МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Астраханский государственный университет имени В.Н. Татищева»
(Астраханский государственный университет им. В.Н. Татищева)

Филиал АГУ им. В.Н. Татищева в г. Знаменске Астраханской области

СОГЛАСОВАНО
Руководитель ОПОП
Бориско С.Н.
«13» ноября 2025 г.

УТВЕРЖДАЮ
Председатель ЦК (МО)
Фисенко Т.Ю.
протокол заседания ЦК (МО) №3
от «13» ноября 2025 г.

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ
по учебной дисциплине

Информационные технологии в профессиональной деятельности

Составитель	Бориско С.Н., к.т.н., доцент, завкафедрой ЗнМИ; Мустафаев Н.Г., к.т.н., доцент кафедры ЗнМИ; Тимошкин А.А., к.т.н., доцент кафедры ЗнМИ; Устинов А.С., к.т.н., доцент кафедры ЗнМИ; Каштанов Д.Ю., ассистент кафедры ЗнМИ
Согласовано с работодателями	Литвинов С.П., к.т.н., заместитель командира войсковой части 15644 по научно-исследовательской и испытательной работе; Кирьянов М.Н., ведущий инженер ПАО «Ростелеком»
Наименование специальности	09.02.12 Техническая эксплуатация и сопровождение информационных систем
Квалификация выпускника	Специалист по технической эксплуатации и сопровождению информационных систем
Форма обучения	очная
Год приема (курс)	2026 (2 курс)

Знаменск, 2025 г.

СОДЕРЖАНИЕ

1. Общие положения

Фонд оценочных средств (далее – ФОС) предназначен для контроля и оценки результатов освоения обучающимися учебной дисциплины «Информационные технологии в профессиональной деятельности».

ФОС включают контрольные материалы для проведения текущего контроля успеваемости и промежуточной аттестации обучающихся, разработанные в соответствии с требованиями ФГОС СПО и содержанием рабочей программы учебной дисциплины.

2. Результаты освоения учебной дисциплины, подлежащие проверке

Код компетенции	Планируемые результаты освоения учебной дисциплины		
	Практический опыт	Умения	Знания
ОК 2	-способен применять теоретические знания на практике при работе с различными операционными системами; -умеет анализировать и решать задачи системного администрирования - готов к освоению новых технологий в области операционных систем и сред.	-определять задачи для поиска информации, планировать процесс поиска, выбирать необходимые источники информации; -применять средства информационных технологий для решения профессиональных задач; -использовать современное программное обеспечение в профессиональной деятельности.	-определять задачи для поиска информации, планировать процесс поиска, выбирать необходимые источники информации; -применять средства информационных технологий для решения профессиональных задач; -использовать современное программное обеспечение в профессиональной деятельности.

3. Распределение оценивания результатов обучения по видам контроля

Наименование элемента практического опыта, умений или знаний	Наименование оценочного средства текущего контроля и промежуточной аттестации	
	Текущий контроль	Промежуточная аттестация
ПО.1. способен применять теоретические знания на практике при работе с различными операционными системами ПО.2. умеет анализировать и решать задачи системного администрирования; ПО.3 готов к освоению новых технологий в области операционных систем и сред	Компьютерное тестирование на знание терминологии по теме. Контрольные задания, решение задач по теме	Вопросы к экзамену
У1.определять задачи для поиска информации, планировать процесс поиска,		

выбирать необходимые источники информации; У2. применять средства информационных технологий для решения профессиональных задач; У3. использовать современное программное обеспечение в профессиональной деятельности.		
З1.определять задачи для поиска информации, планировать процесс поиска, выбирать необходимые источники информации З2. применять средства информационных технологий для решения профессиональных задач; З3. использовать современное программное обеспечение в профессиональной деятельности.		

4. Контрольные задания для оценки результатов освоения учебной дисциплины

4.1. Контрольные задания для текущего контроля

Раздел 1. Современные педагогические технологии в учебном процессе начальной школы

Тестовые задания

Вопросы для устного ответа

1. Технологизация педагогического процесса в современном начальном образовании.
2. Классификации педагогических технологий.
3. Интерактивность как основа инновационных технологий образования.
4. Понятие «здоровьесберегающая технология».
5. Здоровьесберегающие технологии, их функции, виды, основополагающие принципы и компоненты.
6. Классификация здоровьесберегающих технологий.

Раздел 2. Современные педагогические технологии в воспитательной и внеурочной деятельности

Тестовые задания

1. (ОВ) Инструмент на основе ИИ, который помогает программисту писать, дополнять и объяснять код, — это:
a) Visual Studio Code
b) GitHub Copilot
c) Docker
d) Postman

2. (МВ) Какие из перечисленных задач можно эффективно решать с помощью современных AI-ассистентов для программистов?
a) Генерация шаблонного кода (boilerplate)
b) Написание документации по коду

c) Поиск сложных логических ошибок (багов) в алгоритмах
d) Рефакторинг и оптимизация существующего кода
e) Полная автономная разработка сложного приложения

3. (CO) Процесс, при котором разработчик пишет описание желаемой функции на естественном языке, а ИИ-инструмент генерирует соответствующий код, называется _____ программирование.
Ответ: _____

4. (OB) Главный риск бездумного использования ИИ-генераторов кода заключается в:
a) Увеличении скорости разработки
b) Появлении скрытых уязвимостей или некорректной логики, которую разработчик может не заметить
c) Необходимости платить за подписку
d) Том, что ИИ заменит всех программистов

5. (OB) Система контроля версий Git является:
a) Централизованной (клиент-серверной)
b) Распределенной (каждая копия репозитория полная)
c) Только облачной (работает исключительно через GitHub)
d) Файловым менеджером для кода

6. (MB) Какие из перечисленных практик являются частью успешной workflow в Git?
a) Использование осмысленных сообщений коммитов (commit messages)
b) Ведение короткоживущих feature-веток
c) Прямая работа в ветке main/master
d) Регулярные слияния (merge) или перебазирования (rebase) с основной веткой

7. (CO) Команда Git, которая «связывает» локальный репозиторий с удаленным (например, на GitHub), называется _____.
Ответ (команда): _____

8. (CO) Язык разметки, используемый для создания документации с простым синтаксисом (например, файлы README.md), называется _____.
Ответ: _____

9. (MB) Какие из следующих сервисов относятся к категории «Облако как услуга» (XaaS)?
a) IaaS (Инфраструктура как услуга, напр., AWS EC2, Yandex Cloud Compute)
b) PaaS (Платформа как услуга, напр., Heroku, Yandex Cloud Functions)
c) SaaS (ПО как услуга, напр., Google Docs, Notion)
d) FaaS (Функция как услуга, напр., AWS Lambda, Cloud Functions)

10. (OB) Инструмент, который позволяет «упаковать» приложение и все его зависимости в изолированный контейнер для одинакового запуска в любой среде, — это:
a) Git
b) Docker
c) Kubernetes
d) Jenkins

11. (CO) Практика автоматизации процессов сборки, тестирования и развертывания приложений называется _____ (CI/CD).
Ответ (полное название на английском): _____

12. (OB) Сервис, который предоставляет готовую среду для хостинга Git-репозиториев, код-ревью и CI/CD (например, GitHub, GitLab, Bitbucket), называется:
a) Облачное хранилище
b) Система управления репозиториями
c) Платформа для удаленной работы
d) Файловый хостинг

13. (MB) Какие инструменты входят в типичный стек современного fullstack-разработчика?
a) Редактор кода/IDE (VS Code, IntelliJ IDEA)
b) Менеджер пакетов (npm, pip, Maven)
c) Инструменты для работы с API (Postman, Insomnia)
d) Графический редактор (Adobe Photoshop)

14. (CO) Онлайн-платформа, где разработчики могут публиковать фрагменты кода, демонстрировать проекты и создавать цифровое портфолио, например, CodePen или JSFiddle, называется _____.
Ответ (общий термин): _____

15. (OB) Платформа типа «low-code/no-code» (например, Bubble, Airtable) позволяет:
a) Создавать приложения с минимальным написанием традиционного кода, используя визуальные конструкторы
b) Писать код только на ассемблере
c) Автоматически исправлять все баги в программе
d) Заменять собой все языки программирования

16. (OB) Практика «цифровой гигиены» для разработчика НЕ включает:
a) Использование менеджера паролей и двухфакторной аутентификации (2FA)
b) Регулярное обновление ПО и зависимостей (dependencies)
c) Хранение секретов (API-ключей, паролей) прямо в коде репозитория
d) Использование VPN при работе в публичных сетях Wi-Fi

17. (MB) Какие из перечисленных действий повышают безопасность исходного кода и проекта?
a) Регулярный аудит зависимостей на наличие известных уязвимостей
b) Использование .gitignore для исключения конфиденциальных файлов из репозитория
c) Настройка секретов (secrets) в переменных окружения или специальных vault-хранилищах CI/CD
d) Отключение проверки подписей коммитов (GPG signatures)

18. (CO) Тип вредоносного ПО, который блокирует доступ к данным или системе и требует выкуп за восстановление, называется _____.
Ответ: _____

19. (CO) Принцип безопасности, согласно которому пользователь или процесс должны иметь только те минимальные права доступа, которые необходимы для выполнения их задач, называется принципом _____.
Ответ: _____

20. (ОВ) Уязвимость, возникающая при неправильной обработке пользовательского ввода, который позволяет злоумышленнику внедрить и выполнить вредоносный код на стороне сервера, — это:
a) Инъекция (Injection), например, SQL- или Command-инъекция
b) XSS (Межсайтовый скриптинг)
c) CSRF (Межсайтовая подделка запроса)
d) Фишинг (Phishing)

Ключ для проверки

Раздел 1:

1. **b)** GitHub Copilot (или аналоги: Tabnine, Codeium, Amazon CodeWhisperer)
2. **a, b, d** (c — пока с этим ИИ справляется хуже человека; e — пока нереалистично для сложных проектов)
3. **генеративное** / **естественно-языковое** / **prompt-based** (промптовое)
4. **b)** Появлении скрытых уязвимостей или некорректной логики, которую разработчик может не заметить

Раздел 2:
5. **b)** Распределенной (каждая копия репозитория полная)
6. **a, b, d** (c — плохая практика, ведущая к хаосу)
7. git remote add
8. **Markdown**

Раздел 3:
9. **a, b, c, d** (Все относятся к облачным сервисам по модели XaaS)
10. **b)** Docker
11. **Continuous Integration and Continuous Delivery/Deployment**
12. **b)** Система управления репозиториями (или Git-хостинг)

Раздел 4:
13. **a, b, c** (d — инструмент дизайнера, не обязательный в стеке разработчика)
14. **песочница для кода** / **code playground** / **online editor**
15. **a)** Создавать приложения с минимальным написанием традиционного кода, используя визуальные конструкторы

Раздел 5:
16. **c)** Хранение секретов (API-ключей, паролей) прямо в коде репозитория (это грубое нарушение безопасности)
17. **a, b, c** (d — ослабляет безопасность)
18. **программа-вымогатель** / **ransomware**
19. **наименьших привилегий** (least privilege)
20. **a)** Инъекция (Injection), например, SQL- или Command-инъекция

Контрольные задания

Блок 1: ИИ-ассистенты в разработке (GitHub Copilot / ChatGPT)

Задание 1. «Промпт-инженерия для генерации кода»
Цель: Научиться эффективно формулировать запросы (промпты) для ИИ-ассистента.

Сценарий: Вам нужно создать функцию на Python для веб-скрапинга (парсинга).

1. **Неэффективный промпт:** «Напиши код для парсинга».
   - Сгенерируйте код по этому запросу и проанализируйте проблемы (неконкретность, отсутствие контекста).
2. **Эффективный промпт:** Создайте детализированный промпт, включающий:
   - **Контекст:** «Я разрабатываю инструмент для анализа цен конкурентов».
   - **Задача:** «Напиши функцию на Python с именем `parse_product_data`».
   - **Технические требования:** «Функция должна принимать URL страницы товара. Использовать библиотеки `requests` и `BeautifulSoup`. Извлекать: название товара (тег `h1`), цену (класс `.price`), наличие на складе (текст "В наличии"). Обрабатывать ошибки соединения и отсутствия тегов. Возвращать словарь или `None` в случае ошибки».
   - **Стиль:** «Код должен быть документирован в формате docstring, следовать PEP8». Запрос должен умещаться в 3-4 предложения.
3. **Анализ и рефакторинг:**
   - Возьмите сгенерированный ИИ код.
   - Проверьте его на наличие потенциальных проблем: безопасность (user-agent, таймауты), обработку edge-кейсов (изменение структуры сайта).
   - Напишите 2 3 unit-теста с использованием `pytest` для проверки функции.

Задание 2. «Код-ревью с ИИ»
Цель: Использовать ИИ для анализа и улучшения существующего кода.

Дан фрагмент неоптимального JavaScript кода:

```javascript
function findDuplicates(arr) {
    let result = [];
    for (let i = 0; i < arr.length; i++) {
        for (let j = 0; j < arr.length; j++) {
            if (i !== j && arr[i] === arr[j] && !result.includes(arr[i])) {
                result.push(arr[i]);
            }
        }
    }
    return result;
}
```

Ваши действия:

1. **Анализ:** Сформулируйте промпт для ИИ (ChatGPT, Claude) с просьбой провести **код-ревью**. Укажите, что нужно: найти узкие места по производительности, предложить более оптимальные решения, оценить читаемость.

2. **Оптимизация:** На основе ответа ИИ и своих знаний перепишите функцию. Объясните, почему новая реализация лучше (сложность O(n) vs O(n^2), использование Set или Map).
3. **Бенчмарк:** Напишите небольшой скрипт, который замеряет время выполнения обеих функций на массиве из 10 000 случайных чисел. Представьте результаты.

Блок 2: Git, CI/CD и командная работа

Задание 3. «Моделирование Git Workflow для исправления критического бага»
Ситуация: В production-ветке (main) обнаружен критический баг. Ветка develop уже содержит много нового кода для следующего релиза.

Требуется:

1. **Создайте план действий** (Git-команды) для команды из 3-х человек:

- **Разработчик А:** Создает ветку hotfix/auth-bug от main, исправляет баг.
- **Разработчик Б:** Проводит код-ревью через Pull Request (PR) и мержит hotfix в main.
- **Разработчик В:** Переносит (cherry-pick) это исправление в ветку develop.

2. **Смоделируйте историю коммитов** с помощью git log --graph --oneline --all. Используйте учебный репозиторий или нарисуйте схему.
3. **Настройте GitHub Actions workflow** (файл .github/workflows/ci.yml), который для hotfix-веток будет:

- Запускать тесты.
- Проверять линтером (например, flake8 для Python или ESLint для JS).
- Собирать проект (если применимо).
- **Не** деплоить в прод (деплой только из main после успешного мержа).

Задание 4. «Разработка и автоматизация публикации документации»
Цель: Настроить автоматическое обновление документации при каждом обновлении main.

Шаги:

1. Создайте в репозитории папку /docs. Напишите в ней файл API.md на Markdown с описанием 2-3 гипотетических API-эндпоинтов вашего проекта.
2. Создайте конфигурационный файл для **MkDocs** или **Docusaurus** (на выбор), чтобы сгенерировать статический сайт из Markdown.
3. Настройте **GitHub Pages** для хостинга этой документации.
4. Напишите **GitHub Actions workflow**, который при пуше в main:

- Устанавливает зависимости для генератора документации.
- Собирает статический сайт.
- Публикует его на GitHub Pages.
- (Дополнительно) Отправляет уведомление в Slack-канал команды о успешном обновлении.

Блок 3: Облачная инфраструктура и контейнеризация

Задание 5. «Dockerize и деплой микросервиса»
Задача: Упаковать простое веб-приложение (например, на Flask/Express) и развернуть его в облаке.

Часть 1: Контейнеризация

1. Напишите `Dockerfile` для вашего приложения со следующими требованиями:
   - Многоэтапная сборка (multi-stage build).
   - Запуск от непривилегированного пользователя.
   - Использование `.dockerignore`.
   - Открытие нужного порта (например, 8080).
2. Соберите образ и запустите контейнер локально, проверив его работу.

Часть 2: Облачный деплой

1. Выберите облачный провайдер (Yandex Cloud, AWS, GCP, VPS).
2. Напишите `docker-compose.yml` файл, который поднимает ваше приложение и, например, базу данных PostgreSQL.
3. Настройте **обратный прокси (nginx)** в отдельном контейнере, который будет перенаправлять запросы с 80 порта на ваш контейнер с приложением.
4. Опишите шаги для деплоя этого стека на облачный сервер (включая настройку SSH, копирование файлов, запуск `docker-compose up`).

Задание 6. «Создание бессерверной функции (FaaS)»
Цель: Реализовать и развернуть функцию, реагирующую на событие.

Сценарий: Функция должна обрабатывать загрузку нового изображения в облачное хранилище (например, Yandex Object Storage), создавать его thumbnail и сохранять обратно.

План:

1. **Логика функции:** Напишите код на Python/Node.js, который:
   - Триггерится событием создания объекта в бакете `origin-images`.
   - Скачивает изображение.
   - Изменяет его размер (библиотека Pillow/Sharp).
   - Загружает результат в бакет `processed-thumbnails`.
2. **Локальная отладка:** Создайте тестовое событие в формате провайдера и протестируйте функцию локально.
3. **Деплой:** Разверните функцию в выбранном FaaS (Yandex Cloud Functions, AWS Lambda, Google Cloud Functions).
4. **Настройка:** Сконфигурируйте триггер на событие хранилища и права доступа (IAM) для функции.

Блок 4: Кибербезопасность и цифровая гигиена

Задание 7. «Аудит безопасности небольшого проекта»
Цель: Проверить гипотетический стартап-проект на типовые уязвимости.

Дан репозиторий с уязвимым кодом (см. файлы ниже).

`app.py` **(Flask):**

```
python
from flask import Flask, request
import sqlite3
app = Flask(__name__)
```

```
@app.route('/login', methods=['POST'])
def login():
    username = request.form['username']
    password = request.form['password']
    conn = sqlite3.connect('users.db')
    cursor = conn.cursor()
    # УЯЗВИМОСТЬ 1
    query = f"SELECT * FROM users WHERE username='{username}' AND password='{passw
ord}'"
    cursor.execute(query)
    # ... проверка результата
```

package.json **фрагмент:**

```
json
"dependencies": {
    "express": "^4.16.0",
    "lodash": "4.17.10",
    "urllib": "2.34.0"
}
```

config.yaml **в репозитории:**

```
yaml
database:
  host: "prod-db.internal"
  password: "SuPeRSeCrEtPa$$" # УЯЗВИМОСТЬ 3
```

Ваша задача — составить отчет. Для каждого файла:

1. **Найти уязвимость** (например, SQL-инъекция, устаревшая зависимость с CVE, секрет в коде).
2. **Оценить критичность** (низкая, средняя, высокая).
3. **Предложить конкретное исправление** (например, использовать параметризованные запросы, обновить пакет, перенести секрет в переменные окружения и добавить файл в `.gitignore`).
4. **Рекомендовать инструмент** для автоматического поиска таких проблем (например, `git-secrets`, `trufflehog`, `npm audit`, `snyk`).

Задание 8. «Разработка политики цифровой гигиены для команды»
Разработайте практическое руководство (чек-лист) на 1 страницу для новой команды разработки.

Структура:

1. **Управление доступом:** Требования к паролям (менеджер паролей), обязательная 2FA для всех сервисов (GitHub, облако, админки).
2. **Безопасность репозитория:** Правила работы с секретами, настройка защищенных веток (branch protection), обязательный код-ревью перед мержем.
3. **Рабочая станция:** Настройка автообновлений ОС и ПО, использование VPN, блокировка экрана, шифрование диска.
4. **Инциденты:** Куда и как сообщать о подозрительном письме (фишинг) или утечке данных.

5. **Образование:** План ежегодного прохождения коротких курсов по безопасности (например, на Stepik или HTB Academy).

   Формат: Создайте документ в **Notion** или **Google Docs** с использованием четких заголовков, списков и, возможно, QR-кодов на внутренние ресурсы. Поделитесь ссылкой.

Критерии оценки:

- **Блок 1 (ИИ):** Оценивается качество и детализация промптов, глубина анализа сгенерированного кода, умение дополнять и исправлять работу ИИ.
- **Блок 2 (Git/CI-CD):** Оценивается правильность Git-стратегии, работоспособность и оптимальность конфигураций CI/CD, понимание принципов автоматизации.
- **Блок 3 (Облако):** Оценивается корректность Dockerfile и docker-compose, понимание принципов FaaS, реалистичность плана деплоя.
- **Блок 4 (Безопасность):** Оценивается точность поиска уязвимостей, практичность предложенных исправлений, полнота и ясность разработанной политики.
- **Общее:** Практическая применимость решений, использование современных инструментов и практик, структурированность ответов.

4.2 Контрольные задания для промежуточной аттестации

Вопросы для зачета

1. Дайте определение базы данных и СУБД. Каковы их основные функции и преимущества перед файловым хранением?
2. Опишите трёхуровневую архитектуру ANSI/SPARC. Каково назначение каждого уровня (внешнего, концептуального, внутреннего)?
3. Что такое модель данных? Сравните иерархическую, сетевую и реляционную модели.
4. Объясните основные понятия реляционной модели: отношение (таблица), атрибут (столбец), кортеж (строка), домен, ключ (первичный, внешний).
5. Опишите процесс нормализации баз данных. Каковы цели нормализации и проблемы ненормализованных отношений (аномалии)?
6. Что такое ER-диаграмма (Entity-Relationship)? Объясните сущности, атрибуты, связи (1:1, 1:M, M:N) и их графическое обозначение.
7. Сформулируйте основные правила целостности данных в реляционной модели: целостность сущностей и ссылочную целостность.
8. Что такое денормализация? В каких ситуациях она оправдана и каковы её риски?
9. Что такое SQL? На какие группы делятся операторы SQL (DDL, DML, DCL, TCL)? Приведите примеры команд для каждой группы.
10. Объясните разницу между операторами `DELETE`, `TRUNCATE` и `DROP TABLE`.
11. Что такое представление (VIEW)? Каковы его преимущества и ограничения (в контексте обновления данных через представление)?
12. Опишите назначение и принцип работы индексов в БД. Какие типы индексов вы знаете (B-дерево, хэш, полнотекстовый)?
13. Что такое транзакция в контексте БД? Сформулируйте и объясните свойства транзакций ACID.

14. Опишите проблема параллельного доступа к данным. Что такое «грязное» чтение, неповторяющееся чтение и фантомное чтение?
15. Что такое уровни изоляции транзакций? Опишите стандартные уровни (Read Uncommitted, Read Committed, Repeatable Read, Serializable) и решаемые ими проблемы.
16. Объясните назначение и использование оператора JOIN. В чём разница между INNER JOIN, LEFT/RIGHT OUTER JOIN и CROSS JOIN?
17. Что такое агрегатные функции в SQL? Приведите примеры использования COUNT, SUM, AVG, MIN, MAX вместе с оператором GROUP BY.
18. Объясните разницу между подзапросами (подселектами) и операциями соединения (JOIN). В каких случаях предпочтительнее каждый из подходов?

19. Каковы основные задачи администратора базы данных (DBA)? Разделите их на технические и эксплуатационные.
20. Что такое план выполнения запроса (execution plan)? Как администратор может использовать его для оптимизации производительности?
21. Опишите основные принципы и стратегии резервного копирования БД: полное, инкрементальное, дифференциальное копирование.
22. Что такое журнал транзакций (transaction log)? Какова его роль в обеспечении отказоустойчивости и восстановлении базы данных?
23. Объясните процесс восстановления базы данных после сбоя. В чём разница между восстановлением на момент времени (Point-in-Time Recovery) и восстановлением из резервной копии?
24. Что такое мониторинг производительности БД? Какие ключевые метрики (процессор, память, дисковый ввод-вывод, блокировки) необходимо отслеживать?
25. Опишите методы управления пользователями и правами доступа в СУБД. Что такое ролевая модель (RBAC) и как она упрощает администрирование?
26. Что такое миграция и обновление БД? Какие основные риски связаны с этим процессом и как их минимизировать?

27. Сформулируйте триаду информационной безопасности (CIA) применительно к базам данных. Дайте краткое пояснение каждому компоненту.
28. Объясните разницу между аутентификацией и авторизацией в контексте доступа к СУБД.
29. Что такое контроль доступа? Сравните дискреционный (DAC) и мандатный (MAC) механизмы управления доступом.
30. Опишите основные угрозы безопасности баз данных (SQL-инъекции, несанкционированный доступ, внутренние угрозы, DDoS).
31. Что такое SQL-инъекция (SQL Injection)? Объясните принцип атаки и основные методы защиты (параметризованные запросы, stored procedures, input validation).
32. Для чего предназначено аудирование (auditing) в БД? Какие события следует обязательно регистрировать в журналах аудита?
33. Опишите методы шифрования данных в БД. В чём разница между шифрованием на уровне столбцов, прозрачным шифрованием данных (TDE) и шифрованием на уровне приложения?
34. Что такое маскирование (обфускация) данных? В каких сценариях оно применяется (например, в тестовых средах)?
35. Объясните концепцию «безопасность по умолчанию» (security by default) и «принцип наименьших привилегий» (least privilege) применительно к настройке СУБД.
36. Каковы основные требования стандартов безопасности (например, PCI DSS, GDPR) к хранению и обработке персональных данных в базах данных?

37. Что такое обработка запросов в СУБД? Опишите основные этапы: парсинг, оптимизация, выполнение.

38. Какие факторы влияют на производительность запросов? Опишите роль индексов, статистики, фрагментации и аппаратных ресурсов.
39. Что такое блокировки (locks) в СУБД? Опишите разницу между пессимистичными и оптимистичными блокировками.
40. Что такое тупик (deadlock) в многопользовательской среде? Опишите классический пример «обедающих философов» и способы предотвращения или обнаружения взаимных блокировок.

Критерии оценки

Оценка «5» - (отлично)

При ответе материал изложен грамотным языком в определенной логической последовательности, точно использована терминология, полно раскрыто содержание материала в объеме, предусмотренном программой, продемонстрировано усвоение ранее изученных сопутствующих вопросов. Возможны одна - две неточности при освещении второстепенных вопросов.

Оценка «4» - (хорошо)

Ответ удовлетворяет в основном требованиям на оценку «5», но при этом имеет один из недостатков: в изложении допущены небольшие пробелы; допущены один – два недочета при освещении основного содержания ответа, исправленные по замечанию преподавателя; допущены ошибка или более двух недочетов при освещении второстепенных вопросов, легко исправленные по замечанию преподавателя.

Оценка «3» - (удовлетворительно)

При ответе неполно или непоследовательно раскрыто содержание материала, но показано общее понимание, имелись затруднения или допущены ошибки в определении понятий.

Оценка «2» - (неудовлетворительно)

При ответе не раскрыто основное содержание учебного материала; обнаружено незнание или непонимание обучающимся большей или наиболее важной части учебного материала; допущены ошибки в определении понятий, допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными умениями по данной теме в полной мере.