испытательными лабораториями;
- оценку рисков и угроз при сертификации;
- применение международных стандартов (например, ISO 27001);

- особенности сертификации специфических систем (государственных, финансовых, систем обработки персональных данных и т. д.).

Проблемы и решения

Типичные проблемы:

- несоответствие системы требованиям;
- недостаточная документация;
- технические сбои при испытаниях;
- организационные сложности.

Пути решения:

- тщательная подготовка системы и документации до начала сертификации;
- регулярный аудит системы на соответствие требованиям;
- привлечение квалифицированных специалистов или консультантов;
- использование современных инструментов тестирования и анализа уязвимостей.

Перспективы развития

Направления развития в сфере сертификации информационных систем:

- внедрение новых стандартов и технологий;
- развитие методов тестирования и анализа рисков;
- автоматизация процессов сертификации;
- совершенствование методик оценки соответствия.

Таким образом, сертификация информационных систем — комплексный процесс, требующий знаний нормативно-правовой базы, технических аспектов и практических навыков. В производственной практике важно научиться применять эти знания для обеспечения безопасности и качества информационных систем.

Задание 5.1. Оформление требований и разработка технического задания по сертификации информационной системы (базы данных).

Оформление требований и разработка технического задания (ТЗ) для сертификации информационной системы (ИС), включая базу данных, требуют учёта нормативных требований, специфики системы и целей сертификации. Основные документы и подходы едины для Windows и Linux, но могут различаться детали реализации технических решений.

Нормативная база

При разработке требований и ТЗ необходимо опираться на следующие документы:

- **Приказ ФСТЭК России от 2 июня 2020 г. №76** — устанавливает уровни доверия к средствам технической защиты информации и средствам обеспечения безопасности информационных технологий.
- **Приказ ФСТЭК России от 3 апреля 2018 г. №55** — регулирует систему сертификации средств защиты информации.
- **Приказ ФСТЭК России от 11 февраля 2013 г. №17** — требования о защите информации в государственных информационных системах.
- **Федеральный закон от 27 июля 2006 г. №152-ФЗ «О персональных данных»** — если ИС обрабатывает персональные данные.
- **Федеральный закон от 26 июля 2017 г. №187-ФЗ «О безопасности критической информационной инфраструктуры РФ»** — если ИС относится к КИИ.
- **ГОСТ 34.602 «Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы»** — базовый стандарт для разработки ТЗ.
- **ГОСТ Р 56939–2024 «Защита информации. Разработка безопасного программного обеспечения. Общие требования»** — применим при сертификации серийного производства.

Также могут использоваться профили защиты ФСТЭК — набор требований для конкретных типов сертифицируемых изделий (операционных систем, средств доверенной загрузки и т. д.).

Этапы подготовки к сертификации

1. **Определение схемы сертификации**:
   - единичный образец;
   - партия;
   - серийное производство.
2. **Определение уровня доверия** в соответствии с Приказами ФСТЭК.
3. **Анализ угроз безопасности информации**. Используются данные из банка угроз ФСТЭК (bdu.fstec.ru).

4. **Разработка комплекта документов**, включая ТЗ, технические условия, формуляр, руководство пользователя и администратора.

Требования, которые должны быть отражены в документации

- **Проектирование архитектуры безопасности**:
  - описание безопасности процесса инициализации системы;
  - обеспечение собственной защиты от несанкционированного доступа;
  - невозможность обхода функций безопасности.
- **Функциональная спецификация**:
  - назначение и способы использования интерфейсов функций безопасности;
  - идентификация параметров, связанных с интерфейсами;
  - идентификация интерфейсов, не влияющих на функции безопасности.
- **Проектирование системы**:
  - перечень подсистем, реализующих функции безопасности;
  - подсистем, поддерживающих выполнение функций безопасности;
  - подсистем, не влияющих на выполнение функций безопасности.
- **Управление конфигурацией**:
  - уникальная маркировка системы;
  - элементы конфигурации, включая документацию;
  - порядок управления изменениями системы и документации.

Для баз данных дополнительно могут требоваться требования к контролю целостности данных, шифрованию, разграничению доступа к объектам БД, аудиту операций с данными.

Структура технического задания

ТЗ разрабатывается с учётом ГОСТ 34.602 и может включать следующие разделы:

1. **Общие положения** — цели создания системы, её назначение.
2. **Характеристики объекта автоматизации** — описание базы данных, её структура, объём, используемые технологии.
3. **Требования к системе** — функциональные, надёжности, безопасности, к составу технических средств, информационной и программной совместимости.

4. **Стадии и этапы разработки** — график работ.

5. **Порядок контроля и приёмки** — критерии оценки соответствия требованиям.

6. **Приложения** — схемы, таблицы, расчёты и другие материалы.

В разделе требований к безопасности информации должны быть указаны:

- класс защищённости ИС;
- перечень нормативных актов, стандартов, которым должна соответствовать система;
- объекты защиты;
- требования к мерам и средствам защиты информации;
- требования к защите при взаимодействии с другими системами и сетями.

Особенности для Windows и Linux

Критерий	Windows	Linux
Средства защиты	Часто используются сертифицированные СЗИ от НСД (например, Dallas Lock для Windows), средства доверенной загрузки, антивирусные системы	Применяются СЗИ, совместимые с Linux (например, Dallas Lock Linux), системы контроля целостности, средства шифрования
Шифрование	Может использоваться встроенное шифрование (BitLocker) или сторонние решения	Часто применяются инструменты вроде LUKS для шифрования дисков, OpenSSL для шифрования данных
Контроль доступа	Реализуется через Active Directory, групповые политики	Используются механизмы управления доступом на уровне ОС (ACL), SELinux/AppArmor для

Критерий	Windows	Linux
		усиленной защиты
Аудит и журналирование	События регистрируются в Event Log	Используются системные журналы (например, через journalctl), инструменты вроде auditd

Дополнительные рекомендации

- При сертификации серийного производства убедитесь, что процесс разработки соответствует ГОСТ Р 56939–2024.
- Для средств, использующих криптографию, потребуется сертификация ФСБ России.
- Подготовьте программу и методику сертификационных испытаний, которые должны включать описание системы, методы испытаний, применяемые средства.
- Учитывайте требования к защите информации при информационном взаимодействии с другими системами и сетями.

Перед началом работ рекомендуется проконсультироваться с органами по сертификации (ФСТЭК, ФСБ) и испытательными лабораториями, чтобы уточнить актуальные требования и порядок проведения процедур.

Задание 5.2. Выбор сертификатов. Сроки их действия.

Выбор сертификатов и управление их сроками действия зависят от контекста использования: это может быть SSL/TLS-сертификаты для веб-серверов, сертификаты электронной подписи, сертификаты для аутентификации в системах и т. д. Рассмотрим ключевые аспекты для Windows и Linux.

Выбор сертификатов.

Критерии выбора:

- **Назначение.** Определите, для чего нужен сертификат: защита веб-трафика, электронная подпись, аутентификация в сети и т. д. Это определит требования к типу сертификата и центру сертификации (CA).
- **Центр сертификации (CA).** Выбирайте доверенные CA, чьи сертификаты включены в стандартные хранилища браузеров и ОС. Для публичных

сайтов часто используют Let’s Encrypt, для корпоративных решений — внутренние или сторонние CA (например, Microsoft AD CS, OpenXPKI).

- **Тип сертификата.** Например, для веб-серверов важны:
  - **OV (Organization Validated)** — подтверждает существование организации.
  - **EV (Extended Validation)** — требует более строгой проверки организации.
- **Поддержка алгоритмов и протоколов.** Убедитесь, что сертификат совместим с используемыми криптографическими алгоритмами (например, RSA, ECC) и версиями TLS/SSL.
- **Срок действия.** Учитывайте политику CA относительно длительности действия сертификатов. Например, Let’s Encrypt планирует сократить срок действия сертификатов до 45 дней.

Особенности для Windows:

- Для интеграции с Active Directory и корпоративными системами часто используют сертификаты, выданные внутренним CA (например, через Microsoft AD CS).
- Сертификаты для электронной подписи и аутентификации могут требовать совместимости с конкретными криптопровайдерами (например, КриптоПро CSP).

Особенности для Linux:

- При работе с открытыми CA (например, Let’s Encrypt) убедитесь, что система поддерживает автоматизацию через ACME-клиенты (certbot, acme.sh).
- Для корпоративных решений могут потребоваться сертификаты, совместимые с конкретными дистрибутивами и сервисами (например, Apache, Nginx).

Сроки действия сертификатов.

Стандартные сроки:

- SSL/TLS-сертификаты обычно действуют от 90 дней до 1 года, но некоторые CA предлагают более длительные сроки.

- Сертификаты электронной подписи и аутентификации могут иметь срок действия от 1 до 3 лет в зависимости от политики организации и требований регулятора.

Как проверить срок действия:

- **В Windows.** Используйте оснастку **certmgr.msc** (Управление сертификатами) или командлет `Get-ChildItem` в PowerShell для просмотра сертификатов в хранилищах. В свойствах сертификата можно увидеть даты начала и окончания действия.
- **В Linux.** Используйте утилиту **OpenSSL**:

```
bash
openssl x509 -in cert.pem -text -noout
```

или для проверки удалённого сервера:

```
bash
echo | openssl s_client -connect example.com:443 2>/dev/null | openssl x509 -noout -dates
```

Автоматизация мониторинга:

- В Windows можно использовать задачи планировщика заданий или сторонние решения для отправки уведомлений о приближающемся истечении срока действия.
- В Linux удобно применять скрипты с OpenSSL, интегрированные в системы мониторинга (например, Zabbix, Prometheus) или cron-задачи. Например, скрипт может проверять срок действия и отправлять email-уведомление за 30 дней до окончания.

Рекомендации по управлению

Аспект	Рекомендации
Резервное копирование	Сохраняйте копии приватных ключей и сертификатов в безопасном месте. Для CA регулярно создавайте резервные копии базы данных сертификатов.
Обновление	Настройте автоматическую ротацию сертификатов

Аспект	Рекомендации
	(например, через certbot для Let’s Encrypt) или регулярные задачи для их продления.
Отзыв сертификатов	Используйте CRL (Certificate Revocation List) или OCSP (Online Certificate Status Protocol) для отзыва скомпрометированных сертификатов.
Безопасность	Храните приватные ключи в защищённых хранилищах (например, в аппаратных модулях — HSM). Ограничьте доступ к ним.
Тестирование	Перед полным развёртыванием новых сертификатов проводите тестирование в изолированной среде.

Выбор сертификатов зависит от их назначения, требований к безопасности и интеграции с инфраструктурой. Регулярно проверяйте сроки действия сертификатов и настройте автоматизацию для их обновления и мониторинга. В корпоративных средах используйте централизованные системы управления сертификатами (PKI), чтобы упростить администрирование и повысить безопасность.

При использовании на предприятии практики баз данных: анализ используемой базы данных; выявление необходимости изменения ее структуры; формирование технического задания на изменение структуры БД; доработка базы данных в соответствии с техническим заданием; решение сопутствующих вопросов, связанных с администрированием БД и серверов предприятия.

При отсутствии на предприятии практики баз данных: анализ предметной области; формирование технического задания на проектирование БД; разработка базы данных в соответствии с техническим заданием; решение сопутствующих вопросов, связанных с администрированием БД.

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 ПРОИЗВОДСТВЕННОЙ 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 ПП

ЗАДАНИЕ

на производственную 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 производственной практики от

филиала КузГТУ в г.Белово /_____/_____

подпись

ПРИЛОЖЕНИЕ В. Дневник по ПП

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 ПРОИЗВОДСТВЕННОЙ 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 практики от

ОРГАНИЗАЦИИ /_____/_____

подпись

Закончил практику «___» _____ 20__.

Дата	Содержание работы	Оценка	Подпись

ПРИЛОЖЕНИЕ Г. Бланк аттестационного листа по ПП

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 производственной 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 ПП

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 производственной практики

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 самостоятельно	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 производственной практике УП.07.01
для студентов очной формы обучения
по направлению специальности
09.02.07 «Информационные системы и программирование»

Публикуется в авторской редакции