МИНИСТЕРСТВО ОБРАЗОВАНИЯ РОСТОВСКОЙ ОБЛАСТИ

ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССТОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ РОСТОВСКОЙ ОБЛАСТИ «РОСТОВСКИЙ-НА-ДОНУ КОЛЛЕДЖ СВЯЗИ И ИНФОРМАТИКИ»

ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

по профессиональному модулю

ПМ.02 Администрирование баз данных

по специальности

09.02.13 Интеграция решений с применением технологий искусственного интеллекта

(базовой подготовки)

г. Ростов-на-Дону, 2026 г.

РАССМОТРЕНО
на заседании цикловой комиссии
Программирование
Протокол № 1 от 31.08.2026 года
Председатель ЦК Сулавко С.Н.

УТВЕРЖДАЮ:
Зам. директора по УМР
И.В. Подцатова
«16» ноября 2026 г.

Комплект контрольно-оценочных средств по ПМ.02 «Администрирование баз данных» разработан в соответствии с рабочей программой по ПМ, разработанной в 2026 году по специальности 09.02.13 Интеграция решений с применением технологий искусственного интеллекта.

Организация-разработчик: государственное бюджетное профессиональное образовательное учреждение Ростовской области «Ростовский-на-Дону колледж связи и информатики» Разработчики:

Ухов П.Э., Малая М.А., Сулавко С.Н. преподаватели государственного бюджетного профессионального образовательного учреждения Ростовской области «Ростовский-на-Дону колледж связи и информатики».

СОДЕРЖАНИЕ

1. Формы промежуточной аттестации по профессиональному модулю.....4
2. Результаты освоения модуля, подлежащие проверке ……4
3. Оценка освоения теоретического курса профессионального модуля.....10
4. Контроль приобретения практического опыта. Оценка по учебной и (или) производственной практике.....382
5. Контрольно-оценочные материалы для квалификационного экзамена ……388

Общие положения

Результатом освоения профессионального модуля является готовность обучающегося к выполнению вида профессиональной деятельности (в соответствии с рабочей программой ПМ) и сформированность профессиональных и общих компетенций.

Формой аттестации по профессиональному модулю является экзамен по модулю.

1. Формы промежуточной аттестации по профессиональному модулю

Таблица 1

Элементы модуля, профессиональный модуль	Формы промежуточной аттестации
1	**2**
МДК 02.01 Управление и автоматизация баз данных	Дифференцированный зачет
МДК 02.02 Технология разработки и защиты баз данных	Экзамен
МДК 02.03 Конвейеры обработки и контроль качества данных для систем ИИ	Дифференцированный зачет
УП.02	Зачет
ПП.02	Зачет
ПМ 02	**Экзамен по модулю**

2. Результаты освоения модуля, подлежащие проверке

2.1. Профессиональные и общие компетенции

В результате контроля и оценки по профессиональному модулю осуществляется комплексная проверка следующих профессиональных и общих компетенций:

Таблица 2

Профессиональные компетенции	Показатели оценки результата
ПК 2.1	- Идентификация проблем, связанных с нормальным функционированием базы данных - Восстановление системы
ПК 2.2	- Администрирование сервера баз данных - Участие в администрировании отдельных компонентов серверов
ПК 2.3	- Документирование результатов аудита безопасности информации - Использование процедуры резервного копирования баз данных - Использование процедуры восстановления баз данных
ПК 2.4	- Подготовка документации по формированию требований к хранилищам банка данных
ПК 2.5	- Проектирование, разработка и эксплуатация баз

	данных
ПК 2.6	- Планирование процедур резервного копирования данных - Запуск процедуры резервного копирования данных - Мониторинг выполнения процедур резервного копирования данных - Контроль завершения процедуры резервного копирования данных - Проведение повторной процедуры резервного копирования данных в случае ее нештатного завершения - Хранение резервных копий БД
ПК 2.7	- Назначение приоритета запросов к БД - Оптимизация выполнения статистически значимых запросов к БД
ПК 2.8	- Создание программного кода в соответствии с техническим заданием (готовыми спецификациями) - Оптимизация программного кода с использованием специализированных программных средств
ПК 2.9	- Определение источников больших данных для анализа, идентификация внешних и внутренних источников данных для проведения аналитических работ - Получение и фильтрация больших объемов данных из гетерогенных источников - Извлечение, проверка и очистка больших объемов данных из гетерогенных источников - Агрегация и разработка представления больших объемов данных из гетерогенных источников - Оценка соответствия набора данных предметной области и задачам аналитических работ
ПК 2.10	- Оценка возможностей имеющейся у исполнителя методологической и технологической инфраструктуры анализа больших данных - Анализ предметной области - Применение теоретических и прикладных основ анализа больших данных - Применение современных методов и инструментальных средств анализа больших данных - Использование современного опыта анализа больших данных - Осуществление стратегического и оперативного руководства кадровыми и материальными ресурсами в процессах получения, размещения, хранения, распределения, миграции, архивирования и удаления данных - Осуществление стратегического и оперативного руководства материально-техническими ресурсами и элементами технологической инфраструктуры больших данных - Разработка моделей данных, адаптированных к технологиям больших данных - Интеграция больших данных с системами хранения данных организации - Создание и интеграция облачных репозиториев с системами хранения данных организации - Создание параллельных систем хранения и обработки

	информации - Разработка методов и регламентов реализации различных режимов обработки больших данных, в том числе режима реального времени - Мониторинг и оценка производительности обработки данных в организации, разработка предложений по повышению производительности обработки больших данных - Разработка методов и регламентов преобразования данных - Разработка и согласование стратегии формирования информационных активов организации - Разработка методов и регламентов оценки информационных активов организации - Формирование состава и координация деятельности персонала, отвечающего за приобретение и получение больших данных - Оценка эффективности системы хранения и обработки данных организации - Разработка предложений по развитию и совершенствованию системы получения, хранения, передачи, обработки больших данных
ПК 2.11	- Формирование целей, приоритетов и ограничений управления качеством данных, адаптация целей к изменениям внутренних и внешних условий, потребностей и возможностей, в том числе технологических - Разработка стратегии управления качеством больших данных - Выделение ресурсов для управления качеством больших данных - Реализация стратегии обеспечения качества больших данных - Анализ реализации стратегии обеспечения качества больших данных - Проведение оценки качества больших данных - Разрешение проблем и инцидентов с качеством больших данных - Оценка рисков использования больших данных

Таблица 3

Общие компетенции	**Показатели оценки результата**
ОК 01	- Выбор способов решения задач профессиональной деятельности применительно к различным контекстам
ОК 02	- Использование современных средств поиска, анализа и интерпретации информации, и информационные технологии
ОК 05	- Осуществление устной и письменной коммуникации на государственном языке Российской Федерации
ОК 09	- Использование профессиональной документацией на государственном и иностранном языках

2.2. Общие и (или) профессиональные компетенции, проверяемые дополнительно: _

2.3. Требования к портфолио

Тип портфолио: портфолио работ

Выберите из предложенных вариантов: портфолио документов, портфолио работ, портфолио смешанного типа.

Проверяемые результаты обучения:

Основные требования

Требования к структуре и оформлению портфолио:

1. В портфолио входят практические работы, входящие в МДК.02.01, МДК 02.02 и МДК 02.03 отзывы руководителей учебной и производственной практик
2. Практические работы оформлены в виде отчета.
3. Оформление отчета соответствует требованиям документа «Правила оформления текстовых документов в учебном процессе»

Показатели оценки портфолио

Таблица 4

Коды и наименования проверяемых компетенций или их сочетаний	**Показатели оценки результата**	**Оценка (да / нет)**
ОК 01	- Выбор способов решения задач профессиональной деятельности применительно к различным контекстам	
ОК 02	- Использование современных средств поиска, анализа и интерпретации информации, и информационные технологии	
ОК 05	- Осуществление устной и письменной коммуникации на государственном языке Российской Федерации	
ОК 09	- Использование профессиональной документацией на государственном и иностранном языках	
ПК 2.1	- Идентификация проблем, связанных с нормальным функционированием базы данных Восстановление системы	
ПК 2.2	- Администрирование сервера баз данных - Участие в администрировании отдельных компонентов серверов	
ПК 2.3	- Документирование результатов аудита безопасности информации - Использование процедуры резервного копирования баз данных	

	- Использование процедуры восстановления баз данных	
ПК 2.4	- Подготовка документации по формированию требований к хранилищам банка данных	
ПК 2.5	- Проектирование, разработка и эксплуатация баз данных	
ПК 2.6	- Планирование процедур резервного копирования данных - Запуск процедуры резервного копирования данных - Мониторинг выполнения процедур резервного копирования данных - Контроль завершения процедуры резервного копирования данных - Проведение повторной процедуры резервного копирования данных в случае ее нештатного завершения - Хранение резервных копий БД	
ПК 2.7	- Назначение приоритета запросов к БД - Оптимизация выполнения статистически значимых запросов к БД	
ПК 2.8	- Создание программного кода в соответствии с техническим заданием (готовыми спецификациями) - Оптимизация программного кода с использованием специализированных программных средств	
ПК 2.9	- Определение источников больших данных для анализа, идентификация внешних и внутренних источников данных для проведения аналитических работ - Получение и фильтрация больших объемов данных из гетерогенных источников - Извлечение, проверка и очистка больших объемов данных из гетерогенных источников - Агрегация и разработка представления больших объемов данных из гетерогенных источников - Оценка соответствия набора данных предметной области и задачам аналитических работ	
ПК 2.10	- Оценка возможностей имеющейся у исполнителя методологической и технологической инфраструктуры анализа больших данных - Анализ предметной области - Применение теоретических и прикладных основ анализа больших данных - Применение современных методов и инструментальных средств анализа больших данных - Использование современного опыта анализа больших данных - Осуществление стратегического и оперативного руководства кадровыми и материальными ресурсами в процессах получения, размещения, хранения, распределения, миграции, архивирования и удаления данных - Осуществление стратегического и оперативного руководства материально-техническими ресурсами и	

	элементами технологической инфраструктуры больших данных - Разработка моделей данных, адаптированных к технологиям больших данных - Интеграция больших данных с системами хранения данных организации - Создание и интеграция облачных репозиториев с системами хранения данных организации - Создание параллельных систем хранения и обработки информации - Разработка методов и регламентов реализации различных режимов обработки больших данных, в том числе режима реального времени - Мониторинг и оценка производительности обработки данных в организации, разработка предложений по повышению производительности обработки больших данных - Разработка методов и регламентов преобразования данных - Разработка и согласование стратегии формирования информационных активов организации - Разработка методов и регламентов оценки информационных активов организации - Формирование состава и координация деятельности персонала, отвечающего за приобретение и получение больших данных - Оценка эффективности системы хранения и обработки данных организации - Разработка предложений по развитию и совершенствованию системы получения, хранения, передачи, обработки больших данных	
ПК 2.11	- Формирование целей, приоритетов и ограничений управления качеством данных, адаптация целей к изменениям внутренних и внешних условий, потребностей и возможностей, в том числе технологических - Разработка стратегии управления качеством больших данных - Выделение ресурсов для управления качеством больших данных - Реализация стратегии обеспечения качества больших данных - Анализ реализации стратегии обеспечения качества больших данных - Проведение оценки качества больших данных - Разрешение проблем и инцидентов с качеством больших данных - Оценка рисков использования больших данных	

3. Оценка освоения теоретического курса профессионального модуля

3.1. Задания для оценки освоения МДК 02.01

Тема 1.1. Установка и настройка программного обеспечения для администрирования баз данных.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных
ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации
ПК 2.5. Подготавливать данные для базы знаний
ПК 2.6. Осуществлять управление базами данных
ПК 2.7. Осуществлять автоматизацию баз данных

Вопросы к устному опросу по теме:

1. Что такое программное обеспечение для администрирования баз данных?
Ответ: Программное обеспечение для администрирования баз данных — это совокупность программ, обеспечивающих создание, настройку, обслуживание и контроль работы баз данных. В него входят сервер системы управления базами данных (СУБД), клиентские утилиты командной строки, графические средства администрирования, средства резервного копирования, мониторинга и драйверы подключения.

2. Что такое СУБД и какие СУБД наиболее распространены?
Ответ: СУБД (система управления базами данных) — это комплекс программ для создания баз данных, хранения данных, доступа к ним и управления ими. Наиболее распространены реляционные СУБД MySQL, PostgreSQL, Oracle Database, Microsoft SQL Server, а также нереляционная документоориентированная СУБД MongoDB. В России также используются Postgres Pro и другие отечественные СУБД на основе PostgreSQL.

3. Какие клиентские средства администрирования баз данных вы знаете?
Ответ: Для работы из командной строки используются утилиты mysql, psql, sqlplus, sqlcmd и mongosh. К графическим средствам относятся MySQL Workbench, pgAdmin, Oracle SQL Developer, SQL Server Management Studio (SSMS), MongoDB Compass, а также универсальные клиенты DBeaver и DataGrip, поддерживающие множество СУБД.

4. Какие действия необходимо выполнить перед установкой СУБД?
Ответ: Перед установкой необходимо определить назначение сервера и требуемую версию СУБД, проверить соответствие компьютера системным требованиям (операционная система, процессор, оперативная память, место на диске), проверить совместимость с другим программным обеспечением, свободность сетевого порта, наличие прав администратора, загрузить дистрибутив из официального источника и спланировать размещение файлов данных.

5. Какие способы установки СУБД существуют?
Ответ: СУБД можно установить с помощью графического установщика (например, MySQL Installer или установщика SQL Server в Windows), через менеджер пакетов операционной

системы (apt, dnf, yum в Linux), из пакетов RPM и DEB, из исходного кода, в виде Docker-контейнера, а также использовать облачную управляемую базу данных.

6. Почему дистрибутивы СУБД следует загружать только из официальных источников?
Ответ: Дистрибутивы из неофициальных источников могут содержать вредоносный код, быть изменёнными или устаревшими. Официальные источники и репозитории обеспечивают подлинность и целостность пакетов, которая проверяется по цифровой подписи или контрольной сумме, а также своевременное получение обновлений безопасности.

7. Что такое служба (сервис) СУБД?
Ответ: Служба СУБД — это фоновый процесс операционной системы, в котором работает сервер базы данных и который принимает подключения клиентов. В Windows службы управляются через оснастку «Службы» или командой sc, в Linux — через systemd командой systemctl. Обычно служба настраивается на автоматический запуск при загрузке системы.

8. Как управлять службой СУБД в Linux?
Ответ: Для управления службой используется утилита systemctl: systemctl start — запуск, systemctl stop — остановка, systemctl restart — перезапуск, systemctl status — проверка состояния, systemctl enable — включение автоматического запуска при загрузке. Например, systemctl status postgresql показывает, работает ли сервер PostgreSQL.

9. Какие сетевые порты по умолчанию используют основные СУБД?
Ответ: MySQL использует порт 3306, PostgreSQL — 5432, Oracle Database (слушатель) — 1521, Microsoft SQL Server — 1433, MongoDB — 27017. При необходимости порт изменяется в конфигурации, а на межсетевом экране открывается доступ к нему только для разрешённых адресов.

10. Что такое конфигурационные файлы СУБД? Приведите примеры.
Ответ: Конфигурационные файлы содержат параметры работы сервера СУБД: сетевые настройки, параметры памяти, расположение файлов данных, журналирование. Примеры: my.cnf или my.ini в MySQL, postgresql.conf и pg_hba.conf в PostgreSQL, mongod.conf в MongoDB, файлы параметров инициализации init.ora или spfile, а также listener.ora и tnsnames.ora в Oracle.

11. Какие параметры настраиваются после установки СУБД?
Ответ: Настраиваются сетевой адрес и порт прослушивания, разрешённые подключения и способы аутентификации, пароли администраторов, кодировка и правила сортировки, часовой пояс, параметры использования памяти, максимальное количество подключений, расположение файлов данных и журналов, параметры журналирования, автоматический запуск службы и резервное копирование.

12. Почему важно правильно выбрать кодировку базы данных?
Ответ: Кодировка определяет, какие символы могут храниться в базе данных. При неправильной кодировке русские буквы и другие символы могут искажаться или не сохраняться. Рекомендуется использовать кодировку UTF-8 (в MySQL — utf8mb4), поддерживающую все символы, включая эмодзи. Изменить кодировку существующей базы с данными сложно, поэтому её выбирают при создании.

13. Какие меры по обеспечению безопасности выполняются при первоначальной настройке СУБД?
Ответ: Устанавливаются надёжные пароли для учётных записей администраторов, удаляются анонимные пользователи и тестовые базы, запрещается удалённый вход администратора, сервер прослушивает только необходимые сетевые интерфейсы, настраивается межсетевой

экран, создаются отдельные учётные записи с минимальными правами для приложений, включается шифрование соединений и устанавливаются обновления безопасности.

14. Что делает утилита mysql_secure_installation?
Ответ: Утилита mysql_secure_installation выполняет первоначальную настройку безопасности MySQL: позволяет задать пароль и политику паролей, удалить анонимных пользователей, запретить удалённый вход пользователя root, удалить тестовую базу данных test и перезагрузить таблицы привилегий.

15. Для чего предназначены файлы postgresql.conf и pg_hba.conf?
Ответ: Файл postgresql.conf содержит основные параметры работы сервера PostgreSQL: адреса прослушивания (listen_addresses), порт, количество подключений, параметры памяти и журналирования. Файл pg_hba.conf определяет правила аутентификации клиентов: какие пользователи, к каким базам данных, с каких адресов и каким методом могут подключаться, например scram-sha-256.

16. Какие параметры памяти настраиваются в СУБД?
Ответ: Настраиваются размер общего буферного кэша для данных (shared_buffers в PostgreSQL, innodb_buffer_pool_size в MySQL), объём памяти для операций сортировки и соединения (work_mem в PostgreSQL), максимальный объём памяти сервера (max server memory в SQL Server, кэш WiredTiger в MongoDB). Значения выбираются с учётом объёма оперативной памяти сервера и нагрузки.

17. Что такое переменные окружения ORACLE_HOME, ORACLE_BASE и ORACLE_SID?
Ответ: ORACLE_BASE — корневой каталог установки программного обеспечения Oracle. ORACLE_HOME — каталог, в котором установлена конкретная версия программного обеспечения Oracle Database. ORACLE_SID — идентификатор экземпляра базы данных, с которым работают утилиты. Каталог $ORACLE_HOME/bin добавляется в переменную PATH, чтобы запускать sqlplus, lsnrctl и другие утилиты без указания полного пути.

18. Какие способы аутентификации поддерживает Microsoft SQL Server?
Ответ: SQL Server поддерживает режим проверки подлинности Windows, при котором пользователи входят с учётными записями Windows или Active Directory, и смешанный режим, при котором дополнительно используются логины SQL Server с паролями, хранящимися на сервере. Режим проверки подлинности Windows считается более безопасным. Изменение режима требует перезапуска службы.

19. Что такое репликация в MongoDB и для чего она нужна?
Ответ: Репликация — это поддержание нескольких копий данных на разных серверах. В MongoDB она реализуется с помощью набора реплик (replica set), состоящего из основного узла (primary), принимающего запись, и вторичных узлов (secondary), копирующих данные. При отказе основного узла автоматически выбирается новый, что обеспечивает отказоустойчивость. Рекомендуется не менее трёх узлов.

20. Как проверить результаты установки и настройки СУБД?
Ответ: Проверяется состояние службы, прослушивание нужного порта (например, командой ss -tlnp или netstat), возможность подключения клиентом локально и с удалённого компьютера, версия СУБД, значения настроенных параметров, применение правил аутентификации и прав доступа, отсутствие ошибок в журналах сервера, а также выполнение тестовых запросов на создание таблицы, запись и чтение данных.

21. Какими командами можно проверить значения параметров СУБД?
Ответ: В MySQL используются команды SHOW VARIABLES и SELECT @@имя_переменной, в PostgreSQL — SHOW имя_параметра и запрос к представлению pg_settings, в Oracle — SHOW PARAMETER в SQL*Plus, в SQL Server — системная

процедура sp_configure и представление sys.configurations, в MongoDB — команды db.adminCommand({getParameter: ...}) и rs.conf() для набора реплик.

22. Для чего используются журналы (логи) СУБД при контроле настройки?
Ответ: Журналы сервера СУБД содержат сведения о запуске и остановке сервера, ошибках, неудачных попытках входа, медленных запросах и предупреждениях о неправильных настройках. Анализ журналов после установки и изменения конфигурации позволяет убедиться, что сервер запустился без ошибок и параметры применены, а также выявить причины проблем.

23. Почему результаты установки и настройки СУБД необходимо документировать?
Ответ: Документация позволяет восстановить или повторить настройку сервера, передать сопровождение другому специалисту, контролировать изменения конфигурации и расследовать проблемы. В документации указываются версия СУБД, путь установки, порт, расположение файлов данных и журналов, изменённые параметры, созданные учётные записи и роли без указания паролей.

24. Что такое контейнеризация СУБД?
Ответ: Контейнеризация СУБД — это запуск сервера базы данных в изолированном Docker-контейнере, созданном из готового образа, содержащего СУБД и все необходимые компоненты. Контейнер запускается одинаково на любом компьютере с Docker, не требует установки СУБД в операционную систему и позволяет быстро развернуть несколько СУБД разных версий.

25. Какие преимущества и недостатки имеет запуск СУБД в контейнерах?
Ответ: Преимущества: быстрое развёртывание и удаление, одинаковое окружение у всех разработчиков, возможность запуска нескольких версий СУБД, описание конфигурации в виде кода, удобство для тестирования и обучения. Недостатки: необходимость правильно настроить хранение данных в томах, дополнительные накладные расходы на ввод-вывод, сложность настройки высокой доступности, а для промышленных нагруженных баз требуется особая осторожность.

26. Какие официальные Docker-образы СУБД существуют и как задаются их основные параметры?
Ответ: Существуют образы mysql, postgres, mongo, mcr.microsoft.com/mssql/server, а также образы Oracle Database Free из реестра Oracle. Основные параметры задаются переменными окружения: MYSQL_ROOT_PASSWORD для MySQL, POSTGRES_PASSWORD и POSTGRES_DB для PostgreSQL, MONGO_INITDB_ROOT_USERNAME и MONGO_INITDB_ROOT_PASSWORD для MongoDB, ACCEPT_EULA=Y и MSSQL_SA_PASSWORD для SQL Server.

27. Почему при запуске СУБД в контейнере необходимо использовать тома (volumes)?
Ответ: Данные, записанные внутри контейнера, теряются при его удалении. Тома хранят данные вне контейнера, поэтому база данных сохраняется при перезапуске, обновлении образа и пересоздании контейнера. Том монтируется в каталог данных СУБД, например /var/lib/postgresql/data для PostgreSQL, /var/lib/mysql для MySQL, /data/db для MongoDB.

28. Что такое Docker Compose и как он используется для развёртывания СУБД?
Ответ: Docker Compose — это инструмент для описания и запуска многоконтейнерных приложений с помощью файла docker-compose.yaml. В нём описываются сервисы СУБД и средств администрирования (например, PostgreSQL и pgAdmin), образы, переменные

окружения, порты, тома, сети и проверки работоспособности. Все сервисы запускаются одной командой docker compose up -d.

29. Как выполнить начальную инициализацию базы данных в контейнере?
Ответ: Официальные образы MySQL, PostgreSQL и MongoDB при первом запуске с пустым каталогом данных выполняют скрипты из каталога /docker-entrypoint-initdb.d. В этот каталог монтируются файлы SQL или скрипты, создающие базы данных, таблицы, пользователей и тестовые данные. Это позволяет автоматически получать готовую к работе базу.

30. Как проверить работоспособность СУБД, запущенной в контейнере?
Ответ: Состояние контейнера проверяется командами docker ps и docker compose ps, журналы — командой docker logs. Подключение к СУБД выполняется командой docker exec с запуском клиента внутри контейнера, например docker exec -it db psql -U postgres, или с компьютера через опубликованный порт. В docker-compose.yaml настраивается проверка healthcheck, например pg_isready для PostgreSQL или mysqladmin ping для MySQL.

Критерии оценки устного опроса:

- оценка «отлично» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов и смог ответить на уточняющие вопросы;
- оценка «хорошо» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов, но не смог ответить на уточняющие вопросы;
- оценка «удовлетворительно» выставляется студенту, если он смог изложить суть основного вопроса, но не смог ответить по существу дополнительных вопросов;
- оценка «неудовлетворительно» выставляется студенту, если он не смог изложить суть основного вопроса.

Практическое занятие №1. Установка СУБД MySQL и настройка службы на локальном администраторов с базами данных.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных
ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации
ПК 2.5. Подготавливать данные для базы знаний
ПК 2.6. Осуществлять управление базами данных
ПК 2.7. Осуществлять автоматизацию баз данных

Цель: Научиться устанавливать СУБД MySQL на локальный компьютер, настраивать службу сервера, выполнять первоначальную настройку безопасности и основных параметров, а также проверять результаты установки и настройки.

Задание:
1. Проверить соответствие компьютера системным требованиям MySQL.
2. Загрузить дистрибутив MySQL из официального источника.
3. Установить сервер MySQL и клиентские средства администрирования.
4. Настроить службу MySQL и её автоматический запуск.
5. Выполнить первоначальную настройку безопасности.
6. Настроить основные параметры сервера в конфигурационном файле.
7. Создать тестовую базу данных и пользователя, проверить подключение.
8. Проверить результаты установки и настройки и сделать вывод.

Этапы выполнения:
Этап 1. Подготовка к установке.
Определяется версия операционной системы, объём оперативной памяти и свободного места на диске, проверяется, что порт 3306 не занят другим приложением, и что у пользователя есть права администратора.

Этап 2. Загрузка дистрибутива.
Для Windows с официального сайта загружается MySQL Installer, для Linux подключается официальный репозиторий MySQL или используется пакет из репозитория дистрибутива.

Этап 3. Установка.
В Windows в установщике выбираются компоненты MySQL Server, MySQL Shell и MySQL Workbench. В Linux сервер устанавливается командой, например sudo apt install mysql-server. При установке задаётся пароль пользователя root и метод аутентификации.

Этап 4. Настройка службы.
В Windows при установке задаётся имя службы (например, MySQL80), учётная запись службы и автоматический запуск, затем состояние проверяется в оснастке «Службы». В Linux служба включается командой systemctl enable mysql и проверяется командой systemctl status mysql.

Этап 5. Настройка безопасности.
В Linux запускается утилита mysql_secure_installation: задаётся политика паролей, удаляются анонимные пользователи и тестовая база, запрещается удалённый вход root. В Windows соответствующие параметры проверяются и при необходимости изменяются SQL-командами.

Этап 6. Настройка параметров.
В конфигурационном файле my.ini (Windows) или my.cnf (Linux) в разделе [mysqld] задаются кодировка character-set-server=utf8mb4, правило сортировки, порт, адрес прослушивания bind-

address, максимальное количество подключений max_connections, размер буферного пула innodb_buffer_pool_size и журнал ошибок. После изменения служба перезапускается.

Этап 7. Создание базы и пользователя.
С помощью клиента mysql или MySQL Workbench создаётся база данных, пользователь с паролем и правами только на эту базу. Выполняется подключение под новым пользователем, создание таблицы, вставка и чтение данных.

Этап 8. Контроль и оформление результата.
Проверяются состояние службы, версия сервера (SELECT VERSION()), значения параметров (SHOW VARIABLES), прослушиваемый порт, журнал ошибок. Оформляется отчёт с описанием выполненных действий, снимками экрана и выводами.

Пример решения:
Задание: Установить MySQL 8 на компьютер администратора баз данных под управлением Ubuntu и подготовить базу данных для учебного приложения.
Дано: Компьютер с Ubuntu, 8 ГБ оперативной памяти, 50 ГБ свободного места. Требуется: служба запускается автоматически, кодировка utf8mb4, сервер доступен только локально, создана база library и пользователь library_app с правами только на неё.

Решение по этапам:
1. Проверяем системные требования и свободность порта командой ss -tlnp | grep 3306 — порт свободен.
2. Обновляем список пакетов и устанавливаем сервер командами sudo apt update и sudo apt install mysql-server.
3. Проверяем службу командой systemctl status mysql — служба активна, включаем автозапуск командой sudo systemctl enable mysql.
4. Запускаем sudo mysql_secure_installation: включаем проверку надёжности паролей, удаляем анонимных пользователей и базу test, запрещаем удалённый вход root.
5. В файле /etc/mysql/mysql.conf.d/mysqld.cnf задаём bind-address = 127.0.0.1, character-set-server = utf8mb4, collation-server = utf8mb4_0900_ai_ci, max_connections = 100, innodb_buffer_pool_size = 2G.
6. Перезапускаем службу командой sudo systemctl restart mysql и проверяем, что она запущена.
7. Подключаемся командой sudo mysql и создаём базу: CREATE DATABASE library; пользователя: CREATE USER 'library_app'@'localhost' IDENTIFIED BY 'надёжный_пароль'; и выдаём права: GRANT SELECT, INSERT, UPDATE, DELETE ON library.* TO 'library_app'@'localhost';
8. Подключаемся командой mysql -u library_app -p library, создаём таблицу books, вставляем и читаем запись — операции выполняются успешно. Попытка создать базу данных под этим пользователем завершается ошибкой доступа, что подтверждает ограничение прав.
9. Проверяем параметры: SELECT VERSION(); SHOW VARIABLES LIKE 'character_set_server'; SHOW VARIABLES LIKE 'bind_address'; — значения соответствуют настройкам. В журнале /var/log/mysql/error.log ошибок нет.
10. Оформляем отчёт.

Ответ: СУБД MySQL 8 установлена, служба настроена на автоматический запуск, выполнена первоначальная настройка безопасности, сервер доступен только локально и использует кодировку utf8mb4. Создана база данных library и пользователь library_app с минимально

необходимыми правами. Контроль подтвердил корректность установки и применения параметров.

Критерии оценки практических заданий:
1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Практическое занятие №2. Установка PostgreSQL и настройка параметров конфигурации.
Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных
ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации
ПК 2.5. Подготавливать данные для базы знаний
ПК 2.6. Осуществлять управление базами данных
ПК 2.7. Осуществлять автоматизацию баз данных

Цель: Научиться устанавливать СУБД PostgreSQL, настраивать параметры сервера в файлах postgresql.conf и pg_hba.conf, управлять параметрами с помощью SQL-команд и проверять применение настроек.

Задание:
1. Установить сервер PostgreSQL и клиентские средства.
2. Проверить работу службы и подключиться к серверу.
3. Задать пароль пользователя postgres.
4. Настроить сетевые параметры и количество подключений.
5. Настроить параметры памяти и журналирования.
6. Настроить правила аутентификации в pg_hba.conf.
7. Применить изменения и проверить значения параметров.
8. Проверить подключения в соответствии с правилами и сделать вывод.

Этапы выполнения:
Этап 1. Установка.
В Windows используется официальный установщик, в котором выбираются компоненты сервера, pgAdmin и утилиты командной строки, указываются каталог данных, пароль и порт.

В Linux PostgreSQL устанавливается из официального репозитория командой, например sudo apt install postgresql.

Этап 2. Проверка службы и подключение.
Проверяется состояние службы командой systemctl status postgresql или в оснастке «Службы». Выполняется подключение командой sudo -u postgres psql, проверяется версия запросом SELECT version();.

Этап 3. Пароль администратора.
Для пользователя postgres задаётся надёжный пароль командой ALTER USER postgres PASSWORD '...'; или командой \password в psql.

Этап 4. Сетевые параметры.
Определяется расположение конфигурационных файлов запросом SHOW config_file;. В postgresql.conf задаются listen_addresses (адреса прослушивания), port и max_connections.

Этап 5. Параметры памяти и журналирования.
Задаются shared_buffers (обычно около 25% оперативной памяти), work_mem, maintenance_work_mem, effective_cache_size, а также параметры журналирования: logging_collector, log_min_duration_statement для записи медленных запросов, log_connections.

Этап 6. Настройка pg_hba.conf.
Добавляются правила, разрешающие подключение определённым пользователям к определённым базам с определённых адресов с методом аутентификации scram-sha-256. Удалённые подключения разрешаются только из нужной подсети.

Этап 7. Применение изменений.
Параметры, не требующие перезапуска, применяются командой SELECT pg_reload_conf();, параметры памяти и сети требуют перезапуска службы. Альтернативно параметры изменяются командой ALTER SYSTEM SET, которая записывает их в файл

postgresql.auto.conf. Значения проверяются командой SHOW и запросом к pg_settings, в столбце pending_restart отображается необходимость перезапуска.

Этап 8. Контроль и оформление.
Проверяется подключение разрешённого пользователя и отказ в подключении запрещённым. Анализируется журнал сервера. Оформляется отчёт.

Пример решения:
Задание: Установить PostgreSQL на сервер отдела и настроить параметры для работы учебного приложения.
Дано: Сервер с Ubuntu и 16 ГБ оперативной памяти. Приложение работает с компьютеров подсети 192.168.10.0/24 и подключается к базе shop под пользователем shop_app. Требуется до 150 подключений, журналирование запросов дольше 500 мс.

Решение по этапам:
1. Устанавливаем PostgreSQL командой sudo apt install postgresql, проверяем службу командой systemctl status postgresql — служба активна.
2. Подключаемся командой sudo -u postgres psql и задаём пароль пользователя postgres командой \password.
3. Создаём пользователя и базу: CREATE ROLE shop_app LOGIN PASSWORD 'надёжный_пароль'; CREATE DATABASE shop OWNER shop_app;
4. Определяем путь к файлам командой SHOW config_file;.
5. В postgresql.conf задаём listen_addresses = 'localhost,192.168.10.5', max_connections = 150, shared_buffers = 4GB, effective_cache_size = 12GB, work_mem = 16MB, logging_collector = on, log_min_duration_statement = 500.
6. В pg_hba.conf добавляем правило: host shop shop_app 192.168.10.0/24 scram-sha-256.
7. Перезапускаем службу командой sudo systemctl restart postgresql, так как изменены параметры памяти и сети.
8. Проверяем значения: SHOW shared_buffers; SHOW max_connections; SELECT name, setting, pending_restart FROM pg_settings WHERE name IN ('work_mem', 'log_min_duration_statement'); — значения применены, перезапуск не требуется.
9. С компьютера подсети выполняем psql -h 192.168.10.5 -U shop_app -d shop — подключение успешно. Попытка подключения к базе postgres под пользователем shop_app отклоняется по правилам pg_hba.conf. Выполняем медленный запрос SELECT pg_sleep(1); и находим его в журнале сервера.
10. Оформляем отчёт.

Ответ: PostgreSQL установлен и настроен: сервер прослушивает только нужные адреса, допускает до 150 подключений, параметры памяти рассчитаны для 16 ГБ оперативной памяти, включено журналирование медленных запросов. Правила pg_hba.conf разрешают пользователю shop_app подключаться только к базе shop из подсети отдела. Проверка подтвердила применение всех параметров.

Критерии оценки практических заданий:
1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Практическое занятие №3. Установка Oracle Database и настройка окружения (переменные PATH, ORACLE_HOME).
Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к

различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных
ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации
ПК 2.5. Подготавливать данные для базы знаний
ПК 2.6. Осуществлять управление базами данных
ПК 2.7. Осуществлять автоматизацию баз данных

Цель: Научиться устанавливать СУБД Oracle Database, настраивать переменные окружения ORACLE_HOME, ORACLE_BASE, ORACLE_SID и PATH, проверять работу слушателя и подключаться к базе данных с помощью SQL*Plus и Oracle SQL Developer.

Задание:
1. Проверить системные требования и подготовить операционную систему.
2. Установить Oracle Database (например, бесплатную редакцию Oracle Database Free).
3. Выполнить создание и настройку экземпляра базы данных.
4. Настроить переменные окружения ORACLE_BASE, ORACLE_HOME, ORACLE_SID и PATH.
5. Проверить работу слушателя (listener).
6. Подключиться к базе данных с помощью SQL*Plus.
7. Создать пользователя и проверить подключение через Oracle SQL Developer.
8. Проверить результаты и сделать вывод.

Этапы выполнения:
Этап 1. Подготовка системы.
Проверяется соответствие требованиям: поддерживаемая версия операционной системы (например, Oracle Linux, RHEL или Windows), объём оперативной памяти, свободное место на диске, наличие прав администратора. В Linux устанавливается пакет предварительной настройки, создающий пользователя oracle и необходимые параметры ядра.

Этап 2. Установка.
Дистрибутив загружается с официального сайта Oracle. В Linux пакет RPM устанавливается командой dnf install, в Windows запускается установщик, в котором указываются каталог установки и пароль администратора.

Этап 3. Создание базы данных.
В Linux выполняется сценарий настройки, создающий контейнерную базу данных, подключаемую базу данных и слушатель, при этом задаётся пароль пользователей SYS, SYSTEM и PDBADMIN. В Windows база создаётся установщиком автоматически.

Этап 4. Настройка переменных окружения.
В Linux в файл ~/.bash_profile пользователя oracle добавляются строки export ORACLE_BASE, export ORACLE_HOME, export ORACLE_SID и export PATH=$ORACLE_HOME/bin:$PATH, после чего файл применяется командой source. В Windows переменные проверяются и при необходимости задаются в окне «Переменные среды» свойств системы. Значения

проверяются командами echo $ORACLE_HOME в Linux и echo %ORACLE_HOME% в Windows.

Этап 5. Проверка слушателя.
Командой lsnrctl status проверяется, что слушатель запущен, прослушивает порт 1521 и обслуживает нужные службы базы данных.

Этап 6. Подключение через SQL*Plus.
Выполняется подключение командой sqlplus / as sysdba. Проверяется состояние экземпляра запросом SELECT instance_name, status FROM v$instance; и список подключаемых баз командой SHOW PDBS.

Этап 7. Создание пользователя и проверка SQL Developer.
В подключаемой базе данных создаётся пользователь с паролем и правами CREATE SESSION и CREATE TABLE. В Oracle SQL Developer создаётся подключение с указанием сервера, порта 1521 и имени службы, проверяется вход и создание таблицы.

Этап 8. Оформление результата.
Оформляется отчёт, содержащий значения переменных окружения, результаты проверки слушателя, сведения об экземпляре, снимки экрана подключений и выводы.

Пример решения:
Задание: Установить Oracle Database Free на учебный сервер с Oracle Linux и настроить окружение администратора.
Дано: Сервер с Oracle Linux, 8 ГБ оперативной памяти. Требуется, чтобы администратор мог запускать утилиты Oracle из любого каталога, а разработчик подключался к подключаемой базе данных через SQL Developer.

Решение по этапам:
1. Устанавливаем пакет предварительной настройки командой sudo dnf install oracle-database-preinstall для нужной версии, который создаёт пользователя oracle и настраивает параметры ядра.
2. Устанавливаем пакет Oracle Database Free командой sudo dnf install с указанием загруженного файла RPM.
3. Запускаем сценарий настройки базы данных из каталога /etc/init.d с параметром configure, задаём пароль администраторов. Создаются контейнерная база данных FREE и подключаемая база FREEPDB1, слушатель на порту 1521.
4. Входим под пользователем oracle и добавляем в ~/.bash_profile строки: export ORACLE_BASE=/opt/oracle; export ORACLE_HOME=/opt/oracle/product/23ai/dbhomeFree; export ORACLE_SID=FREE; export PATH=$ORACLE_HOME/bin:$PATH. Применяем командой source ~/.bash_profile.
5. Проверяем: echo $ORACLE_HOME выводит путь к каталогу установки, which sqlplus показывает путь к утилите в $ORACLE_HOME/bin.
6. Выполняем lsnrctl status — слушатель запущен, прослушивает порт 1521, обслуживает службы FREE и FREEPDB1.
7. Подключаемся командой sqlplus / as sysdba, запрос к v$instance показывает экземпляр FREE в состоянии OPEN, команда SHOW PDBS показывает FREEPDB1 в режиме READ WRITE.
8. Переключаемся в подключаемую базу командой ALTER SESSION SET CONTAINER = FREEPDB1; и создаём пользователя: CREATE USER dev_user IDENTIFIED BY

"надёжный_пароль"; GRANT CREATE SESSION, CREATE TABLE TO dev_user; ALTER USER dev_user QUOTA 100M ON USERS;
9. В SQL Developer создаём подключение: сервер — адрес учебного сервера, порт 1521, имя службы FREEPDB1, пользователь dev_user. Подключение успешно, таблица создаётся.
10. Оформляем отчёт.

Ответ: Oracle Database Free установлена, создана контейнерная база FREE с подключаемой базой FREEPDB1. Переменные окружения ORACLE_BASE, ORACLE_HOME, ORACLE_SID и PATH настроены в профиле пользователя oracle, утилиты Oracle запускаются из любого каталога. Слушатель работает на порту 1521, создан пользователь dev_user, подключение через SQL*Plus и SQL Developer проверено.

Критерии оценки практических заданий:
1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Практическое занятие №4. Установка MongoDB и настройка репликации для отказоустойчивости.
Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных
ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации
ПК 2.5. Подготавливать данные для базы знаний
ПК 2.6. Осуществлять управление базами данных
ПК 2.7. Осуществлять автоматизацию баз данных

Цель: Научиться устанавливать СУБД MongoDB, настраивать набор реплик из нескольких узлов, проверять репликацию данных и автоматическое переключение основного узла при отказе.

Задание:
1. Установить MongoDB и клиентскую оболочку mongosh.
2. Подготовить три экземпляра MongoDB для набора реплик.
3. Настроить параметры репликации в конфигурационных файлах.
4. Инициализировать набор реплик.
5. Проверить состояние набора реплик.
6. Проверить репликацию данных на вторичные узлы.
7. Смоделировать отказ основного узла и проверить автоматическое переключение.
8. Сделать вывод об отказоустойчивости конфигурации.

Этапы выполнения:
Этап 1. Установка.
MongoDB Community Server устанавливается из официального репозитория MongoDB в Linux или с помощью установщика в Windows. Отдельно устанавливается оболочка mongosh и при необходимости MongoDB Compass. Также узлы можно развернуть в Docker-контейнерах.

Этап 2. Подготовка экземпляров.
Для каждого из трёх узлов готовится отдельный каталог данных, журнал и порт (на разных серверах — порт 27017, на одном компьютере — например, 27017, 27018, 27019) или отдельный контейнер.

Этап 3. Настройка конфигурации.
В конфигурационном файле mongod.conf каждого узла указываются путь к данным, порт, адрес прослушивания и имя набора реплик в разделе replication: replSetName. Для защиты взаимодействия узлов настраивается файл ключа (keyFile) и включается авторизация.

Этап 4. Инициализация набора реплик.
Все экземпляры запускаются. На одном узле в mongosh выполняется команда rs.initiate() с перечнем участников набора реплик и их адресами.

Этап 5. Проверка состояния.
Командой rs.status() проверяется, что один узел имеет состояние PRIMARY, а остальные — SECONDARY, и все узлы доступны. Командой rs.conf() проверяется конфигурация набора.

Этап 6. Проверка репликации.
На основном узле создаётся база данных и вставляются документы с параметром записи writeConcern: { w: "majority" }. На вторичном узле разрешается чтение и проверяется наличие тех же документов.

Этап 7. Моделирование отказа.
Основной узел останавливается. Командой rs.status() проверяется, что один из вторичных узлов автоматически стал основным. Проверяется, что запись данных продолжается. Затем

отключённый узел запускается снова и проверяется, что он вернулся в набор как вторичный и получил пропущенные данные.

Этап 8. Оформление результата.
Оформляется отчёт, содержащий конфигурационные файлы, команды инициализации, результаты rs.status() до и после отказа, результаты проверки репликации и выводы.

Пример решения:
Задание: Развернуть отказоустойчивую конфигурацию MongoDB для хранения журнала заказов интернет-магазина.
Дано: Требуется, чтобы при отказе одного сервера база данных продолжала принимать запись. Для учебной работы используются три контейнера Docker: mongo1, mongo2, mongo3 в общей сети.
Решение по этапам:
1. Создаём docker-compose.yaml с тремя сервисами на основе образа mongo, каждый запускается командой mongod --replSet rs0 --bind_ip_all и имеет отдельный том для каталога /data/db.
2. Запускаем контейнеры командой docker compose up -d и проверяем их состояние командой docker compose ps.
3. Подключаемся к первому узлу командой docker exec -it mongo1 mongosh.
4. Инициализируем набор реплик: rs.initiate({ _id: "rs0", members: [{ _id: 0, host: "mongo1:27017" }, { _id: 1, host: "mongo2:27017" }, { _id: 2, host: "mongo3:27017" }] }).
5. Выполняем rs.status(): mongo1 — PRIMARY, mongo2 и mongo3 — SECONDARY.
6. На основном узле вставляем документы: use shop; db.orders.insertOne({ order_id: 1, amount: 2500 }, { writeConcern: { w: "majority" } }).
7. Подключаемся к mongo2, выполняем db.getMongo().setReadPref("secondaryPreferred") и db.orders.find() — документ присутствует.
8. Останавливаем основной узел командой docker stop mongo1. Через несколько секунд rs.status() на mongo2 показывает, что mongo2 стал PRIMARY. Вставляем новый заказ — запись выполняется успешно.
9. Запускаем mongo1 командой docker start mongo1. Узел возвращается в состояние SECONDARY, и на нём появляется заказ, добавленный во время его отключения.
10. Оформляем отчёт и указываем, что в промышленной среде узлы размещаются на разных серверах, включаются авторизация и файл ключа.

Ответ: Развёрнут набор реплик rs0 из трёх узлов MongoDB. Проверено, что данные реплицируются на вторичные узлы, при отказе основного узла автоматически выбирается новый основной узел и запись продолжается, а восстановленный узел догоняет пропущенные изменения. Конфигурация обеспечивает отказоустойчивость при отказе одного узла.

Критерии оценки практических заданий:
1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Практическое занятие №5. Установка Microsoft SQL Server и настройка параметров аутентификации (назначение ролей и прав доступа).
Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности

ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных
ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации
ПК 2.5. Подготавливать данные для базы знаний
ПК 2.6. Осуществлять управление базами данных
ПК 2.7. Осуществлять автоматизацию баз данных

Цель: Научиться устанавливать Microsoft SQL Server, настраивать режим проверки подлинности и сетевые протоколы, создавать логины и пользователей баз данных, назначать серверные и пользовательские роли и права доступа, проверять разграничение доступа.

Задание:
1. Установить Microsoft SQL Server (редакция Developer или Express) и SQL Server Management Studio.
2. Выбрать и настроить режим проверки подлинности.
3. Настроить сетевой протокол TCP/IP и порт.
4. Создать учебную базу данных.
5. Создать логины SQL Server и Windows.
6. Создать пользователей базы данных и назначить им роли.
7. Назначить и запретить отдельные права доступа к объектам.
8. Проверить разграничение доступа и сделать вывод.

Этапы выполнения:
Этап 1. Установка.
С официального сайта Microsoft загружается установщик SQL Server. При установке выбираются компоненты ядра СУБД, имя экземпляра, учётные записи служб и режим проверки подлинности, назначаются администраторы SQL Server. Отдельно устанавливается SQL Server Management Studio (SSMS).

Этап 2. Режим проверки подлинности.
Выбирается режим проверки подлинности Windows или смешанный режим. При смешанном режиме задаётся надёжный пароль для логина sa, который затем рекомендуется отключить или переименовать. Режим можно изменить в свойствах сервера в SSMS на странице «Безопасность» с последующим перезапуском службы.

Этап 3. Сетевые настройки.
В SQL Server Configuration Manager включается протокол TCP/IP, задаётся порт 1433, служба перезапускается. На межсетевом экране открывается порт для разрешённых компьютеров.

Этап 4. Создание базы данных.
Создаётся учебная база данных и таблицы с тестовыми данными.

Этап 5. Создание логинов.
Создаются логины SQL Server командой CREATE LOGIN ... WITH PASSWORD с проверкой политики паролей и логины Windows командой CREATE LOGIN [ДОМЕН\Пользователь]

FROM WINDOWS. При необходимости логину назначается серверная роль, например dbcreator, командой ALTER SERVER ROLE ... ADD MEMBER.

Этап 6. Создание пользователей и назначение ролей.
В базе данных для логинов создаются пользователи командой CREATE USER ... FOR LOGIN. Назначаются встроенные роли базы данных: db_datareader (чтение всех таблиц), db_datawriter (изменение данных), db_ddladmin (изменение структуры), db_owner (полный доступ), а также создаются пользовательские роли командой CREATE ROLE.

Этап 7. Назначение прав.
Права на отдельные объекты назначаются командами GRANT (предоставить), DENY (запретить) и REVOKE (отозвать). Например, ролям предоставляется право SELECT на представление и запрещается доступ к таблице с конфиденциальными данными.

Этап 8. Проверка и оформление.
Выполняется подключение под каждым логином и проверяется, что разрешённые действия выполняются, а запрещённые завершаются ошибкой доступа. Действующие права проверяются с помощью функции fn_my_permissions и системных представлений. Оформляется отчёт.

Пример решения:
Задание: Установить SQL Server и настроить доступ к базе данных отдела кадров.
Дано: База данных HR содержит таблицы employees (сотрудники), departments (отделы) и salaries (зарплаты). Пользователи: администратор базы, специалист отдела кадров, который читает и изменяет данные сотрудников, и аналитик, который читает только данные сотрудников и отделов без доступа к зарплатам. Аналитик подключается логином SQL Server из стороннего приложения.

Решение по этапам:
1. Устанавливаем SQL Server Developer с экземпляром по умолчанию, выбираем смешанный режим проверки подлинности, задаём надёжный пароль sa, назначаем администратором учётную запись Windows администратора базы данных. Устанавливаем SSMS.
2. В SQL Server Configuration Manager включаем TCP/IP на порту 1433 и перезапускаем службу SQL Server.
3. Создаём базу HR и таблицы с тестовыми данными.
4. Отключаем логин sa командой ALTER LOGIN sa DISABLE; для повышения безопасности.
5. Создаём логин Windows для специалиста отдела кадров: CREATE LOGIN [COMPANY\hr_spec] FROM WINDOWS; и логин SQL Server для аналитика: CREATE LOGIN analyst WITH PASSWORD = 'надёжный_пароль', CHECK_POLICY = ON;
6. В базе HR создаём пользователей: CREATE USER hr_spec FOR LOGIN [COMPANY\hr_spec]; CREATE USER analyst FOR LOGIN analyst;
7. Назначаем специалисту роли чтения и записи: ALTER ROLE db_datareader ADD MEMBER hr_spec; ALTER ROLE db_datawriter ADD MEMBER hr_spec;
8. Создаём роль для аналитиков: CREATE ROLE analysts; GRANT SELECT ON dbo.employees TO analysts; GRANT SELECT ON dbo.departments TO analysts; DENY SELECT ON dbo.salaries TO analysts; ALTER ROLE analysts ADD MEMBER analyst;
9. Проверяем: под логином analyst запрос к employees выполняется, запрос к salaries завершается ошибкой «Отказано в разрешении SELECT», попытка изменения данных запрещена. Под логином hr_spec чтение и изменение данных выполняются, создание таблицы запрещено.
10. Оформляем отчёт с перечнем логинов, пользователей, ролей и прав.

Ответ: Microsoft SQL Server установлен в смешанном режиме проверки подлинности, включён протокол TCP/IP на порту 1433, логин sa отключён. Созданы логины Windows и SQL Server, пользователи базы HR, назначены встроенные роли и пользовательская роль analysts с

запретом доступа к таблице зарплат. Проверка подтвердила, что каждый пользователь может выполнять только разрешённые ему действия.

Критерии оценки практических заданий:
1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Тема 1.2. Установка и настройка программного обеспечения (ПО) для обеспечения работы пользователей с базами данных.
Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных
ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации
ПК 2.5. Подготавливать данные для базы знаний
ПК 2.6. Осуществлять управление базами данных

Вопросы к устному опросу по теме:
1. Что такое программное обеспечение для поддержки работы пользователей с базами данных?
Ответ: Это клиентские программы и компоненты, позволяющие пользователям и приложениям подключаться к базам данных и работать с ними: графические клиенты для выполнения запросов и просмотра данных, веб-интерфейсы, драйверы и библиотеки подключения для языков программирования (ODBC, JDBC, библиотеки Python), а также средства импорта, экспорта и построения отчётов.

2. Чем клиентское программное обеспечение отличается от серверного?
Ответ: Серверное программное обеспечение (сервер СУБД) хранит данные, обрабатывает запросы и управляет доступом. Клиентское программное обеспечение устанавливается на компьютеры пользователей, отправляет запросы серверу по сети и отображает результаты. Один сервер обслуживает множество клиентов.

3. Какие требования учитываются при выборе клиентского ПО для пользователей?
Ответ: Учитываются поддерживаемые СУБД и их версии, операционная система пользователей, задачи пользователей (выполнение запросов, просмотр данных, проектирование, отчёты), удобство интерфейса, поддержка русского языка, условия лицензирования и стоимость, поддержка безопасных подключений, требования информационной безопасности организации.

4. Что такое строка подключения и какие параметры она содержит?
Ответ: Строка подключения — это набор параметров, необходимых клиенту для соединения с базой данных. Она содержит тип СУБД или драйвер, адрес сервера, порт, имя базы данных,

имя пользователя и пароль, а также дополнительные параметры: кодировку, режим шифрования, время ожидания подключения.

5. Что такое драйверы баз данных ODBC и JDBC?
Ответ: ODBC (Open Database Connectivity) — это стандартный программный интерфейс доступа к базам данных, позволяющий приложениям работать с разными СУБД через соответствующие драйверы. JDBC (Java Database Connectivity) — аналогичный интерфейс для приложений на языке Java. Многие клиенты, например DBeaver, используют драйверы JDBC.

6. Что такое MySQL Workbench и какие возможности он предоставляет?
Ответ: MySQL Workbench — это официальное графическое средство для работы с MySQL. Оно позволяет создавать подключения, выполнять SQL-запросы в редакторе, просматривать и редактировать данные, проектировать базы данных в виде ER-диаграмм и генерировать по ним SQL-код, управлять пользователями и правами, выполнять экспорт и импорт данных, контролировать состояние сервера.

7. Что такое режим безопасных обновлений (Safe Updates) в MySQL Workbench?
Ответ: Режим безопасных обновлений запрещает выполнение команд UPDATE и DELETE без условия WHERE по ключевому столбцу или без ограничения LIMIT. Это защищает от случайного изменения или удаления всех строк таблицы. Режим включается и отключается в настройках SQL-редактора.

8. Что такое pgAdmin и в каких режимах он работает?
Ответ: pgAdmin — это официальное графическое средство администрирования и разработки для PostgreSQL. Он работает в настольном режиме как приложение на компьютере пользователя и в серверном режиме как веб-приложение, доступное нескольким пользователям через браузер. Серверный режим можно развернуть, например, в Docker-контейнере.

9. Какие основные инструменты есть в pgAdmin?
Ответ: В pgAdmin есть обозреватель объектов для просмотра серверов, баз данных, схем и таблиц, редактор запросов Query Tool с подсветкой синтаксиса и планом выполнения, просмотр и редактирование данных, инструмент построения ER-диаграмм, средства резервного копирования и восстановления, управление ролями и правами, панель мониторинга активности сервера.

10. Что такое SQL Server Management Studio (SSMS)?
Ответ: SSMS — это интегрированная среда Microsoft для управления экземплярами SQL Server и базами данных Azure SQL. Она включает обозреватель объектов, редактор запросов T-SQL, средства управления безопасностью, резервным копированием, заданиями агента SQL Server, монитор активности, диаграммы баз данных и шаблоны запросов.

11. Какие параметры указываются при подключении к SQL Server в SSMS?
Ответ: Указываются тип сервера (ядро СУБД), имя сервера и экземпляра, например localhost или СЕРВЕР\ИМЯ_ЭКЗЕМПЛЯРА, способ проверки подлинности (Windows или SQL Server), логин и пароль, а также параметры шифрования соединения и доверия сертификату сервера. Дополнительно можно указать базу данных по умолчанию.

12. Что такое DBeaver и каковы его преимущества?
Ответ: DBeaver — это универсальный кроссплатформенный клиент для работы с базами данных. Бесплатная редакция Community поддерживает MySQL, PostgreSQL, SQL Server, Oracle, SQLite и многие другие реляционные СУБД через драйверы JDBC. Преимущества:

единый интерфейс для разных СУБД, автоматическая загрузка драйверов, редактор SQL, ER-диаграммы, экспорт и импорт данных, подключение через SSH-туннель.

13. Что такое SSH-туннель и зачем он используется при подключении к базе данных?
Ответ: SSH-туннель — это защищённый канал, при котором клиент сначала подключается по протоколу SSH к промежуточному серверу, а затем через него к серверу базы данных. Это позволяет работать с базой данных, порт которой закрыт для прямого доступа извне, и шифрует передаваемые данные. Поддержка SSH-туннелей есть в DBeaver, MySQL Workbench и pgAdmin.

14. Какие библиотеки Python используются для работы с базами данных?
Ответ: Для MySQL используются PyMySQL и mysql-connector-python, для PostgreSQL — psycopg2 и psycopg (версия 3), для SQL Server — pyodbc и pymssql, для Oracle — python-oracledb, для MongoDB — pymongo. Встроенный модуль sqlite3 работает с SQLite. Библиотека SQLAlchemy предоставляет единый интерфейс и объектно-реляционное отображение для разных СУБД.

15. Что такое стандарт DB-API 2.0 в Python?
Ответ: DB-API 2.0 (PEP 249) — это стандарт интерфейса доступа к базам данных в Python. Он определяет единый порядок работы: создание подключения функцией connect(), получение курсора, выполнение запросов методом execute(), получение результатов методами fetchone(), fetchall(), фиксацию изменений методом commit() и откат методом rollback(). Благодаря стандарту код для разных СУБД похож.

16. Почему в запросах из Python необходимо использовать параметризацию?
Ответ: При подстановке пользовательских данных в текст SQL-запроса через форматирование строк возникает угроза SQL-инъекции. Параметризованный запрос передаёт значения отдельно от текста запроса, например cursor.execute("SELECT * FROM users WHERE id = %s", (user_id,)), и библиотека сама безопасно их обрабатывает.

17. Как безопасно хранить параметры подключения к базе данных в приложениях и клиентах?
Ответ: Пароли не следует указывать в коде программ и передавать в открытом виде. Их хранят в переменных окружения, файлах конфигурации, исключённых из системы контроля версий (например, .env), в защищённом хранилище паролей клиента или в системах управления секретами. Для пользователей создаются отдельные учётные записи с минимальными правами.

18. Как настроить шифрование соединения клиента с сервером базы данных?
Ответ: На сервере включается поддержка SSL/TLS и устанавливается сертификат. В клиенте задаётся режим шифрования: в PostgreSQL — параметр sslmode (например, require или verify-full), в MySQL — ssl-mode, в SQL Server — параметр Encrypt и проверка сертификата. При необходимости в клиенте указывается корневой сертификат для проверки подлинности сервера.

19. Какие настройки клиентского ПО влияют на удобство и безопасность работы пользователей?
Ответ: Это кодировка, формат даты и чисел, ограничение количества строк в результатах запроса, режим автоматической фиксации транзакций, подтверждение опасных операций, цветовая маркировка подключений к рабочим и тестовым базам, время ожидания запросов, автосохранение скриптов, сохранение паролей и использование SSH-туннелей.

20. Почему важно различать режим автоматической фиксации транзакций в клиентах?
Ответ: В режиме автоматической фиксации каждая команда сразу сохраняется в базе и не может быть отменена. В ручном режиме изменения сохраняются только после команды COMMIT и могут быть отменены командой ROLLBACK. Пользователь должен знать

текущий режим, чтобы не потерять изменения и не оставить открытые транзакции, блокирующие данные других пользователей.

21. Как проверить результаты установки и настройки клиентского ПО?
Ответ: Проверяется успешный запуск программы и подключение к серверу с нужными учётными данными, отображение структуры базы, выполнение тестовых запросов на чтение и, если разрешено, изменение данных, корректность отображения кириллицы, работа шифрования соединения, ограничения прав пользователя, а также экспорт и импорт данных.

22. Какие типичные ошибки возникают при подключении клиента к базе данных и как их устранить?
Ответ: Типичные ошибки: сервер недоступен или порт закрыт межсетевым экраном, сервер прослушивает только локальный адрес, неверный логин или пароль, отсутствие разрешения на подключение с адреса клиента (например, в pg_hba.conf), неверное имя базы или экземпляра, ошибки сертификата при шифровании, отсутствие драйвера. Для устранения проверяют настройки сервера, сети, учётной записи и журналы сервера.

23. Что такое облачные базы данных?
Ответ: Облачные базы данных — это базы данных, размещённые на инфраструктуре облачного провайдера. Чаще всего они предоставляются как управляемый сервис (DBaaS, Database as a Service), при котором провайдер выполняет установку, обновление, резервное копирование, мониторинг и обеспечивает отказоустойчивость, а пользователь работает только с базой данных.

24. Какие облачные сервисы баз данных вы знаете?
Ответ: В России распространены Yandex Managed Service for PostgreSQL, MySQL, ClickHouse и MongoDB, управляемые базы данных VK Cloud, Selectel и Cloud.ru. Среди зарубежных сервисов известны Amazon RDS, Azure SQL Database, Google Cloud SQL и MongoDB Atlas.

25. Каковы преимущества и недостатки облачных баз данных?
Ответ: Преимущества: быстрое создание базы, отсутствие затрат на оборудование, автоматическое резервное копирование и обновления, простое масштабирование, встроенная отказоустойчивость и мониторинг. Недостатки: зависимость от провайдера и сети, постоянные расходы, ограниченный доступ к настройкам сервера, требования законодательства к месту хранения данных, например о локализации персональных данных граждан РФ.

26. Как подключиться к облачной базе данных?
Ответ: В консоли облачного провайдера создаётся кластер базы данных, пользователь и база. Для подключения получают адрес хоста и порт, настраивают доступ: разрешают публичный доступ или подключение из облачной сети, добавляют адрес клиента в группы безопасности. Загружают SSL-сертификат провайдера и указывают его в клиенте, так как облачные сервисы обычно требуют шифрованного соединения.

27. Что такое веб-интерфейсы администрирования баз данных?
Ответ: Веб-интерфейсы администрирования — это приложения, работающие в браузере и позволяющие управлять базами данных без установки клиента на компьютер пользователя. Примеры: phpMyAdmin для MySQL и MariaDB, Adminer для многих СУБД, pgAdmin в серверном режиме, Mongo Express, а также консоли управления облачных провайдеров со встроенным редактором SQL.

28. Какие меры безопасности необходимы при использовании веб-интерфейсов администрирования?
Ответ: Доступ к веб-интерфейсу должен осуществляться только по протоколу HTTPS, из доверенных сетей или через VPN, с надёжной аутентификацией, желательно многофакторной.

Необходимо своевременно обновлять приложение, не публиковать его открыто в интернете, использовать учётные записи с минимальными правами и вести журнал действий.

29. Как развернуть веб-интерфейс администрирования с помощью Docker?
Ответ: Веб-интерфейс запускается из официального образа, например dpage/pgadmin4 для pgAdmin, phpmyadmin для phpMyAdmin или adminer. Параметры задаются переменными окружения, например PGADMIN_DEFAULT_EMAIL и PGADMIN_DEFAULT_PASSWORD для pgAdmin или PMA_HOST для phpMyAdmin. Веб-интерфейс обычно описывается в docker-compose.yaml вместе с контейнером СУБД.

30. Как организовать сопровождение клиентского ПО в организации?
Ответ: Необходимо вести перечень установленного клиентского ПО и его версий, своевременно устанавливать обновления, использовать единые настройки и шаблоны подключений, централизованно управлять учётными записями и правами, обучать пользователей, документировать порядок установки и настройки, а также контролировать соблюдение лицензионных условий.

Критерии оценки устного опроса:

- оценка «отлично» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов и смог ответить на уточняющие вопросы;
- оценка «хорошо» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов, но не смог ответить на уточняющие вопросы;
- оценка «удовлетворительно» выставляется студенту, если он смог изложить суть основного вопроса, но не смог ответить по существу дополнительных вопросов;
- оценка «неудовлетворительно» выставляется студенту, если он не смог изложить суть основного вопроса.

Практическое занятие №6. Установка и настройка клиента SQL Workbench для работы с базой данных MySQL.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных
ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации
ПК 2.5. Подготавливать данные для базы знаний
ПК 2.6. Осуществлять управление базами данных

Цель: Научиться устанавливать и настраивать клиент MySQL Workbench, создавать подключения к серверу MySQL, выполнять запросы, работать с данными и моделью базы данных, а также проверять результаты настройки.

Задание:
1. Загрузить и установить MySQL Workbench.
2. Создать и протестировать подключение к серверу MySQL.
3. Настроить параметры SQL-редактора.
4. Выполнить SQL-запросы и работу с данными таблиц.
5. Построить ER-диаграмму базы данных методом обратного проектирования.
6. Выполнить экспорт и импорт данных.
7. Настроить подключение через SSH-туннель.
8. Проверить результаты настройки и сделать вывод.

Этапы выполнения:
Этап 1. Установка.
MySQL Workbench загружается с официального сайта MySQL или устанавливается через MySQL Installer в Windows, в Linux — из официального репозитория. Проверяется, что программа запускается без ошибок.

Этап 2. Создание подключения.
Создаётся новое подключение: задаются имя подключения, метод Standard (TCP/IP), адрес сервера, порт 3306, имя пользователя и база данных по умолчанию. Пароль сохраняется в защищённом хранилище. Подключение проверяется кнопкой Test Connection.

Этап 3. Настройка редактора.
В параметрах программы задаются ограничение количества строк в результатах запроса, режим безопасных обновлений, режим автоматической фиксации транзакций, время ожидания запросов и шрифт редактора.

Этап 4. Работа с запросами и данными.
В редакторе выполняются запросы на выборку, вставку, изменение и удаление данных. Проверяется редактирование данных в табличном виде и корректность отображения кириллицы.

Этап 5. Построение ER-диаграммы.
С помощью функции обратного проектирования (Reverse Engineer) по существующей базе данных строится ER-диаграмма со связями между таблицами.

Этап 6. Экспорт и импорт.
С помощью функций Data Export и Data Import выполняется экспорт базы данных в файл SQL и её восстановление в новую базу. Результаты запроса экспортируются в файл CSV.

Этап 7. Подключение через SSH.
Создаётся подключение методом Standard TCP/IP over SSH с указанием SSH-сервера, пользователя и ключа или пароля, а также адреса MySQL относительно SSH-сервера.

Этап 8. Оформление результата.
Оформляется отчёт, содержащий параметры подключений без паролей, выполненные запросы, ER-диаграмму, результаты экспорта и импорта и выводы.

Пример решения:
Задание: Установить и настроить MySQL Workbench для работы сотрудника с учебной базой данных библиотеки.

Дано: Сервер MySQL 8 с базой library, содержащей таблицы books, readers и loans, работает на компьютере 192.168.10.20. Сотруднику создан пользователь library_user с правами на чтение и изменение данных. Удалённый доступ к порту 3306 закрыт, доступен только SSH.

Решение по этапам:
1. Загружаем MySQL Workbench с официального сайта и устанавливаем на компьютер сотрудника.
2. Создаём подключение «Библиотека» методом Standard TCP/IP over SSH: SSH-сервер 192.168.10.20:22, пользователь SSH, адрес MySQL 127.0.0.1, порт 3306, пользователь library_user, база по умолчанию library.
3. Нажимаем Test Connection — подключение успешно, пароль сохраняем в защищённом хранилище.
4. В настройках SQL-редактора включаем режим безопасных обновлений и ограничиваем вывод 1000 строками.
5. Выполняем запрос SELECT title, author FROM books WHERE year > 2000; — данные на русском языке отображаются корректно.
6. Пытаемся выполнить UPDATE books SET year = 2024; без условия — Workbench блокирует запрос в режиме безопасных обновлений.
7. Выполняем Database → Reverse Engineer для базы library и получаем ER-диаграмму с тремя таблицами и связями loans — books и loans — readers.
8. Выполняем экспорт базы library в файл SQL через Data Export и экспорт результата запроса списка выданных книг в CSV.
9. Проверяем, что пользователь library_user не может создать новую базу — выводится ошибка доступа, что соответствует назначенным правам.
10. Оформляем отчёт.

Ответ: MySQL Workbench установлен и настроен для работы сотрудника с базой library через защищённое SSH-подключение. Включён режим безопасных обновлений и ограничение вывода строк. Проверены выполнение запросов, корректное отображение кириллицы, построение ER-диаграммы, экспорт данных и ограничения прав пользователя.

Критерии оценки практических заданий:
1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Практическое занятие №7. Установка и настройка pgAdmin для управления PostgreSQL.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных
ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации
ПК 2.5. Подготавливать данные для базы знаний

ПК 2.6. Осуществлять управление базами данных

Цель: Научиться устанавливать pgAdmin в настольном и серверном режимах, регистрировать серверы PostgreSQL, выполнять запросы, управлять объектами базы данных и ролями, выполнять резервное копирование и проверять результаты настройки.

Задание:
1. Установить pgAdmin в настольном режиме.
2. Развернуть pgAdmin в серверном режиме с помощью Docker.
3. Зарегистрировать сервер PostgreSQL.
4. Выполнить запросы в Query Tool и проанализировать план выполнения.
5. Создать объекты базы данных средствами pgAdmin.
6. Создать роль пользователя и назначить права.
7. Выполнить резервное копирование и восстановление базы данных.
8. Проверить результаты и сделать вывод.

Этапы выполнения:
Этап 1. Установка в настольном режиме.
pgAdmin загружается с официального сайта или устанавливается вместе с PostgreSQL. При первом запуске задаётся мастер-пароль для защиты сохранённых паролей подключений.

Этап 2. Развёртывание в серверном режиме.
pgAdmin запускается в контейнере из образа dpage/pgadmin4 с переменными окружения PGADMIN_DEFAULT_EMAIL и PGADMIN_DEFAULT_PASSWORD, опубликованным портом и томом для хранения настроек. Вход выполняется через браузер.

Этап 3. Регистрация сервера.
Через команду регистрации сервера указываются имя сервера, адрес, порт 5432, база данных для обслуживания, пользователь и пароль, при необходимости режим SSL и SSH-туннель. Проверяется отображение баз данных в обозревателе объектов.

Этап 4. Работа с Query Tool.
В редакторе запросов выполняются запросы, просматриваются результаты, строится план выполнения (Explain) для анализа производительности.

Этап 5. Создание объектов.
С помощью диалогов pgAdmin создаются база данных, схема, таблица со столбцами и ограничениями, индекс. Просматривается SQL-код, сформированный pgAdmin.

Этап 6. Роли и права.
Создаётся роль с правом входа и паролем, назначаются права на схему и таблицы через вкладку Privileges или команды GRANT. Выполняется подключение под новой ролью.

Этап 7. Резервное копирование.
С помощью функции Backup выполняется резервное копирование базы данных в файл, с помощью функции Restore — восстановление в новую базу. pgAdmin использует для этого утилиты pg_dump и pg_restore.

Этап 8. Оформление результата.
Оформляется отчёт, содержащий параметры регистрации сервера, выполненные запросы и план выполнения, созданные объекты, роли и права, результаты резервного копирования и выводы.

Пример решения:
Задание: Установить pgAdmin для команды разработчиков и настроить доступ к базе данных интернет-магазина.

Дано: Сервер PostgreSQL с базой shop работает в контейнере db. Разработчики должны работать с базой через браузер. Требуется роль разработчика с правами чтения и изменения данных в схеме public без права удаления таблиц.

Решение по этапам:
1. Добавляем в docker-compose.yaml сервис pgadmin на основе образа dpage/pgadmin4 с переменными PGADMIN_DEFAULT_EMAIL и PGADMIN_DEFAULT_PASSWORD, портом 5050:80 и томом pgadmin_data.
2. Запускаем docker compose up -d и открываем в браузере адрес http://localhost:5050, входим под учётной записью администратора pgAdmin.
3. Регистрируем сервер «Магазин»: адрес db, порт 5432, пользователь postgres. В обозревателе отображается база shop.
4. В Query Tool выполняем запрос SELECT * FROM orders WHERE customer_id = 100; и строим план выполнения — выполняется последовательное сканирование таблицы.
5. Создаём через диалог pgAdmin индекс по столбцу customer_id, повторно строим план — используется сканирование по индексу.
6. Создаём роль developer с правом входа и паролем, выполняем GRANT USAGE ON SCHEMA public TO developer; GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA public TO developer;
7. Регистрируем подключение под ролью developer и проверяем: чтение и изменение данных выполняются, команда DROP TABLE orders завершается ошибкой доступа.
8. Выполняем Backup базы shop в формате custom, затем Restore в новую базу shop_copy и сравниваем количество строк в таблицах — совпадает.
9. Проверяем, что после перезапуска контейнера pgadmin зарегистрированные серверы сохранились благодаря тому.
10. Оформляем отчёт.

Ответ: pgAdmin развёрнут в серверном режиме в Docker и доступен разработчикам через браузер, сервер PostgreSQL зарегистрирован. С помощью плана выполнения обнаружен и устранён недостаток индекса. Создана роль developer с правами изменения данных без права удаления таблиц, выполнено резервное копирование и восстановление базы. Проверка подтвердила корректность настройки.

Критерии оценки практических заданий:
1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Практическое занятие №8. Установка и настройка Microsoft Management Studio (SSMS) для работы с SQL Server.
Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных
ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по

защите информации
ПК 2.5. Подготавливать данные для базы знаний
ПК 2.6. Осуществлять управление базами данных

Цель: Научиться устанавливать и настраивать SQL Server Management Studio, подключаться к серверу с разными способами проверки подлинности, работать с обозревателем объектов и редактором запросов, использовать средства мониторинга и проверять результаты настройки.

Задание:
1. Загрузить и установить SSMS.
2. Подключиться к SQL Server с проверкой подлинности Windows и SQL Server.
3. Настроить параметры шифрования подключения.
4. Настроить параметры среды и редактора запросов.
5. Создать базу данных и таблицы через обозреватель объектов и с помощью T-SQL.
6. Использовать шаблоны запросов и зарегистрированные серверы.
7. Использовать монитор активности и план выполнения запроса.
8. Проверить результаты и сделать вывод.

Этапы выполнения:
Этап 1. Установка.
Установщик SSMS загружается с официального сайта Microsoft и запускается с правами администратора. После установки проверяется запуск программы.

Этап 2. Подключение к серверу.
В окне подключения выбирается тип сервера «Ядро СУБД», указывается имя сервера и экземпляра, способ проверки подлинности. Выполняется подключение с проверкой подлинности Windows и с логином SQL Server.

Этап 3. Настройка шифрования.
В параметрах подключения задаётся режим шифрования. Если на учебном сервере используется самоподписанный сертификат, включается параметр доверия сертификату сервера, а в рабочей среде на сервер устанавливается доверенный сертификат.

Этап 4. Настройка среды.
В параметрах среды настраиваются шрифт и цвет редактора, нумерация строк, вывод результатов в виде таблицы или текста, ограничение времени выполнения запроса, цветовая маркировка подключений к разным серверам.

Этап 5. Создание объектов.
Через обозреватель объектов создаётся база данных с указанием файлов данных и журнала. В редакторе запросов с помощью T-SQL создаются таблицы с первичными и внешними ключами, вставляются данные.

Этап 6. Шаблоны и зарегистрированные серверы.
С помощью обозревателя шаблонов создаются типовые запросы, например создание пользователя или процедуры. В окне зарегистрированных серверов добавляются серверы, к которым часто выполняется подключение.

Этап 7. Мониторинг и планы выполнения.
Открывается монитор активности для просмотра процессов, ожиданий и ресурсоёмких запросов. Для запроса включается отображение фактического плана выполнения и анализируются его операции.

Этап 8. Оформление результата.
Оформляется отчёт с параметрами подключений, настройками среды, кодом созданных объектов, снимками монитора активности и плана выполнения и выводами.

Пример решения:

Задание: Установить и настроить SSMS для администратора базы данных учёта заявок.
Дано: Экземпляр SQL Server работает на сервере SRV-DB01 в смешанном режиме проверки подлинности с самоподписанным сертификатом. Администратор работает с учётной записью Windows, приложение — с логином app_user.

Решение по этапам:
1. Загружаем и устанавливаем SSMS, запускаем программу.
2. Подключаемся к серверу SRV-DB01 с проверкой подлинности Windows — подключение успешно, в обозревателе объектов отображаются базы данных.
3. Повторно подключаемся с логином app_user и проверкой подлинности SQL Server. Возникает ошибка доверия сертификату, так как сертификат самоподписанный. Для учебного сервера включаем доверие сертификату сервера, подключение успешно, и фиксируем в отчёте необходимость установки доверенного сертификата в рабочей среде.
4. В параметрах среды включаем нумерацию строк и устанавливаем красный цвет строки состояния для подключений к серверу SRV-DB01.
5. Создаём через обозреватель объектов базу Requests, а в редакторе запросов — таблицы clients и requests с внешним ключом и тестовыми данными.
6. Добавляем SRV-DB01 в зарегистрированные серверы и с помощью шаблона создаём хранимую процедуру получения заявок клиента.
7. Включаем фактический план выполнения для запроса поиска заявок по номеру клиента — используется полное сканирование таблицы. Создаём индекс и повторяем запрос — используется поиск по индексу.
8. Открываем монитор активности, просматриваем активные сеансы и последние ресурсоёмкие запросы.
9. Проверяем, что под логином app_user в обозревателе объектов доступна только база Requests.
10. Оформляем отчёт.

Ответ: SSMS установлен и настроен для администратора: проверены подключения с проверкой подлинности Windows и SQL Server, настроено доверие сертификату для учебного сервера, цветовая маркировка подключений и параметры редактора. Созданы база данных, таблицы и процедура, проанализирован план выполнения и оптимизирован запрос индексом, использован монитор активности.

Критерии оценки практических заданий:
1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Практическое занятие №9. Установка и настройка DBeaver для подключения к различным типам баз данных.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных

ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации
ПК 2.5. Подготавливать данные для базы знаний
ПК 2.6. Осуществлять управление базами данных

Цель: Научиться устанавливать и настраивать универсальный клиент DBeaver, создавать подключения к различным СУБД, управлять драйверами, настраивать типы подключений и безопасность, выполнять запросы, сравнивать и переносить данные между базами.

Задание:
1. Загрузить и установить DBeaver Community.
2. Создать подключения к MySQL, PostgreSQL и SQL Server.
3. Проверить загрузку и настройку драйверов.
4. Настроить типы подключений для тестовой и рабочей баз.
5. Настроить подключение через SSH-туннель и с шифрованием SSL.
6. Выполнить запросы и построить ER-диаграммы.
7. Выполнить перенос данных из одной СУБД в другую.
8. Проверить результаты и сделать вывод.

Этапы выполнения:
Этап 1. Установка.
DBeaver Community загружается с официального сайта и устанавливается. Для его работы используется встроенная среда Java. Проверяется запуск программы.

Этап 2. Создание подключений.
С помощью мастера нового подключения выбирается тип СУБД, указываются адрес сервера, порт, база данных, пользователь и пароль. Подключения группируются по папкам, например «Учебные» и «Рабочие».

Этап 3. Драйверы.
При первом подключении DBeaver предлагает загрузить драйвер JDBC. Проверяется версия драйвера в менеджере драйверов, при необходимости он обновляется.

Этап 4. Типы подключений.
Для каждого подключения задаётся тип: разработка, тестирование или рабочая база. Для рабочей базы включаются цветовая маркировка, подтверждение выполнения запросов и

ручная фиксация транзакций, а также режим только чтения, если пользователю не нужно изменять данные.

Этап 5. SSH и SSL.
На вкладке SSH настраивается подключение через SSH-сервер. На вкладке SSL включается шифрование соединения и указывается сертификат.

Этап 6. Запросы и ER-диаграммы.
В SQL-редакторе выполняются запросы к разным базам, используется автодополнение. Для базы данных строится ER-диаграмма.

Этап 7. Перенос данных.
С помощью функции экспорта данных таблица из одной СУБД переносится в другую с сопоставлением столбцов и типов данных. Результаты запросов экспортируются в CSV и Excel.

Этап 8. Оформление результата.
Оформляется отчёт, содержащий перечень подключений и их параметры без паролей, настройки типов подключений, результаты запросов, ER-диаграммы, результаты переноса данных и выводы.

Пример решения:
Задание: Настроить DBeaver для аналитика, работающего с базами данных разных отделов.
Дано: Отдел продаж использует MySQL (база sales), склад — PostgreSQL (база warehouse), бухгалтерия — SQL Server (база finance). Базы sales и warehouse являются рабочими, аналитику требуется только чтение. Кроме того, имеется тестовая база PostgreSQL analytics_test, в которую нужно перенести справочник товаров из MySQL.

Решение по этапам:
1. Устанавливаем DBeaver Community.
2. Создаём подключения к sales (MySQL), warehouse (PostgreSQL), finance (SQL Server) и analytics_test (PostgreSQL), при первом подключении загружаем драйверы.
3. Для подключений sales, warehouse и finance задаём тип «Рабочая» с красной маркировкой, включаем режим только чтения и подтверждение выполнения запросов.
4. Для analytics_test задаём тип «Разработка» с зелёной маркировкой и автоматической фиксацией.
5. Для подключения к finance настраиваем SSH-туннель через сервер бухгалтерии, так как порт 1433 недоступен из сети аналитиков.
6. Выполняем в sales запрос итоговых продаж за месяц — результат получен. Попытка выполнить UPDATE в sales блокируется режимом только чтения.
7. Строим ER-диаграмму базы warehouse для изучения структуры.
8. С помощью функции экспорта переносим таблицу products из MySQL в analytics_test с сопоставлением типов: столбец типа DATETIME преобразован в timestamp, количество строк в исходной и целевой таблицах совпадает — 1250.
9. Экспортируем результат запроса продаж в Excel для отчёта.
10. Оформляем отчёт.

Ответ: DBeaver установлен и настроен для подключения к MySQL, PostgreSQL и SQL Server из одного интерфейса. Рабочие базы защищены режимом только чтения и цветовой маркировкой, подключение к базе бухгалтерии выполняется через SSH-туннель. Справочник

товаров успешно перенесён из MySQL в PostgreSQL, построена ER-диаграмма и выполнен экспорт данных.

Критерии оценки практических заданий:
1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Практическое занятие №10. Установка и настройка библиотек Python для взаимодействия с базами данных (pymysql, psycopg2).
Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных
ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации
ПК 2.5. Подготавливать данные для базы знаний
ПК 2.6. Осуществлять управление базами данных

Цель: Научиться устанавливать и настраивать библиотеки PyMySQL и psycopg2 для работы с MySQL и PostgreSQL из программ на Python, безопасно хранить параметры подключения, выполнять параметризованные запросы, управлять транзакциями и обрабатывать ошибки.

Задание:
1. Создать виртуальное окружение Python.
2. Установить библиотеки PyMySQL и psycopg2.
3. Настроить хранение параметров подключения в переменных окружения.
4. Разработать функции подключения к MySQL и PostgreSQL.
5. Выполнить параметризованные запросы на чтение и изменение данных.
6. Реализовать управление транзакциями и обработку ошибок.
7. Выполнить перенос данных между базами с помощью Python.
8. Проверить результаты и сделать вывод.

Этапы выполнения:
Этап 1. Виртуальное окружение.
Создаётся виртуальное окружение командой python -m venv venv и активируется. Это изолирует зависимости проекта от системных пакетов.

Этап 2. Установка библиотек.
Выполняется установка командой pip install pymysql psycopg2-binary python-dotenv. Пакет psycopg2-binary содержит скомпилированные компоненты и не требует установки

компилятора, для рабочей среды может использоваться пакет psycopg2. Список зависимостей сохраняется командой pip freeze > requirements.txt.

Этап 3. Параметры подключения.
Параметры подключения (адрес, порт, база, пользователь, пароль) записываются в файл .env, который добавляется в .gitignore. В программе они загружаются с помощью python-dotenv и читаются функцией os.getenv().

Этап 4. Функции подключения.
Разрабатываются функции, создающие подключения: pymysql.connect() с параметрами host, port, user, password, database, charset="utf8mb4" и psycopg2.connect() с параметрами host, port, dbname, user, password. Проверяется подключение запросом версии сервера.

Этап 5. Параметризованные запросы.
Выполняются запросы с передачей параметров отдельно от текста запроса через заполнитель %s. Результаты получаются методами fetchone() и fetchall(). Для получения строк в виде словарей используется курсор DictCursor в PyMySQL и RealDictCursor в psycopg2.

Этап 6. Транзакции и ошибки.
Изменения фиксируются методом commit(), при ошибке выполняется rollback(). Используются конструкции with и блоки try/except для обработки исключений подключения и выполнения запросов, а подключения закрываются после работы.

Этап 7. Перенос данных.
Разрабатывается скрипт, который читает данные из MySQL и пакетно вставляет их в PostgreSQL методом executemany() или функцией execute_values() из psycopg2.extras.

Этап 8. Оформление результата.
Оформляется отчёт, содержащий команды установки, файл requirements.txt, код программы без паролей, результаты выполнения и проверки данных и выводы.

Пример решения:
Задание: Разработать программу на Python для выгрузки новых заказов из базы MySQL интернет-магазина в аналитическую базу PostgreSQL.
Дано: В MySQL в базе shop есть таблица orders (id, customer_id, amount, created_at). В PostgreSQL в базе analytics есть таблица orders_copy с такой же структурой. Требуется ежедневно переносить заказы за указанную дату.

Решение по этапам:
1. Создаём и активируем виртуальное окружение venv.
2. Устанавливаем библиотеки командой pip install pymysql psycopg2-binary python-dotenv и сохраняем requirements.txt.
3. Создаём файл .env с параметрами MYSQL_HOST, MYSQL_USER, MYSQL_PASSWORD, PG_HOST, PG_USER, PG_PASSWORD и другими, добавляем его в .gitignore.
4. Разрабатываем функции get_mysql_conn() и get_pg_conn(), читающие параметры из переменных окружения. Проверяем подключения запросами SELECT VERSION(); — оба сервера отвечают.
5. Разрабатываем функцию чтения заказов из MySQL с параметризованным запросом: cursor.execute("SELECT id, customer_id, amount, created_at FROM orders WHERE DATE(created_at) = %s", (date,)).
6. Разрабатываем функцию записи в PostgreSQL с использованием execute_values() и конструкции ON CONFLICT (id) DO NOTHING, чтобы повторный запуск не создавал дубликатов.
7. Оборачиваем запись в транзакцию: при успехе вызывается commit(), при исключении — rollback() с записью ошибки в журнал.

8. Проверяем защиту от SQL-инъекции: передаём в качестве даты строку «2026-10-01' OR '1'='1» — запрос не выполняется как вредоносный, возникает ошибка преобразования даты.
9. Запускаем программу за дату 2026-10-01: перенесено 342 заказа, количество записей в обеих базах за эту дату совпадает. Повторный запуск не создаёт дубликатов.
10. Оформляем отчёт.

Ответ: В виртуальном окружении установлены и настроены библиотеки PyMySQL и psycopg2, параметры подключения хранятся в переменных окружения. Разработана программа переноса заказов из MySQL в PostgreSQL с параметризованными запросами, пакетной вставкой, управлением транзакциями и защитой от дубликатов. Проверка подтвердила корректность переноса и защиту от SQL-инъекций.

Критерии оценки практических заданий:
1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Тема 1.3. Управление доступом к базам данных.
Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных
ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации
ПК 2.5. Подготавливать данные для базы знаний
ПК 2.6. Осуществлять управление базами данных

Вопросы к устному опросу по теме:
1. Что такое управление доступом к базам данных?
Ответ: Управление доступом к базам данных — это совокупность мер, определяющих, какие пользователи могут подключаться к базе данных и какие действия с какими объектами им разрешено выполнять. Оно включает идентификацию и аутентификацию пользователей, назначение, изменение и отзыв прав, контроль соблюдения прав и аудит действий.

2. Чем аутентификация отличается от авторизации?
Ответ: Аутентификация — это проверка подлинности пользователя, то есть подтверждение того, что он тот, за кого себя выдаёт, например по паролю, сертификату или учётной записи операционной системы. Авторизация — это определение того, какие действия разрешено выполнять уже аутентифицированному пользователю. Сначала выполняется аутентификация, затем авторизация.

3. Что такое привилегии в СУБД и какие виды привилегий существуют?
Ответ: Привилегия — это право на выполнение определённого действия в базе данных. Системные привилегии дают право на действия на уровне сервера или базы, например создание баз данных, пользователей, таблиц. Объектные привилегии дают право на действия

с конкретными объектами: SELECT, INSERT, UPDATE, DELETE для таблиц, EXECUTE для процедур, а также права на отдельные столбцы.

4. Что такое роль в СУБД и зачем она нужна?
Ответ: Роль — это именованный набор привилегий, который можно назначать пользователям. Вместо назначения прав каждому пользователю отдельно права назначаются роли, а пользователи включаются в роль. Это упрощает управление доступом, уменьшает количество ошибок и позволяет быстро изменять права целой группы пользователей.

5. Что такое принцип минимальных привилегий?
Ответ: Принцип минимальных привилегий означает, что каждый пользователь и каждое приложение получают только те права, которые необходимы для выполнения их задач. Например, приложению для отчётов предоставляется только право чтения нужных таблиц. Это снижает ущерб от ошибок, злоупотреблений и взлома учётной записи.

6. Как назначаются права доступа с помощью команды GRANT?
Ответ: Команда GRANT предоставляет привилегии или роли пользователям и ролям. Синтаксис: GRANT привилегии ON объект TO пользователь. Например, GRANT SELECT, INSERT ON shop.orders TO 'manager'@'localhost'; в MySQL или GRANT SELECT ON orders TO analyst; в PostgreSQL. Также командой GRANT назначаются роли: GRANT analyst_role TO ivanov.

7. Как отзываются права доступа с помощью команды REVOKE?
Ответ: Команда REVOKE отзывает ранее предоставленные привилегии или роли. Синтаксис: REVOKE привилегии ON объект FROM пользователь. Например, REVOKE DELETE ON orders FROM manager. Изменение прав в большинстве СУБД вступает в силу для новых операций без перезапуска сервера.

8. Что означает параметр WITH GRANT OPTION?
Ответ: Параметр WITH GRANT OPTION позволяет пользователю, получившему привилегию, передавать её другим пользователям. Использовать его следует осторожно, так как это затрудняет контроль распределения прав. При отзыве привилегии с параметром CASCADE отзываются и права, переданные этим пользователем другим.

9. Что такое команда DENY в Microsoft SQL Server?
Ответ: Команда DENY явно запрещает пользователю или роли выполнение действия. Запрет имеет приоритет над разрешениями, полученными через другие роли. Например, если пользователь входит в роль db_datareader, но для него выполнена команда DENY SELECT ON salaries, он не сможет читать таблицу зарплат. Команда REVOKE, в отличие от DENY, только удаляет ранее выданное разрешение или запрет.

10. Как создаются пользователи и роли в MySQL?
Ответ: Пользователь создаётся командой CREATE USER 'имя'@'хост' IDENTIFIED BY 'пароль', где хост определяет, с каких адресов разрешено подключение. В MySQL 8 роли создаются командой CREATE ROLE, назначаются командой GRANT роль TO пользователь, а для автоматической активации роли при входе используется команда SET DEFAULT ROLE. Права пользователя просматриваются командой SHOW GRANTS.

11. Как устроена система ролей в PostgreSQL?
Ответ: В PostgreSQL пользователи и группы реализованы единым понятием роли. Роль с атрибутом LOGIN может подключаться и используется как пользователь, роль без LOGIN используется как группа. Роли могут быть членами других ролей и наследовать их права. Роли

создаются командой CREATE ROLE, членство назначается командой GRANT группа TO пользователь.

12. Что такое права по умолчанию в PostgreSQL?
Ответ: Права по умолчанию задаются командой ALTER DEFAULT PRIVILEGES и автоматически назначаются на объекты, которые будут созданы в будущем. Например, ALTER DEFAULT PRIVILEGES IN SCHEMA public GRANT SELECT ON TABLES TO analysts; гарантирует, что аналитики получат право чтения новых таблиц без повторного выполнения GRANT.

13. Какие встроенные роли существуют в Oracle Database?
Ответ: В Oracle есть встроенные роли CONNECT (право подключения к базе данных), RESOURCE (права на создание таблиц, процедур и других объектов в своей схеме), DBA (полные административные права), SELECT_CATALOG_ROLE (просмотр словаря данных) и другие. В новых версиях есть роль DB_DEVELOPER_ROLE для разработчиков. Роль DBA следует назначать только администраторам.

14. Какие встроенные роли существуют в Microsoft SQL Server?
Ответ: Серверные роли: sysadmin (полные права на сервере), serveradmin (настройка сервера), securityadmin (управление логинами), dbcreator (создание баз данных) и другие. Роли базы данных: db_owner (полные права в базе), db_datareader (чтение всех таблиц), db_datawriter (изменение данных), db_ddladmin (изменение структуры), db_securityadmin (управление ролями и правами), db_backupoperator (резервное копирование).

15. Как изменить права доступа пользователя при изменении его обязанностей?
Ответ: Необходимо определить новые обязанности, исключить пользователя из ролей, которые ему больше не нужны, включить в новые роли, отозвать отдельные лишние привилегии командой REVOKE, проверить итоговые права и зафиксировать изменение в журнале изменений. При увольнении сотрудника учётная запись блокируется или удаляется, а принадлежащие ей объекты передаются другому владельцу.

16. Как заблокировать учётную запись пользователя базы данных?
Ответ: В MySQL используется команда ALTER USER ... ACCOUNT LOCK, в PostgreSQL — ALTER ROLE ... NOLOGIN, в Oracle — ALTER USER ... ACCOUNT LOCK, в SQL Server — ALTER LOGIN ... DISABLE. Блокировка сохраняет учётную запись и её объекты, но запрещает подключение. Также можно задать автоматическую блокировку после нескольких неудачных попыток входа.

17. Как просмотреть права пользователей в различных СУБД?
Ответ: В MySQL используется команда SHOW GRANTS FOR пользователь. В PostgreSQL — команды psql \du и \dp, функция has_table_privilege() и представление information_schema.role_table_grants. В Oracle — представления DBA_ROLE_PRIVS, DBA_SYS_PRIVS и DBA_TAB_PRIVS. В SQL Server — функция fn_my_permissions, представления sys.database_permissions и sys.database_role_members, а также вкладка «Действующие разрешения» в SSMS.

18. Как организуется контроль соблюдения прав доступа?
Ответ: Контроль включает регулярный пересмотр прав пользователей и ролей, выявление учётных записей с избыточными правами и неиспользуемых учётных записей, проверку выполнения запрещённых действий от имени пользователей, ведение и анализ журналов

аудита, контроль неудачных попыток входа и изменений прав, а также сравнение фактических прав с утверждённой матрицей доступа.

19. Что такое матрица доступа?
Ответ: Матрица доступа — это таблица, в которой строки соответствуют пользователям или ролям, столбцы — объектам базы данных, а на пересечении указываются разрешённые действия. Матрица утверждается руководством и служит основой для настройки прав в СУБД и для проверки соблюдения прав при аудите.

20. Что такое SQL-инъекция?
Ответ: SQL-инъекция — это атака, при которой злоумышленник внедряет фрагмент SQL-кода через входные данные приложения, если они подставляются в запрос без проверки. Например, ввод ' OR '1'='1 в поле логина может позволить войти без пароля. С помощью инъекций можно читать, изменять и удалять данные или выполнять административные команды.

21. Какие меры защищают от SQL-инъекций?
Ответ: Основная мера — использование параметризованных запросов, при которых данные передаются отдельно от текста запроса. Также применяются хранимые процедуры без динамического SQL, проверка и ограничение входных данных, ORM-библиотеки, минимальные права учётной записи приложения, скрытие подробных сообщений об ошибках и межсетевые экраны веб-приложений.

22. Почему минимальные права учётной записи приложения снижают последствия SQL-инъекции?
Ответ: Даже если злоумышленник смог внедрить SQL-код, он выполняется с правами учётной записи приложения. Если у неё есть только права чтения и записи нужных таблиц, злоумышленник не сможет удалить таблицы, создать пользователей, прочитать другие базы или выполнить системные команды. Ущерб от атаки существенно ограничивается.

23. Что такое аудит безопасности базы данных?
Ответ: Аудит безопасности — это регистрация и анализ событий, связанных с безопасностью: подключений и неудачных попыток входа, изменения пользователей и прав, изменения структуры базы, доступа к конфиденциальным данным и действий администраторов. Аудит позволяет обнаружить нарушения, расследовать инциденты и подтвердить соблюдение требований.

24. Какие средства аудита есть в разных СУБД?
Ответ: В PostgreSQL используются параметры журналирования log_connections, log_statement и расширение pgaudit. В MySQL — общий журнал запросов и плагины аудита. В Oracle — единый аудит (Unified Auditing) с политиками CREATE AUDIT POLICY и представлением UNIFIED_AUDIT_TRAIL. В SQL Server — механизм SQL Server Audit со спецификациями аудита сервера и базы данных.

25. Что такое шифрование данных при хранении?
Ответ: Шифрование при хранении защищает файлы данных, журналы и резервные копии, чтобы при краже носителя данные нельзя было прочитать. Применяется прозрачное шифрование данных TDE в SQL Server и Oracle, шифрование табличных пространств InnoDB в MySQL, шифрование на уровне файловой системы или диска. Для отдельных столбцов применяется шифрование функциями, например расширением pgcrypto в PostgreSQL или механизмом Always Encrypted в SQL Server.

26. Что такое шифрование соединения с базой данных?
Ответ: Шифрование соединения защищает данные, передаваемые между клиентом и сервером, от перехвата с помощью протоколов SSL/TLS. На сервере настраиваются сертификат и закрытый ключ, а также можно требовать обязательного шифрования для всех

или отдельных пользователей, например параметром require_secure_transport в MySQL, правилами hostssl в pg_hba.conf PostgreSQL или параметром REQUIRE SSL для пользователя MySQL.

27. Какие методы аутентификации используются в MySQL?
Ответ: В MySQL 8 по умолчанию используется метод caching_sha2_password, обеспечивающий надёжное хранение паролей. Устаревший метод mysql_native_password не рекомендуется. Также поддерживаются аутентификация через сокет операционной системы (auth_socket), а в коммерческой редакции — LDAP и Kerberos. Политика паролей задаётся компонентом validate_password.

28. Что такое построчная безопасность (Row-Level Security)?
Ответ: Построчная безопасность — это механизм, ограничивающий доступ пользователей к отдельным строкам таблицы в зависимости от условия, например пользователь видит только записи своего отдела. В SQL Server она реализуется функцией-предикатом и командой CREATE SECURITY POLICY, в PostgreSQL — командами ALTER TABLE ... ENABLE ROW LEVEL SECURITY и CREATE POLICY, в Oracle — механизмом Virtual Private Database.

29. Что такое политика паролей и какие требования она включает?
Ответ: Политика паролей — это набор правил для паролей пользователей: минимальная длина, обязательное использование букв разного регистра, цифр и специальных символов, запрет повторного использования прежних паролей, срок действия пароля, блокировка учётной записи после нескольких неудачных попыток входа. В Oracle политика задаётся профилями, в MySQL — компонентом validate_password и параметрами ALTER USER.

30. Какие параметры конфигурационного файла MySQL влияют на безопасность доступа?
Ответ: На безопасность влияют параметры bind-address (адрес прослушивания), skip-networking (запрет сетевых подключений), require_secure_transport (обязательное шифрование), local_infile (загрузка локальных файлов), secure_file_priv (каталог для операций с файлами), skip-name-resolve (отключение разрешения имён), параметры политики паролей validate_password и параметры журналирования. Параметр skip-grant-tables, отключающий проверку прав, недопустим в рабочей среде.

Критерии оценки устного опроса:

- оценка «отлично» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов и смог ответить на уточняющие вопросы;
- оценка «хорошо» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов, но не смог ответить на уточняющие вопросы;
- оценка «удовлетворительно» выставляется студенту, если он смог изложить суть основного вопроса, но не смог ответить по существу дополнительных вопросов;
- оценка «неудовлетворительно» выставляется студенту, если он не смог изложить суть основного вопроса.

Практическое занятие №11. Создание пользователей и групп в MySQL и назначение прав доступа (GRANT, REVOKE). Настройка ролей и прав доступа в PostgreSQL для различных пользователей.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке

Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных
ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации
ПК 2.5. Подготавливать данные для базы знаний
ПК 2.6. Осуществлять управление базами данных

Цель: Научиться создавать пользователей и группы (роли) в MySQL и PostgreSQL, назначать и отзывать права доступа командами GRANT и REVOKE, использовать роли для группового управления правами и проверять действующие права пользователей.

Задание:
1. Составить матрицу доступа для пользователей учебной базы данных.
2. Создать пользователей и роли в MySQL.
3. Назначить роли права доступа и включить в них пользователей в MySQL.
4. Изменить и отозвать права доступа командой REVOKE в MySQL.
5. Создать групповые роли и роли пользователей в PostgreSQL.
6. Назначить права на схемы, таблицы и права по умолчанию в PostgreSQL.
7. Проверить действующие права пользователей в обеих СУБД.
8. Сделать вывод о соответствии прав матрице доступа.

Этапы выполнения:
Этап 1. Составление матрицы доступа.
Определяются категории пользователей, объекты базы данных и разрешённые действия. Результат оформляется в виде таблицы.

Этап 2. Создание пользователей и ролей в MySQL.
Командой CREATE ROLE создаются роли, соответствующие группам пользователей. Командой CREATE USER создаются пользователи с указанием хоста подключения и пароля.

Этап 3. Назначение прав в MySQL.
Командой GRANT ролям назначаются привилегии на базы данных и таблицы, затем роли назначаются пользователям. Командой SET DEFAULT ROLE роли делаются активными при входе.

Этап 4. Изменение прав в MySQL.
С помощью команды REVOKE у роли или пользователя отзываются отдельные привилегии, проверяется применение изменений командой SHOW GRANTS.

Этап 5. Создание ролей в PostgreSQL.
Создаются групповые роли без права входа (NOLOGIN) и роли пользователей с правом входа (LOGIN) и паролем. Пользователи включаются в групповые роли командой GRANT.

Этап 6. Назначение прав в PostgreSQL.
Групповым ролям назначаются права CONNECT на базу данных, USAGE на схему, права на таблицы и последовательности. Командой ALTER DEFAULT PRIVILEGES задаются права на будущие таблицы. У роли PUBLIC отзываются лишние права.

Этап 7. Проверка прав.
От имени каждого пользователя выполняются разрешённые и запрещённые действия. Права проверяются командами SHOW GRANTS в MySQL, \du и \dp и функцией has_table_privilege() в PostgreSQL.

Этап 8. Оформление результата.
Оформляется отчёт, содержащий матрицу доступа, выполненные команды, результаты проверки и выводы.

Пример решения:
Задание: Настроить доступ к базе данных интернет-магазина для менеджеров, аналитиков и стажёра.

Дано: База данных shop содержит таблицы products, orders и customers. Менеджеры читают и изменяют заказы и товары, читают клиентов. Аналитики только читают все таблицы. Стажёр читает только товары. Используются MySQL 8 (база shop) и PostgreSQL (база shop, схема public).

Решение по этапам:
1. Составляем матрицу доступа: managers — SELECT, INSERT, UPDATE для orders и products, SELECT для customers; analysts — SELECT для всех таблиц; стажёр — SELECT для products.
2. В MySQL создаём роли и пользователей: CREATE ROLE 'managers', 'analysts'; CREATE USER 'petrov'@'%' IDENTIFIED BY '...'; CREATE USER 'sidorova'@'%' IDENTIFIED BY '...'; CREATE USER 'intern'@'%' IDENTIFIED BY '...';
3. Назначаем права ролям: GRANT SELECT, INSERT, UPDATE ON shop.orders TO 'managers'; GRANT SELECT, INSERT, UPDATE ON shop.products TO 'managers'; GRANT SELECT ON shop.customers TO 'managers'; GRANT SELECT ON shop.* TO 'analysts'; стажёру — GRANT SELECT ON shop.products TO 'intern'@'%';
4. Назначаем роли и делаем их активными: GRANT 'managers' TO 'petrov'@'%'; GRANT 'analysts' TO 'sidorova'@'%'; SET DEFAULT ROLE ALL TO 'petrov'@'%', 'sidorova'@'%';
5. По решению руководства менеджерам запрещено добавлять товары: REVOKE INSERT ON shop.products FROM 'managers'; Проверяем командой SHOW GRANTS FOR 'petrov'@'%' USING 'managers'; — право INSERT на products отсутствует.
6. В PostgreSQL создаём групповые роли: CREATE ROLE managers NOLOGIN; CREATE ROLE analysts NOLOGIN; и пользователей: CREATE ROLE petrov LOGIN PASSWORD '...' IN ROLE managers; CREATE ROLE sidorova LOGIN PASSWORD '...' IN ROLE analysts; CREATE ROLE intern LOGIN PASSWORD '...';
7. Назначаем права: REVOKE ALL ON DATABASE shop FROM PUBLIC; GRANT CONNECT ON DATABASE shop TO managers, analysts, intern; GRANT USAGE ON SCHEMA public TO managers, analysts, intern; GRANT SELECT, INSERT, UPDATE ON orders TO managers; GRANT SELECT, UPDATE ON products TO managers; GRANT SELECT ON customers TO managers; GRANT SELECT ON ALL TABLES IN SCHEMA public TO analysts; ALTER DEFAULT PRIVILEGES IN SCHEMA public GRANT SELECT ON TABLES TO analysts; GRANT SELECT ON products TO intern;
8. Проверяем в PostgreSQL: SELECT has_table_privilege('intern', 'customers', 'SELECT'); возвращает false, has_table_privilege('petrov', 'orders', 'UPDATE') — true. Под пользователем sidorova команда DELETE FROM orders завершается ошибкой доступа.
9. Создаём новую таблицу returns и проверяем, что аналитики сразу могут её читать благодаря правам по умолчанию.
10. Сравниваем фактические права с матрицей доступа и оформляем отчёт.

Ответ: В MySQL и PostgreSQL созданы пользователи и групповые роли менеджеров и аналитиков, права назначены в соответствии с матрицей доступа, у менеджеров отозвано право добавления товаров. В PostgreSQL настроены права по умолчанию для новых таблиц и

отозваны права роли PUBLIC. Проверка подтвердила, что каждый пользователь может выполнять только разрешённые действия.

Критерии оценки практических заданий:
1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Практическое занятие №12. Управление правами доступа в Microsoft SQL Server с использованием SQL Server Management Studio (SSMS). Настройка аутентификации и шифрования соединения в MySQL.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных
ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации
ПК 2.5. Подготавливать данные для базы знаний
ПК 2.6. Осуществлять управление базами данных

Цель: Научиться управлять логинами, пользователями, ролями и разрешениями в Microsoft SQL Server с помощью SSMS, а также настраивать методы аутентификации, политику паролей и обязательное шифрование соединений в MySQL.

Задание:
1. Создать логины и пользователей базы данных в SQL Server через SSMS.
2. Назначить встроенные и пользовательские роли базы данных.
3. Назначить, запретить и отозвать разрешения на объекты через вкладку «Защищаемые объекты».
4. Проверить действующие разрешения пользователей в SQL Server.
5. Настроить метод аутентификации и политику паролей в MySQL.
6. Настроить SSL/TLS-шифрование на сервере MySQL.
7. Настроить обязательное шифрование соединений для пользователей MySQL и проверить его.
8. Сделать вывод о результатах настройки.

Этапы выполнения:
Этап 1. Логины и пользователи в SQL Server.
В SSMS в разделе «Безопасность → Имена входа» создаются логины SQL Server и Windows. В разделе «Безопасность → Пользователи» нужной базы данных создаются пользователи, сопоставленные логинам.

Этап 2. Роли базы данных.
В свойствах пользователя на странице «Членство» назначаются встроенные роли базы данных. В разделе «Роли базы данных» создаются пользовательские роли и добавляются участники.

Этап 3. Разрешения на объекты.
В свойствах роли или пользователя на странице «Защищаемые объекты» выбираются таблицы, представления и процедуры и отмечаются разрешения: предоставить, предоставить с правом передачи или запретить. Проверяется SQL-код, формируемый SSMS кнопкой «Скрипт».

Этап 4. Проверка разрешений.
Используется кнопка «Действующие разрешения» в свойствах объекта, запросы к функции fn_my_permissions, а также выполнение запросов от имени пользователя с помощью EXECUTE AS USER и возврат командой REVERT.

Этап 5. Аутентификация в MySQL.
Проверяется метод аутентификации пользователей запросом к таблице mysql.user. Настраивается компонент validate_password, для пользователей задаются срок действия пароля и блокировка после неудачных попыток входа командой ALTER USER.

Этап 6. Настройка SSL/TLS в MySQL.
Проверяется наличие сертификатов сервера (MySQL 8 создаёт их автоматически) и значения параметров ssl_ca, ssl_cert, ssl_key. В конфигурационном файле включается параметр require_secure_transport = ON, служба перезапускается.

Этап 7. Проверка шифрования.
Для отдельных пользователей задаётся требование ALTER USER ... REQUIRE SSL. Выполняется подключение клиентом с параметром --ssl-mode=REQUIRED и проверка

командой SHOW SESSION STATUS LIKE 'Ssl_cipher'. Проверяется, что подключение без шифрования (--ssl-mode=DISABLED) отклоняется.

Этап 8. Оформление результата.
Оформляется отчёт, содержащий снимки настроек SSMS, сформированные скрипты, результаты проверки разрешений, параметры аутентификации и шифрования MySQL и выводы.

Пример решения:
Задание: Настроить доступ к базе данных учёта заявок в SQL Server и защитить подключения к серверу MySQL сайта компании.
Дано: В SQL Server база Requests содержит таблицы clients, requests и процедуру close_request. Операторы должны читать заявки и закрывать их только через процедуру. На сервере MySQL работают база сайта и пользователь web_app, подключающийся с сервера приложения.

Решение по этапам:
1. В SSMS создаём логин SQL Server operator1 с проверкой политики паролей и пользователя operator1 в базе Requests.
2. Создаём роль базы данных operators и добавляем в неё пользователя operator1.
3. На странице «Защищаемые объекты» роли operators предоставляем SELECT на таблицы clients и requests и EXECUTE на процедуру close_request, запрещаем UPDATE и DELETE на таблицу requests. Сохраняем скрипт: GRANT SELECT ON dbo.requests TO operators; GRANT EXECUTE ON dbo.close_request TO operators; DENY UPDATE, DELETE ON dbo.requests TO operators;
4. Проверяем: EXECUTE AS USER = 'operator1'; SELECT * FROM fn_my_permissions('dbo.requests', 'OBJECT'); REVERT; — отображаются только разрешения SELECT. От имени operator1 прямое изменение заявки завершается ошибкой, а вызов процедуры close_request успешно закрывает заявку.
5. В MySQL проверяем метод аутентификации: SELECT user, host, plugin FROM mysql.user; — у web_app используется caching_sha2_password.
6. Устанавливаем компонент проверки паролей и задаём для пользователя политику: ALTER USER 'web_app'@'10.0.0.15' PASSWORD EXPIRE INTERVAL 90 DAY FAILED_LOGIN_ATTEMPTS 5 PASSWORD_LOCK_TIME 1;
7. Проверяем параметры SSL: SHOW VARIABLES LIKE 'ssl%'; — сертификаты сервера присутствуют. В my.cnf добавляем require_secure_transport = ON и перезапускаем службу.
8. Задаём требование шифрования для пользователя: ALTER USER 'web_app'@'10.0.0.15' REQUIRE SSL;
9. С сервера приложения подключаемся с параметром --ssl-mode=REQUIRED, команда SHOW SESSION STATUS LIKE 'Ssl_cipher'; показывает используемый шифр. Подключение с --ssl-mode=DISABLED отклоняется сервером.
10. Оформляем отчёт.
Ответ: В SQL Server с помощью SSMS создана роль operators, которой разрешены чтение заявок и их закрытие только через хранимую процедуру, прямое изменение запрещено командой DENY. Проверка через EXECUTE AS и fn_my_permissions подтвердила корректность прав. В MySQL для пользователя web_app настроены надёжный метод

аутентификации, срок действия пароля и блокировка при подборе, включено обязательное шифрование соединений, незашифрованные подключения отклоняются.

Критерии оценки практических заданий:
1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Практическое занятие №13. Использование встроенных ролей в Oracle Database для управления доступом. Конфигурация прав доступа для разных уровней пользователей (администратор, аналитик, пользователь) в PostgreSQL.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных
ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации
ПК 2.5. Подготавливать данные для базы знаний
ПК 2.6. Осуществлять управление базами данных

Цель: Научиться использовать встроенные роли Oracle Database для назначения типовых наборов прав, просматривать состав ролей в словаре данных, а также настраивать многоуровневую модель доступа (администратор, аналитик, пользователь) в PostgreSQL.

Задание:
1. Изучить состав встроенных ролей Oracle по представлениям словаря данных.
2. Создать пользователей Oracle и назначить им встроенные роли в соответствии с задачами.
3. Проверить права пользователей Oracle и ограничить избыточные права.
4. Определить уровни пользователей и их права в PostgreSQL.
5. Создать групповые роли администратора, аналитика и пользователя в PostgreSQL.
6. Назначить права на схемы, таблицы и представления для каждого уровня.
7. Проверить доступ пользователей каждого уровня.
8. Сделать вывод о результатах настройки.

Этапы выполнения:
Этап 1. Изучение встроенных ролей Oracle.
С помощью запросов к представлениям DBA_ROLES, ROLE_SYS_PRIVS и DBA_SYS_PRIVS изучается, какие системные привилегии входят во встроенные роли CONNECT, RESOURCE, DBA и SELECT_CATALOG_ROLE.

Этап 2. Создание пользователей Oracle.
В подключаемой базе данных создаются пользователи командой CREATE USER с паролем и табличным пространством. Назначаются встроенные роли в соответствии с задачами:

CONNECT для подключения, RESOURCE для разработчиков, SELECT_CATALOG_ROLE для специалиста по мониторингу. Роль DBA назначается только администратору.

Этап 3. Проверка прав в Oracle.
Права проверяются запросами к DBA_ROLE_PRIVS и DBA_SYS_PRIVS и представлению SESSION_ROLES под пользователем. Выявляются избыточные права, например привилегия UNLIMITED TABLESPACE, и отзываются командой REVOKE. Для пользователей задаются квоты на табличное пространство.

Этап 4. Определение уровней в PostgreSQL.
Определяются права каждого уровня: администратор базы — управление объектами схемы; аналитик — чтение всех таблиц и создание объектов в собственной схеме для расчётов; пользователь — чтение и изменение отдельных таблиц через приложение.

Этап 5. Создание групповых ролей.
Создаются групповые роли db_admins, db_analysts и db_users без права входа и учётные записи сотрудников с правом входа, включённые в соответствующие группы.

Этап 6. Назначение прав.
Администраторам назначается владение схемой или права на создание и изменение объектов, аналитикам — права чтения и права на собственную схему, пользователям — права на отдельные таблицы и представления. Задаются права по умолчанию.

Этап 7. Проверка доступа.
От имени пользователя каждого уровня выполняются разрешённые и запрещённые действия, результаты фиксируются в таблице.

Этап 8. Оформление результата.
Оформляется отчёт, содержащий состав встроенных ролей Oracle, команды назначения, модель уровней доступа PostgreSQL, результаты проверки и выводы.

Пример решения:
Задание: Настроить доступ к учебной базе данных отдела продаж в Oracle и PostgreSQL для сотрудников разных уровней.
Дано: В Oracle в подключаемой базе FREEPDB1 работают администратор Смирнов, разработчик Козлов и специалист по мониторингу Орлова. В PostgreSQL база sales со схемой sales содержит таблицы orders, customers и представление v_sales_summary. Уровни: администратор Смирнов, аналитик Иванова, пользователь-менеджер Петров.

Решение по этапам:
1. Выполняем SELECT privilege FROM role_sys_privs WHERE role = 'RESOURCE'; — роль содержит права CREATE TABLE, CREATE SEQUENCE, CREATE PROCEDURE и другие. Роль CONNECT содержит только CREATE SESSION.
2. Создаём пользователей Oracle: CREATE USER smirnov IDENTIFIED BY "..."; CREATE USER kozlov IDENTIFIED BY "..." DEFAULT TABLESPACE users QUOTA 200M ON users; CREATE USER orlova IDENTIFIED BY "...";
3. Назначаем роли: GRANT DBA TO smirnov; GRANT CONNECT, RESOURCE TO kozlov; GRANT CONNECT, SELECT_CATALOG_ROLE TO orlova;
4. Проверяем: SELECT grantee, granted_role FROM dba_role_privs WHERE grantee IN ('SMIRNOV','KOZLOV','ORLOVA'); Под пользователем orlova запрос к v$session выполняется, создание таблицы — ошибка недостаточных привилегий. Под kozlov создание таблицы выполняется в пределах квоты.
5. В PostgreSQL создаём роли: CREATE ROLE db_admins NOLOGIN; CREATE ROLE db_analysts NOLOGIN; CREATE ROLE db_users NOLOGIN; CREATE ROLE smirnov LOGIN

PASSWORD '...' IN ROLE db_admins; CREATE ROLE ivanova LOGIN PASSWORD '...' IN ROLE db_analysts; CREATE ROLE petrov LOGIN PASSWORD '...' IN ROLE db_users;
6. Назначаем права администраторам: ALTER SCHEMA sales OWNER TO db_admins; Аналитикам: GRANT USAGE ON SCHEMA sales TO db_analysts; GRANT SELECT ON ALL TABLES IN SCHEMA sales TO db_analysts; CREATE SCHEMA analytics AUTHORIZATION db_analysts;
7. Пользователям: GRANT USAGE ON SCHEMA sales TO db_users; GRANT SELECT, INSERT, UPDATE ON sales.orders TO db_users; GRANT SELECT ON sales.v_sales_summary TO db_users; Права по умолчанию: ALTER DEFAULT PRIVILEGES FOR ROLE db_admins IN SCHEMA sales GRANT SELECT ON TABLES TO db_analysts;
8. Проверяем: petrov добавляет заказ — успешно, читает customers — ошибка доступа; ivanova читает все таблицы и создаёт таблицу в схеме analytics — успешно, изменение orders — ошибка; smirnov создаёт новую таблицу в схеме sales, и ivanova сразу может читать её.
9. Заносим результаты проверки в таблицу «пользователь — действие — ожидаемый и фактический результат», расхождений нет.
10. Оформляем отчёт.

Ответ: В Oracle изучен состав встроенных ролей и назначены роли CONNECT, RESOURCE, SELECT_CATALOG_ROLE и DBA в соответствии с задачами сотрудников, для разработчика задана квота. В PostgreSQL реализована трёхуровневая модель доступа с групповыми ролями администраторов, аналитиков и пользователей и правами по умолчанию. Проверка подтвердила соответствие фактических прав модели доступа.

Критерии оценки практических заданий:
1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Практическое занятие №14. Создание политики безопасности в Microsoft SQL Server для ограничения действий пользователей. Проверка и настройка доступа к базе данных через файл конфигурации в MySQL.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных
ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации
ПК 2.5. Подготавливать данные для базы знаний
ПК 2.6. Осуществлять управление базами данных

Цель: Научиться создавать политики безопасности в Microsoft SQL Server для ограничения доступа пользователей к строкам таблиц и их действий, а также проверять и настраивать параметры безопасности доступа к серверу MySQL в конфигурационном файле.

Задание:
1. Определить требования к ограничению действий пользователей в SQL Server.
2. Создать функцию-предикат для построчной безопасности.
3. Создать политику безопасности с предикатами фильтрации и блокировки.
4. Дополнить политику запретами на опасные действия и маскированием данных.
5. Проверить работу политики от имени разных пользователей.
6. Проверить параметры безопасности в конфигурационном файле MySQL.
7. Изменить параметры конфигурации MySQL и проверить их применение.
8. Сделать вывод о результатах настройки.

Этапы выполнения:
Этап 1. Определение требований.
Определяется, какие строки таблиц и какие действия должны быть доступны каждому пользователю, например менеджер видит и изменяет только записи своих клиентов.

Этап 2. Функция-предикат.
Создаётся встроенная табличная функция с параметром WITH SCHEMABINDING, которая возвращает строку, если доступ к записи разрешён, например при совпадении значения столбца с именем текущего пользователя USER_NAME() или при принадлежности пользователя к роли руководителей.

Этап 3. Политика безопасности.
Командой CREATE SECURITY POLICY создаётся политика, в которую добавляются предикат фильтрации (FILTER PREDICATE), скрывающий чужие строки при чтении, и предикат блокировки (BLOCK PREDICATE), запрещающий добавлять и изменять чужие строки. Политика включается параметром STATE = ON.

Этап 4. Дополнительные ограничения.
Для ограничения действий используются команды DENY, например запрет удаления строк и изменения структуры, а для конфиденциальных столбцов включается динамическое маскирование данных (MASKED WITH).

Этап 5. Проверка политики.
С помощью EXECUTE AS USER выполняются запросы от имени разных пользователей, проверяется, какие строки они видят и какие изменения могут выполнить.

Этап 6. Проверка конфигурации MySQL.
Определяется расположение конфигурационного файла и проверяются параметры: bind-address, skip-networking, require_secure_transport, local_infile, secure_file_priv, skip-name-

resolve, параметры журналирования. Проверяется отсутствие параметра skip-grant-tables. Проверяются права доступа к самому файлу конфигурации и к файлам с паролями клиентов.

Этап 7. Изменение конфигурации.
В конфигурационный файл вносятся необходимые изменения, служба перезапускается, применение параметров проверяется командой SHOW VARIABLES и попытками подключения.

Этап 8. Оформление результата.
Оформляется отчёт, содержащий код функции и политики безопасности, результаты проверки от имени пользователей, исходные и изменённые параметры конфигурации MySQL, результаты проверки и выводы.

Пример решения:
Задание: Ограничить доступ менеджеров к данным клиентов в SQL Server и усилить защиту сервера MySQL.
Дано: В SQL Server в базе CRM таблица dbo.clients содержит столбцы id, name, phone, manager_login. Менеджеры manager1 и manager2 должны видеть и изменять только своих клиентов, руководитель из роли heads — всех. Удалять клиентов никто, кроме администратора, не может, телефон должен маскироваться для менеджеров. Сервер MySQL используется только приложением на том же сервере.

Решение по этапам:
1. Создаём схему security и функцию-предикат: CREATE FUNCTION security.fn_client_access(@manager_login sysname) RETURNS TABLE WITH SCHEMABINDING AS RETURN SELECT 1 AS result WHERE @manager_login = USER_NAME() OR IS_MEMBER('heads') = 1;
2. Создаём политику: CREATE SECURITY POLICY security.ClientsPolicy ADD FILTER PREDICATE security.fn_client_access(manager_login) ON dbo.clients, ADD BLOCK PREDICATE security.fn_client_access(manager_login) ON dbo.clients AFTER INSERT, ADD BLOCK PREDICATE security.fn_client_access(manager_login) ON dbo.clients AFTER UPDATE WITH (STATE = ON);
3. Запрещаем удаление: DENY DELETE ON dbo.clients TO managers; и включаем маскирование телефона: ALTER TABLE dbo.clients ALTER COLUMN phone ADD MASKED WITH (FUNCTION = 'partial(2,"XXXXXXX",2)'); Руководителям выдаём право UNMASK.
4. Проверяем: EXECUTE AS USER = 'manager1'; SELECT * FROM dbo.clients; REVERT; — отображаются только клиенты manager1 с замаскированными телефонами. Попытка manager1 добавить клиента с manager_login = 'manager2' блокируется. Попытка удалить клиента завершается ошибкой доступа. Руководитель видит всех клиентов с полными телефонами.
5. В MySQL определяем расположение конфигурации командой mysqld --verbose --help | grep -A 1 "Default options" и открываем файл /etc/mysql/mysql.conf.d/mysqld.cnf.
6. Выявляем недостатки: bind-address = 0.0.0.0 (сервер доступен из сети), local_infile = ON, параметр secure_file_priv пуст.
7. Изменяем параметры: bind-address = 127.0.0.1, local_infile = 0, secure_file_priv = /var/lib/mysql-files, skip-name-resolve, log_error = /var/log/mysql/error.log. Убеждаемся, что skip-grant-tables отсутствует. Проверяем, что файл конфигурации принадлежит root и недоступен для записи другим пользователям, а файл ~/.my.cnf с паролем имеет права 600.
8. Перезапускаем службу и проверяем: SHOW VARIABLES LIKE 'bind_address'; — 127.0.0.1, SHOW VARIABLES LIKE 'local_infile'; — OFF. Попытка подключения к MySQL с другого компьютера отклоняется, приложение на сервере работает.
9. Фиксируем исходные и новые значения параметров в таблице.
10. Оформляем отчёт.

Ответ: В SQL Server создана политика безопасности ClientsPolicy с предикатами фильтрации и блокировки, благодаря которой менеджеры видят и изменяют только своих клиентов,

удаление запрещено командой DENY, а телефоны замаскированы. В конфигурации MySQL выявлены и устранены небезопасные параметры: сервер прослушивает только локальный адрес, отключена загрузка локальных файлов, ограничен каталог файловых операций. Проверка подтвердила применение всех ограничений.

Критерии оценки практических заданий:
1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Практическое занятие №15. Реализация сценария управления доступом через роли и группы пользователей в Oracle Database. Аудит действий пользователей в базе данных с помощью встроенных инструментов PostgreSQL.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных
ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации
ПК 2.5. Подготавливать данные для базы знаний
ПК 2.6. Осуществлять управление базами данных

Цель: Научиться реализовывать сценарий управления доступом в Oracle Database с помощью пользовательских ролей, вложенных ролей и профилей, а также настраивать аудит действий пользователей в PostgreSQL с помощью параметров журналирования и триггеров аудита.

Задание:
1. Разработать сценарий управления доступом для групп пользователей в Oracle.
2. Создать пользовательские роли, в том числе вложенные, и назначить им привилегии.
3. Создать профиль с политикой паролей и назначить его пользователям.
4. Проверить доступ и смоделировать изменение обязанностей сотрудника.
5. Настроить параметры журналирования действий пользователей в PostgreSQL.
6. Создать таблицу и триггеры аудита изменений данных в PostgreSQL.
7. Выполнить действия пользователей и проанализировать журналы аудита.
8. Сделать вывод о результатах.

Этапы выполнения:
Этап 1. Разработка сценария.
Определяются группы пользователей, их обязанности, необходимые права на объекты и иерархия ролей, например роль старшего специалиста включает роль специалиста и дополнительные права.

Этап 2. Создание ролей в Oracle.
Командой CREATE ROLE создаются роли групп, командой GRANT им назначаются объектные и системные привилегии, роли включаются друг в друга. Роли назначаются

пользователям, при необходимости задаются роли по умолчанию командой ALTER USER ... DEFAULT ROLE.

Этап 3. Профиль безопасности.
Командой CREATE PROFILE создаётся профиль с ограничениями: количество неудачных попыток входа, время блокировки, срок действия пароля, запрет повторного использования паролей. Профиль назначается пользователям командой ALTER USER ... PROFILE.

Этап 4. Проверка и изменение доступа.
От имени пользователей выполняются разрешённые и запрещённые действия, просматривается представление SESSION_ROLES. Моделируется перевод сотрудника: у него отзывается одна роль и назначается другая, права проверяются повторно.

Этап 5. Журналирование в PostgreSQL.
В postgresql.conf или командой ALTER SYSTEM настраиваются параметры log_connections, log_disconnections, log_statement (например, ddl или mod) и log_line_prefix с указанием времени, пользователя, базы и адреса клиента. Конфигурация перечитывается функцией pg_reload_conf().

Этап 6. Триггеры аудита.
Создаётся таблица аудита, доступная для записи только через функцию триггера. Создаётся функция триггера с параметром SECURITY DEFINER, записывающая имя таблицы, операцию, пользователя, время, старые и новые значения в формате JSON. Триггеры создаются для контролируемых таблиц.

Этап 7. Анализ аудита.
Под разными пользователями выполняются изменения данных и структуры, а также неудачные попытки входа. Анализируются журнал сервера и таблица аудита: кто, когда и какие действия выполнял.

Этап 8. Оформление результата.
Оформляется отчёт, содержащий сценарий доступа, команды создания ролей и профиля, результаты проверки, настройки журналирования, код триггеров аудита, выдержки из журналов и выводы.

Пример решения:
Задание: Реализовать управление доступом к базе данных бухгалтерии в Oracle и аудит изменений платёжных данных в PostgreSQL.
Дано: В Oracle в схеме acc таблицы invoices и payments. Бухгалтеры читают счета и платежи и добавляют платежи, старший бухгалтер дополнительно изменяет и удаляет платежи. В PostgreSQL база billing содержит таблицу payments, изменения которой должны фиксироваться с указанием пользователя.

Решение по этапам:
1. Создаём роли в Oracle: CREATE ROLE accountant; CREATE ROLE senior_accountant; GRANT SELECT ON acc.invoices TO accountant; GRANT SELECT, INSERT ON acc.payments TO accountant; GRANT accountant TO senior_accountant; GRANT UPDATE, DELETE ON acc.payments TO senior_accountant;
2. Создаём профиль: CREATE PROFILE acc_profile LIMIT FAILED_LOGIN_ATTEMPTS 3 PASSWORD_LOCK_TIME 1 PASSWORD_LIFE_TIME 90 PASSWORD_REUSE_MAX 5;
3. Создаём пользователей: CREATE USER kuznetsova IDENTIFIED BY "..." PROFILE acc_profile; CREATE USER volkov IDENTIFIED BY "..." PROFILE acc_profile; GRANT

CREATE SESSION TO kuznetsova, volkov; GRANT accountant TO kuznetsova; GRANT senior_accountant TO volkov;
4. Проверяем: kuznetsova добавляет платёж — успешно, удаление платежа — ошибка недостаточных привилегий. volkov удаляет ошибочный платёж — успешно. SELECT * FROM session_roles; под volkov показывает роли SENIOR_ACCOUNTANT и ACCOUNTANT. Три неверных ввода пароля блокируют учётную запись kuznetsova.
5. Моделируем повышение kuznetsova: REVOKE accountant FROM kuznetsova; GRANT senior_accountant TO kuznetsova; — после повторного подключения ей доступно удаление платежей.
6. В PostgreSQL выполняем: ALTER SYSTEM SET log_connections = on; ALTER SYSTEM SET log_disconnections = on; ALTER SYSTEM SET log_statement = 'ddl'; ALTER SYSTEM SET log_line_prefix = '%m [%p] %u@%d %r '; SELECT pg_reload_conf(); Параметры подключения применяются к новым сеансам.
7. Создаём таблицу audit_log (id, table_name, operation, user_name, changed_at, old_data jsonb, new_data jsonb) и запрещаем пользователям доступ к ней. Создаём функцию audit_trigger() с SECURITY DEFINER, вставляющую в audit_log значения TG_TABLE_NAME, TG_OP, session_user, now(), to_jsonb(OLD), to_jsonb(NEW), и триггер: CREATE TRIGGER payments_audit AFTER INSERT OR UPDATE OR DELETE ON payments FOR EACH ROW EXECUTE FUNCTION audit_trigger();
8. Под пользователем cashier изменяем сумму платежа и удаляем платёж, под пользователем admin добавляем столбец в таблицу, выполняем вход с неверным паролем.
9. Анализируем: в audit_log записаны UPDATE и DELETE с пользователем cashier и старыми и новыми значениями, в журнале сервера — команда ALTER TABLE от admin и неудачная попытка входа с адресом клиента. Отмечаем, что для расширенного аудита чтения данных может использоваться расширение pgaudit.
10. Оформляем отчёт.

Ответ: В Oracle реализован сценарий управления доступом через роли accountant и senior_accountant с вложенностью ролей и профиль acc_profile с политикой паролей, проверено изменение доступа при переводе сотрудника. В PostgreSQL настроено журналирование подключений и изменений структуры и создан триггер аудита, фиксирующий изменения платежей с указанием пользователя и значений. Анализ журналов позволил точно установить, кто и какие действия выполнял.

Критерии оценки практических заданий:
1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Тема 1.4. Резервное копирование баз данных.
Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных
ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по

защите информации
ПК 2.5. Подготавливать данные для базы знаний
ПК 2.6. Осуществлять управление базами данных

Вопросы к устному опросу по теме:
1. Что такое резервное копирование базы данных?
Ответ: Резервное копирование базы данных — это создание копии данных, структуры и при необходимости журналов транзакций базы данных, которая хранится отдельно от рабочей базы и позволяет восстановить данные после сбоя оборудования, ошибки пользователя, удаления данных, повреждения файлов, атаки вируса-шифровальщика или другого инцидента.

2. Для чего необходимо резервное копирование?
Ответ: Резервное копирование необходимо для защиты от потери данных, обеспечения непрерывности работы организации, восстановления после ошибок и атак, выполнения требований законодательства и регламентов о хранении данных, а также для переноса баз данных на другой сервер и создания тестовых копий.

3. Чем логическое резервное копирование отличается от физического?
Ответ: Логическое резервное копирование сохраняет данные в виде SQL-команд или переносимого формата, позволяющего воссоздать объекты и данные, например с помощью mysqldump, pg_dump, mongodump. Физическое резервное копирование копирует файлы данных и журналов на уровне хранения, например с помощью RMAN, pg_basebackup или BACKUP DATABASE в SQL Server. Физическое копирование быстрее для больших баз, логическое удобнее для переноса и выборочного восстановления.

4. Какие виды резервного копирования по объёму данных существуют?
Ответ: Существуют полное резервное копирование (копируются все данные), дифференциальное (копируются изменения с момента последней полной копии) и инкрементальное (копируются изменения с момента последней резервной копии любого вида). Также выполняется резервное копирование журналов транзакций для восстановления на определённый момент времени.

5. Чем дифференциальное резервное копирование отличается от инкрементального?
Ответ: Дифференциальная копия содержит все изменения с момента последней полной копии, поэтому её размер растёт со временем, но для восстановления нужны только полная и последняя дифференциальная копии. Инкрементальная копия содержит изменения только с момента предыдущей копии, поэтому она меньше и создаётся быстрее, но для восстановления нужна полная копия и вся цепочка инкрементальных копий.

6. Что такое горячее и холодное резервное копирование?
Ответ: Холодное резервное копирование выполняется при остановленной базе данных, что гарантирует согласованность файлов, но требует простоя. Горячее (онлайн) резервное копирование выполняется во время работы базы данных без остановки пользователей. Для его согласованности используются транзакционные механизмы и журналы, например режим ARCHIVELOG в Oracle или параметр --single-transaction в mysqldump.

7. Что такое RPO и RTO?
Ответ: RPO (Recovery Point Objective) — допустимая потеря данных, выраженная во времени, например не более 15 минут изменений. RTO (Recovery Time Objective) — допустимое время

восстановления работы системы после сбоя, например не более 2 часов. Эти показатели определяют частоту и вид резервного копирования и выбор способа восстановления.

8. Что такое восстановление на момент времени (Point-in-Time Recovery)?
Ответ: Восстановление на момент времени — это восстановление базы данных в состояние на определённый момент, например за минуту до ошибочного удаления данных. Для этого восстанавливается полная резервная копия и последовательно применяются журналы транзакций до нужного момента. Такая возможность есть в PostgreSQL с архивированием WAL, в Oracle с архивными журналами, в SQL Server с копиями журнала транзакций, в MySQL с двоичными журналами.

9. Как запускается процедура резервного копирования в MySQL?
Ответ: Логическое резервное копирование выполняется утилитой mysqldump, например mysqldump -u backup_user -p --single-transaction --routines --triggers --events shop > shop.sql. Параметр --single-transaction обеспечивает согласованную копию таблиц InnoDB без блокировок, --routines, --triggers и --events включают в копию процедуры, триггеры и события. Для всех баз используется параметр --all-databases.

10. Чем утилита pg_dump отличается от pg_dumpall в PostgreSQL?
Ответ: Утилита pg_dump создаёт резервную копию одной базы данных и поддерживает форматы: текстовый SQL, custom, directory с параллельной выгрузкой и tar. Утилита pg_dumpall создаёт копию всех баз данных кластера, а также глобальных объектов — ролей и табличных пространств — только в текстовом формате SQL. Параметр pg_dumpall --globals-only выгружает только роли и табличные пространства.

11. Какие форматы резервных копий pg_dump существуют и как они восстанавливаются?
Ответ: Текстовый формат (-Fp) содержит SQL-команды и восстанавливается утилитой psql. Формат custom (-Fc) сжимается, позволяет выбирать объекты для восстановления и восстанавливается утилитой pg_restore. Формат directory (-Fd) хранит каждую таблицу в отдельном файле и поддерживает параллельную выгрузку и загрузку параметром -j. Формат tar (-Ft) также восстанавливается через pg_restore.

12. Какие модели восстановления существуют в Microsoft SQL Server?
Ответ: В SQL Server существуют модели восстановления SIMPLE (простая), FULL (полная) и BULK_LOGGED (с неполным протоколированием). В простой модели журнал транзакций автоматически усекается и копии журнала не выполняются, поэтому восстановление возможно только на момент резервной копии. В полной модели выполняются копии журнала транзакций и возможно восстановление на момент времени.

13. Какие виды резервных копий поддерживает SQL Server?
Ответ: SQL Server поддерживает полные резервные копии базы данных (BACKUP DATABASE), дифференциальные копии (BACKUP DATABASE ... WITH DIFFERENTIAL), копии журнала транзакций (BACKUP LOG), копии отдельных файлов и файловых групп, а также копии только для копирования (COPY_ONLY), не нарушающие последовательность плановых копий.

14. Как выполняется резервное копирование MongoDB?
Ответ: Логическое резервное копирование выполняется утилитой mongodump, а восстановление — утилитой mongorestore. Например, mongodump --uri="mongodb://..." --gzip --archive=shop.gz. Для набора реплик используется параметр --oplog, обеспечивающий

согласованность копии на момент окончания выгрузки. Для больших баз применяются снимки файловой системы или облачные средства резервного копирования.

15. Что такое RMAN?
Ответ: RMAN (Recovery Manager) — это встроенная утилита Oracle Database для резервного копирования, восстановления и проверки базы данных. RMAN выполняет полные и инкрементальные копии, копирует архивные журналы и управляющие файлы, проверяет блоки данных на повреждения, ведёт каталог резервных копий и автоматически удаляет устаревшие копии в соответствии с политикой хранения.

16. Что такое режим ARCHIVELOG в Oracle и зачем он нужен?
Ответ: В режиме ARCHIVELOG заполненные журналы повторного выполнения (redo) сохраняются в архивные журналы до их перезаписи. Это позволяет выполнять горячее резервное копирование без остановки базы и восстанавливать базу на любой момент времени. В режиме NOARCHIVELOG возможно только холодное резервное копирование и восстановление на момент копии.

17. Как выполняется мониторинг процедуры резервного копирования?
Ответ: Отслеживается ход выполнения, скорость, размер копии и ошибки. В SQL Server прогресс отображается параметром STATS и в представлении sys.dm_exec_requests (столбец percent_complete). В Oracle используются представления V$SESSION_LONGOPS и V$RMAN_BACKUP_JOB_DETAILS. Для утилит mysqldump, pg_dump и mongodump используется подробный режим вывода (--verbose) и журналирование вывода скриптов. Также отслеживаются загрузка дисков и свободное место.

18. Как контролировать успешное завершение резервного копирования?
Ответ: Проверяется код завершения утилиты (0 — успешно), отсутствие ошибок в журнале, наличие и ненулевой размер файла копии, сведения о копии в истории СУБД (например, в msdb.dbo.backupset в SQL Server или командой LIST BACKUP в RMAN). По результатам отправляется уведомление администратору, а отсутствие свежей копии считается инцидентом.

19. Что такое верификация резервных копий и зачем она нужна?
Ответ: Верификация — это проверка того, что резервная копия не повреждена и из неё можно восстановить данные. Без проверки копия может оказаться непригодной именно в момент аварии. Проверка включает сверку контрольных сумм, проверку структуры копии средствами СУБД и пробное восстановление на тестовом сервере.

20. Какие средства проверки резервных копий есть в разных СУБД?
Ответ: В SQL Server используются параметр WITH CHECKSUM при создании копии и команда RESTORE VERIFYONLY. В Oracle — команды RMAN VALIDATE DATABASE, RESTORE DATABASE VALIDATE и CROSSCHECK BACKUP. В PostgreSQL — команда pg_restore --list для копий pg_dump, утилита pg_verifybackup для копий pg_basebackup. Для всех СУБД применяются контрольные суммы файлов (sha256sum) и пробное восстановление.

21. Почему пробное восстановление считается наиболее надёжной проверкой резервной копии?
Ответ: Только фактическое восстановление подтверждает, что копия полна, не повреждена, совместима с версией СУБД, включает все необходимые объекты и права, а процедура восстановления выполнима за допустимое время. После восстановления проверяются

количество записей, целостность связей и работа приложения. Пробное восстановление следует выполнять регулярно по графику.

22. Что такое ротация резервных копий?
Ответ: Ротация — это порядок хранения и удаления резервных копий по заданным правилам, чтобы хранить нужное количество копий за разные периоды и не переполнять хранилище. Например, хранятся ежедневные копии за 7 дней, еженедельные за 4 недели и ежемесячные за 12 месяцев, а более старые копии удаляются автоматически.

23. Что такое схема ротации «дед — отец — сын» (GFS)?
Ответ: Схема GFS (Grandfather — Father — Son) предусматривает три уровня копий: ежедневные («сыновья»), которые хранятся неделю, еженедельные («отцы»), которые хранятся месяц, и ежемесячные («деды»), которые хранятся год и дольше. Схема позволяет восстановить данные за длительный период при ограниченном количестве хранимых копий.

24. Что такое правило 3-2-1?
Ответ: Правило 3-2-1 означает, что необходимо иметь не менее трёх копий данных (рабочие данные и две резервные копии), хранить их не менее чем на двух разных типах носителей и хотя бы одну копию хранить за пределами основной площадки, например в другом здании или облаке. Современный вариант правила дополнительно предусматривает неизменяемую или отключённую от сети копию для защиты от шифровальщиков.

25. Какие требования предъявляются к хранению резервных копий?
Ответ: Копии хранятся отдельно от рабочего сервера, защищаются от несанкционированного доступа и изменения, шифруются при хранении и передаче, хранятся в течение установленного срока, имеют понятные имена с датой и описание. Доступ к хранилищу копий предоставляется ограниченному кругу лиц, а пароли и ключи шифрования хранятся отдельно от копий.

26. Как политика хранения реализуется в RMAN?
Ответ: Политика хранения задаётся командой CONFIGURE RETENTION POLICY: окном восстановления (RECOVERY WINDOW OF 7 DAYS) или количеством копий (REDUNDANCY 2). Команда REPORT OBSOLETE показывает копии, не нужные по политике, команда DELETE OBSOLETE удаляет их, а команда CROSSCHECK проверяет фактическое наличие копий на носителях.

27. Как автоматизировать резервное копирование?
Ответ: Резервное копирование автоматизируется с помощью скриптов, запускаемых по расписанию планировщиком cron в Linux или планировщиком заданий Windows, заданиями агента SQL Server и планами обслуживания, заданиями планировщика Oracle или RMAN-скриптами, а также специализированными системами резервного копирования. Скрипт создаёт копию, проверяет результат, выполняет ротацию, журналирует действия и отправляет уведомления.

28. Как безопасно передавать пароль утилитам резервного копирования в автоматических скриптах?
Ответ: Пароль не указывается в тексте скрипта и в командной строке, так как он будет виден в списке процессов и журналах. Для MySQL используется файл ~/.my.cnf с разделом [mysqldump] и правами 600 или утилита mysql_config_editor, для PostgreSQL — файл ~/.pgpass с правами 600, для Oracle — аутентификация средствами операционной системы. Для резервного копирования создаётся отдельный пользователь с минимальными правами.

29. Что такое стратегия резервного копирования и восстановления?
Ответ: Стратегия резервного копирования и восстановления — это документ, определяющий, какие данные копируются, какими способами, с какой частотой, где и сколько хранятся копии,

как проверяется их пригодность, кто отвечает за процедуры и как выполняется восстановление. Стратегия основывается на значениях RPO и RTO, объёме данных и требованиях бизнеса.

30. Какие разделы включает план восстановления базы данных?
Ответ: План восстановления включает перечень баз данных и их критичность, значения RPO и RTO, расположение резервных копий, пошаговые инструкции восстановления для разных сценариев (сбой диска, удаление данных, полная потеря сервера), ответственных лиц и контакты, порядок проверки восстановленной базы и уведомления пользователей, а также график учебных восстановлений.

Критерии оценки устного опроса:

- оценка «отлично» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов и смог ответить на уточняющие вопросы;
- оценка «хорошо» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов, но не смог ответить на уточняющие вопросы;
- оценка «удовлетворительно» выставляется студенту, если он смог изложить суть основного вопроса, но не смог ответить по существу дополнительных вопросов;
- оценка «неудовлетворительно» выставляется студенту, если он не смог изложить суть основного вопроса.

Практическое занятие №16. Создание резервной копии базы данных MySQL с использованием утилиты mysqldump. Резервное копирование базы данных PostgreSQL с помощью pg_dump и pg_dumpall.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных
ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации
ПК 2.5. Подготавливать данные для базы знаний
ПК 2.6. Осуществлять управление базами данных

Цель: Научиться создавать логические резервные копии баз данных MySQL с помощью mysqldump и баз данных PostgreSQL с помощью pg_dump и pg_dumpall, выбирать параметры

и форматы копий, контролировать завершение копирования и проверять возможность восстановления.

Задание:
1. Создать пользователя для резервного копирования с минимально необходимыми правами.
2. Выполнить полную резервную копию базы данных MySQL с помощью mysqldump.
3. Выполнить резервные копии отдельных таблиц и только структуры базы MySQL.
4. Выполнить резервную копию базы данных PostgreSQL с помощью pg_dump в разных форматах.
5. Выполнить резервную копию всего кластера PostgreSQL и глобальных объектов с помощью pg_dumpall.
6. Проконтролировать завершение копирования и рассчитать контрольные суммы файлов.
7. Восстановить базы данных из резервных копий в тестовые базы и сравнить данные.
8. Сделать вывод о результатах.

Этапы выполнения:
Этап 1. Подготовка.
В MySQL создаётся пользователь для резервного копирования с правами SELECT, SHOW VIEW, TRIGGER, EVENT, LOCK TABLES и PROCESS. Пароль сохраняется в файле ~/.my.cnf

с правами 600. В PostgreSQL копирование выполняется от имени владельца базы или суперпользователя, пароль сохраняется в файле ~/.pgpass.

Этап 2. Полная копия MySQL.
Выполняется команда mysqldump с параметрами --single-transaction, --routines, --triggers, --events и сжатием результата утилитой gzip. Имя файла включает название базы и дату.

Этап 3. Выборочные копии MySQL.
Выполняется копирование отдельных таблиц указанием их имён после имени базы и копирование только структуры параметром --no-data.

Этап 4. Копии PostgreSQL с pg_dump.
Выполняется копирование базы в текстовом формате (-Fp) и в формате custom (-Fc) с параметром --verbose. Для большой базы используется формат directory (-Fd) с параллельной выгрузкой (-j).

Этап 5. Копия кластера PostgreSQL.
Командой pg_dumpall создаётся копия всех баз данных кластера, а командой pg_dumpall --globals-only — копия ролей и табличных пространств.

Этап 6. Контроль завершения.
Проверяется код завершения команд (echo $?), размер файлов, отсутствие ошибок в выводе, наличие в конце SQL-файла mysqldump строки «Dump completed». Для каждого файла рассчитывается контрольная сумма командой sha256sum.

Этап 7. Восстановление и сравнение.
Копия MySQL восстанавливается в тестовую базу командой mysql, копия PostgreSQL формата custom — командой pg_restore в новую базу. Сравнивается количество записей в основных таблицах исходной и восстановленной баз.

Этап 8. Оформление результата.
Оформляется отчёт, содержащий выполненные команды, размеры и контрольные суммы файлов, результаты восстановления и сравнения и выводы.

Пример решения:
Задание: Создать резервные копии базы данных интернет-магазина в MySQL и базы данных бухгалтерии в PostgreSQL.
Дано: В MySQL база shop содержит таблицы products, orders, customers и хранимые процедуры. В PostgreSQL кластер содержит базы billing и hr и роли пользователей.

Решение по этапам:
1. Создаём пользователя MySQL: CREATE USER 'backup'@'localhost' IDENTIFIED BY '...'; GRANT SELECT, SHOW VIEW, TRIGGER, EVENT, LOCK TABLES, PROCESS ON *.* TO

'backup'@'localhost'; Сохраняем параметры в ~/.my.cnf в разделе [mysqldump] и задаём права chmod 600.
2. Выполняем полную копию: mysqldump --single-transaction --routines --triggers --events shop | gzip > shop_2026-10-08.sql.gz
3. Копируем только таблицу заказов: mysqldump --single-transaction shop orders > shop_orders_2026-10-08.sql и только структуру: mysqldump --no-data --routines shop > shop_schema.sql
4. В PostgreSQL выполняем: pg_dump -U postgres -Fc --verbose -f billing_2026-10-08.dump billing и текстовую копию: pg_dump -U postgres -Fp -f billing_2026-10-08.sql billing
5. Выполняем копию кластера: pg_dumpall -U postgres -f cluster_2026-10-08.sql и копию ролей: pg_dumpall -U postgres --globals-only -f globals_2026-10-08.sql
6. Проверяем коды завершения — все команды вернули 0. Размер копии shop — 48 МБ, billing.dump — 120 МБ. Рассчитываем контрольные суммы sha256sum и сохраняем их в файл checksums.txt.
7. Проверяем содержимое копии формата custom командой pg_restore --list billing_2026-10-08.dump — отображается список таблиц, индексов и функций.
8. Восстанавливаем MySQL: создаём базу shop_test и выполняем gunzip < shop_2026-10-08.sql.gz | mysql shop_test. Восстанавливаем PostgreSQL: createdb billing_test и pg_restore -U postgres -d billing_test billing_2026-10-08.dump
9. Сравниваем количество записей: в таблице orders исходной и восстановленной баз по 15 420 записей, в таблице payments — по 87 305. Хранимые процедуры в shop_test присутствуют.
10. Оформляем отчёт.

Ответ: Созданы полная, выборочная и структурная резервные копии базы MySQL shop с помощью mysqldump без блокировки таблиц, копии базы PostgreSQL billing в текстовом формате и формате custom с помощью pg_dump, а также копия всего кластера и ролей с помощью pg_dumpall. Успешное завершение подтверждено кодами возврата и контрольными суммами, а пробное восстановление показало полное совпадение данных.

Критерии оценки практических заданий:
1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Практическое занятие №17. Настройка и выполнение резервного копирования в Microsoft SQL Server с использованием SSMS. Автоматизация резервного копирования базы данных MongoDB с использованием скриптов.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных
ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации
ПК 2.5. Подготавливать данные для базы знаний
ПК 2.6. Осуществлять управление базами данных

Цель: Научиться выполнять и настраивать резервное копирование баз данных SQL Server средствами SSMS и T-SQL, контролировать его выполнение и результаты, а также разрабатывать скрипт автоматического резервного копирования MongoDB с ротацией, журналированием и проверкой результата.

Задание:
1. Проверить и при необходимости изменить модель восстановления базы данных SQL Server.
2. Выполнить полное резервное копирование базы через мастер SSMS.
3. Выполнить резервное копирование журнала транзакций и копию средствами T-SQL.
4. Проконтролировать выполнение и проверить резервную копию.
5. Настроить регулярное резервное копирование с помощью плана обслуживания или задания агента SQL Server.
6. Разработать скрипт резервного копирования MongoDB с помощью mongodump.
7. Добавить в скрипт ротацию копий, журналирование и проверку результата, настроить запуск по расписанию.
8. Проверить восстановление из копий и сделать вывод.

Этапы выполнения:
Этап 1. Модель восстановления.
В свойствах базы данных в SSMS на странице «Параметры» проверяется модель восстановления. Для возможности копирования журнала транзакций устанавливается полная модель (FULL).

Этап 2. Полное копирование через SSMS.
В контекстном меню базы выбирается «Задачи → Создать резервную копию». Указываются тип копии «Полная», путь к файлу, на странице параметров носителя включаются проверка резервной копии и контрольная сумма, на странице параметров — сжатие.

Этап 3. Копирование средствами T-SQL.
Выполняются команды BACKUP DATABASE и BACKUP LOG с параметрами CHECKSUM, COMPRESSION и STATS, позволяющим отслеживать процент выполнения.

Этап 4. Контроль и проверка.
Ход выполнения отслеживается по сообщениям STATS и в представлении sys.dm_exec_requests. История копий просматривается запросом к msdb.dbo.backupset. Копия проверяется командой RESTORE VERIFYONLY, содержимое — командой RESTORE HEADERONLY.

Этап 5. Регулярное копирование.
С помощью мастера планов обслуживания или задания агента SQL Server настраиваются ежедневная полная копия, частые копии журнала и удаление старых файлов копий. В

редакции Express, где нет агента, используется планировщик заданий Windows и утилита sqlcmd.

Этап 6. Скрипт MongoDB.
Разрабатывается скрипт bash, выполняющий mongodump с параметрами --uri, --gzip, --archive и --oplog для набора реплик. Имя архива включает дату и время. Параметры подключения читаются из защищённого файла.

Этап 7. Ротация, журналирование и расписание.
В скрипт добавляются проверка кода завершения mongodump, проверка размера архива, удаление копий старше заданного срока командой find с параметром -mtime, запись событий в журнал и отправка уведомления при ошибке. Запуск настраивается в cron.

Этап 8. Проверка восстановления и оформление.
База SQL Server восстанавливается под другим именем, архив MongoDB восстанавливается командой mongorestore в тестовую базу. Проверяются данные. Оформляется отчёт.

Пример решения:
Задание: Настроить резервное копирование базы заявок в SQL Server и базы каталога товаров в MongoDB.
Дано: База SQL Server Requests работает в простой модели восстановления. Требуется ежедневная полная копия и копии журнала каждые 30 минут. База MongoDB catalog работает в наборе реплик rs0, требуется ежедневная копия с хранением 7 дней.

Решение по этапам:
1. Переводим базу в полную модель: ALTER DATABASE Requests SET RECOVERY FULL;
2. Через мастер SSMS выполняем полную копию в файл D:\Backup\Requests_full.bak с проверкой и контрольной суммой. Мастер сообщает об успешном завершении.
3. Выполняем копию журнала: BACKUP LOG Requests TO DISK = 'D:\Backup\Requests_log_1030.trn' WITH CHECKSUM, COMPRESSION, STATS = 10; — в окне сообщений отображается ход выполнения 10%, 20% … 100%.
4. Проверяем историю: SELECT database_name, type, backup_start_date, backup_finish_date, backup_size FROM msdb.dbo.backupset WHERE database_name = 'Requests' ORDER BY backup_finish_date DESC; — отображаются полная копия (тип D) и копия журнала (тип L). Выполняем RESTORE VERIFYONLY FROM DISK = 'D:\Backup\Requests_full.bak' WITH CHECKSUM; — копия корректна.
5. Создаём план обслуживания: полная копия ежедневно в 01:00, копия журнала каждые 30 минут, удаление файлов копий старше 14 дней.
6. Разрабатываем скрипт /opt/scripts/mongo_backup.sh: формирует имя catalog_ДАТА_ВРЕМЯ.gz, выполняет mongodump --uri="$MONGO_URI" --gzip --archive="$FILE" --oplog, проверяет код завершения и что размер файла больше нуля, записывает результат в /var/log/mongo_backup.log.
7. Добавляем ротацию: find /backup/mongo -name "catalog_*.gz" -mtime +7 -delete, а при ошибке скрипт завершает работу с кодом 1 и отправляет письмо администратору.
8. Настраиваем cron: 0 2 * * * /opt/scripts/mongo_backup.sh — ежедневный запуск в 02:00. Через неделю проверяем, что в каталоге хранится 7 архивов.
9. Восстанавливаем копию SQL Server под именем Requests_test и архив MongoDB командой mongorestore --gzip --archive=catalog_2026-10-08_0200.gz --nsFrom="catalog.*" --nsTo="catalog_test.*". Количество документов в коллекции products совпадает с исходной — 25 400.
10. Оформляем отчёт.

Ответ: Для базы SQL Server Requests установлена полная модель восстановления, выполнены полная копия через SSMS и копия журнала средствами T-SQL с контрольными суммами, контроль выполнен по истории msdb и командой RESTORE VERIFYONLY, настроен план

обслуживания. Для MongoDB разработан скрипт автоматического резервного копирования с проверкой результата, ротацией, журналированием и запуском по расписанию cron. Пробное восстановление подтвердило пригодность копий.

Критерии оценки практических заданий:
1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Практическое занятие №18. Создание и управление резервными копиями Oracle Database с помощью RMAN (Recovery Manager). Настройка расписания автоматического резервного копирования в MySQL с использованием CRON.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных
ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации
ПК 2.5. Подготавливать данные для базы знаний
ПК 2.6. Осуществлять управление базами данных

Цель: Научиться выполнять резервное копирование Oracle Database с помощью RMAN, настраивать политику хранения, проверять и удалять устаревшие копии, а также настраивать автоматическое резервное копирование MySQL по расписанию с помощью cron.

Задание:
1. Перевести базу данных Oracle в режим ARCHIVELOG.
2. Настроить параметры RMAN: политику хранения, автоматическое копирование управляющего файла, сжатие.
3. Выполнить полное резервное копирование базы данных с архивными журналами.
4. Просмотреть, проверить и удалить устаревшие копии средствами RMAN.
5. Разработать скрипт резервного копирования MySQL.
6. Настроить запуск скрипта по расписанию с помощью cron.
7. Проверить выполнение заданий по расписанию и журналы.
8. Сделать вывод о результатах.

Этапы выполнения:
Этап 1. Режим ARCHIVELOG.
Режим базы проверяется запросом SELECT log_mode FROM v$database;. Для перевода в режим ARCHIVELOG база останавливается (SHUTDOWN IMMEDIATE), запускается в

режиме MOUNT, выполняется ALTER DATABASE ARCHIVELOG и база открывается. Проверяется настройка области быстрого восстановления (db_recovery_file_dest).

Этап 2. Настройка RMAN.
Выполняется подключение командой rman target /. Текущие настройки просматриваются командой SHOW ALL. Задаются политика хранения CONFIGURE RETENTION POLICY, автоматическое копирование управляющего файла CONFIGURE CONTROLFILE AUTOBACKUP ON и сжатие копий.

Этап 3. Резервное копирование.
Выполняется команда BACKUP DATABASE PLUS ARCHIVELOG. Ход выполнения отслеживается в выводе RMAN и в представлении V$SESSION_LONGOPS.

Этап 4. Управление копиями.
Копии просматриваются командами LIST BACKUP SUMMARY и LIST BACKUP OF DATABASE. Проверка выполняется командами CROSSCHECK BACKUP и RESTORE DATABASE VALIDATE. Устаревшие копии определяются командой REPORT OBSOLETE и удаляются командой DELETE OBSOLETE. Результаты заданий просматриваются в представлении V$RMAN_BACKUP_JOB_DETAILS.

Этап 5. Скрипт для MySQL.
Разрабатывается скрипт bash, выполняющий mysqldump всех нужных баз с параметром --single-transaction, сжатие, проверку кода завершения, расчёт контрольной суммы, ротацию

старых копий и запись журнала. Параметры подключения хранятся в файле ~/.my.cnf с правами 600.

Этап 6. Настройка cron.
Командой crontab -e добавляется строка расписания из пяти полей (минута, час, день месяца, месяц, день недели) и пути к скрипту с перенаправлением вывода в журнал. Проверяется, что скрипт имеет право на выполнение.

Этап 7. Проверка расписания.
Командой crontab -l проверяется расписание, в системном журнале проверяется запуск задания, в каталоге копий — появление новых файлов, в журнале скрипта — сообщения о результате.

Этап 8. Оформление результата.
Оформляется отчёт, содержащий настройки RMAN, выводы команд LIST и REPORT, текст скрипта MySQL, строку crontab, результаты проверки и выводы.

Пример решения:
Задание: Настроить резервное копирование базы данных Oracle отдела кадров и ежедневное автоматическое копирование базы MySQL сайта.
Дано: База Oracle работает в режиме NOARCHIVELOG, требуется хранить копии, позволяющие восстановиться за последние 7 дней. База MySQL site должна копироваться ежедневно в 02:30 с хранением копий 14 дней.

Решение по этапам:
1. Проверяем режим: SELECT log_mode FROM v$database; — NOARCHIVELOG. Выполняем SHUTDOWN IMMEDIATE; STARTUP MOUNT; ALTER DATABASE ARCHIVELOG; ALTER DATABASE OPEN; Повторная проверка показывает ARCHIVELOG.
2. Подключаемся rman target / и настраиваем: CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 7 DAYS; CONFIGURE CONTROLFILE AUTOBACKUP ON; CONFIGURE DEVICE TYPE DISK BACKUP TYPE TO COMPRESSED BACKUPSET;
3. Выполняем BACKUP DATABASE PLUS ARCHIVELOG; — RMAN создаёт наборы копий файлов данных, архивных журналов и автоматическую копию управляющего файла.
4. Проверяем: LIST BACKUP SUMMARY; показывает созданные наборы, RESTORE DATABASE VALIDATE; завершается без ошибок, CROSSCHECK BACKUP; подтверждает наличие файлов. Запрос к V$RMAN_BACKUP_JOB_DETAILS показывает статус COMPLETED.
5. Через неделю выполняем REPORT OBSOLETE; — отображаются копии старше окна восстановления, удаляем их командой DELETE OBSOLETE;
6. Создаём скрипт /opt/scripts/mysql_backup.sh: mysqldump --single-transaction --routines --triggers site | gzip > /backup/mysql/site_$(date +%F).sql.gz; при ненулевом коде завершения запись об ошибке в журнал и выход с кодом 1; расчёт sha256sum; удаление копий старше 14 дней командой find /backup/mysql -name "site_*.sql.gz" -mtime +14 -delete.
7. Делаем скрипт исполняемым командой chmod +x и проверяем ручным запуском — создан файл копии.
8. Добавляем в crontab строку: 30 2 * * * /opt/scripts/mysql_backup.sh >> /var/log/mysql_backup.log 2>&1
9. На следующий день проверяем: в системном журнале есть запись о запуске задания в 02:30, в каталоге /backup/mysql появился новый файл, в /var/log/mysql_backup.log — сообщение об успешном завершении.
10. Оформляем отчёт.

Ответ: База данных Oracle переведена в режим ARCHIVELOG, в RMAN настроены окно восстановления 7 дней, автоматическое копирование управляющего файла и сжатие, выполнено полное копирование с архивными журналами, проверка копий и удаление

устаревших. Для MySQL разработан скрипт резервного копирования с проверкой результата, контрольными суммами и ротацией, который ежедневно запускается cron в 02:30. Проверка подтвердила выполнение копирования по расписанию.

Критерии оценки практических заданий:
1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Практическое занятие №19. Проверка целостности и восстановления данных из резервной копии в PostgreSQL. Выполнение дифференциального резервного копирования в Microsoft SQL Server.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных
ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации
ПК 2.5. Подготавливать данные для базы знаний
ПК 2.6. Осуществлять управление базами данных

Цель: Научиться проверять целостность резервных копий PostgreSQL и выполнять восстановление данных с последующей проверкой, а также выполнять дифференциальное

резервное копирование в SQL Server и восстанавливать базу из цепочки полной и дифференциальной копий.

Задание:

1. Проверить целостность файла резервной копии PostgreSQL по контрольной сумме и средствами pg_restore.
2. Выполнить полное восстановление базы PostgreSQL в тестовую базу.
3. Выполнить выборочное восстановление отдельной таблицы.
4. Проверить целостность и полноту восстановленных данных.
5. Выполнить полную копию базы SQL Server и внести изменения в данные.
6. Выполнить дифференциальные резервные копии SQL Server.
7. Восстановить базу SQL Server из полной и дифференциальной копий.
8. Сравнить размеры копий и время восстановления, сделать вывод.

Этапы выполнения:

Этап 1. Проверка целостности копии PostgreSQL.
Рассчитывается контрольная сумма файла копии и сравнивается с сохранённой при создании. Командой pg_restore --list проверяется, что файл читается и содержит ожидаемые объекты.

Этап 2. Полное восстановление.
Создаётся тестовая база данных, при необходимости восстанавливаются роли из копии pg_dumpall --globals-only. Выполняется pg_restore с параметрами --verbose и --exit-on-error, при необходимости с параллельной загрузкой -j.

Этап 3. Выборочное восстановление.
С помощью параметра -t в pg_restore восстанавливается отдельная таблица, например ошибочно удалённая. Альтернативно с помощью списка объектов (-l и -L) восстанавливаются выбранные объекты.

Этап 4. Проверка восстановленных данных.
Сравнивается количество записей в таблицах исходной и восстановленной баз, проверяются ограничения целостности и внешние ключи, наличие индексов, функций и прав. Для выявления повреждений индексов может использоваться расширение amcheck.

Этап 5. Полная копия SQL Server.
Выполняется полная резервная копия базы командой BACKUP DATABASE с параметром CHECKSUM. Затем в базу вносятся изменения.

Этап 6. Дифференциальные копии.
Выполняется команда BACKUP DATABASE ... WITH DIFFERENTIAL. После новых изменений создаётся ещё одна дифференциальная копия, и сравниваются размеры копий.

Этап 7. Восстановление SQL Server.
Полная копия восстанавливается с параметром NORECOVERY, затем последняя дифференциальная копия — с параметром RECOVERY. Восстановление выполняется в базу

с другим именем с параметром MOVE для файлов данных и журнала. Проверяется наличие всех изменений.

Этап 8. Оформление результата.
Оформляется отчёт, содержащий команды проверки и восстановления, результаты сравнения данных, размеры копий, время восстановления и выводы.

Пример решения:
Задание: Проверить пригодность резервной копии базы PostgreSQL billing и организовать дифференциальное копирование базы SQL Server Requests.
Дано: Имеется копия billing_2026-10-08.dump в формате custom и файл контрольных сумм. Сотрудник ошибочно удалил таблицу tariffs в рабочей базе billing. База SQL Server Requests имеет размер 2 ГБ, ежедневно изменяется около 3% данных.

Решение по этапам:
1. Выполняем sha256sum -c checksums.txt — для файла billing_2026-10-08.dump результат OK. Команда pg_restore --list billing_2026-10-08.dump выводит список объектов, таблица tariffs присутствует.
2. Создаём тестовую базу createdb billing_check и выполняем pg_restore -U postgres -d billing_check --verbose --exit-on-error -j 4 billing_2026-10-08.dump — восстановление завершено без ошибок за 3 минуты.
3. Сравниваем количество записей в основных таблицах billing_check с данными отчёта на момент копирования — расхождений нет, внешние ключи и индексы присутствуют.
4. Восстанавливаем удалённую таблицу в рабочую базу: pg_restore -U postgres -d billing -t tariffs billing_2026-10-08.dump — таблица восстановлена, 214 записей. Восстанавливаем права на таблицу.
5. В SQL Server выполняем полную копию: BACKUP DATABASE Requests TO DISK = 'D:\Backup\Requests_full.bak' WITH CHECKSUM, COMPRESSION; — размер 410 МБ.
6. Добавляем 500 заявок и выполняем BACKUP DATABASE Requests TO DISK = 'D:\Backup\Requests_diff1.bak' WITH DIFFERENTIAL, CHECKSUM, COMPRESSION; — размер 18 МБ.
7. Добавляем ещё 700 заявок и выполняем вторую дифференциальную копию Requests_diff2.bak — размер 39 МБ, так как она содержит все изменения с момента полной копии.
8. Восстанавливаем: RESTORE DATABASE Requests_test FROM DISK = 'D:\Backup\Requests_full.bak' WITH MOVE 'Requests' TO 'D:\Data\Requests_test.mdf', MOVE 'Requests_log' TO 'D:\Data\Requests_test_log.ldf', NORECOVERY; затем RESTORE DATABASE Requests_test FROM DISK = 'D:\Backup\Requests_diff2.bak' WITH RECOVERY;
9. Проверяем: в Requests_test присутствуют все 1200 добавленных заявок, первая дифференциальная копия для восстановления не потребовалась. Общее время восстановления — 2 минуты.
10. Оформляем отчёт.

Ответ: Целостность резервной копии PostgreSQL подтверждена контрольной суммой, списком объектов и пробным восстановлением в тестовую базу, ошибочно удалённая таблица tariffs восстановлена выборочно. В SQL Server выполнены полная и две дифференциальные копии, база успешно восстановлена из полной и последней дифференциальной копии.

Дифференциальные копии значительно меньше полной, но их размер растёт со временем, что необходимо учитывать при планировании.

Критерии оценки практических заданий:
1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Практическое занятие №20. Создание инкрементального резервного копирования в Oracle Database. Разработка стратегии резервного копирования и восстановления для базы данных предприятия.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных
ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации
ПК 2.5. Подготавливать данные для базы знаний
ПК 2.6. Осуществлять управление базами данных

Цель: Научиться выполнять инкрементальное резервное копирование Oracle Database с помощью RMAN, использовать отслеживание изменённых блоков, а также разрабатывать стратегию резервного копирования и восстановления базы данных предприятия на основе требований RPO и RTO.

Задание:
1. Включить отслеживание изменённых блоков в Oracle.
2. Выполнить инкрементальную копию уровня 0.
3. Выполнить дифференциальные и кумулятивные инкрементальные копии уровня 1.
4. Сравнить размер и время создания копий разных уровней и проверить их.
5. Проанализировать требования предприятия к сохранности и доступности данных.
6. Определить виды, частоту, места и сроки хранения резервных копий.
7. Разработать порядок проверки копий и план восстановления для основных сценариев аварий.
8. Оформить стратегию резервного копирования и восстановления.

Этапы выполнения:
Этап 1. Отслеживание изменённых блоков.
Выполняется команда ALTER DATABASE ENABLE BLOCK CHANGE TRACKING с указанием файла. Это позволяет RMAN при инкрементальном копировании читать только

изменённые блоки, а не все файлы данных. Состояние проверяется в представлении V$BLOCK_CHANGE_TRACKING.

Этап 2. Копия уровня 0.
Выполняется команда BACKUP INCREMENTAL LEVEL 0 DATABASE PLUS ARCHIVELOG. Копия уровня 0 содержит все блоки и является основой для последующих инкрементальных копий.

Этап 3. Копии уровня 1.
После изменения данных выполняется дифференциальная инкрементальная копия BACKUP INCREMENTAL LEVEL 1 DATABASE, содержащая изменения с момента последней копии уровня 0 или 1. Затем выполняется кумулятивная копия BACKUP INCREMENTAL LEVEL 1 CUMULATIVE DATABASE, содержащая все изменения с момента последней копии уровня 0.

Этап 4. Сравнение и проверка.
Командой LIST BACKUP и запросом к V$RMAN_BACKUP_JOB_DETAILS сравниваются размер и время создания копий. Выполняется проверка RESTORE DATABASE VALIDATE.

Этап 5. Анализ требований.
Определяются базы данных предприятия, их объём, скорость изменения и критичность, допустимая потеря данных (RPO) и допустимое время восстановления (RTO), требования законодательства к сроку хранения, доступные ресурсы хранения.

Этап 6. Определение параметров копирования.
Для каждой базы определяются виды копий (полные, дифференциальные или инкрементальные, копии журналов), их расписание, места хранения по правилу 3-2-1, схема ротации, сроки хранения, шифрование копий и ответственные лица.

Этап 7. Проверка и план восстановления.
Определяется порядок ежедневного контроля выполнения копий, автоматической проверки и регулярного пробного восстановления. Разрабатываются пошаговые инструкции восстановления для сценариев: ошибочное удаление данных, повреждение файла данных, полная потеря сервера, атака шифровальщика.

Этап 8. Оформление стратегии.
Стратегия оформляется в виде документа с разделами: цели и область действия, требования RPO и RTO, расписание копирования, хранение и ротация, контроль и проверка, план восстановления, ответственность, порядок пересмотра.

Пример решения:
Задание: Организовать инкрементальное резервное копирование базы Oracle и разработать стратегию резервного копирования для торгового предприятия.
Дано: База Oracle учётной системы объёмом 200 ГБ, ежедневно изменяется около 2% данных. Требования: RPO — не более 15 минут, RTO — не более 4 часов. На предприятии также работают база MySQL сайта (20 ГБ, RPO — 24 часа) и база PostgreSQL отдела кадров (5 ГБ,

содержит персональные данные). Имеется дисковое хранилище в серверной и арендованное облачное хранилище в России.

Решение по этапам:
1. Включаем отслеживание изменений: ALTER DATABASE ENABLE BLOCK CHANGE TRACKING USING FILE '/u01/oradata/bct.chg'; — в V$BLOCK_CHANGE_TRACKING статус ENABLED.
2. Выполняем BACKUP INCREMENTAL LEVEL 0 DATABASE PLUS ARCHIVELOG; — размер копии 92 ГБ со сжатием, время 1 час 50 минут.
3. После рабочего дня выполняем BACKUP INCREMENTAL LEVEL 1 DATABASE; — размер 3,1 ГБ, время 9 минут благодаря отслеживанию изменённых блоков. Через день выполняем BACKUP INCREMENTAL LEVEL 1 CUMULATIVE DATABASE; — размер 6,4 ГБ, так как копия содержит изменения за два дня.
4. Выполняем RESTORE DATABASE VALIDATE; — ошибок нет. Сравниваем: копии уровня 1 создаются в 10–12 раз быстрее копии уровня 0.
5. Для базы Oracle определяем: копия уровня 0 — еженедельно в воскресенье в 01:00, уровня 1 — ежедневно в 01:00, архивные журналы — каждые 15 минут, что обеспечивает RPO 15 минут. Окно восстановления RMAN — 14 дней.
6. Для базы MySQL: ежедневная полная копия mysqldump в 02:30, хранение по схеме GFS — 7 ежедневных, 4 еженедельных, 12 ежемесячных копий. Для базы PostgreSQL: ежедневная копия pg_dump и копия ролей, копии шифруются, так как содержат персональные данные.
7. Хранение по правилу 3-2-1: рабочие базы, копии на дисковом хранилище в серверной и зашифрованные копии в облачном хранилище на территории России с защитой от удаления. Доступ к хранилищу копий — только у администраторов баз данных.
8. Контроль: ежедневная автоматическая проверка выполнения заданий с уведомлением об ошибках, еженедельная проверка RESTORE VALIDATE и pg_restore --list, ежемесячное пробное восстановление каждой базы на тестовый сервер с замером времени для подтверждения RTO.
9. Разрабатываем план восстановления для четырёх сценариев. Например, при полной потере сервера Oracle: подготовка нового сервера, восстановление управляющего файла из автокопии, RESTORE DATABASE и RECOVER DATABASE с применением инкрементальных копий и архивных журналов — расчётное время 3 часа, что укладывается в RTO.
10. Оформляем документ «Стратегия резервного копирования и восстановления баз данных» с указанием ответственных и пересмотром не реже одного раза в год.

Ответ: В Oracle включено отслеживание изменённых блоков и выполнены инкрементальные копии уровня 0 и уровня 1 (дифференциальная и кумулятивная), подтверждено значительное сокращение размера и времени создания копий уровня 1. Разработана стратегия резервного копирования и восстановления для трёх баз данных предприятия, обеспечивающая требования RPO и RTO, хранение по правилу 3-2-1 с шифрованием, ротацию, регулярную проверку копий и план восстановления для основных сценариев аварий.

Критерии оценки практических заданий:
1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Тема 1.5. Восстановление баз данных.
Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам

ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных
ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации
ПК 2.5. Подготавливать данные для базы знаний
ПК 2.6. Осуществлять управление базами данных

Вопросы к устному опросу по теме:
1. Что такое восстановление базы данных?
Ответ: Восстановление базы данных — это процесс возврата базы данных в работоспособное и согласованное состояние с использованием резервных копий и журналов транзакций после сбоя, повреждения или потери данных. Восстановление включает восстановление файлов или данных из копии и при необходимости применение изменений из журналов до нужного момента времени.

2. В каких ситуациях выполняется восстановление базы данных?
Ответ: Восстановление выполняется при отказе дисков и серверов, повреждении файлов данных, ошибочном удалении или изменении данных пользователем или приложением, атаке вируса-шифровальщика, неудачном обновлении программного обеспечения, а также при переносе базы на новый сервер и создании тестовой копии.

3. Какие виды восстановления существуют?
Ответ: Выделяют полное восстановление базы данных, частичное восстановление отдельных таблиц, коллекций, файлов или табличных пространств, восстановление на определённый момент времени, восстановление на другой сервер или под другим именем, а также аварийное восстановление всей системы на резервной площадке.

4. Какие действия выполняются перед запуском процедуры восстановления?
Ответ: Перед восстановлением определяются причина и масштаб сбоя, выбирается момент, на который нужно восстановить данные, находится нужная резервная копия и проверяется её целостность, оценивается время восстановления, уведомляются пользователи, прекращается доступ приложений к базе, при возможности сохраняется текущее состояние базы и журналов, проверяется наличие свободного места на диске.

5. Почему перед восстановлением важно сохранить текущее состояние базы?
Ответ: Восстановление перезаписывает текущие данные. Если ошибочно выбрана копия или процедура завершится неудачно, без сохранённого состояния будут потеряны изменения, сделанные после резервного копирования. В SQL Server для этого выполняется резервная копия заключительного фрагмента журнала (tail-log backup), в других СУБД — копия текущих файлов или журналов.

6. Почему восстановление часто выполняют сначала в отдельную базу?
Ответ: Восстановление в отдельную базу позволяет проверить содержимое копии, найти нужные данные и убедиться в корректности результата, не затрагивая рабочую базу. После

проверки нужные данные переносятся в рабочую базу или восстановленная база заменяет рабочую. Это снижает риск ошибочной перезаписи данных.

7. Как выполняется восстановление базы MySQL из копии mysqldump?
Ответ: Создаётся целевая база данных, если копия сделана без параметра --databases, затем файл копии выполняется клиентом mysql: mysql -u root -p shop < shop.sql. Для сжатой копии используется gunzip < shop.sql.gz | mysql shop. Копия mysqldump содержит команды создания таблиц и вставки данных, а также отключение проверки внешних ключей на время загрузки.

8. Чем восстановление через pg_restore отличается от восстановления через psql в PostgreSQL?
Ответ: Копии в текстовом формате SQL восстанавливаются выполнением файла через psql. Копии в форматах custom, directory и tar восстанавливаются утилитой pg_restore, которая позволяет выбирать отдельные объекты, изменять порядок восстановления, удалять объекты перед созданием (--clean) и выполнять восстановление параллельно (-j), что значительно ускоряет работу.

9. Что означают параметры NORECOVERY, RECOVERY и STANDBY при восстановлении в SQL Server?
Ответ: Параметр NORECOVERY оставляет базу в состоянии восстановления, чтобы можно было применить следующие копии: дифференциальную и журналы. Параметр RECOVERY завершает восстановление, откатывает незавершённые транзакции и делает базу доступной. Параметр STANDBY делает базу доступной только для чтения между применением журналов.

10. Как выполняется восстановление MongoDB?
Ответ: Восстановление из копии mongodump выполняется утилитой mongorestore. Параметр --drop удаляет существующие коллекции перед восстановлением, --gzip и --archive используются для сжатых архивов, --nsInclude позволяет восстановить отдельные базы и коллекции, --nsFrom и --nsTo — восстановить под другим именем, --oplogReplay — применить записи журнала операций для согласованности.

11. Чем команда RESTORE отличается от RECOVER в RMAN?
Ответ: Команда RESTORE извлекает файлы данных, управляющие файлы или файлы параметров из резервных копий и записывает их на диск. Команда RECOVER применяет к восстановленным файлам инкрементальные копии, архивные и оперативные журналы повторного выполнения, приводя базу в согласованное состояние на текущий или заданный момент времени.

12. Что означает открытие базы Oracle с параметром RESETLOGS?
Ответ: Параметр RESETLOGS используется после неполного восстановления, восстановления на момент времени или восстановления управляющего файла из копии. Он создаёт новую инкарнацию базы данных и сбрасывает последовательность журналов, так как изменения после точки восстановления отбрасываются. После открытия с RESETLOGS рекомендуется сразу выполнить новую резервную копию.

13. Как выполняется восстановление на определённый момент времени?
Ответ: Восстанавливается полная резервная копия, сделанная до нужного момента, затем последовательно применяются журналы транзакций с остановкой на заданном времени или позиции: в MySQL — двоичные журналы утилитой mysqlbinlog с параметром --stop-datetime, в PostgreSQL — архив WAL с параметром recovery_target_time, в SQL Server — копии журнала с параметром STOPAT, в Oracle — команда RECOVER DATABASE UNTIL TIME.

14. Что отслеживается при мониторинге процедуры восстановления?
Ответ: Отслеживаются ход выполнения и процент завершения, скорость восстановления, объём обработанных данных, текущий этап (загрузка данных, создание индексов, применение

журналов), ошибки и предупреждения, использование дискового пространства, процессора и памяти, а также прогнозируемое время окончания.

15. Какие средства мониторинга восстановления есть в разных СУБД?
Ответ: В SQL Server используется параметр STATS и представление sys.dm_exec_requests со столбцами percent_complete и estimated_completion_time. В Oracle — представления V$SESSION_LONGOPS и V$RMAN_STATUS. В PostgreSQL — подробный вывод pg_restore --verbose и представления pg_stat_activity и pg_stat_progress_copy. В MySQL — команда SHOW PROCESSLIST и утилита pv для отображения хода чтения файла. В MongoDB — вывод mongorestore с количеством восстановленных документов.

16. Как оценить время восстановления базы данных?
Ответ: Время оценивается по результатам предыдущих пробных восстановлений с учётом объёма данных, формата копии, скорости дисков и сети, количества индексов и объёма журналов для применения. Фактическое время сравнивается с требуемым RTO. Если оно превышает RTO, меняется стратегия: используются физические копии, параллельное восстановление или резервные серверы.

17. По каким признакам контролируется успешное завершение восстановления?
Ответ: Проверяются код завершения утилиты, отсутствие ошибок в выводе и журналах СУБД, состояние базы данных (доступна для работы), наличие всех объектов, совпадение количества записей с ожидаемым, наличие данных на нужный момент времени, сведения о восстановлении в системных таблицах, например msdb.dbo.restorehistory в SQL Server.

18. Как проверить целостность данных после восстановления?
Ответ: Выполняются средства проверки целостности СУБД: DBCC CHECKDB в SQL Server, CHECK TABLE и утилита mysqlcheck в MySQL, VALIDATE в RMAN для Oracle, расширение amcheck в PostgreSQL, команда validate в MongoDB. Также проверяются ограничения и внешние ключи, сравниваются количество записей и контрольные суммы таблиц с эталонными значениями, выполняются контрольные запросы.

19. Какие действия выполняются после восстановления базы данных?
Ответ: После восстановления проверяются и восстанавливаются пользователи, роли и права, обновляется статистика оптимизатора, проверяются задания и связи с другими системами, подключаются и проверяются приложения, выполняется новая полная резервная копия, пользователи уведомляются о возобновлении работы, инцидент и выполненные действия документируются.

20. Какие типичные ошибки возникают при восстановлении и как их избежать?
Ответ: Типичные ошибки: восстановление не той копии или поверх рабочей базы, отсутствие части цепочки копий или журналов, несовместимость версий СУБД, нехватка места на диске, отсутствие ролей и владельцев объектов, неверные пути к файлам на новом сервере, повреждённая копия. Чтобы их избежать, нужны регулярные пробные восстановления, документированные инструкции и проверка копий.

21. Почему восстановление необходимо документировать?
Ответ: Документирование фиксирует причину инцидента, выбранную копию, выполненные команды, время начала и окончания, результаты проверки и потерянные данные. Эти сведения нужны для отчёта руководству, анализа причин сбоя, улучшения стратегии резервного копирования и инструкций, а также для подтверждения выполнения требований RPO и RTO.

22. Что такое Disaster Recovery?
Ответ: Disaster Recovery (аварийное восстановление) — это совокупность планов, технологий и процедур, обеспечивающих восстановление работы информационных систем и данных после крупных аварий: пожара, затопления, отключения электроэнергии, отказа всего центра

обработки данных, масштабной кибератаки. Аварийное восстановление является частью обеспечения непрерывности бизнеса.

23. Что включает план аварийного восстановления (DRP)?
Ответ: План аварийного восстановления включает перечень критичных систем и баз данных с приоритетами восстановления, значения RPO и RTO, описание резервной площадки и инфраструктуры, расположение резервных копий, пошаговые сценарии восстановления, критерии объявления аварии, роли и контакты ответственных, порядок оповещения, проверки после восстановления, возврата на основную площадку и график тестирования плана.

24. Какие виды резервных площадок существуют?
Ответ: Горячая площадка содержит постоянно работающее оборудование с актуальными копиями данных и позволяет переключиться за минуты. Тёплая площадка содержит оборудование и периодически обновляемые данные и требует нескольких часов на запуск. Холодная площадка — это помещение и базовая инфраструктура без готовых систем, восстановление на ней занимает дни. Также используется облачная резервная площадка.

25. Чем репликация отличается от резервного копирования с точки зрения аварийного восстановления?
Ответ: Репликация поддерживает актуальную копию базы на другом сервере и позволяет быстро переключиться при отказе оборудования, обеспечивая малые RPO и RTO. Однако ошибочное удаление данных или шифрование тоже реплицируется. Резервные копии позволяют вернуться к состоянию в прошлом. Поэтому репликация не заменяет резервное копирование, и для аварийного восстановления используются оба механизма.

26. Чем переключение при сбое (failover) отличается от планового переключения (switchover)?
Ответ: Failover — это аварийное переключение работы на резервный сервер при отказе основного, возможна потеря последних изменений. Switchover — это плановое переключение ролей основного и резервного серверов без потери данных, например для обслуживания основного сервера или тестирования плана аварийного восстановления.

27. Какие технологии аварийного восстановления предоставляют СУБД?
Ответ: В Oracle используется Data Guard с резервной базой данных, в SQL Server — группы доступности Always On и доставка журналов (Log Shipping), в PostgreSQL — потоковая репликация и архивирование WAL, в MySQL — репликация и групповая репликация (Group Replication), в MongoDB — наборы реплик с узлами в разных центрах обработки данных.

28. Какие виды тестирования плана аварийного восстановления существуют?
Ответ: Существуют проверка документов плана, штабные учения (обсуждение действий по сценарию аварии за столом), пошаговый разбор сценария с участниками, моделирование аварии на тестовом стенде, параллельное тестирование с запуском систем на резервной площадке без остановки основной и полное переключение на резервную площадку.

29. Почему план аварийного восстановления необходимо регулярно тестировать?
Ответ: Без проверки план может оказаться неработоспособным: копии повреждены, инструкции устарели, изменилась инфраструктура, сотрудники не знают своих действий, фактическое время восстановления превышает RTO. Тестирование выявляет проблемы заранее, подтверждает фактические RPO и RTO и обучает персонал. По результатам каждого теста план обновляется.

30. Что такое сценарий (runbook) восстановления и какие требования к нему предъявляются?
Ответ: Сценарий восстановления — это пошаговая инструкция действий для конкретного вида аварии с точными командами, ожидаемыми результатами, точками проверки и действиями при ошибках. Он должен быть понятен специалисту, который выполняет его

впервые, храниться в доступном при аварии месте, содержать контакты ответственных и актуализироваться после каждого изменения системы и каждого теста.

Критерии оценки устного опроса:

- оценка «отлично» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов и смог ответить на уточняющие вопросы;
- оценка «хорошо» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов, но не смог ответить на уточняющие вопросы;
- оценка «удовлетворительно» выставляется студенту, если он смог изложить суть основного вопроса, но не смог ответить по существу дополнительных вопросов;
- оценка «неудовлетворительно» выставляется студенту, если он не смог изложить суть основного вопроса.

Практическое занятие №21. Восстановление базы данных MySQL из резервной копии, созданной с помощью mysqldump.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных
ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации
ПК 2.5. Подготавливать данные для базы знаний
ПК 2.6. Осуществлять управление базами данных

Цель: Научиться выполнять восстановление базы данных MySQL из резервной копии mysqldump, отслеживать ход восстановления, контролировать его завершение и проверять восстановленные данные.

Задание:
1. Определить причину восстановления и выбрать нужную резервную копию.
2. Проверить целостность и содержимое файла резервной копии.
3. Подготовить сервер и целевую базу данных к восстановлению.
4. Выполнить восстановление базы данных из копии.
5. Отслеживать ход выполнения восстановления.
6. Проконтролировать завершение восстановления.
7. Проверить восстановленные объекты, данные и права пользователей.
8. Оформить отчёт о восстановлении.

Этапы выполнения:
Этап 1. Анализ ситуации и выбор копии.
Определяется, что именно утрачено и на какой момент нужно восстановить данные. Из каталога копий выбирается последняя пригодная копия.

Этап 2. Проверка копии.
Проверяется контрольная сумма файла, просматриваются начало и конец файла: в начале указаны версия сервера и база, в конце должна быть строка «Dump completed». Определяется, создана ли копия с параметром --databases (содержит команду CREATE DATABASE) или без него.

Этап 3. Подготовка.
Проверяется свободное место на диске, доступ приложений к базе временно прекращается. При восстановлении поверх существующей базы она переименовывается или предварительно

сохраняется копия текущего состояния. При необходимости создаётся пустая целевая база с нужной кодировкой.

Этап 4. Восстановление.
Файл копии загружается клиентом mysql: mysql -u root -p имя_базы < файл.sql или для сжатой копии через gunzip. Для больших копий используется утилита pv для отображения хода чтения файла.

Этап 5. Мониторинг.
Ход восстановления отслеживается по индикатору pv, командой SHOW PROCESSLIST в другом сеансе (видно, какая таблица загружается) и по журналу ошибок сервера.

Этап 6. Контроль завершения.
Проверяется код завершения команды, отсутствие сообщений об ошибках, наличие всех таблиц, представлений, процедур и триггеров.

Этап 7. Проверка данных и прав.
Сравнивается количество записей в таблицах с ожидаемым, выполняются контрольные запросы, проверяется работа приложения. Проверяются права пользователей приложения на восстановленную базу, так как mysqldump одной базы не содержит учётных записей.

Этап 8. Оформление.
Оформляется отчёт о восстановлении: причина, использованная копия, команды, время начала и окончания, результаты проверки, потерянные данные.

Пример решения:
Задание: Восстановить базу данных интернет-магазина после повреждения диска.
Дано: На сервере MySQL вышел из строя диск с базой shop, сервер переустановлен. Имеется ежедневная копия shop_2026-10-08.sql.gz, созданная mysqldump без параметра --databases, и файл контрольных сумм. Пользователь приложения shop_app.

Решение по этапам:
1. Определяем, что база shop утрачена полностью, выбираем последнюю копию shop_2026-10-08.sql.gz. Изменения после 02:30 будут восстановлены отдельно по журналам, если они сохранились.
2. Проверяем копию: sha256sum -c checksums.txt — OK, zcat shop_2026-10-08.sql.gz | tail -n 1 — «Dump completed on 2026-10-08 02:31:47».
3. Проверяем свободное место командой df -h — 120 ГБ свободно при размере базы около 6 ГБ.
4. Создаём базу: CREATE DATABASE shop CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci;
5. Запускаем восстановление с отображением хода: pv shop_2026-10-08.sql.gz | gunzip | mysql -u root -p shop
6. В другом сеансе выполняем SHOW PROCESSLIST; — видна загрузка данных в таблицу orders. Индикатор pv показывает процент обработки и оставшееся время, восстановление заняло 14 минут.
7. Проверяем код завершения — 0, ошибок нет. SHOW TABLES; показывает все 18 таблиц, SHOW PROCEDURE STATUS WHERE Db = 'shop'; — 4 процедуры.
8. Сравниваем количество записей с отчётом, сформированным при резервном копировании: orders — 15 420, customers — 3 812, совпадает.
9. Восстанавливаем права пользователя приложения: CREATE USER 'shop_app'@'10.0.0.15' IDENTIFIED BY '...'; GRANT SELECT, INSERT, UPDATE, DELETE ON shop.* TO 'shop_app'@'10.0.0.15'; Подключаем приложение и выполняем тестовый заказ.
10. Выполняем новую резервную копию и оформляем отчёт.

Ответ: База данных shop восстановлена из копии mysqldump за 14 минут. Перед восстановлением проверены контрольная сумма и полнота копии, ход восстановления отслеживался с помощью pv и SHOW PROCESSLIST. Проверка подтвердила наличие всех таблиц и процедур и совпадение количества записей, восстановлены права пользователя приложения, работа магазина возобновлена.

Критерии оценки практических заданий:

1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Практическое занятие №22. Восстановление PostgreSQL базы данных из дампа (pg_restore).

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных
ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации
ПК 2.5. Подготавливать данные для базы знаний
ПК 2.6. Осуществлять управление базами данных

Цель: Научиться восстанавливать базы данных PostgreSQL из дампов в форматах custom и directory с помощью pg_restore, использовать параллельное и выборочное восстановление, отслеживать ход и контролировать завершение восстановления.

Задание:
1. Просмотреть содержимое дампа и определить параметры восстановления.
2. Подготовить роли и целевую базу данных.
3. Выполнить полное восстановление базы с параллельной загрузкой.
4. Отслеживать ход восстановления.
5. Выполнить восстановление с заменой существующих объектов.
6. Выполнить выборочное восстановление с помощью списка объектов.
7. Проверить результат и обновить статистику.
8. Оформить отчёт.

Этапы выполнения:
Этап 1. Анализ дампа.
Командой pg_restore --list просматривается список объектов дампа, определяются формат, версия, исходная база и владельцы объектов.

Этап 2. Подготовка.
Если в целевом кластере нет нужных ролей, они создаются или восстанавливаются из копии pg_dumpall --globals-only. Создаётся пустая база данных командой createdb с нужным владельцем, кодировкой и локалью.

Этап 3. Полное восстановление.
Выполняется pg_restore -d база -j количество_потоков --verbose --exit-on-error файл. Параметр -j ускоряет восстановление данных и создание индексов на многоядерном сервере. При отсутствии ролей используется параметр --no-owner.

Этап 4. Мониторинг.
Ход отслеживается по выводу --verbose (какой объект обрабатывается), в представлении pg_stat_activity (выполняющиеся команды COPY и CREATE INDEX) и в представлении pg_stat_progress_copy (количество загруженных строк и байт).

Этап 5. Восстановление с заменой.
Для восстановления поверх существующей базы используются параметры --clean --if-exists, которые удаляют существующие объекты перед их созданием.

Этап 6. Выборочное восстановление.
Список объектов сохраняется в файл командой pg_restore -l, в файле ненужные строки комментируются знаком «;», затем выполняется восстановление по списку параметром -L. Отдельные таблицы восстанавливаются параметром -t.

Этап 7. Проверка результата.
Проверяются код завершения, наличие объектов (\dt, \df), количество записей, права. Выполняется обновление статистики командой ANALYZE или утилитой vacuumdb --analyze-in-stages, так как статистика оптимизатора после загрузки должна быть пересчитана.

Этап 8. Оформление.
Оформляется отчёт, содержащий команды, вывод pg_restore, время восстановления при разном количестве потоков, результаты проверки и выводы.

Пример решения:

Задание: Восстановить базу данных бухгалтерии на тестовом сервере и вернуть ошибочно изменённые справочники в рабочей базе.
Дано: Имеются дамп billing_2026-10-08.dump в формате custom (размер 1,2 ГБ) и копия ролей globals.sql. На тестовом сервере 8 ядер процессора. В рабочей базе ошибочно изменены справочники tariffs и services.

Решение по этапам:
1. Выполняем pg_restore --list billing_2026-10-08.dump | head — дамп создан из базы billing, содержит 42 таблицы, 65 индексов и 12 функций.
2. На тестовом сервере восстанавливаем роли: psql -U postgres -f globals.sql, создаём базу: createdb -U postgres -O billing_owner -E UTF8 billing.
3. Запускаем: pg_restore -U postgres -d billing -j 8 --verbose --exit-on-error billing_2026-10-08.dump
4. В другом сеансе выполняем SELECT relid::regclass, tuples_processed FROM pg_stat_progress_copy; — видно количество загруженных строк таблицы payments. В pg_stat_activity видны параллельные команды создания индексов.
5. Восстановление завершено за 6 минут (в однопоточном режиме ранее — 17 минут), код завершения 0.
6. Проверяем: \dt показывает 42 таблицы, количество записей в payments — 87 305, совпадает с отчётом. Выполняем vacuumdb -U postgres --analyze-in-stages billing.
7. Для рабочей базы формируем список: pg_restore -l billing_2026-10-08.dump > list.txt и оставляем в нём только строки TABLE DATA для tariffs и services.
8. Сохраняем текущие версии справочников в таблицы tariffs_bak и services_bak, очищаем справочники командой TRUNCATE в транзакции.
9. Выполняем pg_restore -U postgres -d billing -L list.txt billing_2026-10-08.dump — данные справочников восстановлены, сверяем их с тестовой базой — совпадают.
10. Оформляем отчёт.

Ответ: База данных billing восстановлена на тестовом сервере из дампа формата custom с помощью pg_restore в 8 потоков, что сократило время восстановления с 17 до 6 минут. Ход восстановления контролировался через вывод --verbose и pg_stat_progress_copy, после завершения проверены объекты и данные и обновлена статистика. В рабочей базе с помощью списка объектов выборочно восстановлены справочники без затрагивания остальных данных.

Критерии оценки практических заданий:
1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Практическое занятие №23. Восстановление базы данных Microsoft SQL Server из полной резервной копии с использованием SSMS.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных

ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации
ПК 2.5. Подготавливать данные для базы знаний
ПК 2.6. Осуществлять управление базами данных

Цель: Научиться восстанавливать базу данных SQL Server из полной резервной копии с помощью мастера SSMS, настраивать параметры восстановления, отслеживать ход выполнения и проверять результат.

Задание:
1. Просмотреть сведения о резервной копии.
2. Выполнить восстановление базы данных под новым именем с перемещением файлов.
3. Выполнить восстановление поверх существующей базы с созданием копии заключительного фрагмента журнала.
4. Отслеживать ход восстановления.
5. Проконтролировать завершение по истории восстановления.
6. Проверить целостность базы командой DBCC CHECKDB.
7. Проверить и восстановить сопоставление пользователей базы с логинами.
8. Оформить отчёт.

Этапы выполнения:
Этап 1. Сведения о копии.
В мастере восстановления на странице «Общие» выбирается источник «Устройство» и файл копии, просматриваются набор копий, дата и тип. Средствами T-SQL сведения получаются командами RESTORE HEADERONLY и RESTORE FILELISTONLY.

Этап 2. Восстановление под новым именем.
В поле «База данных» назначения указывается новое имя. На странице «Файлы» включается перемещение файлов в новый каталог с новыми именами, чтобы не конфликтовать с файлами существующей базы.

Этап 3. Восстановление поверх существующей базы.
На странице «Параметры» включаются перезапись существующей базы (WITH REPLACE), закрытие существующих подключений и создание резервной копии заключительного

фрагмента журнала перед восстановлением. Выбирается состояние восстановления RESTORE WITH RECOVERY.

Этап 4. Мониторинг.
Ход восстановления отображается в мастере в процентах. При восстановлении через скрипт используется параметр STATS, а в другом окне — запрос к sys.dm_exec_requests со столбцами percent_complete и estimated_completion_time.

Этап 5. Контроль завершения.
Проверяется сообщение об успешном завершении, состояние базы в обозревателе объектов, запись в таблице msdb.dbo.restorehistory.

Этап 6. Проверка целостности.
Выполняется команда DBCC CHECKDB, которая проверяет логическую и физическую целостность объектов базы. Результат не должен содержать ошибок согласованности.

Этап 7. Проверка пользователей.
Если база восстановлена на другом сервере, пользователи базы могут быть не сопоставлены с логинами («потерянные» пользователи). Они выявляются запросом к sys.database_principals и sys.server_principals и сопоставляются командой ALTER USER ... WITH LOGIN.

Этап 8. Оформление.
Оформляется отчёт, содержащий настройки мастера, сформированный скрипт, сведения о ходе и времени восстановления, результаты DBCC CHECKDB и проверки пользователей.

Пример решения:
Задание: Восстановить базу данных складского учёта после ошибочного обновления структуры.
Дано: Полная копия Warehouse_full.bak создана в 01:00. В 11:00 неудачный скрипт обновления повредил структуру таблиц рабочей базы Warehouse. Нужно сначала проверить копию, восстановив её под другим именем, затем восстановить рабочую базу.

Решение по этапам:
1. Выполняем RESTORE HEADERONLY FROM DISK = 'D:\Backup\Warehouse_full.bak'; — тип копии «полная», дата 2026-10-08 01:00. RESTORE FILELISTONLY показывает логические имена Warehouse и Warehouse_log.
2. В мастере SSMS указываем устройство Warehouse_full.bak, имя базы назначения Warehouse_check, на странице «Файлы» перемещаем файлы в D:\Data\Warehouse_check.mdf и D:\Data\Warehouse_check_log.ldf.
3. Восстановление выполнено, ход отображался в процентах. Проверяем в Warehouse_check структуру таблиц — она корректна, данные на 01:00 присутствуют.
4. Переводим приложение в режим обслуживания и уведомляем пользователей.
5. Запускаем мастер для базы Warehouse: на странице «Параметры» включаем перезапись существующей базы, закрытие подключений и копию заключительного фрагмента журнала (сохраняем на случай необходимости анализа изменений).
6. Сохраняем сформированный скрипт и выполняем его. В другом окне выполняем SELECT percent_complete, estimated_completion_time FROM sys.dm_exec_requests WHERE command = 'RESTORE DATABASE'; — отображается ход выполнения.
7. Проверяем историю: SELECT destination_database_name, restore_date, restore_type FROM msdb.dbo.restorehistory ORDER BY restore_date DESC; — записи о восстановлении Warehouse_check и Warehouse.
8. Выполняем DBCC CHECKDB ('Warehouse') WITH NO_INFOMSGS; — ошибок согласованности не обнаружено.

9. Проверяем пользователей — все сопоставлены с логинами, так как восстановление выполнялось на том же сервере. Подключаем приложение, проверяем основные операции. Данные, введённые с 01:00 до 11:00, вводятся повторно по первичным документам.
10. Удаляем базу Warehouse_check, выполняем новую полную копию и оформляем отчёт.
Ответ: С помощью мастера SSMS резервная копия сначала проверена восстановлением в базу Warehouse_check с перемещением файлов, затем рабочая база Warehouse восстановлена с перезаписью и созданием копии заключительного фрагмента журнала. Ход восстановления отслеживался через sys.dm_exec_requests, завершение подтверждено историей restorehistory, целостность — командой DBCC CHECKDB. Работа складского учёта возобновлена.

Критерии оценки практических заданий:
1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Практическое занятие №24. Восстановление базы данных MongoDB из резервного архива.
Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных
ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации
ПК 2.5. Подготавливать данные для базы знаний
ПК 2.6. Осуществлять управление базами данных

Цель: Научиться восстанавливать базы данных и отдельные коллекции MongoDB из архивов mongodump с помощью mongorestore, восстанавливать данные под другим именем, применять журнал операций и проверять результат восстановления.

Задание:
1. Проверить резервный архив и определить его содержимое.
2. Выполнить пробное восстановление без записи данных.
3. Восстановить базу данных в тестовое пространство имён.
4. Восстановить отдельную коллекцию в рабочую базу.
5. Выполнить полное восстановление с удалением существующих коллекций и применением журнала операций.
6. Отслеживать ход восстановления.
7. Проверить количество документов, индексы и целостность коллекций.
8. Оформить отчёт.

Этапы выполнения:
Этап 1. Проверка архива.
Проверяется контрольная сумма архива и дата создания. Определяется, создан ли архив со сжатием (--gzip) и с журналом операций (--oplog).

Этап 2. Пробное восстановление.
Выполняется mongorestore с параметром --dryRun и --verbose, который показывает, какие базы и коллекции будут восстановлены, без записи данных.

Этап 3. Восстановление в тестовое пространство имён.
С помощью параметров --nsFrom и --nsTo база восстанавливается под другим именем, например shop.* в shop_test.*, чтобы проверить данные без влияния на рабочую базу.

Этап 4. Восстановление коллекции.
С помощью параметра --nsInclude восстанавливается только нужная коллекция. При необходимости существующая коллекция предварительно переименовывается.

Этап 5. Полное восстановление.
Выполняется mongorestore с параметрами --drop (удаление коллекций перед восстановлением) и --oplogReplay (применение журнала операций из архива для согласованности на момент окончания копирования).

Этап 6. Мониторинг.
Ход восстановления отслеживается по выводу mongorestore: количество восстановленных документов по коллекциям, процент выполнения, ошибки. Нагрузка на сервер контролируется командой mongostat.

Этап 7. Проверка.
Сравнивается количество документов (countDocuments) с ожидаемым, проверяется наличие индексов (getIndexes), выполняется проверка коллекций командой db.коллекция.validate(), выполняются контрольные запросы.

Этап 8. Оформление.
Оформляется отчёт, содержащий команды, вывод mongorestore, результаты проверок и выводы.

Пример решения:
Задание: Восстановить ошибочно очищенную коллекцию отзывов в базе интернет-магазина.

Дано: Архив shop_2026-10-08_0200.gz создан командой mongodump --gzip --archive --oplog в наборе реплик. В 15:00 сотрудник ошибочно выполнил db.reviews.deleteMany({}) в рабочей базе shop. Остальные коллекции не пострадали.

Решение по этапам:
1. Проверяем контрольную сумму архива — OK, дата создания 2026-10-08 02:00.
2. Выполняем пробное восстановление: mongorestore --gzip --archive=shop_2026-10-08_0200.gz --dryRun --verbose — в архиве коллекции products, orders, reviews, customers базы shop.
3. Восстанавливаем базу в тестовое пространство имён: mongorestore --uri="$URI" --gzip --archive=shop_2026-10-08_0200.gz --nsFrom="shop.*" --nsTo="shop_test.*"
4. Проверяем в mongosh: use shop_test; db.reviews.countDocuments() — 48 215 отзывов.
5. Переименовываем пустую коллекцию в рабочей базе: db.reviews.renameCollection("reviews_empty").
6. Восстанавливаем только отзывы: mongorestore --uri="$URI" --gzip --archive=shop_2026-10-08_0200.gz --nsInclude="shop.reviews" — в выводе отображается количество восстановленных документов и сообщение «48215 document(s) restored successfully».
7. Проверяем: db.reviews.countDocuments() — 48 215, db.reviews.getIndexes() — индексы по product_id и created_at восстановлены, db.reviews.validate() — valid: true.
8. Отзывы, оставленные с 02:00 до 15:00, были удалены и отсутствуют в копии. Восстанавливаем их из журнала приложения, фиксируем это в отчёте.
9. Удаляем базу shop_test и коллекцию reviews_empty после подтверждения результата.
10. Оформляем отчёт и рекомендуем ограничить права сотрудников на удаление документов и добавить в набор реплик узел с задержкой репликации.

Ответ: Коллекция reviews восстановлена из архива mongodump: содержимое архива проверено пробным восстановлением, данные проверены в тестовом пространстве имён, затем коллекция выборочно восстановлена в рабочую базу с индексами. Проверка количества документов, индексов и команды validate подтвердила корректность. Отзывы, добавленные после копирования, восстановлены по журналу приложения.

Критерии оценки практических заданий:
1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Практическое занятие №25. Восстановление Oracle Database с использованием RMAN (Recovery Manager).

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных
ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации
ПК 2.5. Подготавливать данные для базы знаний

ПК 2.6. Осуществлять управление базами данных

Цель: Научиться выявлять повреждённые и утраченные файлы данных Oracle Database, восстанавливать их с помощью команд RMAN RESTORE и RECOVER без остановки всей базы, отслеживать ход восстановления и проверять результат.

Задание:
1. Смоделировать утрату файла данных табличного пространства.
2. Определить повреждённые файлы и проверить наличие нужных копий.
3. Перевести утраченный файл данных в автономный режим.
4. Восстановить файл данных командой RESTORE.
5. Применить журналы командой RECOVER.
6. Вернуть файл данных в рабочий режим.
7. Проверить данные и состояние базы, отследить ход восстановления.
8. Оформить отчёт.

Этапы выполнения:
Этап 1. Моделирование сбоя.
На учебной базе в режиме ARCHIVELOG с актуальной резервной копией создаётся табличное пространство с тестовой таблицей, затем файл данных этого табличного пространства удаляется или переименовывается средствами операционной системы.

Этап 2. Диагностика.
При обращении к таблице возникает ошибка о невозможности открыть файл. Запросами к V$RECOVER_FILE и V$DATAFILE определяются номер и имя утраченного файла. В RMAN командой LIST BACKUP OF DATAFILE проверяется наличие копии файла.

Этап 3. Перевод в автономный режим.
Файл данных переводится в автономный режим командой ALTER DATABASE DATAFILE номер OFFLINE, чтобы остальная база продолжала работать.

Этап 4. Восстановление файла.
В RMAN выполняется RESTORE DATAFILE номер — файл извлекается из резервной копии.

Этап 5. Применение журналов.
Выполняется RECOVER DATAFILE номер — к файлу применяются инкрементальные копии и архивные журналы, чтобы привести его к текущему состоянию.

Этап 6. Возврат в рабочий режим.
Файл переводится в рабочий режим командой ALTER DATABASE DATAFILE номер ONLINE.

Этап 7. Проверка и мониторинг.
Во время восстановления ход отслеживается в V$SESSION_LONGOPS и V$RMAN_STATUS. После восстановления проверяются данные таблицы, включая записи, добавленные после резервного копирования, и выполняется VALIDATE DATAFILE.

Этап 8. Оформление.
Оформляется отчёт, содержащий ход моделирования сбоя, диагностику, команды RMAN и их вывод, результаты проверки и выводы.

Пример решения:
Задание: Восстановить утраченный файл данных табличного пространства SALES без остановки базы данных.

Дано: База Oracle работает в режиме ARCHIVELOG, вчера выполнена полная копия RMAN. Сегодня в таблицу sales.orders табличного пространства SALES добавлено 300 записей, после чего файл данных sales01.dbf был удалён ошибкой администратора.

Решение по этапам:
1. При запросе SELECT COUNT(*) FROM sales.orders; возникает ошибка ORA-01116 о невозможности открыть файл данных.
2. Выполняем SELECT file#, error FROM v$recover_file; и SELECT file#, name FROM v$datafile; — утрачен файл номер 14 /u01/oradata/FREE/FREEPDB1/sales01.dbf.
3. Подключаемся rman target / и выполняем LIST BACKUP OF DATAFILE 14; — копия от вчерашнего дня существует.
4. Выполняем ALTER DATABASE DATAFILE 14 OFFLINE; — остальные табличные пространства продолжают работать.
5. Выполняем RESTORE DATAFILE 14; — RMAN восстанавливает файл из набора копий. В другом сеансе запрос к V$SESSION_LONGOPS показывает процент выполнения.
6. Выполняем RECOVER DATAFILE 14; — RMAN применяет архивные журналы, вывод содержит сообщение «media recovery complete».
7. Выполняем ALTER DATABASE DATAFILE 14 ONLINE;
8. Проверяем SELECT COUNT(*) FROM sales.orders; — количество записей включает 300 записей, добавленных после резервного копирования.
9. Выполняем VALIDATE DATAFILE 14; — повреждённых блоков нет. Запрос к V$RMAN_STATUS показывает успешное завершение операций.
10. Выполняем новую резервную копию и оформляем отчёт.

Ответ: Утраченный файл данных табличного пространства SALES восстановлен с помощью RMAN командами RESTORE и RECOVER без остановки базы данных: остальные пользователи продолжали работу. Благодаря архивным журналам восстановлены и данные, добавленные после резервного копирования, потери данных нет. Проверка VALIDATE подтвердила целостность файла.

Критерии оценки практических заданий:
1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Практическое занятие №26. Восстановление данных из резервной копии MySQL с проверкой целостности данных.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных
ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации
ПК 2.5. Подготавливать данные для базы знаний
ПК 2.6. Осуществлять управление базами данных

Цель: Научиться восстанавливать базу MySQL на определённый момент времени с использованием резервной копии mysqldump и двоичных журналов, а также проверять

целостность и согласованность восстановленных данных встроенными средствами MySQL и контрольными запросами.

Задание:
1. Определить момент ошибки, на который нужно восстановить данные.
2. Определить позицию двоичного журнала, соответствующую резервной копии.
3. Восстановить базу из резервной копии в отдельную базу.
4. Применить изменения из двоичных журналов до момента ошибки.
5. Проверить физическую целостность таблиц.
6. Проверить логическую целостность и согласованность данных.
7. Сравнить контрольные суммы таблиц и перенести восстановленные данные.
8. Оформить отчёт.

Этапы выполнения:
Этап 1. Определение момента ошибки.
По журналам приложения и двоичным журналам MySQL (утилита mysqlbinlog с параметрами --base64-output=DECODE-ROWS --verbose) находится время и позиция ошибочной команды.

Этап 2. Позиция двоичного журнала.
В начале файла копии, созданной с параметром --source-data=2 (в старых версиях --master-data=2), находится закомментированная команда с именем файла и позицией двоичного журнала на момент копии.

Этап 3. Восстановление копии.
Создаётся отдельная база, в неё загружается резервная копия клиентом mysql.

Этап 4. Применение двоичных журналов.
С помощью mysqlbinlog изменения применяются с позиции копии до момента, предшествующего ошибке (--stop-datetime или --stop-position). При восстановлении в базу с другим именем используется параметр --rewrite-db.

Этап 5. Физическая целостность.
Выполняются команды CHECK TABLE для таблиц или утилита mysqlcheck --check для базы. Все таблицы должны иметь статус OK.

Этап 6. Логическая целостность.
Контрольными запросами проверяется отсутствие записей с нарушенными ссылками (дочерние записи без родительских), отсутствие дубликатов в уникальных полях, соответствие итоговых сумм документов сумме их строк, допустимые значения полей.

Этап 7. Контрольные суммы и перенос.
Командой CHECKSUM TABLE рассчитываются контрольные суммы таблиц, не затронутых ошибкой, и сравниваются с рабочей базой. Восстановленные данные переносятся в рабочую базу в транзакции.

Этап 8. Оформление.
Оформляется отчёт, содержащий найденный момент ошибки, команды восстановления, результаты проверок целостности и выводы.

Пример решения:
Задание: Восстановить данные интернет-магазина после ошибочного удаления заказов.
Дано: Копия shop_2026-10-08.sql создана в 02:30 с параметром --source-data=2. Двоичные журналы включены. В 14:30 сотрудник выполнил DELETE FROM orders WHERE status = 'new' вместо обновления статуса, удалив 640 заказов, в том числе созданных после 02:30.

Решение по этапам:
1. Анализируем двоичный журнал: mysqlbinlog --base64-output=DECODE-ROWS --verbose binlog.000245 | grep -n "DELETE FROM `shop`.`orders`" — ошибочная операция выполнена в 14:30:12.
2. В начале копии находим строку: -- CHANGE REPLICATION SOURCE TO SOURCE_LOG_FILE='binlog.000244', SOURCE_LOG_POS=157;
3. Создаём базу shop_restore и загружаем копию: mysql shop_restore < shop_2026-10-08.sql
4. Применяем изменения до ошибки: mysqlbinlog --start-position=157 --stop-datetime="2026-10-08 14:30:11" --database=shop --rewrite-db="shop->shop_restore" binlog.000244 binlog.000245 | mysql
5. Проверяем физическую целостность: mysqlcheck --check shop_restore — все таблицы OK.
6. Проверяем ссылочную целостность: SELECT COUNT(*) FROM order_items i LEFT JOIN orders o ON o.id = i.order_id WHERE o.id IS NULL; — 0 записей. Проверяем, что сумма каждого заказа равна сумме его позиций, — расхождений нет.
7. Сравниваем CHECKSUM TABLE для таблицы products в shop_restore и shop — совпадают, значит, изменения до 14:30 применены полностью.
8. Определяем удалённые заказы: SELECT COUNT(*) FROM shop_restore.orders r LEFT JOIN shop.orders o ON o.id = r.id WHERE o.id IS NULL; — 640 записей.
9. Переносим в транзакции: INSERT INTO shop.orders SELECT r.* FROM shop_restore.orders r LEFT JOIN shop.orders o ON o.id = r.id WHERE o.id IS NULL; — вставлено 640 записей, повторно проверяем ссылочную целостность в рабочей базе.
10. Удаляем базу shop_restore и оформляем отчёт.

Ответ: С помощью резервной копии и двоичных журналов база восстановлена на момент 14:30:11, непосредственно перед ошибочным удалением. Проверки CHECK TABLE, ссылочной целостности, итоговых сумм и контрольных сумм таблиц подтвердили целостность и согласованность восстановленных данных. В рабочую базу возвращены все 640 удалённых заказов, включая созданные после резервного копирования.

Критерии оценки практических заданий:
1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Практическое занятие №27. Восстановление базы данных PostgreSQL на новый сервер с сохранением всех параметров.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных
ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации
ПК 2.5. Подготавливать данные для базы знаний
ПК 2.6. Осуществлять управление базами данных

Цель: Научиться переносить и восстанавливать базу данных PostgreSQL на новый сервер с сохранением ролей, прав, табличных пространств, расширений, параметров конфигурации, кодировки и локали, а также проверять соответствие параметров нового и старого серверов.

Задание:
1. Собрать сведения о параметрах исходного сервера и базы данных.
2. Создать резервные копии базы и глобальных объектов.
3. Подготовить новый сервер PostgreSQL той же или более новой версии.
4. Перенести параметры конфигурации и правила аутентификации.
5. Восстановить роли, табличные пространства и базу данных.
6. Проверить расширения, права и параметры базы и ролей.
7. Сравнить параметры и данные исходного и нового серверов.
8. Оформить отчёт.

Этапы выполнения:
Этап 1. Сбор сведений.
На исходном сервере определяются версия PostgreSQL, кодировка и локаль баз данных, установленные расширения (\dx), табличные пространства, изменённые параметры

конфигурации (запрос к pg_settings, где source не равен default), параметры уровня базы и ролей (ALTER DATABASE ... SET и ALTER ROLE ... SET), содержимое pg_hba.conf.

Этап 2. Резервные копии.
Выполняются pg_dumpall --globals-only для ролей и табличных пространств и pg_dump -Fc для базы данных. Копируются файлы postgresql.conf и pg_hba.conf. Параметры уровня базы данных сохраняются в pg_dump при использовании параметра --create.

Этап 3. Подготовка нового сервера.
Устанавливается PostgreSQL, кластер инициализируется с той же кодировкой и локалью. Устанавливаются пакеты расширений, используемых в базе. Создаются каталоги для табличных пространств.

Этап 4. Перенос конфигурации.
Изменённые параметры переносятся в postgresql.conf нового сервера с учётом его ресурсов и версии (параметры памяти пересчитываются). Переносятся правила pg_hba.conf с заменой адресов при необходимости. Сервер перезапускается.

Этап 5. Восстановление.
Выполняется psql -f globals.sql для создания ролей и табличных пространств, затем pg_restore --create -d postgres для создания базы с сохранёнными параметрами и восстановления данных.

Этап 6. Проверка объектов и прав.
Проверяются установленные расширения, владельцы объектов, права на схемы и таблицы, параметры уровня базы и ролей.

Этап 7. Сравнение серверов.
Сравниваются изменённые параметры конфигурации, кодировка и локаль, количество объектов, количество записей в таблицах, результаты контрольных запросов. Выполняется обновление статистики. Приложение переключается на новый сервер и проверяется.

Этап 8. Оформление.
Оформляется отчёт, содержащий таблицу сравнения параметров, выполненные команды, результаты проверок и выводы.

Пример решения:
Задание: Перенести базу данных отдела кадров на новый сервер с сохранением всех параметров.
Дано: Исходный сервер PostgreSQL 15, база hr с кодировкой UTF8 и локалью ru_RU.UTF-8, расширения pgcrypto и uuid-ossp, роли hr_admin, hr_user, табличное пространство hr_archive.

Для базы задан параметр search_path. Новый сервер имеет вдвое больше оперативной памяти и PostgreSQL 16.

Решение по этапам:
1. На исходном сервере выполняем SELECT name, setting FROM pg_settings WHERE source = 'configuration file'; и сохраняем результат. Выполняем \l hr — UTF8, ru_RU.UTF-8, \dx — pgcrypto, uuid-ossp, \db — hr_archive в /data/hr_archive.
2. Выполняем pg_dumpall -U postgres --globals-only -f globals.sql и pg_dump -U postgres -Fc --create -f hr.dump hr. Копируем postgresql.conf и pg_hba.conf.
3. На новом сервере устанавливаем PostgreSQL 16 и пакет contrib с расширениями, инициализируем кластер с параметрами --encoding=UTF8 --locale=ru_RU.UTF-8, создаём каталог /data/hr_archive с владельцем postgres.
4. Переносим изменённые параметры в postgresql.conf, увеличиваем shared_buffers и effective_cache_size пропорционально памяти. Переносим правила pg_hba.conf, перезапускаем службу.
5. Восстанавливаем глобальные объекты: psql -U postgres -f globals.sql — созданы роли и табличное пространство.
6. Восстанавливаем базу: pg_restore -U postgres --create -d postgres -j 4 hr.dump — база создана с исходной кодировкой и параметром search_path.
7. Проверяем: \dx в базе hr — расширения установлены; SELECT datname, setconfig FROM pg_db_role_setting JOIN pg_database ON oid = setdatabase; — search_path сохранён; права роли hr_user на таблицы совпадают с исходными; таблица archive_records находится в табличном пространстве hr_archive.
8. Сравниваем количество записей во всех таблицах запросом к pg_stat_user_tables после ANALYZE — совпадает. Сравниваем изменённые параметры конфигурации — отличаются только параметры памяти, изменённые намеренно.
9. Переключаем приложение отдела кадров на новый сервер, проверяем вход пользователей и формирование отчётов.
10. Оформляем отчёт с таблицей сравнения параметров.

Ответ: База данных hr перенесена на новый сервер PostgreSQL 16 с сохранением ролей, прав, табличного пространства, расширений, кодировки, локали, параметров уровня базы и правил аутентификации. Параметры памяти пересчитаны с учётом ресурсов нового сервера. Сравнение параметров и данных подтвердило полное соответствие, приложение успешно работает с новым сервером.

Критерии оценки практических заданий:
1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Практическое занятие №28. Выполнение восстановления базы данных Microsoft SQL Server из дифференциальной резервной копии.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках

ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных
ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации
ПК 2.5. Подготавливать данные для базы знаний
ПК 2.6. Осуществлять управление базами данных

Цель: Научиться определять необходимую цепочку резервных копий SQL Server и восстанавливать базу данных из полной, дифференциальной копий и копий журнала транзакций, в том числе на определённый момент времени, с помощью SSMS и T-SQL.

Задание:
1. Изучить историю резервных копий базы и определить цепочку для восстановления.
2. Восстановить полную копию с параметром NORECOVERY.
3. Восстановить последнюю дифференциальную копию.
4. Применить копии журнала транзакций до нужного момента времени.
5. Завершить восстановление и открыть базу.
6. Выполнить то же восстановление с помощью мастера SSMS и сравнить план восстановления.
7. Проверить результат, целостность и время восстановления.
8. Оформить отчёт.

Этапы выполнения:
Этап 1. Определение цепочки.
Запросом к msdb.dbo.backupset и msdb.dbo.backupmediafamily определяются копии базы: полные (тип D), дифференциальные (тип I) и журнала (тип L), их время и файлы. Для восстановления выбираются последняя полная копия до нужного момента, последняя дифференциальная копия после неё и копии журнала после дифференциальной.

Этап 2. Восстановление полной копии.
Выполняется RESTORE DATABASE ... FROM DISK ... WITH NORECOVERY, при необходимости с параметрами MOVE и REPLACE. База остаётся в состоянии «Восстановление».

Этап 3. Восстановление дифференциальной копии.
Выполняется RESTORE DATABASE ... FROM DISK = 'файл дифференциальной копии' WITH NORECOVERY. Промежуточные дифференциальные копии не применяются.

Этап 4. Применение журналов.
Последовательно применяются копии журнала командой RESTORE LOG ... WITH NORECOVERY. Для последней копии указывается параметр STOPAT с моментом времени, на который нужно восстановить базу.

Этап 5. Завершение восстановления.
Выполняется RESTORE DATABASE ... WITH RECOVERY, база становится доступной.

Этап 6. Восстановление через SSMS.
В мастере восстановления на странице «Общие» выбирается база-источник, кнопкой «Временная шкала» задаётся момент восстановления, и мастер автоматически формирует

план восстановления из нужных копий. План сравнивается с самостоятельно определённой цепочкой.

Этап 7. Проверка.
Проверяется наличие данных на заданный момент и отсутствие изменений после него, выполняется DBCC CHECKDB, фиксируется время восстановления и сравнивается с RTO.

Этап 8. Оформление.
Оформляется отчёт, содержащий цепочку копий, скрипт восстановления, снимок плана восстановления SSMS, результаты проверки и выводы.

Пример решения:
Задание: Восстановить базу заявок на момент, предшествующий ошибочному массовому изменению данных.
Дано: Для базы Requests в полной модели восстановления выполняются полная копия по воскресеньям, дифференциальные копии ежедневно в 01:00 и копии журнала каждые 30 минут. В среду в 10:47 ошибочный запрос изменил статус всех заявок на «Закрыта».

Решение по этапам:
1. Запрос к msdb.dbo.backupset показывает: полная копия — воскресенье 01:00, дифференциальные — понедельник, вторник, среда 01:00, копии журнала в среду — 01:30 … 10:30, 11:00.
2. Определяем цепочку: полная копия воскресенья, дифференциальная копия среды 01:00, копии журнала среды с 01:30 до 11:00. Дифференциальные копии понедельника и вторника не нужны.
3. Восстанавливаем в отдельную базу: RESTORE DATABASE Requests_pit FROM DISK = 'D:\Backup\Requests_full_sun.bak' WITH MOVE 'Requests' TO 'D:\Data\Requests_pit.mdf', MOVE 'Requests_log' TO 'D:\Data\Requests_pit_log.ldf', NORECOVERY, STATS = 10;
4. RESTORE DATABASE Requests_pit FROM DISK = 'D:\Backup\Requests_diff_wed.bak' WITH NORECOVERY;
5. Последовательно применяем 19 копий журнала с NORECOVERY, для копии 11:00 указываем: RESTORE LOG Requests_pit FROM DISK = 'D:\Backup\Requests_log_1100.trn' WITH STOPAT = '2026-10-07 10:46:59', RECOVERY;
6. Проверяем в Requests_pit: заявки имеют корректные статусы, а заявки, созданные до 10:46, присутствуют.
7. Сравниваем с мастером SSMS: при выборе на временной шкале момента 10:46:59 мастер формирует тот же план — полная, дифференциальная копия среды и 19 копий журнала.
8. Выполняем DBCC CHECKDB ('Requests_pit') — ошибок нет. Время восстановления — 9 минут, что укладывается в RTO 1 час.
9. Переносим правильные статусы в рабочую базу запросом UPDATE с соединением по номеру заявки, сохраняя изменения, сделанные после 10:47 другими пользователями.
10. Оформляем отчёт.

Ответ: Определена минимальная цепочка копий: полная, последняя дифференциальная и копии журнала. База восстановлена на момент 10:46:59, непосредственно перед ошибочным изменением, с использованием параметра STOPAT. План, сформированный мастером SSMS,

совпал с самостоятельно определённой цепочкой. Использование дифференциальной копии сократило количество применяемых журналов и время восстановления до 9 минут.

Критерии оценки практических заданий:

1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Практическое занятие №29. Настройка сценария аварийного восстановления базы данных MongoDB.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных
ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации
ПК 2.5. Подготавливать данные для базы знаний
ПК 2.6. Осуществлять управление базами данных

Цель: Научиться проектировать и настраивать сценарий аварийного восстановления MongoDB с использованием набора реплик, распределённого между площадками, узла с

задержкой репликации и резервных копий, а также выполнять переключение и восстановление при авариях.

Задание:
1. Определить требования к аварийному восстановлению базы данных.
2. Спроектировать размещение узлов набора реплик на основной и резервной площадках.
3. Настроить набор реплик с приоритетами узлов и узлом с задержкой репликации.
4. Разработать пошаговые сценарии восстановления для разных видов аварий.
5. Протестировать сценарий отказа основного узла.
6. Протестировать сценарий потери основной площадки с принудительной переконфигурацией.
7. Протестировать восстановление после логической ошибки с помощью узла с задержкой.
8. Оценить фактические RPO и RTO и оформить план аварийного восстановления.

Этапы выполнения:
Этап 1. Требования.
Определяются критичность базы, допустимые RPO и RTO, возможные аварии: отказ сервера, потеря площадки, логическая ошибка, атака шифровальщика.

Этап 2. Проектирование.
Определяется количество и размещение узлов: основная площадка содержит узлы с высоким приоритетом, резервная — узлы с низким приоритетом и скрытый узел с задержкой. Учитывается, что для выбора основного узла требуется большинство голосов.

Этап 3. Настройка набора реплик.
Набор реплик настраивается командами rs.initiate() или rs.reconfig() с параметрами priority для каждого узла. Для узла с задержкой задаются priority: 0, hidden: true и secondaryDelaySecs. Настраивается запись с подтверждением большинства (writeConcern majority).

Этап 4. Разработка сценариев.
Для каждого вида аварии описываются признаки, действия, команды, проверки и ответственные. Сценарии включают проверку состояния командой rs.status(), переключение приложения, восстановление из резервной копии.

Этап 5. Тест отказа узла.
Основной узел останавливается, проверяется автоматический выбор нового основного узла, время переключения и продолжение записи.

Этап 6. Тест потери площадки.
Останавливаются все узлы основной площадки. Так как большинство голосов утрачено, оставшиеся узлы не могут выбрать основной узел. Выполняется принудительная

переконфигурация rs.reconfig(cfg, {force: true}) с оставшимися узлами, приложение переключается на резервную площадку.

Этап 7. Тест логической ошибки.
На основном узле выполняется ошибочное удаление данных. На узле с задержкой, который ещё не получил это изменение, репликация приостанавливается путём его остановки, данные выгружаются утилитой mongodump и восстанавливаются в рабочую базу.

Этап 8. Оценка и оформление.
Фиксируются фактическое время переключения и восстановления и объём потерянных данных, сравниваются с требованиями. Оформляется план аварийного восстановления.

Пример решения:
Задание: Настроить аварийное восстановление базы заказов MongoDB для компании с двумя центрами обработки данных.
Дано: Основная площадка — ЦОД-1, резервная — ЦОД-2. Требования: при отказе одного сервера RTO — 1 минута, при потере ЦОД-1 RTO — 30 минут, RPO — не более 5 минут. Имеются ежедневные копии mongodump.

Решение по этапам:
1. Проектируем набор реплик rs0 из пяти узлов: в ЦОД-1 — db1 (priority 3) и db2 (priority 2), в ЦОД-2 — db3 (priority 1), db4 (priority 1) и db5 со скрытым режимом и задержкой 1 час.
2. Настраиваем: rs.initiate({ _id: "rs0", members: [{_id: 0, host: "db1:27017", priority: 3}, {_id: 1, host: "db2:27017", priority: 2}, {_id: 2, host: "db3:27017", priority: 1}, {_id: 3, host: "db4:27017", priority: 1}, {_id: 4, host: "db5:27017", priority: 0, hidden: true, secondaryDelaySecs: 3600}] })
3. В строке подключения приложения указываем все узлы и параметр w=majority.
4. Разрабатываем сценарии: «Отказ сервера», «Потеря ЦОД-1», «Логическая ошибка», «Атака шифровальщика» с командами и проверками.
5. Тест 1: останавливаем db1. Через 11 секунд db2 становится основным, запись продолжается. После запуска db1 он снова становится основным благодаря высокому приоритету.
6. Тест 2: останавливаем db1 и db2. Оставшиеся три узла из пяти составляют большинство, и db3 автоматически становится основным за 12 секунд. Затем дополнительно останавливаем db4 — у двух оставшихся узлов нет большинства, основной узел отсутствует.
7. Выполняем принудительную переконфигурацию на db3: cfg = rs.conf(); cfg.members = cfg.members.filter(m => ["db3:27017", "db5:27017"].includes(m.host)); rs.reconfig(cfg, {force: true}); и временно задаём db5 priority 0 без задержки. db3 становится основным, приложение работает на ЦОД-2, время выполнения сценария — 18 минут.
8. Тест 3: восстанавливаем исходную конфигурацию и на основном узле выполняем db.orders.deleteMany({status: "new"}). Через 10 минут останавливаем db5, запускаем его как отдельный экземпляр без параметра replSet, выгружаем коллекцию orders утилитой mongodump и восстанавливаем удалённые заказы в рабочую базу.
9. Оцениваем: при отказе сервера RTO — 11 секунд, при потере площадки — 18 минут, RPO — 0 при записи с подтверждением большинства. Требования выполнены.
10. Оформляем план аварийного восстановления.

Ответ: Настроен набор реплик MongoDB из пяти узлов на двух площадках с приоритетами и скрытым узлом с задержкой репликации 1 час. Разработаны и протестированы сценарии отказа сервера, потери основной площадки с принудительной переконфигурацией и

восстановления после логической ошибки. Фактические RTO (11 секунд и 18 минут) и RPO соответствуют требованиям, план аварийного восстановления оформлен.

Критерии оценки практических заданий:
1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Практическое занятие №30. Разработка и тестирование сценария восстановления Oracle Database после сбоя.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных
ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации
ПК 2.5. Подготавливать данные для базы знаний
ПК 2.6. Осуществлять управление базами данных

Цель: Научиться разрабатывать пошаговый сценарий полного восстановления Oracle Database после потери сервера или всех файлов базы данных и тестировать его на резервном

сервере с восстановлением файла параметров, управляющего файла, файлов данных и применением журналов с помощью RMAN.

Задание:
1. Определить сценарий аварии и требования к восстановлению.
2. Собрать сведения, необходимые для восстановления: DBID, расположение копий, структуру каталогов.
3. Разработать пошаговый сценарий восстановления.
4. Подготовить резервный сервер для тестирования.
5. Восстановить файл параметров и управляющий файл из автокопии.
6. Восстановить файлы данных и применить журналы.
7. Открыть базу данных, проверить результат и отследить ход восстановления.
8. Оценить фактические RPO и RTO и обновить сценарий.

Этапы выполнения:
Этап 1. Сценарий аварии.
Определяется авария, например полная потеря сервера базы данных, и требования: допустимые RPO и RTO, приоритет восстановления базы.

Этап 2. Сбор сведений.
Заранее фиксируются идентификатор базы данных DBID (из вывода RMAN или V$DATABASE), имя базы, версия Oracle, расположение резервных копий и архивных журналов, структура каталогов, параметры автокопии управляющего файла.

Этап 3. Разработка сценария.
Сценарий содержит шаги: подготовка сервера и установка той же версии Oracle, настройка переменных окружения, доступ к резервным копиям, восстановление файла параметров и управляющего файла, восстановление и восстановление с применением журналов, открытие базы с RESETLOGS, проверки, переключение приложений, создание новой копии. Для каждого шага указываются команды и ожидаемый результат.

Этап 4. Подготовка резервного сервера.
На резервном сервере устанавливается та же версия Oracle, создаются каталоги, настраиваются ORACLE_HOME и ORACLE_SID, предоставляется доступ к резервным копиям.

Этап 5. Файл параметров и управляющий файл.
В RMAN задаётся DBID командой SET DBID, экземпляр запускается в режиме NOMOUNT, выполняется RESTORE SPFILE FROM AUTOBACKUP, экземпляр перезапускается с восстановленным файлом параметров, выполняется RESTORE CONTROLFILE FROM AUTOBACKUP и база монтируется.

Этап 6. Файлы данных и журналы.
Выполняются RESTORE DATABASE и RECOVER DATABASE. RMAN применяет инкрементальные копии и доступные архивные журналы. При отсутствии последних журналов восстановление выполняется до последнего доступного журнала.

Этап 7. Открытие и проверка.
База открывается командой ALTER DATABASE OPEN RESETLOGS. Ход восстановления отслеживается через V$SESSION_LONGOPS и V$RMAN_STATUS. Проверяются состояние

экземпляра и подключаемых баз, наличие данных на последний момент, работа слушателя, выполняется VALIDATE DATABASE.

Этап 8. Оценка и обновление.
Фиксируется время каждого шага и общее время, определяется объём потерянных данных. Сценарий дополняется замечаниями, выявленными при тестировании.

Пример решения:
Задание: Разработать и протестировать сценарий восстановления базы данных Oracle учётной системы после полной потери сервера.
Дано: База PROD, режим ARCHIVELOG, еженедельная копия уровня 0, ежедневные копии уровня 1, архивные журналы копируются на сетевое хранилище каждые 15 минут, автокопия управляющего файла включена. Требования: RTO — 4 часа, RPO — 15 минут. Для теста используется резервный сервер с той же версией Oracle.

Решение по этапам:
1. Фиксируем в сценарии DBID базы — 1234567890 (из вывода rman target / при подключении), пути к копиям /backup/rman и к архивным журналам /backup/arch, структуру каталогов /u01/oradata/PROD.
2. Разрабатываем сценарий из 12 шагов с командами, ожидаемыми результатами и ответственными.
3. На резервном сервере создаём каталоги, задаём ORACLE_SID=PROD и ORACLE_HOME, подключаем сетевое хранилище с копиями.
4. Запускаем rman target / и выполняем: SET DBID 1234567890; STARTUP NOMOUNT; RESTORE SPFILE FROM AUTOBACKUP DB_RECOVERY_FILE_DEST='/backup/rman' DB_NAME='PROD'; STARTUP FORCE NOMOUNT;
5. Выполняем RESTORE CONTROLFILE FROM AUTOBACKUP; ALTER DATABASE MOUNT; CATALOG START WITH '/backup/'; — RMAN регистрирует копии и архивные журналы.
6. Выполняем RESTORE DATABASE; — в другом сеансе запрос к V$SESSION_LONGOPS показывает процент восстановления файлов данных. Восстановление заняло 1 час 40 минут.
7. Выполняем RECOVER DATABASE; — RMAN применяет копию уровня 1 и архивные журналы до последнего доступного, затем сообщает об отсутствии следующего журнала, что ожидаемо.
8. Выполняем ALTER DATABASE OPEN RESETLOGS; — база открыта. Проверяем V$INSTANCE (OPEN), SHOW PDBS, lsnrctl status, наличие последних записей в таблице документов — последний документ создан за 11 минут до момента «аварии».
9. Выполняем VALIDATE DATABASE — ошибок нет. Общее время теста — 2 часа 35 минут, потеря данных — 11 минут. Выявлено, что в сценарии не был описан шаг регистрации копий командой CATALOG, добавляем его.
10. Выполняем новую полную копию восстановленной базы, обновляем сценарий и оформляем отчёт о тестировании.

Ответ: Разработан пошаговый сценарий восстановления Oracle Database после полной потери сервера и протестирован на резервном сервере: из автокопии восстановлены файл параметров и управляющий файл, восстановлены файлы данных, применены инкрементальная копия и архивные журналы, база открыта с RESETLOGS. Фактическое RTO составило 2 часа 35

минут, RPO — 11 минут, что соответствует требованиям. По результатам тестирования сценарий дополнен пропущенным шагом.

Критерии оценки практических заданий:
1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Тема 1.6. Мониторинг событий, возникающих в процессе работы баз данных.
Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных
ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации
ПК 2.5. Подготавливать данные для базы знаний
ПК 2.7. Осуществлять автоматизацию баз данных

Вопросы к устному опросу по теме:
1. Что такое мониторинг работы баз данных?
Ответ: Мониторинг работы баз данных — это постоянное наблюдение за состоянием и производительностью СУБД и баз данных: доступностью сервера, нагрузкой, выполнением запросов, использованием ресурсов, ошибками, блокировками и событиями безопасности. Его цель — своевременно обнаружить отклонения от штатного режима, устранить их до того, как они повлияют на пользователей, и накопить данные для планирования.

2. Какие задачи решает мониторинг баз данных?
Ответ: Мониторинг позволяет контролировать доступность и производительность баз данных, выявлять медленные запросы, блокировки и взаимоблокировки, обнаруживать ошибки и сбои, отслеживать заполнение дисков и рост данных, контролировать резервное копирование и репликацию, выявлять подозрительные действия пользователей и планировать развитие ресурсов.

3. Какие основные показатели работы СУБД отслеживаются?
Ответ: Отслеживаются доступность сервера, количество подключений и активных сеансов, количество запросов и транзакций в секунду, время выполнения запросов, доля попаданий в кэш, количество блокировок и ожиданий, использование процессора, памяти, дисков и сети, размер баз данных и журналов, отставание реплик, количество ошибок и неудачных попыток входа.

4. Что такое штатный режим работы базы данных и базовая линия (baseline)?
Ответ: Штатный режим — это нормальное состояние работы базы данных при обычной нагрузке, когда показатели находятся в допустимых пределах. Базовая линия — это набор типичных значений показателей для разных периодов (рабочий день, ночь, конец месяца),

полученных при наблюдении. Отклонение текущих значений от базовой линии указывает на возможную проблему.

5. Какие отклонения от штатного режима работы базы данных встречаются чаще всего?
Ответ: Чаще всего встречаются резкое замедление запросов, рост числа подключений до предела, длительные блокировки и взаимоблокировки, транзакции, долго находящиеся в незавершённом состоянии, нехватка места на диске, рост журналов, отставание реплик, ошибки в журналах, массовые неудачные попытки входа, неожиданные изменения структуры базы.

6. Какими способами обнаруживаются отклонения?
Ответ: Отклонения обнаруживаются сравнением показателей с пороговыми значениями и базовой линией, анализом журналов СУБД на наличие ошибок и предупреждений, автоматическими проверками систем мониторинга с оповещениями, событийными триггерами, жалобами пользователей, а также методами обнаружения аномалий, выявляющими необычные изменения показателей.

7. Каков порядок анализа и устранения отклонений?
Ответ: Фиксируется факт и время отклонения, собираются данные: показатели, журналы, активные сеансы и запросы. Определяется причина — конкретный запрос, блокировка, нехватка ресурса, ошибка конфигурации или внешнее событие. Выполняются неотложные действия для восстановления работы, затем устраняется первопричина. Результат проверяется, инцидент документируется, при необходимости корректируются пороги мониторинга.

8. Что такое Performance Schema в MySQL?
Ответ: Performance Schema — это встроенный механизм MySQL для сбора сведений о выполнении сервера на низком уровне: выполнении запросов, ожиданиях, блокировках, вводе-выводе, использовании памяти, подключениях. Данные хранятся в памяти в таблицах базы performance_schema. Сбор настраивается таблицами setup_instruments (что измерять) и setup_consumers (куда сохранять).

9. Что такое схема sys в MySQL?
Ответ: Схема sys — это набор представлений, функций и процедур, которые представляют данные Performance Schema и Information Schema в удобном для администратора виде. Например, представление sys.statement_analysis показывает самые ресурсоёмкие запросы, sys.statements_with_full_table_scans — запросы с полным сканированием таблиц, sys.schema_unused_indexes — неиспользуемые индексы, sys.innodb_lock_waits — ожидания блокировок.

10. Что показывает представление pg_stat_activity в PostgreSQL?
Ответ: Представление pg_stat_activity содержит строку для каждого процесса сервера: идентификатор процесса, пользователя, базу данных, адрес клиента, имя приложения, состояние (active, idle, idle in transaction), текст текущего или последнего запроса, время начала запроса и транзакции, тип и событие ожидания. Оно используется для анализа текущих подключений и запросов.

11. Как с помощью pg_stat_activity найти проблемные сеансы?
Ответ: Длительные запросы находятся по разнице текущего времени и query_start для сеансов в состоянии active, незавершённые транзакции — по состоянию idle in transaction и времени xact_start, заблокированные сеансы — по wait_event_type = 'Lock' и функции

pg_blocking_pids(). Проблемный запрос отменяется функцией pg_cancel_backend(), сеанс завершается функцией pg_terminate_backend().

12. Что такое расширение pg_stat_statements?
Ответ: pg_stat_statements — это расширение PostgreSQL, накапливающее статистику выполнения всех запросов: количество вызовов, общее и среднее время, количество строк, обращения к буферам. Запросы с разными значениями параметров объединяются по шаблону. Расширение подключается через параметр shared_preload_libraries и позволяет найти самые медленные и самые частые запросы.

13. Какие средства мониторинга встроены в Microsoft SQL Server?
Ответ: В SQL Server есть динамические административные представления (DMV), например sys.dm_exec_requests, sys.dm_exec_sessions, sys.dm_exec_query_stats, sys.dm_os_wait_stats, монитор активности в SSMS, хранилище запросов Query Store, расширенные события (Extended Events), механизм SQL Server Audit, журнал ошибок и отчёты панели мониторинга.

14. Какие средства мониторинга есть в Oracle Database?
Ответ: В Oracle используются динамические представления V$SESSION, V$SQL, V$SYSTEM_EVENT, V$SESSION_LONGOPS, журнал предупреждений alert.log, репозиторий рабочей нагрузки AWR и автоматический диагностический монитор ADDM (требуют лицензии Diagnostics Pack), бесплатная альтернатива Statspack, а также Oracle Enterprise Manager для графического мониторинга.

15. Для чего используются журналы (логи) СУБД при мониторинге?
Ответ: Журналы содержат сведения о запуске и остановке сервера, ошибках, предупреждениях, подключениях и отключениях, медленных запросах, блокировках, контрольных точках, работе автоочистки и действиях пользователей. Анализ журналов позволяет обнаружить отклонения, установить их причины и время, а также расследовать инциденты безопасности.

16. Какие параметры журналирования настраиваются в PostgreSQL?
Ответ: Настраиваются назначение журнала log_destination (stderr, csvlog, jsonlog), сборщик журналов logging_collector, каталог и имя файлов log_directory и log_filename, ротация log_rotation_age и log_rotation_size, префикс строки log_line_prefix, запись медленных запросов log_min_duration_statement, запись команд log_statement, подключений log_connections и log_disconnections, ожиданий блокировок log_lock_waits, временных файлов log_temp_files.

17. Что такое log_line_prefix и какие сведения в него включают?
Ответ: log_line_prefix — это параметр PostgreSQL, задающий формат начала каждой строки журнала с помощью специальных последовательностей: %m — время с миллисекундами, %p — номер процесса, %u — пользователь, %d — база данных, %r — адрес и порт клиента, %a — имя приложения, %e — код ошибки SQLSTATE, %c — идентификатор сеанса. Подробный префикс позволяет быстро определить источник события.

18. Что такое событийные триггеры в PostgreSQL?
Ответ: Событийные триггеры — это триггеры, которые срабатывают не на изменение данных таблицы, а на события уровня базы данных: начало и окончание выполнения DDL-команд (ddl_command_start, ddl_command_end), удаление объектов (sql_drop), перезапись таблицы (table_rewrite), а в новых версиях — вход в базу данных (login). Они используются для аудита изменений структуры, запрета опасных команд и отправки оповещений.

19. Как организовать оповещения из базы данных PostgreSQL?
Ответ: Функция триггера записывает событие в таблицу оповещений и отправляет уведомление командой pg_notify() в именованный канал. Внешняя программа подписывается

на канал командой LISTEN и при получении уведомления отправляет сообщение администратору по электронной почте или в мессенджер. Также можно периодически опрашивать таблицу оповещений внешней системой мониторинга.

20. Как протоколируются события доступа к данным в SQL Server?
Ответ: Для протоколирования используется механизм SQL Server Audit. Создаётся объект аудита сервера с указанием места записи (файл, журнал безопасности или журнал приложений Windows), затем спецификация аудита сервера для событий входа и изменения прав и спецификация аудита базы данных для доступа к таблицам (SELECT, INSERT, UPDATE, DELETE). Журнал читается функцией sys.fn_get_audit_file или через SSMS.

21. Чем внутренний мониторинг СУБД отличается от внешнего?
Ответ: Внутренний мониторинг выполняется встроенными средствами СУБД: системными представлениями, журналами, Performance Schema, AWR. Он даёт подробные сведения, но требует подключения к каждому серверу. Внешний мониторинг выполняется отдельными системами (Zabbix, Prometheus, Grafana), которые централизованно собирают показатели многих серверов, хранят историю, строят графики и отправляют оповещения.

22. Что такое Zabbix и как он используется для мониторинга СУБД?
Ответ: Zabbix — это система мониторинга с открытым исходным кодом. На серверы устанавливается агент Zabbix, который собирает показатели операционной системы и СУБД. Для MySQL, PostgreSQL, MongoDB, SQL Server и Oracle существуют готовые шаблоны с элементами данных, триггерами и графиками. Триггеры Zabbix определяют проблемы по условиям, а действия отправляют оповещения.

23. Как устроен мониторинг СУБД с помощью Prometheus и Grafana?
Ответ: Для каждой СУБД запускается экспортёр, который получает показатели из СУБД и предоставляет их в формате Prometheus: mysqld_exporter, postgres_exporter, mongodb_exporter, экспортёры для SQL Server и Oracle. Prometheus периодически собирает показатели и хранит их как временные ряды, правила оповещения передают предупреждения в Alertmanager, а Grafana отображает показатели на дашбордах.

24. Какие требования предъявляются к учётной записи системы мониторинга в СУБД?
Ответ: Для системы мониторинга создаётся отдельная учётная запись только с правами, необходимыми для чтения статистики, без доступа к пользовательским данным. Например, в PostgreSQL используется встроенная роль pg_monitor, в MySQL — права PROCESS, REPLICATION CLIENT и SELECT на performance_schema, в SQL Server — право VIEW SERVER STATE. Пароль хранится в защищённом виде.

25. Как настраиваются пороги и оповещения, чтобы избежать «усталости от оповещений»?
Ответ: Пороги устанавливаются на основе базовой линии, а не произвольно. Используются уровни важности (предупреждение, критично), задержка срабатывания, чтобы кратковременные всплески не вызывали оповещений, группировка связанных оповещений и подавление зависимых. Оповещения направляются только ответственным и содержат понятное описание и рекомендации. Ненужные оповещения регулярно пересматриваются.

26. Что такое проактивный анализ производительности СУБД?
Ответ: Проактивный анализ — это регулярное исследование показателей и запросов с целью выявления проблем до того, как они приведут к сбоям или жалобам. Он включает анализ тенденций роста нагрузки и данных, поиск медленных и неэффективных запросов,

неиспользуемых и отсутствующих индексов, анализ ожиданий, проверку конфигурации и планирование ресурсов.

27. Что такое предиктивная аналитика в мониторинге СУБД?
Ответ: Предиктивная аналитика — это прогнозирование будущих значений показателей на основе исторических данных, например момента заполнения диска, превышения лимита подключений или роста времени отклика. Применяются линейная экстраполяция, учёт сезонности и методы машинного обучения для обнаружения аномалий. Это позволяет заранее увеличить ресурсы или оптимизировать систему.

28. Какие средства прогнозирования есть в системах мониторинга?
Ответ: В Prometheus используется функция predict_linear(), прогнозирующая значение показателя через заданное время, например заполнение диска через 24 часа. В Zabbix используются функции триггеров forecast() и timeleft(), вычисляющие прогнозное значение и время до достижения порога. В Grafana и других системах есть средства обнаружения аномалий и прогнозирования на основе машинного обучения.

29. Что такое хранилище запросов (Query Store) в SQL Server?
Ответ: Query Store — это механизм SQL Server, сохраняющий историю запросов, их планов выполнения и статистики времени выполнения. Он позволяет найти запросы, производительность которых ухудшилась, например из-за смены плана, сравнить планы, принудительно закрепить хороший план и анализировать тенденции нагрузки за длительный период.

30. Как организовать комплексный мониторинг баз данных в организации?
Ответ: Определяются критичные базы и ключевые показатели, собирается базовая линия, настраиваются встроенные средства (журналы, статистика запросов, аудит) и внешняя система мониторинга с дашбордами и оповещениями, назначаются ответственные и порядок реагирования, разрабатываются инструкции по типовым отклонениям. Регулярно выполняется проактивный анализ, а результаты используются для оптимизации и планирования ресурсов.

Критерии оценки устного опроса:

- оценка «отлично» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов и смог ответить на уточняющие вопросы;
- оценка «хорошо» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов, но не смог ответить на уточняющие вопросы;
- оценка «удовлетворительно» выставляется студенту, если он смог изложить суть основного вопроса, но не смог ответить по существу дополнительных вопросов;
- оценка «неудовлетворительно» выставляется студенту, если он не смог изложить суть основного вопроса.

Практическое занятие №31. Настройка и использование утилиты MySQL Performance Schema для мониторинга работы базы данных. Использование утилиты pg_stat_activity в PostgreSQL для отслеживания активных соединений и запросов.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста

ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных
ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации
ПК 2.5. Подготавливать данные для базы знаний
ПК 2.7. Осуществлять автоматизацию баз данных

Цель: Научиться настраивать Performance Schema в MySQL и использовать её таблицы и представления схемы sys для анализа запросов, ожиданий и блокировок, а также отслеживать активные соединения, длительные запросы и блокировки в PostgreSQL с помощью pg_stat_activity.

Задание:
1. Проверить включение Performance Schema и настроить инструменты и потребители событий.
2. Создать тестовую нагрузку на базу MySQL.
3. Найти самые ресурсоёмкие запросы и запросы с полным сканированием таблиц.
4. Выявить блокировки и неиспользуемые индексы в MySQL.
5. Просмотреть активные подключения PostgreSQL в pg_stat_activity.
6. Найти длительные запросы и незавершённые транзакции в PostgreSQL.
7. Смоделировать блокировку, найти блокирующий сеанс и устранить проблему.
8. Сделать вывод и сформулировать рекомендации.

Этапы выполнения:
Этап 1. Настройка Performance Schema.
Проверяется параметр performance_schema командой SHOW VARIABLES. В таблице setup_instruments включаются нужные инструменты (например, statement/% и wait/lock/%), в

таблице setup_consumers — потребители, например events_statements_history_long. Для сохранения настроек после перезапуска параметры указываются в конфигурационном файле.

Этап 2. Тестовая нагрузка.
Выполняется набор запросов к базе, в том числе запросы без подходящих индексов, или используется утилита нагрузочного тестирования mysqlslap.

Этап 3. Анализ запросов.
Запросами к таблице events_statements_summary_by_digest и представлениям sys.statement_analysis и sys.statements_with_full_table_scans находятся запросы с наибольшим общим временем выполнения и запросы с полным сканированием таблиц.

Этап 4. Блокировки и индексы.
С помощью представлений sys.innodb_lock_waits и таблиц data_locks и data_lock_waits анализируются ожидания блокировок. Представлением sys.schema_unused_indexes находятся неиспользуемые индексы.

Этап 5. Активные подключения PostgreSQL.
Выполняется запрос к pg_stat_activity с выбором полей pid, usename, datname, client_addr, application_name, state, query_start, wait_event_type, query. Подсчитывается количество подключений по состояниям.

Этап 6. Длительные запросы и транзакции.
Находятся активные запросы, выполняющиеся дольше заданного времени, и сеансы в состоянии idle in transaction с давно открытой транзакцией.

Этап 7. Блокировки в PostgreSQL.
В двух сеансах моделируется блокировка: первый изменяет строку без фиксации транзакции, второй пытается изменить ту же строку. С помощью pg_blocking_pids() определяется блокирующий сеанс, который завершается функцией pg_terminate_backend(). Для

предотвращения подобных ситуаций настраивается параметр idle_in_transaction_session_timeout.

Этап 8. Оформление.
Оформляется отчёт, содержащий настройки Performance Schema, выполненные запросы и их результаты, найденные проблемы и рекомендации.

Пример решения:
Задание: Проанализировать причины замедления базы данных интернет-магазина в MySQL и зависания операций в PostgreSQL.
Дано: Пользователи жалуются на медленный поиск заказов в MySQL. В PostgreSQL периодически зависает обновление остатков на складе.

Решение по этапам:
1. Проверяем SHOW VARIABLES LIKE 'performance_schema'; — ON. Включаем: UPDATE performance_schema.setup_consumers SET ENABLED = 'YES' WHERE NAME = 'events_statements_history_long';
2. Выполняем SELECT query, exec_count, total_latency, rows_examined_avg FROM sys.statement_analysis LIMIT 5; — первое место занимает запрос поиска заказов по номеру телефона клиента: 4 200 вызовов, в среднем просматривается 150 000 строк.
3. Представление sys.statements_with_full_table_scans подтверждает полное сканирование таблицы orders. Создаём индекс по столбцу phone, повторный анализ показывает снижение просматриваемых строк до единиц.
4. Запрос к sys.schema_unused_indexes показывает 3 неиспользуемых индекса в таблице products, которые предлагаются к удалению после согласования.
5. В PostgreSQL выполняем SELECT state, count(*) FROM pg_stat_activity WHERE datname = 'warehouse' GROUP BY state; — 42 сеанса idle, 5 active, 3 idle in transaction.
6. Находим давние транзакции: SELECT pid, usename, application_name, now() - xact_start AS duration, query FROM pg_stat_activity WHERE state = 'idle in transaction' ORDER BY duration DESC; — сеанс отчётного приложения держит открытую транзакцию 47 минут.
7. Находим заблокированные сеансы: SELECT pid, pg_blocking_pids(pid), wait_event_type, query FROM pg_stat_activity WHERE cardinality(pg_blocking_pids(pid)) > 0; — обновления остатков ждут сеанс отчётного приложения.
8. После согласования завершаем блокирующий сеанс: SELECT pg_terminate_backend(12345); — обновления остатков выполнены.
9. Устанавливаем ALTER SYSTEM SET idle_in_transaction_session_timeout = '10min'; SELECT pg_reload_conf(); и передаём разработчикам отчётного приложения рекомендацию закрывать транзакции.
10. Оформляем отчёт.

Ответ: С помощью Performance Schema и схемы sys найден ресурсоёмкий запрос с полным сканированием таблицы заказов, после создания индекса проблема устранена, выявлены неиспользуемые индексы. С помощью pg_stat_activity и pg_blocking_pids найдена причина

зависания обновлений — незавершённая транзакция отчётного приложения, блокировка устранена и настроено автоматическое завершение простаивающих транзакций.

Критерии оценки практических заданий:

1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Практическое занятие №32. Настройка алертинга (уведомлений) в PostgreSQL на основе событийных триггеров.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных
ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации
ПК 2.5. Подготавливать данные для базы знаний
ПК 2.7. Осуществлять автоматизацию баз данных

Цель: Научиться создавать событийные триггеры PostgreSQL для отслеживания изменений структуры базы данных и удаления объектов, формировать журнал событий и отправлять оповещения администратору с помощью механизма LISTEN/NOTIFY.

Задание:
1. Определить события, о которых необходимо оповещать администратора.
2. Создать таблицу для хранения событий.
3. Создать функцию событийного триггера для фиксации DDL-команд.
4. Создать функцию событийного триггера для фиксации удаления объектов.
5. Создать событийные триггеры и отправку уведомлений через pg_notify.
6. Разработать программу-получатель уведомлений и отправку сообщений администратору.
7. Настроить запрет опасных команд в рабочей базе и протестировать оповещения.
8. Сделать вывод о работе системы оповещений.

Этапы выполнения:
Этап 1. Определение событий.
Определяется перечень контролируемых событий: создание, изменение и удаление таблиц, индексов, функций, изменение прав. Определяются получатели и канал оповещений.

Этап 2. Таблица событий.
Создаётся таблица с полями: время события, пользователь, адрес клиента, тег команды, тип и имя объекта, текст команды. Права на изменение таблицы предоставляются только администратору.

Этап 3. Функция для DDL-команд.
Создаётся функция, возвращающая тип event_trigger. В ней в цикле по результатам функции pg_event_trigger_ddl_commands() в таблицу записываются тег команды, тип и идентификатор объекта, имя пользователя (session_user), адрес клиента (inet_client_addr()) и текст запроса (current_query()).

Этап 4. Функция для удаления объектов.
Создаётся функция, использующая pg_event_trigger_dropped_objects() для получения сведений об удалённых объектах, так как после удаления объект уже недоступен.

Этап 5. Создание триггеров и уведомлений.
Командами CREATE EVENT TRIGGER ... ON ddl_command_end и ON sql_drop создаются событийные триггеры. В функции добавляется вызов pg_notify('dba_alerts', сообщение), передающий краткое описание события в канал. Создание событийных триггеров требует прав суперпользователя.

Этап 6. Программа-получатель.
На Python с библиотекой psycopg2 разрабатывается программа, которая выполняет LISTEN dba_alerts, ожидает уведомления и при их получении отправляет сообщение администратору по электронной почте или в мессенджер и записывает его в журнал. Программа запускается как служба.

Этап 7. Запрет и тестирование.
Создаётся событийный триггер ON ddl_command_start, который в рабочее время запрещает удаление таблиц командой RAISE EXCEPTION. Выполняются создание, изменение и

удаление объектов, проверяются записи в таблице событий, получение оповещений и срабатывание запрета.

Этап 8. Оформление.
Оформляется отчёт, содержащий код функций и триггеров, код программы-получателя, результаты тестирования и выводы.

Пример решения:
Задание: Настроить оповещение администратора об изменениях структуры рабочей базы данных бухгалтерии.
Дано: База billing используется в рабочее время с 09:00 до 18:00. Ранее неконтролируемое изменение структуры таблицы разработчиком привело к сбою приложения. Администратор получает оповещения в Telegram.

Решение по этапам:
1. Определяем контролируемые события: все DDL-команды и удаление объектов.
2. Создаём схема dba и таблицу dba.ddl_events (id, event_time timestamptz default now(), user_name, client_addr inet, command_tag, object_type, object_identity, query).
3. Создаём функцию dba.log_ddl() RETURNS event_trigger, которая для каждой записи из pg_event_trigger_ddl_commands() вставляет строку в dba.ddl_events и выполняет pg_notify('dba_alerts', format('%s: %s %s пользователем %s', now(), r.command_tag, r.object_identity, session_user)).
4. Создаём функцию dba.log_drop(), которая аналогично обрабатывает строки pg_event_trigger_dropped_objects() и отправляет уведомление об удалении.
5. Создаём триггеры: CREATE EVENT TRIGGER trg_log_ddl ON ddl_command_end EXECUTE FUNCTION dba.log_ddl(); CREATE EVENT TRIGGER trg_log_drop ON sql_drop EXECUTE FUNCTION dba.log_drop();
6. Создаём функцию dba.block_drop() и триггер ON ddl_command_start WHEN TAG IN ('DROP TABLE'), который при времени с 09:00 до 18:00 выполняет RAISE EXCEPTION 'Удаление таблиц в рабочее время запрещено'.
7. Разрабатываем программу alerts_listener.py: подключение к базе, выполнение LISTEN dba_alerts, ожидание уведомлений через select() и conn.poll(), отправка текста уведомления в Telegram через API бота, запись в журнал. Настраиваем запуск как службы systemd.
8. Тестируем: ALTER TABLE payments ADD COLUMN comment text; — в dba.ddl_events появилась запись, в Telegram пришло сообщение «ALTER TABLE public.payments пользователем dev_ivanov».
9. Выполняем DROP TABLE test_tmp; в 11:00 — команда отклонена с сообщением о запрете. В 19:00 удаление выполнено, получено оповещение об удалении.
10. Оформляем отчёт.

Ответ: С помощью событийных триггеров ddl_command_end и sql_drop настроена фиксация всех изменений структуры базы billing с указанием пользователя, адреса и текста команды. Механизм pg_notify и программа-получатель обеспечивают мгновенную отправку

оповещений администратору в Telegram, а событийный триггер ddl_command_start запрещает удаление таблиц в рабочее время. Тестирование подтвердило работу всех механизмов.

Критерии оценки практических заданий:

1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Практическое занятие №33. Создание и настройка собственного формата логов в PostgreSQL.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных
ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации
ПК 2.5. Подготавливать данные для базы знаний
ПК 2.7. Осуществлять автоматизацию баз данных

Цель: Научиться настраивать журналирование PostgreSQL: формат строки журнала, вывод в форматах CSV и JSON, ротацию файлов, состав регистрируемых событий, а также анализировать журналы средствами SQL и утилиты pgBadger.

Задание:
1. Определить требования к содержанию и формату журналов.
2. Настроить сборщик журналов, каталог, имена файлов и ротацию.
3. Разработать собственный формат строки журнала с помощью log_line_prefix.
4. Настроить состав регистрируемых событий.
5. Включить журналирование в форматах CSV и JSON.
6. Сгенерировать события и проверить содержимое журналов.
7. Загрузить журнал в таблицу и проанализировать события с помощью SQL и pgBadger.
8. Сделать вывод о настроенном формате журналов.

Этапы выполнения:
Этап 1. Требования.
Определяется, какие сведения должны содержаться в журнале: время с миллисекундами, процесс, пользователь, база данных, адрес клиента, приложение, код ошибки, идентификатор сеанса. Определяются срок хранения и способ анализа журналов.

Этап 2. Сборщик и ротация.
Включается logging_collector, задаются log_directory, log_filename с шаблоном даты (например, postgresql-%Y-%m-%d.log), log_rotation_age, log_rotation_size и log_truncate_on_rotation. Изменение logging_collector требует перезапуска сервера.

Этап 3. Формат строки.
В параметре log_line_prefix задаётся собственный формат с помощью последовательностей %m, %p, %c, %l, %u, %d, %r, %a, %e и разделителей, удобных для чтения и автоматической обработки.

Этап 4. Состав событий.
Настраиваются log_connections, log_disconnections, log_min_duration_statement, log_statement, log_lock_waits, log_temp_files, log_checkpoints, log_autovacuum_min_duration, log_min_messages и log_min_error_statement.

Этап 5. Форматы CSV и JSON.
В параметре log_destination дополнительно указываются csvlog и jsonlog (jsonlog доступен с PostgreSQL 15). Эти форматы содержат все поля независимо от log_line_prefix и удобны для загрузки в системы анализа.

Этап 6. Генерация и проверка событий.
Выполняются подключения, медленный запрос, запрос с ошибкой, блокировка, создание временного файла. Проверяется, что события записаны во все журналы в нужном формате.

Этап 7. Анализ.
Журнал CSV загружается в таблицу, структура которой описана в документации PostgreSQL, командой COPY и анализируется SQL-запросами. С помощью утилиты pgBadger по журналу формируется HTML-отчёт со статистикой запросов, ошибок, подключений и блокировок.

Этап 8. Оформление.
Оформляется отчёт, содержащий параметры журналирования, примеры строк журнала, результаты анализа и выводы.

Пример решения:

Задание: Настроить журналы PostgreSQL сервера интернет-магазина для расследования инцидентов и анализа производительности.
Дано: Журналы ведутся в стандартном формате без указания пользователя и адреса клиента, что затрудняло расследование недавнего инцидента. Требуется хранение журналов 7 дней и запись запросов дольше 300 мс.

Решение по этапам:
1. Определяем обязательные поля: время, процесс, сеанс, пользователь, база, адрес клиента, приложение, код ошибки.
2. Выполняем: ALTER SYSTEM SET logging_collector = on; ALTER SYSTEM SET log_directory = 'log'; ALTER SYSTEM SET log_filename = 'postgresql-%a.log'; ALTER SYSTEM SET log_rotation_age = '1d'; ALTER SYSTEM SET log_rotation_size = 0; ALTER SYSTEM SET log_truncate_on_rotation = on; — шаблон %a (день недели) обеспечивает хранение 7 файлов с перезаписью.
3. Задаём формат: ALTER SYSTEM SET log_line_prefix = '%m [%p] сеанс=%c польз=%u база=%d клиент=%r прил=%a код=%e ';
4. Задаём события: log_connections = on, log_disconnections = on, log_min_duration_statement = 300, log_statement = 'ddl', log_lock_waits = on, log_temp_files = 0, log_checkpoints = on.
5. Задаём ALTER SYSTEM SET log_destination = 'stderr,csvlog,jsonlog'; и перезапускаем сервер.
6. Выполняем подключение под пользователем shop_app, запрос SELECT pg_sleep(0.5);, запрос к несуществующей таблице и изменение структуры таблицы.
7. Проверяем журнал: строка «2026-10-08 15:12:03.481 MSK [24811] сеанс=6705a1c3.60eb польз=shop_app база=shop клиент=10.0.0.15(51544) прил=shop-api код=42P01 ERROR: relation "orderz" does not exist». Аналогичные события присутствуют в файлах .csv и .json.
8. Создаём таблицу postgres_log по структуре из документации и загружаем журнал: COPY postgres_log FROM '/var/lib/postgresql/16/main/log/postgresql-Thu.csv' WITH csv; Выполняем SELECT user_name, error_severity, count(*) FROM postgres_log GROUP BY 1, 2; — видно распределение ошибок по пользователям.
9. Формируем отчёт pgBadger: pgbadger -f csv log/postgresql-Thu.csv -o report.html — в отчёте отображаются самые медленные запросы, количество подключений по часам и ошибки.
10. Оформляем отчёт.

Ответ: Настроено журналирование PostgreSQL с собственным форматом строки, содержащим время с миллисекундами, процесс, сеанс, пользователя, базу, адрес клиента, приложение и код ошибки, ротацией с хранением 7 дней и записью подключений, медленных запросов, изменений структуры и блокировок. Журналы дополнительно ведутся в форматах CSV и JSON, загружаются в таблицу для анализа SQL-запросами и обрабатываются pgBadger.

Критерии оценки практических заданий:
1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Практическое занятие №34. Протоколирование событий доступа к данным в Microsoft SQL Server и анализ логов.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке

Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.1. Выявлять проблемы, возникающие в процессе эксплуатации баз данных
ПК 2.2. Осуществлять процедуры администрирования баз данных
ПК 2.3. Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации
ПК 2.5. Подготавливать данные для базы знаний
ПК 2.7. Осуществлять автоматизацию баз данных

Цель: Научиться настраивать механизм SQL Server Audit для протоколирования входов в систему, изменений прав и доступа к конфиденциальным данным, а также читать и анализировать журналы аудита средствами SSMS и T-SQL.

Задание:
1. Определить события доступа, подлежащие протоколированию.
2. Создать объект аудита сервера с записью в файл.
3. Создать спецификацию аудита сервера для событий входа и изменения прав.
4. Создать спецификацию аудита базы данных для доступа к конфиденциальной таблице.
5. Сгенерировать события от имени разных пользователей.
6. Прочитать журнал аудита через SSMS и функцией sys.fn_get_audit_file.
7. Проанализировать журнал: выявить неудачные входы и подозрительный доступ к данным.
8. Сделать вывод и сформулировать рекомендации.

Этапы выполнения:
Этап 1. Определение событий.
Определяется перечень протоколируемых событий: успешные и неудачные входы, изменения логинов, ролей и прав, чтение и изменение таблиц с персональными и финансовыми данными.

Этап 2. Объект аудита сервера.
Командой CREATE SERVER AUDIT создаётся аудит с записью в файл (TO FILE) с указанием каталога, максимального размера и количества файлов, а также действия при невозможности записи (ON_FAILURE). Аудит включается командой ALTER SERVER AUDIT ... WITH (STATE = ON).

Этап 3. Спецификация аудита сервера.
Командой CREATE SERVER AUDIT SPECIFICATION добавляются группы действий FAILED_LOGIN_GROUP, SUCCESSFUL_LOGIN_GROUP, SERVER_ROLE_MEMBER_CHANGE_GROUP, SERVER_PRINCIPAL_CHANGE_GROUP, и спецификация включается.

Этап 4. Спецификация аудита базы данных.
Командой CREATE DATABASE AUDIT SPECIFICATION в нужной базе добавляются действия SELECT, INSERT, UPDATE, DELETE на конфиденциальную таблицу для роли public (то есть для всех пользователей), а также группа DATABASE_ROLE_MEMBER_CHANGE_GROUP.

Этап 5. Генерация событий.
Выполняются вход с неверным паролем, успешные входы, чтение и изменение конфиденциальной таблицы разными пользователями, добавление пользователя в роль.

Этап 6. Чтение журнала.
В SSMS журнал открывается через контекстное меню объекта аудита «Просмотреть журналы аудита». Средствами T-SQL журнал читается запросом к sys.fn_get_audit_file с выбором полей

event_time, action_id, succeeded, server_principal_name, database_name, object_name, statement, client_ip, application_name.

Этап 7. Анализ.
Запросами к журналу подсчитываются неудачные входы по логинам и адресам, определяется, кто и когда обращался к конфиденциальной таблице, выявляется доступ вне рабочего времени и из необычных приложений, проверяются изменения прав.

Этап 8. Оформление.
Оформляется отчёт, содержащий скрипты создания аудита и спецификаций, выдержки из журнала, результаты анализа и рекомендации.

Пример решения:
Задание: Настроить протоколирование доступа к данным о зарплатах в базе отдела кадров и проанализировать журнал за день.
Дано: База HR содержит таблицу dbo.salaries. Доступ к ней имеют сотрудники бухгалтерии через приложение «Зарплата». Служба безопасности требует фиксировать все обращения к таблице и попытки подбора паролей.

Решение по этапам:
1. Создаём аудит: CREATE SERVER AUDIT HR_Audit TO FILE (FILEPATH = 'D:\Audit\', MAXSIZE = 100 MB, MAX_ROLLOVER_FILES = 20) WITH (ON_FAILURE = CONTINUE); ALTER SERVER AUDIT HR_Audit WITH (STATE = ON);
2. Создаём спецификацию сервера: CREATE SERVER AUDIT SPECIFICATION HR_Server_Spec FOR SERVER AUDIT HR_Audit ADD (FAILED_LOGIN_GROUP), ADD (SUCCESSFUL_LOGIN_GROUP), ADD (SERVER_ROLE_MEMBER_CHANGE_GROUP) WITH (STATE = ON);
3. В базе HR создаём спецификацию: CREATE DATABASE AUDIT SPECIFICATION HR_DB_Spec FOR SERVER AUDIT HR_Audit ADD (SELECT, INSERT, UPDATE, DELETE ON dbo.salaries BY public), ADD (DATABASE_ROLE_MEMBER_CHANGE_GROUP) WITH (STATE = ON);
4. Генерируем события: 5 неудачных входов логина admin_test с одного адреса, чтение таблицы бухгалтером через приложение «Зарплата», чтение таблицы логином dev_petrov через SSMS в 20:15, добавление dev_petrov в роль db_datareader.
5. Открываем журнал в SSMS через «Просмотреть журналы аудита» — события отображаются с временем, действием и пользователем.
6. Анализируем неудачные входы: SELECT server_principal_name, client_ip, count(*) AS attempts FROM sys.fn_get_audit_file('D:\Audit*.sqlaudit', DEFAULT, DEFAULT) WHERE action_id = 'LGIF' GROUP BY server_principal_name, client_ip; — 5 попыток входа admin_test с адреса 10.0.5.77.
7. Анализируем доступ к зарплатам: SELECT event_time, server_principal_name, action_id, application_name, statement FROM sys.fn_get_audit_file('D:\Audit*.sqlaudit', DEFAULT, DEFAULT) WHERE object_name = 'salaries' ORDER BY event_time; — выявлено чтение таблицы логином dev_petrov из SSMS в нерабочее время.
8. По событию DATABASE_ROLE_MEMBER_CHANGE_GROUP устанавливаем, что dev_petrov был добавлен в роль db_datareader администратором в 19:50 без заявки.
9. Формируем рекомендации: исключить dev_petrov из роли и запретить доступ к таблице salaries командой DENY, заблокировать адрес 10.0.5.77 до выяснения, настроить ежедневную выгрузку и проверку журнала аудита, ограничить права на изменение аудита.
10. Оформляем отчёт для службы безопасности.

Ответ: Настроен механизм SQL Server Audit с записью в файл: спецификация сервера протоколирует входы и изменения ролей, спецификация базы — все обращения к таблице зарплат. Анализ журнала средствами SSMS и функции sys.fn_get_audit_file выявил попытки подбора пароля, несанкционированный доступ разработчика к данным о зарплатах в

нерабочее время и выдачу ему прав без заявки. Сформулированы меры по устранению нарушений.

Критерии оценки практических заданий:
1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

3.2. Задания для оценки освоения МДК 02.02

1. Оценка освоения теоретического курса профессионального модуля

1.1. Задания для оценки освоения МДК 02.02 Технология разработки и защиты баз данных

Тема 1. Основы хранения и обработки данных. Проектирование БД.

Проверяемые результаты обучения:

- ОК 01 Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
- ОК 02 Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной деятельности
- ОК 05 Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
- ОК 09 Пользоваться профессиональной документацией на государственном и иностранном языках
- ПК 2.1 Выявлять проблемы, возникающие в процессе эксплуатации баз данных.
- ПК 2.2 Осуществлять процедуры администрирования баз данных.
- ПК 2.3 Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации.
- ПК 2.4 Формировать требования хранилищ банка данных для обучения.
- ПК 2.5 Подготавливать данные для базы знаний.

Критерии оценки устного опроса

- оценка **«отлично»** выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов и смог ответить на уточняющие вопросы;
- оценка **«хорошо»** выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов, но не смог ответить на уточняющие вопросы;

- оценка **«удовлетворительно»** выставляется студенту, если он смог изложить суть основного вопроса, но не смог ответить по существу дополнительных вопросов;
- оценка **«неудовлетворительно»** выставляется студенту, если он не смог изложить суть основного вопроса

Вопросы к устному опросу по теме:

1. Дайте определение отношению (таблица).

Ответ: Отношение в базе данных представляет собой структуру данных, состоящую из строк и столбцов, где каждая строка представляет отдельную запись, а каждый столбец - атрибут или поле.

2. Дайте определение атрибуту.

Ответ: Атрибут - это характеристика или свойство объекта, который хранится в базе данных. В контексте таблицы это было бы столбцом, представляющим одну характеристику записи.

3. Дайте определение типу данных.

Ответ: Тип данных определяет, какого рода данные могут быть хранены в определенном столбце таблицы, например, целые числа, строки, даты, и т. д.

4. Дайте определение домену.

Ответ: Домен - это набор допустимых значений для конкретного атрибута в таблице базы данных. Он определяет спецификацию значений, которые могут быть использованы для данного атрибута.

5. Дайте определение базе данных

Ответ: База данных - это организованная коллекция данных, обычно хранящаяся и обрабатываемая с помощью базы данных, чтобы обеспечить доступ, обновление и управление данными.

6. Дайте определение таблице.

Ответ: Таблица в базе данных представляет собой структурированное представление данных, организованных в виде строк (записей) и столбцов (атрибутов).

7. Дайте определение ключам.

Ответ: Ключи в базе данных используются для уникальной идентификации записей в таблице. Примерами ключей являются первичный ключ, внешний ключ и уникальный ключ.

8. Дайте определение связям.

Ответ: Связи в базе данных определяют основные отношения между различными таблицами, обеспечивая связь между данными в них.

9. Дайте определение индексам.

Ответ: Индексы в базе данных используются для оптимизации процесса

поиска и сортировки данных, позволяя обеспечивать быстрый доступ к данным в таблицах. Они ускоряют выполнение запросов к базе данных.

10. Приведите отличия реляционных баз данных от не реляционные базы данных.

Ответ:

Реляционные базы данных (SQL базы данных):

1. Тип данных: Реляционные базы данных хранят данные в таблицах, состоящих из строк и столбцов, где каждая строка представляет отдельную запись, а каждый столбец - атрибут или поле данных.

2. Язык запросов: Для доступа и обработки данных в реляционных базах данных используется структурированный язык запросов SQL (Structured Query Language).

3. Нормализация: Реляционные базы данных используют принципы нормализации данных для уменьшения дублирования информации и обеспечения целостности данных.

4. Атомарность транзакций: Реляционные базы данных поддерживают атомарность транзакций, что означает, что либо все операции в транзакции выполняются успешно, либо ни одна из них не происходит.

Не реляционные базы данных (NoSQL базы данных):

1. Структура данных: Не реляционные базы данных могут хранить данные в различных форматах, таких как документы, графы, столбцы и ключ-значение.

2. Гибкость: NoSQL базы данных обладают большей гибкостью и способностью обработки неструктурированных данных, что делает их подходящими для больших объемов данных и изменяющихся требований.

3. Горизонтальное масштабирование: NoSQL базы данных обеспечивают легкое горизонтальное масштабирование для обработки больших объемов данных и высоких нагрузок.

4. Отсутствие языка запросов SQL: В отличие от реляционных баз данных, NoSQL базы данных могут использовать различные языки запросов или API для доступа к данным.

5. Подход к связям: NoSQL базы данных могут иметь другие подходы к управлению связями между данными, так как они не всегда используют жесткую таблицу для хранения информации.

11. Дайте определение объектно-ориентированным базам данных.

Ответ: Объектно-ориентированные базы данных (SQL):

Объектно-ориентированные базы данных - это базы данных, которые используют объектно-ориентированный подход для хранения и обработки данных. В таких базах данных данные хранятся в виде объектов, которые могут иметь свои собственные методы и свойства. Запросы к объектно-ориентированным базам данных обычно выполняются с использованием объектно-ориентированных языков программирования, таких как Java или C++.

12. Дайте определение NoSQL базы данных.

Ответ: NoSQL базы данных: NoSQL базы данных - это группа баз данных,

которые не используют классическую реляционную модель данных (SQL), а предлагают альтернативные подходы к хранению и обработке данных. NoSQL базы данных могут быть ориентированы на хранение документов, графов, столбцов или ключ-значение. Они обладают гибкостью и масштабируемостью, что делает их особенно полезными для работы с большими объемами данных и изменяющимися требованиями.

13. Дайте определение классам объектов.

Ответ: Классы объектов - это шаблоны или модели, определяющие структуру и поведение объектов в объектно-ориентированном программировании. Классы определяют атрибуты (поля данных) и методы (функции), которые могут быть применены к объектам этого класса. В контексте объектно-ориентированных баз данных, классы объектов определяют структуру данных, которая будет храниться в базе данных и с которой можно будет работать при выполнении запросов и операций.

13. Вопрос: Какие основные этапы проектирования базы данных выделяют?
    Ответ: Основные этапы: концептуальное, логическое и физическое проектирование.
14. Вопрос: В чём суть концептуального проектирования?
    Ответ: На этом этапе создаётся концептуальная модель данных, отражающая бизнес-требования и сущности предметной области без привязки к конкретной СУБД.
15. Вопрос: Что происходит на этапе логического проектирования?

Ответ: Концептуальная модель преобразуется в логическую схему, обычно в терминах реляционной модели: определяются таблицы, ключи, связи, но без учёта особенностей хранения.

16. Вопрос: Чем отличается физическое проектирование?

Ответ: На этом этапе логическая схема адаптируется под конкретную СУБД: выбираются типы данных, индексы, структура хранения, оптимизируются запросы.

17. Вопрос: Что такое нормализация базы данных?

Ответ: Это процесс организации данных для устранения избыточности и предотвращения аномалий при обновлении, добавлении и удалении данных.

18. Вопрос: В чём суть первой нормальной формы (1НФ)?

Ответ: Таблица находится в 1НФ, если все её атрибуты атомарны (не содержат повторяющихся групп или вложенных таблиц).

19. Вопрос: Что такое вторая нормальная форма (2НФ)?

Ответ: Таблица в 2НФ, если она в 1НФ и каждый неключевой атрибут полностью зависит от всего первичного ключа (нет частичных зависимостей).

20. Вопрос: В чём заключается третья нормальная форма (3НФ)?

Ответ: Таблица в 3НФ, если она в 2НФ и в ней отсутствуют транзитивные зависимости неключевых атрибутов от первичного ключа.

21. Вопрос: Что такое нормальная форма Бойса-Кодда (НФБК)?

Ответ: Это более строгая форма: таблица в НФБК, если каждый детерминант является потенциальным ключом. Устраняет все аномалии, связанные с функциональными зависимостями.

22. Вопрос: Что такое логическая независимость данных?

Ответ: Это возможность изменять логическую схему базы данных (например,

добавлять новые атрибуты или таблицы) без необходимости изменять приложения, работающие с БД.

23. Вопрос: В чём суть физической независимости данных?

Ответ: Это возможность изменять физическую структуру хранения данных (например, способы индексации, файловую организацию) без изменения логической схемы и приложений.

24. Вопрос: Что такое представление (VIEW) в реляционной базе данных?

А) Физическая копия таблицы, хранящаяся на диске

Б) Виртуальная таблица, результат выполнения SQL-запроса, не хранящая данные физически ✅

В) Тип индекса для ускорения поиска

Г) Резервная копия базы данных

25. Вопрос: Какая структура данных в СУБД предназначена для ускорения поиска записей по определённым полям?

А) Триггер

Б) Представление

В) Индекс ✅

Г) Хранимая процедура

26. Вопрос: Что такое кластерный индекс в СУБД?

А) Индекс, который физически упорядочивает строки таблицы в соответствии со значениями ключа ✅

Б) Индекс, созданный сразу по нескольким таблицам

В) Индекс, использующий хеш-функцию для поиска

Г) Временный индекс, удаляемый после перезагрузки сервера

27. Вопрос: Какое основное преимущество использования представлений (VIEW)?

А) Увеличение скорости вставки данных

Б) Упрощение сложных запросов и ограничение доступа к данным ✅

В) Физическое ускорение работы СУБД

Г) Автоматическое резервное копирование данных

Вопрос: Какова основная цель нормализации базы данных?

А) Увеличение объёма хранимых данных

Б) Устранение избыточности данных и аномалий при их обновлении ✅

В) Ускорение работы сети

Г) Шифрование информации в базе данных

28. Вопрос: Какое условие должно выполняться для приведения таблицы к Первой нормальной форме (1НФ)?

А) Все атрибуты должны быть атомарными (неделимыми) ✅

Б) В таблице должен быть только один ключ

В) Все поля должны быть числовыми

Г) Таблица должна содержать не более 10 столбцов

29. Вопрос: Что характеризует Третья нормальная форма (3НФ)?

А) Отсутствие повторяющихся групп атрибутов

Б) Все неключевые атрибуты полностью зависят от первичного ключа и не зависят друг от друга ✅

В) Наличие составного первичного ключа

Г) Отсутствие в таблице внешних ключей

30. Вопрос: Какая аномалия возникает при удалении записи из ненормализованной таблицы, когда вместе с данными удаляется и информация, которую нужно было сохранить?

А) Аномалия обновления

Б) Аномалия вставки

В) Аномалия удаления ✅

Г) Аномалия транзакции

31. Вопрос: Какой язык используется для описания структуры базы данных (создания таблиц, индексов, ограничений)?

А) DML (Data Manipulation Language)

Б) DDL (Data Definition Language) ✅

В) DCL (Data Control Language)

Г) TCL (Transaction Control Language)

32. Вопрос: Что такое ER-диаграмма (Entity-Relationship)?

А) Схема алгоритма работы программы

Б) Графическая модель «сущность-связь» для проектирования структуры БД ✅

В) Диаграмма последовательности выполнения запросов

Г) Таблица с результатами тестирования производительности

33. Вопрос: Какой метод описания схемы БД предполагает использование графических нотаций (UML, IDEF1X, ER)?

А) Логический метод

Б) Визуальный (графический) метод ✅

В) Текстовый метод

Г) Программный метод

34. Вопрос: Что такое первичный ключ (PRIMARY KEY) таблицы?

А) Поле, которое может содержать значения NULL

Б) Поле или набор полей, однозначно идентифицирующих каждую запись в таблице ✅

В) Поле, используемое только для сортировки

Г) Внешнее поле, ссылающееся на другую таблицу

35. Вопрос: Какое ограничение обеспечивает ссылочную целостность между таблицами?

А) UNIQUE

Б) CHECK

В) FOREIGN KEY ✅

Г) NOT NULL

36. Вопрос: Что обеспечивает ограничение CHECK в СУБД?

А) Уникальность значений в столбце

Б) Проверку значений по заданному логическому условию ✅

В) Ссылку на другую таблицу

Г) Автоматическое удаление записей

37. Вопрос: Какая модель данных основана на таблицах, связанных между собой через ключи?

А) Иерархическая модель

Б) Сетевая модель

В) Реляционная модель ✅
Г) Объектная модель

Практические занятия к теме 1.

Практическое занятие №1. Создание концептуальной модели базы данных с использованием диаграммы "сущность-связь" (ER-диаграмма).

Цели работы:

1. Выполнить анализ предметной области.
2. Выполнить построение концептуальной модели базы данных с использованием Case-средств.

Проверяемые результаты обучения:

- ОК 01 Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
- ОК 02 Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной деятельности
- ОК 05 Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
- ОК 09 Пользоваться профессиональной документацией на государственном и иностранном языках
- ПК 2.1 Выявлять проблемы, возникающие в процессе эксплуатации баз данных.
- ПК 2.4 Формировать требования хранилищ банка данных для обучения.

Ход работы:

1. Проектирование концептуальной модели базы данных «Магазин»

У вас небольшая консалтинговая компания, специализирующаяся на разработке баз данных. Вы получили контракт на разработку модели данных для системы управления базой данных небольшого розничного магазина под названием Oracle Baseball League (OBL).

Магазин Oracle Baseball League обслуживает все местное население и продает экипировку для бейсбола. У магазина OBL два типа покупателей. Одна категория — это индивидуальные покупатели, приобретающие такие товары,

как мячи, бутсы, перчатки, футболки, футболки с принтом и шорты. Другая категория — представители спортивных команд, приобретающие форму и экипировку от лица команды.

Команды и индивидуальные покупатели могут приобрести любой товар, имеющийся на складе, но команды получают скидку от цены по прейскуранту в зависимости от числа игроков. Когда покупатель размещает заказ, в базе данных регистрируются позиции этого заказа.

В штате OBL работают три продавца-консультанта, которые официально обслуживают только команды, но иногда работают и с претензиями индивидуальных клиентов.

Определение объектов

Проанализируйте текст данного сценария и определите потенциальные объекты, которые должны быть представлены в реляционной базе данных. В описании сценария объекты, как правило, являются существительными, но не каждое существительное может стать объектом. Будьте внимательны и помните, что вам необходимо определить потенциальные объекты, а не создать точный список.

Решение:

Магазин Oracle Baseball League обслуживает все местное население и продает экипировку для бейсбола. У магазина OBL два типа клиентов (customer). Одна категория — это индивидуальные покупатели, приобретающие такие товары (item), как мячи, бутсы, перчатки, футболки, футболки с принтом и шорты. Другая категория — представители спортивных команд (team), приобретающие форму и экипировку для команды.

Команды и индивидуальные покупатели могут приобрести любой товар, имеющийся на складе (inventory list), но команды получают скидку от цены по прейскуранту в зависимости от числа игроков. Когда покупатель размещает заказ (order), в базе данных регистрируются позиции этого заказа.

В штате OBL работают три продавца-консультанта (sales representative), которые официально обслуживают только команды, но иногда работают и с претензиями индивидуальных клиентов.

Потенциальные объекты:

- Покупатель
- Команда
- Продавец
- Заказ
- Товар
- Товары на складе

Связи

- Связи должны иметь имена, которые помогают описать связь между объектами.
- На диаграмме имя связи в любом направлении указано рядом с начальной точкой линии связи (см. слайд 5).
- Примеры:
  - DEPARTMENT **содержит** EMPLOYEE.
  - EMPLOYEE **назначен для** DEPARTMENT.

Copyright © 2017, Корпорация Oracle и аффилированные с ней лица.
Все права защищены.

Компоненты связи

- Компонентами связи являются:
  - **Имя:** метка, которая появляется рядом с объектом, для которого она назначена. Имена связи должны быть указаны в нижнем регистре.
  - **Кардинальность:** минимальное и максимальное число значений в связи.
    - Одна и только одна соответствующая запись
    - Одна или более соответствующих записей
  - **Обязательность:** должна ли существовать связь.
    - Необязательная (ноль соответствующих записей)
    - Обязательная (хотя бы одна соответствующая запись в каждом объекте)

Copyright © 2017, Корпорация Oracle и аффилированные с ней лица. Все права защищены.

Внешний ключ

- Связи в концептуальной модели данных сопоставляются с внешними ключами в таблице физической базы данных.
- Внешний ключ (FK) — это столбец или комбинация столбцов в одной таблице, которая ссылается на первичный ключ в той же таблице или в другой таблице.

Copyright © 2017, Корпорация Oracle и аффилированные с ней лица. Все права защищены.

Компоненты связи

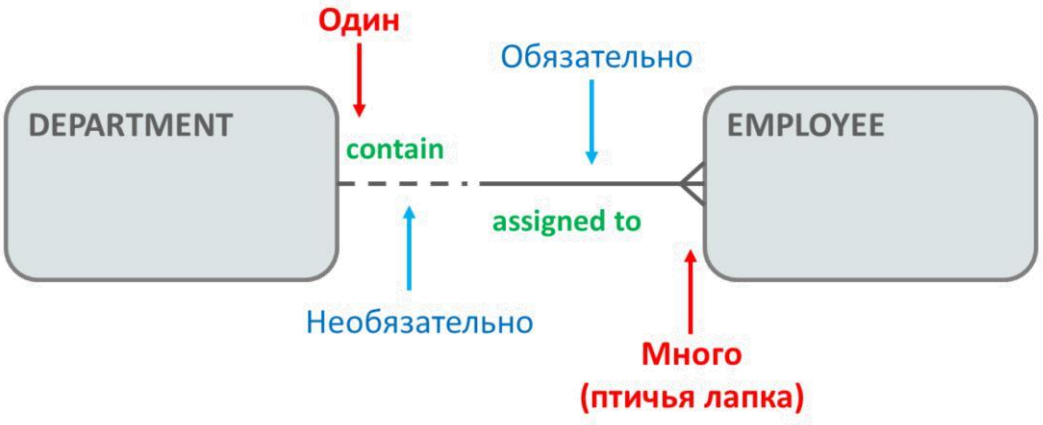
- При чтении бизнес-правил используйте следующие фразы:
  - **Необязательно:** используйте «может быть» или «может».
  - **Обязательно:** используйте «должно быть» или «должно».
  - **Линия:** используйте «один и только один».
  - **Птичья лапка:** используйте «один или более».
- Синтаксис бизнес-правил:

  Каждый объект entity1 {должен быть или может быть} имя связи {«один или более» или «один и только один»} объект entity2.

Copyright © 2017, Корпорация Oracle и аффилированные с ней лица. Все права защищены.

Компоненты связи

- Имя
- Кардинальность
- Обязательность



Copyright © 2017, Корпорация Oracle и аффилированные с ней лица. Все права защищены.

Типы связей

- Все связи представляют собой информационные требования и правила ведения бизнеса.
  - «Многие к одному» (M:1) или «один ко многим» (1:M)
  - «Многие ко многим» (M:M)
  - «Один к одному» (1:1)

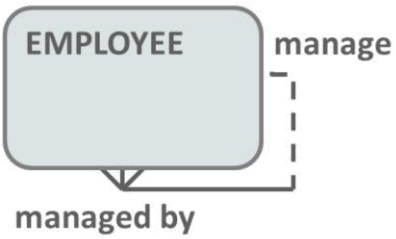


Пример связи 1:M

Copyright © 2017, Корпорация Oracle и аффилированные с ней лица.
Все права защищены.

Рекурсивные связи

- Рекурсивные связи — это связи объекта с самим собой



- Бизнес-правила:
  - Каждый сотрудник (EMPLOYEE) может руководить одним или более сотрудниками (EMPLOYEE).
  - Каждым сотрудником (EMPLOYEE) должен руководить один и только один сотрудник (EMPLOYEE)

Copyright © 2017, Корпорация Oracle и аффилированные с ней лица.
Все права защищены.

Рекурсивность связи

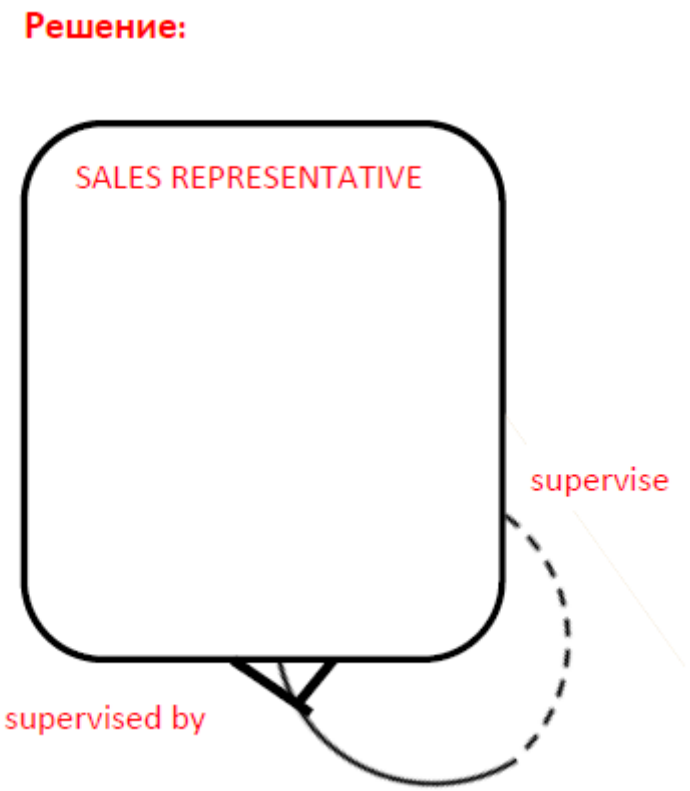
Рекурсивная связь позволяет представить связь, которую объект имеет сам с собой.

Для рекурсивной связи требуется, чтобы один атрибут в объекте зависел от другого атрибута в этом объекте. Убедитесь в наличии обоих атрибутов в объекте, перед тем как начертить рекурсивную связь.

Описание решения:

Один из продавцов-консультантов руководит распределением команд для других, поэтому здесь используется рекурсивная связь.

Решение:



Построить матрицу связей.

Матрица связей показывает, состоит ли объект строки, расположенный слева, в каких-либо отношениях с объектом столбца в верхней части матрицы.

- Если объект строки связан с объектом столбца, имя этой связи отображается в области пересечения.
- Если объект строки не связан с объектом столбца, область пересечения пуста.
- Чтобы представить рекурсивную связь, разместите имя этой связи по диагонали.

Часть 1.

Создайте матрицу связей, чтобы проверить связи, найденные за все время работы с проектом.

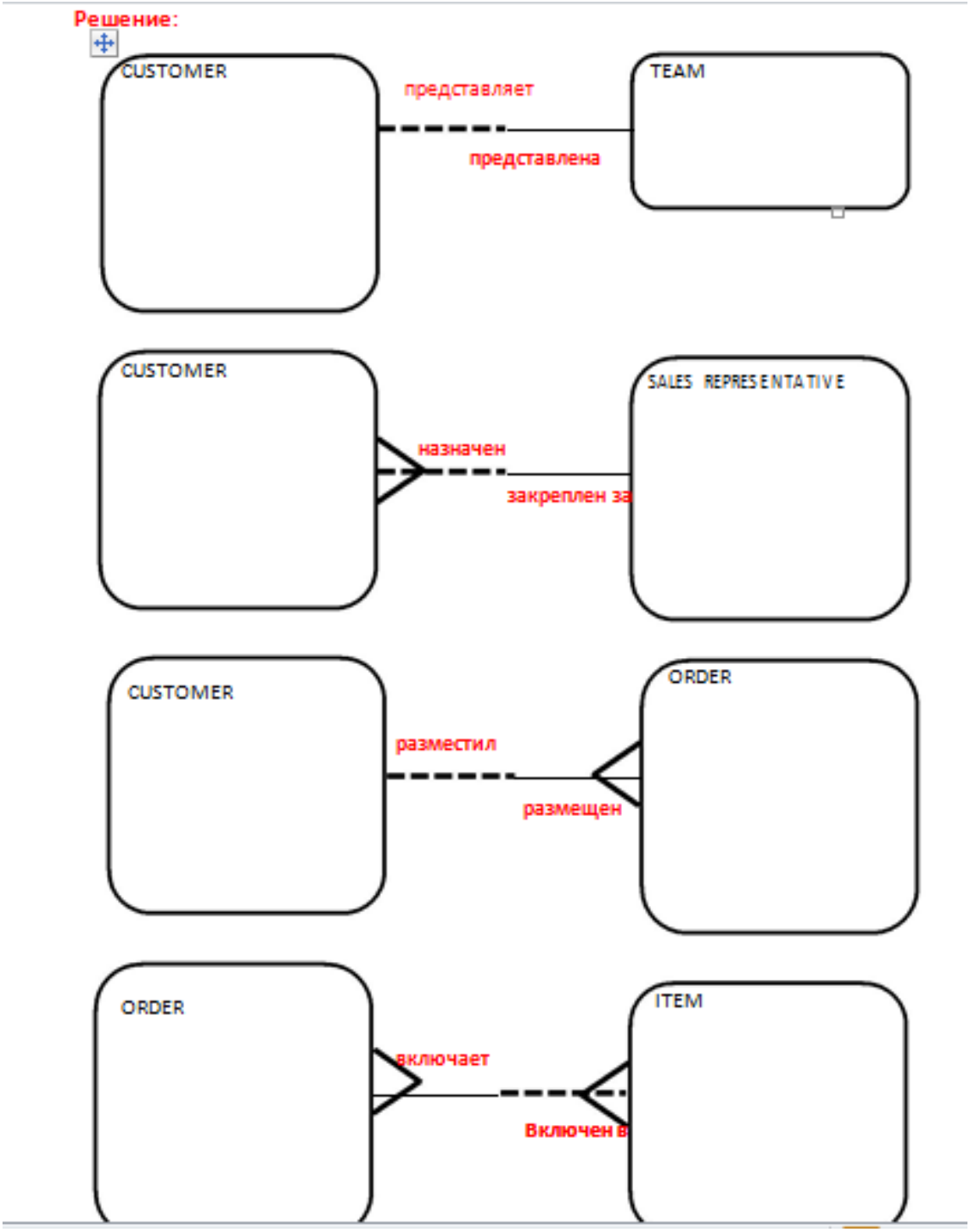
Используйте описания объектов, чтобы создать имена связей в области пересечения.

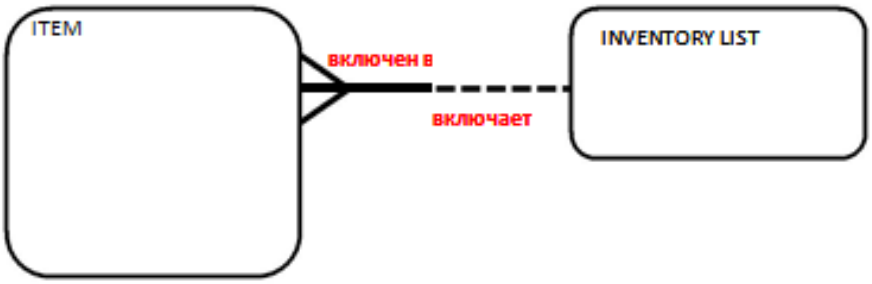
		customer	item	team	order	sales representative	inventory list
2	customer			представляет	размещает	назначен	
3	item				входит		расположено
4	inventory list		содержит				
5	order	размещён	включает				
6	sales representative	закреплён				Главный продавец руководит другими	
7	team	представляет					
8							
9							
10							
11							

Часть 2.

Используя имена, добавленные в области пересечения в матрице связей,

добавьте правильные метки к соответствующим связям на ERD-диаграмме.

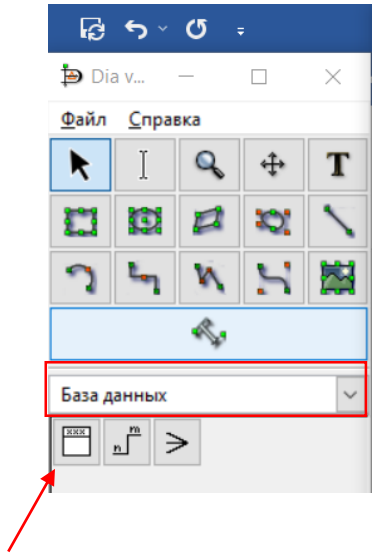


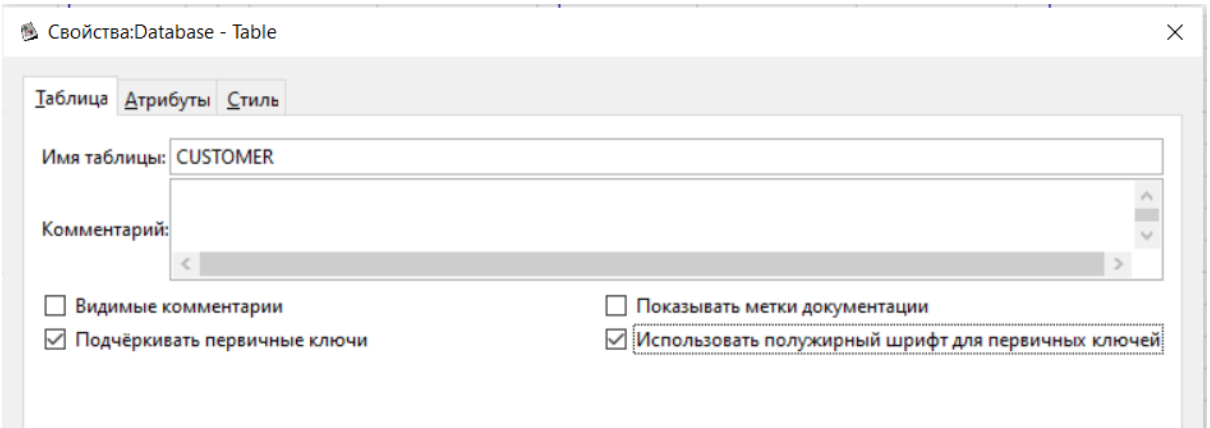


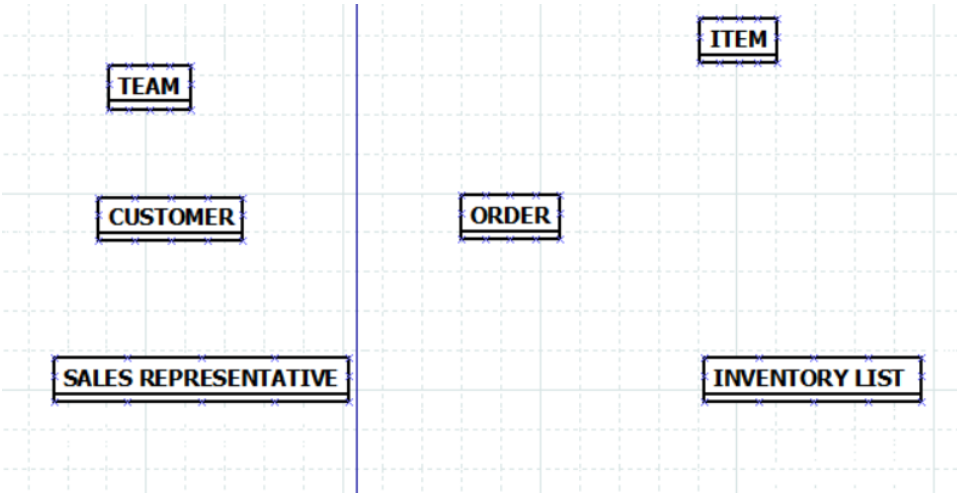
2. Построение концептуальной модели базы данных с использованием Case-средств.

2.1. Запустить программу dia.

2.2. Создайте объекты: CUSTOMER, TEAM, SALES REPRESENTATIVE, ORDER, ITEM, INVENTORY LIST используя элемент «Таблица»

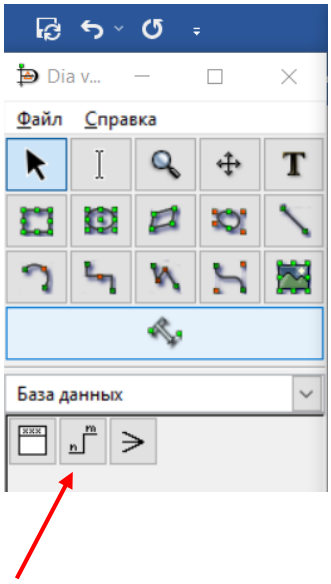




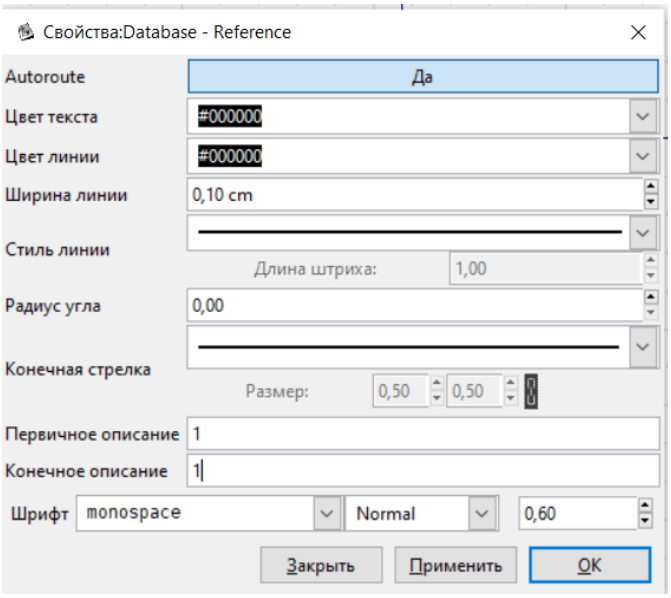


2.3. Создайте связь между объектами. Можно создать связь M:N (многие ко многим), связь 1:N (один ко многим), определяющую связь 1:N (один ко многим, ассоциативная) или связь 1:1 (один к одному).

Для связи сущностей используйте элемент «Ссылка».



Выберите элемент и протяните от одной сущности к другой. Двойной щелчок на линии связи откроет свойства связи, которые можно поменять.

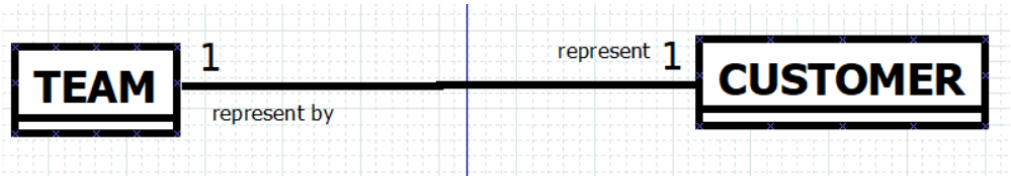


Результат:



2.4. Присвойте имя связи

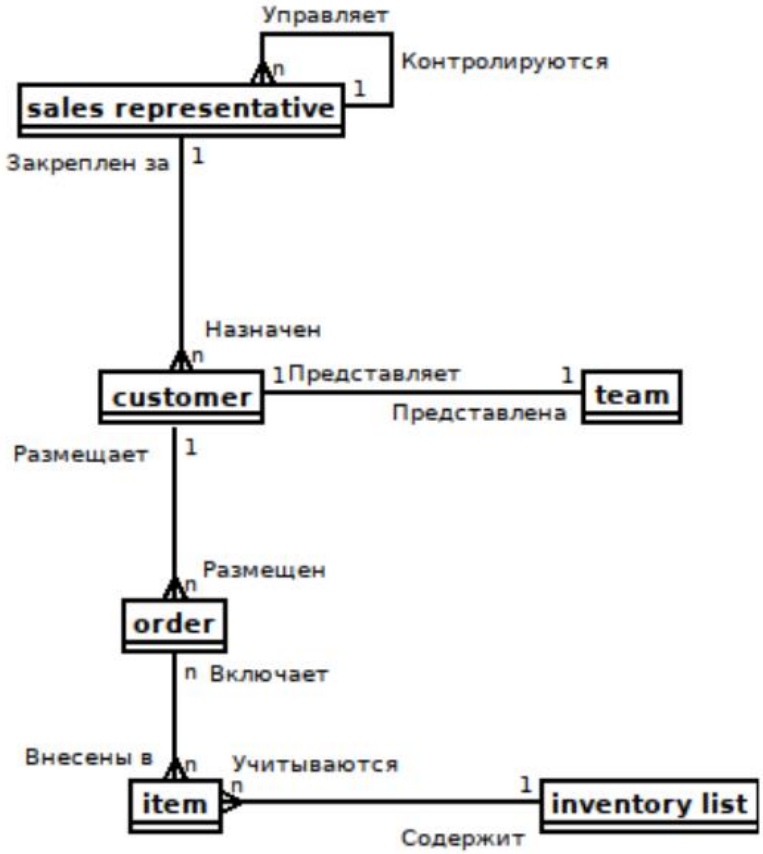
Результат:



2.5. Аналогично соедините связями остальные сущности.

2.6. Сохраните свою работу

Результат работы



Критерии оценки.

«5» «отлично»

- студент продемонстрировал готовность и способность к самостоятельному выполнению видов работ в стандартных ситуациях;

- все работы выполнил правильно и уверенно;

- проявил активность в освоении новых практических умений;

- содержание отчета по практическим работам полноценно отражает информацию по выполненным видам работ, установленных программой практики;

- оформление отчета по практическим работам полностью соответствует

предъявляемым требованиям;

- при выполнении практических работ студент уверенно продемонстрировал обладание необходимыми компонентами общих и профессиональных компетенций;

«4» «хорошо»

- студент продемонстрировал готовность и способность к самостоятельному выполнению видов работ в стандартных ситуациях;
- все работы выполнил правильно и уверенно;
- содержание отчета по практическим работам в целом правильно отражает информацию по выполненным видам работ;
- оформление отчета по практическим работам соответствует предъявляемым требованиям, однако имеются замечания (некоторая неаккуратность, неполное описание проделанной работы и освоенных умений);
- при выполнении практических работ студент продемонстрировал обладание компонентами общих и профессиональных компетенций;
- характеристики студента положительные, имеются отдельные замечания и рекомендации;

«3» «удовлетворительно»

- при выполнении видов работ студент нуждался во внешнем сопровождении и контроле;
- в отчете по практическим работам содержится описание видов выполняемых работ согласно программе дисциплины, однако, записи неполные и неточные;
- отчет по практическим работам оформлен с отклонениями от установленных требований;

- при выполнении практических работ студент продемонстрировал обладание отдельными компонентами общих и профессиональных компетенций;

«2» «неудовлетворительно»

– студент не справился с запланированными видами работ;

– при выполнении практических работ студент продемонстрировал слабое обладание отдельными компонентами общих и профессиональных компетенций;

– отчет студентом не подготовлен.

Практическое занятие №2. Разработка логической модели базы данных на основе ER-диаграммы. Нормализация базы данных.

Цели работы:

1. Выполнить разработку логической модели базы данных на основе ER-диаграммы.
2. Выполнить нормализацию базы данных.

Проверяемые результаты обучения:

- ОК 01 Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
- ОК 02 Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной деятельности
- ОК 05 Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
- ОК 09 Пользоваться профессиональной документацией на государственном и иностранном языках
- ПК 2.1 Выявлять проблемы, возникающие в процессе эксплуатации баз данных.
- ПК 2.4 Формировать требования хранилищ банка данных для обучения.

Ход работы:

Сценарий проекта:

У вас небольшая консалтинговая компания, специализирующаяся на разработке баз данных. Вы получили контракт на разработку модели данных для системы управления базой данных небольшого розничного магазина под названием Oracle Baseball League (OBL).

Магазин Oracle Baseball League обслуживает все местное население и продает экипировку для бейсбола. У магазина OBL два типа покупателей. Одна категория — это индивидуальные покупатели, приобретающие такие товары, как мячи, бутсы, перчатки, футболки, футболки с принтом и шорты. Другая категория — представители спортивных команд, приобретающие форму и экипировку от лица команды.

Команды и индивидуальные покупатели могут приобрести любой товар, имеющийся на складе, но команды получают скидку от цены по прейскуранту в зависимости от числа игроков. Когда покупатель размещает заказ, в базе данных регистрируются позиции этого заказа.

В штате OBL работают три продавца-консультанта, которые официально обслуживают только команды, но иногда работают и с претензиями индивидуальных клиентов.

Проектирование логической схемы

Шаг 1.

Определение атрибутов

С целью уточнить характеристики системы была проведена встреча с менеджером магазина Oracle Baseball League. Ниже приведена запись беседы.

Запись встречи

Интервьюер: Согласно предоставленной вами информации,у вас есть два типа клиентов: индивидуальные покупатели и команды. Какую

информацию о клиентах вы храните и как вы различаете эти типы клиентов?

Менеджер: Для всех клиентов в магазине отслеживаются: полное имя, адрес, номер телефона, адрес электронной почты, а также сведения о том, относятся ли они к какой-либо команде. Наша система также отслеживает текущий баланс каждого клиента

Интервьюер: Вы сказали, что покупатели могут заказать любой товар, имеющийся на складе. Какие виды товаров они могут купить?

Менеджер: Индивидуальные покупатели могут купить такие товары, как мячи, бутсы, перчатки, футболки, футболки с принтом и шорты. Кроме того, команды могут заказать весь набор униформ, а также мячи, костюмы для разминки и командные футболки, а также получить скидку относительно прейскуранта, которая зависит от количества игроков в команде. Когда команда покупает товары из магазина, мы требуем, чтобы клиент, зарегистрированный для этой команды, разместил для нее заказ.

Интервьюер: Нужно ли регистрировать в системе какие-то особые сведения о товарах?

Менеджер: Клиенты никогда не покупают товары вслепую, поэтому для товаров всегда доступны описание и цена. Отслеживание товарных запасов — важнейшая часть нашего бизнеса, поэтому помимо описаний и цен мы в настоящее время отслеживаем название, цвет (если применимо), размер (если применимо) и категорию товара. Наши товары относятся к трем категориям: одежда, экипировка и прочее. Кроме того, мы отслеживаем для наших запасов оптовую цену единицы товара и количество имеющихся в наличии единиц — при отсутствии единиц товара в систему записывается ноль.

Интервьюер: Как вы определяете,какие товары заказывали покупатели?

Менеджер: Когда клиент размещает заказ,мы записываем следующие сведения о покупке:дату, приобретаемые предметы, размер и цвет товара,а также количество единиц и цену длякаждой единицы товара. Еще мы хотели бы хранить общую стоимость заказа для всех заказанных товаров.

Интервьюер: В вашей компании есть три продавца-консультанта.Каковы их обязанности?

Менеджер: Каждомуклиенту,представляющему команду,назначается собственныйпродавец-консультант — продавцы работают за комиссионные, поэтому двум продавцам никогда не разрешатся звонить одному и тому же клиенту. Как правило, продавцы-консультанты работают с командами, но они также могут работать с претензиями индивидуальных покупателей.

Интервьюер: Как сведения о продавцах регистрируются в системе?

Менеджер: Для каждого из трех продавцов в магазине отслеживаются:имя,адрес,номертелефона, адрес электронной почты, общая сумма комиссионных и ставка комиссионного вознаграждения.

Проанализируйте текст данного сценария и определите потенциальные атрибуты, в которых будет храниться информация о ранее определенных объектах.

Решение:

Потенциальные объекты и атрибуты

Customer

- name
- address
- phone number
- email
- team they belong to – принадлежность к команде
- current balance

Team

- discount
- number of players
- name

Sales Representative

- name
- address

- phone number
- email
- total commission - общая комиссия
- commission rate - ставка комиссионного вознаграждения

Order

- date
- items purchased – приобретаемые товары
- item size
- color
- number of units – количество единиц
- price
- total price - общая сумма заказа

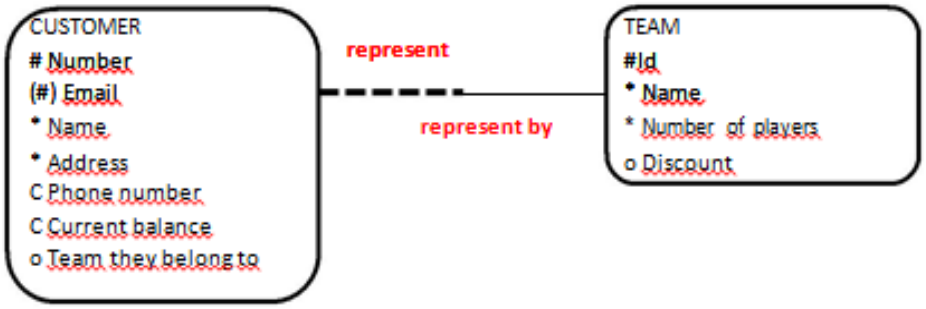
Item

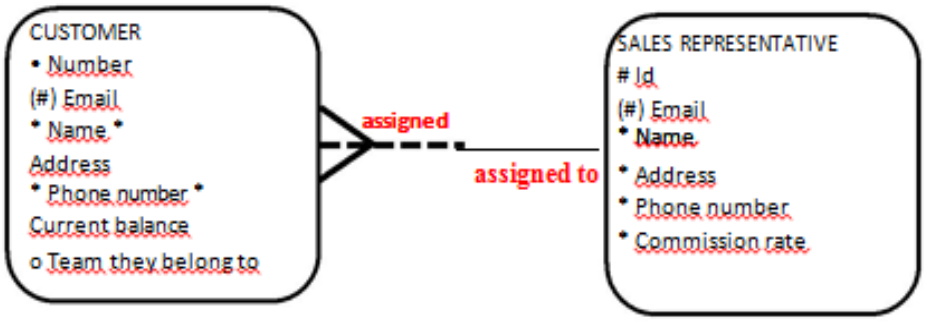
- description
- price
- name
- color
- razmer
- category

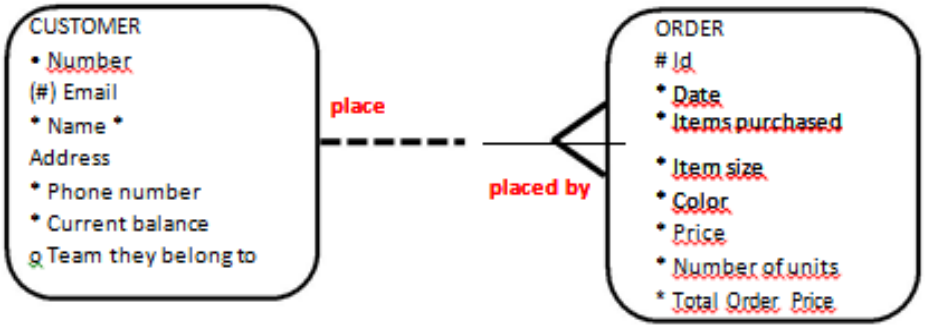
Inventory List

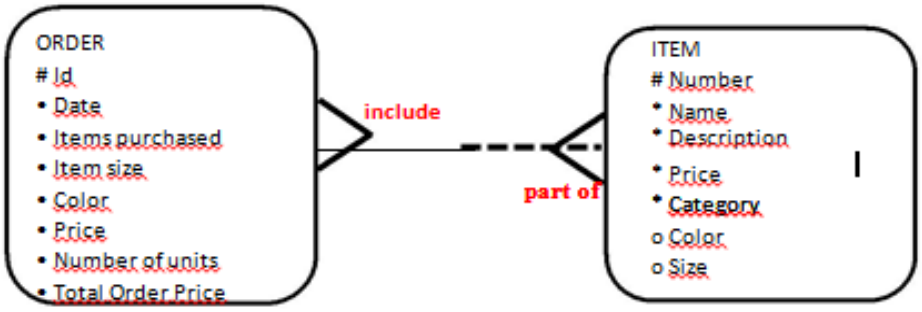
- cost of the unit – цена за единицу
- units on hand – количество в наличии

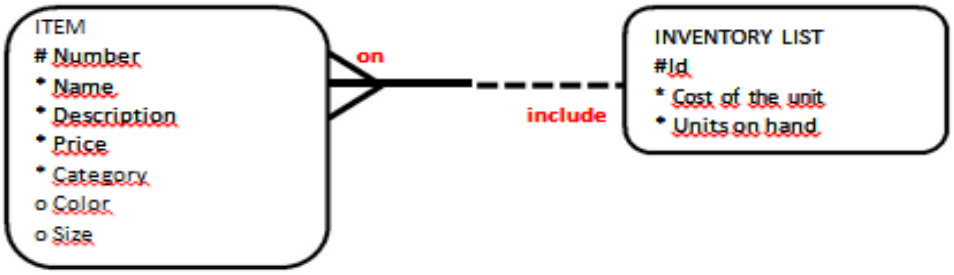
Решение:


CUSTOMER
Number
(#) Email
* Name
* Address
C Phone number
C Current balance
o Team they belong to
represent
represent by
TEAM
#Id
* Name
* Number of players
o Discount


CUSTOMER
• Number
(#) Email
* Name *
Address
* Phone number *
Current balance
o Team they belong to
assigned
assigned to
SALES REPRESENTATIVE
Id
(#) Email
* Name
* Address
* Phone number
* Commission rate


CUSTOMER
• Number
(#) Email
* Name *
Address
* Phone number
* Current balance
o Team they belong to
place
placed by
ORDER
Id
* Date
* Items purchased
* Item size
* Color
* Price
* Number of units
* Total Order Price


ORDER
Id
• Date
• Items purchased
• Item size
• Color
• Price
• Number of units
• Total Order Price
include
part of
ITEM
Number
* Name
* Description
* Price
* Category
o Color
o Size


ITEM
Number
* Name
* Description
* Price
* Category
o Color
o Size
on
include
INVENTORY LIST
#Id
* Cost of the unit
* Units on hand

Шаг 2. Моделирование связей между объектами

Диаграмма «объект-связь» (ERD) позволяет наглядно представить информацию о системе и позволяет достичь четырех целей:

• Собрать всю необходимую информацию.

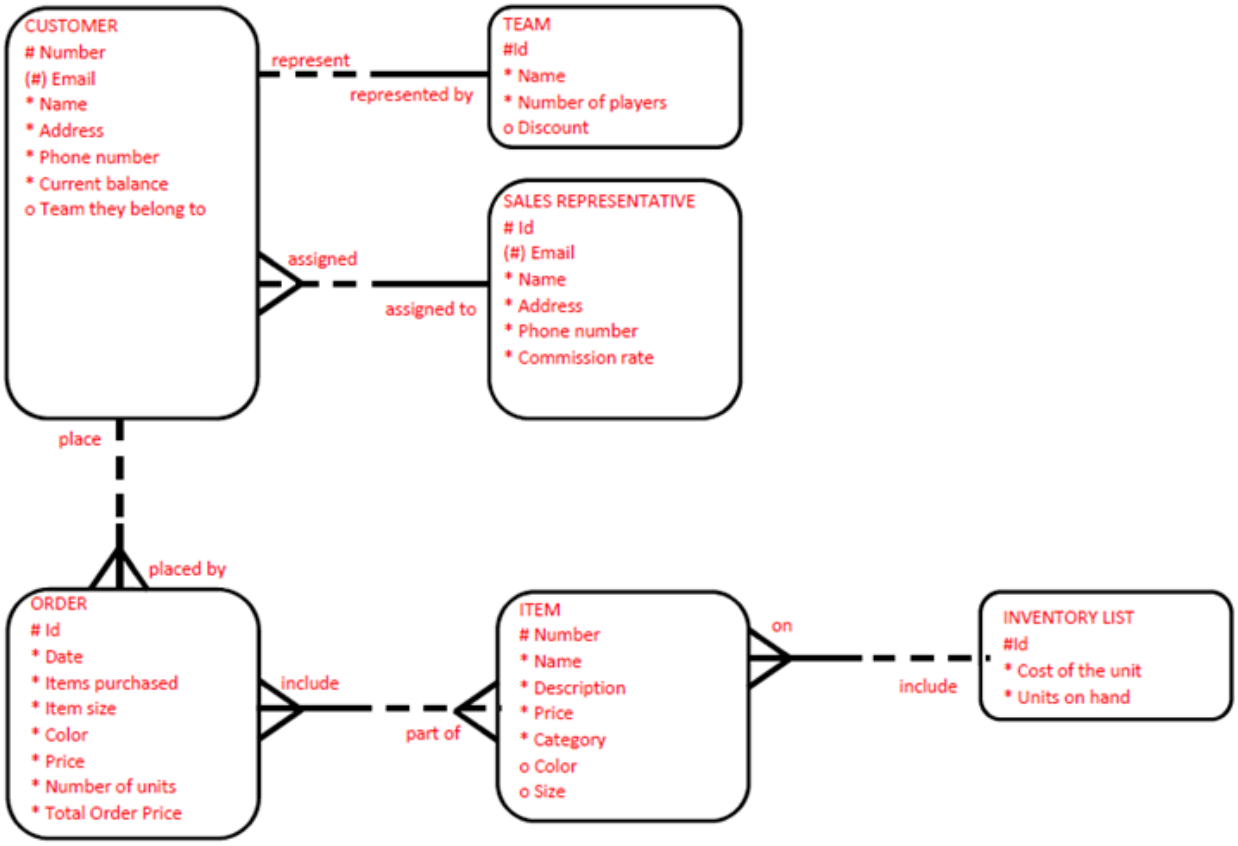
• Убедиться, что информация отображается только один раз.

• Не тратить время на моделирование информации, производной от других уже смоделированных данных.

• Представить информацию в предсказуемой логической форме.

Поскольку объекты, их атрибуты и связи между объектами уже определены, можно приступать к построению итоговой ERD-диаграммы, где будут отражены связи между компонентами системы.

Используя информацию, уже собранную в данном проекте, создайте диаграмму «объект-связь» (ERD), которая соответствует четырем вышеуказанным целям. Создайте свою диаграмму ERD, следуя правилам построения диаграмм.



Шаг 3. Разрешение связей «многие ко многим»

Когда была разработана начальная диаграмма «объект-связь» (ERD), ее показали клиенту, чтобы подтвердить, что его бизнес правильно отражен в этом проекте. Далее приведена расшифровка с того совещания:

Запись встречи

Интервьюер: Отмечаете ли вы какие-либо серьезные проблемы в проекте вашей новой системы?

Менеджер: Кажется, вы охватили все. Не значит ли это, что он почти закончен?

Интервьюер: Не совсем, нужно еще многое сделать, прежде чем это станет работающей системой баз данных. Хочу задать вам несколько вопросов, чтобы прояснить некоторые моменты до того, как мы закончим проект.

Менеджер: Нет проблем.

Интервьюер: Клиент может предоставить команду; имеет ли каждая команда одного назначенного представителя, или много сотрудников могут размещать

заказы от имени команды?

Менеджер: Только один человек может быть официальным представителем команды.

Интервьюер: Может ли один и тот же человек представлять несколько команд?

Менеджер: Нет, они могут назначаться только для разных команд.

Интервьюер: На одном из предыдущих совещаний вы сказали, что для каждого представителя команды назначен один продавец-консультант. Как назначаются продавцы-консультанты для команд?

Менеджер: Один из продавцов-консультантов назначается на роль руководителя и несет ответственность за назначение отдельных команд для всех продавцов-консультантов, включая самого себя.

Интервьюер: Вы также сказали, что продавцы-консультанты иногда рассматривают жалобы отдельных клиентов?

Менеджер: Да, если у клиента возникла проблема с заказом, продавец-консультант помогает ее устранить.

Интервьюер: Когда клиент размещает заказ, может ли этот заказ быть перемещен кому-то другому?

Менеджер: Нет, это заказ между компанией и лицом, разместившим заказ; это нельзя изменить.

Интервьюер: Клиент может зарегистрировать только один адрес как место назначения для доставки или может иметь несколько адресов?

Менеджер: Мы разрешаем нашим клиентам зарегистрировать несколько адресов, чтобы доставка заказов осуществлялась в место, которое лучше всего подходит клиентам. Адрес может быть выделен только для одного клиента.

Интервьюер: Если вы храните несколько адресов для клиентов, храните ли вы также несколько адресов для продавцов-консультантов?

Менеджер: Нет, мы храним только текущие адреса наших сотрудников, если они переезжают, мы обновляем информацию. Нет необходимости

отслеживать историю их адресов.

Интервьюер: Возвращаясь к клиентам, может ли клиент быть как представителем команды, так и отдельным клиентом?

Менеджер: Безусловно, но у него должны быть две отдельные учетные записи, и покупки с одной из них не влияют на покупки с другой.

Интервьюер: Команды получают скидку с учетом количества игроков, как это работает?

Менеджер: В команде должно быть не менее 9 игроков для получения скидки. Если в команде как минимум девять игроков, она получает скидку в размере 5%, если в ней как минимум 25 игроков, она получает скидку в размере 10%, а для команд из 40 игроков и более скидка составляет 20%. Команде предоставляется соответствующая скидка.

Интервьюер: Собираетесь ли вы внедрить что-то, что может повлиять на систему сейчас или в будущем?

Менеджер: Мы рассматриваем возможность внедрения схемы клубных карточек для отдельных клиентов. Это будет необязательная схема, которая позволит владельцам клубных карточек участвовать в специальных распродажах, где они могут покупать товары по сниженной цене.

Интервьюер: Хотели бы вы записывать, что каждый владелец клубной карточки покупает на этих распродажах.

Менеджер: Нет, единственным ограничением будет допуск в магазин только клиентов, у которых есть клубные карточки. Нам не нужно знать, какие товары они покупают.

Интервьюер: Итак, вы хотели бы, чтобы мы отслеживали цены товаров с течением времени, чтобы вы могли их изменить?

Менеджер: Да, это облегчит нам жизнь.

В реляционных базах данных можно иметь три следующих типа связей, которые соответствуют требованиям к информации и правилам бизнеса:

• «Многие к одному» (M:1) или «один ко многим» (1:M)

• «Многие ко многим» (M:M)

• «Один к одному» (1:1)

Если в вашей ERD-диаграмме имеются связи «многие ко многим» (M:M), вам потребуется разрешить их, прежде чем двигаться дальше. Следует помнить, что для разрешения связи M:M необходимо выполнить следующее:

• Удалите связь M:M.

• Создайте новый объект (пересечение).

• Создайте две определяющие связи со стороной «многие» на новом объекте пересечения.

• При необходимости определите, и создайте дополнительные атрибуты в объекте пересечения. Подумайте о том, как будет осуществляться хранение, каково количество для каждого товара и сколько товаров было фактически доставлено клиенту.

• Оцените, не приводят ли две существующие определяющие связи к назначению UID для объекта пересечения, или не следует ли создать искусственный UID для нового объекта.

• Добавьте метки связи для новых определяющих связей.

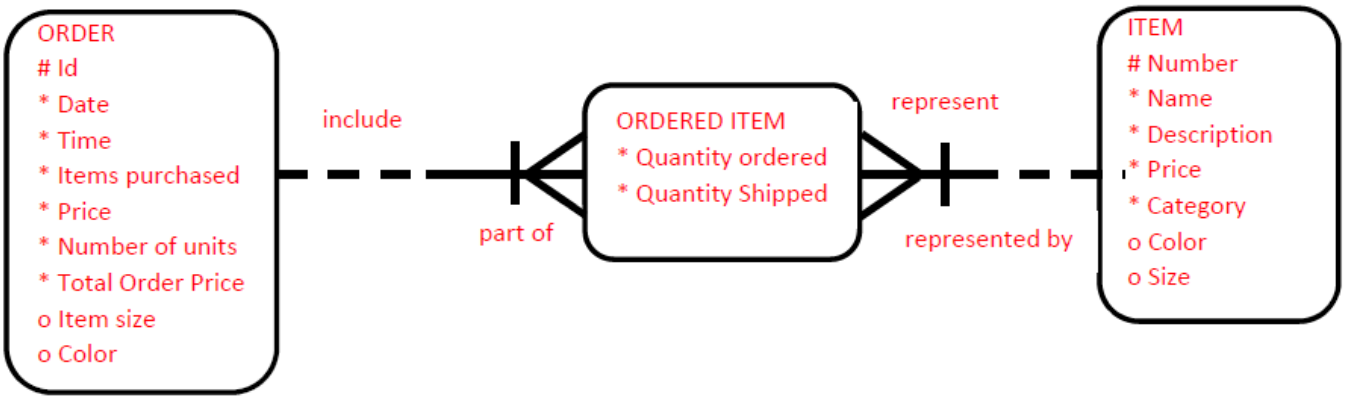
Описание решения.

Связь M:M между объектами **ORDER** и **ITEM** необходимо решить с помощью объекта пересечения.

Это объект пересечения с именем **ORDERED_ITEM**, в котором хранится информация о количестве заказанных товаров и о том, сколько из них было отправлено. Это дает нам возможность доставлять товары разными партиями.

Идентификатор UID для **ORDERED_ITEM** использует **ассоциативную связь** (barred) для применения UID из обоих объектов ORDER и ITEM в качестве UID.

Решение:



Quantity ordered – заказанное количество

Quantity Shipped – отгруженное количество

Шаг 5. Отслеживание изменений данных

Отслеживание изменений данных с течением времени.

Информация, существующая в одном объекте, может хранить только одно (или текущее) значение. Если мы изменим это значение, вся историческая информация будет потеряна. Чтобы хранить текущие и исторические или альтернативные данные, нам, возможно, придется добавить в модель объекты и связи для размещения этой дополнительной информации.

Вот ряд вопросов, которые можно задать, чтобы определить необходимость исторических данных:

• Требуется ли журнал аудита?

• Могут ли со временем изменяться значения атрибутов?

• Могут ли со временем изменяться связи?

• Нужно ли запрашивать более старые данные?

Используя данный сценарий, добавьте дополнительные объекты и соответствующие связи, которые необходимы для хранения каких-либо данных, которые могут со временем изменяться, или при необходимости альтернативных данных.

Описание решения:

В данном сценарии передана следующая информация:

Интервьюер: *Клиент может зарегистрировать только один адрес как место назначения для доставки или может иметь несколько адресов?*

Менеджер: *Мы разрешаем нашим клиентам зарегистрировать несколько адресов, чтобы доставка заказов осуществлялась в место, которое лучше всего подходит клиентам. Адреса каждого клиента должны быть уникальными.*

Интервьюер: *Если вы храните несколько адресов для клиентов, храните ли вы также несколько адресов для продавцов-консультантов?*

Менеджер: *Нет, мы храним только текущие адреса наших сотрудников, если они переезжают, мы обновляем информацию. Нет необходимости отслеживать историю их адресов.*

Это говорит о том, что нам следует извлечь адрес из объекта CUSTOMER и создать связь 1:M, позволяющую клиенту получать товары на несколько адресов.

Менеджер также утверждает, что, хотя поле адреса хранится для представителя отдела продаж (SLAES REPRESENTATIVE), на этом этапе нет бизнес-требования делать что-либо с этим атрибутом.

Кроме того, сценарий дает следующую информацию.

Интервьюер: *Собираетесь ли вы внедрить что-то, что может повлиять на систему сейчас или в будущем?*

Менеджер: *Мы рассматриваем возможность внедрения схемы клубных карточек для отдельных клиентов. Это будет необязательная схема, которая позволит владельцам клубных карточек участвовать в специальных распродажах, где они могут покупать товары по сниженной цене.*

Интервьюер: *Хотели бы вы записывать, что каждый владелец клубной карточки покупает на этих распродажах.*

Менеджер: *Нет, единственным ограничением будет допуск в магазин только клиентов, у которых есть клубные карточки. Нам не нужно знать, какие товары они покупают.*

Интервьюер: *Итак, вы хотели бы, чтобы мы отслеживали цены товаров с течением времени, чтобы вы могли их изменить?*

Менеджер: *Да, это облегчит нам жизнь.*

Так как цены товаров могут меняться, нам нужно удалить цену из товара (ITEM) и поместить ее в собственный объект под названием «История цены» (PRICE HISTORY).

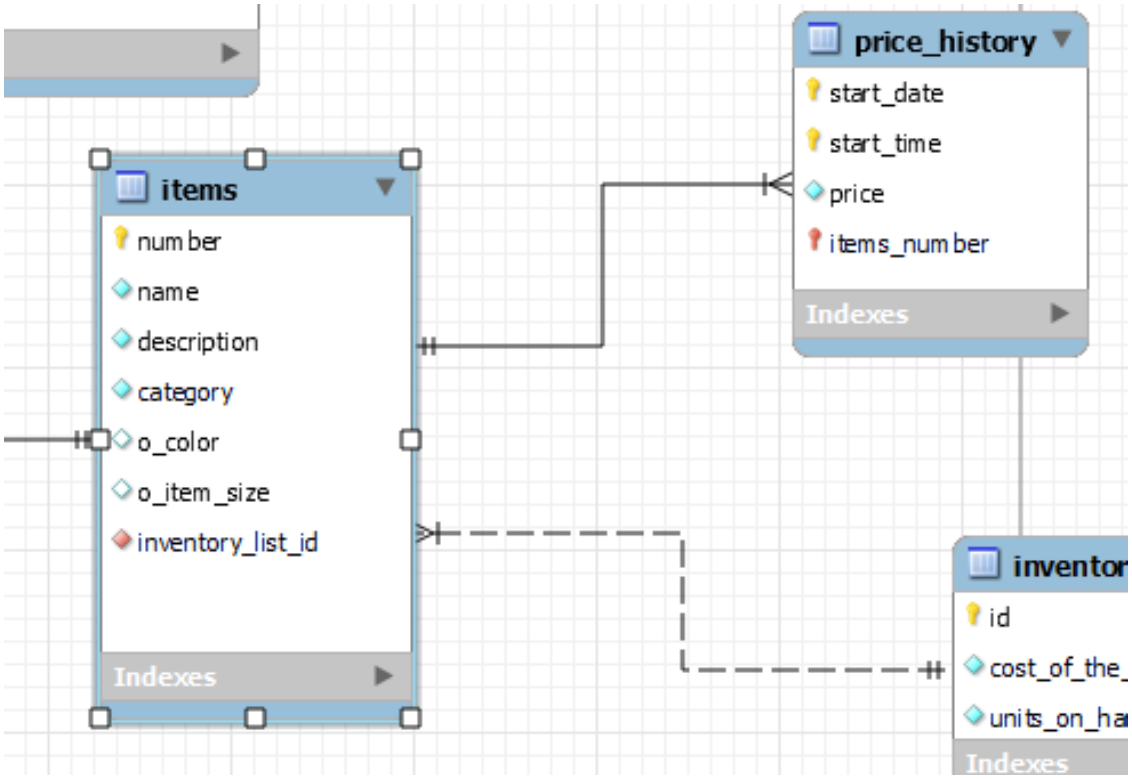
Объект PRICE HISTORY записывает дату и время начала изменения цены.

Атрибуты окончания необязательны, так как они должны хранить текущую цену, которая еще не закончилась.

PRICE HISTORY имеет составной UID, состоящий из даты начала и времени начала, что в сочетании с ассоциативной связью с ITEM создает уникальное значение.

С помощью ассоциативной связи для извлечения UID из ITEM создается связь между объектами ITEM и PRICE HISTORY.

Требуется отслеживать изменение цен на товары. Для того была создана новая сущность с именем price_history, которая содержит поля для хранения даты времени и товара, на который менялась цена. Поле price было удалено на сущности items.



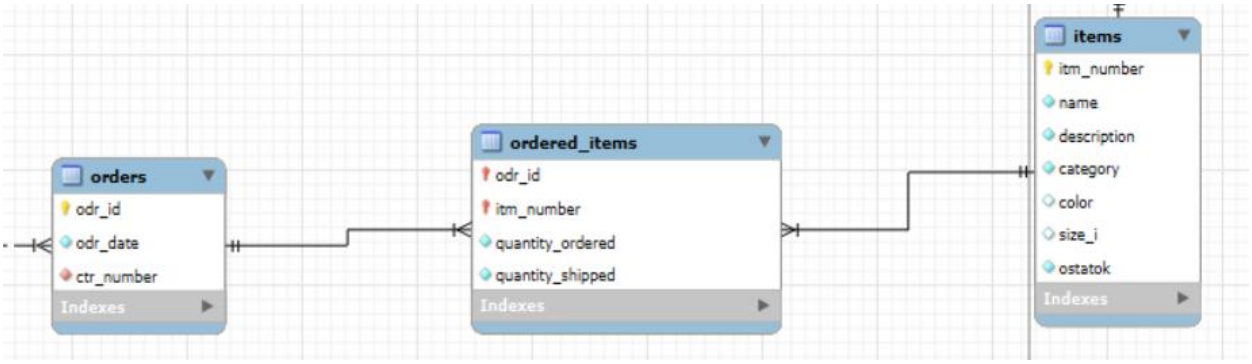
2.Выполнить построение логической модели базы данных с использованием Case-средств.

a. Создайте копию предыдущей работы и назовите файл «Логическая модель».
b. Откройте созданный файл. В нем будет продолжена работа.
c. Создайте атрибуты на каждом объекте и создайте UID для каждого объекта.

✓ Дважды щелкнуть по сущности CUSTOMER. Внесите необходимые атрибуты с указанием первичных, внешних ключей, ограничениями уникальности, признаками обязательности и необязательности.

✓ Установите необходимые связи.

***Связь** является **идентифицирующей** тогда и только тогда, когда первичный ключ дочерней сущности содержит внешний ключ, идущий от родительской сущности. Если такой вещи нет - **связь** будет неидентифицирующей.*



Упражнение 1. Нормализация и бизнес-правила

Проверка данных путем нормализации.
Преобразование данных в ненормализованную форму.

Перед началом нормализации необходимо преобразовать данные в ненормализованную форму (UNF).

- Ненормализованная форма (UNF)
  - Удалите все расчетные поля, которые могут быть получены из других атрибутов.
  - Проверьте, что каждый объект имеет уникальный идентификатор.
  - Удалите повторяющиеся данные, если информация хранится в нескольких объектах.

Возьмите данные, которые на данный момент представлены в ERD-диаграмме, и преобразуйте их в ненормализованную форму, чтобы эти данные находились в форме UNF.

Описание решения:

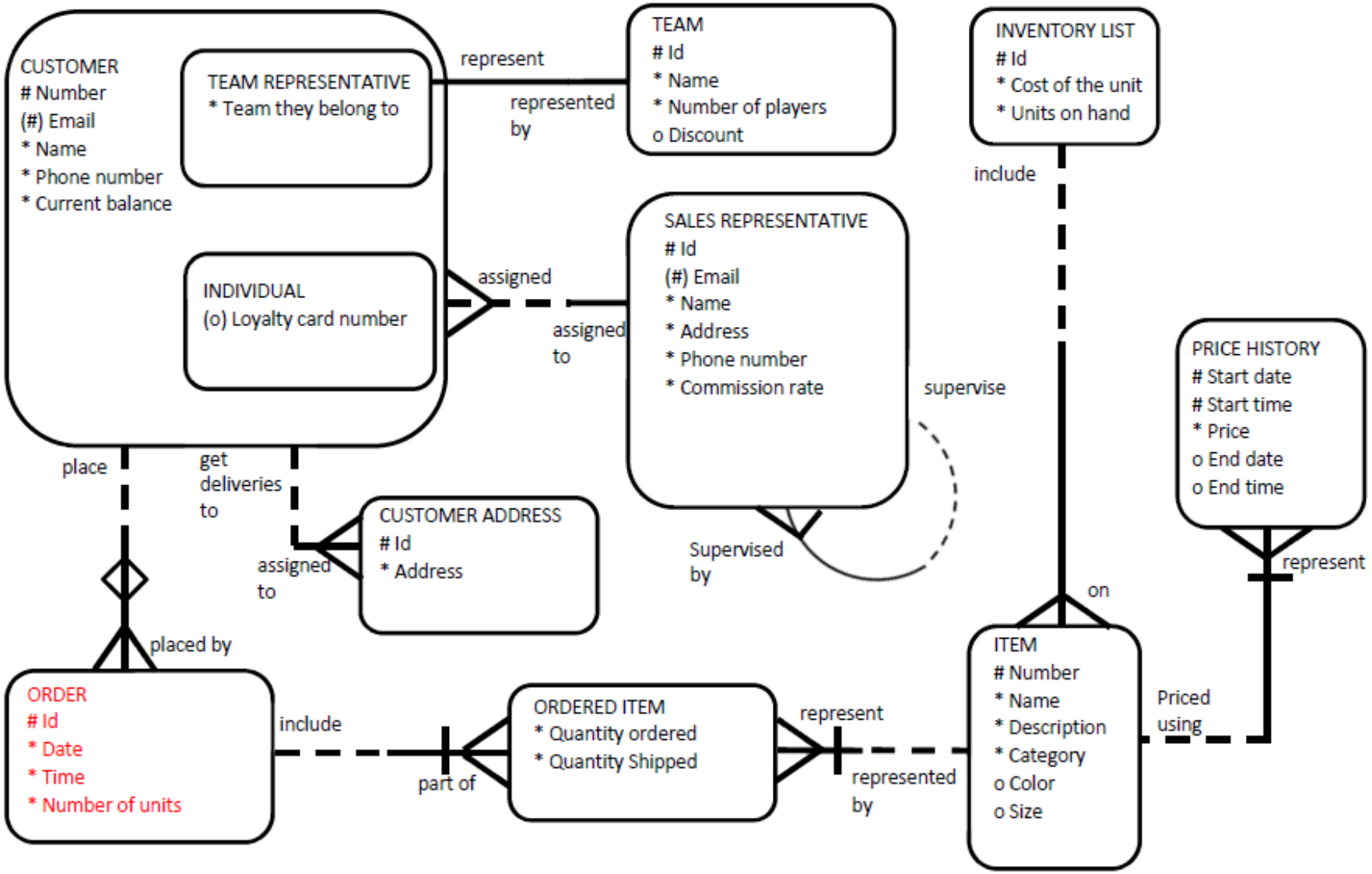
Рассчитанное значение поля «Общая цена заказа» было удалено из объекта ORDER, так как его можно рассчитать с помощью цены товаров.

Дублированные значения цвета и размера между объектами ORDER и ITEM были удалены из объекта ORDER и оставлены в объекте ITEM.

Дублированные атрибуты купленных товаров и заказанного количества были удалены из объекта ORDER и оставлены в объекте ORDERED ITEM.

Дублированное значение цены между объектами ORDER и ITEM PRICE было удалено из объекта ORDER и оставлено в объекте ITEM PRICE.

Решение:


CUSTOMER
Number
(#) Email
* Name
* Phone number
* Current balance
TEAM REPRESENTATIVE
* Team they belong to
INDIVIDUAL
(o) Loyalty card number
represent
represented by
TEAM
Id
* Name
* Number of players
o Discount
assigned
assigned to
SALES REPRESENTATIVE
Id
(#) Email
* Name
* Address
* Phone number
* Commission rate
supervise
Supervised by
INVENTORY LIST
Id
* Cost of the unit
* Units on hand
include
on
PRICE HISTORY
Start date
Start time
* Price
o End date
o End time
represent
Priced using
ITEM
Number
* Name
* Description
* Category
o Color
o Size
place
get deliveries to
assigned to
CUSTOMER ADDRESS
Id
* Address
placed by
ORDER
Id
* Date
* Time
* Number of units
include
part of
ORDERED ITEM
* Quantity ordered
* Quantity Shipped
represent
represented by

Нормализация и бизнес-правила

Проверка данных путем нормализации

Первая нормальная форма

Нормализация — концепция реляционной базы данных, но ее принципы применимы к моделированию данных.

- Первая нормальная форма (1NF)
- Данные являются элементарными (все атрибуты должны быть с одним значением).
- Записи в столбце должны быть одинакового типа.
- В таблице не должно быть повторяющихся строк, это значит, что таблица должна иметь группу столбцов, которая уникально определяют строку.

Возьмите ненормализованные данные, которые на данный момент представлены в ERD-диаграмме, и примените принципы первой нормальной формы, чтобы эти данные находились в форме 1NF.

Описание решения:

Все атрибуты, которые могут содержать более одного значения, разбиты на атрибуты с одним значением. Это атрибуты имени и адреса в объектах.

Терминология моделирования данных и сопоставление.

Часть 1. Создание документа сопоставления таблиц

Моделирование данных позволяет вам понимать и применять правила именования Oracle для концептуальной модели данных, а также применять их при создании физической модели данных. Физическая модель должна таким образом представлять информацию, хранящуюся в ERD, чтобы облегчить создание реляционной базы данных.

Используя ERD-диаграмму, созданную в данном проекте (или рекомендуемую в качестве отправной точки в данном документе), создайте документ сопоставления таблиц, где будет представлено преобразование терминологии концептуальной модели данных в термины физической модели данных.

Инструкции по сопоставлению таблиц

- В первой строке диаграммы таблицы содержатся имя и краткое имя таблицы.
  - Имя таблицы — это множественное число от имени объекта.

- Краткое имя таблицы будет использоваться для именования связей по внешнему ключу и должно быть создано согласно соответствующей стратегии для имени объекта.

- Столбец «Key Type» (Тип ключа) должен содержать значения: pk для первичного ключа, uk для уникального ключа, fk для внешнего ключа. Эта ячейка остается пустой, если столбец не входит в состав ключа.

  При сопоставлении атрибутов внешнего ключа не забывайте о следующих указаниях.

  - Вы создаете внешний ключ на той стороне связи, где есть знак «птичьей лапки».
  - Вы берете первичный ключ из связанной таблицы и добавляете его, используя в качестве идентификатора <**краткое имя таблицы>_<имя атрибута>.**
  - Для внешних ключей не существует определенного порядка идентификации, но мы их нумеруем, чтобы показать, что имеется более одного внешнего ключа.
  - Для связи 1:1 внешний ключ можно добавить в наиболее подходящую таблицу.

- Столбец «Optionality » (Обязательность) должен содержать звездочку (*), если этот столбец обязательный, или букву «о» в нижнем регистре, если он дополнительный.
- Третий столбец предназначен для имени столбца, которое должно совпадать с уникальным именем атрибута, за исключением того, что пробелы заменены символами подчеркивания. Имя столбца не должно содержать имя объекта или таблицы, потому что столбцы указываются с именем таблицы
- Не забывайте, что вам нужно следовать правилам именования на всех этапах, и вы должны быть последовательными в своем подходе к именованию.
- Будьте внимательны, не назначайте ключевые слова, как имена таблиц и столбцов, — это приведет к проблемам при попытке создать физическую модель.
- В качестве примера для вас заполнена первая таблица (customers).

Решение:

Имя таблицы	**Краткое имя таблицы**	
customers	ctr	
Key Type	**Optionality**	**Column Name**
pk	*	ctr_number
uk	*	email
	*	first_name
	*	last_name
	*	phone_number

	*	current_balance
uk	o	loyalty_card_number
fk1	o	tem_id
fk2	o	sre_id

Имя таблицы	**Краткое имя таблицы**	
customers_addresses	cas	
Key Type	**Optionality**	**Column Name**
pk	*	cas_id
	*	address_line
	*	city
	*	postal_code
fk	*	ctr_number

Имя таблицы	**Краткое имя таблицы**	
teams	tem	
Key Type	**Optionality**	**Column Name**
pk	*	tem_id
	*	name
	*	number_of_players
	o	discount

Имя таблицы	**Краткое имя таблицы**	
sales_representatives	sre	
Key Type	**Optionality**	**Column Name**
pk	*	sre_id
uk	*	email
	*	first_name
	*	last_name
	*	phone_number
	*	commission_rate
fk	*	supervisor_id

Имя таблицы	**Краткое имя таблицы**	
sales_representatives_address	**sas**	
Key Type	**Optionality**	**Column Name**
Pk/fk	*	sre_id
	*	address_line
	*	city
	*	postal_code

Имя таблицы	**Краткое имя таблицы**	
orders	odr	
Key Type	**Optionality**	**Column Name**
pk	*	odr_id
	*	odr_date
	*	odr_time
	*	number_of_units
fk	*	ctr_number

Имя таблицы	**Краткое имя таблицы**	
ordered_items	oim	
Key Type	**Optionality**	**Column Name**
	*	quantity_ordered
	*	quantity_shipped
pk, fk1	*	odr_id
pk, fk2	*	itm_number

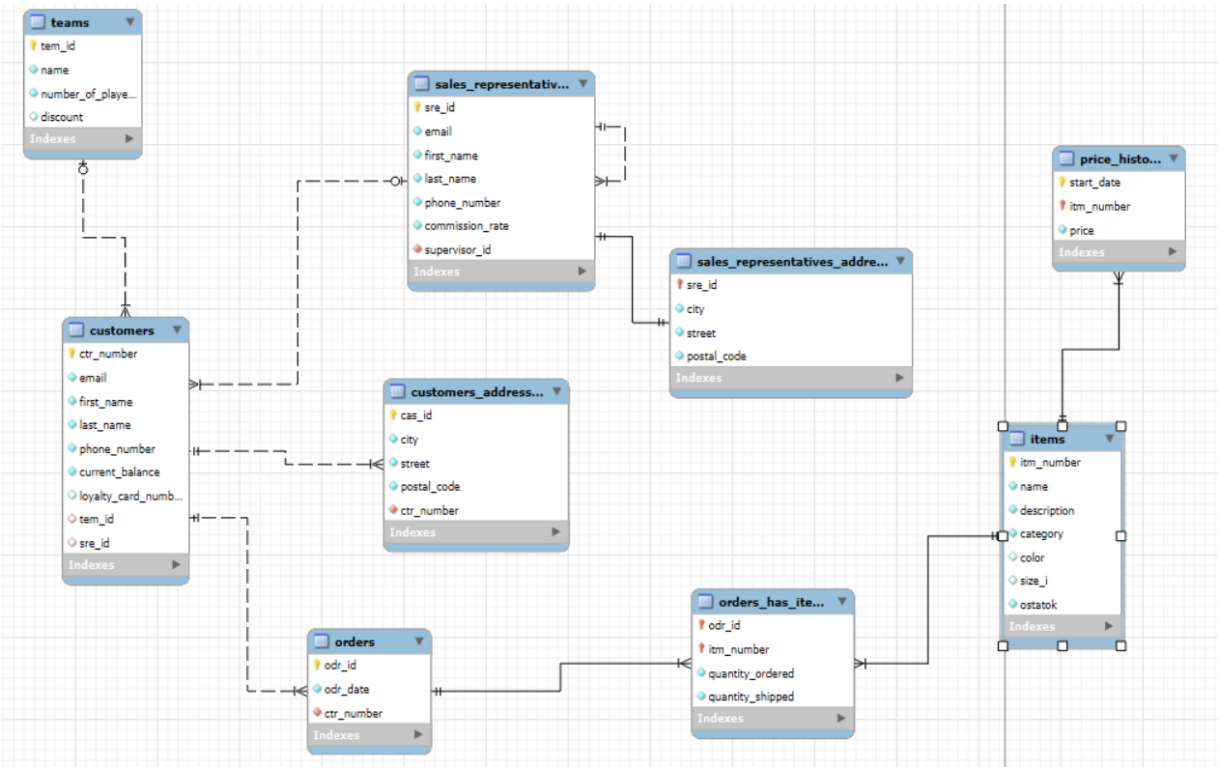
Имя таблицы	**Краткое имя таблицы**	
items	itm	
Key Type	**Optionality**	**Column Name**
pk	*	itm_number
	*	name
	*	description
	*	category
	o	color
	o	razmer
fk	*	ilt_id

Имя таблицы	**Краткое имя таблицы**	
inventory_list	ilt	
Key Type	**Optionality**	**Column Name**
pk	*	Ilt_id
	*	cost
	*	units

Имя таблицы	**Краткое имя таблицы**	
price_history	phy	
Key Type	**Optionality**	**Column Name**

pk	*	start_date
	*	price
pk, fk1	*	itm_number

Результат работы



Критерии оценки.

«5» «отлично»

- студент продемонстрировал готовность и способность к самостоятельному выполнению видов работ в стандартных ситуациях;

- все работы выполнил правильно и уверенно;

- проявил активность в освоении новых практических умений;

- содержание отчета по практическим работам полноценно отражает информацию по выполненным видам работ, установленных программой практики;

- оформление отчета по практическим работам полностью соответствует предъявляемым требованиям;

- при выполнении практических работ студент уверенно продемонстрировал

обладание необходимыми компонентами общих и профессиональных компетенций;

«4» «хорошо»

- студент продемонстрировал готовность и способность к самостоятельному выполнению видов работ в стандартных ситуациях;

- все работы выполнил правильно и уверенно;

- содержание отчета по практическим работам в целом правильно отражает информацию по выполненным видам работ;

- оформление отчета по практическим работам соответствует предъявляемым требованиям, однако имеются замечания (некоторая неаккуратность, неполное описание проделанной работы и освоенных умений);

- при выполнении практических работ студент продемонстрировал обладание компонентами общих и профессиональных компетенций;

- характеристики студента положительные, имеются отдельные замечания и рекомендации;

«3» «удовлетворительно»

- при выполнении видов работ студент нуждался во внешнем сопровождении и контроле;

- в отчете по практическим работам содержится описание видов выполняемых работ согласно программе дисциплины, однако, записи неполные и неточные;

- отчет по практическим работам оформлен с отклонениями от установленных требований;

- при выполнении практических работ студент продемонстрировал обладание отдельными компонентами общих и профессиональных компетенций;

«**2» «неудовлетворительно»**

– студент не справился с запланированными видами работ;

– при выполнении практических работ студент продемонстрировал слабое обладание отдельными компонентами общих и профессиональных компетенций;

– отчет студентом не подготовлен.

Практическое занятие №3. Разработка физической модели базы данных. Создание базы данных с использованием языка SQL.

Цель работы:

1. Выполнить разработку физической модели базы данных «Магазин».
2. Создание базы данных с использованием языка SQL.

Проверяемые результаты обучения:

- ОК 01 Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
- ОК 02 Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной деятельности
- ОК 05 Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
- ОК 09 Пользоваться профессиональной документацией на государственном и иностранном языках
- ПК 2.1 Выявлять проблемы, возникающие в процессе эксплуатации баз данных.
- ПК 2.2 Осуществлять процедуры администрирования баз данных.
- ПК 2.3 Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации.
- ПК 2.4 Формировать требования хранилищ банка данных для обучения.
- ПК 2.5 Подготавливать данные для базы знаний.

Ход работы:

Моделирование данных позволяет вам понимать и применять правила именования для концептуальной модели данных, а также применять их при создании физической модели данных. Физическая модель должна таким образом представлять информацию, хранящуюся в ERD, чтобы облегчить создание реляционной базы данных.

Используя документ сопоставления таблиц, созданный в части 1 данного упражнения, заполните таблицы, указав также соответствующие типы данных и размеры.

Инструкции по сопоставлению таблиц

1. Рекомендуемый отправной пункт, указанный в данном документе, показывает связи между таблицами с образцами данных для каждого поля.
2. Цветовое кодирование указывает, имеются ли связи с первичным ключом или внешним ключом.

3. Столбцы, содержащие значения внешнего ключа, должны иметь одинаковый тип данных (и предпочтительный размер), чтобы содержать данные, которые реализуют эту связь.
4. Данные приведены только в качестве примера, и не следует считать, что эти данные максимального размера.

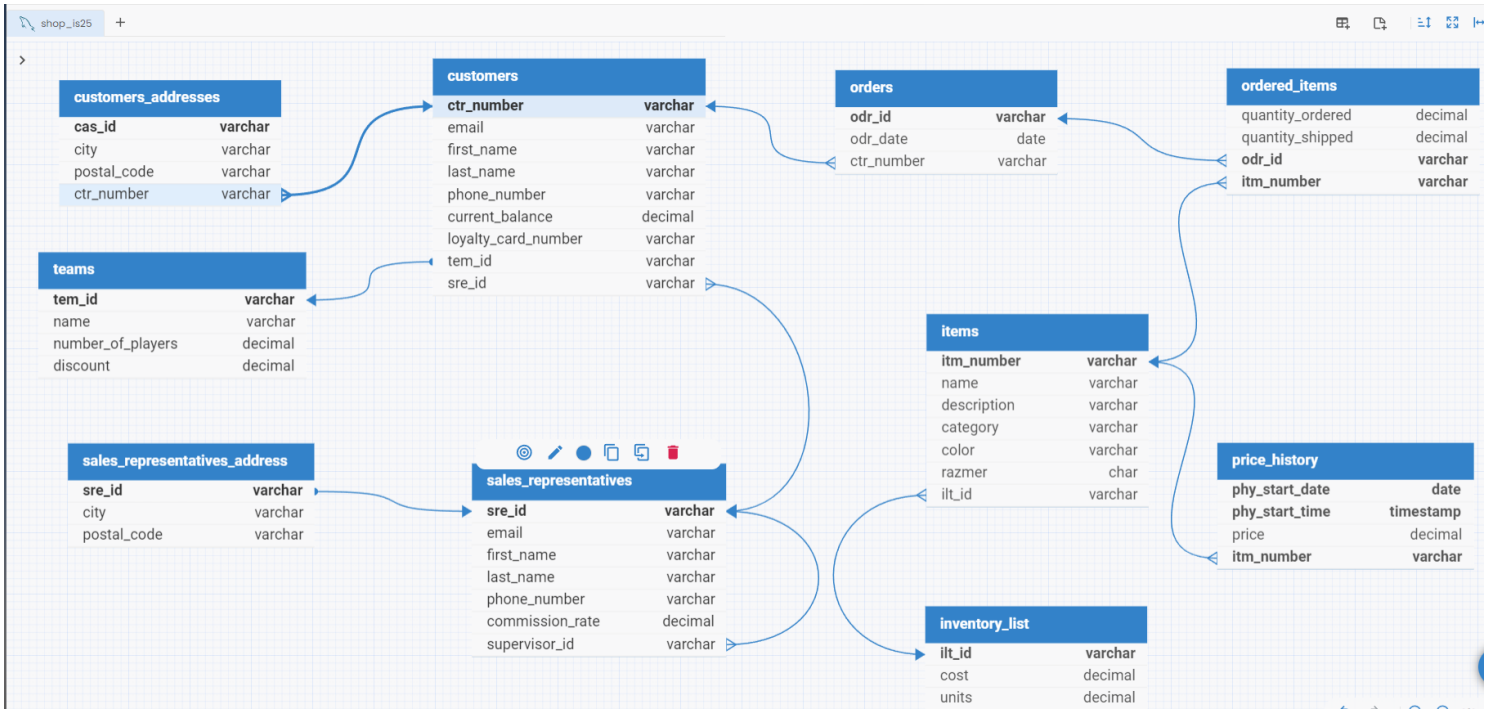
В Для каждого поля следует рассчитать максимальный размер, который, по вашему мнению, лучше всего подходит для типа информации, которая будет там храниться.

В качестве примера для вас заполнена первая таблица (customers).

Имя таблицы	**Краткое имя таблицы**			
customers	ctr			
Key Type	**Optionality**	**Column Name**	**Data type**	**size**
pk	*	ctr_number	VARCHAR	6
uk	*	email	VARCHAR	50
	*	first_name	VARCHAR	20
	*	last_name	VARCHAR	30
	*	phone_number	VARCHAR	11
	*	current_balance	DECIMAL	6,2
uk	o	loyalty_card_number	VARCHAR	6
fk1	o	tem_id	VARCHAR	4
fk2	o	sre_id	VARCHAR	4

Используя данный пример и следующий шаблон, заполните отдельную таблицу для каждого объекта в концептуальной модели данных, где указаны тип данных и размер (в соответствующих случаях) для каждого атрибута.

Результат работы



6. Оформите словарь данных по всем таблицам согласно примеру, приведённому ниже.

Словарь данных

Имя таблицы	**Краткое имя таблицы**					
покупатели / customers	ctr					
Key Type (Тип ключа)	Optionality (Обязательность/ необязательность)	ColumnName (Название столбца)	Datatype (Тип данных)	Size (Размер поля)	Constraints (Ограничения)	Comment (Комментарий)
pk	*	ctr_number	VARCHAR	6	PRIMARY KEY (ctr_number)	Номер покупателя
uk	*	email	VARCHAR	50	UNIQUE KEY email (email)	Электронная почта покупателя
	*	first_name	VARCHAR	20		Фамилия покупателя
	*	last_name	VARCHAR	30		Имя покупателя
	*	phone_number	VARCHAR	11		Номер телефона
	*	current_balance	NUMERIC	6,2	CONSTRAINT customers_chk_1 CHECK ((current_balanc	Текущий баланс покупателя

					e > = 0))	
	o	loyalty_card_number	VARCHAR	6		Номер карты лояльности
fk1	o	tem_id	VARCHAR	4	CONSTRAINT customers_fk0 FOREIGN KEY (tem_id) REFERENCES teams (tem_id)	Код команды
fk2	o	sre_id	VARCHAR	4	CONSTRAINT customers_fk1 FOREIGN KEY (sre_id) REFERENCES sales_representatives (sre_id)	Код продавца

7. Опишите пользователей и их привилегии

Пользователь	Описание, обязанности	Привилегии в БД
root	Администратор. Полный доступ ко всем объектам.	ALL PRIVILEGES, GRANT

8. Выполните генерацию SQL-файла разработанной БД.
9. Выполните создание БД из готового SQL-файла.

Критерии оценки.

«5» «отлично»

- студент продемонстрировал готовность и способность к самостоятельному выполнению видов работ в стандартных ситуациях;

- все работы выполнил правильно и уверенно;

- проявил активность в освоении новых практических умений;

- содержание отчета по практическим работам полноценно отражает информацию по выполненным видам работ, установленных программой практики;

- оформление отчета по практическим работам полностью соответствует

предъявляемым требованиям;

- при выполнении практических работ студент уверенно продемонстрировал обладание необходимыми компонентами общих и профессиональных компетенций;

«4» «хорошо»

- студент продемонстрировал готовность и способность к самостоятельному выполнению видов работ в стандартных ситуациях;

- все работы выполнил правильно и уверенно;

- содержание отчета по практическим работам в целом правильно отражает информацию по выполненным видам работ;

- оформление отчета по практическим работам соответствует предъявляемым требованиям, однако имеются замечания (некоторая неаккуратность, неполное описание проделанной работы и освоенных умений);

- при выполнении практических работ студент продемонстрировал обладание компонентами общих и профессиональных компетенций;

- характеристики студента положительные, имеются отдельные замечания и рекомендации;

«3» «удовлетворительно»

- при выполнении видов работ студент нуждался во внешнем сопровождении и контроле;

- в отчете по практическим работам содержится описание видов выполняемых работ согласно программе дисциплины, однако, записи неполные и неточные;

- отчет по практическим работам оформлен с отклонениями от установленных требований;

- при выполнении практических работ студент продемонстрировал обладание отдельными компонентами общих и профессиональных компетенций;

«**2» «неудовлетворительно»**

– студент не справился с запланированными видами работ;

– при выполнении практических работ студент продемонстрировал слабое

обладание отдельными компонентами общих и профессиональных компетенций;

– отчет студентом не подготовлен.

Тема 2. Разработка и администрирование БД.

Проверяемые результаты обучения:

- ОК 01 Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
- ОК 02 Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной деятельности
- ОК 05 Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
- ОК 09 Пользоваться профессиональной документацией на государственном и иностранном языках
- ПК 2.1 Выявлять проблемы, возникающие в процессе эксплуатации баз данных.
- ПК 2.2 Осуществлять процедуры администрирования баз данных.
- ПК 2.3 Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации.
- ПК 2.4 Формировать требования хранилищ банка данных для обучения.
- ПК 2.5 Подготавливать данные для базы знаний.

Критерии оценки устного опроса:

− оценка **«отлично»** выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов и смог ответить на уточняющие вопросы;

− оценка **«хорошо»** выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов, но не смог ответить на уточняющие вопросы;

− оценка **«удовлетворительно»** выставляется студенту, если он смог изложить суть основного вопроса, но не смог ответить по существу дополнительных вопросов;

− оценка **«неудовлетворительно»** выставляется студенту, если он не смог изложить суть основного вопроса

Вопросы к устному опросу по теме:

1. Что такое SQL и для чего он используется?

Ответ: SQL (*Structured Query Language*) — это язык структурированных запросов, предназначенный для работы с реляционными базами данных. Он используется для создания, модификации, управления и извлечения данных из баз данных.

2. Какова история возникновения SQL?

Ответ: SQL был разработан в начале 1970-х годов в компании IBM под названием SEQUEL (*Structured English Query Language*) для работы с прототипом реляционной СУБД System R. В 1986 году был принят первый стандарт ANSI/ISO, и язык получил современное название SQL.

3. Какие основные типы команд существуют в SQL?
Ответ: В SQL выделяют четыре основных типа команд: - DDL (*Data Definition Language*) — язык определения данных. - DML (*Data Manipulation Language*) — язык манипулирования данными. - DCL (*Data Control Language*) — язык управления доступом к данным. - TCL (*Transaction Control Language*) — язык управления транзакциями.
4. Что такое DDL и какие команды к нему относятся?
Ответ: DDL — это команды для определения и изменения структуры базы данных. К ним относятся:
- CREATE — создание объектов (таблиц, индексов и др.).
- ALTER — изменение структуры существующих объектов.
- DROP — удаление объектов.
- TRUNCATE — удаление всех данных из таблицы.
5. Что такое DML и какие команды к нему относятся?
Ответ: DML — это команды для работы с данными внутри таблиц:
- SELECT — выборка данных.
- INSERT — добавление новых записей.
- UPDATE — изменение существующих записей.
- DELETE — удаление записей.
6. Что такое DCL и какие команды к нему относятся?
Ответ: DCL — это команды для управления правами доступа к данным:
- GRANT — предоставление прав пользователю.
- REVOKE — отзыв ранее предоставленных прав.
7. Что такое TCL и какие команды к нему относятся?
Ответ: TCL — это команды для управления транзакциями:
- COMMIT — сохранение изменений, сделанных в транзакции.
- ROLLBACK — отмена изменений, сделанных в транзакции.
- SAVEPOINT — установка точки сохранения внутри транзакции.
8. В чём разница между командами DELETE и TRUNCATE?
Ответ: DELETE удаляет строки по условию и может быть отменена с помощью ROLLBACK. TRUNCATE удаляет все строки из таблицы без условий, работает быстрее и не может быть отменена в рамках транзакции (в большинстве СУБД).
9. Для чего используется команда GRANT?
Ответ: Команда GRANT используется для предоставления определённых прав доступа (например, на выборку или изменение данных) пользователям или ролям базы данных.
10. Какую роль играет SQL в современных информационных системах? Ответ: SQL является стандартом для работы с реляционными базами данных, используется в большинстве корпоративных и веб-приложений для хранения, обработки и анализа данных. Благодаря стандартизации, знания SQL применимы к различным СУБД (PostgreSQL, MySQL, Oracle, MS SQL Server и др.).

11. Какова общая структура оператора SELECT в SQL?
Ответ:
Оператор SELECT имеет следующий базовый синтаксис:
SELECT столбец1, столбец2, ...

FROM имя_таблицы
WHERE условие;
- **SELECT** — перечисляет столбцы для выборки (или * для всех).
- **FROM** — указывает таблицу, из которой берутся данные.
- **WHERE** — задаёт условия фильтрации строк (необязательно).

12. Какие основные типы данных поддерживаются в SQL?
Ответ:
В SQL существуют следующие основные типы данных:
- **Числовые:** INTEGER, SMALLINT, BIGINT, DECIMAL, NUMERIC, FLOAT, REAL.
- **Символьные:** CHAR, VARCHAR, TEXT.
- **Дата и время:** DATE, TIME, DATETIME, TIMESTAMP.
- **Логические:** BOOLEAN.
- **Бинарные:** BLOB, BYTEA.

13. Как объявить столбец с типом данных VARCHAR в SQL?
Ответ:
Пример объявления столбца с типом VARCHAR:
CREATE TABLE users (
id INT PRIMARY KEY,
name VARCHAR(50)
);
```
Здесь **name** — столбец, который может хранить до 50 символов.

14. В чём разница между типами данных CHAR и VARCHAR?
Ответ:
- **CHAR(n)** — фиксированная длина. Если строка короче n символов, она дополняется пробелами.
- **VARCHAR(n)** — переменная длина. Хранит только фактическое количество символов, не более n.

15. Как использовать оператор INSERT для добавления данных в таблицу?
Ответ:
Пример синтаксиса INSERT:
INSERT INTO users (id, name, age)
VALUES (1, 'Иван', 25);
- **INSERT INTO** — указывает таблицу.
- **(столбцы)** — перечисляются столбцы (можно опустить, если вставляются все).
- **VALUES** — задаёт значения для вставки.

16. Как изменить тип данных существующего столбца в таблице?
Ответ:
Для изменения типа данных используется оператор ALTER TABLE:
```

sql
ALTER TABLE users
ALTER COLUMN age TYPE BIGINT;
Синтаксис может отличаться в зависимости от СУБД.

17. Что такое NULL в SQL и как с ним работать?
Ответ:
NULL — специальное значение, обозначающее отсутствие данных. Для проверки на NULL используются операторы:
- **IS NULL** — значение отсутствует.
- **IS NOT NULL** — значение присутствует.
Пример:
SELECT * FROM users WHERE age IS NULL;

18. Какие правила именования идентификаторов (таблиц, столбцов) существуют в SQL?
Ответ:
- Имена не должны содержать пробелов (используются подчёркивания или CamelCase).
- Не должны совпадать с зарезервированными словами SQL.
- В некоторых СУБД имена чувствительны к регистру и могут требовать кавычек.

19. Как задать значение по умолчанию для столбца при создании таблицы?
Ответ:
С помощью ключевого слова DEFAULT:
CREATE TABLE orders (
id INT PRIMARY KEY,
status VARCHAR(20) DEFAULT 'новый'
);
Если значение не указано при вставке, будет использовано 'новый'.

20. Как ограничить значения в столбце с помощью CHECK?
Ответ:
Ограничение CHECK задаёт условие для допустимых значений:
CREATE TABLE products (
price DECIMAL CHECK (price > 0)
);

21. Как выполнить сортировку данных в SQL?
Ответ: Для сортировки используется оператор **ORDER BY**.
SELECT * FROM employees ORDER BY salary DESC; ```
- **ASC** — по возрастанию (по умолчанию).
- **DESC** — по убыванию.
22. Что такое группировка данных и как она реализуется в SQL?
Ответ: Группировка выполняется с помощью оператора **GROUP BY**. Он объединяет строки с одинаковыми значениями в указанных столбцах. SELECT

department, COUNT(*) FROM employees GROUP BY department; ```
23.Какие основные агрегатные функции существуют в SQL?
Ответ:
- **COUNT()** — количество строк.
- **SUM()** — сумма значений.
- **AVG()** — среднее значение.
- **MAX()** — максимальное значение.
- **MIN()** — минимальное значение.
24.Как использовать фильтрацию для сгруппированных данных?
Ответ: Для фильтрации групп используется оператор **HAVING**.
SELECT department, AVG(salary) FROM employees GROUP BY department
HAVING AVG(salary) > 50000; ```
25. В чём разница между INNER JOIN и LEFT JOIN?
Ответ:
- **INNER JOIN** возвращает только те строки, которые есть в обеих таблицах.
- **LEFT JOIN** возвращает все строки из левой таблицы и соответствующие строки из правой (если нет — NULL).
26. Что такое RIGHT JOIN и FULL JOIN?
Ответ:
- **RIGHT JOIN** — все строки из правой таблицы и соответствующие из левой.
- **FULL JOIN** — все строки из обеих таблиц (объединяет LEFT и RIGHT JOIN).
27. Какие операции над множествами поддерживает SQL?
Ответ:
- **UNION** — объединяет результаты двух запросов, удаляя дубликаты.
- **UNION ALL** — объединяет с сохранением дубликатов.
- **INTERSECT** — возвращает только совпадающие строки.
- **EXCEPT / MINUS** — возвращает строки из первого запроса, которых нет во втором.
28. Что такое индекс в SQL и для чего он нужен?
Ответ: Индекс — это специальная структура, ускоряющая поиск и сортировку данных по определённым столбцам. Он работает как алфавитный указатель в книге.
29. Какие типы индексов существуют?
Ответ:
- **Обычный (B-tree)** — для большинства случаев.
- **Уникальный** — не допускает дубликатов.
- **Составной** — по нескольким столбцам.
- **Полнотекстовый** — для поиска по тексту.
30. Как оптимизировать SQL-запросы?
Ответ:
- Использовать индексы по столбцам, участвующим в WHERE, JOIN, ORDER BY.
- Избегать SELECT * (выбирать только нужные столбцы).

- Ограничивать количество строк с помощью LIMIT.
- Анализировать планы выполнения запросов (EXPLAIN).
- Избегать сложных подзапросов, если можно использовать JOIN.

31. Какая утилита командной строки MySQL предназначена для создания логической резервной копии базы данных в виде набора SQL-операторов?

А) mysqlimport
Б) mysqldump ✓
В) mysqlcheck
Г) mysqladmin

32. Какой SQL-оператор используется для выгрузки данных из таблицы в текстовый файл на сервере?

А) LOAD DATA INFILE
Б) INSERT INTO ... SELECT
В) SELECT ... INTO OUTFILE ✓
Г) EXPORT TABLE TO FILE

33. Какой формат данных наиболее универсален для обмена информацией между различными СУБД и приложениями благодаря своей текстовой структуре с разделителями?

А) XML
Б) CSV ✓
В) JSON
Г) Parquet

34. Какой SQL-оператор используется для создания нового пользователя в СУБД MySQL?

А) ADD USER
Б) NEW USER
В) CREATE USER ✓
Г) INSERT USER

35. Что такое «роль» в системе управления доступом СУБД?

А) Отдельный пользователь с правами администратора
Б) Именованный набор привилегий, который можно назначать пользователям ✓
В) Таблица с перечнем всех пользователей системы
Г) Временная учётная запись для гостевого доступа

36. Какой оператор применяется для назначения привилегий пользователю или роли?

А) ASSIGN
Б) ALLOW
В) GRANT ✓
Г) PERMIT

37. Какой уровень привилегий в MySQL предоставляет права на выполнение операций во всех базах данных сервера?

А) Уровень таблицы
Б) Уровень базы данных

В) Глобальный уровень ✅
Г) Уровень столбца
38. Какой тип резервного копирования сохраняет только те данные, которые изменились с момента последнего полного бэкапа?
А) Полное копирование
Б) Дифференциальное копирование
В) Инкрементальное копирование ✅
Г) Зеркальное копирование
39. Какая команда MySQL используется для восстановления базы данных из SQL-дампа, созданного утилитой mysqldump?
А) mysql database_name < backup.sql ✅
Б) mysqldump --restore database_name
В) RESTORE DATABASE database_name FROM backup.sql
Г) mysqlimport database_name backup.sql
40. Что такое «транзакционно-консистентный бэкап»?
А) Копия, сделанная во время планового отключения сервера
Б) Копия, в которой все незавершённые транзакции либо зафиксированы, либо откачены, что гарантирует целостность данных ✅
В) Копия, содержащая только структуру таблиц без данных
Г) Копия, сделанная исключительно на уровне файловой системы

Практические занятия к теме 2

Практическое занятие №4. Создание базы данных и таблиц с использованием языка SQL. Реализация ограничений целостности
Проверяемые результаты обучения:

- ОК 01 Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
- ОК 02 Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной деятельности
- ОК 05 Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
- ОК 09 Пользоваться профессиональной документацией на государственном и иностранном языках
- ПК 2.1 Выявлять проблемы, возникающие в процессе эксплуатации баз данных.
- ПК 2.2 Осуществлять процедуры администрирования баз данных.
- ПК 2.3 Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации.
- ПК 2.4 Формировать требования хранилищ банка данных для обучения.
- ПК 2.5 Подготавливать данные для базы знаний.

Цели работы

Тип цели	Формулировка
Дидактическая	Сформировать понимание типов данных MySQL, принципов проектирования реляционных таблиц и видов ограничений целостности (PRIMARY KEY, FOREIGN KEY, UNIQUE, NOT NULL, CHECK, DEFAULT).
Практическая	Научиться создавать базу данных и таблицы с использованием оператора CREATE TABLE, применять различные ограничения целостности и проверять их корректную работу на тестовых данных.
Развивающая	Развить навыки проектирования структуры БД с учётом предметной области, понимание принципов ссылочной целостности и каскадных операций.

Материально-техническое обеспечение

Компонент	Описание
Оборудование	ПК с доступом к локальной сети или интернету
ПО	Установленная СУБД MySQL (версии 8.0+), клиент для работы с БД (MySQL Workbench, DBeaver, phpMyAdmin или консольный клиент mysql)
Раздаточный материал	Описание предметной области «Интернет-магазин», шпаргалка по типам данных и ограничениям MySQL

Краткие торические сведения:

Ограничения целостности — это правила, которые СУБД автоматически проверяет при выполнении операций INSERT, UPDATE, DELETE. Они гарантируют корректность, непротиворечивость и надёжность данных.

Основные типы ограничений в MySQL:

Ограничение	Назначение	Пример
PRIMARY KEY	Уникально идентифицирует каждую запись. Не допускает NULL. В таблице может быть только один.	id INT PRIMARY KEY
FOREIGN KEY	Обеспечивает ссылочную целостность — связывает поле одной таблицы с PRIMARY KEY другой.	FOREIGN KEY (customer_id) REFERENCES customers(id)
UNIQUE	Гарантирует уникальность значений в столбце. Допускает одно значение NULL.	email VARCHAR(100) UNIQUE
NOT NULL	Запрещает хранение пустых (NULL) значений.	name VARCHAR(100) NOT NULL
CHECK	Проверяет значение по заданному логическому условию (в MySQL 8.0.16+).	price DECIMAL(10,2) CHECK (price > 0)
DEFAULT	Устанавливает значение по умолчанию, если оно не указано при вставке.	status VARCHAR(20) DEFAULT 'новый'

Каскадные операции (для FOREIGN KEY):

- ON DELETE CASCADE — при удалении родительской записи автоматически удаляются все связанные дочерние.
- ON UPDATE CASCADE — при изменении ключа родителя автоматически обновляются значения в дочерних таблицах.
- ON DELETE SET NULL — при удалении родителя в дочерних записях поле устанавливается в NULL.

Ход работы

Этап 1. Создание базы данных (10 мин)

1. Откройте клиент MySQL и создайте базу данных:

```
CREATE DATABASE online_store;

USE online_store;
```

2. Проверьте текущую базу данных:

```
SELECT DATABASE();
```

Этап 2. Создание таблицы «Клиенты» (15 мин)

3. Создайте таблицу customers с использованием различных ограничений

```
CREATE TABLE customers (
    id INT AUTO_INCREMENT PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    phone VARCHAR(20),
    registration_date DATE DEFAULT (CURDATE()),
    is_active BOOLEAN DEFAULT TRUE,
    CHECK (email LIKE '%@%.%')
);
```

4. Проверьте структуру таблицы

```
DESCRIBE customers;
```

5. Попробуйте вставить некорректные данные, чтобы проверить работу ограничений:

```
-- Должно выполниться успешно
INSERT INTO customers (first_name, last_name, email, phone)
VALUES ('Иван', 'Петров', 'ivan@example.com', '+79991234567');
-- Должно вызвать ошибку (дубликат email - нарушение UNIQUE)
```

```
INSERT INTO customers (first_name, last_name, email)
VALUES ('Пётр', 'Сидоров', 'ivan@example.com');


-- Должно вызвать ошибку (email без '@' - нарушение CHECK)
INSERT INTO customers (first_name, last_name, email)
VALUES ('Анна', 'Иванова', 'anna-mail.ru');
```

Этап 3. Создание таблицы «Товары» (15 мин)

6. Создайте таблицу products

```
CREATE TABLE products (
    id INT AUTO_INCREMENT PRIMARY KEY,
    article VARCHAR(20) NOT NULL UNIQUE,
    name VARCHAR(150) NOT NULL,
    description TEXT,
    price DECIMAL(10, 2) NOT NULL,
    stock_quantity INT NOT NULL DEFAULT 0,
    category VARCHAR(50),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    CHECK (price > 0),
    CHECK (stock_quantity >= 0)
);
```

7. Добавьте тестовые данные:

```
INSERT INTO products (article, name, description, price, stock_quantity, category)
VALUES
```

('SMART-001', 'Смартфон X1', 'Современный смартфон', 25000.00, 50, 'Электроника'),

('LAPTOP-002', 'Ноутбук Pro', 'Мощный ноутбук для работы', 75000.00, 20, 'Электроника'),

('HEAD-003', 'Наушники Wireless', 'Беспроводные наушники', 5000.00, 100, 'Аксессуары');

8. Проверьте ограничение CHECK:

-- Должно вызвать ошибку (отрицательная цена)

INSERT INTO products (article, name, price, stock_quantity)

VALUES ('TEST-999', 'Тестовый товар', -100.00, 10);

Этап 4. Создание таблицы «Заказы» со ссылочной целостностью (20 мин)

9. Создайте таблицу orders, связанную с customers:

```
CREATE TABLE orders (
    id INT AUTO_INCREMENT PRIMARY KEY,
    customer_id INT NOT NULL,
    order_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    status VARCHAR(20) DEFAULT 'новый',
    total_amount DECIMAL(12, 2) DEFAULT 0.00,
    delivery_address TEXT NOT NULL,
    FOREIGN KEY (customer_id) REFERENCES customers(id)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    CHECK (status IN ('новый', 'в обработке', 'отправлен', 'доставлен', 'отменён')),
    CHECK (total_amount >= 0)
);
```

10. Создайте таблицу order_items (позиции заказа) — она связывает orders и products

```
CREATE TABLE order_items (
    id INT AUTO_INCREMENT PRIMARY KEY,
    order_id INT NOT NULL,
    product_id INT NOT NULL,
    quantity INT NOT NULL,
    unit_price DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (order_id) REFERENCES orders(id)
        ON DELETE CASCADE,
    FOREIGN KEY (product_id) REFERENCES products(id)
        ON DELETE RESTRICT,
    CHECK (quantity > 0),
    CHECK (unit_price > 0),
    UNIQUE (order_id, product_id)
);
```

Этап 5. Проверка ссылочной целостности (15 мин)

11. Добавьте тестовые заказы и позиции

```
-- Заказ от клиента с id=1
INSERT INTO orders (customer_id, delivery_address, total_amount)
VALUES (1, 'г. Москва, ул. Ленина, д. 10', 30000.00);


-- Позиции заказа
```

```
INSERT INTO order_items (order_id, product_id, quantity, unit_price)
VALUES
(1, 1, 1, 25000.00),
(1, 3, 1, 5000.00);
```

12.Проверьте работу ссылочной целостности

```
-- Должно вызвать ошибку (несуществующий customer_id)
INSERT INTO orders (customer_id, delivery_address)
VALUES (999, 'г. Санкт-Петербург');

-- Должно вызвать ошибку (несуществующий product_id)
INSERT INTO order_items (order_id, product_id, quantity, unit_price)
VALUES (1, 999, 2, 1000.00);
```

13. Проверьте каскадное удаление:

```
-- Удаляем клиента — его заказ и позиции заказа удалятся автоматически
DELETE FROM customers WHERE id = 1;

-- Проверяем результат
SELECT * FROM orders;
SELECT * FROM order_items;
```

Этап 6. Просмотр структуры и ограничений (10 мин)

14.Просмотрите информацию о всех созданных таблицах:

```
SHOW TABLES;
```

```
SHOW CREATE TABLE customers;

SHOW CREATE TABLE orders;

SHOW CREATE TABLE order_items;
```

15. Просмотрите информацию о внешних ключах через information_schema:

```
SELECT
   TABLE_NAME,
   COLUMN_NAME,
   CONSTRAINT_NAME,
   REFERENCED_TABLE_NAME,
   REFERENCED_COLUMN_NAME
FROM INFORMATION_SCHEMA.KEY_COLUMN_USAGE
WHERE REFERENCED_TABLE_SCHEMA = 'online_store';
```

Этап 7. Модификация структуры (5 мин)

16. Добавьте новое ограничение CHECK к существующей таблице через ALTER TABLE

```
ALTER TABLE customers
ADD CONSTRAINT chk_phone CHECK (phone IS NULL OR phone LIKE '+7%');
```

Добавьте новое поле с ограничением:

```
ALTER TABLE products
ADD COLUMN discount_percent DECIMAL(4,2) DEFAULT 0.00
CHECK (discount_percent BETWEEN 0 AND 100);
```

Критерии оценивания

Критерий	Показатели успешности	Баллы
Корректность	Все таблицы созданы без синтаксических ошибок,	0–3

Критерий	Показатели успешности	Баллы
SQL-кода	используются правильные типы данных	
Применение ограничений	Верно использованы все 6 типов ограничений (PK, FK, UNIQUE, NOT NULL, CHECK, DEFAULT)	0–3
Ссылочная целостность	Корректно настроены FOREIGN KEY с каскадными операциями, связи между таблицами логичны	0–2
Тестирование ограничений	Студент продемонстрировал работу ограничений на примерах некорректных данных	0–2
Оформление отчёта	Отчёт заполнен полностью, выводы логичны и аргументированы	0–2
ИТОГО		0–12
Порог зачёта	9 баллов (75%)	✅
Бонус	+1 балл за самостоятельную модификацию структуры через ALTER TABLE с добавлением новых ограничений	

Критерии оценки.

«5» «отлично»

- студент продемонстрировал готовность и способность к самостоятельному выполнению видов работ в стандартных ситуациях;

- все работы выполнил правильно и уверенно;

- проявил активность в освоении новых практических умений;

- содержание отчета по практическим работам полноценно отражает информацию по выполненным видам работ, установленных программой практики;

- оформление отчета по практическим работам полностью соответствует предъявляемым требованиям;

- при выполнении практических работ студент уверенно продемонстрировал обладание необходимыми компонентами общих и профессиональных компетенций;

«4» «хорошо»

- студент продемонстрировал готовность и способность к самостоятельному выполнению видов работ в стандартных ситуациях;

- все работы выполнил правильно и уверенно;

- содержание отчета по практическим работам в целом правильно отражает информацию по выполненным видам работ;

- оформление отчета по практическим работам соответствует предъявляемым требованиям, однако имеются замечания (некоторая неаккуратность, неполное описание проделанной работы и освоенных умений);

- при выполнении практических работ студент продемонстрировал обладание компонентами общих и профессиональных компетенций;

- характеристики студента положительные, имеются отдельные замечания и рекомендации;

«3» «удовлетворительно»

- при выполнении видов работ студент нуждался во внешнем сопровождении и контроле;

- в отчете по практическим работам содержится описание видов выполняемых работ согласно программе дисциплины, однако, записи неполные и неточные;

- отчет по практическим работам оформлен с отклонениями от установленных требований;

- при выполнении практических работ студент продемонстрировал обладание отдельными компонентами общих и профессиональных компетенций;

«**2» «неудовлетворительно»**

– студент не справился с запланированными видами работ;

– при выполнении практических работ студент продемонстрировал слабое обладание отдельными компонентами общих и профессиональных компетенций;

– отчет студентом не подготовлен.

Практическое занятие №5. Написание и выполнение SQL-запросов. Настройка индексов.

Проверяемые результаты обучения:

- ОК 01 Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
- ОК 02 Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной деятельности
- ОК 05 Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
- ОК 09 Пользоваться профессиональной документацией на государственном и иностранном языках
- ПК 2.1 Выявлять проблемы, возникающие в процессе эксплуатации баз данных.
- ПК 2.2 Осуществлять процедуры администрирования баз данных.
- ПК 2.3 Проводить аудит систем безопасности баз данных с использованием

регламентов по защите информации.
- ПК 2.4 Формировать требования хранилищ банка данных для обучения.
- ПК 2.5 Подготавливать данные для базы знаний.

Цели работы

Тип цели	**Формулировка**
Дидактическая	Закрепить знания о типах соединений (JOIN), агрегатных функциях, группировке данных, а также о назначении и типах индексов в СУБД MySQL.
Практическая	Научиться составлять сложные SQL-запросы с использованием JOIN, GROUP BY, HAVING, подзапросов; создавать и анализировать индексы с помощью оператора EXPLAIN.
Развивающая	Развить навыки анализа производительности запросов, принятия обоснованных решений об индексации и оптимизации SQL-кода.

Материально-техническое обеспечение

Компонент	**Описание**
Оборудование	ПК с доступом к локальной сети или интернету
ПО	Установленная СУБД MySQL (версии 8.0+), клиент для работы с БД (MySQL Workbench, DBeaver, phpMyAdmin или консольный клиент mysql)
Раздаточный материал	База данных online_store (созданная в предыдущей практической работе), шпаргалка по синтаксису SQL и оператору EXPLAIN

Краткие теоретические сведения

1. Типы соединений (JOIN) в MySQL

Тип JOIN	**Описание**
INNER	Возвращает только те строки, для которых есть совпадение в обеих

Тип JOIN	Описание
JOIN	таблицах
LEFT JOIN	Возвращает все строки из левой таблицы + совпадения из правой (если нет совпадения — NULL)
RIGHT JOIN	Возвращает все строки из правой таблицы + совпадения из левой
CROSS JOIN	Декартово произведение — каждая строка одной таблицы с каждой строкой другой

2. Агрегатные функции и группировка

Функция	Назначение
COUNT(*) / COUNT(column)	Подсчёт строк / непустых значений
SUM(column)	Сумма значений
AVG(column)	Среднее значение
MIN(column) / MAX(column)	Минимальное / максимальное значение

Важно: при использовании агрегатных функций с GROUP BY фильтр по группам задаётся через HAVING, а не через WHERE.

3. Индексы в MySQL

Индекс — это специальная структура данных (обычно B-Tree), которая ускоряет поиск строк по определённым полям без полного сканирования таблицы.

Типы индексов:

- PRIMARY KEY — уникальный индекс первичного ключа
- UNIQUE — гарантирует уникальность значений
- INDEX (обычный) — ускоряет поиск без ограничений уникальности
- FULLTEXT — для полнотекстового поиска

Оператор EXPLAIN показывает план выполнения запроса. Ключевые столбцы:

- **type** — тип соединения (ALL = полное сканирование, ref/const = использование индекса)
- **possible_keys** — индексы, которые *могли бы* быть использованы
- **key** — индекс, который MySQL *реально* выбрал
- **rows** — оценочное количество читаемых строк

Ход работы

Этап 1. Подготовка тестовой среды (10 мин)

1. Откройте клиент MySQL и убедитесь, что база данных online_store существует и содержит таблицы, созданные в предыдущей практической работе:

```
USE online_store;

SHOW TABLES;
```

2. Если база данных отсутствует, создайте её и заполните тестовыми данными:

```
CREATE DATABASE IF NOT EXISTS online_store;

USE online_store;



-- Таблица клиентов

CREATE TABLE IF NOT EXISTS customers (

  id INT AUTO_INCREMENT PRIMARY KEY,

  first_name VARCHAR(50) NOT NULL,

  last_name VARCHAR(50) NOT NULL,

  email VARCHAR(100) NOT NULL UNIQUE,

  phone VARCHAR(20),

  city VARCHAR(50),
```

```
    registration_date DATE DEFAULT (CURDATE())
);

-- Таблица товаров
CREATE TABLE IF NOT EXISTS products (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(150) NOT NULL,
    category VARCHAR(50),
    price DECIMAL(10, 2) NOT NULL,
    stock_quantity INT NOT NULL DEFAULT 0
);

-- Таблица заказов
CREATE TABLE IF NOT EXISTS orders (
    id INT AUTO_INCREMENT PRIMARY KEY,
    customer_id INT NOT NULL,
    order_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    status VARCHAR(20) DEFAULT 'новый',
    total_amount DECIMAL(12, 2) DEFAULT 0.00,
    FOREIGN KEY (customer_id) REFERENCES customers(id)
);

-- Таблица позиций заказа
```

```
CREATE TABLE IF NOT EXISTS order_items (
    id INT AUTO_INCREMENT PRIMARY KEY,
    order_id INT NOT NULL,
    product_id INT NOT NULL,
    quantity INT NOT NULL,
    unit_price DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (order_id) REFERENCES orders(id),
    FOREIGN KEY (product_id) REFERENCES products(id)
);
```

Заполните таблицы тестовыми данными (не менее 20 клиентов, 15 товаров, 30 заказов):

```
INSERT INTO customers (first_name, last_name, email, phone, city) VALUES
('Иван', 'Петров', 'ivan@mail.ru', '+79991111111', 'Москва'),
('Анна', 'Сидорова', 'anna@mail.ru', '+79992222222', 'Санкт-Петербург'),
('Пётр', 'Иванов', 'petr@mail.ru', '+79993333333', 'Москва'),
('Ольга', 'Смирнова', 'olga@mail.ru', '+79994444444', 'Казань'),
('Дмитрий', 'Кузнецов', 'dmitry@mail.ru', '+79995555555', 'Москва'),
('Елена', 'Попова', 'elena@mail.ru', '+79996666666', 'Новосибирск'),
('Сергей', 'Васильев', 'sergey@mail.ru', '+79997777777', 'Санкт-Петербург'),
('Мария', 'Зайцева', 'maria@mail.ru', '+79998888888', 'Екатеринбург'),
('Алексей', 'Соловьёв', 'alexey@mail.ru', '+79999999999', 'Москва'),
('Татьяна', 'Павлова', 'tatyana@mail.ru', '+79990000000', 'Казань');


INSERT INTO products (name, category, price, stock_quantity) VALUES
```

('Смартфон X1', 'Электроника', 25000.00, 50),

('Ноутбук Pro', 'Электроника', 75000.00, 20),

('Наушники Wireless', 'Аксессуары', 5000.00, 100),

('Клавиатура механическая', 'Аксессуары', 8000.00, 40),

('Монитор 24"', 'Электроника', 20000.00, 30),

('Мышь оптическая', 'Аксессуары', 1500.00, 150),

('Планшет Mini', 'Электроника', 35000.00, 25),

('Веб-камера HD', 'Аксессуары', 4000.00, 60);

INSERT INTO orders (customer_id, status, total_amount) VALUES

(1, 'доставлен', 30000.00),

(2, 'в обработке', 75000.00),

(3, 'доставлен', 13000.00),

(1, 'новый', 5000.00),

(4, 'отправлен', 45000.00),

(5, 'доставлен', 20000.00),

(2, 'доставлен', 8000.00),

(6, 'новый', 35000.00),

(7, 'в обработке', 25000.00),

(3, 'отменён', 0.00);

INSERT INTO order_items (order_id, product_id, quantity, unit_price) VALUES

(1, 1, 1, 25000.00),

(1, 3, 1, 5000.00),

(2, 2, 1, 75000.00),

(3, 3, 1, 5000.00),

(3, 4, 1, 8000.00),

(4, 3, 1, 5000.00),

(5, 7, 1, 35000.00),

(5, 3, 2, 5000.00),

(6, 5, 1, 20000.00),

(7, 4, 1, 8000.00);

Этап 2. Простые SQL-запросы (15 мин)

4. Выведите всех клиентов из Москвы, отсортированных по фамилии:

   SELECT first_name, last_name, email

   FROM customers

   WHERE city = 'Москва'

   ORDER BY last_name;

5. Найдите товары категории «Электроника» с ценой выше 20 000 рублей:

   SELECT name, price

   FROM products

   WHERE category = 'Электроника' AND price > 20000;

6. Выведите 5 самых дорогих товаров:

   SELECT name, price

   FROM products

   ORDER BY price DESC

LIMIT 5;

Этап 3. Запросы с JOIN (20 мин)

7. Выведите список заказов с указанием ФИО клиента и суммы заказа:

```
SELECT
    o.id AS order_id,
    CONCAT(c.first_name, ' ', c.last_name) AS customer_name,
    o.total_amount,
    o.status
FROM orders o
INNER JOIN customers c ON o.customer_id = c.id;
```

8. Выведите полную информацию о позициях заказа: название товара, количество, цена, ФИО клиента:

```
SELECT
    CONCAT(c.first_name, ' ', c.last_name) AS customer_name,
    p.name AS product_name,
    oi.quantity,
    oi.unit_price,
    (oi.quantity * oi.unit_price) AS subtotal
FROM order_items oi
INNER JOIN orders o ON oi.order_id = o.id
INNER JOIN customers c ON o.customer_id = c.id
INNER JOIN products p ON oi.product_id = p.id;
```

9. Найдите клиентов, которые **никогда не делали заказов** (используйте LEFT

JOIN):

```
SELECT c.first_name, c.last_name, c.email
FROM customers c
LEFT JOIN orders o ON c.id = o.customer_id
WHERE o.id IS NULL;
```

Этап 4. Агрегатные функции и группировка (15 мин)

10.Подсчитайте количество заказов по каждому статусу:

```
SELECT status, COUNT(*) AS order_count
FROM orders
GROUP BY status;
```

11.Найдите общую сумму продаж по каждой категории товаров:

```
SELECT
  p.category,
  SUM(oi.quantity * oi.unit_price) AS total_sales
FROM order_items oi
INNER JOIN products p ON oi.product_id = p.id
GROUP BY p.category;
```

12.Определите клиентов, потративших более 30 000 рублей:

```
SELECT
    CONCAT(c.first_name, ' ', c.last_name) AS customer_name,
    SUM(o.total_amount) AS total_spent
FROM customers c
INNER JOIN orders o ON c.id = o.customer_id
```

```
    GROUP BY c.id
    HAVING total_spent > 30000;
```

13. Найдите средний чек по городам:

```
SELECT
  c.city,
  AVG(o.total_amount) AS avg_order_amount,
  COUNT(o.id) AS order_count
FROM customers c
INNER JOIN orders o ON c.id = o.customer_id
GROUP BY c.city;
```

Этап 5. Подзапросы (10 мин)

14. Найдите товары, цена которых выше средней цены по всем товарам:

```
SELECT name, price
FROM products
WHERE price > (SELECT AVG(price) FROM products);
```

15. Найдите клиентов, которые сделали более 2 заказов:

```
SELECT first_name, last_name
FROM customers
WHERE id IN (
  SELECT customer_id
  FROM orders
  GROUP BY customer_id
```

```
    HAVING COUNT(*) > 2
);
```

Этап 6. Анализ производительности и настройка индексов (15 мин)

16. Проанализируйте план выполнения запроса ДО создания индекса:

```
EXPLAIN SELECT * FROM customers WHERE email = 'ivan@mail.ru';
```

Зафиксируйте в отчёте значения type, possible_keys, key, rows.

17. Проанализируйте запрос с JOIN:

```
EXPLAIN SELECT c.first_name, o.total_amount
FROM customers c
INNER JOIN orders o ON c.id = o.customer_id
WHERE c.city = 'Москва';
```

18. Создайте индекс для поля email таблицы customers:

```
CREATE INDEX idx_customers_email ON customers(email);
```

19. Создайте составной индекс для таблицы orders:

```
CREATE INDEX idx_orders_customer_status ON orders(customer_id, status);
```

20. Создайте индекс для поля city:

```
CREATE INDEX idx_customers_city ON customers(city);
```

21. Повторно выполните EXPLAIN для тех же запросов из пунктов 16 и 17:

```
EXPLAIN SELECT * FROM customers WHERE email = 'ivan@mail.ru';
EXPLAIN SELECT c.first_name, o.total_amount
FROM customers c
INNER JOIN orders o ON c.id = o.customer_id
WHERE c.city = 'Москва';
```

Сравните значения type, key, rows с результатами до индексации.

22.Просмотрите все созданные индексы:

```
SHOW INDEX FROM customers;

SHOW INDEX FROM orders;
```

Этап 7. Удаление индексов и очистка (5 мин)

23.Удалите один из созданных индексов:

```
DROP INDEX idx_customers_city ON customers;
```

24.Убедитесь, что индекс удалён:

```
SHOW INDEX FROM customers;
```

Критерии оценки.

«5» «отлично»

- студент продемонстрировал готовность и способность к самостоятельному выполнению видов работ в стандартных ситуациях;
- все работы выполнил правильно и уверенно;
- проявил активность в освоении новых практических умений;
- содержание отчета по практическим работам полноценно отражает информацию по выполненным видам работ, установленных программой практики;
- оформление отчета по практическим работам полностью соответствует предъявляемым требованиям;
- при выполнении практических работ студент уверенно продемонстрировал обладание необходимыми компонентами общих и профессиональных компетенций;

«4» «хорошо»

- студент продемонстрировал готовность и способность к самостоятельному выполнению видов работ в стандартных ситуациях;
- все работы выполнил правильно и уверенно;
- содержание отчета по практическим работам в целом правильно отражает информацию по выполненным видам работ;
- оформление отчета по практическим работам соответствует предъявляемым требованиям, однако имеются замечания (некоторая неаккуратность, неполное описание проделанной работы и освоенных умений);
- при выполнении практических работ студент продемонстрировал обладание компонентами общих и профессиональных компетенций;
- характеристики студента положительные, имеются отдельные замечания и

рекомендации;

«3» «удовлетворительно»

- при выполнении видов работ студент нуждался во внешнем сопровождении и контроле;
- в отчете по практическим работам содержится описание видов выполняемых работ согласно программе дисциплины, однако, записи неполные и неточные;
- отчет по практическим работам оформлен с отклонениями от установленных требований;
- при выполнении практических работ студент продемонстрировал обладание отдельными компонентами общих и профессиональных компетенций;

«2» «неудовлетворительно»

– студент не справился с запланированными видами работ;
– при выполнении практических работ студент продемонстрировал слабое обладание отдельными компонентами общих и профессиональных компетенций;
– отчет студентом не подготовлен.

Практическое занятие №6. Реализация хранимых процедур и триггеров для автоматизации работы с базой данных.

Проверяемые результаты обучения:

- ОК 01 Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
- ОК 02 Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной деятельности
- ОК 05 Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
- ОК 09 Пользоваться профессиональной документацией на государственном и иностранном языках
- ПК 2.1 Выявлять проблемы, возникающие в процессе эксплуатации баз данных.
- ПК 2.2 Осуществлять процедуры администрирования баз данных.
- ПК 2.3 Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации.
- ПК 2.4 Формировать требования хранилищ банка данных для обучения.
- ПК 2.5 Подготавливать данные для базы знаний.

Цели работы

Тип цели	Формулировка
Дидактическая	Сформировать понимание назначения хранимых процедур и триггеров в СУБД MySQL, их типов, синтаксиса создания и областей применения для автоматизации бизнес-логики.
Практическая	Научиться создавать хранимые процедуры с параметрами, переменными, циклами и условиями; разрабатывать триггеры различных типов (BEFORE/AFTER, INSERT/UPDATE/DELETE); применять их для автоматизации типовых операций в базе данных.
Развивающая	Развить навыки проектирования серверной бизнес-логики, понимания событийной модели работы СУБД и оптимизации производительности за счёт переноса вычислений на сторону сервера БД.

Материально-техническое обеспечение

Компонент	Описание
Оборудование	ПК с доступом к локальной сети или интернету
ПО	Установленная СУБД MySQL (версии 8.0+), клиент для работы с БД (MySQL Workbench, DBeaver, phpMyAdmin или консольный клиент mysql)
Раздаточный материал	База данных online_store (созданная в предыдущих практических работах), шпаргалка по синтаксису CREATE PROCEDURE и CREATE TRIGGER

Краткие теоретические сведения

1. Хранимые процедуры (Stored Procedures)

Хранимая процедура — это именованный набор SQL-операторов, сохранённый в базе данных и выполняемый на стороне сервера СУБД.

Преимущества:

- Снижение сетевого трафика (один вызов вместо десятков запросов)
- Повторное использование кода
- Повышение безопасности (пользователь работает через процедуру, не имея прямого доступа к таблицам)
- Централизация бизнес-логики

Базовый синтаксис:

```
DELIMITER //

CREATE PROCEDURE procedure_name([IN|OUT|INOUT] param_name datatype)

BEGIN

   -- Тело процедуры

END //

DELIMITER ;
```

Типы параметров:

- IN — передаётся в процедуру (по умолчанию)
- OUT — возвращается из процедуры
- INOUT — и передаётся, и возвращается

Управляющие конструкции:

- IF ... THEN ... ELSEIF ... ELSE ... END IF;
- CASE ... WHEN ... THEN ... END CASE;
- WHILE ... DO ... END WHILE;
- LOOP ... END LOOP;

2. Триггеры (Triggers)

Триггер — это специальная хранимая процедура, которая автоматически выполняется при наступлении определённого события в таблице (INSERT, UPDATE, DELETE).

Типы триггеров:

Типы триггеров:

Тип	**Когда срабатывает**	**Назначение**
BEFORE INSERT	Перед вставкой	Валидация, модификация данных
AFTER INSERT	После вставки	Логирование, обновление связанных таблиц
BEFORE UPDATE	Перед обновлением	Проверка корректности изменений
AFTER UPDATE	После обновления	Аудит изменений, синхронизация
BEFORE DELETE	Перед удалением	Защита от удаления, архивирование
AFTER DELETE	После удаления	Очистка связанных данных, логирование

Специальные ссылки в триггерах:

- NEW.column_name — новое значение столбца (для INSERT/UPDATE)
- OLD.column_name — старое значение столбца (для UPDATE/DELETE)

Базовый синтаксис:

```
DELIMITER //

CREATE TRIGGER trigger_name

BEFORE|AFTER INSERT|UPDATE|DELETE

ON table_name

FOR EACH ROW

BEGIN

   -- Тело триггера
```

```
END //

DELIMITER ;
```

Ход работы

Этап 1. Подготовка тестовой среды (10 мин)

1. Откройте клиент MySQL и убедитесь, что база данных online_store существует:

```
USE online_store;

SHOW TABLES;
```

2. Если база данных отсутствует или в ней нет данных, создайте её и заполните тестовыми данными (используйте скрипт из практической работы №5).

3. Создайте дополнительную таблицу для хранения логов аудита:

```
CREATE TABLE IF NOT EXISTS audit_log (
    id INT AUTO_INCREMENT PRIMARY KEY,
    table_name VARCHAR(50) NOT NULL,
    operation VARCHAR(20) NOT NULL,
    record_id INT,
    old_value TEXT,
    new_value TEXT,
    changed_by VARCHAR(100) DEFAULT CURRENT_USER(),
    changed_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

Создайте таблицу для хранения бонусов клиентов

```
CREATE TABLE IF NOT EXISTS customer_bonus (
    customer_id INT PRIMARY KEY,
```

```
    bonus_points INT DEFAULT 0,
    total_spent DECIMAL(12, 2) DEFAULT 0.00,
    FOREIGN KEY (customer_id) REFERENCES customers(id) ON DELETE
CASCADE
);
```

Этап 2. Создание простых хранимых процедур (15 мин)

5. Создайте процедуру для получения информации о клиенте по его ID

```
DELIMITER //
CREATE PROCEDURE get_customer_info(IN p_customer_id INT)
BEGIN
    SELECT
        id,
        CONCAT(first_name, ' ', last_name) AS full_name,
        email,
        phone,
        city,
        registration_date
    FROM customers
    WHERE id = p_customer_id;
END //
DELIMITER ;
```

Вызовите созданную процедуру

```
CALL get_customer_info(1);
```

```
CALL get_customer_info(5);
```

Создайте процедуру для подсчёта количества заказов клиента

```
DELIMITER //
CREATE PROCEDURE get_customer_order_count(
    IN p_customer_id INT,
    OUT p_order_count INT
)
BEGIN
    SELECT COUNT(*) INTO p_order_count
    FROM orders
    WHERE customer_id = p_customer_id;
END //
DELIMITER ;
```

8. Вызовите процедуру с OUT-параметром:

```
CALL get_customer_order_count(1, @order_count);
SELECT @order_count AS 'Количество заказов клиента';
```

Этап 3. Процедуры с бизнес-логикой (20 мин)

9. Создайте процедуру для расчёта итоговой суммы заказа на основе его позиций:

```
DELIMITER //
CREATE PROCEDURE calculate_order_total(IN p_order_id INT)
BEGIN
    DECLARE v_total DECIMAL(12, 2);
```

```
    SELECT SUM(quantity * unit_price) INTO v_total
    FROM order_items
    WHERE order_id = p_order_id;

    UPDATE orders
    SET total_amount = COALESCE(v_total, 0.00)
    WHERE id = p_order_id;

    SELECT
        o.id AS order_id,
        o.total_amount,
        COUNT(oi.id) AS items_count
    FROM orders o
    LEFT JOIN order_items oi ON o.id = oi.order_id
    WHERE o.id = p_order_id
    GROUP BY o.id, o.total_amount;
END //
DELIMITER ;
```

10.Протестируйте процедуру

```
-- Добавим новую позицию в заказ
INSERT INTO order_items (order_id, product_id, quantity, unit_price)
VALUES (1, 2, 1, 75000.00);
```

```
-- Пересчитаем сумму заказа

CALL calculate_order_total(1);
```

11.Создайте процедуру для начисления бонусов клиенту (1% от суммы заказа)

```
DELIMITER //

  CREATE PROCEDURE award_customer_bonus(IN p_customer_id INT, IN
p_order_amount DECIMAL(12, 2))

  BEGIN

  DECLARE v_bonus INT;



  SET v_bonus = FLOOR(p_order_amount * 0.01);



  INSERT INTO customer_bonus (customer_id, bonus_points, total_spent)

  VALUES (p_customer_id, v_bonus, p_order_amount)

  ON DUPLICATE KEY UPDATE

     bonus_points = bonus_points + v_bonus,

     total_spent = total_spent + p_order_amount;



  SELECT

     cb.customer_id,

     CONCAT(c.first_name, ' ', c.last_name) AS customer_name,

     cb.bonus_points,

     cb.total_spent

  FROM customer_bonus cb
```

```
    INNER JOIN customers c ON cb.customer_id = c.id
    WHERE cb.customer_id = p_customer_id;
END //
DELIMITER ;
```

12. Протестируйте процедуру:

```sql
CALL award_customer_bonus(1, 30000.00);
CALL award_customer_bonus(1, 5000.00);
CALL award_customer_bonus(2, 75000.00);
```

Этап 4. Процедуры с циклами и условиями (15 мин)

13. Создайте процедуру для массового повышения цен на товары определённой категории:

```sql
DELIMITER //
CREATE PROCEDURE increase_prices_by_category(
    IN p_category VARCHAR(50),
    IN p_percent DECIMAL(5, 2)
)
```

```
BEGIN
    DECLARE v_done INT DEFAULT FALSE;
    DECLARE v_product_id INT;
    DECLARE v_old_price DECIMAL(10, 2);
    DECLARE v_new_price DECIMAL(10, 2);
    DECLARE v_updated_count INT DEFAULT 0;

    DECLARE cur CURSOR FOR
        SELECT id, price FROM products WHERE category = p_category;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET v_done = TRUE;

    OPEN cur;

    read_loop: LOOP
        FETCH cur INTO v_product_id, v_old_price;
        IF v_done THEN
            LEAVE read_loop;
        END IF;

        SET v_new_price = v_old_price * (1 + p_percent / 100);

        UPDATE products
        SET price = v_new_price
```

```
        WHERE id = v_product_id;

        SET v_updated_count = v_updated_count + 1;
    END LOOP;

    CLOSE cur;

    SELECT
        p_category AS category,
        v_updated_count AS products_updated,
        CONCAT('Цены повышены на ', p_percent, '%') AS message;
END //
DELIMITER ;
```

14. Протестируйте процедуру:

```sql
-- Повышаем цены на аксессуары на 10%
CALL increase_prices_by_category('Аксессуары', 10.00);

-- Проверяем результат
SELECT name, category, price FROM products WHERE category = 'Аксессуары';
```

15. Создайте процедуру с условной логикой для определения статуса клиента:

```sql
DELIMITER //
CREATE PROCEDURE get_customer_status(IN p_customer_id INT)
BEGIN
    DECLARE v_total_spent DECIMAL(12, 2);
    DECLARE v_order_count INT;
    DECLARE v_status VARCHAR(20);

    SELECT
        COALESCE(SUM(total_amount), 0),
        COUNT(*)
    INTO v_total_spent, v_order_count
    FROM orders
    WHERE customer_id = p_customer_id AND status != 'отменён';

    SET v_status = CASE
        WHEN v_total_spent >= 100000 THEN 'VIP'
        WHEN v_total_spent >= 50000 THEN 'Премиум'
        WHEN v_total_spent >= 10000 THEN 'Активный'
        WHEN v_order_count > 0 THEN 'Новичок'
        ELSE 'Потенциальный'
```

```
    END;

    SELECT
        p_customer_id AS customer_id,
        v_total_spent AS total_spent,
        v_order_count AS order_count,
        v_status AS customer_status;
END //
DELIMITER ;
```

16. Протестируйте процедуру для разных клиентов:

```sql
CALL get_customer_status(1);
CALL get_customer_status(2);
CALL get_customer_status(999); -- Несуществующий клиент
```

Этап 5. Создание триггеров для автоматизации (20 мин)

17. Создайте триггер для автоматического пересчёта суммы заказа при добавлении позиции:

```sql
DELIMITER //
CREATE TRIGGER trg_after_order_item_insert
AFTER INSERT ON order_items
FOR EACH ROW
BEGIN
    UPDATE orders
    SET total_amount = (
        SELECT COALESCE(SUM(quantity * unit_price), 0)
        FROM order_items
        WHERE order_id = NEW.order_id
    )
    WHERE id = NEW.order_id;
END //
DELIMITER ;
```

18. Создайте триггер для обновления суммы заказа при изменении позиции:

```sql
DELIMITER //
CREATE TRIGGER trg_after_order_item_update
AFTER UPDATE ON order_items
FOR EACH ROW
```

```
BEGIN
    UPDATE orders
    SET total_amount = (
        SELECT COALESCE(SUM(quantity * unit_price), 0)
        FROM order_items
        WHERE order_id = NEW.order_id
    )
    WHERE id = NEW.order_id;
END //
DELIMITER ;
```

19. Создайте триггер для автоматического уменьшения остатка на складе при создании заказа:

```sql
DELIMITER //
CREATE TRIGGER trg_before_order_item_insert
BEFORE INSERT ON order_items
FOR EACH ROW
BEGIN
    DECLARE v_stock INT;

    SELECT stock_quantity INTO v_stock
    FROM products
```

```
    WHERE id = NEW.product_id;

    IF v_stock < NEW.quantity THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Недостаточно товара на складе';
    END IF;

    UPDATE products
    SET stock_quantity = stock_quantity - NEW.quantity
    WHERE id = NEW.product_id;
END //
DELIMITER ;
```

20. Протестируйте триггеры:

```sql
-- Проверяем остаток товара
SELECT id, name, stock_quantity FROM products WHERE id = 3;

-- Добавляем позицию в заказ (остаток должен уменьшиться, сумма заказа — пересчитаться)
INSERT INTO order_items (order_id, product_id, quantity, unit_price)
VALUES (2, 3, 2, 5000.00);
```

```
-- Проверяем результат
SELECT id, name, stock_quantity FROM products WHERE id = 3;
SELECT id, total_amount FROM orders WHERE id = 2;

-- Пытаемся заказать больше, чем есть на складе (должна быть ошибка)
INSERT INTO order_items (order_id, product_id, quantity, unit_price)
VALUES (3, 3, 9999, 5000.00);
```

Этап 6. Триггеры для аудита и логирования (10 мин)

21. Создайте триггер для логирования изменений цен на товары:

```sql
DELIMITER //
CREATE TRIGGER trg_after_product_update
AFTER UPDATE ON products
FOR EACH ROW
BEGIN
    IF OLD.price != NEW.price THEN
        INSERT INTO audit_log (table_name, operation, record_id, old_value, new_value)
        VALUES (
```

```
            'products',
            'UPDATE',
            NEW.id,
            CONCAT('Цена: ', OLD.price),
            CONCAT('Цена: ', NEW.price)
        );
    END IF;
END //
DELIMITER ;
```

22. Создайте триггер для логирования удаления заказов:

```sql
DELIMITER //
CREATE TRIGGER trg_before_order_delete
BEFORE DELETE ON orders
FOR EACH ROW
BEGIN
    INSERT INTO audit_log (table_name, operation, record_id, old_value)
    VALUES (
        'orders',
        'DELETE',
        OLD.id,
```

CONCAT('Заказ №', OLD.id, ' на сумму ', OLD.total_amount, ' от клиента №', OLD.customer_id)

);

END //

DELIMITER ;

```

23. Протестируйте триггеры аудита:

```sql

-- Изменяем цену товара

UPDATE products SET price = 27000.00 WHERE id = 1;

UPDATE products SET price = 80000.00 WHERE id = 2;

-- Просматриваем лог изменений цен

SELECT * FROM audit_log WHERE table_name = 'products';

-- Создаём тестовый заказ и удаляем его

INSERT INTO orders (customer_id, status, total_amount, delivery_address)

VALUES (1, 'новый', 1000.00, 'Тестовый адрес');

SET @test_order_id = LAST_INSERT_ID();

DELETE FROM orders WHERE id = @test_order_id;

```
-- Просматриваем лог удалений
SELECT * FROM audit_log WHERE operation = 'DELETE';
```

Этап 7. Комплексный сценарий (10 мин)

24. Создайте процедуру, которая использует все разработанные компоненты для оформления заказа:

```sql
DELIMITER //
CREATE PROCEDURE place_order(
    IN p_customer_id INT,
    IN p_product_id INT,
    IN p_quantity INT,
    IN p_delivery_address TEXT
)
BEGIN
    DECLARE v_order_id INT;
    DECLARE v_unit_price DECIMAL(10, 2);
    DECLARE v_total DECIMAL(12, 2);

    -- Получаем цену товара
```

```
    SELECT price INTO v_unit_price
    FROM products
    WHERE id = p_product_id;

    -- Создаём заказ
    INSERT INTO orders (customer_id, status, delivery_address)
    VALUES (p_customer_id, 'новый', p_delivery_address);

    SET v_order_id = LAST_INSERT_ID();

    -- Добавляем позицию (сработают триггеры проверки остатка и пересчёта суммы)
    INSERT INTO order_items (order_id, product_id, quantity, unit_price)
    VALUES (v_order_id, p_product_id, p_quantity, v_unit_price);

    -- Получаем итоговую сумму
    SELECT total_amount INTO v_total
    FROM orders
    WHERE id = v_order_id;

    -- Начисляем бонусы
    CALL award_customer_bonus(p_customer_id, v_total);

    -- Возвращаем информацию о заказе
```

```
    SELECT
        o.id AS order_id,
        CONCAT(c.first_name, ' ', c.last_name) AS customer_name,
        p.name AS product_name,
        oi.quantity,
        oi.unit_price,
        o.total_amount,
        cb.bonus_points AS bonus_awarded
    FROM orders o
    INNER JOIN customers c ON o.customer_id = c.id
    INNER JOIN order_items oi ON o.id = oi.order_id
    INNER JOIN products p ON oi.product_id = p.id
    INNER JOIN customer_bonus cb ON c.id = cb.customer_id
    WHERE o.id = v_order_id;
END //
DELIMITER ;
```

25. Протестируйте комплексную процедуру:

```sql
CALL place_order(3, 4, 1, 'г. Казань, ул. Баумана, д. 5');
CALL place_order(1, 6, 3, 'г. Москва, ул. Тверская, д. 15');
```

Этап 8. Управление процедурами и триггерами (5 мин)

26. Просмотрите все созданные процедуры:

```
SHOW PROCEDURE STATUS WHERE Db = 'online_store';

SHOW CREATE PROCEDURE get_customer_info;
```

27. Просмотрите все созданные триггеры:

```
SHOW TRIGGERS FROM online_store;
```

28. Удалите один из триггеров (если необходимо):

```
DROP TRIGGER IF EXISTS trg_after_product_update;
```

29. Удалите процедуру (если необходимо):

```
DROP PROCEDURE IF EXISTS get_customer_info;
```

Критерии оценки.

«5» «отлично»

- студент продемонстрировал готовность и способность к самостоятельному выполнению видов работ в стандартных ситуациях;
- все работы выполнил правильно и уверенно;
- проявил активность в освоении новых практических умений;
- содержание отчета по практическим работам полноценно отражает информацию по выполненным видам работ, установленных программой практики;
- оформление отчета по практическим работам полностью соответствует предъявляемым требованиям;
- при выполнении практических работ студент уверенно продемонстрировал обладание необходимыми компонентами общих и профессиональных компетенций;

«4» «хорошо»

- студент продемонстрировал готовность и способность к самостоятельному выполнению видов работ в стандартных ситуациях;

- все работы выполнил правильно и уверенно;

- содержание отчета по практическим работам в целом правильно отражает информацию по выполненным видам работ;

- оформление отчета по практическим работам соответствует предъявляемым требованиям, однако имеются замечания (некоторая неаккуратность, неполное описание проделанной работы и освоенных умений);

- при выполнении практических работ студент продемонстрировал обладание компонентами общих и профессиональных компетенций;

- характеристики студента положительные, имеются отдельные замечания и рекомендации;

«3» «удовлетворительно»

- при выполнении видов работ студент нуждался во внешнем сопровождении и контроле;

- в отчете по практическим работам содержится описание видов выполняемых работ согласно программе дисциплины, однако, записи неполные и неточные;

- отчет по практическим работам оформлен с отклонениями от установленных требований;

- при выполнении практических работ студент продемонстрировал обладание отдельными компонентами общих и профессиональных компетенций;

«**2» «неудовлетворительно»**

– студент не справился с запланированными видами работ;

– при выполнении практических работ студент продемонстрировал слабое обладание отдельными компонентами общих и профессиональных компетенций;

– отчет студентом не подготовлен.

Практическое занятие №7. Резервное копирование и восстановление баз данных. Мониторинг производительности и состояния базы данных.

Проверяемые результаты обучения:

- ОК 01 Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
- ОК 02 Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной деятельности
- ОК 05 Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного

контекста
- ОК 09 Пользоваться профессиональной документацией на государственном и иностранном языках
- ПК 2.1 Выявлять проблемы, возникающие в процессе эксплуатации баз данных.
- ПК 2.2 Осуществлять процедуры администрирования баз данных.
- ПК 2.3 Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации.
- ПК 2.4 Формировать требования хранилищ банка данных для обучения.
- ПК 2.5 Подготавливать данные для базы знаний.

Цели работы

Тип цели	**Формулировка**
Дидактическая	Сформирование понимания стратегий резервного копирования (полное, дифференциальное, инкрементальное), механизмов восстановления БД, а также инструментов мониторинга производительности и состояния СУБД MySQL.
Практическая	Научиться создавать резервные копии с помощью утилиты mysqldump, восстанавливать базы данных из дампов, выполнять point-in-time recovery через binary log, а также анализировать производительность с помощью системных переменных, information_schema, performance_schema и slow query log.
Развивающая	Развить навыки администрирования СУБД, принятия обоснованных решений по выбору стратегии резервного копирования и оптимизации производительности на основе данных мониторинга.

Материально-техническое обеспечение

Компонент	**Описание**
Оборудование	ПК с доступом к локальной сети или интернету
ПО	Установленная СУБД MySQL (версии 8.0+), клиент для работы с БД (MySQL Workbench, DBeaver, phpMyAdmin или консольный клиент mysql), доступ к командной строке ОС
Раздаточный материал	База данных online_store (созданная в предыдущих практических работах), шпаргалка по утилите mysqldump и командам мониторинга

Краткие теоретические сведения

1. Типы резервного копирования

Тип	**Описание**	**Преимущества**	**Недостатки**
Полное (Full)	Копируются все данные БД целиком	Простое восстановление, полная целостность	Требует много времени и места
Дифференциальное (Differential)	Копируются данные, изменённые с	Быстрее полного, восстановление из	Растёт со временем

Тип	Описание	Преимущества	Недостатки
	момента последнего **полного** бэкапа	2 источников	
Инкрементальное (Incremental)	Копируются данные, изменённые с момента **любого предыдущего** бэкапа	Самое быстрое и компактное	Долгое восстановление (цепочка бэкапов)

2. Утилита mysqldump

Стандартная утилита MySQL для создания **логической** резервной копии в виде SQL-скрипта.

Основные флаги:

- --single-transaction — согласованный снимок для InnoDB без блокировки
- --routines — включить хранимые процедуры и функции
- --triggers — включить триггеры
- --events — включить события планировщика
- --all-databases — сделать дамп всех БД
- --databases db1 db2 — дамп нескольких БД
- --no-data — только структура (DDL) без данных

3. Binary Log (binlog)

Двоичный журнал, в котором MySQL фиксирует все изменения данных. Используется для:

- Репликации между серверами
- Point-in-time recovery (восстановление на конкретный момент времени)
- Инкрементального резервного копирования

4. Мониторинг производительности

- **SHOW PROCESSLIST** — список текущих подключений и выполняемых запросов
- **SHOW STATUS** — статистика работы сервера
- **information_schema** — метаданные БД (таблицы, индексы, процессы)
- **performance_schema** — детальная статистика производительности в реальном времени
- **Slow Query Log** — журнал медленных запросов
- **EXPLAIN ANALYZE** (MySQL 8.0.18+) — реальный план выполнения с замером времени

Ход работы

Этап 1. Подготовка тестовой среды (10 мин)

1. Откройте клиент MySQL и убедитесь, что база данных online_store существует и содержит данные:

```
USE online_store;
SHOW TABLES;
SELECT COUNT(*) AS total_customers FROM customers;
SELECT COUNT(*) AS total_orders FROM orders;
```

Проверьте, включён ли binary log на сервере

```
SHOW VARIABLES LIKE 'log_bin';
SHOW VARIABLES LIKE 'binlog_format';
SHOW BINARY LOGS;
```

💡 *Если binary log отключён, обратитесь к преподавателю для его включения в конфигурации my.cnf (параметры server-id=1, log-bin=mysql-bin, binlog_format=ROW).*

3. Проверьте статус медленных запросов:

```
SHOW VARIABLES LIKE 'slow_query_log';
SHOW VARIABLES LIKE 'long_query_time';
```

Этап 2. Создание резервных копий с помощью mysqldump (15 мин)

4. Откройте **командную строку** (не клиент MySQL!) и создайте полное резервное копирование одной базы данных:

```
mysqldump -u root -p --single-transaction --routines --triggers --events online_store > backup_online_store_full.sql
```

5.Создайте дамп только структуры (без данных)

```
mysqldump -u root -p --no-data online_store > backup_online_store_schema.sql
```

6. Создайте дамп всех баз данных сервера:

```
mysqldump -u root -p --single-transaction --routines --triggers --events --all-databases > backup_all_databases.sql
```

7.Создайте дамп отдельных таблиц

```
mysqldump -u root -p online_store customers orders > backup_tables.sql
```

8. Просмотрите содержимое созданного файла дампа:

```
head -n 50 backup_online_store_full.sql
```

Этап 3. Восстановление базы данных из дампа (15 мин)

9. **Сценарий аварии:** удалите таблицу order_items и часть данных из customers:

```
USE online_store;
DROP TABLE order_items;
```

```
DELETE FROM customers WHERE id > 5;
SELECT COUNT(*) FROM customers; -- Должно быть меньше, чем было
```

10.Восстановите базу данных из полного дампа через командную строку:

```
mysql -u root -p online_store < backup_online_store_full.sql
```

11.Проверьте результат восстановления

```
USE online_store;
SHOW TABLES;
SELECT COUNT(*) FROM customers;
SELECT COUNT(*) FROM order_items;
```

12.Альтернативный способ восстановления — через клиент MySQL:

```
SOURCE /path/to/backup_online_store_full.sql;
```

Этап 4. Мониторинг состояния БД (10 мин)

13.Просмотрите список активных подключений и выполняемых запросов:

```
SHOW PROCESSLIST;
SHOW FULL PROCESSLIST; -- С полными текстами запросов
```

Получите общую статистику работы сервера

```
SHOW GLOBAL STATUS;
SHOW GLOBAL STATUS LIKE 'Threads_%';
SHOW GLOBAL STATUS LIKE 'Questions';
SHOW GLOBAL STATUS LIKE 'Slow_queries';
SHOW GLOBAL STATUS LIKE 'Innodb_buffer_pool_%';
```

14.Проанализируйте ключевые метрики:

```
-- Количество подключений с начала работы сервера
SHOW GLOBAL STATUS LIKE 'Connections';

-- Количество прерванных подключений
SHOW GLOBAL STATUS LIKE 'Aborted_connects';

-- Размер InnoDB Buffer Pool
SHOW GLOBAL STATUS LIKE 'Innodb_buffer_pool_pages_total';
SHOW GLOBAL STATUS LIKE 'Innodb_buffer_pool_pages_free';
```

15.Посмотрите информацию о таблицах через information_schema

```
SELECT
   TABLE_NAME,
   TABLE_ROWS,
   ROUND((DATA_LENGTH + INDEX_LENGTH) / 1024 / 1024, 2) AS size_mb,
   ENGINE
FROM information_schema.TABLES
WHERE TABLE_SCHEMA = 'online_store';
```

Этап 5. Анализ медленных запросов (10 мин)

16.Включите slow query log (если не включён)

```
SET GLOBAL slow_query_log = 'ON';
SET GLOBAL long_query_time = 1; -- Запросы дольше 1 секунды
SHOW VARIABLES LIKE 'slow_query_log_file';
```

17.Выполните «тяжёлый» запрос для генерации записи в slow log

```
SELECT
   c.first_name,
   c.last_name,
   COUNT(o.id) AS order_count,
   SUM(o.total_amount) AS total_spent
FROM customers c
LEFT JOIN orders o ON c.id = o.customer_id
GROUP BY c.id
HAVING total_spent > 0;
```

18.Просмотрите slow query log через командную строку

```
tail -n 30 /var/lib/mysql/slow.log
```

19.Используйте EXPLAIN ANALYZE для детального анализа запроса

```
EXPLAIN ANALYZE
SELECT c.city, AVG(o.total_amount) AS avg_amount
FROM customers c
INNER JOIN orders o ON c.id = o.customer_id
GROUP BY c.city;
```

Этап 6. Мониторинг через performance_schema (10 мин)

20.Найдите 10 самых медленных запросов:

```
SELECT
   DIGEST_TEXT,
   COUNT_STAR AS exec_count,
   ROUND(AVG_TIMER_WAIT / 1000000000, 3) AS avg_time_sec,
   ROUND(SUM_TIMER_WAIT / 1000000000, 3) AS total_time_sec
FROM performance_schema.events_statements_summary_by_digest
ORDER BY AVG_TIMER_WAIT DESC
LIMIT 10;
```

21.Проанализируйте использование индексов

```
SELECT
   OBJECT_SCHEMA,
   OBJECT_NAME,
   INDEX_NAME,
   COUNT_STAR,
   COUNT_READ,
   COUNT_WRITE
FROM performance_schema.table_io_waits_summary_by_index_usage
WHERE OBJECT_SCHEMA = 'online_store'
ORDER BY COUNT_STAR DESC;
```

22.Найдите таблицы без используемых индексов

```
SELECT
   OBJECT_SCHEMA,
   OBJECT_NAME
FROM performance_schema.table_io_waits_summary_by_index_usage
WHERE INDEX_NAME IS NULL AND COUNT_STAR > 0;
```

23.Посмотрите статистику ожидания по типам событий

```
SELECT
   EVENT_NAME,
   COUNT_STAR,
   ROUND(SUM_TIMER_WAIT / 1000000000, 3) AS total_wait_sec
FROM performance_schema.events_waits_summary_global_by_event_name
WHERE COUNT_STAR > 0
ORDER BY SUM_TIMER_WAIT DESC
LIMIT 10;
```

Критерии оценки.

«5» «отлично»

- студент продемонстрировал готовность и способность к самостоятельному выполнению видов работ в стандартных ситуациях;
- все работы выполнил правильно и уверенно;
- проявил активность в освоении новых практических умений;
- содержание отчета по практическим работам полноценно отражает информацию по выполненным видам работ, установленных программой практики;
- оформление отчета по практическим работам полностью соответствует предъявляемым требованиям;
- при выполнении практических работ студент уверенно продемонстрировал обладание необходимыми компонентами общих и профессиональных компетенций;

«4» «хорошо»

- студент продемонстрировал готовность и способность к самостоятельному

выполнению видов работ в стандартных ситуациях;

- все работы выполнил правильно и уверенно;

- содержание отчета по практическим работам в целом правильно отражает информацию по выполненным видам работ;

- оформление отчета по практическим работам соответствует предъявляемым требованиям, однако имеются замечания (некоторая неаккуратность, неполное описание проделанной работы и освоенных умений);

- при выполнении практических работ студент продемонстрировал обладание компонентами общих и профессиональных компетенций;

- характеристики студента положительные, имеются отдельные замечания и рекомендации;

«3» «удовлетворительно»

- при выполнении видов работ студент нуждался во внешнем сопровождении и контроле;

- в отчете по практическим работам содержится описание видов выполняемых работ согласно программе дисциплины, однако, записи неполные и неточные;

- отчет по практическим работам оформлен с отклонениями от установленных требований;

- при выполнении практических работ студент продемонстрировал обладание отдельными компонентами общих и профессиональных компетенций;

«**2» «неудовлетворительно»**

– студент не справился с запланированными видами работ;

– при выполнении практических работ студент продемонстрировал слабое обладание отдельными компонентами общих и профессиональных компетенций;

– отчет студентом не подготовлен.

Практическое занятие №8. Реализация ролевой модели безопасности (создание ролей и управление их правами). Обеспечение безопасности данных в БД. Шифрование и хеширование данных.

Проверяемые результаты обучения:

- ОК 01 Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
- ОК 02 Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной деятельности
- ОК 05 Осуществлять устную и письменную коммуникацию на государственном

языке Российской Федерации с учетом особенностей социального и культурного контекста
- ОК 09 Пользоваться профессиональной документацией на государственном и иностранном языках
- ПК 2.1 Выявлять проблемы, возникающие в процессе эксплуатации баз данных.
- ПК 2.2 Осуществлять процедуры администрирования баз данных.
- ПК 2.3 Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации.
- ПК 2.4 Формировать требования хранилищ банка данных для обучения.
- ПК 2.5 Подготавливать данные для базы знаний.

Цели работы

Тип цели	Формулировка
Дидактическая	Сформировать понимание принципов ролевой модели безопасности в СУБД MySQL, механизмов управления доступом, а также методов защиты конфиденциальных данных с помощью шифрования и хеширования.
Практическая	Научиться создавать роли и назначать им привилегии, управлять пользователями и их правами, применять функции шифрования (AES_ENCRYPT/AES_DECRYPT) и хеширования (SHA2, MD5) для защиты персональных данных, реализовывать маскирование данных через представления.
Развивающая	Развить навыки проектирования безопасной архитектуры БД, понимания принципа минимальных привилегий и принятия обоснованных решений по защите информации в соответствии с требованиями 152-ФЗ.

Материально-техническое обеспечение

Компонент	Описание
Оборудование	ПК с доступом к локальной сети или интернету
ПО	Установленная СУБД MySQL (версии 8.0+), клиент для работы с БД (MySQL Workbench, DBeaver, phpMyAdmin или консольный клиент mysql)

Компонент	Описание
Раздаточный материал	База данных online_store (созданная в предыдущих практических работах), шпаргалка по командам управления доступом и функциям шифрования

Краткие теоретические сведения

1. Ролевая модель безопасности (MySQL 8.0+)

Роль — это именованный набор привилегий, который можно назначать пользователям. Это упрощает управление доступом: вместо назначения прав каждому пользователю отдельно, права назначаются роли, а роль — пользователям.

Основные команды:

```sql
CREATE ROLE 'role_name';                    -- Создание роли
GRANT privilege ON db.* TO 'role_name';     -- Назначение прав роли
GRANT 'role_name' TO 'user'@'host';         -- Назначение роли пользователю
SET DEFAULT ROLE 'role_name' TO 'user'@'host'; -- Активация роли по умолчанию
REVOKE 'role_name' FROM 'user'@'host';      -- Отзыв роли
DROP ROLE 'role_name';                      -- Удаление роли
```

2. Уровни привилегий в MySQL

Уровень	Пример	Описание
Глобальный	`GRANT ALL ON *.*`	Права на все БД сервера
Уровень БД	`GRANT SELECT ON online_store.*`	Права на все таблицы конкретной БД
Уровень таблицы	`GRANT INSERT ON online_store.orders`	Права на конкретную таблицу
Уровень столбца	`GRANT SELECT (email) ON online_store.customers`	Права на конкретный столбец

3. Принцип минимальных привилегий

Пользователь (или роль) должен получать только те права, которые необходимы для выполнения его задач, и не более того. Это ключевой принцип информационной безопасности.

4. Шифрование данных

Симметричное шифрование (AES):

- AES_ENCRYPT(str, key_str) — шифрует строку с использованием алгоритма AES и заданного ключа
- AES_DECRYPT(crypt_str, key_str) — расшифровывает данные
- Результат — бинарная строка, для хранения используется тип VARBINARY

Пример:

```
INSERT INTO customers (phone) VALUES (AES_ENCRYPT('+79991234567', 'secret_key'));
SELECT AES_DECRYPT(phone, 'secret_key') FROM customers;
```

5. Хеширование данных

Хеш-функции — необратимое преобразование данных. Используется для хранения паролей.

Функция	Длина хеша	Безопасность
`MD5(str)`	32 символа	⛔ Устарела, не рекомендуется
`SHA1(str)`	40 символов	⛔ Устарела
`SHA2(str, hash_length)`	224/256/384/512 бит	✅ Рекомендуется
`PASSWORD(str)`	—	Внутренний механизм MySQL

Пример:

```
INSERT INTO users (password_hash) VALUES (SHA2('MyStrongPassword123!', 256));
```

6. Маскирование данных через представления

Представления (VIEW) позволяют показывать пользователям только замаскированную часть данных, скрывая конфиденциальные поля:

```
CREATE VIEW v_customers_masked AS
SELECT
    id,
    first_name,
```

```
    last_name,
    CONCAT(LEFT(email, 2), '***@', SUBSTRING_INDEX(email, '@', -1)) AS
email_masked
FROM customers;
```

Ход работы

Этап 1. Подготовка тестовой среды (10 мин)

1. Откройте клиент MySQL под учётной записью root и убедитесь, что база данных online_store существует:

```
USE online_store;
SHOW TABLES;
```

2. Создайте таблицу для хранения учётных записей пользователей системы:

```
CREATE TABLE IF NOT EXISTS users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) NOT NULL UNIQUE,
    password_hash VARCHAR(255) NOT NULL,
    role VARCHAR(50) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

3. Добавьте в таблицу customers столбцы для хранения зашифрованных данных:

```
ALTER TABLE customers
ADD COLUMN phone_encrypted VARBINARY(255),
ADD COLUMN passport VARBINARY(255);
```

Этап 2. Создание ролей и назначение привилегий (15 мин)

4. Создайте роли для различных категорий пользователей системы:

```
-- Роль для менеджеров (чтение данных о клиентах и заказах)
CREATE ROLE 'role_manager';
GRANT SELECT ON online_store.customers TO 'role_manager';
```

```
GRANT SELECT ON online_store.orders TO 'role_manager';
GRANT SELECT ON online_store.order_items TO 'role_manager';
GRANT SELECT ON online_store.products TO 'role_manager';

-- Роль для операторов (чтение клиентов, создание заказов)
CREATE ROLE 'role_operator';
GRANT SELECT, INSERT, UPDATE ON online_store.orders TO 'role_operator';
GRANT SELECT, INSERT, UPDATE ON online_store.order_items TO
'role_operator';
GRANT SELECT ON online_store.customers TO 'role_operator';
GRANT SELECT ON online_store.products TO 'role_operator';

-- Роль для бухгалтеров (только финансовые данные)
CREATE ROLE 'role_accountant';
GRANT SELECT ON online_store.orders TO 'role_accountant';
GRANT SELECT (id, first_name, last_name, email) ON online_store.customers TO
'role_accountant';

-- Роль для аудиторов (только чтение логов)
CREATE ROLE 'role_auditor';
GRANT SELECT ON online_store.audit_log TO 'role_auditor';
```

5. Проверьте созданные роли:

```
SHOW GRANTS FOR 'role_manager';
SHOW GRANTS FOR 'role_operator';
```

Этап 3. Создание пользователей и назначение ролей (15 мин)

6. Создайте пользователей MySQL для разных ролей:

```
-- Менеджер
CREATE USER 'manager_ivanov'@'%' IDENTIFIED BY 'ManagerPass123!';
GRANT 'role_manager' TO 'manager_ivanov'@'%';
```

```
SET DEFAULT ROLE 'role_manager' TO 'manager_ivanov'@'%';

-- Оператор
CREATE USER 'operator_petrova'@'%' IDENTIFIED BY 'OperatorPass456!';
GRANT 'role_operator' TO 'operator_petrova'@'%';
SET DEFAULT ROLE 'role_operator' TO 'operator_petrova'@'%';

-- Бухгалтер
CREATE USER 'accountant_sidorova'@'%' IDENTIFIED BY
'AccountPass789!';
GRANT 'role_accountant' TO 'accountant_sidorova'@'%';
SET DEFAULT ROLE 'role_accountant' TO 'accountant_sidorova'@'%';

-- Аудитор
CREATE USER 'auditor_smirnov'@'%' IDENTIFIED BY 'AuditPass321!';
GRANT 'role_auditor' TO 'auditor_smirnov'@'%';
SET DEFAULT ROLE 'role_auditor' TO 'auditor_smirnov'@'%';
```

7. Проверьте назначенные роли:

```
SELECT * FROM mysql.role_edges;
SHOW GRANTS FOR 'manager_ivanov'@'%';
```

Этап 4. Проверка прав доступа (10 мин)

8. Откройте **новое соединение** под пользователем manager_ivanov (в MySQL Workbench или через консоль):

```
mysql -u manager_ivanov -p
```

9. Проверьте доступность операций:

```
-- Должно выполниться успешно
USE online_store;
SELECT * FROM customers LIMIT 5;
SELECT * FROM orders LIMIT 5;

-- Должно вызвать ошибку (нет прав на INSERT)
```

```
INSERT INTO customers (first_name, last_name, email)
VALUES ('Тест', 'Тестов', 'test@mail.ru');

-- Должно вызвать ошибку (нет доступа к таблице audit_log)
SELECT * FROM audit_log;
```

10. Откройте соединение под пользователем accountant_sidorova и проверьте:

```
-- Должно выполниться (но только определённые столбцы)
SELECT id, first_name, last_name, email FROM customers;

-- Должно вызвать ошибку (нет доступа к столбцу phone)
SELECT phone FROM customers;
```

Этап 5. Хеширование паролей пользователей (10 мин)

11. Добавьте пользователей в таблицу users с хешированными паролями:

```
INSERT INTO users (username, password_hash, role) VALUES
('ivanov', SHA2('ManagerPass123!', 256), 'manager'),
('petrova', SHA2('OperatorPass456!', 256), 'operator'),
('sidorova', SHA2('AccountPass789!', 256), 'accountant'),
('smirnov', SHA2('AuditPass321!', 256), 'auditor');
```

12. Проверьте, как хранятся пароли:

```
SELECT username, password_hash, role FROM users;
```

Обратите внимание: пароли хранятся в виде необратимых хешей. Даже администратор БД не может узнать исходный пароль.

13. Имитируйте проверку пароля при входе пользователя:

```
-- Пользователь ввёл правильный пароль
SELECT COUNT(*) AS auth_result
FROM users
WHERE username = 'ivanov'
  AND password_hash = SHA2('ManagerPass123!', 256);

-- Пользователь ввёл неправильный пароль
SELECT COUNT(*) AS auth_result
```

```
FROM users
WHERE username = 'ivanov'
  AND password_hash = SHA2('WrongPassword', 256);
```

14.Сравните разные алгоритмы хеширования:

```
SELECT
  'MyPassword123' AS plain_password,
  MD5('MyPassword123') AS md5_hash,
  SHA1('MyPassword123') AS sha1_hash,
  SHA2('MyPassword123', 256) AS sha256_hash,
  SHA2('MyPassword123', 512) AS sha512_hash;
```

Этап 6. Шифрование конфиденциальных данных (15 мин)

15.Зашифруйте телефон и паспортные данные клиентов с помощью AES:

```
-- Ключ шифрования (в реальности хранится в защищённом месте!)
SET @encryption_key = 'SuperSecretKey2026!';


-- Шифрование данных
UPDATE customers
SET
  phone_encrypted = AES_ENCRYPT(phone, @encryption_key),
  passport = AES_ENCRYPT(
    CONCAT('Серия 4500 Номер ', 100000 + id),
    @encryption_key
  )
WHERE id <= 5;
```

16.Проверьте, как данные хранятся в базе:

```
SELECT
  id,
  first_name,
  phone,
  phone_encrypted,
  HEX(phone_encrypted) AS encrypted_hex,
  passport
FROM customers
WHERE id <= 5;
```

Обратите внимание: в столбце phone_encrypted хранятся бинарные данные, которые невозможно прочитать без ключа.

17. Расшифруйте данные для просмотра:

```
SELECT
   id,
   first_name,
   CAST(AES_DECRYPT(phone_encrypted, @encryption_key) AS CHAR) AS
phone_decrypted,
   CAST(AES_DECRYPT(passport, @encryption_key) AS CHAR) AS
passport_decrypted
FROM customers
WHERE id <= 5;
```

18. Проверьте, что без правильного ключа расшифровка невозможна:

```
SELECT
   id,
   CAST(AES_DECRYPT(phone_encrypted, 'WrongKey') AS CHAR) AS
wrong_decrypt
FROM customers
WHERE id = 1;
```

Результат будет NULL или нечитаемые символы.

Этап 7. Маскирование данных через представления (10 мин)

19. Создайте представление с маскированными данными для операторов:

```
CREATE VIEW v_customers_masked AS
SELECT
   id,
   first_name,
   last_name,
   CONCAT(LEFT(email, 2), '***@', SUBSTRING_INDEX(email, '@', -1)) AS
```

```
    email_masked,
      CONCAT('+7 (', SUBSTRING(CAST(AES_DECRYPT(phone_encrypted,
    @encryption_key) AS CHAR), 3, 3),
          ') ***-**-',
          RIGHT(CAST(AES_DECRYPT(phone_encrypted, @encryption_key) AS
    CHAR), 2)) AS phone_masked,
      city
    FROM customers
    WHERE phone_encrypted IS NOT NULL;
```

20. Предоставьте операторам доступ к представлению:

```
    GRANT SELECT ON online_store.v_customers_masked TO 'role_operator';
```

21. Проверьте работу представления:

Данные отображаются в замаскированном виде: iv***@mail.ru, +7 (999) ***-**-67.

```
SELECT * FROM v_customers_masked LIMIT 5;
```

22. Создайте представление для аудита (только обезличенные данные):

```
CREATE VIEW v_audit_anonymized AS
SELECT
  id,
  table_name,
  operation,
  changed_at
FROM audit_log;
```

Этап 8. Управление правами и отзыв доступа (10 мин)

23. Создайте временного пользователя для тестирования

```
    CREATE USER 'temp_user'@'%' IDENTIFIED BY 'TempPass123!';
    GRANT SELECT ON online_store.products TO 'temp_user'@'%';
```

24. Проверьте права временного пользователя:

```
    SHOW GRANTS FOR 'temp_user'@'%';
```

25. Отозовите права и удалите пользователя:

```
REVOKE SELECT ON online_store.products FROM 'temp_user'@'%';
DROP USER 'temp_user'@'%';
```

26. Измените права существующего пользователя (повышение привилегий):

```
-- Дадём оператору дополнительные права на создание заказов
GRANT INSERT ON online_store.customers TO 'role_operator';
```

27. Просмотрите всех пользователей и их роли:

```
SELECT User, Host FROM mysql.user WHERE User NOT LIKE 'mysql.%' AND User
!= 'root';
SELECT * FROM mysql.role_edges;
```

Этап 9. Комплексный сценарий безопасности (5 мин)

28. Создайте процедуру для безопасной регистрации нового пользователя:

```
DELIMITER //
CREATE PROCEDURE register_user_secure(
    IN p_username VARCHAR(50),
    IN p_password VARCHAR(100),
    IN p_role VARCHAR(50)
)
BEGIN
    DECLARE v_password_hash VARCHAR(255);

    -- Проверка сложности пароля (минимум 8 символов)
    IF CHAR_LENGTH(p_password) < 8 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Пароль должен содержать минимум 8
символов';
    END IF;

    -- Хеширование пароля
    SET v_password_hash = SHA2(p_password, 256);

    -- Вставка пользователя
```

```
        INSERT INTO users (username, password_hash, role)
        VALUES (p_username, v_password_hash, p_role);

        SELECT
            'Пользователь успешно зарегистрирован' AS message,
            p_username AS username,
            p_role AS role;
    END //
    DELIMITER ;
```

29. Протестируйте процедуру:

```
    -- Успешная регистрация
    CALL register_user_secure('new_manager', 'SecurePass123!', 'manager');

    -- Ошибка: слишком короткий пароль
    CALL register_user_secure('bad_user', 'short', 'operator');
```

30. Создайте процедуру для безопасной аутентификации:

```
DELIMITER //
CREATE PROCEDURE authenticate_user(
    IN p_username VARCHAR(50),
    IN p_password VARCHAR(100)
)
BEGIN
    DECLARE v_user_count INT;

    SELECT COUNT(*) INTO v_user_count
    FROM users
    WHERE username = p_username
      AND password_hash = SHA2(p_password, 256);

    IF v_user_count = 1 THEN
        SELECT 'Успешная аутентификация' AS result, p_username AS username;
```

```
    ELSE
        SELECT 'Неверное имя пользователя или пароль' AS result;
    END IF;
END //
DELIMITER ;
```

31.Протестируйте аутентификацию:

```
CALL authenticate_user('ivanov', 'ManagerPass123!'); -- Успех
CALL authenticate_user('ivanov', 'WrongPassword');   -- Ошибка
```

Критерии оценки.

«5» «отлично»

- студент продемонстрировал готовность и способность к самостоятельному выполнению видов работ в стандартных ситуациях;
- все работы выполнил правильно и уверенно;
- проявил активность в освоении новых практических умений;
- содержание отчета по практическим работам полноценно отражает информацию по выполненным видам работ, установленных программой практики;
- оформление отчета по практическим работам полностью соответствует предъявляемым требованиям;
- при выполнении практических работ студент уверенно продемонстрировал обладание необходимыми компонентами общих и профессиональных компетенций;

«4» «хорошо»

- студент продемонстрировал готовность и способность к самостоятельному выполнению видов работ в стандартных ситуациях;
- все работы выполнил правильно и уверенно;
- содержание отчета по практическим работам в целом правильно отражает информацию по выполненным видам работ;
- оформление отчета по практическим работам соответствует предъявляемым требованиям, однако имеются замечания (некоторая неаккуратность, неполное описание проделанной работы и освоенных умений);
- при выполнении практических работ студент продемонстрировал обладание компонентами общих и профессиональных компетенций;
- характеристики студента положительные, имеются отдельные замечания и рекомендации;

«3» «удовлетворительно»

- при выполнении видов работ студент нуждался во внешнем сопровождении и контроле;

- в отчете по практическим работам содержится описание видов выполняемых работ согласно программе дисциплины, однако, записи неполные и неточные;

- отчет по практическим работам оформлен с отклонениями от установленных требований;

- при выполнении практических работ студент продемонстрировал обладание отдельными компонентами общих и профессиональных компетенций;

«**2» «неудовлетворительно»**

– студент не справился с запланированными видами работ;

– при выполнении практических работ студент продемонстрировал слабое обладание отдельными компонентами общих и профессиональных компетенций;

– отчет студентом не подготовлен.

Тема 3. Разработка и управление БД NoSQL

Проверяемые результаты обучения:

- ОК 01 Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
- ОК 02 Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной деятельности
- ОК 05 Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
- ОК 09 Пользоваться профессиональной документацией на государственном и иностранном языках
- ПК 2.8 Разрабатывать объекты баз данных NoSQL

Критерии оценки устного опроса

- оценка **«отлично»** выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов и смог ответить на уточняющие вопросы;
- оценка **«хорошо»** выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов, но не смог ответить на уточняющие вопросы;
- оценка **«удовлетворительно»** выставляется студенту, если он смог изложить суть основного вопроса, но не смог ответить по существу дополнительных вопросов;
- оценка **«неудовлетворительно»** выставляется студенту, если он не смог изложить суть основного вопроса.

Вопросы к устному опросу по теме:

Вопрос 1. Что изначально подразумевалось под аббревиатурой NoSQL при её появлении?

- А) No SQL (никакого SQL)
- Б) Not Only SQL (не только SQL) ✅
- В) New Operational SQL (новый операционный SQL)
- Г) Non-Object SQL (необъектный SQL)

Вопрос 2. Какая основная проблема традиционных реляционных СУБД (RDBMS) стимулировала развитие NoSQL?

- А) Невозможность выполнения сложных JOIN-запросов
- Б) Сложности с горизонтальным масштабированием и обработкой больших объёмов неструктурированных данных (Big Data) ✅
- В) Отсутствие поддержки транзакций ACID
- Г) Высокая стоимость лицензирования открытого ПО

Вопрос 3. Теорема CAP утверждает, что распределённая система может одновременно гарантировать только два из трёх свойств. Какие это свойства?

- А) Consistency (Согласованность), Availability (Доступность), Partition tolerance (Устойчивость к разделению) ✅
- Б) Concurrency (Параллелизм), Atomicity (Атомарность), Performance (Производительность)
- В) Consistency (Согласованность), Accuracy (Точность), Persistence (Сохраняемость)
- Г) Availability (Доступность), Portability (Портативность), Performance (Производительность)

Вопрос 4. Что означает принцип «Eventually Consistent» (согласованность в конечном счёте), часто используемый в NoSQL?

- А) Данные никогда не согласовываются между узлами
- Б) Система гарантирует мгновенную согласованность данных на всех узлах
- В) Система гарантирует, что при отсутствии новых обновлений все чтения в конечном итоге вернут последнее записанное значение ✅

- Г) Данные согласуются только при ручном запуске администратором

Вопрос 5. Какая модель данных представляет информацию в виде пар «ключ-значение», где значение может быть любой структурой данных?

- А) Документоориентированная
- Б) Графовая
- В) Ключ-значение (Key-Value) ✅
- Г) Колоночная (Wide-column)

Вопрос 6. В документоориентированных базах данных (например, MongoDB) данные чаще всего хранятся в формате:

- А) XML или HTML
- Б) JSON или BSON ✅
- В) CSV или TSV
- Г) Protobuf или Avro

Вопрос 7. Для решения каких задач лучше всего подходит графовая модель данных?

- А) Для хранения больших объёмов логов и метрик
- Б) Для анализа сложных связей и отношений между сущностями (социальные сети, рекомендательные системы) ✅
- В) Для хранения простых справочников с быстрым доступом по ключу
- Г) Для выполнения сложных аналитических агрегаций по всем столбцам таблицы

Вопрос 8. Что является основной единицей хранения в колоночных (wide-column) базах данных?

- А) Отдельная строка (row)
- Б) Семейство колонок (column family), где строки группируются по ключам, а колонки могут динамически добавляться ✅
- В) Отдельный документ в формате JSON
- Г) Узел (node) и ребро (edge)

Вопрос 9. Что понимается под термином «шардирование» (sharding) в архитектуре NoSQL?

- А) Создание резервных копий данных на другом сервере
- Б) Горизонтальное разделение данных и их распределение по нескольким серверам (узлам) ✅
- В) Вертикальное масштабирование путём добавления оперативной памяти на один сервер
- Г) Шифрование данных на уровне дискового массива

Вопрос 10. Какой тип масштабирования является основным и наиболее эффективным для большинства NoSQL СУБД?

- А) Вертикальное масштабирование (Scale-up)
- Б) Горизонтальное масштабирование (Scale-out) ✅
- В) Диагональное масштабирование
- Г) Масштабирование путём оптимизации SQL-запросов

Вопрос 11. Какую роль играет механизм репликации в распределённых NoSQL базах данных?

- А) Ускоряет выполнение сложных аналитических запросов с JOIN
- Б) Обеспечивает высокую доступность (High Availability) и отказоустойчивость за счёт хранения копий данных на нескольких узлах ✅
- В) Позволяет сжимать данные для экономии дискового пространства
- Г) Автоматически нормализует схему данных

Вопрос 12. Что такое «Gossip protocol» (протокол сплетен), часто используемый в NoSQL (например, в Cassandra)?

- А) Протокол для шифрования трафика между клиентом и сервером
- Б) Механизм децентрализованного обмена информацией о состоянии узлов в кластере ✅
- В) Протокол для резервного копирования данных в облако
- Г) Язык запросов для графовых баз данных

Вопрос 13. К какому типу NoSQL баз данных относится MongoDB?

- А) Ключ-значение
- Б) Документоориентированная ✅
- В) Графовая

- Г) Колоночная

Вопрос 14. Какая из перечисленных СУБД является классическим примером in-memory (в оперативной памяти) хранилища ключ-значение, часто используемого для кэширования?

- А) PostgreSQL
- Б) Redis ✅
- В) Neo4j
- Г) HBase

Вопрос 15. Apache Cassandra и ScyllaDB являются яркими представителями:

- А) Графовых баз данных
- Б) Ширококолоночных (wide-column) баз данных ✅
- В) Поисковых движков (Search Engines)
- Г) Объектно-ориентированных баз данных

Вопрос 16. Какая СУБД специально разработана для работы с графами и использует язык запросов Cypher?

- А) Elasticsearch
- Б) CouchDB
- В) Neo4j ✅
- Г) DynamoDB

Вопрос 17. Почему при проектировании схем в NoSQL (особенно в документоориентированных БД) часто применяется денормализация данных?

- А) Чтобы сэкономить дисковое пространство
- Б) Чтобы избежать дорогостоящих операций объединения (JOIN) и оптимизировать скорость чтения ✅
- В) Чтобы обеспечить строгую согласованность данных (ACID)
- Г) Чтобы упростить процесс миграции схемы при её изменении

Вопрос 18. В документоориентированной базе данных (например, MongoDB) когда целесообразно использовать паттерн «Встраивание» (Embedding) вместо «Ссылки» (Referencing)?

- А) Когда дочерние данные очень большие и часто обновляются независимо от родителя
- Б) Когда дочерние данные всегда запрашиваются вместе с родительскими и их размер невелик ✅
- В) Когда одни и те же дочерние данные должны быть связаны с множеством разных родителей
- Г) Когда требуется строгая ссылочная целостность на уровне СУБД

Вопрос 19. На что в первую очередь следует обращать внимание при выборе первичного ключа (Partition Key) в распределённых NoSQL базах (например, Cassandra или DynamoDB)?

- А) На то, чтобы ключ был как можно длиннее
- Б) На обеспечение равномерного распределения данных по узлам кластера во избежание «горячих точек» (hot spots) ✅
- В) На то, чтобы ключ обязательно был числовым и автоинкрементным
- Г) На возможность быстрого выполнения операций GROUP BY по этому ключу

Вопрос 20. Как в NoSQL базах данных обычно решается проблема эволюции схемы (изменения структуры данных со временем)?

- А) Схема жёстко зафиксирована, любое изменение требует остановки базы данных и выполнения DDL-скриптов
- Б) Изменения структуры обрабатываются на уровне приложения (code-first), новые поля просто добавляются в новые записи без блокировки старых ✅
- В) Эволюция схемы в NoSQL невозможна, требуется создание новой базы данных
- Г) Схема автоматически нормализуется фоновым процессом СУБД

Практические занятия к теме 3.

Практическое занятие №9. Базовые операции CRUD

Проверяемые результаты обучения:

- ОК 01 Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
- ОК 02 Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной деятельности
- ОК 05 Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
- ОК 09 Пользоваться профессиональной документацией на государственном и иностранном языках
- ПК 2.8 Разрабатывать объекты баз данных NoSQL

Цели работы

Тип цели	**Формулировка**
Дидактическая	Сформировать понимание документоориентированной модели данных MongoDB, принципов хранения информации в виде BSON-документов и коллекции, а также базовых операций CRUD (Create, Read, Update, Delete).
Практическая	Научиться выполнять операции вставки (insertOne, insertMany), чтения (find, findOne с операторами запросов), обновления (updateOne, updateMany с операторами $set, $push, $inc) и удаления (deleteOne, deleteMany) документов в MongoDB.
Развивающая	Развить навыки проектирования документной модели данных, понимания различий между реляционным и документоориентированным подходами, а также работы с вложенными структурами и массивами в MongoDB.

Материально-техническое обеспечение

Компонент	**Описание**
Оборудование	ПК с доступом к локальной сети или интернету
ПО	Установленная СУБД MongoDB (версии 6.0+), MongoDB Shell (mongosh) или графический клиент MongoDB Compass, опционально — драйвер для языка программирования (Python/Node.js)
Раздаточный материал	Шпаргалка по синтаксису CRUD-операций MongoDB, описание предметной области «Интернет-магазин»

Краткие теоретические сведения

1. Основные понятия MongoDB

Термин	**Описание**	**Аналог в RDBMS**
Database	База данных — контейнер для коллекций	База данных
Collection	Коллекция документов (не имеет жёсткой схемы)	Таблица
Document	Единица хранения данных в формате BSON (бинарный JSON)	Строка (row)
Field	Поле внутри документа (ключ-значение)	Столбец (column)
_id	Уникальный идентификатор документа (создаётся автоматически)	Primary Key

2. Операции CRUD

Операция	**Описание**	**Основные команды MongoDB**
Create	Вставка документов	insertOne(), insertMany()
Read	Чтение (поиск) документов	find(), findOne()
Update	Обновление документов	updateOne(), updateMany(), replaceOne()
Delete	Удаление документов	deleteOne(), deleteMany()

3. Операторы запросов

Оператор	**Назначение**	**Пример**
$eq, $ne	Равенство / неравенство	{age: {$eq: 25}}
$gt, $gte, $lt, $lte	Сравнение (>, >=, <, <=)	{price: {$gt: 1000}}
$in, $nin	Вхождение / невхождение в массив	{city: {$in: ['Москва', 'Казань']}}
$and, $or, $not	Логические операторы	{$or: [{age: {$lt: 18}}, {age: {$gt: 60}}]}
$regex	Регулярные выражения	{name: {$regex: '^И'}}
$exists	Проверка наличия поля	{phone: {$exists: true}}

4. Операторы обновления

Оператор	**Назначение**
$set	Устанавливает значение поля
$unset	Удаляет поле из документа
$inc	Увеличивает числовое поле на заданное значение
$push	Добавляет элемент в массив
$pull	Удаляет элемент из массива
$rename	Переименовывает поле
$min, $max	Обновляет поле, если новое значение меньше/больше текущего

Ход работы

Этап 1. Подключение к MongoDB и создание базы данных (10 мин)

1. Откройте MongoDB Shell (mongosh) или MongoDB Compass и подключитесь к локальному серверу:

bash

```
mongosh
```

2. Создайте базу данных и переключитесь на неё:

Javascript

```
use online_store
```

3. Проверьте текущую базу данных:

Javascript

```
db
```

4. Просмотрите список существующих коллекций (пока их нет):

javascript

```
show collections
```

Этап 2. Операции Create — вставка документов (15 мин)

5. Вставьте **один документ** в коллекцию customers:

Javascript

```
db.customers.insertOne({
   first_name: "Иван",
   last_name: "Петров",
   email: "ivan@mail.ru",
   phone: "+79991111111",
   city: "Москва",
   registration_date: new Date("2024-01-15"),
   is_active: true,
   addresses: [
      {type: "home", city: "Москва", street: "ул. Ленина, д. 10"},
      {type: "work", city: "Москва", street: "ул. Тверская, д. 5"}
   ]
})
```

Обратите внимание: MongoDB автоматически добавил поле _id типа ObjectId.

6. Вставьте **несколько документов** сразу:

Javascript

```
db.customers.insertMany([
  {
    first_name: "Анна", last_name: "Сидорова",
    email: "anna@mail.ru", phone: "+79992222222",
    city: "Санкт-Петербург", registration_date: new Date("2024-02-20"),
    is_active: true
  },
  {
    first_name: "Пётр", last_name: "Иванов",
    email: "petr@mail.ru", phone: "+79993333333",
    city: "Москва", registration_date: new Date("2024-03-10"),
    is_active: false
  },
  {
    first_name: "Ольга", last_name: "Смирнова",
    email: "olga@mail.ru", phone: "+79994444444",
    city: "Казань", registration_date: new Date("2024-04-05"),
    is_active: true
  },
  {
    first_name: "Дмитрий", last_name: "Кузнецов",
    email: "dmitry@mail.ru", phone: "+79995555555",
    city: "Новосибирск", registration_date: new Date("2024-05-12"),
    is_active: true
  },
  {
    first_name: "Елена", last_name: "Попова",
    email: "elena@mail.ru", phone: "+79996666666",
    city: "Москва", registration_date: new Date("2024-06-18"),
    is_active: true
  }
])
```

7. Создайте коллекцию products и заполните её товарами:

Javascript

```
db.products.insertMany([
  {
    name: "Смартфон X1",
    category: "Электроника",
    price: 25000,
    stock_quantity: 50,
    tags: ["новинка", "хит продаж"],
    specs: {screen: "6.5\"", ram: "8 GB", storage: "256 GB"}
```

```
  },
  {
    name: "Ноутбук Pro",
    category: "Электроника",
    price: 75000,
    stock_quantity: 20,
    tags: ["премиум"],
    specs: {screen: "15.6\"", ram: "16 GB", storage: "512 GB SSD"}
  },
  {
    name: "Наушники Wireless",
    category: "Аксессуары",
    price: 5000,
    stock_quantity: 100,
    tags: ["хит продаж", "беспроводные"],
    specs: {type: "over-ear", battery: "30 hours"}
  },
  {
    name: "Клавиатура механическая",
    category: "Аксессуары",
    price: 8000,
    stock_quantity: 40,
    tags: ["для геймеров"],
    specs: {switches: "Cherry MX Blue", layout: "RU/EN"}
  },
  {
    name: "Монитор 24\"",
    category: "Электроника",
    price: 20000,
    stock_quantity: 30,
    tags: ["IPS", "Full HD"],
    specs: {diagonal: "24\"", resolution: "1920x1080"}
  }
])
```

8. Проверьте количество вставленных документов:

```
javascript

db.customers.countDocuments()
db.products.countDocuments()
```

Этап 3. Операции Read — чтение и поиск документов (20 мин)

9. Найдите **всех клиентов** из коллекции:

```
Javascript

db.customers.find()
```

Для удобного отображения используйте .pretty() (в старых версиях) или просто вывод в mongosh.

10. Найдите **одного конкретного клиента** по email:

Javascript

```
db.customers.findOne({email: "ivan@mail.ru"})
```

11. Найдите клиентов из **Москвы** с проекцией (только нужные поля):

Javascript

```
db.customers.find(
  {city: "Москва"},
  {first_name: 1, last_name: 1, email: 1, _id: 0}
)
```

1 — включить поле, 0 — исключить поле.

12. Используйте **операторы сравнения** — найдите товары дороже 10 000 рублей:

Javascript

```
db.products.find(
  {price: {$gt: 10000}},
  {name: 1, price: 1, _id: 0}
)
```

13. Найдите товары в диапазоне цен от 5 000 до 25 000:

Javascript

```
db.products.find(
  {price: {$gte: 5000, $lte: 25000}},
  {name: 1, price: 1, _id: 0}
)
```

14. Используйте оператор $in — найдите клиентов из Москвы и Казани:

Javascript

```
db.customers.find(
  {city: {$in: ["Москва", "Казань"]}},
  {first_name: 1, last_name: 1, city: 1, _id: 0}
)
```

15. Используйте **логические операторы** — найдите активных клиентов из Москвы:

Javascript

```
db.customers.find(
   {$and: [{city: "Москва"}, {is_active: true}]},
   {first_name: 1, last_name: 1, city: 1, _id: 0}
)
```

16. Используйте $or — найдите товары категории «Электроника» ИЛИ дешевле 6 000:

Javascript

```
db.products.find(
   {$or: [{category: "Электроника"}, {price: {$lt: 6000}}]},
   {name: 1, category: 1, price: 1, _id: 0}
)
```

17. Используйте $regex — найдите товары, название которых начинается на «С»:

Javascript

```
db.products.find(
   {name: {$regex: "^С"}},
   {name: 1, _id: 0}
)
```

18. Поиск по **вложенным полям** (dot notation) — найдите товары с оперативной памятью 16 GB:

Javascript

```
db.products.find(
   {"specs.ram": "16 GB"},
   {name: 1, "specs.ram": 1, _id: 0}
)
```

19. Поиск по **массивам** — найдите товары с тегом «хит продаж»:

Javascript

```
db.products.find(
   {tags: "хит продаж"},
   {name: 1, tags: 1, _id: 0}
)
```

20. Используйте **сортировку и лимит** — топ-3 самых дорогих товара:

Javascript

```
db.products.find(
   {},
   {name: 1, price: 1, _id: 0}
).sort({price: -1}).limit(3)
```

-1 — по убыванию, 1 — по возрастанию.

21.Используйте $exists — найдите клиентов, у которых есть поле addresses:
javascript

```
db.customers.find(
   {addresses: {$exists: true}},
   {first_name: 1, last_name: 1, _id: 0}
)
```

Этап 4. Операции Update — обновление документов (20 мин)

22.Обновите **один документ** — измените город клиента:
Javascript

```
db.customers.updateOne(
   {email: "anna@mail.ru"},
   {$set: {city: "Екатеринбург"}}
)
```

23.Проверьте результат:
Javascript

```
db.customers.findOne({email: "anna@mail.ru"})
```

24.Используйте $inc — увеличьте цену товара на 10%:
Javascript

```
db.products.updateOne(
   {name: "Наушники Wireless"},
   {$inc: {price: 500}}
)
```

25.Используйте $push — добавьте новый тег к товару:
Javascript

```
db.products.updateOne(
   {name: "Ноутбук Pro"},
   {$push: {tags: "для работы"}}
)
```

26.Используйте $unset — удалите поле из документа:
Javascript

```
db.customers.updateOne(
   {email: "petr@mail.ru"},
   {$unset: {phone: ""}}
```

```
)
```

27. Используйте $rename — переименуйте поле:

Javascript

```
db.customers.updateOne(
   {email: "ivan@mail.ru"},
   {$rename: {"first_name": "name_first"}}
)
```

Верните обратно для дальнейшей работы:

Javascript

```
db.customers.updateOne(
   {email: "ivan@mail.ru"},
   {$rename: {"name_first": "first_name"}}
)
```

28. Обновите **несколько документов** сразу — деактивируйте всех неактивных клиентов из Москвы:

Javascript

```
db.customers.updateMany(
   {city: "Москва", is_active: false},
   {$set: {status: "заблокирован"}}
)
```

29. Используйте upsert (update + insert) — обновите или создайте документ:

Javascript

```
db.customers.updateOne(
   {email: "newclient@mail.ru"},
   {
      $set: {
         first_name: "Новый",
         last_name: "Клиент",
         city: "Самара",
         is_active: true
      }
   },
   {upsert: true}
)
```

Если клиент с таким email не найден, он будет создан.

30. Используйте $addToSet — добавляет элемент в массив только если его там ещё нет:

Javascript

```
db.products.updateOne(
   {name: "Смартфон X1"},
   {$addToSet: {tags: "новинка"}}  // не добавит, т.к. уже есть
)

db.products.updateOne(
   {name: "Смартфон X1"},
   {$addToSet: {tags: "5G"}}  // добавит
)
```

31. Используйте $pull — удалите элемент из массива:

javascript

```
db.products.updateOne(
   {name: "Смартфон X1"},
   {$pull: {tags: "5G"}}
)
```

Этап 5. Операции Delete — удаление документов (10 мин)

32. Удалите **один документ** по условию:

javascript

```
db.customers.deleteOne({email: "newclient@mail.ru"})
```

33. Удалите **все документы** по условию — все неактивные клиенты:

javascript

```
db.customers.deleteMany({is_active: false})
```

34. Проверьте, сколько документов осталось:

javascript

```
db.customers.countDocuments()
```

35. **Полная очистка коллекции** (удаление всех документов, но сама коллекция остаётся):

javascript

```
// Создадим временную коллекцию для демонстрации
db.temp_collection.insertMany([{a: 1}, {a: 2}, {a: 3}])
db.temp_collection.deleteMany({})
db.temp_collection.countDocuments()  // 0
```

36. **Удаление коллекции целиком** (вместе с индексами и метаданными):

javascript

```
db.temp_collection.drop()
show collections  // коллекции больше нет
```

Этап 6. Комплексные сценарии (10 мин)

37. Создайте коллекцию orders с вложенными документами:

Javascript

```
db.orders.insertMany([
  {
    customer_email: "ivan@mail.ru",
    order_date: new Date("2024-07-01"),
    status: "доставлен",
    items: [
      {product: "Смартфон X1", quantity: 1, price: 25000},
      {product: "Наушники Wireless", quantity: 1, price: 5000}
    ],
    delivery_address: {city: "Москва", street: "ул. Ленина, д. 10"},
    total_amount: 30000
  },
  {
    customer_email: "olga@mail.ru",
    order_date: new Date("2024-07-15"),
    status: "в обработке",
    items: [
      {product: "Ноутбук Pro", quantity: 1, price: 75000}
    ],
    delivery_address: {city: "Казань", street: "ул. Баумана, д. 5"},
    total_amount: 75000
  },
  {
    customer_email: "elena@mail.ru",
    order_date: new Date("2024-08-01"),
    status: "доставлен",
    items: [
      {product: "Монитор 24\"", quantity: 2, price: 20000}
    ],
    delivery_address: {city: "Москва", street: "ул. Тверская, д. 15"},
```

```
    total_amount: 40000
  }
])
```

38.Найдите заказы с суммой больше 35 000 рублей:
Javascript

```
db.orders.find(
  {total_amount: {$gt: 35000}},
  {customer_email: 1, total_amount: 1, status: 1, _id: 0}
)
```

39.Найдите заказы, содержащие товар «Наушники Wireless»:
Javascript

```
db.orders.find(
  {"items.product": "Наушники Wireless"},
  {customer_email: 1, items: 1, _id: 0}
)
```

40.Обновите статус заказа и добавьте трек-номер:
javascript

```
db.orders.updateOne(
  {customer_email: "olga@mail.ru", status: "в обработке"},
  {
    $set: {
      status: "отправлен",
      tracking_number: "RU123456789"
    }
  }
)
```

41.Добавьте новый товар в существующий заказ:
Javascript

```
db.orders.updateOne(
  {customer_email: "ivan@mail.ru"},
  {$push: {items: {product: "Клавиатура механическая", quantity: 1, price: 8000}}}
)
```

42.Используйте **агрегацию** для подсчёта количества заказов по статусам:
javascript

```
db.orders.aggregate([
   {$group: {_id: "$status", count: {$sum: 1}}},
   {$sort: {count: -1}}
])
```

Этап 7. Просмотр структуры и очистка (5 мин)

43.Просмотрите статистику по коллекции:

Javascript

```
db.customers.stats()
db.products.stats()
```

44.Просмотрите все коллекции в базе данных:

Javascript

```
show collections
```

45.Просмотрите все базы данных на сервере:

javascript

```
show dbs
```

Критерии оценки.

«5» «отлично»

- студент продемонстрировал готовность и способность к самостоятельному выполнению видов работ в стандартных ситуациях;
- все работы выполнил правильно и уверенно;
- проявил активность в освоении новых практических умений;
- содержание отчета по практическим работам полноценно отражает информацию по выполненным видам работ, установленных программой практики;
- оформление отчета по практическим работам полностью соответствует предъявляемым требованиям;
- при выполнении практических работ студент уверенно продемонстрировал обладание необходимыми компонентами общих и профессиональных компетенций;

«4» «хорошо»

- студент продемонстрировал готовность и способность к самостоятельному выполнению видов работ в стандартных ситуациях;
- все работы выполнил правильно и уверенно;
- содержание отчета по практическим работам в целом правильно отражает

информацию по выполненным видам работ;

- оформление отчета по практическим работам соответствует предъявляемым требованиям, однако имеются замечания (некоторая неаккуратность, неполное описание проделанной работы и освоенных умений);

- при выполнении практических работ студент продемонстрировал обладание компонентами общих и профессиональных компетенций;

- характеристики студента положительные, имеются отдельные замечания и рекомендации;

«3» «удовлетворительно»

- при выполнении видов работ студент нуждался во внешнем сопровождении и контроле;

- в отчете по практическим работам содержится описание видов выполняемых работ согласно программе дисциплины, однако, записи неполные и неточные;

- отчет по практическим работам оформлен с отклонениями от установленных требований;

- при выполнении практических работ студент продемонстрировал обладание отдельными компонентами общих и профессиональных компетенций;

«**2» «неудовлетворительно»**

– студент не справился с запланированными видами работ;
– при выполнении практических работ студент продемонстрировал слабое обладание отдельными компонентами общих и профессиональных компетенций;
– отчет студентом не подготовлен.

Практическое занятие №10. Продвинутые запросы и агрегация. Индексация и оптимизация производительности

Проверяемые результаты обучения:

- ОК 01 Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
- ОК 02 Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной деятельности
- ОК 05 Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
- ОК 09 Пользоваться профессиональной документацией на государственном и иностранном языках
- ПК 2.8 Разрабатывать объекты баз данных NoSQL

Цели работы

Тип цели	Формулировка
Дидактическая	Сформировать понимание принципов работы агрегационного конвейера MongoDB, механизмов индексации различных типов и подходов к оптимизации производительности запросов.
Практическая	Научиться строить сложные агрегационные пайплайны с использованием стадий $match, $group, $project, $unwind, $lookup, $addFields, $bucket; создавать различные типы индексов; анализировать производительность запросов с помощью .explain().
Развивающая	Развить навыки анализа плана выполнения запросов, принятия обоснованных решений по индексации и оптимизации производительности документоориентированной базы данных.

Материально-техническое обеспечение

Компонент	Описание
Оборудование	ПК с доступом к локальной сети или интернету
ПО	Установленная СУБД MongoDB (версии 6.0+), MongoDB Shell (mongosh) или графический клиент MongoDB Compass
Раздаточный материал	База данных online_store (созданная в предыдущей практической работе), шпаргалка по агрегационному конвейеру и типам индексов

Краткие теоретические сведения

1. Агрегационный конвейер (Aggregation Pipeline)
Агрегационный конвейер — это механизм обработки данных, при котором документы проходят через последовательность стадий (стадий-операций), каждая из которых трансформирует данные.

Основные стадии:

Стадия	Назначение
$match	Фильтрация документов (аналог WHERE)
$group	Группировка документов по ключу (аналог GROUP BY)
$project	Выборка и преобразование полей (аналог SELECT)
$sort	Сортировка документов
$limit / $skip	Ограничение и пропуск количества документов
$unwind	«Разворачивает» массив — создаёт отдельный документ для каждого элемента массива
$lookup	LEFT OUTER JOIN с другой коллекцией
$addFields	Добавление новых вычисляемых полей

Стадия	Назначение
$bucket	Группировка документов в «корзины» по диапазонам
$facet	Параллельное выполнение нескольких агрегаций
$count	Подсчёт количества документов

Аккумуляторы (используются в $group):

Аккумулятор	Назначение
$sum	Сумма значений
$avg	Среднее значение
$min / $max	Минимальное / максимальное значение
$first / $last	Первое / последнее значение
$push	Формирует массив значений
$addToSet	Формирует массив уникальных значений

2. Типы индексов в MongoDB

Тип индекса	Назначение	Пример
Single field	Индекс по одному полю	{email: 1}
Compound	Составной индекс по нескольким полям	{city: 1, registration_date: -1}
Multikey	Автоматически создаётся для полей-массивов	{tags: 1}
Text	Для полнотекстового поиска	{name: "text", description: "text"}
2dsphere	Для геопространственных запросов	{location: "2dsphere"}
TTL	Автоматическое удаление документов по истечении времени	{created_at: 1} с expireAfterSeconds
Unique	Гарантирует уникальность значений	{email: 1} с unique: true
Partial	Индексирует только документы, удовлетворяющие условию	partialFilterExpression: {status: "active"}
Hashed	Хеш-индекс для шардирования	{customer_id: "hashed"}

3. Оптимизация производительности

Правило ESR для составных индексов:

- **E**quality — поля с точным совпадением (первыми)
- **S**ort — поля сортировки
- **R**ange — поля с диапазонами (последними)

Covered query — запрос, полностью покрываемый индексом (не требует чтения документов из коллекции).

Метод .explain() — показывает план выполнения запроса:

- queryPlanner — какой индекс выбран
- executionStats — реальная статистика выполнения
- allPlansExecution — все рассмотренные планы

Ключевые метрики:

- totalDocsExamined — сколько документов просмотрено
- totalKeysExamined — сколько ключей индекса просмотрено
- executionTimeMillis — время выполнения
- indexOnly — true, если запрос покрыт индексом

Ход работы

Этап 1. Подготовка тестовой среды (10 мин)

1. Откройте MongoDB Shell и переключитесь на базу данных online_store:

   use online_store

2. Проверьте наличие коллекций и их содержимое:

javascript

```
show collections
db.customers.countDocuments()
db.products.countDocuments()
db.orders.countDocuments()
```

3. Если коллекции пусты, заполните их тестовыми данными (используйте скрипт из практической работы №9).

4. Добавьте дополнительные данные для агрегации — расширьте коллекцию заказов:

javascript

```
db.orders.insertMany([
  {
    customer_email: "dmitry@mail.ru",
    order_date: new Date("2024-08-10"),
    status: "доставлен",
    items: [
      {product: "Смартфон X1", quantity: 1, price: 25000, category: "Электроника"},
      {product: "Наушники Wireless", quantity: 2, price: 5000, category:
"Аксессуары"}
    ],
    delivery_address: {city: "Новосибирск", street: "ул. Красный проспект, д. 20"},
    total_amount: 35000,
```

```
    payment_method: "карта"
  },
  {
    customer_email: "anna@mail.ru",
    order_date: new Date("2024-08-15"),
    status: "доставлен",
    items: [
      {product: "Ноутбук Pro", quantity: 1, price: 75000, category: "Электроника"},
      {product: "Клавиатура механическая", quantity: 1, price: 8000, category:
"Аксессуары"}
    ],
    delivery_address: {city: "Екатеринбург", street: "ул. Мира, д. 7"},
    total_amount: 83000,
    payment_method: "наличные"
  },
  {
    customer_email: "ivan@mail.ru",
    order_date: new Date("2024-09-01"),
    status: "в обработке",
    items: [
      {product: "Монитор 24\"", quantity: 1, price: 20000, category: "Электроника"}
    ],
    delivery_address: {city: "Москва", street: "ул. Ленина, д. 10"},
    total_amount: 20000,
    payment_method: "карта"
  },
  {
    customer_email: "elena@mail.ru",
    order_date: new Date("2024-09-05"),
    status: "отправлен",
    items: [
      {product: "Смартфон X1", quantity: 2, price: 25000, category: "Электроника"},
      {product: "Наушники Wireless", quantity: 1, price: 5000, category:
"Аксессуары"}
    ],
    delivery_address: {city: "Москва", street: "ул. Тверская, д. 15"},
    total_amount: 55000,
    payment_method: "карта"
  }
])
```

Этап 2. Простые агрегации — $group и $match (15 мин)

5. Подсчитайте количество заказов по статусам:

javascript

```
db.orders.aggregate([
   {$group: {_id: "$status", count: {$sum: 1}}}
])
```

6. Найдите общую сумму продаж по городам доставки:

javascript

```
db.orders.aggregate([
   {$group: {
      _id: "$delivery_address.city",
      total_sales: {$sum: "$total_amount"},
      order_count: {$sum: 1}
   }},
   {$sort: {total_sales: -1}}
])
```

7. Найдите средний чек по способам оплаты:

javascript

```
db.orders.aggregate([
   {$group: {
      _id: "$payment_method",
      avg_amount: {$avg: "$total_amount"},
      max_amount: {$max: "$total_amount"},
      min_amount: {$min: "$total_amount"}
   }}
])
```

8. Используйте $match в начале пайплайна для фильтрации (это оптимизация!):

Javascript

```
db.orders.aggregate([
   {$match: {status: "доставлен"}},
   {$group: {
      _id: "$delivery_address.city",
      total_sales: {$sum: "$total_amount"},
      order_count: {$sum: 1}
   }},
   {$sort: {total_sales: -1}}
])
```

9. Найдите клиентов, потративших более 50 000 рублей:

Javascript

```
db.orders.aggregate([
  {$group: {
    _id: "$customer_email",
    total_spent: {$sum: "$total_amount"},
    order_count: {$sum: 1}
  }},
  {$match: {total_spent: {$gt: 50000}}},
  {$sort: {total_spent: -1}}
])
```

🔑 Обратите внимание: $match после $group работает как HAVING в SQL.

Этап 3. Работа с массивами — $unwind (15 мин)

10. «Разверните» массив items в заказах — каждый товар станет отдельным документом:

javascript

```
db.orders.aggregate([
  {$unwind: "$items"}
])
```

11. Найдите самые продаваемые товары:

javascript

```
db.orders.aggregate([
  {$unwind: "$items"},
  {$group: {
    _id: "$items.product",
    total_quantity: {$sum: "$items.quantity"},
    total_revenue: {$sum: {$multiply: ["$items.quantity", "$items.price"]}}
  }},
  {$sort: {total_quantity: -1}}
])
```

12. Найдите продажи по категориям товаров:

Javascript

```
db.orders.aggregate([
  {$unwind: "$items"},
  {$group: {
    _id: "$items.category",
```

```
      total_quantity: {$sum: "$items.quantity"},
      total_revenue: {$sum: {$multiply: ["$items.quantity", "$items.price"]}}
   }},
   {$sort: {total_revenue: -1}}
])
```

13.Используйте $unwind с опциями для обработки пустых массивов:
javascript

```
db.orders.aggregate([
   {$unwind: {
      path: "$items",
      preserveNullAndEmptyArrays: true
   }}
])
```

Этап 4. Связывание коллекций — $lookup (аналог JOIN) (15 мин)

14.Объедините заказы с информацией о клиентах (LEFT JOIN):
Javascript

```
db.orders.aggregate([
   {$lookup: {
      from: "customers",
      localField: "customer_email",
      foreignField: "email",
      as: "customer_info"
   }},
   {$unwind: "$customer_info"},
   {$project: {
      order_date: 1,
      total_amount: 1,
      customer_name: {
         $concat: ["$customer_info.first_name", " ", "$customer_info.last_name"]
      },
      customer_city: "$customer_info.city"
   }}
])
```

15.Найдите клиентов с их заказами (один клиент — массив заказов):
Javascript

```
db.customers.aggregate([
   {$lookup: {
      from: "orders",
      localField: "email",
      foreignField: "customer_email",
```

```
      as: "orders"
   }},
   {$project: {
      first_name: 1,
      last_name: 1,
      email: 1,
      orders_count: {$size: "$orders"},
      total_spent: {$sum: "$orders.total_amount"}
   }},
   {$sort: {total_spent: -1}}
])
```

16.Используйте $lookup с подзапросом (новый синтаксис MongoDB 3.6+):
Javascript

```
db.customers.aggregate([
   {$lookup: {
      from: "orders",
      let: {customer_email: "$email"},
      pipeline: [
         {$match: {
            $expr: {$eq: ["$customer_email", "$$customer_email"]},
            status: "доставлен"
         }},
         {$project: {order_date: 1, total_amount: 1}}
      ],
      as: "delivered_orders"
   }}
])
```

Этап 5. Продвинутые стадии агрегации (15 мин)

17.Используйте $addFields для добавления вычисляемых полей:
javascript

```
db.orders.aggregate([
   {$addFields: {
      year: {$year: "$order_date"},
      month: {$month: "$order_date"},
      discount_amount: {$multiply: ["$total_amount", 0.1]}
   }},
   {$project: {
      customer_email: 1,
      order_date: 1,
      total_amount: 1,
      year: 1,
      month: 1,
```

```
      discount_amount: 1
   }}
])
```

18.Используйте $bucket для группировки заказов по ценовым диапазонам:
Javascript

```
db.orders.aggregate([
   {$bucket: {
      groupBy: "$total_amount",
      boundaries: [0, 20000, 50000, 100000],
      default: "Другое",
      output: {
         count: {$sum: 1},
         orders: {$push: {
            id: "$_id",
            customer: "$customer_email",
            amount: "$total_amount"
         }}
      }
   }}
])
```

19.Используйте $facet для выполнения нескольких агрегаций одновременно:
Javascript

```
db.orders.aggregate([
   {$facet: {
      "by_status": [
         {$group: {_id: "$status", count: {$sum: 1}}}
      ],
      "by_payment": [
         {$group: {_id: "$payment_method", count: {$sum: 1}, total: {$sum:
"$total_amount"}}}
      ],
      "top_customers": [
         {$group: {_id: "$customer_email", total: {$sum: "$total_amount"}}},
         {$sort: {total: -1}},
         {$limit: 3}
      ]
   }}
])
```

20.Используйте $group с $push для формирования массивов:
javascript

```
db.orders.aggregate([
   {$group: {
      _id: "$customer_email",
      order_ids: {$push: "$_id"},
      amounts: {$push: "$total_amount"},
      total: {$sum: "$total_amount"}
   }}
])
```

21.Используйте $count для подсчёта:

javascript

```
db.orders.aggregate([
   {$match: {status: "доставлен"}},
   {$count: "delivered_orders_count"}
])
```

Этап 6. Создание индексов (15 мин)

22.Просмотрите существующие индексы:

javascript

```
db.customers.getIndexes()
db.orders.getIndexes()
```

23.Создайте single field index для поля email:

javascript

```
db.customers.createIndex({email: 1})
```

24.Создайте unique index для email:

Javascript

```
// Сначала удалим предыдущий индекс
db.customers.dropIndex("email_1")

// Создадим уникальный
db.customers.createIndex({email: 1}, {unique: true})
```

25.Проверьте работу уникального индекса:

javascript

```
// Должно вызвать ошибку дублирования
db.customers.insertOne({
   first_name: "Тест",
   last_name: "Тестов",
   email: "ivan@mail.ru",
```

```
  city: "Москва"
})
```

26.Создайте compound index для частого запроса:
javascript

```
db.orders.createIndex({status: 1, order_date: -1})
```

27.Создайте text index для полнотекстового поиска по товарам:
javascript

```
db.products.createIndex({name: "text", category: "text"})

// Используем текстовый поиск
db.prod

ucts.find({$text: {$search: "смартфон"}})
db.products.find({$text: {$search: "ноутбук премиум"}})
```

28.Создайте TTL index для автоматического удаления временных данных:
javascript

```
// Создадим коллекцию сессий
db.sessions.insertMany([
  {user_id: 1, token: "abc123", created_at: new Date()},
  {user_id: 2, token: "def456", created_at: new Date(Date.now() - 1000 * 60 * 60 * 24 *
8)} // 8 дней назад
])

// TTL индекс — документы удалятся через 7 дней (604800 секунд)
db.sessions.createIndex({created_at: 1}, {expireAfterSeconds: 604800})
```

29.Создайте partial index — индексируем только активные заказы:
javascript

```
db.orders.createIndex(
  {customer_email: 1, order_date: -1},
  {partialFilterExpression: {status: {$in: ["новый", "в обработке", "отправлен"]}}}
)
```

30.Создайте multikey index (автоматически для массивов):
javascript

```
db.products.createIndex({tags: 1})

// Проверим работу
```

```
db.products.find({tags: "хит продаж"})
```

Этап 7. Анализ производительности с .explain() (15 мин)

31.Проанализируйте запрос БЕЗ индекса — найдём заказы по email клиента:

javascript

```
// Сначала удалим индекс, если он есть
db.orders.dropIndex("customer_email_1_order_date_-1")

db.orders.find({customer_email: "ivan@mail.ru"}).explain("executionStats")
```

Обратите внимание на totalDocsExamined — должно быть равно количеству всех документов (COLLSCAN).

32.Создайте индекс и повторите анализ:

javascript

```
db.orders.createIndex({customer_email: 1})

db.orders.find({customer_email: "ivan@mail.ru"}).explain("executionStats")
```

Теперь totalDocsExamined должно быть равно количеству найденных документов, а totalKeysExamined — тоже мало.

33.Сравните запросы с сортировкой:

javascript

```
// Без индекса для сортировки
db.orders.find({status: "доставлен"})
  .sort({order_date: -1})
  .explain("executionStats")

// С составным индексом
db.orders.find({status: "доставлен"})
  .sort({order_date: -1})
  .explain("executionStats")
```

34.Проверьте covered query — запрос, полностью покрытый индексом:

javascript

```
// Создадим индекс, покрывающий запрос
db.customers.createIndex({city: 1, email: 1, first_name: 1})

// Запрос использует только поля из индекса
db.customers.find(
  {city: "Москва"},
  {email: 1, first_name: 1, _id: 0}
```

).explain("executionStats")

В выводе должно быть "indexOnly": true или "totalDocsExamined": 0.

35.Проанализируйте агрегационный запрос:
javascript

```
db.orders.explain("executionStats").aggregate([
   {$match: {status: "доставлен"}},
   {$group: {_id: "$customer_email", total: {$sum: "$total_amount"}}}
])
```

36.Используйте allPlansExecution для детального анализа:
javascript

```
db.orders.find({
   status: "доставлен",
   total_amount: {$gt: 30000}
}).explain("allPlansExecution")
```

Этап 8. Оптимизация запросов (10 мин)

37.Примените правило ESR для составного индекса:
javascript

```
// Частый запрос: фильтр по городу и дате, сортировка по дате
db.customers.createIndex({city: 1, registration_date: -1})

// Оптимальный запрос
db.customers.find({city: "Москва"})
   .sort({registration_date: -1})
   .explain("executionStats")
```

38.Используйте hint для принудительного выбора индекса:
javascript

```
1
```

39.Сравните производительность с $in и $or:
javascript

```
// Вариант 1: $in
db.orders.find({status: {$in: ["новый", "в обработке"]}})
   .explain("executionStats")

// Вариант 2: $or
db.orders.find({$or: [{status: "новый"}, {status: "в обработке"}]})
```

```
    .explain("executionStats")
```

40.Используйте projection для уменьшения объёма данных:

javascript

```
// Без проекции — возвращается весь документ
db.orders.find({status: "доставлен"}).explain("executionStats")

// С проекцией — только нужные поля
db.orders.find(
    {status: "доставлен"},
    {customer_email: 1, total_amount: 1, _id: 0}
).explain("executionStats")
```

41.Проанализируйте влияние .limit() на производительность:

javascript

```
db.orders.find({status: "доставлен"})
    .sort({total_amount: -1})
    .limit(10)
    .explain("executionStats")
```

Этап 9. Комплексный сценарий (5 мин)

42.Постройте сложный аналитический отчёт — топ-5 товаров по выручке с информацией о клиентах:

javascript

```
db.orders.aggregate([
    {$match: {status: "доставлен"}},
    {$unwind: "$items"},
    {$lookup: {
        from: "customers",
        localField: "customer_email",
        foreignField: "email",
        as: "customer"
    }},
    {$unwind: "$customer"},
    {$group: {
        _id: "$items.product",
        total_quantity: {$sum: "$items.quantity"},
        total_revenue: {$sum: {$multiply: ["$items.quantity", "$items.price"]}},
        unique_customers: {$addToSet: "$customer.email"},
        cities: {$addToSet: "$customer.city"}
    }},
    {$addFields: {
```

```
      customer_count: {$size: "$unique_customers"}
    }},
    {$sort: {total_revenue: -1}},
    {$limit: 5},
    {$project: {
      product: "$_id",
      total_quantity: 1,
      total_revenue: 1,
      customer_count: 1,
      cities: 1,
      _id: 0
    }}
])
```

43.Создайте индексы для оптимизации этого запроса:
javascript

```
db.orders.createIndex({status: 1})
db.customers.createIndex({email: 1})
```

44.Проанализируйте финальный запрос:
javascript

```
db.orders.explain("executionStats").aggregate([...]) // тот же пайплайн
```

Критерии оценки.

«5» «отлично»

- студент продемонстрировал готовность и способность к самостоятельному выполнению видов работ в стандартных ситуациях;
- все работы выполнил правильно и уверенно;
- проявил активность в освоении новых практических умений;
- содержание отчета по практическим работам полноценно отражает информацию по выполненным видам работ, установленных программой практики;
- оформление отчета по практическим работам полностью соответствует предъявляемым требованиям;
- при выполнении практических работ студент уверенно продемонстрировал обладание необходимыми компонентами общих и профессиональных компетенций;

«4» «хорошо»

- студент продемонстрировал готовность и способность к самостоятельному выполнению видов работ в стандартных ситуациях;
- все работы выполнил правильно и уверенно;
- содержание отчета по практическим работам в целом правильно отражает

информацию по выполненным видам работ;

- оформление отчета по практическим работам соответствует предъявляемым требованиям, однако имеются замечания (некоторая неаккуратность, неполное описание проделанной работы и освоенных умений);

- при выполнении практических работ студент продемонстрировал обладание компонентами общих и профессиональных компетенций;

- характеристики студента положительные, имеются отдельные замечания и рекомендации;

«3» «удовлетворительно»

- при выполнении видов работ студент нуждался во внешнем сопровождении и контроле;

- в отчете по практическим работам содержится описание видов выполняемых работ согласно программе дисциплины, однако, записи неполные и неточные;

- отчет по практическим работам оформлен с отклонениями от установленных требований;

- при выполнении практических работ студент продемонстрировал обладание отдельными компонентами общих и профессиональных компетенций;

«**2» «неудовлетворительно»**

– студент не справился с запланированными видами работ;

– при выполнении практических работ студент продемонстрировал слабое обладание отдельными компонентами общих и профессиональных компетенций;

– отчет студентом не подготовлен.

Тема 4. Векторные базы данных

Проверяемые результаты обучения:

- ОК 01 Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
- ОК 02 Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной деятельности
- ОК 05 Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
- ОК 09 Пользоваться профессиональной документацией на государственном и иностранном языках

Критерии оценки устного опроса

- оценка **«отлично»** выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов и смог ответить на уточняющие вопросы;

- оценка **«хорошо»** выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов, но не смог ответить на уточняющие вопросы;
- оценка **«удовлетворительно»** выставляется студенту, если он смог изложить суть основного вопроса, но не смог ответить по существу дополнительных вопросов;
- оценка **«неудовлетворительно»** выставляется студенту, если он не смог изложить суть основного вопроса.

Вопросы к устному опросу по теме:

Вопрос 1. Что такое большая языковая модель (LLM)?
А) База данных для хранения текстовых документов
Б) Нейросетевая модель, обученная на огромных объёмах текста для понимания и генерации естественного языка ✅
В) Операционная система для обработки естественного языка
Г) Язык программирования для работы с текстом
Вопрос 2. Какой параметр LLM контролирует «креативность» и разнообразие генерируемого текста?
А) max_tokens
Б) temperature ✅
В) top_k
Г) batch_size
Вопрос 3. Что такое «галлюцинация» в контексте работы LLM?
А) Ошибка компиляции кода модели
Б) Ситуация, когда модель генерирует уверенный, но фактически неверный или выдуманный ответ ✅
В) Процесс дообучения модели на новых данных
Г) Сбой в работе API модели
Вопрос 4. Какой из подходов позволяет LLM «помнить» контекст предыдущих сообщений в диалоге?
А) Использование постоянной базы данных
Б) Передача истории сообщений (role-based messages) в каждом запросе ✅
В) Автоматическое кеширование на стороне сервера
Г) Использование cookies браузера
Вопрос 5. Какая архитектура лежит в основе большинства современных LLM (GPT, LLaMA, Claude)?
А) Рекуррентные нейронные сети (RNN)
Б) Свёрточные нейронные сети (CNN)
В) Архитектура Transformer ✅
Г) Генеративно-состязательные сети (GAN)
Вопрос 6. Что такое «токен» в контексте работы с LLM?
А) Единица аутентификации пользователя в API
Б) Минимальная единица текста, на которую разбивается входная строка перед обработкой моделью ✅
В) Отдельный документ в векторной базе данных

Г) Ключ шифрования для защиты данных
Вопрос 7. Почему важно учитывать количество токенов при работе с LLM?
А) Токены определяют скорость интернета
Б) У каждой модели есть лимит контекстного окна, а стоимость API зависит от количества токенов ✅
В) Токены влияют на цвет текста в выводе
Г) Токены используются только для шифрования данных
Вопрос 8. Какая техника промпт-инжиниринга предполагает предоставление модели нескольких примеров «запрос-ответ» перед основным запросом?
А) Zero-shot prompting
Б) Few-shot prompting ✅
В) Chain-of-Thought
Г) Role prompting
Вопрос 9. Что такое RAG (Retrieval-Augmented Generation)?
А) Метод сжатия текста перед отправкой в LLM
Б) Подход, при котором LLM дополняется поиском релевантных документов из внешнего источника (например, векторной БД) перед генерацией ответа ✅
В) Алгоритм обучения нейронной сети
Г) Формат хранения данных в JSON
Вопрос 10. Какая техника промпт-инжиниринга побуждает модель рассуждать пошагово перед выдачей ответа?
А) Role prompting
Б) Chain-of-Thought (CoT) prompting ✅
В) Zero-shot prompting
Г) Temperature tuning
Вопрос 11. Что такое «эмбеддинг» (embedding) в контексте векторных баз данных?
А) Метод шифрования данных
Б) Числовое векторное представление семантического смысла текста, изображения или другого объекта ✅
В) Процесс сжатия базы данных
Г) Тип индекса в реляционной СУБД
Вопрос 12. Какая метрика чаще всего используется для измерения «похожести» между векторами в векторных БД?
А) Метрика Хэмминга
Б) Косинусное сходство (cosine similarity) ✅
В) Расстояние Левенштейна
Г) Коэффициент корреляции Пирсона
Вопрос 13. Какое главное отличие векторной базы данных от традиционной реляционной?
А) Векторная БД хранит только числа
Б) Векторная БД оптимизирована для поиска по семантической близости, а не по точному совпадению значений ✅
В) Векторная БД не поддерживает API
Г) Векторная БД работает только с изображениями
Вопрос 14. Какой алгоритм чаще всего используется для построения

приближённого поиска ближайших соседей (ANN) в векторных БД?
А) B-Tree
Б) HNSW (Hierarchical Navigable Small World) ✅
В) Hash-индекс
Г) R-Tree
Вопрос 15. Какая из перечисленных СУБД является специализированной векторной базой данных?
А) Microsoft Excel
Б) Pinecone ✅
В) Apache HTTP Server
Г) Git
Вопрос 16. Что такое «коллекция» в векторной базе данных?
А) Группа пользователей с одинаковыми правами
Б) Логический контейнер для хранения векторов и их метаданных, аналог таблицы в реляционной БД ✅
В) Набор API-ключей для доступа к базе
Г) Файл резервной копии
Вопрос 17. Какой параметр ОБЯЗАТЕЛЬНО должен быть указан при создании коллекции в векторной БД?
А) Имя администратора
Б) Размерность (dimension) векторов ✅
В) Цвет интерфейса
Г) Версия Python
Вопрос 18. Что такое «метаданные» вектора в векторной базе данных?
А) Служебная информация о самом векторе (например, источник, дата, категория), используемая для дополнительной фильтрации при поиске ✅
Б) Хеш-сумма вектора
В) Пароль для доступа к вектору
Г) Размер вектора в байтах
Вопрос 19. Какой тип индекса в векторной БД обеспечивает быстрый поиск по точному совпадению значений метаданных (например, по категории)?
А) Векторный HNSW-индекс
Б) Скалярный индекс (scalar index / metadata filter) ✅
В) Полнотекстовый индекс
Г) Геопространственный индекс
Вопрос 20. Что произойдёт при попытке добавить в коллекцию вектор размерностью 512, если коллекция создана для векторов размерностью 768?
А) Вектор будет автоматически дополнен нулями
Б) Вектор будет усечён до 512
В) Операция вызовет ошибку из-за несовпадения размерностей ✅
Г) Вектор будет сохранён в отдельной коллекции
Вопрос 21. Какой протокол чаще всего используется для взаимодействия с API современных векторных баз данных?
А) FTP
Б) HTTP/REST API ✅

В) SMTP
Г) SNMP
Вопрос 22. Что такое API-ключ при работе с векторной базой данных (например, Pinecone)?
А) Пароль от операционной системы
Б) Уникальный секретный токен для аутентификации запросов к сервису ✅
В) Ключ шифрования векторов
Г) Имя пользователя в БД
Вопрос 23. Какой HTTP-метод обычно используется для добавления новых векторов в коллекцию через REST API?
А) GET
Б) POST ✅
В) DELETE
Г) HEAD
Вопрос 24. Что такое «rate limit» при работе с API LLM и векторных БД?
А) Ограничение на размер загружаемого файла
Б) Ограничение на количество запросов за определённый период времени ✅
В) Максимальная длина текста в запросе
Г) Минимальная сумма оплаты за использование API
Вопрос 25. Какой формат данных чаще всего используется для обмена с API векторных баз данных?
А) XML
Б) CSV
В) JSON ✅
Г) YAML
Вопрос 26. Какая популярная Python-библиотека используется для работы с различными LLM и векторными базами данных через единый интерфейс?
А) NumPy
Б) LangChain ✅
В) Matplotlib
Г) Django
Вопрос 27. Какой Python-пакет является официальным клиентом для работы с OpenAI API?
А) requests
Б) openai ✅
В) tensorflow
Г) flask
Вопрос 28. Какая Python-библиотека предназначена для локального векторного поиска и часто используется как in-memory векторная БД?
А) FAISS (Facebook AI Similarity Search) ✅
Б) Pandas
В) Scikit-learn
Г) BeautifulSoup
Вопрос 29. Какой метод Python-клиента Pinecone используется для добавления векторов в индекс?

А) index.insert()
Б) index.upsert() ✅
В) index.push()
Г) index.add()
Вопрос 30. Какая функция OpenAI Python API используется для получения эмбеддингов текста?
А) openai.ChatCompletion.create()
Б) openai.Embedding.create() (или client.embeddings.create() в новой версии) ✅
В) openai.Completion.create()
Г) openai.Image.create()

3.3. Задания для оценки освоения МДК 02.03

Тема 1.1. Введение в конвейеры обработки данных для ИИ.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.9. Осуществлять конвейерную обработку данных для систем ИИ

Вопросы к устному опросу по теме:

1. Что такое конвейер обработки данных (Data Pipeline)?

Ответ: Конвейер обработки данных — это последовательность автоматизированных этапов, по которым данные проходят от источника до конечной точки использования. На этих этапах данные собираются, очищаются, преобразуются, проверяются и сохраняются в нужном формате.

2. Для чего нужны конвейеры обработки данных?

Ответ: Конвейеры нужны для автоматизации регулярной обработки данных, уменьшения ручного труда и ошибок, обеспечения своевременной доставки данных и их единообразного качества. Они позволяют повторять одну и ту же обработку многократно с одинаковым результатом.

3. Из каких основных этапов состоит конвейер обработки данных?

Ответ: Типовой конвейер включает сбор данных из источников, их загрузку (приём), очистку, преобразование, проверку качества, хранение и передачу потребителю, например модели машинного обучения или системе отчётности.

4. Какую роль играют конвейеры данных в жизненном цикле ИИ-систем?

Ответ: Конвейеры обеспечивают ИИ-систему данными на всех этапах: при подготовке обучающих выборок, обучении и переобучении моделей, а также при работе модели в эксплуатации. От качества конвейера напрямую зависит качество и надёжность модели.

5. Какие этапы выделяют в жизненном цикле ИИ-системы?

Ответ: Обычно выделяют постановку задачи, сбор и подготовку данных, разработку и обучение модели, оценку качества, развёртывание, мониторинг и сопровождение, включая переобучение модели на новых данных.

6. Почему подготовка данных считается одним из самых трудоёмких этапов создания ИИ?

Ответ: Исходные данные часто бывают неполными, разнородными, содержат ошибки, дубликаты и пропуски. Их очистка, разметка и приведение к единому формату требуют значительного времени, но без этого модель будет обучаться на некачественной информации.

7. Что такое источник данных? Приведите примеры.

Ответ: Источник данных — это система или ресурс, из которого конвейер получает исходную информацию. Примерами являются реляционные базы данных, файлы CSV и JSON, журналы (логи) приложений, веб-API, датчики IoT и пользовательские формы.

8. Что такое ETL?

Ответ: ETL (Extract, Transform, Load) — это архитектура обработки данных, при которой данные сначала извлекаются из источников, затем преобразуются на промежуточном сервере и только после этого загружаются в целевое хранилище.

9. Что такое ELT?

Ответ: ELT (Extract, Load, Transform) — это архитектура, при которой данные извлекаются и сразу загружаются в целевое хранилище в исходном виде, а преобразование выполняется уже внутри хранилища его вычислительными средствами.

10. В чём основное различие между ETL и ELT?

Ответ: Различие заключается в порядке этапов и месте преобразования данных. В ETL данные преобразуются до загрузки во внешнем инструменте, а в ELT — после загрузки внутри хранилища. ELT лучше подходит для больших объёмов данных и облачных хранилищ, ETL — когда в хранилище должны попадать только очищенные данные.

11. Что происходит на этапе извлечения (Extract)?

Ответ: На этапе извлечения данные считываются из одного или нескольких источников: баз данных, файлов, API, систем журналирования. Извлечение может быть полным (вся выборка целиком) или инкрементальным (только новые и изменённые записи).

12. Что происходит на этапе преобразования (Transform)?

Ответ: На этапе преобразования данные очищаются от ошибок и дубликатов, заполняются пропуски, приводятся к единым форматам и единицам измерения, объединяются из разных источников, агрегируются, а также формируются новые признаки для моделей ИИ.

13. Что происходит на этапе загрузки (Load)?

Ответ: На этапе загрузки подготовленные данные записываются в целевое хранилище: базу данных, хранилище данных, озеро данных или хранилище признаков. Загрузка может выполняться полностью или добавлением только новых данных.

14. Что такое хранилище данных (Data Warehouse)?

Ответ: Хранилище данных — это централизованная база, в которой хранятся структурированные, очищенные и согласованные данные из разных источников. Оно предназначено для аналитики, построения отчётов и подготовки данных для моделей.

15. Что такое озеро данных (Data Lake)?

Ответ: Озеро данных — это хранилище, в котором данные сохраняются в исходном виде в больших объёмах, включая структурированные, полуструктурированные и неструктурированные данные: таблицы, тексты, изображения, аудио. Структура данных определяется уже при их чтении.

16. Чем пакетная обработка данных отличается от потоковой?

Ответ: При пакетной обработке данные накапливаются и обрабатываются порциями по расписанию, например раз в сутки. При потоковой обработке данные обрабатываются непрерывно, по мере поступления, почти в реальном времени. Выбор зависит от того, насколько быстро нужен результат.

17. Что такое очистка данных?

Ответ: Очистка данных — это процесс выявления и исправления ошибок в данных: удаление дубликатов, обработка пропущенных значений, исправление некорректных записей, устранение выбросов и приведение данных к единому формату.

18. Что такое валидация данных в конвейере?

Ответ: Валидация данных — это автоматическая проверка того, что данные соответствуют ожидаемым правилам: нужному формату, типам, допустимым диапазонам значений, обязательности полей и структуре. Она позволяет остановить ошибочные данные до того, как они попадут в модель.

19. Что такое оркестрация конвейеров данных?

Ответ: Оркестрация — это управление запуском и порядком выполнения задач конвейера: планирование по расписанию, учёт зависимостей между этапами, повторный запуск при сбоях и контроль выполнения. Для этого используются оркестраторы, например Apache Airflow.

20. Что такое Apache Airflow и для чего он используется?

Ответ: Apache Airflow — это открытая платформа для разработки, планирования и мониторинга конвейеров данных. Конвейеры в ней описываются на языке Python в виде графов задач (DAG), где указываются этапы обработки и связи между ними.

21. Что такое DAG в контексте конвейеров данных?

Ответ: DAG (направленный ациклический граф) — это способ описания конвейера, в котором задачи являются узлами, а стрелки показывают порядок их выполнения. Граф не содержит циклов, поэтому выполнение всегда идёт от начальных задач к конечным.

22. Какие инструменты используются для потоковой передачи и обработки данных?

Ответ: Для потоковой передачи данных широко применяется Apache Kafka — распределённая платформа обмена сообщениями. Для обработки потоков используются Apache Spark Structured Streaming, Apache Flink и аналогичные системы.

23. Что такое Apache Spark и где он применяется?

Ответ: Apache Spark — это фреймворк для распределённой обработки больших объёмов данных на кластере компьютеров. Он применяется для пакетной и потоковой обработки, преобразования данных, аналитики и подготовки данных для машинного обучения.

24. Какие библиотеки Python применяются для обработки данных в конвейерах?

Ответ: Чаще всего используются pandas для работы с табличными данными, NumPy для числовых вычислений, Polars для быстрой обработки больших таблиц, а также scikit-learn, в котором есть механизм Pipeline для объединения этапов предобработки и обучения модели.

25. Что такое инструменты ELT-класса? Приведите примеры.

Ответ: Это средства, которые помогают загружать данные в хранилище и преобразовывать их внутри него. Например, Airbyte и Fivetran автоматизируют извлечение и загрузку данных из множества источников, а dbt позволяет описывать преобразования данных на SQL внутри хранилища.

26. Что такое хранилище признаков (Feature Store)?

Ответ: Хранилище признаков — это система для хранения, версионирования и повторного использования признаков, подготовленных для моделей машинного обучения. Оно обеспечивает одинаковые признаки при обучении модели и при её работе в эксплуатации.

27. Что такое версионирование данных и зачем оно нужно?

Ответ: Версионирование данных — это сохранение истории изменений наборов данных, аналогично версионированию программного кода. Оно позволяет воспроизвести обучение модели на тех же данных, сравнить результаты и откатиться к прежней версии. Для этого используется, например, DVC.

28. Что такое мониторинг конвейера данных?

Ответ: Мониторинг конвейера — это постоянное отслеживание его работы: успешности запусков, времени выполнения, объёма обработанных данных, ошибок и показателей качества данных. Он позволяет быстро обнаруживать сбои и изменения во входных данных.

29. Что такое дрейф данных и почему его важно отслеживать?

Ответ: Дрейф данных — это изменение статистических свойств входных данных со временем по сравнению с данными, на которых обучалась модель. Из-за дрейфа качество прогнозов модели снижается, поэтому его отслеживают и при необходимости переобучают модель.

30. Какие требования предъявляются к качественному конвейеру данных?

Ответ: Качественный конвейер должен быть надёжным и устойчивым к сбоям, масштабируемым, воспроизводимым, автоматизированным, контролируемым через мониторинг и журналирование, а также обеспечивать защиту данных и соответствие требованиям к их качеству.

Критерии оценки устного опроса:

- оценка «отлично» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов и смог ответить на уточняющие вопросы;
- оценка «хорошо» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов, но не смог ответить на уточняющие вопросы;
- оценка «удовлетворительно» выставляется студенту, если он смог изложить суть основного вопроса, но не смог ответить по существу дополнительных вопросов;
- оценка «неудовлетворительно» выставляется студенту, если он не смог изложить суть основного вопроса.

Тема 1.2. Источники данных и их классификация.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.9. Осуществлять конвейерную обработку данных для систем ИИ

Вопросы к устному опросу по теме:

1. Что такое источник данных?

Ответ: Источник данных — это система, устройство или ресурс, из которого поступает информация для обработки. Источниками данных могут быть базы данных, файлы, веб-сервисы, датчики, журналы работы приложений, социальные сети и действия пользователей.

2. По каким признакам классифицируют источники данных?

Ответ: Источники данных классифицируют по степени структурированности данных, по происхождению (внутренние и внешние), по способу поступления (пакетные и потоковые), по типу содержимого (числа, тексты, изображения, звук, видео) и по способу доступа (базы данных, файлы, API).

3. Чем внутренние источники данных отличаются от внешних?

Ответ: Внутренние источники находятся внутри организации: учётные системы, базы клиентов, журналы серверов, данные сотрудников. Внешние источники находятся за пределами организации: открытые наборы данных, сервисы сторонних компаний, государственные реестры, социальные сети.

4. Что такое структурированные данные?

Ответ: Структурированные данные — это данные, организованные по заранее заданной схеме в виде таблиц со строками и столбцами, где каждый столбец имеет определённый тип. Примерами являются таблицы реляционных баз данных и электронные таблицы.

5. Приведите примеры структурированных данных.

Ответ: К структурированным данным относятся таблицы с заказами клиентов, списки сотрудников, банковские транзакции, данные учёта товаров на складе, расписания и результаты измерений, записанные в табличном виде.

6. Что такое полуструктурированные данные?

Ответ: Полуструктурированные данные — это данные, которые не имеют жёсткой табличной схемы, но содержат метки, теги или ключи, описывающие их структуру. Такие данные могут иметь вложенность и разный набор полей у разных записей.

7. Приведите примеры полуструктурированных данных.

Ответ: К полуструктурированным данным относятся файлы JSON и XML, ответы веб-API, журналы (логи) приложений с полями, электронные письма с заголовками, документы в NoSQL-базах данных, например MongoDB.

8. Что такое неструктурированные данные?

Ответ: Неструктурированные данные — это данные, не имеющие заранее определённой схемы и внутренней разметки. Их нельзя напрямую представить в виде таблицы, и для анализа требуется специальная обработка.

9. Приведите примеры неструктурированных данных.

Ответ: К неструктурированным данным относятся тексты статей и сообщений, изображения, фотографии, аудиозаписи, видео, сканированные документы и записи телефонных разговоров.

10. В чём основные трудности обработки неструктурированных данных?

Ответ: Неструктурированные данные имеют большой объём, разнообразны по форме и не содержат явной структуры. Для их использования требуется извлекать признаки, например распознавать текст, объекты на изображениях или речь, что требует специальных методов и больших вычислительных ресурсов.

11. Что такое формат хранения данных?

Ответ: Формат хранения данных — это правило, по которому данные записываются в файл или передаются между системами. Формат определяет структуру записи, способ кодирования, возможность сжатия и скорость чтения и записи данных.

12. Что такое формат CSV?

Ответ: CSV (Comma-Separated Values) — это текстовый формат для хранения табличных данных, в котором каждая строка файла соответствует одной записи, а значения полей разделяются запятой или другим разделителем, например точкой с запятой.

13. Каковы достоинства и недостатки формата CSV?

Ответ: Достоинства CSV: простота, читаемость человеком, поддержка практически всеми программами. Недостатки: отсутствие сведений о типах данных, невозможность хранить вложенные структуры, большой размер файлов и медленное чтение при больших объёмах, проблемы с разделителями и кодировками.

14. Что такое формат JSON?

Ответ: JSON (JavaScript Object Notation) — это текстовый формат обмена данными, основанный на парах «ключ — значение». Он поддерживает объекты, массивы, строки, числа, логические значения и вложенные структуры.

15. Где применяется формат JSON?

Ответ: JSON широко применяется для обмена данными между веб-сервисами через API, хранения настроек программ, записи журналов, хранения документов в NoSQL-базах данных и описания разметки данных для машинного обучения.

16. Что такое формат Parquet?

Ответ: Parquet — это бинарный колоночный формат хранения данных, разработанный в рамках проекта Apache. В нём данные хранятся по столбцам, а не по строкам, вместе со схемой данных и с применением сжатия.

17. Каковы преимущества формата Parquet?

Ответ: Parquet занимает значительно меньше места за счёт сжатия, позволяет читать только нужные столбцы, хранит типы данных и схему, быстро обрабатывается аналитическими системами. Поэтому он широко используется в хранилищах и озёрах данных.

18. Что такое формат Avro?

Ответ: Avro — это бинарный строковый формат сериализации данных проекта Apache. Схема данных в Avro описывается в формате JSON и хранится вместе с данными, что позволяет корректно читать файлы разными программами.

19. Где применяется формат Avro?

Ответ: Avro применяется для передачи данных в потоковых системах, например Apache Kafka, для обмена сообщениями между сервисами и для записи больших потоков данных, когда важна быстрая запись и поддержка изменения схемы.

20. В чём разница между строковым и колоночным форматами хранения?

Ответ: В строковом формате данные одной записи хранятся рядом, поэтому он удобен для быстрой записи и чтения целых записей. В колоночном формате рядом хранятся значения одного столбца, поэтому он лучше подходит для аналитики и выборки отдельных полей. Avro и CSV являются строковыми форматами, Parquet — колоночным.

21. Что такое схема данных?

Ответ: Схема данных — это описание структуры данных: перечень полей, их названия, типы, обязательность и связи между ними. Схема позволяет проверять корректность данных и правильно их интерпретировать.

22. Что такое эволюция схемы данных?

Ответ: Эволюция схемы — это возможность изменять структуру данных со временем, например добавлять или удалять поля, без потери совместимости со старыми данными и программами. Хорошая поддержка эволюции схемы есть в формате Avro, частично — в Parquet.

23. Как выбрать формат хранения данных?

Ответ: Выбор формата зависит от типа и объёма данных, способа их использования и требований к скорости. CSV подходит для небольших таблиц и обмена данными, JSON — для вложенных данных и веб-сервисов, Parquet — для аналитики и больших наборов данных, Avro — для потоковой передачи и обмена сообщениями.

24. Какие особенности имеют данные для машинного обучения?

Ответ: Данные для машинного обучения должны быть репрезентативными, достаточными по объёму, качественными, размеченными (для обучения с учителем), сбалансированными и актуальными. Также они должны быть представлены в виде, пригодном для обработки моделью, то есть в виде числовых признаков.

25. Что такое признак (feature) и целевая переменная?

Ответ: Признак — это измеримое свойство объекта, которое используется моделью для обучения, например возраст клиента или цена товара. Целевая переменная — это значение, которое модель должна научиться предсказывать, например факт покупки или стоимость квартиры.

26. Что такое разметка данных?

Ответ: Разметка данных — это процесс присвоения данным правильных ответов (меток), необходимых для обучения модели с учителем. Например, указание категории для изображения, тональности для отзыва или выделение объектов на фотографии.

27. Что такое обучающая, валидационная и тестовая выборки?

Ответ: Обучающая выборка используется для обучения модели, валидационная — для подбора параметров и промежуточной оценки, тестовая — для итоговой проверки качества модели на данных, которые она не видела при обучении.

28. Что такое несбалансированность данных?

Ответ: Несбалансированность данных — это ситуация, когда объектов одного класса значительно больше, чем другого, например мошеннических операций намного меньше обычных. Это приводит к тому, что модель плохо распознаёт редкий класс, поэтому применяют специальные методы балансировки.

29. Какие проблемы качества данных встречаются чаще всего?

Ответ: Чаще всего встречаются пропущенные значения, дубликаты, ошибки ввода, выбросы, противоречивые записи, разные форматы одних и тех же данных, устаревшая информация и ошибки в разметке.

30. Как неструктурированные данные подготавливаются для машинного обучения?

Ответ: Неструктурированные данные преобразуются в числовой вид. Тексты разбиваются на слова или токены и переводятся в векторы, изображения приводятся к единому размеру и представляются в виде массивов пикселей, аудио преобразуется в спектрограммы. После этого данные могут использоваться моделью.

Критерии оценки устного опроса:

- оценка «отлично» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов и смог ответить на уточняющие вопросы;
- оценка «хорошо» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов, но не смог ответить на уточняющие вопросы;
- оценка «удовлетворительно» выставляется студенту, если он смог изложить суть основного вопроса, но не смог ответить по существу дополнительных вопросов;
- оценка «неудовлетворительно» выставляется студенту, если он не смог изложить суть основного вопроса.

Тема 1.3. Методы очистки и предобработки данных.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.9. Осуществлять конвейерную обработку данных для систем ИИ

Вопросы к устному опросу по теме:

1. Что такое предобработка данных?
Ответ: предобработка данных — это комплекс операций по подготовке исходных данных к

анализу и обучению моделей. Она включает очистку данных, обработку пропусков, дубликатов и выбросов, масштабирование числовых признаков и кодирование категориальных признаков.

2. Для чего необходима очистка и предобработка данных?
Ответ: предобработка необходима, потому что реальные данные содержат ошибки, пропуски и несогласованные значения. Модель, обученная на таких данных, даёт неточные прогнозы. Качественная подготовка данных повышает точность и надёжность модели.

3. Что такое пропущенные значения в данных?
Ответ: пропущенные значения — это отсутствующие данные в отдельных ячейках таблицы, когда для объекта не указано значение какого-либо признака. В программах они обычно обозначаются как nan, null, none или пустая ячейка.

4. Какие причины появления пропусков в данных вы знаете?
Ответ: пропуски возникают из-за ошибок при вводе данных, сбоев оборудования и датчиков, отказа пользователей заполнять отдельные поля, ошибок при объединении таблиц, потери данных при передаче, а также из-за того, что значение для объекта не существует.

5. Как выявить пропуски в данных?
Ответ: пропуски выявляют с помощью подсчёта пустых значений по каждому столбцу. В библиотеке pandas для этого используются методы isnull() или isna() вместе с sum(), а также метод info(), который показывает количество заполненных значений.

6. Какие существуют способы обработки пропусков?
Ответ: основные способы: удаление строк или столбцов с пропусками, заполнение пропусков средним значением, медианой или модой, заполнение константой, заполнение предыдущим или следующим значением, а также восстановление пропусков с помощью моделей, например метода ближайших соседей.

7. Когда допустимо удаление строк с пропусками?
Ответ: удаление строк допустимо, если пропусков немного (обычно несколько процентов), они распределены случайно и удаление не приведёт к существенной потере информации. При большом количестве пропусков удаление сильно сокращает выборку и может исказить данные.

8. Когда для заполнения пропусков лучше использовать медиану, а не среднее значение?
Ответ: медиану лучше использовать, когда в данных есть выбросы или распределение несимметрично. Среднее значение чувствительно к экстремальным значениям и может сильно сместиться, а медиана устойчива к выбросам.

9. Как заполняются пропуски в категориальных признаках?
Ответ: пропуски в категориальных признаках чаще всего заполняют модой, то есть наиболее часто встречающимся значением, или отдельной категорией, например «неизвестно». Второй способ сохраняет информацию о том, что значение отсутствовало.

10. Что такое дубликаты в данных?
Ответ: дубликаты — это повторяющиеся записи в наборе данных. Полные дубликаты полностью совпадают по всем полям, неполные совпадают по ключевым полям, но могут отличаться написанием, регистром букв или лишними пробелами.

11. Почему дубликаты необходимо удалять?
Ответ: дубликаты искажают статистические показатели, увеличивают вес отдельных объектов при обучении модели и могут привести к попаданию одних и тех же записей одновременно в обучающую и тестовую выборки, что завышает оценку качества модели.

12. Как выявить и удалить дубликаты в pandas?
Ответ: для выявления дубликатов используется метод duplicated(), который отмечает повторяющиеся строки, а для удаления — метод drop_duplicates(). С помощью параметра subset можно указать столбцы, по которым определяется повторение.

13. Что такое выбросы?
Ответ: выбросы — это значения, которые сильно отличаются от большинства остальных значений признака. Они могут быть результатом ошибок измерения или ввода, а могут отражать редкие, но реальные события.

14. Какие причины появления выбросов вы знаете?
Ответ: выбросы появляются из-за ошибок ввода данных (например, лишний ноль), сбоев датчиков, ошибок при переводе единиц измерения, а также из-за реальных редких событий, например очень крупной покупки или мошеннической операции.

15. Как выявить выбросы с помощью межквартильного размаха?
Ответ: вычисляется межквартильный размах iqr = q3 − q1, где q1 и q3 — первый и третий квартили. Выбросами считаются значения меньше q1 − 1,5·iqr или больше q3 + 1,5·iqr. Наглядно выбросы видны на диаграмме размаха (boxplot).

16. Как выявить выбросы с помощью z-оценки?
Ответ: z-оценка показывает, на сколько стандартных отклонений значение удалено от среднего: z = (x − среднее) / стандартное отклонение. Обычно выбросами считаются значения, у которых модуль z-оценки больше 3. Метод хорошо работает для данных с нормальным распределением.

17. Какие способы обработки выбросов существуют?
Ответ: выбросы можно удалить, заменить граничными значениями (ограничение или винзоризация), заменить медианой, преобразовать данные, например логарифмированием, или оставить без изменений, если они отражают реальные события и важны для задачи.

18. Что такое масштабирование признаков и зачем оно нужно?
Ответ: масштабирование — это приведение числовых признаков к сопоставимому диапазону значений. Оно нужно, потому что многие алгоритмы, например метод ближайших соседей, линейные модели и нейронные сети, чувствительны к масштабу, и признаки с большими значениями начинают преобладать над остальными.

19. Что такое нормализация данных?
Ответ: нормализация — это приведение значений признака к заданному диапазону, обычно от 0 до 1. Чаще всего используется формула min-max: x' = (x − min) / (max − min). В scikit-learn для этого применяется minmaxscaler.

20. Что такое стандартизация данных?
Ответ: стандартизация — это преобразование признака так, чтобы его среднее значение стало равным 0, а стандартное отклонение — 1. Используется формула x' = (x − среднее) / стандартное отклонение. В scikit-learn для этого применяется standardscaler.

21. В чём разница между нормализацией и стандартизацией?
Ответ: нормализация переводит значения в фиксированный диапазон от 0 до 1, но чувствительна к выбросам. Стандартизация не ограничивает диапазон, а центрирует данные и выравнивает их разброс, она меньше страдает от выбросов и подходит для алгоритмов, предполагающих нормальное распределение.

22. Что такое робастное масштабирование?
Ответ: робастное масштабирование — это способ масштабирования, при котором из значения

вычитается медиана, а результат делится на межквартильный размах. Такой способ устойчив к выбросам. В scikit-learn для этого применяется robustscaler.

23. Почему параметры масштабирования вычисляются только на обучающей выборке?
Ответ: если вычислять среднее, минимум или максимум по всем данным, то информация из тестовой выборки попадёт в процесс обучения. Это называется утечкой данных и приводит к завышенной оценке качества модели. Поэтому масштабировщик обучается на обучающей выборке, а затем применяется к тестовой.

24. Что такое категориальные признаки?
Ответ: категориальные признаки — это признаки, значения которых являются категориями, а не числами, например город, пол, цвет, марка автомобиля. Они бывают номинальными, когда категории не имеют порядка, и порядковыми, когда порядок есть, например уровень образования или размер одежды.

25. Почему категориальные признаки необходимо кодировать?
Ответ: большинство алгоритмов машинного обучения работают только с числами, поэтому текстовые категории нужно преобразовать в числовой вид. При этом важно не создать ложных зависимостей, например искусственного порядка между категориями, которого на самом деле нет.

26. Что такое метковое (label) кодирование?
Ответ: метковое кодирование — это замена каждой категории целым числом, например «москва» — 0, «казань» — 1, «омск» — 2. Метод простой, но создаёт искусственный порядок между категориями, поэтому для номинальных признаков в линейных моделях его использовать не рекомендуется.

27. Что такое порядковое (ordinal) кодирование?
Ответ: порядковое кодирование — это замена категорий числами с учётом их естественного порядка, например «начальное» — 1, «среднее» — 2, «высшее» — 3. Оно применяется для порядковых признаков, где порядок значений имеет смысл.

28. Что такое прямое (one-hot) кодирование?
Ответ: прямое кодирование — это создание для каждой категории отдельного двоичного столбца, в котором стоит 1, если объект относится к этой категории, и 0 в остальных случаях. Метод не создаёт ложного порядка, но при большом количестве категорий сильно увеличивает число столбцов. В pandas используется get_dummies(), в scikit-learn — onehotencoder.

29. Какие ещё способы кодирования категориальных признаков существуют?
Ответ: применяются частотное кодирование, при котором категория заменяется частотой её встречаемости, и целевое кодирование (target encoding), при котором категория заменяется средним значением целевой переменной для этой категории. Эти методы удобны для признаков с большим количеством категорий.

30. Как объединить этапы предобработки данных в единый конвейер?
Ответ: этапы предобработки объединяют с помощью класса pipeline библиотеки scikit-learn, а для разной обработки числовых и категориальных столбцов используют columntransformer. Это обеспечивает одинаковую обработку обучающих и новых данных, исключает утечку данных и упрощает повторное использование.

Критерии оценки устного опроса:

- Оценка «отлично» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов и смог ответить на уточняющие вопросы;

- Оценка «хорошо» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов, но не смог ответить на уточняющие вопросы;
- Оценка «удовлетворительно» выставляется студенту, если он смог изложить суть основного вопроса, но не смог ответить по существу дополнительных вопросов;
- Оценка «неудовлетворительно» выставляется студенту, если он не смог изложить суть основного вопроса.

Практическое занятие №1. Подготовка данных для ии-конвейера.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.9. Осуществлять конвейерную обработку данных для систем ИИ

Цель: научиться выполнять очистку и предобработку набора данных для использования в ии-конвейере: выявлять и обрабатывать пропуски, дубликаты и выбросы, масштабировать числовые признаки и кодировать категориальные признаки.

Задание:
1. Загрузить набор данных и изучить его структуру.
2. Определить типы признаков: числовые и категориальные.
3. Выявить и обработать пропущенные значения.
4. Выявить и удалить дубликаты.
5. Выявить и обработать выбросы.
6. Выполнить нормализацию или стандартизацию числовых признаков.
7. Выполнить кодирование категориальных признаков.
8. Сделать вывод о готовности данных к использованию в ии-конвейере.

Этапы выполнения:
Этап 1. Загрузка и анализ данных.
На первом этапе набор данных загружается в среду разработки, например с помощью библиотеки pandas. Изучаются количество строк и столбцов, названия и типы признаков, основные статистические характеристики, определяется целевая переменная.

Этап 2. Определение типов признаков.
Признаки разделяются на числовые и категориальные. Для категориальных признаков определяется, являются ли они номинальными или порядковыми. Проверяется корректность типов данных, при необходимости типы исправляются.

Этап 3. Обработка пропусков.
Подсчитывается количество пропусков в каждом столбце. Для каждого признака выбирается способ обработки: удаление, заполнение средним, медианой, модой или отдельной категорией. Выбор обосновывается.

Этап 4. Обработка дубликатов.
Выявляются полные и неполные дубликаты. Перед поиском неполных дубликатов текстовые значения приводятся к единому виду: удаляются лишние пробелы, приводится регистр букв.

Дубликаты удаляются.

Этап 5. Обработка выбросов.
Для числовых признаков выбросы выявляются с помощью межквартильного размаха или z-оценки и диаграмм размаха. Определяется, являются ли выбросы ошибками или реальными значениями, и выбирается способ их обработки.

Этап 6. Масштабирование числовых признаков.
Данные разделяются на обучающую и тестовую выборки. Масштабировщик обучается на обучающей выборке и применяется к обеим выборкам. Выбирается метод: нормализация, стандартизация или робастное масштабирование.

Этап 7. Кодирование категориальных признаков.
Номинальные признаки кодируются прямым (one-hot) кодированием, порядковые — порядковым кодированием. Проверяется, что все признаки после кодирования имеют числовой вид.

Этап 8. Оформление результата.
Проверяется итоговый набор данных: отсутствие пропусков и дубликатов, числовой вид всех признаков. Подготовленные данные сохраняются в файл, например в формате csv или parquet. Оформляется отчёт с описанием выполненных действий и выводами.

Пример решения:
Задание: подготовить набор данных о клиентах банка для модели прогнозирования оттока клиентов.
Дано: таблица содержит 1000 записей и признаки: возраст, город, доход, уровень образования, количество продуктов банка, признак ухода клиента (целевая переменная). В данных есть пропуски в столбцах «возраст» и «город», повторяющиеся записи и подозрительные значения дохода.

Решение по этапам:
1. Загружаем данные с помощью pandas и изучаем их методами info() и describe().
2. Определяем числовые признаки (возраст, доход, количество продуктов) и категориальные (город — номинальный, уровень образования — порядковый).
3. С помощью isnull().sum() находим пропуски: 40 в столбце «возраст» и 25 в столбце «город». Пропуски в возрасте заполняем медианой, в городе — значением «неизвестно».
4. С помощью duplicated() находим 15 повторяющихся записей и удаляем их методом drop_duplicates().
5. По межквартильному размаху выявляем в доходе значения выше верхней границы. Значения с явной ошибкой ввода заменяем верхней границей.
6. Делим данные на обучающую и тестовую выборки в соотношении 80 к 20. Числовые признаки стандартизируем с помощью standardscaler, обученного на обучающей выборке.
7. Признак «город» кодируем прямым кодированием (onehotencoder), признак «уровень образования» — порядковым кодированием (ordinalencoder).
8. Объединяем этапы в pipeline с columntransformer для повторного использования.
9. Проверяем, что в данных нет пропусков и дубликатов, а все признаки числовые.
10. Сохраняем подготовленные данные и оформляем отчёт.

Ответ: данные подготовлены для ии-конвейера: пропуски заполнены медианой и отдельной категорией, дубликаты удалены, выбросы в доходе ограничены, числовые признаки стандартизированы, категориальные признаки закодированы прямым и порядковым кодированием. Этапы предобработки объединены в единый конвейер, данные готовы к обучению модели.

Критерии оценки практических заданий:

1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Тема 1.4. Формирование датасетов для машинного обучения.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.9. Осуществлять конвейерную обработку данных для систем ИИ

Вопросы к устному опросу по теме:

1. Что такое датасет для машинного обучения?
Ответ: датасет — это структурированный набор данных, подготовленный для обучения, настройки и проверки модели машинного обучения. Он состоит из объектов, описанных набором признаков, и, как правило, содержит значения целевой переменной.

2. Какие требования предъявляются к датасету для машинного обучения?
Ответ: датасет должен быть репрезентативным, то есть отражать реальные данные, с которыми будет работать модель, достаточным по объёму, очищенным от ошибок, корректно размеченным, сбалансированным по классам и не содержать утечки информации о целевой переменной.

3. Почему данные необходимо разделять на несколько выборок?
Ответ: разделение необходимо, чтобы объективно оценить качество модели на данных, которые она не видела при обучении. Если проверять модель на тех же данных, на которых она обучалась, оценка будет завышенной и не покажет, как модель работает на новых данных.

4. Что такое обучающая выборка (train)?
Ответ: обучающая выборка — это часть данных, на которой модель обучается, то есть подбирает свои внутренние параметры. Обычно это самая большая часть датасета, составляющая 60–80% данных.

5. Что такое валидационная выборка (validation)?
Ответ: валидационная выборка — это часть данных, которая используется для подбора гиперпараметров модели, сравнения разных моделей и контроля переобучения в процессе разработки. Модель на ней не обучается, но решения разработчика принимаются по её результатам.

6. Что такое тестовая выборка (test)?
Ответ: тестовая выборка — это часть данных, которая используется только один раз, для итоговой оценки качества готовой модели. Она имитирует новые данные, поэтому её нельзя использовать при обучении и подборе параметров.

7. Какие пропорции разделения данных используются чаще всего?
Ответ: чаще всего используются пропорции 70/15/15 или 60/20/20 для обучающей, валидационной и тестовой выборок, а при разделении на две части — 80/20 или 70/30. При очень

больших датасетах на валидацию и тест можно выделять меньшую долю, например по 1–5%.

8. Что такое переобучение модели?
Ответ: переобучение — это ситуация, когда модель слишком точно подстраивается под обучающие данные, включая шум и случайные закономерности, и поэтому хорошо работает на обучающей выборке, но плохо — на новых данных. Обнаружить переобучение помогает сравнение качества на обучающей и валидационной выборках.

9. Что такое утечка данных (data leakage)?
Ответ: утечка данных — это ситуация, когда при обучении модель получает информацию, которая недоступна в реальных условиях, например сведения из тестовой выборки или признаки, напрямую связанные с ответом. Утечка приводит к завышенной оценке качества модели.

10. Как избежать утечки данных при формировании датасета?
Ответ: данные разделяют на выборки до масштабирования, заполнения пропусков и других преобразований, параметры которых вычисляются только по обучающей выборке. Также удаляют дубликаты между выборками и исключают признаки, содержащие информацию о будущем или о целевой переменной.

11. Как разделить данные на выборки с помощью scikit-learn?
Ответ: для разделения используется функция train_test_split() из модуля sklearn.model_selection. В ней задаются доля тестовой выборки (параметр test_size), параметр random_state для воспроизводимости и параметр stratify для стратифицированного разделения.

12. Для чего задаётся параметр random_state при разделении данных?
Ответ: параметр random_state фиксирует начальное значение генератора случайных чисел. Благодаря этому разделение данных получается одинаковым при каждом запуске, что обеспечивает воспроизводимость результатов и позволяет корректно сравнивать модели.

13. Что такое кросс-валидация?
Ответ: кросс-валидация — это метод оценки модели, при котором данные многократно разбиваются на обучающую и проверочную части, модель обучается и проверяется несколько раз, а итоговая оценка вычисляется как среднее по всем разбиениям. Это даёт более надёжную оценку, чем одно разделение.

14. Что такое k-блочная кросс-валидация (k-fold)?
Ответ: при k-блочной кросс-валидации данные делятся на k равных частей. Модель обучается k раз, каждый раз на k−1 частях, а оставшаяся часть используется для проверки. В результате каждая часть данных один раз становится проверочной. Обычно k равно 5 или 10.

15. Как разделяют данные временных рядов?
Ответ: данные временных рядов нельзя перемешивать случайным образом, так как модель не должна обучаться на будущих данных. Их разделяют по времени: более ранние данные используются для обучения, более поздние — для проверки и тестирования. В scikit-learn для этого есть timeseriessplit.

16. Что такое стратифицированная выборка?
Ответ: стратифицированная выборка — это способ разделения данных, при котором соотношение классов целевой переменной в каждой выборке сохраняется таким же, как в исходном датасете. Например, если в данных 10% положительного класса, то и в обучающей, и в тестовой выборке будет около 10%.

17. Для чего применяется стратифицированное разделение?
Ответ: стратифицированное разделение применяется, чтобы при случайном делении в одну из

выборок не попало слишком мало или совсем не попало объектов редкого класса. Это особенно важно для несбалансированных данных и небольших датасетов.

18. Как выполнить стратифицированное разделение в scikit-learn?
Ответ: в функции train_test_split() указывается параметр stratify, которому передаётся столбец с целевой переменной. Для кросс-валидации используется класс stratifiedkfold, который сохраняет соотношение классов в каждом блоке.

19. Что такое несбалансированность классов?
Ответ: несбалансированность классов — это ситуация, когда количество объектов одного класса значительно превышает количество объектов другого. Например, в задаче выявления мошенничества мошеннических операций может быть менее 1% от общего количества.

20. Почему несбалансированность классов является проблемой?
Ответ: модель, обученная на несбалансированных данных, стремится чаще предсказывать преобладающий класс и плохо распознаёт редкий, хотя именно он обычно наиболее важен. При этом показатель точности (accuracy) может быть высоким и вводить в заблуждение.

21. Какие метрики используются для оценки моделей на несбалансированных данных?
Ответ: вместо общей точности используются полнота (recall), точность (precision), f1-мера, roc-auc, pr-auc и матрица ошибок. Эти метрики показывают, насколько хорошо модель распознаёт редкий класс.

22. Какие существуют способы балансировки классов?
Ответ: применяются увеличение числа объектов редкого класса (oversampling), уменьшение числа объектов преобладающего класса (undersampling), их сочетание, генерация синтетических объектов, а также назначение классам разных весов при обучении модели.

23. Что такое случайное увеличение выборки (oversampling)?
Ответ: случайное увеличение выборки — это дублирование случайно выбранных объектов редкого класса до получения нужного соотношения классов. Метод простой, но может приводить к переобучению, так как модель многократно видит одни и те же объекты.

24. Что такое случайное уменьшение выборки (undersampling)?
Ответ: случайное уменьшение выборки — это удаление части объектов преобладающего класса, чтобы выровнять соотношение классов. Метод ускоряет обучение, но приводит к потере части полезной информации, поэтому подходит для больших датасетов.

25. Что такое метод smote?
Ответ: smote (synthetic minority over-sampling technique) — это метод балансировки, который создаёт новые синтетические объекты редкого класса. Новый объект строится между существующим объектом редкого класса и одним из его ближайших соседей того же класса. Метод реализован в библиотеке imbalanced-learn.

26. Что такое взвешивание классов?
Ответ: взвешивание классов — это способ балансировки без изменения данных, при котором ошибкам на объектах редкого класса присваивается больший вес при обучении. Во многих моделях scikit-learn для этого используется параметр class_weight="balanced".

27. Почему балансировку применяют только к обучающей выборке?
Ответ: валидационная и тестовая выборки должны отражать реальное распределение классов, иначе оценка качества модели будет искажённой. Кроме того, балансировка до разделения может привести к попаданию копий или синтетических объектов в тестовую выборку, то есть к утечке данных.

28. Что такое аугментация данных?
Ответ: аугментация данных — это искусственное увеличение объёма обучающей выборки за счёт создания изменённых копий существующих объектов. Изменения подбираются так, чтобы не менялся смысл объекта и его метка класса.

29. Какие методы аугментации применяются для изображений, текстов и звука?
Ответ: для изображений применяются повороты, отражения, обрезка, масштабирование, изменение яркости и контраста, добавление шума. Для текстов — замена слов синонимами, перестановка слов, обратный перевод. Для звука — добавление шума, изменение скорости и высоты тона, сдвиг по времени.

30. Для чего применяется аугментация данных и какие у неё ограничения?
Ответ: аугментация применяется при недостатке данных, для снижения переобучения, повышения устойчивости модели к изменениям входных данных и балансировки классов. Её ограничения: преобразования не должны искажать смысл и метку объекта, а аугментацию применяют только к обучающей выборке.

Критерии оценки устного опроса:

- Оценка «отлично» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов и смог ответить на уточняющие вопросы;
- Оценка «хорошо» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов, но не смог ответить на уточняющие вопросы;
- Оценка «удовлетворительно» выставляется студенту, если он смог изложить суть основного вопроса, но не смог ответить по существу дополнительных вопросов;
- Оценка «неудовлетворительно» выставляется студенту, если он не смог изложить суть основного вопроса.

Тема 1.5. Инструменты подготовки данных.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.9. Осуществлять конвейерную обработку данных для систем ИИ

Вопросы к устному опросу по теме:

1. Что такое библиотека numpy и для чего она используется?
Ответ: numpy — это библиотека языка python для работы с многомерными массивами и выполнения математических вычислений. Она используется для быстрой обработки числовых данных, линейной алгебры, статистических расчётов и является основой для многих других библиотек анализа данных и машинного обучения.

2. Что такое массив ndarray в numpy?
Ответ: ndarray — это основной объект numpy, многомерный массив элементов одного типа. Массив имеет размерность (ndim), форму (shape), количество элементов (size) и тип данных (dtype). Благодаря единому типу элементов массивы занимают меньше памяти и обрабатываются быстрее списков python.

3. Почему numpy работает быстрее стандартных списков python?
Ответ: массивы numpy хранят элементы одного типа в непрерывной области памяти, а операции над ними выполняются скомпилированным кодом на языке c. Кроме того, numpy поддерживает векторизованные операции, которые обрабатывают весь массив сразу без циклов на python.

4. Что такое векторизация в numpy?
Ответ: векторизация — это выполнение операций сразу над всеми элементами массива без явного использования циклов. Например, выражение a * 2 умножает каждый элемент массива a на 2. Векторизация делает код короче и значительно ускоряет вычисления.

5. Что такое транслирование (broadcasting) в numpy?
Ответ: транслирование — это механизм, позволяющий выполнять операции над массивами разной формы. Массив меньшего размера автоматически растягивается до формы большего массива без фактического копирования данных, например при прибавлении одного числа ко всем элементам матрицы.

6. Какие основные операции с массивами поддерживает numpy?
Ответ: numpy поддерживает создание массивов (array, zeros, ones, arange, linspace), изменение формы (reshape), индексацию и срезы, арифметические операции, статистические функции (mean, median, std, min, max), сортировку, объединение массивов и операции линейной алгебры.

7. Что такое библиотека pandas и для чего она используется?
Ответ: pandas — это библиотека python для обработки и анализа табличных данных. Она позволяет загружать данные из разных источников, очищать, фильтровать, группировать, объединять и преобразовывать их, а также сохранять результат в различных форматах.

8. Какие основные структуры данных есть в pandas?
Ответ: основными структурами являются series и dataframe. Series — это одномерный массив значений с индексом, аналог одного столбца таблицы. Dataframe — это двумерная таблица со строками и столбцами, где каждый столбец является объектом series.

9. Как загрузить и сохранить данные с помощью pandas?
Ответ: для загрузки данных используются функции read_csv(), read_excel(), read_json(), read_parquet(), read_sql(). Для сохранения применяются методы to_csv(), to_excel(), to_json(), to_parquet(). Это позволяет работать с основными форматами хранения данных.

10. Какие методы pandas используются для первичного анализа данных?
Ответ: для первичного анализа используются методы head() и tail() для просмотра первых и последних строк, info() для просмотра типов и количества заполненных значений, describe() для получения статистических характеристик, shape для определения размера таблицы и value_counts() для подсчёта значений.

11. Как выполняется фильтрация и выборка данных в pandas?
Ответ: выборка столбцов выполняется по их названиям, выборка строк — с помощью loc (по меткам) и iloc (по номерам позиций). Фильтрация выполняется по логическим условиям, например df[df["возраст"] > 18], а также с помощью метода query().

12. Как выполняется группировка данных в pandas?
Ответ: группировка выполняется методом groupby(), после которого применяются агрегирующие функции: sum(), mean(), count(), min(), max() или agg() для нескольких функций сразу. Например, можно вычислить средний доход клиентов по каждому городу.

13. Как объединить несколько таблиц в pandas?

Ответ: для объединения таблиц по ключевым столбцам используется функция merge(), аналогичная оператору join в sql, с типами объединения inner, left, right и outer. Для соединения таблиц друг под другом или рядом используется функция concat().

14. Какие методы pandas применяются для очистки данных?
Ответ: для очистки используются isnull() и isna() для поиска пропусков, fillna() для их заполнения, dropna() для удаления, duplicated() и drop_duplicates() для работы с дубликатами, replace() для замены значений, astype() для изменения типов данных и строковые методы str для обработки текста.

15. Какие ограничения имеет pandas при работе с большими данными?
Ответ: pandas загружает все данные в оперативную память одного компьютера и в основном выполняет вычисления в одном потоке. Поэтому при объёме данных, превышающем доступную память, работа становится невозможной или очень медленной, и требуются распределённые инструменты.

16. Что такое большие данные (big data)?
Ответ: большие данные — это наборы данных такого объёма, скорости поступления и разнообразия, что их невозможно эффективно обработать традиционными средствами на одном компьютере. Для их обработки используются распределённые системы, работающие на кластере из многих компьютеров.

17. Что такое apache spark и pyspark?
Ответ: apache spark — это платформа для распределённой обработки больших данных на кластере компьютеров. Pyspark — это интерфейс для работы со spark на языке python, который позволяет обрабатывать большие объёмы данных с помощью привычного синтаксиса.

18. Как устроена распределённая обработка данных в spark?
Ответ: данные разбиваются на части (партиции), которые обрабатываются параллельно на разных узлах кластера. Управляющая программа (драйвер) распределяет задачи между исполнителями (executors), а затем собирает результаты. Такой подход позволяет обрабатывать данные, не помещающиеся в память одного компьютера.

19. Что такое sparksession?
Ответ: sparksession — это точка входа в программу на pyspark, через которую создаются датафреймы, читаются данные и выполняются запросы. Она создаётся в начале работы с помощью sparksession.builder и метода getorcreate().

20. Что такое dataframe и rdd в pyspark?
Ответ: rdd (resilient distributed dataset) — это базовая распределённая коллекция объектов, устойчивая к сбоям. Dataframe — это распределённая таблица с именованными столбцами и схемой, похожая на dataframe в pandas. Dataframe работает быстрее rdd благодаря встроенному оптимизатору запросов, поэтому сейчас используется чаще.

21. Что такое ленивые вычисления в spark?
Ответ: ленивые вычисления означают, что преобразования данных не выполняются сразу, а только запоминаются в виде плана. Реальные вычисления запускаются только при вызове действия. Это позволяет spark оптимизировать весь план обработки и не выполнять лишних операций.

22. Чем преобразования отличаются от действий в pyspark?
Ответ: преобразования (transformations) создают новый набор данных на основе существующего и выполняются лениво, например select(), filter(), groupby(), withcolumn(), join(). Действия (actions) запускают вычисления и возвращают результат, например show(), count(), collect(), a

также запись данных.

23. Какие операции подготовки данных можно выполнять в pyspark?
Ответ: в pyspark можно читать данные из csv, json, parquet и баз данных, фильтровать и выбирать столбцы, удалять дубликаты (dropduplicates), обрабатывать пропуски (fillna, dropna), группировать и агрегировать данные, объединять таблицы, а также выполнять sql-запросы с помощью spark.sql().

24. В каких случаях следует использовать pyspark, а в каких pandas?
Ответ: pandas удобен для небольших и средних наборов данных, которые помещаются в оперативную память, и для быстрого исследования данных. Pyspark следует использовать, когда объём данных очень большой, данные хранятся на кластере или требуется распределённая обработка в конвейере.

25. Что такое версионирование данных?
Ответ: версионирование данных — это сохранение истории изменений наборов данных с возможностью вернуться к любой предыдущей версии. Оно позволяет точно знать, на каких данных обучалась модель, и воспроизвести результаты эксперимента.

26. Что такое dvc и для чего он используется?
Ответ: dvc (data version control) — это инструмент с открытым исходным кодом для версионирования данных и моделей машинного обучения. Он работает вместе с системой git: git хранит код и небольшие служебные файлы, а сами большие данные хранятся в отдельном хранилище.

27. Почему для версионирования больших данных недостаточно git?
Ответ: git предназначен для хранения текстовых файлов кода и плохо работает с большими двоичными файлами: репозиторий быстро разрастается, операции замедляются, а у сервисов есть ограничения на размер файлов. Dvc решает эту проблему, сохраняя в git только ссылки на данные.

28. Как dvc хранит данные?
Ответ: при добавлении данных dvc создаёт небольшой файл с расширением .dvc, содержащий хеш-сумму и сведения о данных. Этот файл сохраняется в git, а сами данные помещаются в кэш dvc и могут выгружаться в удалённое хранилище: облачное хранилище s3, google drive, сетевую папку или сервер.

29. Какие основные команды dvc вы знаете?
Ответ: команда dvc init инициализирует dvc в проекте, dvc add добавляет данные под контроль версий, dvc remote add настраивает удалённое хранилище, dvc push выгружает данные в хранилище, dvc pull загружает их, dvc checkout восстанавливает нужную версию данных, dvc repro запускает конвейер.

30. Как с помощью dvc описываются конвейеры обработки данных?
Ответ: конвейеры описываются в файле dvc.yaml в виде последовательности этапов. Для каждого этапа указываются команда запуска, входные зависимости и выходные результаты. Команда dvc repro выполняет только те этапы, у которых изменились входные данные или код, что экономит время и обеспечивает воспроизводимость.

Критерии оценки устного опроса:

- Оценка «отлично» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов и смог ответить на уточняющие вопросы;

- Оценка «хорошо» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов, но не смог ответить на уточняющие вопросы;
- Оценка «удовлетворительно» выставляется студенту, если он смог изложить суть основного вопроса, но не смог ответить по существу дополнительных вопросов;
- Оценка «неудовлетворительно» выставляется студенту, если он не смог изложить суть основного вопроса.

Практическое занятие №2. Разметка данных для задач классификации, проверка качества разметки, экспорт в форматы для обучения моделей.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.9. Осуществлять конвейерную обработку данных для систем ИИ

Цель: научиться выполнять разметку данных для задач классификации, проверять качество разметки с помощью показателей согласованности разметчиков и экспортировать размеченные данные в форматы, пригодные для обучения моделей машинного обучения.

Задание:
1. Определить задачу классификации и набор классов.
2. Составить инструкцию для разметчиков с описанием каждого класса.
3. Подготовить данные для разметки.
4. Выполнить разметку данных вручную или с помощью инструмента разметки.
5. Проверить качество разметки: согласованность разметчиков, ошибки и пропуски меток.
6. Устранить расхождения и ошибки в разметке.
7. Проанализировать распределение классов.
8. Экспортировать размеченные данные в форматы для обучения моделей и сделать вывод о готовности датасета.

Этапы выполнения:
Этап 1. Постановка задачи разметки.
Определяется, какую задачу классификации будет решать модель, какие объекты размечаются (тексты, изображения, записи таблицы) и какой набор классов используется. Классы должны быть понятными, не пересекаться между собой и охватывать все возможные случаи.

Этап 2. Разработка инструкции по разметке.
Составляется инструкция для разметчиков, в которой описывается каждый класс, приводятся примеры объектов, относящихся к нему, и правила для спорных случаев. Хорошая инструкция снижает количество ошибок и расхождений между разметчиками.

Этап 3. Подготовка данных.
Данные загружаются и очищаются с помощью pandas: удаляются пустые записи и дубликаты, каждому объекту присваивается уникальный идентификатор. При необходимости выбирается часть данных для разметки.

Этап 4. Разметка данных.

Разметка выполняется вручную в таблице или с помощью специализированных инструментов, например label studio или cvat. Для проверки качества одна и та же часть данных размечается несколькими разметчиками независимо друг от друга.

Этап 5. Проверка качества разметки.
Проверяется, что каждый объект имеет метку и все метки входят в заданный набор классов. Вычисляется доля совпадений меток разных разметчиков и коэффициент каппа коэна (функция cohen_kappa_score из scikit-learn). Значение каппы выше 0,8 говорит о высокой согласованности, от 0,6 до 0,8 — о достаточной, ниже 0,6 — о необходимости уточнить инструкцию.

Этап 6. Устранение ошибок разметки.
Объекты, по которым разметчики не согласились, пересматриваются экспертом или решаются голосованием большинства. По результатам при необходимости уточняется инструкция по разметке.

Этап 7. Анализ распределения классов.
С помощью метода value_counts() подсчитывается количество объектов каждого класса. При сильной несбалансированности планируются меры балансировки или дополнительная разметка объектов редких классов.

Этап 8. Экспорт данных и оформление результата.
Размеченные данные экспортируются в форматы для обучения: csv с колонками «текст» и «метка», json или jsonl, а для изображений — структура папок по классам. Набор данных разделяется на обучающую, валидационную и тестовую выборки и добавляется под контроль версий dvc. Оформляется отчёт с результатами проверки качества и выводами.

Пример решения:
Задание: разметить отзывы покупателей интернет-магазина для модели определения тональности отзыва.
Дано: имеется файл csv с 500 отзывами покупателей. Каждый отзыв нужно отнести к одному из классов: «положительный», «отрицательный», «нейтральный». Разметку выполняют два разметчика.

Решение по этапам:
1. Определяем задачу — многоклассовая классификация текстов по тональности на три класса.
2. Составляем инструкцию: положительный отзыв содержит одобрение товара, отрицательный — жалобы и недовольство, нейтральный — описание без явной оценки. Для смешанных отзывов класс определяется по общему выводу покупателя.
3. Загружаем данные с помощью pandas, удаляем 12 пустых отзывов и 8 дубликатов, присваиваем каждому отзыву идентификатор.
4. Выполняем разметку в label studio. Первые 100 отзывов размечают оба разметчика независимо, остальные распределяются между ними.
5. Проверяем, что все отзывы имеют метку из допустимого набора. Сравниваем разметку 100 общих отзывов: совпадение составляет 88%, коэффициент каппа коэна — 0,81, что говорит о высокой согласованности.
6. Спорные 12 отзывов пересматриваем совместно и назначаем окончательную метку, в инструкцию добавляем правило для отзывов с сарказмом.
7. Анализируем распределение классов: положительных отзывов 55%, отрицательных 30%, нейтральных 15%. Отмечаем необходимость стратифицированного разделения.
8. Экспортируем данные в файл csv с колонками id, text, label и в формат jsonl.
9. Выполняем стратифицированное разделение на обучающую, валидационную и тестовую выборки в соотношении 70/15/15.
10. Добавляем датасет под контроль версий командой dvc add и оформляем отчёт.

Ответ: отзывы размечены на три класса тональности по единой инструкции. Качество разметки подтверждено высокой согласованностью разметчиков (каппа коэна 0,81), расхождения устранены. Данные экспортированы в форматы csv и jsonl, разделены на выборки с сохранением соотношения классов и добавлены под контроль версий dvc. Датасет готов к обучению модели классификации.

Критерии оценки практических заданий:
1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Тема 2.1. Основы оркестрации и введение в Apache Airflow.
Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.10. Осуществлять мониторинг и оркестрацию конвейеров обработки данных

Вопросы к устному опросу по теме:
1. Что такое оркестрация задач?
Ответ: Оркестрация задач — это автоматизированное управление выполнением набора взаимосвязанных задач: определение порядка их запуска, планирование по расписанию, учёт зависимостей, контроль выполнения, обработка сбоев и повторные запуски. Оркестрация обеспечивает согласованную работу всех этапов конвейера.

2. Чем оркестрация отличается от простого запуска скриптов по расписанию?
Ответ: Простой планировщик, например cron, только запускает отдельные скрипты в заданное время и не учитывает связи между ними. Оркестратор учитывает зависимости между задачами, запускает следующую задачу только после успешного завершения предыдущей, повторяет задачи при сбоях, хранит историю запусков и предоставляет средства мониторинга.

3. Зачем нужна оркестрация в конвейерах данных?
Ответ: Конвейеры данных состоят из множества этапов: извлечения, очистки, преобразования, загрузки, обучения моделей. Оркестрация нужна, чтобы эти этапы выполнялись в правильном порядке и в нужное время, сбои обнаруживались и обрабатывались автоматически, а работа всего конвейера была прозрачной и контролируемой.

4. Какие проблемы возникают при отсутствии оркестрации?
Ответ: Без оркестрации задачи приходится запускать вручную или разрозненными скриптами, возникают ошибки из-за неверного порядка выполнения, сбои остаются незамеченными, сложно повторить неудачный запуск, отсутствует единая история выполнения и трудно масштабировать конвейер при росте количества задач.

5. Какие основные функции выполняет оркестратор?
Ответ: Оркестратор описывает задачи и зависимости между ними, планирует запуски по расписанию или по событию, распределяет задачи между исполнителями, отслеживает их

состояние, выполняет повторные попытки при сбоях, отправляет уведомления, ведёт журналы и предоставляет веб-интерфейс для мониторинга.

6. Какие инструменты оркестрации конвейеров данных вы знаете?
Ответ: Наиболее известными инструментами являются Apache Airflow, Luigi, Prefect и Dagster. Также применяются облачные сервисы оркестрации и инструменты для конкретных платформ, например Kubeflow Pipelines для конвейеров машинного обучения в Kubernetes.

7. Что такое Apache Airflow?
Ответ: Apache Airflow — это платформа с открытым исходным кодом для разработки, планирования и мониторинга конвейеров данных. Конвейеры в Airflow описываются в виде кода на языке Python. Платформа была создана в компании Airbnb, а затем передана в фонд Apache Software Foundation.

8. Каковы основные преимущества Apache Airflow?
Ответ: Airflow позволяет описывать конвейеры кодом на Python, имеет удобный веб-интерфейс, большое количество готовых операторов и интеграций с базами данных и облачными сервисами, поддерживает масштабирование, обладает большим сообществом и подробной документацией.

9. Что такое Luigi?
Ответ: Luigi — это библиотека Python для построения конвейеров пакетной обработки данных, разработанная в компании Spotify. В Luigi задачи описываются как классы с указанием зависимостей и результатов. Инструмент простой, но имеет ограниченные возможности планирования и мониторинга по сравнению с Airflow.

10. Что такое Prefect?
Ответ: Prefect — это современный инструмент оркестрации на Python, в котором конвейеры описываются как обычные функции с декораторами flow и task. Он отличается простотой использования, гибким запуском задач и удобной обработкой ошибок, а также предлагает облачный сервис для мониторинга.

11. Что такое Dagster?
Ответ: Dagster — это оркестратор, ориентированный на данные. В нём основной единицей являются активы данных (assets), то есть таблицы, файлы и модели, которые создаёт конвейер. Dagster уделяет большое внимание типизации, тестированию и отслеживанию происхождения данных.

12. Как выбрать инструмент оркестрации?
Ответ: Выбор зависит от сложности и масштаба конвейеров, опыта команды, требований к мониторингу и интеграциям, способа развёртывания и необходимости поддержки сообщества. Airflow подходит для сложных промышленных конвейеров, Luigi — для простых задач, Prefect — для быстрого старта, Dagster — для конвейеров с акцентом на активы данных.

13. Из каких основных компонентов состоит архитектура Apache Airflow?
Ответ: Основными компонентами Airflow являются планировщик (Scheduler), веб-сервер (Webserver), исполнитель (Executor), база метаданных (Metadata Database) и папка с файлами DAG. В распределённой установке также используются рабочие процессы (Workers), обработчик файлов DAG (DAG Processor) и компонент Triggerer.

14. Какую роль выполняет планировщик (Scheduler) в Airflow?
Ответ: Планировщик отслеживает все DAG и задачи, определяет, когда нужно запустить очередной экземпляр конвейера, проверяет выполнение зависимостей между задачами и передаёт готовые к выполнению задачи исполнителю. Планировщик является центральным компонентом Airflow.

15. Какую роль выполняет веб-сервер (Webserver) в Airflow?
Ответ: Веб-сервер предоставляет графический веб-интерфейс, в котором можно просматривать список DAG, их структуру, историю и состояние запусков, журналы задач, вручную запускать и останавливать конвейеры, а также повторно выполнять отдельные задачи. В Airflow 3 функции веб-сервера выполняет API-сервер.

16. Что такое исполнитель (Executor) в Airflow?
Ответ: Исполнитель определяет, каким образом и где будут выполняться задачи, полученные от планировщика: последовательно на одном компьютере, параллельно в нескольких процессах или распределённо на нескольких машинах. Исполнитель настраивается в конфигурации Airflow.

17. Какие виды исполнителей существуют в Airflow?
Ответ: LocalExecutor выполняет задачи параллельно в нескольких процессах на одном компьютере, CeleryExecutor распределяет задачи между рабочими узлами через брокер сообщений, например Redis или RabbitMQ, KubernetesExecutor запускает каждую задачу в отдельном контейнере Kubernetes. В Airflow 2 для тестирования также использовался SequentialExecutor, выполняющий задачи строго по одной.

18. Что такое база метаданных (Metadata Database) в Airflow?
Ответ: База метаданных — это реляционная база данных, в которой Airflow хранит сведения о DAG, запусках конвейеров, состояниях задач, пользователях, подключениях, переменных и настройках. Все компоненты Airflow обмениваются информацией через эту базу.

19. Какие СУБД используются в качестве базы метаданных Airflow?
Ответ: Для промышленной эксплуатации рекомендуется использовать PostgreSQL или MySQL. База SQLite может использоваться только для обучения и тестирования, так как не поддерживает параллельную работу и не подходит для реальной нагрузки.

20. Как взаимодействуют компоненты Airflow между собой?
Ответ: Файлы DAG разбираются и сведения о них сохраняются в базе метаданных. Планировщик на основе этих сведений и расписания определяет задачи для запуска и передаёт их исполнителю. Исполнитель запускает задачи, а результаты их выполнения записываются в базу метаданных. Веб-интерфейс читает данные из базы и отображает состояние конвейеров.

21. Что такое DAG в Airflow?
Ответ: DAG (Directed Acyclic Graph, направленный ациклический граф) — это описание конвейера в Airflow, состоящее из набора задач и зависимостей между ними. Направленность означает, что задачи выполняются в определённом порядке, а ацикличность — что в графе нет циклов и задача не может зависеть сама от себя.

22. Почему граф задач в Airflow должен быть ациклическим?
Ответ: Если бы в графе был цикл, задачи бесконечно ожидали бы друг друга, и конвейер никогда бы не завершился. Ацикличность гарантирует, что у выполнения есть начало и конец, а порядок запуска задач всегда можно однозначно определить.

23. Как описывается DAG в Airflow?
Ответ: DAG описывается в файле на языке Python, который помещается в папку dags. В файле создаётся объект DAG с идентификатором и параметрами расписания, затем определяются задачи с помощью операторов или декоратора task, после чего задаются зависимости между задачами.

24. Какие основные параметры задаются при создании DAG?
Ответ: Основными параметрами являются dag_id — уникальный идентификатор, start_date —

дата начала работы, schedule — расписание запуска, catchup — признак выполнения пропущенных запусков за прошедший период, default_args — общие параметры задач (количество повторов, интервал между повторами, владелец), а также описание и теги.

25. Что такое задача (Task) и оператор (Operator) в Airflow?
Ответ: Задача — это отдельный шаг конвейера внутри DAG. Оператор — это шаблон, определяющий, какое действие выполняет задача. Например, BashOperator выполняет команду командной строки, PythonOperator вызывает функцию Python, а специальные операторы выполняют SQL-запросы или работают с облачными сервисами.

26. Как задаются зависимости между задачами в Airflow?
Ответ: Зависимости задаются с помощью операторов >> и <<, например extract >> transform >> load означает, что transform выполняется после extract, а load — после transform. Также используются методы set_upstream() и set_downstream(). Несколько задач можно передать списком для параллельного выполнения.

27. Какие состояния может иметь задача в Airflow?
Ответ: Задача может находиться в состояниях: none (не запланирована), scheduled (запланирована), queued (в очереди), running (выполняется), success (успешно завершена), failed (завершилась ошибкой), up_for_retry (ожидает повторной попытки), skipped (пропущена), upstream_failed (не выполнена из-за ошибки предыдущей задачи).

28. Какие способы установки Apache Airflow существуют?
Ответ: Airflow можно установить с помощью менеджера пакетов pip в виртуальное окружение Python с использованием файла ограничений версий (constraints), развернуть в контейнерах с помощью Docker и официального файла docker-compose.yaml, установить в Kubernetes через официальный Helm-чарт или использовать облачные управляемые сервисы.

29. Какие основные действия выполняются при установке Airflow через pip?
Ответ: Создаётся виртуальное окружение Python, задаётся переменная окружения AIRFLOW_HOME, устанавливается пакет apache-airflow с файлом ограничений версий, инициализируется база метаданных командой airflow db migrate, создаётся пользователь, затем запускаются планировщик и веб-интерфейс. Для быстрого знакомства можно использовать команду airflow standalone, которая выполняет эти действия автоматически.

30. Как выполняется настройка Apache Airflow?
Ответ: Основные настройки хранятся в конфигурационном файле airflow.cfg в каталоге AIRFLOW_HOME и могут переопределяться переменными окружения. В них указываются путь к папке с DAG, тип исполнителя, строка подключения к базе метаданных, часовой пояс и параметры параллельности. Подключения к внешним системам и переменные настраиваются через веб-интерфейс или командную строку.

Критерии оценки устного опроса:

- оценка «отлично» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов и смог ответить на уточняющие вопросы;
- оценка «хорошо» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов, но не смог ответить на уточняющие вопросы;
- оценка «удовлетворительно» выставляется студенту, если он смог изложить суть основного вопроса, но не смог ответить по существу дополнительных вопросов;
- оценка «неудовлетворительно» выставляется студенту, если он не смог изложить суть основного вопроса.

Тема 2.2. Разработка DAG, операторы, сенсоры и зависимости задач.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.10. Осуществлять мониторинг и оркестрацию конвейеров обработки данных

Вопросы к устному опросу по теме:

1. Какова структура файла DAG в Airflow?
Ответ: Файл DAG — это сценарий на языке Python, который обычно содержит импорт необходимых модулей, задание общих параметров задач (default_args), создание объекта DAG с идентификатором и расписанием, определение задач с помощью операторов или декоратора task и задание зависимостей между задачами.

2. Где размещаются файлы DAG и как Airflow их обнаруживает?
Ответ: Файлы DAG размещаются в папке dags, путь к которой указан в настройках Airflow. Airflow периодически просматривает эту папку, разбирает найденные файлы Python и регистрирует описанные в них DAG, после чего они появляются в веб-интерфейсе.

3. Какими способами можно объявить DAG?
Ответ: DAG можно объявить с помощью контекстного менеджера with DAG(...) as dag, внутри которого создаются задачи, с помощью декоратора @dag, применяемого к функции, или созданием объекта DAG и явной передачей его в каждую задачу через параметр dag.

4. Что такое default_args и для чего они используются?
Ответ: Default_args — это словарь параметров, которые применяются ко всем задачам DAG по умолчанию. В нём обычно указываются владелец (owner), количество повторных попыток (retries), интервал между попытками (retry_delay), отправка уведомлений об ошибках. Это избавляет от повторения одинаковых параметров в каждой задаче.

5. Что такое оператор в Airflow?
Ответ: Оператор — это класс, описывающий одно конкретное действие, которое выполняет задача. При создании объекта оператора внутри DAG появляется задача. Операторы бывают действующими (выполняют действие), передающими данные между системами и сенсорами (ожидают выполнения условия).

6. Для чего используется BashOperator?
Ответ: BashOperator выполняет команду или сценарий командной оболочки Bash. Команда задаётся параметром bash_command. Он применяется для запуска скриптов, копирования и перемещения файлов, вызова консольных утилит и программ.

7. Для чего используется PythonOperator?
Ответ: PythonOperator вызывает функцию на языке Python. Функция передаётся в параметр python_callable, а её аргументы — через параметры op_args и op_kwargs. Он применяется для обработки данных, вызова библиотек Pandas и scikit-learn, обращения к API и другой логики на Python.

8. Что такое TaskFlow API?
Ответ: TaskFlow API — это способ описания задач с помощью декораторов @task и @dag.

Обычная функция Python, помеченная декоратором @task, становится задачей Airflow. Результат функции автоматически передаётся следующей задаче через XCom, а зависимости определяются по вызовам функций, что делает код короче и понятнее.

9. Какие операторы используются для работы с базами данных?
Ответ: Для выполнения SQL-запросов используется универсальный оператор SQLExecuteQueryOperator из пакета провайдера common.sql, который работает с разными СУБД через подключения Airflow. Также существуют операторы для конкретных систем, например для PostgreSQL, MySQL, ClickHouse, и операторы проверки данных, например SQLCheckOperator.

10. Что такое провайдеры (providers) в Airflow?
Ответ: Провайдеры — это отдельно устанавливаемые пакеты, которые расширяют Airflow операторами, сенсорами, хуками и подключениями для работы с конкретными системами: базами данных, облачными сервисами, системами обмена сообщениями. Например, пакет apache-airflow-providers-postgres добавляет поддержку PostgreSQL.

11. Что такое хук (Hook) в Airflow?
Ответ: Хук — это интерфейс для подключения к внешней системе, например базе данных или облачному хранилищу. Хук использует сведения из подключения Airflow и предоставляет методы для работы с системой. Операторы используют хуки внутри себя, а разработчик может вызывать их в функциях Python.

12. Что такое сенсор в Airflow?
Ответ: Сенсор — это особый вид оператора, который ожидает наступления определённого события или выполнения условия и только после этого позволяет запуститься следующим задачам. Например, сенсор может ждать появления файла, завершения задачи в другом DAG или появления записи в таблице.

13. Какие сенсоры вы знаете?
Ответ: Распространены FileSensor (ожидание появления файла), ExternalTaskSensor (ожидание завершения задачи в другом DAG), SqlSensor (ожидание выполнения SQL-условия), HttpSensor (ожидание ответа веб-сервиса), DateTimeSensor (ожидание заданного времени), а также сенсоры облачных хранилищ, например S3KeySensor.

14. Какие параметры настраиваются у сенсоров?
Ответ: Основными параметрами являются poke_interval — интервал между проверками условия, timeout — максимальное время ожидания, после которого сенсор завершается ошибкой, mode — режим работы сенсора, а также soft_fail — признак пропуска задачи вместо ошибки при истечении времени ожидания.

15. Чем отличаются режимы работы сенсора poke и reschedule?
Ответ: В режиме poke сенсор занимает рабочий слот всё время ожидания и периодически проверяет условие. В режиме reschedule сенсор освобождает слот между проверками и снова запускается через заданный интервал. Режим reschedule экономит ресурсы при длительном ожидании. Также существуют отложенные (deferrable) сенсоры, которые передают ожидание компоненту Triggerer.

16. Что такое XCom?
Ответ: XCom (cross-communication) — это механизм Airflow для передачи небольших объёмов данных между задачами. Задача сохраняет значение в XCom, а другая задача может его получить. Данные XCom хранятся в базе метаданных Airflow.

17. Как передаются данные через XCom?
Ответ: Значение передаётся методом xcom_push(), а получается методом xcom_pull() с указанием

идентификатора задачи и ключа. Значение, которое возвращает функция в PythonOperator или задача TaskFlow, автоматически сохраняется в XCom с ключом return_value. В TaskFlow API передача выполняется просто через аргументы функций.

18. Какие ограничения имеет механизм XCom?
Ответ: XCom предназначен только для небольших данных: идентификаторов, путей к файлам, параметров, коротких результатов. Большие наборы данных через XCom передавать нельзя, так как это перегружает базу метаданных. Большие данные сохраняют во внешнее хранилище, а через XCom передают только путь к ним.

19. Что означают понятия upstream и downstream?
Ответ: Upstream-задачи — это задачи, которые должны выполниться до текущей задачи, то есть её предшественники. Downstream-задачи — это задачи, которые выполняются после текущей, то есть её последователи. Например, в цепочке extract >> transform задача extract является upstream для transform.

20. Как задаются зависимости между задачами?
Ответ: Зависимости задаются операторами >> (задача слева выполняется раньше) и << (задача справа выполняется раньше), методами set_downstream() и set_upstream(), а также функцией chain() для длинных цепочек. Для параллельных задач используются списки, например extract >> [clean, validate] >> load.

21. Что такое правила запуска (trigger rules)?
Ответ: Правила запуска определяют, при каком состоянии предшествующих задач запускается текущая задача. По умолчанию действует правило all_success — запуск после успешного завершения всех предшественников. Также используются all_failed, all_done, one_success, one_failed, none_failed и другие.

22. Как настраивается расписание запуска DAG?
Ответ: Расписание задаётся параметром schedule при создании DAG. В нём можно указать cron-выражение, готовую предустановку (например @daily), интервал времени timedelta, расписание (timetable) или значение None, если DAG запускается только вручную или по внешнему событию.

23. Что такое cron-выражение?
Ответ: Cron-выражение — это строка из пяти полей, задающая расписание: минута, час, день месяца, месяц, день недели. Например, выражение 0 2 * * * означает запуск ежедневно в 02:00, а 30 9 * * 1-5 — в 09:30 по рабочим дням. Символ * означает любое значение.

24. Какие предустановленные расписания есть в Airflow?
Ответ: В Airflow есть предустановки @once (один раз), @hourly (каждый час), @daily (каждый день в полночь), @weekly (каждую неделю), @monthly (каждый месяц) и @yearly (каждый год). Они заменяют соответствующие cron-выражения и делают код более понятным.

25. Что такое timetable в Airflow?
Ответ: Timetable (расписание) — это механизм, позволяющий описывать сложные расписания, которые нельзя выразить cron-выражением, например запуск только в рабочие дни с учётом праздников или в разное время в разные дни. Для этого создаётся собственный класс расписания или используются встроенные, например CronTriggerTimetable.

26. Что такое параметр catchup и как он влияет на запуски?
Ответ: Параметр catchup определяет, нужно ли выполнять пропущенные запуски за период от start_date до текущей даты. При значении True Airflow запустит DAG за каждый прошедший интервал, при значении False будет выполнен только запуск за последний интервал.

27. Что такое переменные (Variables) в Airflow?
Ответ: Переменные — это пары «ключ — значение», которые хранятся в базе метаданных Airflow и доступны во всех DAG. В них хранят параметры конвейеров: пути к папкам, адреса сервисов, настройки. Переменные создаются через веб-интерфейс, командную строку или переменные окружения и читаются методом Variable.get() или в шаблонах.

28. Что такое подключения (Connections) в Airflow?
Ответ: Подключения — это сохранённые параметры доступа к внешним системам: тип подключения, адрес сервера, порт, имя базы данных, логин и пароль. Каждое подключение имеет идентификатор conn_id, по которому операторы и хуки получают доступ к системе без указания паролей в коде DAG.

29. Почему пароли и параметры доступа не следует указывать в коде DAG?
Ответ: Код DAG обычно хранится в системе контроля версий и доступен многим разработчикам, поэтому пароли в нём могут быть раскрыты. Хранение параметров в подключениях и переменных Airflow, а также во внешних хранилищах секретов обеспечивает безопасность и позволяет менять настройки без изменения кода.

30. Что такое шаблоны Jinja в Airflow?
Ответ: Шаблоны Jinja позволяют подставлять значения в параметры операторов во время выполнения задачи. Например, выражение {{ ds }} подставляет логическую дату запуска, {{ var.value.имя }} — значение переменной, {{ params.имя }} — параметр DAG. Это позволяет обрабатывать данные за нужный период без изменения кода.

Критерии оценки устного опроса:

- оценка «отлично» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов и смог ответить на уточняющие вопросы;
- оценка «хорошо» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов, но не смог ответить на уточняющие вопросы;
- оценка «удовлетворительно» выставляется студенту, если он смог изложить суть основного вопроса, но не смог ответить по существу дополнительных вопросов;
- оценка «неудовлетворительно» выставляется студенту, если он не смог изложить суть основного вопроса.

Практическое занятие №3. Создание и запуск первого DAG в Apache Airflow.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.10. Осуществлять мониторинг и оркестрацию конвейеров обработки данных

Цель: Научиться создавать DAG в Apache Airflow, использовать операторы, сенсоры и XCom, задавать зависимости между задачами и расписание запуска, запускать DAG и контролировать его выполнение через веб-интерфейс.

Задание:

1. Определить назначение конвейера и состав его задач.
2. Подготовить рабочую среду Airflow и проверить её работоспособность.
3. Создать файл DAG и задать параметры DAG и default_args.
4. Создать задачи с использованием BashOperator, PythonOperator и сенсора.
5. Организовать передачу данных между задачами через XCom.
6. Задать зависимости между задачами и расписание запуска.
7. Запустить DAG и проверить выполнение задач и журналы.
8. Сделать вывод о корректности работы DAG.

Этапы выполнения:
Этап 1. Проектирование конвейера.
Определяется, какую задачу решает конвейер, из каких шагов он состоит и в каком порядке они должны выполняться. Составляется схема задач с указанием зависимостей между ними.

Этап 2. Подготовка среды.
Проверяется, что Airflow установлен и запущен: работает планировщик, открывается веб-интерфейс, известна папка dags. При необходимости создаются переменные и подключения, которые будут использоваться в DAG.

Этап 3. Создание файла DAG.
В папке dags создаётся файл Python. В нём импортируются необходимые модули, задаётся словарь default_args с владельцем, количеством повторов и интервалом между ними, создаётся объект DAG с идентификатором, датой начала, расписанием, параметром catchup и тегами.

Этап 4. Создание задач.
Внутри DAG создаются задачи: сенсор для ожидания входных данных, BashOperator для выполнения команды командной строки и PythonOperator для обработки данных функцией Python. Каждой задаче присваивается уникальный идентификатор task_id.

Этап 5. Передача данных через XCom.
Одна задача возвращает результат работы, который автоматически сохраняется в XCom, а следующая задача получает его методом xcom_pull() или через аргументы функции в TaskFlow API.

Этап 6. Задание зависимостей и расписания.
С помощью оператора >> задаётся порядок выполнения задач. Расписание указывается в параметре schedule в виде cron-выражения или предустановки.

Этап 7. Запуск и проверка DAG.
Проверяется, что DAG появился в веб-интерфейсе без ошибок импорта. DAG включается и запускается вручную. В представлении графа контролируются состояния задач, просматриваются журналы и значения XCom. При ошибках код исправляется, а неудавшиеся задачи запускаются повторно.

Этап 8. Оформление результата.
Оформляется отчёт, в котором приводятся код DAG, схема задач, снимки веб-интерфейса с результатами запуска, описание возникших ошибок и способов их устранения, а также выводы.

Пример решения:
Задание: Создать DAG для ежедневной обработки файла с данными о продажах.
Дано: Каждый день в папку /data/input поступает файл sales.csv. Необходимо дождаться появления файла, подсчитать количество записей, сохранить результат и вывести сообщение о завершении обработки. Конвейер должен запускаться ежедневно в 06:00.

Решение по этапам:
1. Определяем задачи конвейера: ожидание файла, подсчёт записей, вывод отчёта, архивирование файла.
2. Проверяем работу Airflow и создаём подключение fs_default к локальной файловой системе и переменную input_path со значением /data/input.
3. Создаём файл sales_daily.py в папке dags. Задаём default_args: owner — student, retries — 2, retry_delay — 5 минут.
4. Создаём DAG с параметрами: dag_id — sales_daily, start_date — текущая дата, schedule — cron-выражение 0 6 * * *, catchup — False.
5. Создаём задачу wait_file с FileSensor, которая проверяет наличие файла sales.csv каждые 60 секунд с максимальным временем ожидания 1 час в режиме reschedule.
6. Создаём задачу count_rows с PythonOperator: функция читает файл с помощью Pandas и возвращает количество записей, которое автоматически сохраняется в XCom.
7. Создаём задачу report с PythonOperator: функция получает количество записей через xcom_pull() и выводит сообщение в журнал.
8. Создаём задачу archive с BashOperator, которая перемещает обработанный файл в папку архива с добавлением даты {{ ds }} в имя файла.
9. Задаём зависимости: wait_file >> count_rows >> report >> archive.
10. Запускаем DAG вручную в веб-интерфейсе, проверяем, что все задачи завершились со статусом success, просматриваем журналы и значение XCom задачи count_rows.

Ответ: Создан DAG sales_daily, который ежедневно в 06:00 ожидает появления файла с помощью FileSensor, подсчитывает количество записей с помощью PythonOperator, передаёт результат через XCom, выводит отчёт и архивирует файл с помощью BashOperator. Зависимости и расписание заданы корректно, все задачи выполнены успешно, DAG готов к регулярному использованию.

Критерии оценки практических заданий:
1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Тема 2.3. Распределенная обработка данных: основы Apache Spark и PySpark.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.10. Осуществлять мониторинг и оркестрацию конвейеров обработки данных

Вопросы к устному опросу по теме:
1. Что такое распределённые вычисления?
Ответ: Распределённые вычисления — это способ обработки данных, при котором задача разбивается на части, выполняемые параллельно на нескольких компьютерах (узлах), объединённых в кластер. Результаты, полученные на отдельных узлах, затем объединяются в общий результат.

2. Для чего применяются распределённые вычисления при обработке данных?
Ответ: Распределённые вычисления применяются, когда объём данных превышает возможности одного компьютера по памяти, дисковому пространству или скорости обработки. Они позволяют обрабатывать терабайты данных за приемлемое время и наращивать мощность системы добавлением новых узлов.

3. Что такое кластер в контексте распределённых вычислений?
Ответ: Кластер — это группа компьютеров, соединённых сетью и работающих совместно как единая вычислительная система. Каждый компьютер кластера называется узлом. Обычно в кластере есть управляющий узел и рабочие узлы, выполняющие вычисления.

4. Что такое горизонтальное и вертикальное масштабирование?
Ответ: Вертикальное масштабирование — это увеличение мощности одного компьютера за счёт добавления памяти, процессоров или дисков. Горизонтальное масштабирование — это добавление в кластер новых компьютеров. Распределённые системы, такие как Spark, рассчитаны на горизонтальное масштабирование, которое практически не имеет предела.

5. Что такое Apache Spark?
Ответ: Apache Spark — это платформа с открытым исходным кодом для распределённой обработки больших данных. Она поддерживает пакетную и потоковую обработку, SQL-запросы, машинное обучение и работу с графами. Spark выполняет вычисления преимущественно в оперативной памяти, поэтому работает значительно быстрее систем, хранящих промежуточные результаты на диске.

6. Из каких модулей состоит Apache Spark?
Ответ: Spark включает ядро Spark Core, отвечающее за распределённое выполнение задач, модуль Spark SQL для работы со структурированными данными, Structured Streaming для потоковой обработки, библиотеку машинного обучения MLlib и модуль GraphX для обработки графов.

7. Из каких основных компонентов состоит архитектура Spark?
Ответ: Архитектура Spark включает драйвер (Driver), исполнители (Executors) и менеджер кластера (Cluster Manager). Драйвер управляет выполнением приложения, исполнители выполняют вычисления на рабочих узлах, а менеджер кластера выделяет ресурсы для приложения.

8. Какую роль выполняет драйвер (Driver) в Spark?
Ответ: Драйвер — это процесс, в котором выполняется основная программа Spark и создаётся SparkSession. Драйвер преобразует код в план выполнения, разбивает его на этапы и задачи, распределяет задачи между исполнителями, отслеживает их выполнение и собирает итоговые результаты.

9. Какую роль выполняют исполнители (Executors) в Spark?
Ответ: Исполнители — это процессы, запускаемые на рабочих узлах кластера. Они выполняют задачи, полученные от драйвера, хранят данные в памяти или на диске при кэшировании и возвращают результаты драйверу. Каждый исполнитель имеет выделенное количество ядер процессора и объём памяти.

10. Что такое менеджер кластера (Cluster Manager) и какие виды существуют?
Ответ: Менеджер кластера — это система, которая управляет ресурсами кластера и выделяет их приложениям Spark для запуска исполнителей. Используются встроенный менеджер Spark Standalone, Hadoop YARN и Kubernetes. Ранее также применялся Apache Mesos.

11. Что такое задание (job), этап (stage) и задача (task) в Spark?

Ответ: Задание — это совокупность вычислений, запускаемая одним действием. Задание делится на этапы, границы которых определяются операциями перераспределения данных между узлами. Этап, в свою очередь, состоит из задач, каждая из которых обрабатывает одну партицию данных на одном исполнителе.

12. Что такое RDD?
Ответ: RDD (Resilient Distributed Dataset, устойчивый распределённый набор данных) — это базовая структура данных Spark, представляющая собой неизменяемую коллекцию объектов, разделённую на партиции и распределённую по узлам кластера. RDD обрабатывается параллельно и поддерживает трансформации и действия.

13. Как RDD обеспечивает устойчивость к сбоям?
Ответ: Spark хранит для каждого RDD происхождение (lineage), то есть последовательность трансформаций, с помощью которых он был получен из исходных данных. Если партиция потеряна из-за сбоя узла, Spark не хранит копию, а заново вычисляет её по этой последовательности.

14. Что такое DataFrame в Spark?
Ответ: DataFrame — это распределённая таблица данных с именованными столбцами и схемой, в которой указаны типы столбцов. DataFrame похож на таблицу реляционной базы данных или DataFrame библиотеки Pandas, но его данные распределены по узлам кластера.

15. Чем DataFrame API отличается от RDD?
Ответ: RDD работает с произвольными объектами без схемы и требует явного описания каждой операции, что не позволяет Spark оптимизировать вычисления. DataFrame имеет схему и использует декларативные операции, поэтому встроенный оптимизатор Catalyst строит эффективный план выполнения. DataFrame работает быстрее, а код получается короче, поэтому он рекомендуется для большинства задач.

16. Что такое SparkSession и как она создаётся?
Ответ: SparkSession — это единая точка входа в приложение Spark, через которую читаются данные, создаются DataFrame и выполняются SQL-запросы. Она создаётся конструкцией SparkSession.builder.appName("имя").getOrCreate(). Через свойство sparkContext можно получить доступ к низкоуровневому API для работы с RDD.

17. Как прочитать и записать данные в PySpark?
Ответ: Для чтения используется объект spark.read с методами csv(), json(), parquet(), а также format() и load(). Например, spark.read.csv("файл.csv", header=True, inferSchema=True). Для записи используется метод write с указанием формата и режима записи, например df.write.mode("overwrite").parquet("путь").

18. Какие основные операции с DataFrame выполняются в PySpark?
Ответ: Основными операциями являются select() для выбора столбцов, filter() или where() для фильтрации строк, withColumn() для создания и изменения столбцов, groupBy() и agg() для группировки и агрегирования, orderBy() для сортировки, join() для объединения таблиц, dropDuplicates() для удаления дубликатов, fillna() и dropna() для обработки пропусков.

19. Что такое трансформации в Spark?
Ответ: Трансформации — это операции, которые создают новый набор данных на основе существующего, не изменяя исходный. К ним относятся select(), filter(), withColumn(), groupBy(), join(), map(). Трансформации выполняются лениво: они только добавляются в план вычислений и не запускают обработку данных.

20. Что такое действия в Spark?

Ответ: Действия — это операции, которые запускают фактическое выполнение всех накопленных трансформаций и возвращают результат драйверу или записывают данные во внешнее хранилище. К действиям относятся show(), count(), collect(), take(), first(), а также запись данных методом write.

21. Что такое ленивые вычисления и какие преимущества они дают?
Ответ: Ленивые вычисления означают, что Spark не выполняет трансформации сразу, а строит план вычислений и запускает его только при вызове действия. Это позволяет оптимизировать весь план целиком: объединять операции, не читать лишние столбцы, выполнять фильтрацию как можно раньше и избегать ненужных вычислений.

22. Чем узкие трансформации отличаются от широких?
Ответ: При узких трансформациях, например filter() или select(), каждая выходная партиция зависит только от одной входной, и данные не передаются между узлами. При широких трансформациях, например groupBy(), join() или orderBy(), данные необходимо перераспределить между узлами кластера. Такое перераспределение называется shuffle и является самой затратной операцией.

23. Что такое кэширование в Spark и когда его применяют?
Ответ: Кэширование — это сохранение промежуточного набора данных в памяти или на диске исполнителей, чтобы не вычислять его заново при повторном использовании. Кэширование применяют, когда один и тот же DataFrame используется в нескольких действиях, например при многократном анализе или итеративных алгоритмах.

24. Чем отличаются методы cache() и persist()?
Ответ: Метод cache() сохраняет данные с уровнем хранения по умолчанию, для DataFrame это память с переносом на диск. Метод persist() позволяет явно выбрать уровень хранения, например только память, только диск или память и диск. Для освобождения ресурсов после использования применяется метод unpersist().

25. Что такое партиция и партиционирование данных в Spark?
Ответ: Партиция — это часть распределённого набора данных, которая обрабатывается одной задачей на одном исполнителе. Партиционирование — это способ разделения данных на партиции. От количества и размера партиций зависит степень параллельности: слишком мало партиций не загружает кластер, слишком много создаёт лишние накладные расходы.

26. Чем метод repartition() отличается от coalesce()?
Ответ: Метод repartition() может увеличивать или уменьшать количество партиций и выполняет полное перераспределение данных между узлами, обеспечивая равномерное распределение. Метод coalesce() только уменьшает количество партиций, объединяя соседние без полного перераспределения, поэтому работает быстрее. Coalesce() часто применяют перед записью, чтобы уменьшить количество выходных файлов.

27. Что такое партиционирование при записи данных?
Ответ: При записи данных методом partitionBy() данные раскладываются по отдельным папкам в соответствии со значениями выбранного столбца, например по дате или региону. При последующем чтении с фильтром по этому столбцу Spark читает только нужные папки, что значительно ускоряет запросы.

28. Что такое broadcast-переменные?
Ответ: Broadcast-переменные — это переменные только для чтения, которые один раз передаются от драйвера на каждый исполнитель и хранятся там, вместо того чтобы передаваться вместе с каждой задачей. Они применяются для небольших справочников и словарей, используемых при обработке. Создаются методом sparkContext.broadcast().

29. Что такое broadcast-соединение таблиц?
Ответ: Broadcast-соединение — это способ объединения большой и маленькой таблиц, при котором маленькая таблица целиком копируется на все исполнители. Благодаря этому большую таблицу не нужно перераспределять между узлами, и операция shuffle не выполняется. Для явного указания используется функция broadcast() из модуля pyspark.sql.functions.

30. Что такое Spark SQL и как его использовать?
Ответ: Spark SQL — это модуль для обработки структурированных данных с помощью языка SQL. Чтобы выполнить SQL-запрос к DataFrame, его регистрируют как временное представление методом createOrReplaceTempView("имя"), а затем выполняют запрос функцией spark.sql("SELECT ..."). Результатом запроса является новый DataFrame, а запросы оптимизируются тем же оптимизатором Catalyst.

Критерии оценки устного опроса:

- оценка «отлично» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов и смог ответить на уточняющие вопросы;
- оценка «хорошо» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов, но не смог ответить на уточняющие вопросы;
- оценка «удовлетворительно» выставляется студенту, если он смог изложить суть основного вопроса, но не смог ответить по существу дополнительных вопросов;
- оценка «неудовлетворительно» выставляется студенту, если он не смог изложить суть основного вопроса.

Тема 2.4. Потоковая обработка данных: Kafka и Spark Streaming.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.10. Осуществлять мониторинг и оркестрацию конвейеров обработки данных

Вопросы к устному опросу по теме:

1. Что такое пакетная обработка данных (batch processing)?
Ответ: Пакетная обработка — это способ обработки, при котором данные сначала накапливаются за определённый период, а затем обрабатываются целиком одной порцией, обычно по расписанию, например раз в час или раз в сутки. Результат становится доступен только после завершения обработки всей порции.

2. Что такое потоковая обработка данных (stream processing)?
Ответ: Потоковая обработка — это способ обработки, при котором данные обрабатываются непрерывно, по мере их поступления, практически в реальном времени. Поток данных считается бесконечным, а результаты обновляются постоянно с минимальной задержкой.

3. В чём основные различия между пакетной и потоковой обработкой?
Ответ: Пакетная обработка работает с ограниченным набором данных, имеет большую задержку

получения результата, проще в реализации и подходит для отчётов и обучения моделей. Потоковая обработка работает с бесконечным потоком, даёт результат за секунды или миллисекунды, сложнее в реализации и подходит для мониторинга, выявления мошенничества и работы с данными датчиков.

4. Приведите примеры задач, требующих потоковой обработки.
Ответ: Потоковая обработка требуется для выявления мошеннических банковских операций в момент их совершения, мониторинга работы серверов и оборудования, обработки данных датчиков IoT, отслеживания действий пользователей на сайте, формирования рекомендаций в реальном времени и анализа биржевых котировок.

5. Что такое Apache Kafka?
Ответ: Apache Kafka — это распределённая платформа для потоковой передачи событий. Она принимает сообщения от источников, надёжно хранит их в течение заданного времени и передаёт их потребителям. Kafka обеспечивает высокую пропускную способность, масштабируемость и устойчивость к сбоям.

6. Что такое сообщение (событие) в Kafka?
Ответ: Сообщение — это единица данных, передаваемая через Kafka. Оно состоит из ключа, значения, временной метки и необязательных заголовков. Значение содержит сами данные, например информацию о покупке в формате JSON, а ключ используется для распределения сообщений по партициям.

7. Что такое продюсер (producer) в Kafka?
Ответ: Продюсер — это приложение, которое создаёт сообщения и отправляет их в топики Kafka. Продюсер определяет, в какую партицию топика попадёт сообщение: по ключу сообщения, по заданному правилу или равномерно, если ключ не указан.

8. Что такое консьюмер (consumer) в Kafka?
Ответ: Консьюмер — это приложение, которое подписывается на один или несколько топиков и читает из них сообщения. Консьюмер сам отслеживает, какие сообщения он уже прочитал, сохраняя смещение (offset), что позволяет продолжить чтение с нужного места после перезапуска.

9. Что такое группа консьюмеров (consumer group)?
Ответ: Группа консьюмеров — это несколько консьюмеров, совместно читающих один топик. Партиции топика распределяются между участниками группы так, что каждая партиция читается только одним консьюмером группы. Это позволяет обрабатывать поток параллельно и масштабировать обработку.

10. Что такое топик (topic) в Kafka?
Ответ: Топик — это именованная категория или канал, в который продюсеры записывают сообщения и из которого консьюмеры их читают. Например, отдельные топики могут создаваться для заказов, платежей и действий пользователей. Сообщения в топике хранятся в течение заданного времени независимо от того, прочитаны они или нет.

11. Что такое партиция (partition) в Kafka?
Ответ: Партиция — это часть топика, представляющая собой упорядоченный журнал сообщений, в который новые сообщения только добавляются в конец. Топик делится на несколько партиций, которые могут храниться на разных серверах, что обеспечивает параллельную запись и чтение. Порядок сообщений гарантируется только внутри одной партиции.

12. Что такое смещение (offset) в Kafka?
Ответ: Смещение — это порядковый номер сообщения внутри партиции. Каждое новое

сообщение получает следующее по порядку смещение. Консьюмеры сохраняют смещение последнего обработанного сообщения, чтобы после сбоя или перезапуска продолжить чтение с нужной позиции.

13. Что такое брокер (broker) в Kafka?
Ответ: Брокер — это сервер Kafka, который принимает сообщения от продюсеров, хранит партиции топиков на диске и передаёт сообщения консьюмерам. Несколько брокеров объединяются в кластер Kafka, при этом партиции распределяются между брокерами.

14. Как обеспечивается отказоустойчивость в Kafka?
Ответ: Каждая партиция копируется на несколько брокеров в соответствии с коэффициентом репликации. Одна копия является ведущей (leader) и обрабатывает запись и чтение, остальные (followers) синхронизируют данные с ней. При отказе брокера с ведущей копией одна из реплик становится новой ведущей, и данные не теряются.

15. Какую роль выполняют ZooKeeper и KRaft в Kafka?
Ответ: ZooKeeper — это внешняя служба, которая ранее использовалась для хранения метаданных кластера Kafka и выбора управляющего брокера. В современных версиях Kafka ZooKeeper заменён встроенным механизмом KRaft, при котором метаданные хранят и согласуют сами брокеры, что упрощает развёртывание.

16. Какие гарантии доставки сообщений существуют в потоковых системах?
Ответ: Существуют три гарантии: at-most-once (не более одного раза), при которой сообщения могут теряться, но не дублируются; at-least-once (как минимум один раз), при которой сообщения не теряются, но могут дублироваться; exactly-once (ровно один раз), при которой каждое сообщение обрабатывается ровно один раз.

17. Что такое Spark Streaming?
Ответ: Spark Streaming — это возможности Apache Spark для обработки потоков данных. Исторически первым был модуль на основе DStream API, а затем появился Structured Streaming, построенный на DataFrame и Spark SQL. Spark позволяет использовать один и тот же код и подходы для пакетной и потоковой обработки.

18. Что такое микробатчинг?
Ответ: Микробатчинг — это способ потоковой обработки, при котором непрерывный поток данных разбивается на небольшие порции (микробатчи) за короткие интервалы времени, например за одну секунду, и каждая порция обрабатывается как небольшая пакетная задача. Этот режим используется в Spark по умолчанию и даёт задержку от сотен миллисекунд до нескольких секунд.

19. Что такое непрерывная обработка (continuous processing) в Spark?
Ответ: Непрерывная обработка — это экспериментальный режим Structured Streaming, при котором записи обрабатываются по одной непрерывно работающими задачами, без разбиения на микробатчи. Он обеспечивает задержку около одной миллисекунды, но поддерживает ограниченный набор операций и гарантию at-least-once.

20. Что такое DStream API?
Ответ: DStream (Discretized Stream) — это первоначальный API потоковой обработки в Spark, в котором поток представляется как последовательность RDD, каждая из которых содержит данные за один интервал времени. DStream API является устаревшим и не развивается, для новых проектов рекомендуется использовать Structured Streaming.

21. Что такое Structured Streaming?
Ответ: Structured Streaming — это современный механизм потоковой обработки в Spark, в

котором поток данных рассматривается как таблица, к которой непрерывно добавляются новые строки. Запросы к потоку записываются так же, как к обычному DataFrame, а Spark сам выполняет их пошагово по мере поступления данных.

22. Чем Structured Streaming отличается от DStream API?
Ответ: Structured Streaming работает с DataFrame и SQL, использует оптимизатор Catalyst, поддерживает обработку по времени события, окна и watermark, а также обеспечивает гарантию exactly-once при поддерживающих источниках и приёмниках. DStream API работает с RDD, учитывает только время поступления данных в систему и сложнее в использовании.

23. Какие режимы вывода результатов поддерживает Structured Streaming?
Ответ: Поддерживаются три режима вывода: append — в результат добавляются только новые строки, complete — при каждом обновлении выводится вся таблица результатов целиком, update — выводятся только строки, изменившиеся с прошлого обновления. Выбор режима зависит от типа запроса и приёмника данных.

24. Что такое контрольные точки (checkpoint) в Structured Streaming?
Ответ: Контрольные точки — это сохранение в надёжное хранилище информации о ходе обработки потока: прочитанных смещениях и промежуточном состоянии агрегаций. При сбое или перезапуске приложение восстанавливается с последней контрольной точки без потери и повторной обработки данных. Путь задаётся параметром checkpointLocation.

25. Что такое оконные операции в потоковой обработке?
Ответ: Оконные операции — это вычисления над данными, сгруппированными по временным интервалам (окнам), например подсчёт количества заказов за каждые 5 минут. В Spark окна задаются функцией window() с указанием столбца времени, длины окна и при необходимости шага сдвига.

26. Какие виды окон существуют?
Ответ: Существуют фиксированные (tumbling) окна, которые идут подряд и не пересекаются; скользящие (sliding) окна, которые сдвигаются с заданным шагом и могут пересекаться; сессионные (session) окна, длина которых определяется активностью и которые закрываются после заданного периода без событий.

27. Чем время события отличается от времени обработки?
Ответ: Время события — это момент, когда событие произошло в источнике, например время покупки, записанное в сообщении. Время обработки — это момент, когда событие поступило в систему обработки. Из-за задержек в сети события могут приходить с опозданием и не по порядку, поэтому для точных результатов используется время события.

28. Что такое watermark?
Ответ: Watermark — это механизм обработки опоздавших данных, задающий, насколько долго система ждёт событий для уже прошедших временных окон. Например, watermark 10 минут означает, что события, опоздавшие более чем на 10 минут, отбрасываются. Это позволяет завершать окна и освобождать память. В Spark задаётся методом withWatermark().

29. Как прочитать данные из Kafka в Spark Structured Streaming?
Ответ: Для чтения используется spark.readStream.format("kafka") с параметрами kafka.bootstrap.servers (адреса брокеров) и subscribe (имя топика), после чего вызывается load(). Полученный DataFrame содержит столбцы key, value, topic, partition, offset и timestamp. Значение value передаётся в двоичном виде, поэтому его преобразуют в строку и разбирают, например функцией from_json(). Для работы требуется пакет spark-sql-kafka.

30. Как записать результаты потоковой обработки в Kafka или другое хранилище?

Ответ: Для записи используется метод writeStream с указанием формата (kafka, parquet, console и другие), режима вывода, пути контрольных точек и периодичности запуска (trigger), после чего вызывается start(). При записи в Kafka указываются адреса брокеров и топик, а данные должны содержать столбец value. Так строится конвейер Kafka → Spark → Kafka или хранилище данных.

Критерии оценки устного опроса:

- оценка «отлично» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов и смог ответить на уточняющие вопросы;
- оценка «хорошо» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов, но не смог ответить на уточняющие вопросы;
- оценка «удовлетворительно» выставляется студенту, если он смог изложить суть основного вопроса, но не смог ответить по существу дополнительных вопросов;
- оценка «неудовлетворительно» выставляется студенту, если он не смог изложить суть основного вопроса.

Практическое занятие №4. Обработка больших данных с PySpark и интеграция в Airflow.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.10. Осуществлять мониторинг и оркестрацию конвейеров обработки данных

Цель: Научиться разрабатывать скрипт обработки больших данных на PySpark и встраивать его в конвейер Apache Airflow для автоматического запуска по расписанию с контролем выполнения.

Задание:
1. Определить задачу обработки данных и исходный набор данных.
2. Разработать скрипт обработки данных на PySpark.
3. Выполнить в скрипте чтение, очистку, преобразование и агрегирование данных.
4. Применить приёмы оптимизации: кэширование, партиционирование, broadcast-соединение.
5. Сохранить результат обработки в формате Parquet.
6. Создать DAG в Airflow для запуска скрипта PySpark.
7. Запустить DAG и проверить результаты обработки и журналы.
8. Сделать вывод о работоспособности конвейера.

Этапы выполнения:
Этап 1. Анализ задачи и данных.
Определяется, какие данные обрабатываются, их объём, формат и структура, а также какой результат нужно получить. Определяется периодичность обработки.

Этап 2. Разработка скрипта PySpark.
Создаётся файл Python, в котором создаётся SparkSession, а путь к входным данным и дата обработки принимаются как параметры командной строки. Это позволяет запускать скрипт из Airflow для разных дат.

Этап 3. Чтение и очистка данных.
Данные читаются с помощью spark.read с указанием формата и схемы. Удаляются дубликаты,

обрабатываются пропуски, приводятся типы данных, отфильтровываются некорректные записи.

Этап 4. Преобразование и агрегирование.
Добавляются вычисляемые столбцы, выполняется соединение со справочниками, группировка и расчёт показателей. Для соединения с небольшим справочником применяется broadcast-соединение, для повторно используемых данных — кэширование.

Этап 5. Сохранение результата.
Результат записывается в формате Parquet с партиционированием по дате. Перед записью при необходимости уменьшается количество партиций методом coalesce(), чтобы не создавать множество мелких файлов.

Этап 6. Интеграция в Airflow.
Создаётся DAG, в котором скрипт PySpark запускается с помощью SparkSubmitOperator из провайдера Apache Spark или BashOperator с командой spark-submit. В DAG добавляются задачи проверки наличия входных данных и проверки результата, задаются зависимости и расписание.

Этап 7. Запуск и проверка.
DAG запускается в веб-интерфейсе Airflow. Проверяется успешность выполнения задач, просматриваются журналы задачи Spark, проверяется наличие и содержимое выходных файлов. При необходимости используется веб-интерфейс Spark для анализа этапов и задач.

Этап 8. Оформление результата.
Оформляется отчёт, в котором приводятся код скрипта PySpark и DAG, схема конвейера, результаты запуска, описание применённых приёмов оптимизации и выводы.

Пример решения:
Задание: Организовать ежедневную обработку журнала покупок интернет-магазина с помощью PySpark и Airflow.
Дано: Каждый день в хранилище поступает файл purchases_DATA.csv объёмом несколько гигабайт с полями: идентификатор заказа, идентификатор пользователя, идентификатор товара, количество, цена, время покупки. Также имеется небольшой справочник товаров products.csv с категориями. Необходимо ежедневно рассчитывать выручку и количество заказов по категориям товаров.

Решение по этапам:
1. Определяем задачу — ежедневная агрегация выручки и количества заказов по категориям, запуск в 03:00.
2. Создаём скрипт sales_agg.py, принимающий параметры: путь к данным и дату обработки. В начале скрипта создаём SparkSession.
3. Читаем файл покупок с заданной схемой, удаляем дубликаты по идентификатору заказа, удаляем записи с пустыми полями и записи с количеством или ценой меньше либо равными нулю.
4. Добавляем столбец revenue как произведение количества на цену.
5. Читаем справочник товаров и соединяем его с покупками с помощью broadcast-соединения, так как справочник небольшой.
6. Кэшируем полученный DataFrame методом cache(), так как он используется для двух расчётов: агрегации по категориям и подсчёта общего итога.
7. Группируем данные по категории и рассчитываем сумму выручки и количество уникальных заказов.
8. Записываем результат в формате Parquet с партиционированием по дате, предварительно уменьшив количество партиций методом coalesce().
9. Создаём DAG sales_spark_daily с расписанием 0 3 * * *, в котором задача check_input проверяет наличие входного файла, задача spark_job запускает скрипт через SparkSubmitOperator

с передачей даты {{ ds }}, задача check_output проверяет наличие выходных файлов. Зависимости: check_input >> spark_job >> check_output.
10. Запускаем DAG, проверяем, что все задачи завершились успешно, просматриваем журнал Spark и результат в выходной папке.

Ответ: Разработан скрипт PySpark, который читает и очищает журнал покупок, рассчитывает выручку, соединяет данные со справочником товаров с помощью broadcast-соединения, использует кэширование и сохраняет агрегированные результаты в формате Parquet с партиционированием по дате. Скрипт встроен в DAG Airflow с ежедневным запуском через SparkSubmitOperator и проверками входных и выходных данных. Конвейер работает корректно и готов к регулярному использованию.

Критерии оценки практических заданий:
1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Тема 2.5. Контейнеризация конвейеров: Docker и Docker Compose.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.10. Осуществлять мониторинг и оркестрацию конвейеров обработки данных

Вопросы к устному опросу по теме:
1. Что понимают под stream processing?
Ответ: Stream processing (потоковая обработка) — это непрерывная обработка данных по мере их поступления, без предварительного накопления в больших порциях. Данные рассматриваются как бесконечный поток событий, а результаты вычисляются и обновляются постоянно с минимальной задержкой.

2. Что такое событие в потоковой обработке?
Ответ: Событие — это запись о факте, произошедшем в определённый момент времени, например покупка, клик на сайте, показание датчика или вход пользователя в систему. Событие обычно содержит данные о произошедшем и временную метку.

3. Что такое поток данных?
Ответ: Поток данных — это непрерывная, упорядоченная во времени и потенциально бесконечная последовательность событий. В отличие от обычного набора данных, поток не имеет конца, поэтому его нельзя полностью загрузить и обработать за один раз.

4. Какие характеристики важны для систем потоковой обработки?
Ответ: Важными характеристиками являются задержка (время от появления события до получения результата), пропускная способность (количество событий, обрабатываемых в секунду), отказоустойчивость, масштабируемость и гарантии доставки сообщений.

5. Чем обработка без состояния отличается от обработки с состоянием?
Ответ: При обработке без состояния каждое событие обрабатывается независимо от других, например фильтрация или преобразование формата. При обработке с состоянием система хранит промежуточные результаты между событиями, например текущие суммы, счётчики или данные по окнам, что требуется для агрегаций и соединений.

6. В каких областях применяется потоковая обработка данных?
Ответ: Потоковая обработка применяется в банковской сфере для выявления мошенничества, в интернет-магазинах для рекомендаций и анализа поведения пользователей, в промышленности для мониторинга оборудования, в телекоммуникациях, логистике для отслеживания транспорта, а также для сбора и анализа журналов информационных систем.

7. Для чего в потоковой обработке используется Apache Kafka?
Ответ: Kafka используется как центральная платформа передачи событий между системами. Она принимает потоки событий от источников, надёжно хранит их и передаёт системам обработки. Kafka развязывает источники и потребители данных, так что они могут работать независимо друг от друга и с разной скоростью.

8. Как продюсер отправляет сообщения в Kafka?
Ответ: Продюсер подключается к брокерам Kafka, формирует сообщение с ключом и значением, преобразует его в байты (сериализует) и отправляет в указанный топик. Сообщения обычно отправляются пакетами для повышения производительности.

9. Как продюсер определяет партицию для сообщения?
Ответ: Если у сообщения указан ключ, партиция вычисляется по хеш-значению ключа, поэтому сообщения с одинаковым ключом всегда попадают в одну партицию и сохраняют порядок. Если ключ не указан, сообщения распределяются по партициям равномерно. Также можно явно указать номер партиции.

10. Что такое параметр acks у продюсера Kafka?
Ответ: Параметр acks определяет, когда продюсер считает сообщение успешно отправленным. При значении 0 продюсер не ждёт подтверждения, при значении 1 ждёт подтверждения от ведущей реплики, при значении all — подтверждения от всех синхронизированных реплик. Значение all обеспечивает наибольшую надёжность.

11. Как консьюмер читает сообщения из Kafka?
Ответ: Консьюмер подписывается на топик и периодически запрашивает у брокеров новые сообщения, начиная с сохранённого смещения. Полученные сообщения преобразуются из байтов в данные (десериализуются) и обрабатываются, после чего консьюмер фиксирует новое смещение.

12. Что такое фиксация смещения (commit offset)?
Ответ: Фиксация смещения — это сохранение в Kafka номера последнего обработанного консьюмером сообщения. Фиксация может выполняться автоматически через заданные интервалы или вручную после обработки. От момента фиксации зависит, будут ли сообщения теряться или обрабатываться повторно после сбоя.

13. Что такое отставание консьюмера (consumer lag)?
Ответ: Отставание консьюмера — это разница между смещением последнего сообщения в партиции и смещением последнего обработанного консьюмером сообщения. Растущее отставание означает, что консьюмер не успевает обрабатывать поток, и требуется увеличить количество консьюмеров или ускорить обработку.

14. Что такое топик и как он связан с партициями?
Ответ: Топик — это именованный поток сообщений одного типа, например заказы или платежи. Топик делится на партиции — упорядоченные журналы, хранящиеся на разных брокерах. Количество партиций определяет максимальную степень параллельности чтения топика одной группой консьюмеров.

15. Как долго Kafka хранит сообщения?
Ответ: Kafka хранит сообщения в течение заданного срока хранения (retention) или до достижения заданного объёма, независимо от того, прочитаны они или нет. По умолчанию срок хранения составляет 7 дней. Это позволяет повторно читать данные, подключать новых консьюмеров и восстанавливаться после сбоев.

16. Что такое сжатие журнала (log compaction) в Kafka?
Ответ: Сжатие журнала — это режим хранения топика, при котором для каждого ключа сохраняется только последнее сообщение, а более старые сообщения с тем же ключом удаляются. Он применяется, когда важно только текущее состояние объекта, например последний адрес клиента или актуальная цена товара.

17. Что такое Apache Spark Streaming?
Ответ: Spark Streaming — это средства Apache Spark для обработки потоков данных в реальном времени. Они позволяют получать данные из Kafka, файлов и сокетов, обрабатывать их теми же операциями, что и пакетные данные, и записывать результаты в хранилища, базы данных или обратно в Kafka.

18. Какие API для потоковой обработки существуют в Spark?
Ответ: В Spark существуют два API: устаревший DStream API, основанный на последовательности RDD, и современный Structured Streaming, основанный на DataFrame и Spark SQL. Для новых проектов рекомендуется Structured Streaming, так как он проще, производительнее и поддерживает обработку по времени события.

19. Какова модель обработки данных в Structured Streaming?
Ответ: В Structured Streaming поток рассматривается как неограниченная таблица, к которой постоянно добавляются новые строки. Разработчик описывает запрос к этой таблице так же, как к обычному DataFrame, а Spark выполняет его пошагово, обрабатывая только новые данные и обновляя таблицу результатов.

20. Какие источники и приёмники данных поддерживает Structured Streaming?
Ответ: В качестве источников поддерживаются Kafka, файлы в папке (CSV, JSON, Parquet) и сокеты для тестирования. В качестве приёмников используются Kafka, файлы, консоль для отладки, память, а также произвольные системы через методы foreach() и foreachBatch(), например запись в базу данных.

21. Что такое триггер (trigger) в Structured Streaming?
Ответ: Триггер определяет, как часто запускается обработка новых данных. По умолчанию следующий микробатч запускается сразу после завершения предыдущего. Можно задать фиксированный интервал (processingTime), однократную обработку всех доступных данных (availableNow) или непрерывный режим (continuous).

22. Что такое метод foreachBatch()?
Ответ: Метод foreachBatch() позволяет применить к каждому микробатчу произвольную функцию, которая получает данные микробатча как обычный DataFrame. Это удобно для записи в системы, не поддерживаемые напрямую, например в реляционную базу данных, или для записи одного результата в несколько мест.

23. Что такое окно в потоковой обработке?
Ответ: Окно — это временной интервал, по которому группируются события потока для вычисления агрегатов. Поскольку поток бесконечен, подсчитать итог по всем данным невозможно, поэтому вычисления выполняются в пределах окон, например количество заказов за каждую минуту или средняя температура за 10 минут.

24. Чем фиксированные окна отличаются от скользящих?
Ответ: Фиксированные (tumbling) окна имеют одинаковую длину, следуют одно за другим и не пересекаются, поэтому каждое событие попадает ровно в одно окно. Скользящие (sliding) окна сдвигаются с шагом меньше их длины и пересекаются, поэтому одно событие может попасть в несколько окон. Например, окно 10 минут с шагом 5 минут.

25. Что такое сессионные окна?
Ответ: Сессионные окна — это окна переменной длины, которые группируют события одного пользователя или объекта, идущие друг за другом с небольшими перерывами. Окно закрывается, если в течение заданного времени не поступает новых событий. Такие окна используются для анализа сессий пользователей на сайте. В Spark задаются функцией session_window().

26. Как задаются окна в Spark Structured Streaming?
Ответ: Окна задаются функцией window() внутри groupBy(). В функцию передаются столбец с временем события, длина окна и при необходимости шаг сдвига, например groupBy(window("event_time", "10 minutes", "5 minutes")). После группировки применяются агрегирующие функции, например count() или avg().

27. Почему события в потоке могут приходить с опозданием?
Ответ: События могут задерживаться из-за сетевых задержек, временного отключения устройств от сети, буферизации данных на стороне источника, сбоев и повторных отправок. В результате событие с более ранним временем может прийти позже событий с более поздним временем.

28. Что такое watermark и для чего он нужен?
Ответ: Watermark — это пороговое значение времени, определяющее, как долго система ожидает опоздавшие события. Он вычисляется как максимальное время события, полученное системой, минус допустимая задержка. События старше watermark отбрасываются, а окна до этого момента считаются завершёнными, что позволяет выдавать окончательные результаты и освобождать память.

29. Как watermark задаётся в Spark Structured Streaming?
Ответ: Watermark задаётся методом withWatermark() с указанием столбца времени события и допустимой задержки, например withWatermark("event_time", "10 minutes"). Метод вызывается до оконной агрегации. Watermark также используется при удалении дубликатов и соединении двух потоков.

30. Как выбрать величину watermark?
Ответ: Величина watermark выбирается как компромисс между полнотой и задержкой результатов. Большой watermark позволяет учесть больше опоздавших событий, но увеличивает задержку выдачи итогов и объём хранимого состояния. Маленький watermark даёт результаты быстрее, но часть опоздавших событий будет отброшена. Величину определяют, анализируя реальные задержки данных.

Критерии оценки устного опроса:

- оценка «отлично» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов и смог ответить на уточняющие вопросы;

- оценка «хорошо» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов, но не смог ответить на уточняющие вопросы;
- оценка «удовлетворительно» выставляется студенту, если он смог изложить суть основного вопроса, но не смог ответить по существу дополнительных вопросов;
- оценка «неудовлетворительно» выставляется студенту, если он не смог изложить суть основного вопроса.

Практическое занятие №5. Создание Dockerfile для Python-конвейера. Оптимизация размера образа. Настройка Docker Compose для связки: Airflow + PostgreSQL + Spark. Запуск многокомпонентного конвейера. Работа с volumes для сохранения данных.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.10. Осуществлять мониторинг и оркестрацию конвейеров обработки данных

Цель: Научиться создавать Dockerfile для Python-конвейера обработки данных, оптимизировать размер Docker-образа, настраивать Docker Compose для совместной работы Airflow, PostgreSQL и Spark, запускать многокомпонентный конвейер и использовать volumes для сохранения данных.

Задание:
1. Определить состав компонентов конвейера и их назначение.
2. Создать Dockerfile для Python-конвейера.
3. Оптимизировать размер Docker-образа.
4. Собрать образ и проверить его работу.
5. Создать файл docker-compose.yaml для связки Airflow, PostgreSQL и Spark.
6. Настроить volumes для сохранения данных, DAG и журналов.
7. Запустить многокомпонентный конвейер и проверить взаимодействие сервисов.
8. Сделать вывод о работоспособности конвейера и сохранности данных.

Этапы выполнения:
Этап 1. Анализ состава конвейера.
Определяются компоненты конвейера: Python-скрипты обработки данных, Airflow для оркестрации, PostgreSQL в качестве базы метаданных Airflow и хранилища результатов, Spark для обработки больших данных. Составляется схема взаимодействия компонентов.

Этап 2. Создание Dockerfile.
Создаётся файл Dockerfile, в котором указываются базовый образ Python, рабочий каталог, копирование файла зависимостей requirements.txt, установка зависимостей, копирование кода конвейера и команда запуска.

Этап 3. Оптимизация размера образа.
Для уменьшения размера используется облегчённый базовый образ, например python:3.11-slim, установка пакетов без кэша (pip install --no-cache-dir), объединение команд RUN, файл .dockerignore для исключения лишних файлов, а также многоэтапная сборка (multi-stage build). Файл зависимостей копируется отдельно до кода, чтобы использовать кэш слоёв при повторной сборке.

Этап 4. Сборка и проверка образа.
Образ собирается командой docker build с указанием имени и тега. Размер образа проверяется командой docker images. Контейнер запускается командой docker run, проверяется корректность выполнения конвейера.

Этап 5. Создание docker-compose.yaml.
В файле описываются сервисы: postgres (база данных), компоненты Airflow (инициализация, планировщик, веб-интерфейс), spark-master и spark-worker. Для каждого сервиса указываются образ, переменные окружения, порты, зависимости запуска (depends_on) и проверки работоспособности (healthcheck).

Этап 6. Настройка volumes.
Создаются именованные тома для данных PostgreSQL, чтобы база не терялась при пересоздании контейнеров, и монтируются папки хоста с DAG, журналами и обрабатываемыми данными в контейнеры Airflow и Spark.

Этап 7. Запуск и проверка конвейера.
Конвейер запускается командой docker compose up -d. Состояние сервисов проверяется командой docker compose ps, журналы — командой docker compose logs. Проверяется доступ к веб-интерфейсам Airflow и Spark, запускается DAG, отправляющий задачу в Spark и сохраняющий результат в PostgreSQL. Затем контейнеры останавливаются командой docker compose down и запускаются снова для проверки сохранности данных.

Этап 8. Оформление результата.
Оформляется отчёт, в котором приводятся Dockerfile, файл docker-compose.yaml, размеры образа до и после оптимизации, результаты запуска конвейера, проверка сохранности данных и выводы.

Пример решения:
Задание: Контейнеризировать конвейер ежедневной обработки данных о продажах с использованием Airflow, PostgreSQL и Spark.
Дано: Имеется Python-скрипт обработки данных с зависимостями pandas, pyspark и psycopg2, DAG Airflow для его запуска и папка с исходными данными. Необходимо обеспечить запуск всех компонентов одной командой и сохранение данных базы при перезапуске.

Решение по этапам:
1. Определяем сервисы: postgres, airflow-init, airflow-scheduler, airflow-webserver, spark-master, spark-worker.
2. Создаём Dockerfile: базовый образ python:3.11 , рабочий каталог /app, копирование requirements.txt, установка зависимостей, копирование кода, команда запуска скрипта.
3. Собираем образ и фиксируем размер — около 1,6 ГБ.
4. Оптимизируем образ: заменяем базовый образ на python:3.11-slim, устанавливаем зависимости с параметром --no-cache-dir, объединяем команды RUN, добавляем файл .dockerignore с исключением папок данных, виртуального окружения и кэша. Повторная сборка даёт образ размером около 700 МБ.
5. Создаём docker-compose.yaml: сервис postgres на основе образа postgres:16 с переменными окружения для имени базы, пользователя и пароля и проверкой pg_isready; сервисы Airflow на основе официального образа apache/airflow с подключением к PostgreSQL через переменную окружения строки подключения; сервисы spark-master и spark-worker на основе образа Apache Spark.
6. Указываем, что сервисы Airflow запускаются только после успешной проверки работоспособности PostgreSQL (depends_on с условием service_healthy).
7. Настраиваем volumes: именованный том postgres_data для каталога данных PostgreSQL, папки ./dags, ./logs и ./data монтируются в контейнеры Airflow и Spark.
8. Запускаем конвейер командой docker compose up -d, проверяем, что все сервисы имеют

состояние running или healthy, открываем веб-интерфейс Airflow на порту 8080 и веб-интерфейс Spark на порту 8081.
9. Запускаем DAG обработки продаж, проверяем выполнение задачи Spark и наличие результатов в таблице PostgreSQL.
10. Останавливаем контейнеры командой docker compose down, запускаем снова и убеждаемся, что история запусков DAG и данные в PostgreSQL сохранились.

Ответ: Создан Dockerfile для Python-конвейера, размер образа уменьшен примерно с 1,6 ГБ до 700 МБ за счёт облегчённого базового образа, установки без кэша, объединения команд и файла .dockerignore. Настроен docker-compose.yaml, связывающий Airflow, PostgreSQL и Spark, с проверками работоспособности и порядком запуска сервисов. Благодаря volumes данные базы, DAG и журналы сохраняются при перезапуске контейнеров. Многокомпонентный конвейер запускается одной командой и работает корректно.

Критерии оценки практических заданий:
1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Тема 3.1. Концепция Data Quality: метрики, измерения и бизнес-правила.
Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.11. Осуществлять контроль качества данных в конвейерах обработки

Вопросы к устному опросу по теме:
1. Что такое качество данных (Data Quality)?
Ответ: Качество данных — это степень соответствия данных требованиям, предъявляемым к ним для конкретного использования. Данные считаются качественными, если они точные, полные, согласованные, своевременные, уникальные и корректные, то есть пригодны для принятия решений, аналитики и обучения моделей.

2. Почему качество данных важно для организации?
Ответ: На основе данных принимаются управленческие решения, формируются отчёты, обслуживаются клиенты и обучаются модели ИИ. Некачественные данные приводят к ошибочным решениям, финансовым потерям, недовольству клиентов, нарушениям требований законодательства и снижению доверия к информационным системам.

3. Что такое измерения (dimensions) качества данных?
Ответ: Измерения качества данных — это отдельные характеристики, по которым оценивается качество данных. К основным измерениям относятся точность, полнота, согласованность, своевременность, уникальность и валидность. Каждое измерение отражает определённый аспект пригодности данных.

4. Что такое точность данных (accuracy)?

Ответ: Точность — это степень соответствия данных реальным значениям объектов и событий, которые они описывают. Например, адрес клиента в базе данных точен, если совпадает с его фактическим адресом. Точность проверяется сравнением с эталонными источниками или реальными данными.

5. Что такое полнота данных (completeness)?
Ответ: Полнота — это степень наличия всех необходимых данных. Данные неполны, если отсутствуют обязательные значения в полях или отсутствуют целые записи. Например, у клиента не указан номер телефона или в таблице отсутствуют продажи за один из дней.

6. Что такое согласованность данных (consistency)?
Ответ: Согласованность — это отсутствие противоречий в данных внутри одного набора или между разными системами. Например, данные о клиенте в CRM-системе и в бухгалтерской системе должны совпадать, а сумма заказа должна быть равна сумме стоимости его позиций.

7. Что такое своевременность данных (timeliness)?
Ответ: Своевременность — это степень актуальности данных и их доступности в тот момент, когда они нужны. Данные несвоевременны, если поступают с опозданием или устарели. Например, остатки товаров на складе, обновляемые раз в неделю, не подходят для оперативного управления продажами.

8. Что такое уникальность данных (uniqueness)?
Ответ: Уникальность — это отсутствие дублирующихся записей об одном и том же объекте. Каждый объект реального мира, например клиент или заказ, должен быть представлен в данных ровно один раз. Дубликаты искажают подсчёты и статистику.

9. Что такое валидность данных (validity)?
Ответ: Валидность — это соответствие данных установленным правилам: формату, типу, допустимому диапазону значений и перечню разрешённых значений. Например, дата рождения должна быть корректной датой, электронная почта — содержать символ @, а возраст — находиться в разумных пределах.

10. Чем валидность данных отличается от точности?
Ответ: Валидность показывает, соответствует ли значение правилам, а точность — соответствует ли оно действительности. Например, номер телефона в правильном формате является валидным, но может быть неточным, если принадлежит другому человеку. Валидные данные не обязательно точны, а проверить точность обычно сложнее.

11. Что такое метрики качества данных?
Ответ: Метрики качества данных — это количественные показатели, позволяющие измерить качество данных по каждому измерению. Обычно они выражаются в процентах или долях, например доля заполненных значений, доля уникальных записей, доля записей, прошедших проверку правил.

12. Как рассчитывается метрика полноты?
Ответ: Метрика полноты рассчитывается как отношение количества заполненных значений к общему количеству значений, умноженное на 100%. Например, если из 1000 записей поле «телефон» заполнено в 950, полнота составляет 95%. Аналогично рассчитывается доля пропусков.

13. Как рассчитывается метрика уникальности?
Ответ: Метрика уникальности рассчитывается как отношение количества уникальных записей (или уникальных значений ключевого поля) к общему количеству записей, умноженное на 100%. Значение 100% означает отсутствие дубликатов.

14. Как рассчитывается метрика валидности?
Ответ: Метрика валидности рассчитывается как отношение количества записей, прошедших проверку правил валидации, к общему количеству проверенных записей, умноженное на 100%. Например, если 980 из 1000 адресов электронной почты имеют правильный формат, валидность составляет 98%.

15. Как оценивается своевременность данных?
Ответ: Своевременность оценивается по задержке поступления данных (разнице между временем события и временем появления данных в системе), по возрасту данных относительно текущего момента и по доле загрузок, выполненных в установленный срок. Например, данные за прошедшие сутки должны быть загружены до 06:00.

16. Что такое пороговые значения метрик качества?
Ответ: Пороговые значения — это минимально допустимые уровни метрик качества, установленные для конкретных данных. Например, полнота обязательных полей должна быть не ниже 99%, а доля дубликатов — не выше 0,1%. При нарушении порога данные считаются некачественными, и конвейер может остановиться или отправить уведомление.

17. Что такое интегральный показатель качества данных?
Ответ: Интегральный показатель — это обобщённая оценка качества данных, рассчитанная по нескольким метрикам, например как среднее или взвешенная сумма метрик полноты, валидности, уникальности и других. Он позволяет одним числом отслеживать общее состояние качества набора данных.

18. Что такое бизнес-правила валидации данных?
Ответ: Бизнес-правила валидации — это условия, которым должны соответствовать данные исходя из логики предметной области и требований бизнеса. Например, сумма заказа не может быть отрицательной, дата доставки не может быть раньше даты заказа, а скидка не может превышать 50%.

19. Что такое проверка диапазона (range check)?
Ответ: Проверка диапазона — это проверка того, что числовое значение или дата находятся в допустимых пределах. Например, возраст клиента должен быть от 18 до 120 лет, процент — от 0 до 100, температура датчика — в пределах технических возможностей оборудования, а дата события — не в будущем.

20. Что такое проверка формата (format check)?
Ответ: Проверка формата — это проверка того, что значение соответствует установленному шаблону записи. Например, номер телефона должен иметь вид +7XXXXXXXXXX, почтовый индекс — состоять из 6 цифр, дата — иметь формат ГГГГ-ММ-ДД. Для проверки формата часто используются регулярные выражения.

21. Что такое проверка ссылочной целостности (referential integrity)?
Ответ: Проверка ссылочной целостности — это проверка того, что значения внешних ключей в одной таблице существуют в связанной таблице. Например, каждый заказ должен ссылаться на существующего клиента, а каждая позиция заказа — на существующий товар. Записи, ссылающиеся на несуществующие объекты, называются «висячими».

22. Какие ещё виды проверок данных применяются?
Ответ: Кроме проверок диапазона, формата и ссылочной целостности применяются проверки на обязательность заполнения (not null), уникальность, допустимый перечень значений, тип данных, межполевые проверки (например, дата окончания позже даты начала), проверки агрегатов (например, сумма позиций равна итогу) и проверки объёма загруженных данных.

23. Что такое межполевая проверка?
Ответ: Межполевая проверка — это проверка логической связи между значениями нескольких полей одной записи. Например, если в поле «статус» указано «доставлен», поле «дата доставки» должно быть заполнено, а дата закрытия договора должна быть не раньше даты его заключения.

24. Где в конвейере данных выполняются проверки качества?
Ответ: Проверки качества выполняются на нескольких этапах: при поступлении данных из источника, после преобразования данных, перед загрузкой в хранилище и перед использованием данных для обучения моделей. Чем раньше обнаружена ошибка, тем дешевле её исправить и тем меньше систем она затронет.

25. Что делают с записями, не прошедшими проверку качества?
Ответ: Такие записи могут быть отклонены, исправлены автоматически по заданным правилам, помечены как сомнительные или перемещены в отдельную таблицу (карантин) для последующего анализа. При массовых нарушениях обработка останавливается, а ответственным отправляется уведомление.

26. Что означает принцип garbage in, garbage out?
Ответ: Принцип garbage in, garbage out («мусор на входе — мусор на выходе») означает, что качество результата обработки не может быть выше качества исходных данных. Если на вход системы подаются ошибочные данные, то и результаты будут ошибочными, независимо от сложности алгоритмов.

27. Как принцип garbage in, garbage out проявляется в машинном обучении?
Ответ: Модель машинного обучения выявляет закономерности в обучающих данных. Если данные содержат ошибки, пропуски, дубликаты, неверную разметку или смещения, модель усвоит эти ошибки и будет давать неточные или несправедливые прогнозы. Улучшение качества данных часто даёт больший эффект, чем усложнение модели.

28. Какие проблемы качества данных особенно опасны для моделей машинного обучения?
Ответ: Особенно опасны ошибки разметки целевой переменной, утечка данных, несбалансированность классов, систематические смещения в выборке, большое количество пропусков и выбросов, дубликаты между обучающей и тестовой выборками, а также изменение свойств данных со временем (дрейф данных).

29. Что такое профилирование данных?
Ответ: Профилирование данных — это анализ набора данных для получения его статистического описания: типов полей, количества пропусков, уникальных значений, минимальных и максимальных значений, распределений, частых значений и связей между полями. Профилирование помогает обнаружить проблемы качества и сформулировать правила проверки.

30. Какие инструменты применяются для контроля качества данных?
Ответ: Для контроля качества данных применяются библиотеки Great Expectations, Pandera и Soda, встроенные тесты dbt, библиотека Deequ для Spark, а также средства профилирования, например ydata-profiling. Они позволяют описывать правила проверки в виде кода, автоматически выполнять их в конвейере и формировать отчёты о качестве.

Критерии оценки устного опроса:

- оценка «отлично» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов и смог ответить на уточняющие вопросы;

- оценка «хорошо» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов, но не смог ответить на уточняющие вопросы;
- оценка «удовлетворительно» выставляется студенту, если он смог изложить суть основного вопроса, но не смог ответить по существу дополнительных вопросов;
- оценка «неудовлетворительно» выставляется студенту, если он не смог изложить суть основного вопроса.

Тема 3.2. Инструменты авто-валидации: Great Expectations, Pandera.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.11. Осуществлять контроль качества данных в конвейерах обработки

Вопросы к устному опросу по теме:

1. Что такое автоматическая валидация данных?
Ответ: Автоматическая валидация данных — это проверка данных на соответствие заданным правилам качества, выполняемая программно без участия человека. Правила описываются в виде кода или конфигурации и запускаются автоматически при каждом поступлении или обработке данных, а результаты фиксируются в отчётах.

2. Какие преимущества дают инструменты автоматической валидации?
Ответ: Инструменты автоматической валидации позволяют описывать правила качества один раз и применять их многократно, быстро обнаруживать ошибки в данных, останавливать конвейер при нарушении правил, формировать понятные отчёты, хранить правила в системе контроля версий и встраивать проверки в конвейеры и процессы CI/CD.

3. Что такое Great Expectations?
Ответ: Great Expectations (GX) — это библиотека Python с открытым исходным кодом для проверки, документирования и профилирования данных. В ней правила качества описываются в виде ожиданий (expectations), которые проверяются на данных из файлов, датафреймов Pandas и Spark и баз данных SQL.

4. Что такое Data Context в Great Expectations?
Ответ: Data Context — это основной объект Great Expectations, точка входа и центр управления всеми компонентами проекта. Через него настраиваются источники данных, хранятся наборы ожиданий, определения проверок, контрольные точки, результаты валидации и настройки документации. Контекст создаётся функцией get_context().

5. Какие виды Data Context существуют?
Ответ: Существуют временный (Ephemeral) контекст, который хранит настройки только в памяти на время работы программы, файловый (File) контекст, который сохраняет конфигурацию и результаты в папке проекта, и облачный (Cloud) контекст для работы с сервисом GX Cloud.

6. Что такое источник данных (Data Source) и актив данных (Data Asset) в Great Expectations?
Ответ: Источник данных описывает, откуда Great Expectations получает данные: из датафрейма Pandas, Spark, файловой системы или базы данных SQL. Актив данных — это конкретный набор

данных внутри источника, например таблица базы данных, файл или датафрейм. Для проверки из актива формируется пакет данных (Batch).

7. Что такое ожидание (Expectation) в Great Expectations?
Ответ: Ожидание — это проверяемое утверждение о данных, записанное в декларативной форме. Например, ожидание «значения столбца id не должны быть пустыми» или «значения столбца age должны быть от 18 до 100». Каждое ожидание имеет понятное название, начинающееся со слова expect, и параметры.

8. Что такое набор ожиданий (Expectation Suite)?
Ответ: Набор ожиданий — это именованная коллекция ожиданий, описывающая требования к качеству конкретного набора данных. Например, набор orders_suite может содержать все проверки для таблицы заказов. Наборы хранятся в Data Context и могут повторно применяться к новым порциям данных.

9. Что такое результаты валидации (Validation Results)?
Ответ: Результаты валидации — это отчёт о проверке данных набором ожиданий. Для каждого ожидания указывается, выполнено ли оно, сколько значений проверено, сколько значений не соответствует правилу, их процент и примеры неверных значений. Общий признак success показывает, прошли ли данные проверку целиком.

10. Что такое Validation Definition и Checkpoint в Great Expectations?
Ответ: Validation Definition связывает конкретные данные с набором ожиданий, то есть определяет, что и чем проверять. Checkpoint (контрольная точка) объединяет одно или несколько определений проверок и действия, выполняемые после проверки, например обновление документации или отправку уведомлений. Контрольная точка является основным способом запуска валидации в конвейере.

11. Что такое ожидания уровня столбца (column-level)?
Ответ: Ожидания уровня столбца проверяют значения отдельного столбца. К ним относятся проверки отсутствия пропусков (expect_column_values_to_not_be_null), диапазона значений (expect_column_values_to_be_between), уникальности (expect_column_values_to_be_unique), допустимого набора значений (expect_column_values_to_be_in_set), формата по регулярному выражению (expect_column_values_to_match_regex) и типа данных.

12. Что такое ожидания уровня таблицы (table-level)?
Ответ: Ожидания уровня таблицы проверяют свойства набора данных в целом. Например, expect_table_row_count_to_be_between проверяет количество строк, expect_table_columns_to_match_ordered_list — состав и порядок столбцов, expect_table_column_count_to_equal — количество столбцов. Они помогают обнаружить неполную загрузку или изменение структуры данных.

13. Что такое агрегатные ожидания?
Ответ: Агрегатные ожидания проверяют статистические характеристики столбца, а не отдельные значения. Например, expect_column_mean_to_be_between проверяет, что среднее значение находится в заданных пределах, expect_column_max_to_be_between — максимальное значение, expect_column_proportion_of_unique_values_to_be_between — долю уникальных значений. Они помогают выявлять аномальные изменения данных.

14. Для чего используется параметр mostly?
Ответ: Параметр mostly задаёт минимальную долю значений, которые должны удовлетворять ожиданию, чтобы проверка считалась успешной. Например, значение mostly=0.95 означает, что проверка пройдена, если правилу соответствуют не менее 95% значений. Это позволяет допускать небольшое количество отклонений.

15. Что такое пользовательские (custom) ожидания?
Ответ: Пользовательские ожидания — это ожидания, создаваемые разработчиком для проверки специфических бизнес-правил, которых нет среди встроенных. Они создаются путём наследования от базовых классов ожиданий с описанием логики проверки или в виде SQL-запроса, возвращающего строки, нарушающие правило.

16. Что такое Data Docs?
Ответ: Data Docs — это автоматически формируемая HTML-документация Great Expectations, в которой отображаются наборы ожиданий в понятном человеку виде и результаты всех проверок с указанием успешных и неуспешных ожиданий и примеров ошибочных значений. Data Docs служат одновременно документацией к данным и отчётом об их качестве.

17. Как сформировать и открыть Data Docs?
Ответ: Data Docs формируются автоматически действием обновления документации в контрольной точке или вручную методом build_data_docs() объекта Data Context. Открыть документацию в браузере можно методом open_data_docs(). Сайт документации можно размещать на локальном диске или в облачном хранилище для доступа всей команды.

18. Что такое CI/CD и как с ним связана валидация данных?
Ответ: CI/CD (непрерывная интеграция и непрерывная доставка) — это практика автоматической проверки, сборки и развёртывания изменений кода. Встраивание проверок данных в CI/CD позволяет автоматически запускать валидацию при изменении кода конвейера или правил качества и не допускать развёртывания изменений, ломающих данные.

19. Как интегрировать Great Expectations с CI/CD?
Ответ: Наборы ожиданий и конфигурация хранятся в репозитории Git. В конвейере CI/CD, например GitHub Actions или GitLab CI, устанавливаются зависимости и запускается скрипт с выполнением контрольной точки на тестовых данных. Если проверка не пройдена, скрипт завершается с ошибкой, и сборка останавливается. Отчёт Data Docs может сохраняться как результат сборки.

20. Как встроить проверки Great Expectations в конвейер данных?
Ответ: Проверка выполняется отдельным шагом конвейера, например задачей в Airflow или вызовом в Python-скрипте. После запуска контрольной точки анализируется признак success: при успешной проверке обработка продолжается, при неуспешной конвейер останавливается, данные отправляются в карантин, а ответственным направляется уведомление.

21. Что такое Pandera?
Ответ: Pandera — это библиотека Python для валидации табличных данных, в первую очередь датафреймов Pandas. Она позволяет описывать схему данных с типами столбцов и правилами проверки прямо в коде. Pandera также поддерживает Polars, PySpark и другие библиотеки обработки данных.

22. Как описывается схема данных в Pandera?
Ответ: Схема описывается объектом DataFrameSchema, в котором для каждого столбца создаётся объект Column с указанием типа данных, правил проверки (checks), допустимости пропусков (nullable) и уникальности (unique). Альтернативный способ — класс, наследуемый от DataFrameModel, в котором столбцы задаются как аннотированные поля.

23. Что такое Check в Pandera?
Ответ: Check — это правило проверки значений столбца или всего датафрейма. В Pandera есть встроенные проверки, например Check.ge() (больше или равно), Check.in_range() (в диапазоне), Check.isin() (из допустимого набора), Check.str_matches() (соответствие регулярному

выражению). Также можно задать собственную проверку в виде функции.

24. Как выполняется проверка данных в Pandera?
Ответ: Проверка выполняется вызовом метода validate() схемы, которому передаётся датафрейм. Если данные соответствуют схеме, метод возвращает датафрейм, иначе вызывается исключение. Параметр lazy=True позволяет собрать сразу все ошибки, а не останавливаться на первой. Параметр coerce=True автоматически приводит столбцы к нужным типам.

25. Как Pandera используется для проверки функций обработки данных?
Ответ: Pandera предоставляет декораторы, проверяющие данные на входе и выходе функций: check_input, check_output и check_types. Например, функция очистки данных может автоматически проверять, что на вход поступает датафрейм с нужными столбцами, а на выходе отсутствуют пропуски.

26. Чем Pandera отличается от Great Expectations?
Ответ: Pandera — лёгкая библиотека, в которой схема описывается кодом Python рядом с кодом обработки данных, она удобна для проверки датафреймов внутри скриптов и тестов. Great Expectations — более крупная платформа с источниками данных, хранением результатов, контрольными точками и HTML-документацией Data Docs, она лучше подходит для систематического контроля качества данных в организации.

27. Что такое Deequ?
Ответ: Deequ — это библиотека с открытым исходным кодом, разработанная в компании Amazon, для проверки качества больших наборов данных. Она построена на Apache Spark и позволяет выполнять проверки распределённо на кластере. Основной язык библиотеки — Scala, для Python существует обёртка PyDeequ.

28. Какие возможности предоставляет Deequ?
Ответ: Deequ позволяет описывать проверки (например, полноты, уникальности, неотрицательности значений, количества строк) и запускать их через VerificationSuite, рассчитывать метрики качества с помощью анализаторов, автоматически предлагать правила проверки на основе анализа данных, хранить историю метрик и выявлять аномалии в изменении метрик со временем.

29. В каких случаях следует использовать Deequ?
Ответ: Deequ следует использовать, когда данные обрабатываются в Apache Spark и имеют очень большой объём, который невозможно проверить средствами Pandas на одном компьютере. Например, для проверки ежедневных загрузок в озеро данных объёмом в сотни гигабайт.

30. Как выбрать инструмент автоматической валидации данных?
Ответ: Выбор зависит от объёма данных, используемых технологий и требований к отчётности. Pandera подходит для проверки датафреймов Pandas в Python-коде, Great Expectations — для комплексного контроля качества с документацией и интеграцией в конвейеры, Deequ — для проверки больших данных в Spark. Инструменты можно сочетать на разных этапах конвейера.

Критерии оценки устного опроса:

- оценка «отлично» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов и смог ответить на уточняющие вопросы;
- оценка «хорошо» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов, но не смог ответить на уточняющие вопросы;
- оценка «удовлетворительно» выставляется студенту, если он смог изложить суть основного вопроса, но не смог ответить по существу дополнительных вопросов;

- оценка «неудовлетворительно» выставляется студенту, если он не смог изложить суть основного вопроса.

Практическое занятие №6. Создание набора expectations в Great Expectations для проверки входных данных. Настройка проверок типов, диапазонов, пропусков, уникальности. Генерация Data Docs. Интеграция проверок в Python-скрипт обработки данных.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.11. Осуществлять контроль качества данных в конвейерах обработки

Цель: Научиться создавать набор ожиданий в Great Expectations для проверки входных данных, настраивать проверки типов, диапазонов, пропусков и уникальности, формировать отчёты Data Docs и встраивать проверки в Python-скрипт обработки данных.

Задание:
1. Проанализировать входной набор данных и определить требования к его качеству.
2. Установить Great Expectations и создать Data Context.
3. Подключить источник данных и определить пакет данных для проверки.
4. Создать набор ожиданий с проверками типов, диапазонов, пропусков и уникальности.
5. Выполнить валидацию данных и проанализировать результаты.
6. Сформировать и изучить отчёт Data Docs.
7. Встроить проверки в Python-скрипт обработки данных.
8. Сделать вывод о качестве входных данных и работе проверок.

Этапы выполнения:
Этап 1. Анализ входных данных.
Входной набор данных загружается с помощью Pandas и изучается: состав столбцов, типы данных, количество пропусков, минимальные и максимальные значения, количество уникальных значений. На основе анализа и бизнес-требований формулируется перечень правил качества.

Этап 2. Подготовка среды.
Устанавливается библиотека great_expectations. В скрипте создаётся Data Context функцией get_context(). Для сохранения наборов ожиданий и результатов между запусками используется файловый контекст в папке проекта.

Этап 3. Подключение данных.
В Data Context добавляется источник данных Pandas, в нём создаётся актив данных для датафрейма и определение пакета данных (Batch Definition), которое указывает, какие данные будут проверяться.

Этап 4. Создание набора ожиданий.
Создаётся набор ожиданий (Expectation Suite) с понятным именем. В него добавляются ожидания: проверка наличия и порядка столбцов, проверка типов столбцов, проверка отсутствия пропусков в обязательных полях, проверка уникальности ключевого поля, проверка диапазонов числовых значений, проверка допустимых значений категориальных полей и проверка количества строк.

Этап 5. Выполнение валидации.
Создаётся определение проверки (Validation Definition), связывающее данные и набор ожиданий, и контрольная точка (Checkpoint) с действием обновления Data Docs. Контрольная точка запускается, анализируются результаты: какие ожидания выполнены, какие нет и какие значения нарушают правила.

Этап 6. Формирование Data Docs.
Формируется HTML-документация Data Docs и открывается в браузере. Изучаются страницы набора ожиданий и результатов проверки. Для проверки работы правил в данные намеренно вносятся ошибки, и проверка запускается повторно.

Этап 7. Интеграция в скрипт обработки данных.
Проверка встраивается в Python-скрипт обработки данных после загрузки и до обработки. Если проверка пройдена, скрипт продолжает обработку и сохраняет результат. Если проверка не пройдена, скрипт записывает сведения об ошибках в журнал и завершается с ненулевым кодом, чтобы остановить конвейер.

Этап 8. Оформление результата.
Оформляется отчёт, в котором приводятся перечень правил качества, код создания набора ожиданий и скрипта обработки, снимки отчёта Data Docs для корректных и ошибочных данных и выводы.

Пример решения:
Задание: Настроить проверку входных данных о заказах интернет-магазина перед их обработкой.
Дано: Ежедневно поступает файл orders.csv со столбцами: order_id (номер заказа), customer_id (номер клиента), order_date (дата заказа), amount (сумма заказа), status (статус заказа).
Требования: номер заказа уникален и заполнен, номер клиента заполнен, сумма заказа от 0 до 1 000 000 рублей, статус принимает значения «new», «paid», «shipped», «cancelled», в файле не менее 100 строк.

Решение по этапам:
1. Загружаем файл с помощью Pandas и изучаем данные методами info() и describe(), подтверждаем состав столбцов и типы.
2. Устанавливаем great_expectations и создаём файловый Data Context.
3. Добавляем источник данных Pandas orders_source, актив данных orders_asset и определение пакета для целого датафрейма.
4. Создаём набор ожиданий orders_suite.
5. Добавляем ожидания: состав столбцов соответствует списку order_id, customer_id, order_date, amount, status; столбец amount имеет числовой тип; столбцы order_id и customer_id не содержат пропусков; значения order_id уникальны; значения amount находятся в диапазоне от 0 до 1 000 000; значения status входят в допустимый набор; количество строк не менее 100.
6. Создаём определение проверки и контрольную точку orders_checkpoint с действием обновления Data Docs, запускаем её. Все ожидания выполнены, признак success равен True.
7. Вносим в копию данных ошибки: дублируем номер заказа, указываем отрицательную сумму и статус «unknown». Повторная проверка показывает невыполнение трёх ожиданий с примерами ошибочных значений.
8. Формируем и открываем Data Docs, изучаем результаты обеих проверок.
9. Встраиваем проверку в скрипт process_orders.py: после загрузки файла запускается контрольная точка; при успехе данные обрабатываются и сохраняются, при ошибке выводится список невыполненных ожиданий, и скрипт завершается с кодом 1.
10. Оформляем отчёт.

Ответ: Создан набор ожиданий orders_suite, проверяющий состав столбцов, типы, отсутствие

пропусков, уникальность номера заказа, диапазон суммы, допустимые статусы и количество строк. Проверки корректно выявляют ошибки во входных данных, результаты отображаются в отчёте Data Docs. Валидация встроена в скрипт обработки и останавливает его при некачественных данных, что предотвращает попадание ошибок в дальнейшую обработку.

Критерии оценки практических заданий:
1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Тема 3.3. Выявление аномалий в данных (статистические и ML-методы).
Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.11. Осуществлять контроль качества данных в конвейерах обработки

Вопросы к устному опросу по теме:
1. Что такое аномалия в данных?
Ответ: Аномалия — это наблюдение или группа наблюдений, которые существенно отличаются от основной массы данных или от ожидаемого поведения. Аномалии могут быть следствием ошибок в данных, а могут отражать важные редкие события, например мошенничество, отказ оборудования или кибератаку.

2. Чем аномалия отличается от выброса?
Ответ: Понятия близки и часто используются как синонимы. Выбросом обычно называют отдельное значение, статистически удалённое от остальных. Аномалия — более широкое понятие, включающее необычные значения, необычные сочетания признаков, необычное поведение в определённом контексте и необычные последовательности событий.

3. Какие виды аномалий выделяют?
Ответ: Выделяют точечные аномалии — отдельные значения, резко отличающиеся от остальных; контекстные аномалии — значения, необычные только в определённом контексте, например высокая температура зимой; коллективные аномалии — группы наблюдений, которые аномальны вместе, хотя каждое по отдельности выглядит нормально.

4. Для чего выявляют аномалии в данных?
Ответ: Аномалии выявляют для контроля качества данных и обнаружения ошибок, для подготовки данных к обучению моделей, а также как самостоятельную задачу: выявление мошеннических операций, обнаружение неисправностей оборудования, мониторинг информационных систем, обнаружение вторжений в компьютерные сети.

5. На какие группы делятся методы выявления аномалий?
Ответ: Методы делятся на статистические (Z-score, IQR, тест Граббса), методы машинного обучения (Isolation Forest, LOF, DBSCAN, One-Class SVM), методы для временных рядов, а также на основе правил, заданных экспертами. Методы машинного обучения, в свою очередь,

бывают с учителем, без учителя и с частичным привлечением учителя.

6. Как выявляются аномалии методом Z-score?
Ответ: Для каждого значения вычисляется Z-оценка: z = (x − среднее) / стандартное отклонение, показывающая, на сколько стандартных отклонений значение удалено от среднего. Значения, у которых модуль Z-оценки превышает пороговое значение, обычно 3, считаются аномальными.

7. Каковы ограничения метода Z-score?
Ответ: Метод предполагает, что данные распределены нормально, и плохо работает при асимметричных распределениях. Кроме того, среднее и стандартное отклонение сами чувствительны к выбросам: сильные выбросы увеличивают стандартное отклонение и могут «маскировать» другие аномалии.

8. Что такое модифицированная Z-оценка?
Ответ: Модифицированная Z-оценка вычисляется с использованием медианы и медианного абсолютного отклонения (MAD) вместо среднего и стандартного отклонения: M = 0,6745 · (x − медиана) / MAD. Она устойчива к выбросам. Аномальными обычно считаются значения с модулем модифицированной оценки больше 3,5.

9. Как выявляются аномалии методом IQR?
Ответ: Вычисляются первый (Q1) и третий (Q3) квартили и межквартильный размах IQR = Q3 − Q1. Аномальными считаются значения меньше Q1 − 1,5 · IQR или больше Q3 + 1,5 · IQR. Для поиска только экстремальных выбросов вместо 1,5 используется коэффициент 3.

10. Каковы преимущества метода IQR?
Ответ: Метод IQR не требует нормального распределения данных, устойчив к выбросам, так как квартили мало зависят от экстремальных значений, прост в расчёте и наглядно отображается на диаграмме размаха (boxplot).

11. Что такое тест Граббса (Grubbs' test)?
Ответ: Тест Граббса — это статистический тест для обнаружения одного выброса в выборке из нормально распределённых данных. Вычисляется статистика G — максимальное отклонение значения от среднего, делённое на стандартное отклонение, и сравнивается с критическим значением для заданного уровня значимости. Если G больше критического значения, наиболее удалённое значение признаётся выбросом.

12. Как применяется тест Граббса для поиска нескольких выбросов?
Ответ: Тест Граббса выявляет за один раз только одно значение, поэтому для поиска нескольких выбросов он применяется итеративно: найденный выброс удаляется, и тест повторяется на оставшихся данных до тех пор, пока очередной выброс не будет обнаружен. Тест требует нормальности данных и предназначен для небольших выборок.

13. Почему статистические методы не всегда подходят для поиска аномалий?
Ответ: Классические статистические методы обычно анализируют каждый признак отдельно и предполагают определённое распределение. Они не обнаруживают аномалии, которые проявляются только в необычном сочетании нескольких признаков, и плохо работают со сложными многомерными данными. В таких случаях применяются методы машинного обучения.

14. Что такое Isolation Forest?
Ответ: Isolation Forest (изолирующий лес) — это алгоритм выявления аномалий, основанный на ансамбле случайных деревьев. Деревья многократно разбивают данные случайными разрезами по случайно выбранным признакам. Аномальные объекты отличаются от остальных, поэтому изолируются быстрее и находятся ближе к корню дерева, то есть имеют меньшую среднюю длину пути.

15. Какие параметры имеет Isolation Forest и как интерпретируются его результаты?
Ответ: Основные параметры: n_estimators — количество деревьев, max_samples — размер подвыборки для построения дерева, contamination — ожидаемая доля аномалий. В scikit-learn метод predict() возвращает −1 для аномалий и 1 для нормальных объектов, а методы decision_function() и score_samples() возвращают оценку аномальности: чем она ниже, тем более аномален объект.

16. Что такое LOF (Local Outlier Factor)?
Ответ: LOF (локальный коэффициент выброса) — это метод, сравнивающий локальную плотность объекта с плотностью его ближайших соседей. Если объект находится в значительно менее плотной области, чем его соседи, он считается аномальным. Значение LOF около 1 означает нормальный объект, значительно больше 1 — аномальный.

17. В чём особенность метода LOF?
Ответ: LOF учитывает локальную структуру данных, поэтому находит аномалии в данных с кластерами разной плотности, где глобальные методы ошибаются. Основной параметр — количество соседей n_neighbors. Метод чувствителен к масштабу признаков, поэтому данные предварительно масштабируются. Для оценки новых объектов используется параметр novelty=True.

18. Как алгоритм DBSCAN используется для выявления аномалий?
Ответ: DBSCAN — это алгоритм кластеризации на основе плотности. Он объединяет в кластеры объекты, расположенные в плотных областях, а объекты, не попавшие ни в один кластер, помечает как шум с меткой −1. Эти точки шума и рассматриваются как аномалии.

19. Какие параметры имеет DBSCAN?
Ответ: Основные параметры: eps — радиус окрестности, в которой ищутся соседи, и min_samples — минимальное количество объектов в окрестности, необходимое для образования плотной области. От выбора этих параметров сильно зависит результат: слишком малый eps делает аномалиями слишком много объектов, слишком большой — объединяет всё в один кластер.

20. Что такое One-Class SVM?
Ответ: One-Class SVM (одноклассовый метод опорных векторов) — это алгоритм, который обучается на нормальных данных и строит границу, охватывающую область их расположения. Объекты, оказавшиеся за пределами этой границы, считаются аномальными. Основные параметры: ядро (обычно rbf), gamma и nu — верхняя граница доли аномалий.

21. Каковы особенности применения One-Class SVM?
Ответ: One-Class SVM хорошо работает, когда обучающие данные содержат в основном нормальные объекты, и позволяет описывать сложные границы нормальной области. При этом метод требует масштабирования признаков, чувствителен к выбору параметров и медленно работает на больших наборах данных.

22. Как выбрать метод выявления аномалий?
Ответ: Выбор зависит от размерности и объёма данных, вида распределения, наличия размеченных примеров аномалий и требований к интерпретируемости. Для одного признака подходят Z-score и IQR, для многомерных больших данных — Isolation Forest, для данных с кластерами разной плотности — LOF и DBSCAN, при обучении только на нормальных данных — One-Class SVM.

23. Как интерпретировать результаты выявления аномалий?
Ответ: Для каждого найденного объекта анализируются оценка аномальности, значения признаков в сравнении с типичными значениями и признаки, которые сильнее всего отличаются.

Найденные аномалии проверяются экспертами предметной области, чтобы определить, являются ли они ошибками данных, реальными редкими событиями или ложными срабатываниями.

24. Как оценить качество работы метода выявления аномалий?
Ответ: Если есть размеченные данные с известными аномалиями, используются точность (precision), полнота (recall), F1-мера и площадь под ROC- или PR-кривой. Если разметки нет, качество оценивается экспертной проверкой выборки найденных аномалий, сравнением результатов разных методов и анализом доли ложных срабатываний.

25. Какие особенности имеют аномалии во временных рядах?
Ответ: Во временных рядах значения зависят от времени и предыдущих значений, содержат тренд и сезонность. Значение может быть нормальным само по себе, но аномальным для данного момента, например высокие продажи ночью. Поэтому аномалии ищутся с учётом временного контекста.

26. Какие методы применяются для выявления аномалий во временных рядах?
Ответ: Применяются скользящее среднее и скользящее стандартное отклонение с границами допустимых значений, разложение ряда на тренд, сезонность и остаток (например, STL) с поиском аномалий в остатке, модели прогнозирования (ARIMA, Prophet) с оценкой отклонения фактических значений от прогноза, а также Isolation Forest на признаках, построенных по временным окнам.

27. Какие виды аномалий встречаются во временных рядах?
Ответ: Встречаются точечные аномалии — отдельные резкие всплески или провалы, сдвиги уровня — резкое изменение среднего уровня ряда, изменения тренда, нарушение сезонности, а также пропуски и «застывшие» значения, когда датчик долгое время передаёт одно и то же значение.

28. Какие способы визуализации применяются для выявления аномалий?
Ответ: Применяются диаграммы размаха (boxplot) и гистограммы для отдельных признаков, диаграммы рассеяния для пар признаков, графики временных рядов с выделением аномальных точек и границ допустимых значений, а также проекции многомерных данных на плоскость методами PCA или t-SNE с выделением аномалий цветом.

29. Какие способы обработки аномалий существуют?
Ответ: Аномалии можно удалить, исправить, если известно правильное значение, заменить медианой или граничным значением, преобразовать данные (например, логарифмированием), пометить отдельным признаком для модели, перенести в карантин для проверки или оставить без изменений, если они отражают реальные события.

30. Почему аномалии нельзя всегда просто удалять?
Ответ: Аномалии могут содержать самую ценную информацию: мошеннические операции, отказы оборудования, атаки на систему. Их удаление может лишить модель возможности распознавать важные события и исказить представление о данных. Поэтому перед обработкой необходимо установить причину аномалии и решить, является ли она ошибкой или реальным событием.

Критерии оценки устного опроса:

- оценка «отлично» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов и смог ответить на уточняющие вопросы;
- оценка «хорошо» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов, но не смог ответить на уточняющие вопросы;

- оценка «удовлетворительно» выставляется студенту, если он смог изложить суть основного вопроса, но не смог ответить по существу дополнительных вопросов;
- оценка «неудовлетворительно» выставляется студенту, если он не смог изложить суть основного вопроса.

Тема 3.4. Data Drift и Concept Drift: типы и методы обнаружения.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.11. Осуществлять контроль качества данных в конвейерах обработки

Вопросы к устному опросу по теме:

1. Что такое дрейф в машинном обучении?
Ответ: Дрейф — это изменение со временем свойств данных или зависимостей между данными и целевой переменной по сравнению с теми, на которых обучалась модель. Из-за дрейфа модель, хорошо работавшая при внедрении, постепенно начинает давать менее точные прогнозы.

2. Почему возникает дрейф данных?
Ответ: Дрейф возникает из-за изменений в реальном мире: смены поведения и предпочтений клиентов, сезонности, экономических событий, появления новых товаров и конкурентов, изменения законодательства. Также причиной могут быть технические изменения: замена датчиков, изменение формата данных, ошибки в конвейере обработки.

3. Что такое Data Drift?
Ответ: Data Drift (дрейф данных) — это изменение статистического распределения входных данных модели со временем. Например, средний возраст клиентов, обращающихся за кредитом, вырос, или изменилась доля пользователей с мобильных устройств. Модель при этом получает данные, отличающиеся от обучающих.

4. Что такое feature drift?
Ответ: Feature drift (дрейф признаков, ковариатный сдвиг) — это изменение распределения одного или нескольких входных признаков P(X) при неизменной зависимости целевой переменной от признаков. Например, в интернет-магазин стали чаще заходить пользователи из другого региона.

5. Что такое label drift?
Ответ: Label drift (дрейф меток, сдвиг априорных вероятностей) — это изменение распределения целевой переменной P(y). Например, доля мошеннических операций выросла с 1% до 3% или доля клиентов, отказывающихся от услуг, резко увеличилась.

6. Что такое Concept Drift?
Ответ: Concept Drift (дрейф концепции) — это изменение самой зависимости между входными признаками и целевой переменной P(y|X). При тех же значениях признаков правильный ответ становится другим. Например, мошенники изменили схему действий, и ранее безопасные операции стали признаком мошенничества.

7. Чем Data Drift отличается от Concept Drift?

Ответ: При Data Drift меняются сами входные данные, а правила, связывающие их с ответом, остаются прежними. При Concept Drift меняются именно правила, даже если входные данные выглядят так же. Data Drift можно обнаружить по входным данным, а для обнаружения Concept Drift обычно нужны фактические значения целевой переменной.

8. Какие виды дрейфа выделяют по характеру изменения во времени?
Ответ: Выделяют внезапный дрейф — резкое изменение в определённый момент, например после изменения законодательства; постепенный дрейф — медленный переход от старого состояния к новому; инкрементальный дрейф — плавное непрерывное изменение; повторяющийся (сезонный) дрейф — периодическое возвращение к прежним состояниям.

9. Как в общем виде выполняется обнаружение дрейфа?
Ответ: Выбирается эталонный (референсный) набор данных, обычно обучающая выборка или данные за стабильный период, и текущий набор данных за проверяемый период. Для каждого признака и для прогнозов модели распределения двух наборов сравниваются с помощью статистических тестов или мер расстояния. Если различие превышает порог, фиксируется дрейф.

10. Что такое PSI (Population Stability Index)?
Ответ: PSI (индекс стабильности популяции) — это показатель, измеряющий, насколько изменилось распределение признака между эталонным и текущим наборами данных. Значения признака разбиваются на интервалы, для каждого интервала сравниваются доли объектов в двух наборах, и отклонения суммируются.

11. Как рассчитывается и интерпретируется PSI?
Ответ: PSI рассчитывается по формуле $PSI = \Sigma (Ai - Ei) \cdot \ln(Ai / Ei)$, где Ei — доля объектов в интервале в эталонных данных, Ai — в текущих. Значение меньше 0,1 означает отсутствие значимых изменений, от 0,1 до 0,25 — умеренный дрейф, требующий внимания, больше 0,25 — значительный дрейф, требующий действий.

12. Что такое KS-тест (критерий Колмогорова — Смирнова)?
Ответ: Двухвыборочный критерий Колмогорова — Смирнова проверяет гипотезу о том, что две выборки взяты из одного распределения. Статистика теста D равна максимальной разнице между эмпирическими функциями распределения двух выборок. Если p-значение меньше уровня значимости, например 0,05, распределения считаются различными, то есть фиксируется дрейф.

13. Каковы особенности применения KS-теста?
Ответ: KS-тест применяется для непрерывных числовых признаков и не требует предположений о виде распределения. На больших выборках он становится очень чувствительным и может фиксировать статистически значимые, но практически несущественные изменения. В Python тест выполняется функцией ks_2samp() из библиотеки SciPy.

14. Как обнаруживается дрейф категориальных признаков?
Ответ: Для категориальных признаков сравниваются частоты категорий в эталонных и текущих данных. Применяются критерий хи-квадрат, PSI, рассчитанный по категориям, и дивергенция Йенсена — Шеннона. Также отслеживается появление новых категорий, отсутствовавших в обучающих данных.

15. Что такое дивергенция Кульбака — Лейблера?
Ответ: Дивергенция Кульбака — Лейблера — это мера того, насколько одно распределение вероятностей отличается от другого. Она несимметрична, то есть результат зависит от порядка сравниваемых распределений, и не ограничена сверху. На её основе построены PSI и дивергенция Йенсена — Шеннона.

16. Что такое дивергенция Йенсена — Шеннона (Jensen-Shannon divergence)?

Ответ: Дивергенция Йенсена — Шеннона — это симметричная мера различия двух распределений, основанная на дивергенции Кульбака — Лейблера относительно их среднего распределения. При использовании логарифма по основанию 2 её значения находятся в диапазоне от 0 (распределения совпадают) до 1 (распределения полностью различны), что удобно для задания порогов.

17. Что такое расстояние Вассерштейна (Wasserstein distance)?
Ответ: Расстояние Вассерштейна, или расстояние перемещения земли, — это минимальная «работа», необходимая, чтобы преобразовать одно распределение в другое, перемещая вероятностную массу. Оно измеряется в единицах самого признака и показывает, насколько сместилось распределение. В SciPy рассчитывается функцией wasserstein_distance().

18. В чём преимущества расстояния Вассерштейна?
Ответ: Расстояние Вассерштейна учитывает не только факт различия распределений, но и величину смещения значений, хорошо отражает постепенные сдвиги и менее чувствительно к размеру выборки, чем статистические тесты. Однако оно зависит от масштаба признака, поэтому для сравнения разных признаков данные нормализуют.

19. Как выбрать метод обнаружения дрейфа?
Ответ: Выбор зависит от типа признака, объёма данных и требований к интерпретации. Для числовых признаков на небольших выборках подходит KS-тест, на больших — расстояние Вассерштейна и PSI; для категориальных — хи-квадрат, PSI и дивергенция Йенсена — Шеннона. PSI широко применяется в банковской сфере благодаря понятным порогам.

20. Что такое статистический мониторинг дрейфа?
Ответ: Статистический мониторинг — это регулярное сравнение распределений отдельных признаков и прогнозов модели с эталонными с помощью статистических тестов и мер расстояния. Метод прост, понятен и показывает, какие именно признаки изменились, но анализирует признаки по отдельности и может не замечать изменений во взаимосвязях между ними.

21. Что такое обнаружение дрейфа на основе машинного обучения?
Ответ: Это подход, при котором для обнаружения дрейфа используются модели. Например, обучается классификатор, различающий эталонные и текущие данные: если он отличает их с качеством заметно выше случайного (ROC-AUC значительно больше 0,5), значит, данные изменились. Такой подход учитывает взаимосвязи признаков и многомерные изменения.

22. Какие ещё ML-методы применяются для обнаружения дрейфа?
Ответ: Применяются мониторинг метрик качества модели при наличии фактических ответов, специальные алгоритмы обнаружения изменений в потоке, например DDM, ADWIN и Page-Hinkley, отслеживание изменения распределения уверенности модели в прогнозах, а также методы оценки качества модели без истинных меток.

23. Чем статистический мониторинг отличается от ML-based detection?
Ответ: Статистический мониторинг проще, быстрее, легче интерпретируется и указывает на конкретные изменившиеся признаки, но работает с признаками по отдельности. ML-based detection обнаруживает сложные многомерные изменения и изменения взаимосвязей, но сложнее в настройке и интерпретации. На практике подходы часто сочетают.

24. Как обнаружить Concept Drift?
Ответ: Основной способ — мониторинг качества модели на новых данных с известными фактическими ответами: снижение точности, F1-меры или рост ошибки указывают на Concept Drift. Если фактические ответы поступают с задержкой, используют косвенные признаки: изменение распределения прогнозов и уверенности модели, а также алгоритмы обнаружения

изменений в потоке ошибок.

25. Какие инструменты применяются для мониторинга дрейфа?
Ответ: Применяются библиотеки Evidently AI, которая формирует отчёты и панели мониторинга дрейфа данных и качества модели, NannyML, позволяющая оценивать качество модели без истинных меток, Alibi Detect и whylogs, а также средства мониторинга облачных ML-платформ. Метрики дрейфа также выводятся в системы мониторинга, например Grafana.

26. Как определить частоту проверки на дрейф?
Ответ: Частота проверки зависит от скорости изменения данных, критичности модели, объёма поступающих данных и стоимости ошибок. Для моделей выявления мошенничества проверки выполняются ежечасно или ежедневно, для моделей кредитного скоринга — еженедельно или ежемесячно. В каждом периоде должно накапливаться достаточно данных для надёжного сравнения.

27. Как выбрать эталонные данные для сравнения?
Ответ: В качестве эталона обычно используется обучающая выборка модели или данные за период, когда модель работала стабильно и показывала хорошее качество. Можно применять фиксированный эталон или скользящее окно, например данные за предыдущий месяц. Эталон должен быть достаточно большим и учитывать сезонность.

28. Какие действия выполняются при обнаружении дрейфа?
Ответ: Сначала выясняется причина: техническая ошибка в конвейере или реальное изменение данных. При ошибке исправляется конвейер. При реальном изменении оценивается влияние на качество модели, при необходимости модель переобучается на свежих данных, корректируются признаки или пороги принятия решений, а в критических случаях модель временно заменяется резервной.

29. Всегда ли обнаружение дрейфа требует переобучения модели?
Ответ: Нет. Дрейф может затрагивать малозначимые признаки и не влиять на качество модели, быть временным или сезонным, а также быть следствием технической ошибки, которую нужно исправить, а не учитывать при обучении. Решение о переобучении принимается после анализа влияния дрейфа на качество прогнозов.

30. Что такое автоматическое переобучение модели при дрейфе?
Ответ: Автоматическое переобучение — это настройка конвейера, при которой обнаружение значимого дрейфа или снижение качества модели автоматически запускает переобучение на свежих данных, проверку новой модели и её развёртывание при улучшении качества. Также применяется переобучение по расписанию. Автоматизация требует надёжных проверок качества данных и модели перед развёртыванием.

Критерии оценки устного опроса:

- оценка «отлично» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов и смог ответить на уточняющие вопросы;
- оценка «хорошо» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов, но не смог ответить на уточняющие вопросы;
- оценка «удовлетворительно» выставляется студенту, если он смог изложить суть основного вопроса, но не смог ответить по существу дополнительных вопросов;
- оценка «неудовлетворительно» выставляется студенту, если он не смог изложить суть основного вопроса.

Практическое занятие №7. Анализ и визуализация дрейфа данных (Data Drift).

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.11. Осуществлять контроль качества данных в конвейерах обработки

Цель: Научиться выявлять дрейф данных путём сравнения эталонного и текущего наборов данных с помощью статистических тестов и мер расстояния, визуализировать изменения распределений признаков и формулировать выводы о необходимых действиях.

Задание:
1. Подготовить эталонный и текущий наборы данных.
2. Определить признаки для анализа и методы проверки для каждого типа признаков.
3. Рассчитать показатели дрейфа для числовых признаков: PSI, KS-тест, расстояние Вассерштейна.
4. Рассчитать показатели дрейфа для категориальных признаков.
5. Построить визуализации распределений признаков в эталонных и текущих данных.
6. Сформировать отчёт о дрейфе данных с помощью специализированной библиотеки.
7. Проанализировать дрейф во времени.
8. Сделать вывод о наличии дрейфа и необходимых действиях.

Этапы выполнения:
Этап 1. Подготовка данных.
Загружаются данные с помощью Pandas. В качестве эталонного набора выбираются данные, на которых обучалась модель или данные за стабильный период, в качестве текущего — данные за проверяемый период. Проверяется, что наборы имеют одинаковую структуру.

Этап 2. Выбор признаков и методов.
Определяются признаки для анализа и их типы. Для числовых признаков выбираются PSI, KS-тест и расстояние Вассерштейна, для категориальных — PSI по категориям, критерий хи-квадрат и дивергенция Йенсена — Шеннона. Задаются пороговые значения для каждого показателя.

Этап 3. Расчёт показателей для числовых признаков.
Для каждого числового признака рассчитывается PSI по интервалам, построенным по эталонным данным, выполняется KS-тест функцией ks_2samp() и рассчитывается расстояние Вассерштейна функцией wasserstein_distance() библиотеки SciPy.

Этап 4. Расчёт показателей для категориальных признаков.
Для каждого категориального признака рассчитываются доли категорий в двух наборах, PSI, критерий хи-квадрат и дивергенция Йенсена — Шеннона. Проверяется появление новых категорий.

Этап 5. Визуализация распределений.
С помощью библиотек Matplotlib или Seaborn строятся совмещённые гистограммы и графики плотности распределений числовых признаков для эталонных и текущих данных, столбчатые диаграммы долей категорий, а также сводная диаграмма значений PSI по всем признакам с отметкой пороговых значений.

Этап 6. Формирование отчёта о дрейфе.

С помощью библиотеки Evidently AI формируется отчёт о дрейфе данных, в котором для каждого признака автоматически выбирается метод проверки, указывается наличие дрейфа и строятся графики. Отчёт сохраняется в формате HTML.

Этап 7. Анализ дрейфа во времени.
Текущие данные разбиваются на периоды, например по неделям, и для каждого периода рассчитывается PSI ключевых признаков. Строится график изменения PSI во времени с пороговыми линиями, по которому определяется момент появления и характер дрейфа: внезапный, постепенный или сезонный.

Этап 8. Оформление результата.
Оформляется отчёт, содержащий сводную таблицу показателей дрейфа по признакам, визуализации, отчёт Evidently, анализ причин выявленного дрейфа и рекомендации по дальнейшим действиям.

Пример решения:
Задание: Проанализировать дрейф данных модели прогнозирования оттока клиентов интернет-провайдера.
Дано: Модель обучена на данных за январь — июнь. Имеются эталонные данные за этот период (20 000 записей) и текущие данные за июль — сентябрь (12 000 записей). Признаки: возраст, ежемесячный платёж, количество обращений в поддержку, тип тарифа, способ оплаты. Необходимо определить, произошёл ли дрейф.

Решение по этапам:
1. Загружаем эталонные и текущие данные, проверяем совпадение столбцов и типов.
2. Определяем числовые признаки (возраст, ежемесячный платёж, количество обращений) и категориальные (тип тарифа, способ оплаты). Устанавливаем пороги: PSI 0,1 и 0,25, p-значение KS-теста 0,05.
3. Рассчитываем PSI для числовых признаков по 10 интервалам: возраст — 0,03, ежемесячный платёж — 0,31, количество обращений — 0,14.
4. Выполняем KS-тест: для ежемесячного платежа и количества обращений p-значение меньше 0,05, для возраста — больше 0,05. Расстояние Вассерштейна показывает, что ежемесячный платёж в среднем сместился примерно на 250 рублей.
5. Для типа тарифа PSI равен 0,28, появилась новая категория тарифа «Семейный», отсутствовавшая в обучающих данных. Для способа оплаты PSI равен 0,05.
6. Строим совмещённые гистограммы: распределение ежемесячного платежа сместилось вправо, доля обращений в поддержку выросла. Строим сводную диаграмму PSI с пороговыми линиями.
7. Формируем отчёт Evidently AI, который подтверждает дрейф в признаках «ежемесячный платёж», «количество обращений» и «тип тарифа».
8. Рассчитываем PSI ежемесячного платежа по неделям и строим график: резкий рост PSI начинается с первой недели августа, что указывает на внезапный дрейф.
9. Анализируем причину: в августе провайдер повысил тарифы и ввёл новый тариф «Семейный», что совпадает с моментом дрейфа.
10. Формируем рекомендации и оформляем отчёт.

Ответ: Выявлен значительный дрейф признаков «ежемесячный платёж» (PSI 0,31) и «тип тарифа» (PSI 0,28) и умеренный дрейф признака «количество обращений в поддержку» (PSI 0,14). Дрейф внезапный, начался в августе и связан с изменением тарифной политики. Признаки «возраст» и «способ оплаты» стабильны. Рекомендуется проверить качество модели на свежих данных с известным оттоком и переобучить модель с учётом новых тарифов.

Критерии оценки практических заданий:
1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.

3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Тема 3.5. Мониторинг конвейеров: KPI, SLA, логирование и алертинг.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.11. Осуществлять контроль качества данных в конвейерах обработки

Вопросы к устному опросу по теме:

1. Что такое мониторинг конвейеров данных?
Ответ: Мониторинг конвейеров данных — это непрерывное отслеживание работы конвейеров: успешности и длительности запусков, объёма обработанных данных, ошибок, задержек и использования ресурсов. Мониторинг позволяет своевременно обнаруживать сбои и снижение производительности и реагировать на них до того, как пострадают пользователи данных.

2. Для чего необходим мониторинг конвейеров данных?
Ответ: Мониторинг необходим, чтобы гарантировать своевременную доставку качественных данных потребителям: отчётам, аналитикам и моделям машинного обучения. Без мониторинга сбои обнаруживаются поздно, обычно когда пользователи замечают устаревшие или неверные данные, что приводит к ошибочным решениям и потерям.

3. Что такое KPI конвейера данных?
Ответ: KPI (ключевые показатели эффективности) конвейера — это количественные показатели, характеризующие качество и эффективность его работы. К основным KPI относятся пропускная способность, задержка, доля ошибок и доля успешных запусков, а также своевременность доставки данных и использование ресурсов.

4. Что такое пропускная способность (throughput)?
Ответ: Пропускная способность — это объём данных, который конвейер обрабатывает за единицу времени, например количество записей в секунду, мегабайт в минуту или событий в секунду для потоковой обработки. Снижение пропускной способности может указывать на перегрузку или проблемы с ресурсами.

5. Что такое задержка (latency)?
Ответ: Задержка — это время, необходимое для обработки данных. Для потоковых конвейеров это время от появления события до получения результата, для пакетных — длительность выполнения запуска или время от поступления данных до их доступности в хранилище. Рост задержки означает, что данные доставляются медленнее.

6. Что такое доля ошибок (error rate)?
Ответ: Доля ошибок — это отношение количества неудачных запусков, задач или некорректно обработанных записей к их общему количеству за период, выраженное в процентах. Например, если из 1000 запусков задач 15 завершились ошибкой, доля ошибок составляет 1,5%.

7. Что такое доля успешных запусков (success rate)?

Ответ: Доля успешных запусков — это отношение количества успешно завершённых запусков конвейера к общему количеству запусков за период, выраженное в процентах. Это показатель надёжности конвейера, дополняющий долю ошибок до 100%.

8. Какие ещё показатели отслеживают при мониторинге конвейеров?
Ответ: Отслеживают свежесть данных (время с момента последнего обновления), объём загруженных данных по сравнению с ожидаемым, длительность отдельных задач, количество повторных попыток, использование процессора, памяти и диска, длину очередей задач, отставание консьюмеров Kafka и показатели качества данных.

9. Что такое SLI?
Ответ: SLI (Service Level Indicator, показатель уровня обслуживания) — это конкретная измеряемая метрика, характеризующая качество работы сервиса или конвейера. Например, доля запусков, завершённых успешно, или доля дней, когда данные были доступны до 06:00.

10. Что такое SLO?
Ответ: SLO (Service Level Objective, целевой уровень обслуживания) — это целевое значение показателя SLI, которого команда стремится достигать. Например, данные должны быть загружены до 06:00 в 99% дней месяца, а доля успешных запусков должна быть не ниже 99,5%. SLO используется внутри команды для управления надёжностью.

11. Что такое SLA?
Ответ: SLA (Service Level Agreement, соглашение об уровне обслуживания) — это формальное соглашение между поставщиком и потребителем сервиса, в котором закреплены гарантируемые уровни обслуживания и последствия их нарушения, например компенсации. SLA обычно устанавливается менее строгим, чем внутренний SLO, чтобы иметь запас.

12. Чем SLA отличается от SLO?
Ответ: SLO — это внутренняя цель команды по уровню надёжности, её нарушение служит сигналом к улучшению системы. SLA — это внешнее обязательство перед заказчиком, закреплённое в соглашении, нарушение которого влечёт последствия. SLO задаётся строже SLA, чтобы проблемы обнаруживались до нарушения обязательств.

13. Что такое бюджет ошибок (error budget)?
Ответ: Бюджет ошибок — это допустимый объём сбоев, определяемый как разница между 100% и значением SLO. Например, при SLO 99% успешных запусков допускается 1% неудачных. Пока бюджет не исчерпан, команда может вносить изменения, а при его исчерпании основные усилия направляются на повышение надёжности.

14. Что такое наблюдаемость (observability)?
Ответ: Наблюдаемость — это свойство системы, позволяющее понять её внутреннее состояние по данным, которые она выдаёт наружу. Наблюдаемая система позволяет не только обнаружить проблему, но и быстро выяснить её причину. Наблюдаемость строится на трёх видах данных: журналах, метриках и трассировках.

15. Чем мониторинг отличается от наблюдаемости?
Ответ: Мониторинг отвечает на вопрос, работает ли система, отслеживая заранее известные показатели и проблемы. Наблюдаемость позволяет ответить на вопрос, почему система работает неправильно, в том числе при проблемах, которые не были предусмотрены заранее. Мониторинг является частью наблюдаемости.

16. Что такое журналы (logs)?
Ответ: Журналы — это записи о событиях, происходящих в системе, с указанием времени, уровня важности и описания события, например начала и окончания задачи, ошибки с текстом

исключения, количества обработанных записей. Журналы дают подробную информацию для расследования причин сбоев.

17. Какие уровни журналирования существуют?
Ответ: Стандартные уровни журналирования: DEBUG — подробная отладочная информация, INFO — информация о нормальной работе, WARNING — предупреждения о потенциальных проблемах, ERROR — ошибки, при которых операция не выполнена, CRITICAL — критические ошибки, угрожающие работе всей системы. Уровень позволяет фильтровать записи по важности.

18. Что такое структурированное журналирование?
Ответ: Структурированное журналирование — это запись журналов в машиночитаемом формате, обычно JSON, где каждая запись состоит из именованных полей: время, уровень, сообщение, идентификатор конвейера, задачи и запуска. Такие журналы легко искать, фильтровать и анализировать в системах централизованного журналирования.

19. Что такое метрики (metrics)?
Ответ: Метрики — это числовые показатели работы системы, измеряемые через равные промежутки времени и хранимые как временные ряды, например количество обработанных записей, длительность задачи, использование памяти. Метрики занимают мало места, удобны для построения графиков, выявления трендов и настройки оповещений.

20. Что такое трассировки (traces)?
Ответ: Трассировки — это записи пути отдельного запроса или единицы данных через все компоненты распределённой системы с указанием времени, затраченного на каждом этапе. Трассировки позволяют найти, на каком этапе возникает задержка или ошибка. Для их сбора используются OpenTelemetry, Jaeger, Grafana Tempo.

21. Что такое централизованное журналирование и зачем оно нужно?
Ответ: Централизованное журналирование — это сбор журналов со всех серверов, контейнеров и сервисов в единое хранилище с возможностью поиска и анализа. Оно необходимо, потому что конвейеры работают на многих машинах, и искать журналы на каждой вручную неудобно, а при пересоздании контейнеров журналы теряются.

22. Что такое ELK stack?
Ответ: ELK stack — это набор инструментов для централизованного журналирования: Elasticsearch — поисковое хранилище, индексирующее журналы; Logstash — инструмент сбора, обработки и преобразования журналов; Kibana — веб-интерфейс для поиска, анализа и визуализации журналов. Часто дополняется агентами сбора Beats, например Filebeat.

23. Что такое Grafana Loki?
Ответ: Loki — это система хранения и поиска журналов от компании Grafana Labs. В отличие от Elasticsearch, Loki индексирует не весь текст журналов, а только метки (labels), например имя сервиса и конвейера, поэтому требует меньше ресурсов. Журналы просматриваются в Grafana с помощью языка запросов LogQL.

24. Что такое Prometheus?
Ответ: Prometheus — это система мониторинга с открытым исходным кодом, которая собирает и хранит метрики в виде временных рядов. Prometheus сам периодически опрашивает сервисы и экспортёры по протоколу HTTP (модель pull), а для анализа метрик используется язык запросов PromQL. Метрики обычно визуализируются в Grafana.

25. Какие типы метрик поддерживает Prometheus?
Ответ: Prometheus поддерживает четыре типа метрик: counter (счётчик), который только увеличивается, например количество обработанных записей; gauge (датчик), который может

увеличиваться и уменьшаться, например использование памяти; histogram (гистограмма) и summary (сводка), которые описывают распределение значений, например длительности выполнения задач.

26. Как метрики конвейеров передаются в Prometheus?
Ответ: Сервисы предоставляют метрики через специальный HTTP-адрес, который опрашивает Prometheus, для чего используются клиентские библиотеки, например prometheus_client для Python, и экспортёры для готовых систем. Для кратковременных пакетных задач, которые завершаются до опроса, используется Pushgateway, куда задачи сами отправляют метрики.

27. Что такое алертинг?
Ответ: Алертинг — это автоматическая отправка оповещений ответственным специалистам при возникновении проблем, например сбое конвейера, превышении задержки, росте доли ошибок или устаревании данных. Алертинг позволяет быстро реагировать на инциденты без постоянного наблюдения за панелями мониторинга.

28. Что такое правила алертинга и какие требования к ним предъявляются?
Ответ: Правило алертинга — это условие, при выполнении которого отправляется оповещение, например доля ошибок больше 5% в течение 10 минут. Правила должны отражать реальные проблемы, влияющие на пользователей, иметь обоснованные пороги, уровень важности и инструкцию по реагированию. Слишком много лишних оповещений приводит к «усталости от алертов», когда важные сигналы игнорируются.

29. Что такое эскалация оповещений?
Ответ: Эскалация — это порядок передачи оповещения следующему уровню ответственных, если инцидент не подтверждён или не устранён в установленное время. Например, сначала оповещается дежурный инженер, через 15 минут без реакции — руководитель группы, затем — руководитель отдела. Эскалация гарантирует, что критичный инцидент не останется без внимания.

30. Как настраивается интеграция алертинга со Slack, Email и PagerDuty?
Ответ: В Prometheus оповещения обрабатываются компонентом Alertmanager, который группирует их, подавляет повторы и направляет получателям по правилам маршрутизации: в каналы Slack через веб-хуки, на электронную почту через SMTP, в PagerDuty через ключ интеграции для дежурств и эскалации. В Airflow для оповещений используются функции обратного вызова при ошибке задачи и провайдеры Slack и Email.

Критерии оценки устного опроса:

- оценка «отлично» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов и смог ответить на уточняющие вопросы;
- оценка «хорошо» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов, но не смог ответить на уточняющие вопросы;
- оценка «удовлетворительно» выставляется студенту, если он смог изложить суть основного вопроса, но не смог ответить по существу дополнительных вопросов;
- оценка «неудовлетворительно» выставляется студенту, если он не смог изложить суть основного вопроса.

Тема 3.6. Визуализация метрик: создание дашбордов (Grafana, Superset).
Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам

ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.11. Осуществлять контроль качества данных в конвейерах обработки

Вопросы к устному опросу по теме:
1. Что такое дашборд?
Ответ: Дашборд (информационная панель) — это экран, на котором в наглядной форме собраны ключевые показатели и графики, характеризующие состояние системы, процесса или бизнеса. Дашборд позволяет быстро оценить ситуацию, заметить отклонения и принять решение без анализа исходных данных.

2. Для чего применяются дашборды при работе с конвейерами данных?
Ответ: Дашборды применяются для мониторинга работы конвейеров (успешности запусков, длительности, ошибок), контроля свежести и объёма данных, отслеживания показателей качества данных и дрейфа, анализа использования ресурсов, а также для предоставления руководству и заказчикам сведений о выполнении SLA.

3. Какие принципы эффективной визуализации данных вы знаете?
Ответ: Визуализация должна отвечать на конкретный вопрос пользователя, быть простой и не перегруженной, использовать подходящий тип графика, иметь понятные заголовки, подписи осей и единицы измерения, выделять главное цветом и размером, использовать цвет осмысленно и единообразно и не искажать данные, например неверным масштабом осей.

4. Что такое иерархия информации на дашборде?
Ответ: Иерархия информации — это расположение элементов дашборда по степени важности. Наиболее важные сводные показатели размещаются в верхней левой части экрана, так как пользователь начинает просмотр оттуда. Ниже располагаются графики динамики, а в нижней части — подробные таблицы и детализация.

5. Какие типы графиков применяются для визуализации метрик?
Ответ: Применяются линейные графики для динамики во времени, столбчатые диаграммы для сравнения категорий, круговые диаграммы для долей целого, гистограммы для распределений, диаграммы рассеяния для связи двух показателей, тепловые карты для двумерных распределений, таблицы для точных значений, а также отдельные числовые показатели и шкалы.

6. Когда используется линейный график?
Ответ: Линейный график используется для отображения изменения показателя во времени, например количества обработанных записей по часам, длительности запусков конвейера по дням или доли ошибок за неделю. Он хорошо показывает тренды, сезонность, резкие скачки и провалы.

7. Когда используется столбчатая диаграмма?
Ответ: Столбчатая диаграмма используется для сравнения значений показателя между категориями, например количества ошибок по разным конвейерам, объёма данных по источникам или доли пропусков по столбцам таблицы. Для длинных названий категорий удобнее горизонтальные столбцы.

8. Почему круговые диаграммы следует использовать осторожно?
Ответ: Человеку сложно точно сравнивать площади и углы секторов, особенно если секторов

много или их размеры близки. Круговые диаграммы допустимы только для показа долей небольшого количества категорий (2–5), составляющих целое. В остальных случаях лучше использовать столбчатую диаграмму.

9. Что такое числовой показатель (stat) и шкала (gauge) на дашборде?
Ответ: Числовой показатель отображает одно крупное значение метрики, например текущую долю успешных запусков, часто с цветовой индикацией и мини-графиком динамики. Шкала показывает значение относительно минимума, максимума и пороговых зон. Эти элементы используются для ключевых показателей в верхней части дашборда.

10. Что такое пороговые значения (thresholds) на дашборде?
Ответ: Пороговые значения — это границы, при пересечении которых изменяется цвет элемента визуализации. Например, доля успешных запусков выше 99% отображается зелёным, от 95 до 99% — жёлтым, ниже 95% — красным. Пороги позволяют сразу увидеть проблемные показатели.

11. Что такое Apache Superset?
Ответ: Apache Superset — это платформа бизнес-аналитики и визуализации данных с открытым исходным кодом. Она позволяет подключаться к различным базам данных, создавать графики без программирования, выполнять SQL-запросы, объединять графики в интерактивные дашборды и управлять доступом пользователей.

12. Как подключить базу данных в Superset?
Ответ: Подключение выполняется в разделе настроек подключений к базам данных. Указывается тип СУБД и строка подключения в формате SQLAlchemy URI с адресом сервера, портом, именем базы, логином и паролем, например postgresql://пользователь:пароль@сервер:5432/база. Для СУБД должен быть установлен соответствующий драйвер Python.

13. Что такое dataset в Superset?
Ответ: Dataset (набор данных) — это источник данных для построения графиков в Superset. Физический набор данных основан на таблице или представлении базы данных, виртуальный — на SQL-запросе. В наборе данных настраиваются типы столбцов, столбец времени, вычисляемые столбцы и метрики, например COUNT(*) или SUM(amount).

14. Что такое SQL Lab в Superset?
Ответ: SQL Lab — это встроенный редактор SQL-запросов в Superset, позволяющий исследовать данные, выполнять запросы к подключённым базам, просматривать результаты и схемы таблиц. Результат запроса можно сохранить как виртуальный набор данных и использовать для построения визуализаций.

15. Как создаются визуализации и дашборды в Superset?
Ответ: Для создания визуализации выбирается набор данных и тип графика, затем настраиваются метрики, измерения для группировки, фильтры, период времени и оформление. Готовые графики сохраняются и добавляются на дашборд, где их можно расположить, изменить размер и добавить общие фильтры для всех графиков.

16. Что такое Grafana?
Ответ: Grafana — это платформа с открытым исходным кодом для визуализации и мониторинга метрик, журналов и трассировок. Она подключается к различным источникам данных, позволяет создавать интерактивные дашборды в режиме реального времени и настраивать оповещения. Grafana широко применяется для мониторинга инфраструктуры и конвейеров данных.

17. Что такое data source в Grafana?
Ответ: Data source (источник данных) — это подключение Grafana к системе хранения данных, из

которой берутся значения для панелей. Grafana поддерживает Prometheus, Loki, Elasticsearch, PostgreSQL, MySQL, ClickHouse, InfluxDB и многие другие источники. Для каждого источника указываются адрес и параметры доступа.

18. Что такое panel в Grafana?
Ответ: Panel (панель) — это отдельный элемент дашборда, отображающий данные в виде определённой визуализации. Для панели задаются источник данных, запрос (например, на PromQL или SQL), тип визуализации, единицы измерения, пороговые значения и оформление. Основные типы панелей: Time series, Stat, Gauge, Bar chart, Table, Heatmap, Logs.

19. Что такое dashboard в Grafana и как он организуется?
Ответ: Dashboard — это набор панелей, расположенных на одном экране. Панели можно группировать в сворачиваемые строки (rows), задавать общий временной диапазон и интервал автоматического обновления, добавлять аннотации событий. Дашборды сохраняются в формате JSON, что позволяет экспортировать, импортировать и хранить их в системе контроля версий.

20. Что такое переменные (variables) в Grafana?
Ответ: Переменные — это параметры дашборда, значения которых пользователь выбирает из выпадающего списка в верхней части экрана, например имя конвейера, окружение или сервер. Переменные подставляются в запросы панелей в виде $имя_переменной, поэтому один дашборд можно использовать для разных объектов.

21. Какие типы переменных существуют в Grafana?
Ответ: Существуют переменные типа Query, значения которых получаются запросом к источнику данных, Custom со списком значений, заданным вручную, Interval для выбора интервала агрегации, Constant для скрытых постоянных значений, Text box для ввода произвольного значения, а также Data source для выбора источника данных.

22. Что такое templating в Grafana?
Ответ: Templating (шаблонизация) — это механизм использования переменных в запросах, заголовках и настройках панелей, позволяющий создавать универсальные дашборды. Например, при выборе конвейера все панели автоматически показывают его метрики, а функция повтора (repeat) создаёт отдельную панель для каждого выбранного значения переменной.

23. Как настраиваются оповещения (alerts) в Grafana?
Ответ: В Grafana создаются правила оповещения (alert rules), содержащие запрос к источнику данных, условие срабатывания и время, в течение которого условие должно выполняться. Точки контакта (contact points) определяют, куда отправлять оповещения: электронная почта, Slack, Telegram, PagerDuty. Политики уведомлений (notification policies) определяют маршрутизацию и группировку оповещений.

24. Что такое аннотации в Grafana?
Ответ: Аннотации — это отметки событий на графиках временных рядов, например развёртывания новой версии конвейера, сбоя или изменения настроек. Они позволяют сопоставить изменения метрик с событиями и быстрее находить причины проблем. Аннотации добавляются вручную или автоматически по запросу к источнику данных.

25. Чем Grafana отличается от Apache Superset?
Ответ: Grafana ориентирована на мониторинг в реальном времени и визуализацию временных рядов, метрик и журналов, имеет развитую систему оповещений и используется инженерами для контроля работы систем. Superset ориентирован на бизнес-аналитику и исследование данных из реляционных баз, имеет SQL Lab и подходит для аналитиков и бизнес-пользователей.

26. Какие лучшие практики создания дашбордов вы знаете?

Ответ: Дашборд должен создаваться для определённой аудитории и цели, содержать ограниченное количество панелей, начинаться со сводных показателей, использовать единые цвета и единицы измерения, иметь понятные названия и описания панелей, переменные для фильтрации, пороговые значения и ссылки на подробные дашборды. Дашборды следует хранить как код и периодически пересматривать.

27. Какие ошибки часто допускаются при создании дашбордов?
Ответ: Частые ошибки: перегруженность панелями, отсутствие подписей и единиц измерения, неверный тип графика, искажённый масштаб осей, использование большого количества несогласованных цветов, отображение показателей без пороговых значений и контекста, а также создание дашбордов, которые никто не использует для принятия решений.

28. Какие показатели отображаются на дашборде мониторинга конвейера данных?
Ответ: На дашборде мониторинга конвейера отображаются доля успешных запусков, количество неудачных запусков, длительность выполнения конвейеров и задач, время последнего успешного запуска и свежесть данных, объём обработанных данных, пропускная способность, количество повторных попыток, а также использование ресурсов.

29. Какие показатели отображаются на дашборде качества данных (Data Quality)?
Ответ: На дашборде качества данных отображаются интегральный показатель качества, полнота по таблицам и столбцам, доля дубликатов, доля записей, нарушающих правила валидации, количество выполненных и невыполненных проверок, свежесть данных, динамика объёма данных, показатели дрейфа признаков и список последних инцидентов качества.

30. Как обеспечить автоматическое развёртывание и сопровождение дашбордов?
Ответ: Дашборды и источники данных описываются в виде файлов конфигурации (в Grafana — механизм provisioning с файлами YAML и JSON) и хранятся в системе контроля версий. Это позволяет автоматически разворачивать одинаковые дашборды в разных окружениях, отслеживать изменения, восстанавливать прежние версии и проверять изменения перед внедрением.

Критерии оценки устного опроса:

- оценка «отлично» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов и смог ответить на уточняющие вопросы;
- оценка «хорошо» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов, но не смог ответить на уточняющие вопросы;
- оценка «удовлетворительно» выставляется студенту, если он смог изложить суть основного вопроса, но не смог ответить по существу дополнительных вопросов;
- оценка «неудовлетворительно» выставляется студенту, если он не смог изложить суть основного вопроса.

Практическое занятие №8. Настройка дашборда мониторинга конвейера и алертинга.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках

ПК 2.11. Осуществлять контроль качества данных в конвейерах обработки

Цель: Научиться настраивать сбор метрик конвейера данных, создавать в Grafana дашборд мониторинга конвейера с использованием переменных и пороговых значений, а также настраивать правила оповещения и каналы доставки уведомлений.

Задание:
1. Определить ключевые показатели мониторинга конвейера и их пороговые значения.
2. Настроить сбор и хранение метрик конвейера.
3. Подключить источники данных в Grafana.
4. Создать панели для отображения ключевых показателей.
5. Добавить переменные для фильтрации дашборда.
6. Настроить правила оповещения и точки контакта.
7. Проверить срабатывание оповещений, смоделировав сбой конвейера.
8. Сделать вывод о готовности системы мониторинга.

Этапы выполнения:
Этап 1. Определение показателей.
Определяется, какие показатели характеризуют работу конвейера: доля успешных запусков, количество ошибок, длительность выполнения, свежесть данных, объём обработанных записей. Для каждого показателя устанавливаются пороговые значения в соответствии с SLO.

Этап 2. Настройка сбора метрик.
Настраивается передача метрик конвейера в систему хранения. Например, Airflow отправляет метрики по протоколу StatsD в statsd-exporter, откуда их собирает Prometheus, а скрипты обработки записывают сведения о каждом запуске (время, статус, количество записей) в таблицу PostgreSQL. Все компоненты могут разворачиваться с помощью Docker Compose.

Этап 3. Подключение источников данных.
Grafana разворачивается и открывается веб-интерфейс. Добавляются источники данных Prometheus и PostgreSQL, проверяется соединение с каждым из них.

Этап 4. Создание панелей.
Создаётся новый дашборд. В верхней строке размещаются панели типа Stat с долей успешных запусков, количеством ошибок за сутки и временем с момента последнего успешного запуска. Ниже размещаются панели Time series с длительностью запусков и объёмом обработанных записей, а также таблица последних неудачных запусков. Для панелей задаются единицы измерения и пороговые значения с цветовой индикацией.

Этап 5. Добавление переменных.
Создаётся переменная pipeline типа Query, значения которой получаются запросом списка конвейеров из источника данных. Переменная подставляется в запросы всех панелей, что позволяет просматривать показатели выбранного конвейера.

Этап 6. Настройка оповещений.
Создаются точки контакта, например электронная почта и Telegram или Slack. Создаются правила оповещения для ключевых показателей с условиями и длительностью срабатывания, настраивается политика уведомлений с группировкой и интервалом повторов.

Этап 7. Проверка работы оповещений.
Моделируется сбой: в DAG вносится ошибка, и конвейер запускается. Проверяется отображение сбоя на дашборде и поступление оповещения в выбранный канал. После исправления ошибки проверяется отправка уведомления о восстановлении.

Этап 8. Оформление результата.
Дашборд экспортируется в формате JSON. Оформляется отчёт, содержащий перечень показателей и порогов, описание настройки сбора метрик, снимки дашборда, настройки правил оповещения, результаты проверки и выводы.

Пример решения:
Задание: Настроить дашборд мониторинга и оповещения для конвейеров загрузки данных о продажах.
Дано: В Airflow работают три DAG: загрузка заказов, загрузка товаров и расчёт витрины продаж. Данные должны обновляться ежедневно до 06:00. Требуется отслеживать работу конвейеров и оповещать дежурного инженера о сбоях.
Решение по этапам:
1. Определяем показатели и пороги: доля успешных запусков за 7 дней — зелёный от 99%, жёлтый от 95%, красный ниже 95%; время с последнего успешного запуска — красный при превышении 26 часов; длительность запуска — предупреждение при превышении 60 минут.
2. Включаем отправку метрик Airflow по StatsD, разворачиваем statsd-exporter и Prometheus в Docker Compose. Дополнительно создаём в PostgreSQL таблицу pipeline_runs, в которую каждый DAG записывает имя, время начала и окончания, статус и количество обработанных записей.
3. Разворачиваем Grafana, добавляем источники данных Prometheus и PostgreSQL, проверяем подключения.
4. Создаём дашборд «Мониторинг конвейеров продаж». В верхней строке размещаем три панели Stat: доля успешных запусков, количество ошибок за сутки, время с последнего успешного запуска.
5. Добавляем панель Time series с длительностью запусков по дням с пороговой линией 60 минут и столбчатую диаграмму количества обработанных записей по дням.
6. Добавляем таблицу последних 10 неудачных запусков с именем конвейера, временем и сообщением об ошибке.
7. Создаём переменную pipeline со значениями из таблицы pipeline_runs и подставляем её в запросы всех панелей.
8. Создаём точки контакта: электронную почту дежурного и канал Telegram. Создаём правила оповещения: неудачный запуск конвейера, отсутствие успешного запуска более 26 часов, длительность запуска более 60 минут. Настраиваем группировку оповещений по конвейеру и повтор каждые 4 часа.
9. Вносим ошибку в DAG загрузки заказов и запускаем его. На дашборде доля успешных запусков становится жёлтой, в таблице появляется запись об ошибке, в Telegram и на почту приходит оповещение. После исправления приходит уведомление о восстановлении.
10. Экспортируем дашборд в JSON и оформляем отчёт.

Ответ: Настроены сбор метрик конвейеров через StatsD и Prometheus и запись сведений о запусках в PostgreSQL. В Grafana создан дашборд с ключевыми показателями, пороговыми значениями, графиками динамики, таблицей ошибок и переменной для выбора конвейера. Настроены правила оповещения о сбоях, устаревании данных и длительных запусках с доставкой в Telegram и на электронную почту. Проверка показала корректное отображение сбоя и своевременное поступление оповещений.

Критерии оценки практических заданий:
1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Тема 3.7. Безопасность данных и управление доступом в конвейерах.

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.11. Осуществлять контроль качества данных в конвейерах обработки

Вопросы к устному опросу по теме:

1. Что понимают под безопасностью данных в конвейерах?
Ответ: Безопасность данных в конвейерах — это совокупность мер, обеспечивающих конфиденциальность, целостность и доступность данных на всех этапах их обработки: при получении из источников, передаче, преобразовании, хранении и предоставлении пользователям. Она включает защиту от утечек, несанкционированного доступа, изменения и потери данных.

2. Что такое конфиденциальность, целостность и доступность данных?
Ответ: Конфиденциальность означает, что доступ к данным имеют только уполномоченные лица. Целостность означает, что данные не были изменены или повреждены несанкционированным образом. Доступность означает, что данные доступны уполномоченным пользователям тогда, когда они нужны. Эти три свойства составляют основу информационной безопасности.

3. Какие основные угрозы безопасности существуют для конвейеров данных?
Ответ: Основными угрозами являются утечки данных, несанкционированный доступ к данным и системам, инъекции вредоносного кода, перехват данных при передаче, кража учётных данных, ошибки настройки прав доступа, внутренние угрозы со стороны сотрудников, а также атаки на цепочку поставок через уязвимые библиотеки.

4. Что такое утечка данных и каковы её причины?
Ответ: Утечка данных — это несанкционированное раскрытие или передача данных лицам, не имеющим права доступа к ним. Причинами могут быть ошибки настройки хранилищ (например, открытый для всех облачный бакет), пароли в коде и журналах, использование реальных персональных данных в тестовых средах, действия злоумышленников и недобросовестных сотрудников.

5. Что такое несанкционированный доступ?
Ответ: Несанкционированный доступ — это получение доступа к данным или системам лицом, не имеющим на это права, либо выход пользователя за пределы предоставленных ему прав. Он возникает из-за слабых паролей, украденных учётных данных, избыточных прав, уязвимостей программ и отсутствия контроля доступа.

6. Что такое SQL-инъекция и как от неё защититься?
Ответ: SQL-инъекция — это атака, при которой злоумышленник внедряет вредоносный фрагмент SQL-кода через входные данные, которые подставляются в запрос без проверки. Это позволяет читать, изменять или удалять данные. Для защиты используются параметризованные запросы, проверка входных данных и минимальные права учётной записи базы данных.

7. Какие ещё виды инъекций возможны в конвейерах данных?
Ответ: Возможны инъекции команд операционной системы, когда входные данные подставляются в команды оболочки, например в BashOperator; инъекции в шаблоны, когда пользовательские данные попадают в шаблоны Jinja; а также внедрение вредоносного кода через

загружаемые файлы и сторонние библиотеки.

8. Что такое принцип наименьших привилегий (least privilege)?
Ответ: Принцип наименьших привилегий означает, что каждый пользователь, сервис или процесс получает только те права, которые необходимы для выполнения его задач, и только на необходимое время. Например, учётная запись конвейера загрузки может только записывать в определённые таблицы, но не удалять другие таблицы.

9. Как применяется принцип наименьших привилегий в конвейерах данных?
Ответ: Для каждого конвейера создаётся отдельная служебная учётная запись с ограниченными правами, подключения к базам данных используют пользователей только с нужными правами на чтение или запись, доступ к промышленным данным предоставляется ограниченному кругу лиц, права периодически пересматриваются, а временные права отзываются после выполнения задачи.

10. Что такое принцип эшелонированной защиты (defense in depth)?
Ответ: Принцип эшелонированной защиты означает применение нескольких независимых уровней защиты, чтобы при преодолении одного уровня данные оставались защищены другими. Например, сочетаются сетевые ограничения, аутентификация, разграничение прав, шифрование, маскирование данных, аудит и мониторинг.

11. Что такое шифрование данных при хранении (encryption at rest)?
Ответ: Шифрование при хранении — это шифрование данных, записанных на диски, в базы данных, файлы и облачные хранилища. Даже при краже носителя или резервной копии злоумышленник не сможет прочитать данные без ключа. Применяется шифрование дисков, прозрачное шифрование баз данных и шифрование на стороне облачного хранилища.

12. Что такое шифрование данных при передаче (encryption in transit)?
Ответ: Шифрование при передаче — это защита данных при их перемещении по сети между компонентами системы с помощью протоколов TLS и SSL. Оно предотвращает перехват и изменение данных. В конвейерах шифруются соединения с базами данных, обращения к API, обмен с Kafka и доступ к веб-интерфейсам Airflow и Grafana по HTTPS.

13. Как организуется управление ключами шифрования?
Ответ: Ключи шифрования хранятся отдельно от зашифрованных данных в специализированных системах управления ключами, например облачных сервисах KMS или HashiCorp Vault. Доступ к ключам строго ограничивается, ключи периодически заменяются (ротация), а все операции с ними фиксируются в журналах аудита.

14. Как безопасно хранить пароли и секреты в конвейерах данных?
Ответ: Пароли, токены и ключи нельзя хранить в коде и репозиториях. Их хранят в подключениях и переменных Airflow с шифрованием, в переменных окружения или во внешних хранилищах секретов, например HashiCorp Vault или облачных менеджерах секретов, которые Airflow поддерживает как бэкенды секретов. Секреты также маскируются в журналах.

15. Что такое маскирование и обезличивание данных?
Ответ: Маскирование — это замена части данных условными символами, например номер карты отображается как **** **** **** 1234. Обезличивание — это преобразование персональных данных так, чтобы без дополнительной информации невозможно было определить, к какому человеку они относятся. Эти методы применяются в тестовых средах и при предоставлении данных аналитикам.

16. Что такое RBAC?
Ответ: RBAC (Role-Based Access Control) — это управление доступом на основе ролей. Права

доступа назначаются не отдельным пользователям, а ролям, например «аналитик», «инженер данных», «администратор», а пользователи получают права через назначенные им роли. Это упрощает управление правами в больших организациях.

17. Как реализован RBAC в Airflow?
Ответ: В Airflow есть встроенные роли с разным уровнем прав, например Admin, Op, User, Viewer и Public. Роль Viewer позволяет только просматривать DAG, User — запускать и управлять ими, Admin имеет полный доступ, включая управление пользователями и подключениями. Можно создавать собственные роли с доступом только к определённым DAG.

18. Что такое ABAC?
Ответ: ABAC (Attribute-Based Access Control) — это управление доступом на основе атрибутов. Решение о доступе принимается по правилам, учитывающим атрибуты пользователя (отдел, должность), ресурса (уровень конфиденциальности, владелец), действия и окружения (время, местоположение). Например, сотрудник отдела продаж может видеть данные только своего региона в рабочее время.

19. Чем ABAC отличается от RBAC?
Ответ: RBAC проще в настройке и понимании, но при сложных требованиях приводит к большому количеству ролей. ABAC позволяет задавать гибкие и детальные правила доступа без создания множества ролей, но сложнее в проектировании и проверке. На практике часто сочетают оба подхода: роли задают основные права, атрибуты — уточняющие ограничения.

20. Что такое Apache Ranger?
Ответ: Apache Ranger — это система централизованного управления политиками доступа и аудита для экосистемы больших данных: HDFS, Hive, HBase, Kafka и других. Она позволяет задавать детальные политики доступа на уровне баз, таблиц, столбцов и строк, маскировать данные, задавать политики на основе тегов и собирать журналы аудита всех обращений.

21. Что такое AWS IAM?
Ответ: AWS IAM (Identity and Access Management) — это сервис облака Amazon Web Services для управления доступом к ресурсам облака. В IAM создаются пользователи, группы и роли, а права задаются политиками в формате JSON, где указываются разрешённые или запрещённые действия, ресурсы и условия. Для сервисов используются роли с временными учётными данными.

22. Какие лучшие практики применяются при работе с AWS IAM?
Ответ: Не используется учётная запись администратора для повседневной работы, включается многофакторная аутентификация, права назначаются по принципу наименьших привилегий, сервисы получают доступ через роли, а не через постоянные ключи, неиспользуемые учётные данные удаляются, а права регулярно анализируются.

23. Что такое аудит доступа?
Ответ: Аудит доступа — это регистрация и анализ всех действий пользователей и сервисов с данными и системами: кто, когда, к каким данным обращался и какие действия выполнял. Аудит позволяет обнаруживать подозрительную активность, расследовать инциденты и подтверждать соблюдение требований законодательства.

24. Как организуется аудит доступа в конвейерах данных?
Ответ: Включается журналирование действий в базах данных, хранилищах и облачных сервисах, например AWS CloudTrail, используются журналы аудита Apache Ranger и журнал событий Airflow. Журналы аудита собираются централизованно, защищаются от изменения, хранятся установленный срок и анализируются, а на подозрительные события настраиваются оповещения.

25. Что такое compliance в области данных?
Ответ: Compliance (комплаенс) — это соблюдение требований законодательства, отраслевых стандартов и внутренних политик при обработке данных. Для конвейеров данных это означает выполнение требований к сбору, хранению, обработке, защите и удалению персональных и других охраняемых данных.

26. Что такое GDPR?
Ответ: GDPR (General Data Protection Regulation) — это Общий регламент защиты данных Европейского союза, действующий с 2018 года. Он регулирует обработку персональных данных жителей ЕС и применяется также к компаниям из других стран, обрабатывающим такие данные. За нарушения предусмотрены штрафы до 20 миллионов евро или 4% годового мирового оборота компании.

27. Какие основные требования устанавливает GDPR?
Ответ: GDPR требует законного основания для обработки данных, обработки только необходимого объёма данных для определённых целей, обеспечения защиты данных, учёта защиты данных при проектировании систем (privacy by design), уведомления надзорного органа об утечке в течение 72 часов, а также обеспечения прав субъектов: на доступ к данным, исправление, удаление («право на забвение») и перенос данных.

28. Что такое Федеральный закон № 152-ФЗ?
Ответ: Федеральный закон от 27.07.2006 № 152-ФЗ «О персональных данных» — это основной закон Российской Федерации, регулирующий обработку персональных данных. Он устанавливает принципы обработки, условия получения согласия субъектов, права субъектов персональных данных, обязанности операторов по защите данных и ответственность за нарушения.

29. Какие основные требования 152-ФЗ необходимо учитывать в конвейерах данных?
Ответ: Необходимо обрабатывать персональные данные только при наличии законного основания, например согласия субъекта, и только для заявленных целей, обеспечивать запись, хранение и накопление персональных данных граждан РФ в базах данных на территории России (локализация), принимать меры защиты в соответствии с уровнем защищённости, уведомлять Роскомнадзор об обработке и в установленные сроки сообщать об утечках, а также уничтожать данные после достижения целей обработки.

30. Как обеспечить соответствие конвейеров данных требованиям законодательства о персональных данных?
Ответ: Необходимо провести инвентаризацию персональных данных в конвейерах, минимизировать их сбор, применять обезличивание и маскирование, шифровать данные при хранении и передаче, разграничивать доступ, вести аудит, хранить данные в разрешённых местах, определить сроки хранения и процедуры удаления, а также документально оформить политику обработки персональных данных.

Критерии оценки устного опроса:

- оценка «отлично» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов и смог ответить на уточняющие вопросы;
- оценка «хорошо» выставляется студенту, если он смог изложить суть основного вопроса, смог ответить по существу дополнительных вопросов, но не смог ответить на уточняющие вопросы;
- оценка «удовлетворительно» выставляется студенту, если он смог изложить суть основного вопроса, но не смог ответить по существу дополнительных вопросов;
- оценка «неудовлетворительно» выставляется студенту, если он не смог изложить суть основного вопроса.

Практическое занятие №9. Интеграция проверок Data Quality внутрь рабочего DAG (Airflow).

Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.11. Осуществлять контроль качества данных в конвейерах обработки

Цель: Научиться встраивать проверки качества данных в рабочий DAG Apache Airflow, организовывать ветвление конвейера по результатам проверок, изолировать некачественные данные и оповещать ответственных о нарушениях.

Задание:
1. Проанализировать рабочий DAG и определить точки, в которых необходимы проверки качества данных.
2. Сформулировать правила качества данных для каждой точки проверки.
3. Реализовать проверки с помощью Great Expectations, Pandera или SQL-операторов проверки Airflow.
4. Встроить задачи проверки качества в DAG и задать зависимости.
5. Организовать ветвление конвейера в зависимости от результата проверки.
6. Настроить сохранение результатов проверок и оповещения о нарушениях.
7. Проверить работу DAG на корректных и ошибочных данных.
8. Сделать вывод о влиянии проверок на надёжность конвейера.

Этапы выполнения:
Этап 1. Анализ рабочего DAG.
Изучаются задачи существующего DAG и потоки данных между ними. Определяются точки проверки: после извлечения данных из источника (проверка входных данных), после преобразования (проверка результата обработки) и после загрузки в хранилище (проверка целевой таблицы).

Этап 2. Формулирование правил качества.
Для каждой точки проверки формулируются правила: наличие обязательных столбцов, отсутствие пропусков в ключевых полях, уникальность идентификаторов, допустимые диапазоны и форматы значений, ссылочная целостность, ожидаемое количество строк. Для правил определяется уровень критичности: блокирующие и предупреждающие.

Этап 3. Реализация проверок.
Проверки входных данных реализуются набором ожиданий Great Expectations или схемой Pandera, запускаемыми в задаче PythonOperator. Проверки целевой таблицы реализуются операторами SQLColumnCheckOperator и SQLTableCheckOperator из провайдера common.sql или SQL-запросами.

Этап 4. Встраивание проверок в DAG.
Задачи проверки добавляются в DAG между задачами извлечения, преобразования и загрузки. Задаются зависимости так, чтобы обработка продолжалась только после выполнения проверок. Подключения к базам данных используют учётную запись с минимально необходимыми правами.

Этап 5. Ветвление по результатам проверки.
С помощью BranchPythonOperator или декоратора @task.branch конвейер направляется по одной из ветвей: при успешной проверке — к дальнейшей обработке, при нарушении блокирующих правил — к задаче перемещения данных в карантин. Для конечной задачи настраивается подходящее правило запуска (trigger rule).

Этап 6. Сохранение результатов и оповещения.
Результаты каждой проверки (дата, DAG, таблица, количество выполненных и невыполненных правил, процент ошибочных записей) сохраняются в таблицу журнала качества данных в PostgreSQL. Настраивается функция обратного вызова при ошибке (on_failure_callback) и задача оповещения, отправляющая сообщение по электронной почте или в мессенджер.

Этап 7. Проверка работы DAG.
DAG запускается на корректных данных — все проверки проходят, данные загружаются. Затем в данные вносятся ошибки, и DAG запускается повторно — конвейер переходит в ветвь карантина, загрузка не выполняется, отправляется оповещение, результаты фиксируются в журнале качества.

Этап 8. Оформление результата.
Оформляется отчёт, содержащий схему DAG до и после доработки, перечень правил качества, код задач проверки и ветвления, снимки веб-интерфейса Airflow для обоих сценариев, содержимое журнала качества и выводы.

Пример решения:
Задание: Встроить проверки качества данных в DAG ежедневной загрузки данных о заказах в хранилище.
Дано: Рабочий DAG orders_etl содержит задачи extract (выгрузка заказов из API в файл), transform (очистка и расчёт сумм) и load (загрузка в таблицу orders в PostgreSQL). Проверки качества отсутствуют, поэтому ошибочные данные периодически попадают в отчёты.

Решение по этапам:
1. Определяем точки проверки: после extract, после transform и после load.
2. Формулируем правила для входных данных: столбцы order_id, customer_id, order_date, amount присутствуют; order_id заполнен и уникален; amount от 0 до 1 000 000; строк не менее 100. Для результата преобразования: отсутствие пропусков, корректный тип даты. Для целевой таблицы: отсутствие дубликатов order_id, ссылочная целостность с таблицей customers.
3. Создаём задачу validate_raw с PythonOperator, запускающую контрольную точку Great Expectations с набором ожиданий orders_raw_suite и возвращающую результат через XCom.
4. Создаём задачу ветвления check_result с декоратором @task.branch: при успешной проверке возвращается идентификатор задачи transform, при ошибке — задачи quarantine.
5. Создаём задачу quarantine, которая перемещает файл в папку карантина, и задачу notify, отправляющую оповещение с перечнем нарушенных правил.
6. После transform добавляем задачу validate_transformed с проверкой схемы Pandera.
7. После load добавляем задачи SQLColumnCheckOperator (уникальность и отсутствие пропусков order_id) и SQLTableCheckOperator (количество строк за дату загрузки больше 0, отсутствие заказов с несуществующими клиентами).
8. Добавляем задачу log_dq, которая записывает результаты проверок в таблицу dq_log, с правилом запуска all_done, чтобы она выполнялась при любом исходе. Задаём on_failure_callback для отправки оповещения при падении любой задачи.
9. Запускаем DAG на корректных данных: все проверки пройдены, данные загружены, в dq_log записан успешный результат.
10. Вносим в данные дубликаты и отрицательные суммы, запускаем DAG: проверка validate_raw не пройдена, конвейер переходит в ветвь quarantine, загрузка не выполняется, приходит

оповещение, в dq_log записаны нарушенные правила.

Ответ: В DAG orders_etl встроены проверки качества на трёх этапах: входных данных с помощью Great Expectations, результата преобразования с помощью Pandera и целевой таблицы с помощью SQL-операторов проверки Airflow. Организовано ветвление с перемещением некачественных данных в карантин, журналирование результатов проверок и оповещение ответственных. Проверки предотвращают попадание ошибочных данных в хранилище и повышают надёжность конвейера.

Критерии оценки практических заданий:
1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

Практическое занятие №10. Итоговый комплексный проект: полный конвейер (от загрузки до мониторинга с проверкой качества).
Проверяемые компетенции:
ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам
ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности
ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста
ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках
ПК 2.11. Осуществлять контроль качества данных в конвейерах обработки

Цель: Закрепить знания и умения, полученные при изучении модуля, путём разработки полного конвейера обработки данных для ИИ-системы, включающего загрузку, проверку качества, обработку, хранение, оркестрацию, контейнеризацию, мониторинг, оповещение и защиту данных.

Задание:
1. Выбрать предметную область и сформулировать задачу конвейера.
2. Спроектировать архитектуру конвейера и выбрать инструменты.
3. Развернуть инфраструктуру конвейера с помощью Docker Compose.
4. Реализовать загрузку, очистку и преобразование данных.
5. Встроить проверки качества данных на ключевых этапах.
6. Реализовать оркестрацию конвейера в Apache Airflow.
7. Настроить мониторинг, дашборды и оповещения, обеспечить защиту данных и разграничение доступа.
8. Провести испытания конвейера, подготовить документацию и защитить проект.

Этапы выполнения:
Этап 1. Постановка задачи.
Выбирается предметная область, например интернет-торговля, банковские операции или данные датчиков. Определяются источники данных, потребители результатов (отчёты, модель машинного обучения), требования к периодичности, свежести и качеству данных, а также SLO конвейера.

Этап 2. Проектирование архитектуры.

Составляется схема конвейера с указанием этапов и инструментов: источники данных, загрузка, проверка качества, обработка (Pandas или PySpark), хранилище (PostgreSQL), формирование датасета для модели, оркестрация (Airflow), мониторинг (Prometheus, Grafana). Выбирается архитектура ETL или ELT и обосновывается выбор.

Этап 3. Развёртывание инфраструктуры.
Создаётся файл docker-compose.yaml с сервисами Airflow, PostgreSQL, при необходимости Spark, Prometheus и Grafana. Настраиваются volumes для сохранения данных, переменные окружения и проверки работоспособности сервисов. Пароли выносятся в переменные окружения и не попадают в репозиторий.

Этап 4. Реализация обработки данных.
Разрабатываются скрипты загрузки данных из источников, очистки (пропуски, дубликаты, выбросы), преобразования и формирования признаков. Результаты сохраняются в хранилище и в формате Parquet. Подготовленный датасет разделяется на выборки и версионируется с помощью DVC.

Этап 5. Проверки качества данных.
Формулируются правила качества и реализуются с помощью Great Expectations, Pandera или SQL-проверок на этапах загрузки, преобразования и загрузки в хранилище. Результаты проверок сохраняются в журнал качества, формируются отчёты Data Docs. Дополнительно реализуется расчёт показателей дрейфа ключевых признаков.

Этап 6. Оркестрация.
Создаётся DAG, объединяющий все этапы конвейера с зависимостями, ветвлением по результатам проверок, повторными попытками, расписанием и оповещениями при сбоях. Подключения к системам настраиваются через Connections Airflow.

Этап 7. Мониторинг и безопасность.
Настраивается сбор метрик конвейера и создаётся дашборд Grafana с показателями успешности, длительности, свежести и качества данных. Настраиваются правила оповещения. Для доступа к базе данных создаются учётные записи с минимальными правами, в Airflow назначаются роли пользователям, персональные данные маскируются или обезличиваются.

Этап 8. Испытания и защита проекта.
Проводятся испытания: штатный запуск, запуск с ошибочными данными, моделирование сбоя источника и проверка восстановления после перезапуска контейнеров. Подготавливается документация: описание архитектуры, инструкция по развёртыванию, описание правил качества и показателей мониторинга. Проект защищается с демонстрацией работы конвейера.

Пример решения:
Задание: Разработать полный конвейер подготовки данных о продажах интернет-магазина для модели прогнозирования спроса.
Дано: Данные о заказах ежедневно выгружаются из API интернет-магазина в формате JSON, справочник товаров хранится в файле CSV. Подготовленные данные должны ежедневно до 06:00 загружаться в хранилище и использоваться для переобучения модели прогнозирования спроса. Данные содержат персональные сведения покупателей.

Решение по этапам:
1. Формулируем задачу: ежедневная загрузка заказов, проверка качества, расчёт признаков продаж по товарам, загрузка в хранилище, мониторинг. SLO: данные доступны до 06:00 в 99% дней, доля успешных запусков не ниже 99%.
2. Проектируем архитектуру ETL: API и CSV → загрузка → проверка входных данных → очистка и формирование признаков → проверка результата → загрузка в PostgreSQL → проверка

целевой таблицы → расчёт дрейфа → обновление метрик.
3. Создаём docker-compose.yaml с сервисами postgres, airflow-scheduler, airflow-webserver, prometheus, statsd-exporter и grafana, настраиваем тома для данных PostgreSQL, DAG и журналов. Пароли храним в файле .env, исключённом из Git.
4. Реализуем задачу extract, сохраняющую ответ API в файл JSON, и задачу transform на Pandas: удаление дубликатов, заполнение пропусков, удаление заказов с отрицательной суммой, замена имени и телефона покупателя хеш-идентификатором, расчёт признаков (продажи за 7 и 30 дней, день недели, средняя цена).
5. Реализуем проверки: набор ожиданий Great Expectations для входных данных, схему Pandera для результата преобразования, SQL-проверки целевой таблицы. Результаты записываются в таблицу dq_log. Рассчитываем PSI ключевых признаков относительно эталонного периода.
6. Создаём DAG sales_pipeline с расписанием 0 3 * * *, тремя повторными попытками, ветвлением в карантин при нарушении правил качества и оповещением в Telegram при сбое.
7. Настраиваем экспорт метрик Airflow в Prometheus и создаём дашборд Grafana с долей успешных запусков, длительностью, временем последнего обновления, долей ошибочных записей и значениями PSI. Настраиваем оповещения о сбое, задержке и превышении PSI 0,25.
8. Создаём в PostgreSQL пользователя etl_writer с правами записи только в схему хранилища и пользователя analyst_reader только с правом чтения витрин. В Airflow аналитикам назначаем роль Viewer.
9. Проводим испытания: штатный запуск выполнен за 12 минут; при подаче данных с дубликатами конвейер перешёл в ветвь карантина и отправил оповещение; при недоступности API сработали повторные попытки и оповещение; после перезапуска контейнеров история запусков и данные сохранились.
10. Подготавливаем документацию и презентацию проекта, демонстрируем работу конвейера и дашборда.

Ответ: Разработан полный конвейер подготовки данных о продажах, развёрнутый в Docker Compose и управляемый Airflow. Конвейер выполняет загрузку, очистку и формирование признаков, проверяет качество данных на трёх этапах, изолирует ошибочные данные, обезличивает персональные данные, загружает результаты в PostgreSQL и отслеживает дрейф. Работа конвейера отображается на дашборде Grafana с оповещениями о сбоях, доступ разграничен по принципу наименьших привилегий. Испытания подтвердили корректность и надёжность работы конвейера.

Критерии оценки практических заданий:
1. Полнота и правильность ответа – 1 балл.
2. Логичность и последовательность изложения – 1 балл.
3. Соответствие теме задания – 1 балл.
4. Анализ и выводы по теме – 1 балл.
5. Грамотность и аккуратность оформления – 1 балл.

4. Контроль приобретения практического опыта. Оценка по учебной и (или) производственной практике (по профилю специальности)

4.1. Общие положения

Целью оценки по учебной и (или) производственной практике (по профилю специальности) является оценка: 1) профессиональных и общих компетенций; 2) практического опыта и умений.

Оценка по учебной и (или) производственной практике выставляется на основании данных аттестационного листа (характеристики профессиональной деятельности обучающегося/студента на практике) с указанием видов работ, выполненных обучающимся во время практики, их объема, качества выполнения в соответствии с технологией и (или) требованиями организации, в которой проходила практика.

4.2. Виды работ практики и проверяемые результаты обучения по профессиональному модулю[1]

Таблица 8

Иметь практический опыт	Виды и объем работ на учебной практике, требования к их выполнению и/ или условия выполнения	Документ, подтверждающий качество выполнения работ
1	2	3
- Идентификации проблем, связанных с нормальным функционированием базы данных; - Восстановления системы. - Администрирования сервера баз данных; - Участия в администрировании отдельных компонент серверов; - Документирования результатов аудита безопасности информации; - Использования процедуры резервного копирования баз данных; - Использования процедуры восстановления баз данных - Подготовки документации по формированию требований хранилищ банка данных - Проектирования, разработки и эксплуатации баз данных - Планирование процедур резервного копирования данных - Запуск процедуры резервного копирования данных - Мониторинг выполнения процедур резервного копирования данных - Контроль завершения процедуры резервного копирования данных - Проведение повторной	1. Установка СУБД и настройка параметров серверной службы. 2. Настройка средств администрирования и анализ выполнения запросов для выявления и устранения проблем. 3. Создание учетных записей, ролей и назначение прав доступа в СУБД. 4. Настройка аутентификации, аудита и контроль соблюдения политик безопасности. 5. Планирование и выполнение резервного копирования, мониторинг и хранение резервных копий. 6. Восстановление базы данных из резервной копии, проверка целостности и документирование результата. 7. Анализ предметной области, разработка ER-модели и документации	Аттестационный лист о прохождении практики.

процедуры резервного копирования данных в случае ее нештатного завершения - Хранение резервных копий БД - Назначение приоритета запросов к БД - Оптимизация выполнения статистически значимых запросов к БД - Создание программного кода в соответствии с техническим заданием (готовыми спецификациями) - Оптимизация программного кода с использованием специализированных программных средств - Определение источников больших данных для анализа, идентификация внешних и внутренних источников данных для проведения аналитических работ - Получение и фильтрация больших объемов данных из гетерогенных источников - Извлечение, проверка и очистка больших объемов данных из гетерогенных источников - Агрегация и разработка представления больших объемов данных из гетерогенных источников - Оценка соответствия набора данных предметной области и задачам аналитических работ - Возможности имеющейся у исполнителя методологической и технологической инфраструктуры анализа больших данных - Предметная область анализа - Теоретические и прикладные основы анализа больших данных - Современные методы и инструментальные средства анализа больших данных - Современный опыт использования анализа больших данных - Осуществление стратегического и оперативного руководства кадровыми и материальными ресурсами в процессах получения, размещения, хранения, распределения, миграции, архивирования и удаления данных - Осуществление стратегического и оперативного руководства материально-техническими ресурсами и элементами технологической	требований к базе данных. 8. Разработка логической и физической моделей, нормализация данных, определение ключей и связей. 9. Создание таблиц и ограничений целостности средствами SQL, заполнение и импорт данных. 10. Разработка SQL-запросов, процедур и триггеров, анализ плана выполнения и настройка индексов. 11. Реализация ролевой модели безопасности, настройка аудита и защиты данных. 12. Выполнение CRUD-операций и агрегации в NoSQL-базе, настройка индексов и программного кода доступа к данным. 13. Профилирование и классификация источников данных, определение требований к качеству и формату набора данных. 14. Очистка, преобразование, разметка и формирование датасета, контроль качества результата. 15. Разработка DAG с задачами, зависимостями, расписанием и подключениями в Apache Airflow. 16. Реализация обработки данных в PySpark и контейнеризация конвейера в Docker Compose. 17. Настройка автоматических проверок качества и обнаружения аномалий данных. 18. Настройка мониторинга, дашборда, алертинга и контроля доступа к данным конвейера.	

инфраструктуры больших данных - Разработка моделей данных, адаптированных к технологиям больших данных - Интеграция больших данных с системами хранения данных организации - Создание и интеграция облачных репозиториев с системами хранения данных организации - Создание параллельных систем хранения и обработки информации - Разработка методов и регламентов реализации различных режимов обработки больших данных, в том числе режима реального времени - Мониторинг и оценка производительности обработки данных в организации, разработка предложений по повышению производительности обработки больших данных - Разработка методов и регламентов преобразования данных - Разработка и согласование стратегии формирования информационных активов организации - Разработка методов и регламентов оценки информационных активов организации - Формирование состава и координация деятельности персонала, отвечающего за приобретение и получение больших данных - Оценка эффективности системы хранения и обработки данных организации - Разработка предложений по развитию и совершенствованию системы получения, хранения, передачи, обработки больших данных - Формирование целей, приоритетов и ограничений управления качеством данных, адаптация целей к изменениям внутренних и внешних условий, потребностей и возможностей, в том числе технологических - Разработка стратегии управления качеством больших данных - Выделение ресурсов для управления качеством больших данных		

- Реализация стратегии обеспечения качества больших данных - Анализ реализации стратегии обеспечения качества больших данных - Проведение оценки качества больших данных - Разрешение проблем и инцидентов с качеством больших данных - Оценка рисков использования больших данных		

Таблица 9

Иметь практический опыт	**Виды и объем работ на производственной практике (по профилю специальности), требования к их выполнению и/ или условия выполнения**	**Документ, подтверждающий качество выполнения работ**
1	**2**	**3**
- Идентификации проблем, связанных с нормальным функционированием базы данных; - Восстановления системы. - Администрирования сервера баз данных; - Участия в администрировании отдельных компонент серверов; - Документирования результатов аудита безопасности информации; - Использования процедуры резервного копирования баз данных; - Использования процедуры восстановления баз данных - Подготовки документации по формированию требований хранилищ банка данных - Проектирования, разработки и эксплуатации баз данных - Планирование процедур резервного копирования данных - Запуск процедуры резервного копирования данных - Мониторинг выполнения процедур резервного копирования данных - Контроль завершения процедуры резервного копирования данных - Проведение повторной процедуры резервного копирования данных в случае ее нештатного завершения - Хранение резервных копий БД - Назначение приоритета запросов к	1. Установка и конфигурирование СУБД, серверных служб и инструментов администратора базы данных. 2. Настройка программного обеспечения и клиентских средств для подключения пользователей к базе данных. 3. Создание учетных записей, ролей и политик доступа; настройка прав и параметров безопасности базы данных. 4. Планирование и выполнение резервного копирования, контроль завершения операций и организация хранения копий. 5. Выполнение восстановления базы данных и проверка восстановимости резервных копий в штатных и нештатных ситуациях. 6. Мониторинг событий и производительности БД, диагностика проблем и автоматизация регламентных операций администрирования. 7. Анализ требований и разработка концептуальной, логической и физической моделей базы данных.	Аттестационный лист о прохождении практики, характеристика с места практики.

БД - Оптимизация выполнения статистически значимых запросов к БД - Создание программного кода в соответствии с техническим заданием (готовыми спецификациями) - Оптимизация программного кода с использованием специализированных программных средств - Определение источников больших данных для анализа, идентификация внешних и внутренних источников данных для проведения аналитических работ - Получение и фильтрация больших объемов данных из гетерогенных источников - Извлечение, проверка и очистка больших объемов данных из гетерогенных источников - Агрегация и разработка представления больших объемов данных из гетерогенных источников - Оценка соответствия набора данных предметной области и задачам аналитических работ - Возможности имеющейся у исполнителя методологической и технологической инфраструктуры анализа больших данных - Предметная область анализа - Теоретические и прикладные основы анализа больших данных - Современные методы и инструментальные средства анализа больших данных - Современный опыт использования анализа больших данных - Осуществление стратегического и оперативного руководства кадровыми и материальными ресурсами в процессах получения, размещения, хранения, распределения, миграции, архивирования и удаления данных - Осуществление стратегического и оперативного руководства материально-техническими ресурсами и элементами технологической инфраструктуры больших данных - Разработка моделей данных, адаптированных к технологиям больших данных - Интеграция больших данных с системами хранения данных организации - Создание и интеграция облачных репозиториев с системами хранения данных организации - Создание параллельных систем	8. Создание структуры реляционной базы данных средствами SQL, настройка ограничений целостности, индексов и связей. 9. Разработка SQL-запросов, представлений и программных объектов базы данных; оптимизация операций обработки данных. 10. Настройка механизмов защиты, разграничения доступа, аудита, резервирования и безопасной эксплуатации базы данных. 11. Проектирование и создание объектов NoSQL-базы данных, выполнение CRUD-операций, агрегации и индексирования. 12. Создание коллекций и индексов векторной базы данных и настройка взаимодействия с приложением для обработки векторов. 13. Получение данных из разнородных источников, разработка процедур извлечения, преобразования и загрузки данных. 14. Очистка, нормализация, преобразование данных и формирование датасетов для последующего использования в ИИ-системах. 15. Разработка и настройка DAG/задач конвейера, зависимостей, расписаний и процедур оркестрации обработки данных. 16. Настройка распределенной и потоковой обработки данных, контейнеризация компонентов конвейера и проверка их взаимодействия. 17. Разработка правил качества и автоматической валидации данных, выявление аномалий, Data Drift и Concept Drift. 18. Мониторинг KPI и SLA конвейеров, настройка логирования, оповещений и визуализации показателей качества данных.	

хранения и обработки информации - Разработка методов и регламентов реализации различных режимов обработки больших данных, в том числе режима реального времени - Мониторинг и оценка производительности обработки данных в организации, разработка предложений по повышению производительности обработки больших данных - Разработка методов и регламентов преобразования данных - Разработка и согласование стратегии формирования информационных активов организации - Разработка методов и регламентов оценки информационных активов организации - Формирование состава и координация деятельности персонала, отвечающего за приобретение и получение больших данных - Оценка эффективности системы хранения и обработки данных организации - Разработка предложений по развитию и совершенствованию системы получения, хранения, передачи, обработки больших данных - Формирование целей, приоритетов и ограничений управления качеством данных, адаптация целей к изменениям внутренних и внешних условий, потребностей и возможностей, в том числе технологических - Разработка стратегии управления качеством больших данных - Выделение ресурсов для управления качеством больших данных - Реализация стратегии обеспечения качества больших данных - Анализ реализации стратегии обеспечения качества больших данных - Проведение оценки качества больших данных - Разрешение проблем и инцидентов с качеством больших данных - Оценка рисков использования больших данных		

5. Контрольно-оценочные материалы для квалификационного экзамена

СВОДНАЯ ВЕДОМОСТЬ ПО ПРОФЕССИОНАЛЬНОМУ МОДУЛЮ
ПМ.02 Администрирование баз данных

Ф.И.О.«_____»
студент на 3 курсе по специальности СПО 09.02.13 «Интеграция решений с применением технологий искусственного интеллекта»
освоил программу профессионального модуля **ПМ.02** Администрирование баз данных
в объёме 572 час. с «____»_____20__г. по «___»_____ 20__г.

Результаты промежуточной аттестации по элементам профессионального модуля

Элементы модуля	**Формы промежуточной аттестации**	**Оценка**
МДК 02.01 Управление и автоматизация баз данных	Дифференцированный зачет	
МДК 02.02 Технология разработки и защиты баз данных	Экзамен	
МДК 02.03 Конвейеры обработки и контроль качества данных для систем ИИ	Дифференцированный зачет	
УП.02	Зачет	
ПП.02	Зачет	
ПМ 02	**Экзамен по модулю**	

Итоги экзамена по профессиональному модулю

Коды проверяемых компетенций	**Показатели оценки результата**	**Оценка (освоен / не освоен)**
ОК 01	Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам	освоен
ОК 02	Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности	освоен
ОК 05	Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста	освоен
ОК 09	Пользоваться профессиональной документацией на государственном и иностранном языках	освоен
ПК 2.1	Выявлять проблемы, возникающие в процессе эксплуатации баз данных	освоен
ПК 2.2	Осуществлять процедуры администрирования баз данных	освоен
ПК 2.3	Проводить аудит систем безопасности баз данных с использованием регламентов по защите информации	освоен
ПК 2.4	Формировать требования хранилищ банка данных для обучения	освоен
ПК 2.5	Подготавливать данные для базы знаний	освоен
ПК 2.6	Осуществлять управление базами данных	освоен
ПК 2.7	Осуществлять автоматизацию баз данных	освоен
ПК 2.8	Разрабатывать объекты баз данных NoSQL	освоен
ПК 2.9	Осуществлять конвейерную обработку данных для систем ИИ	освоен

ПК 2.10	Осуществлять мониторинг и оркестрацию конвейеров обработки данных	освоен
ПК 2.11	Осуществлять контроль качества данных в конвейерах обработки	освоен
Члены аттестационной комиссии _____ _____		

Лист согласования

Дополнения и изменения к комплекту ФОС по ПМ.__
на_____ учебный год

_____

Дополнения и изменения к комплекту ФОС на _____ учебный год по профессиональному модулю _____

В комплект ФОС внесены следующие изменения:

_____

_____

_____

_____

_____

Дополнения и изменения в комплекте ФОС обсуждены на заседании ЦК _____Протокол № _____ от «____» _____ 20____г.

Председатель ЦК _____