

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Магнитогорский государственный технический университет им. Г.И. Носова»

Многопрофильный колледж

**МЕТОДИЧЕСКИЕ УКАЗАНИЯ
ДЛЯ ЛАБОРАТОРНЫХ ЗАНЯТИЙ
ПРОФЕССИОНАЛЬНОГО МОДУЛЯ**

ПМ.01 Разработка, администрирование и защита баз данных

для обучающихся специальности

09.02.11 Разработка и управление программным обеспечением

СОДЕРЖАНИЕ

1 ВВЕДЕНИЕ	3
2 МЕТОДИЧЕСКИЕ УКАЗАНИЯ	5
Лабораторное занятие №1	5
Лабораторное занятие №2	6
Лабораторное занятие №3	9
Лабораторное занятие №4	11
Лабораторное занятие №5	13
Лабораторное занятие №6	15
Лабораторное занятие №7	18
Лабораторное занятие №8	20
Лабораторное занятие №9	24
Лабораторное занятие №10	25
Лабораторное занятие №11	26
Лабораторное занятие №12	29
Лабораторное занятие №13	30
Лабораторное занятие №14	32
Лабораторное занятие №15	35
Лабораторное занятие №16	37
Лабораторное занятие №17	40
Лабораторное занятие №18	42
Лабораторное занятие №19	44
МДК.01.02. Управление базами данных	2
Лабораторное занятие №20	2
Лабораторное занятие №21	4
Лабораторное занятие №22	6
Лабораторное занятие №23	9
Лабораторное занятие №24	11
Лабораторное занятие №25	13
Лабораторное занятие №26	15
Лабораторное занятие №27	17
Лабораторное занятие №28	20
МДК.01.03 Конфигурирование и программирование в системе 1С	22
Лабораторное занятие №29	22
Лабораторное занятие №30	24
Лабораторное занятие №31	26
Лабораторное занятие №32	28
Лабораторное занятие №33	32
Лабораторное занятие №34	33

1 ВВЕДЕНИЕ

Важную часть теоретической и профессиональной практической подготовки обучающихся составляют практические занятия.

Состав и содержание практических занятий направлены на реализацию Федерального государственного образовательного стандарта среднего профессионального образования.

Ведущей дидактической целью практических занятий является формирование учебных практических умений (использовать современные case – средства для проектирования баз данных; создавать объекты баз данных в современных СУБД; применять стандартные методы для защиты объектов базы данных), необходимых в последующей учебной деятельности.

В соответствии с рабочей программой профессионального модуля «Разработка, администрирование и защита баз данных» предусмотрено проведение практических занятий.

В результате их выполнения, обучающийся должен уметь:

У 1.1.1 анализировать предметную область и выделять основные сущности;

У 1.1.2 разрабатывать концептуальную, логическую и физическую модели баз данных;

У 1.1.3 определять типы данных для полей таблиц;

У 1.2.1 создавать таблицы, индексы, ограничения и другие объекты базы данных

У 1.2.2 оптимизировать запросы к базе данных для повышения производительности

У 1.2.3 разрабатывать хранимые процедуры и триггеры для баз данных;

У 1.3.1 разрабатывать объекты базы данных, такие как таблицы, индексы и связи между ними;

У 1.3.2 программировать и создавать хранимые процедуры, функции и триггеры для обработки данных;

У 1.3.3 управлять данными в базе данных, включая ввод, обновление и удаление данных;

У 1.4.1 создавать пользователей и назначать права доступа;

У 1.4.2 управлять транзакциями и контролировать целостность данных;

У 1.4.3 создавать и восстанавливать резервные копии данных;

У 1.5.1 использовать механизмы аудита для отслеживания доступа к данным;

У 1.5.2 создавать и управлять ролями и правами доступа к данным;

У 1.5.3 устанавливать и настраивать механизмы аутентификации и авторизации пользователей;

Уо 02.01 определять необходимые источники информации; эффективно искать информацию, необходимую для решения задачи и/или проблемы, оформлять результаты поиска;

Уо 02.02 использовать современное программное обеспечение, различные цифровые средства для решения профессиональных задач; проявлять культуру информационной безопасности при использовании информационно-коммуникационных технологий;

Уо 04.01 организовывать работу коллектива и команды;

Уо 04.02 эффективно работать в команде; взаимодействовать с коллегами, руководством, клиентами в ходе профессиональной деятельности;

Уо 04.03 использовать навыки управления проектами в распределении ресурсов и формировании графика выполнения задач;

Содержание практических занятий ориентировано на подготовку обучающихся к освоению профессионального модуля программы подготовки специалистов среднего звена по специальности и овладению **профессиональными компетенциями:**

ПК 1.1. Проектировать базы данных

ПК 1.2. Разрабатывать объекты баз данных в соответствии с результатами анализа предметной области

ПК 1.3. Реализовывать базу данных в конкретной системе управления базами данных

ПК 1.4. Администрировать базы данных

ПК 1.5. Защищать информацию в базе данных с использованием технологии защиты информации

А также формированию *общих компетенций*:

ОК 02. Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности.

ОК 04. Эффективно взаимодействовать и работать в коллективе и команде.

Практические занятия проводятся в рамках соответствующей темы, после освоения дидактических единиц, которые обеспечивают наличие знаний, необходимых для ее выполнения.

2 МЕТОДИЧЕСКИЕ УКАЗАНИЯ

МДК.01.01. Проектирование и разработка баз данных Тема 1.1 Основы хранения и обработки данных.

Лабораторное занятие №1.

Сбор и анализ информации

Цель: Формирование умений собирать и анализировать информацию, выполнять анализ предметной области.

Выполнив работу, вы будете уметь:

У 1.1.1 анализировать предметную область и выделять основные сущности;

Уо 02.01 определять необходимые источники информации; эффективно искать информацию, необходимую для решения задачи и/или проблемы, оформлять результаты поиска

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: Postgre SQL.

Проектирование любой базы данных начинается не с написания кода, а с глубокого анализа предметной области. На этом этапе необходимо понять, какие данные будут храниться в системе, как они связаны между собой и какие задачи должна решать будущая информационная система. Результатом этого этапа является набор требований, который ляжет в основу концептуальной модели (например, ER-диаграммы).

В рамках данной лабораторной работы вы выступите в роли аналитика. Вам предстоит самостоятельно выбрать предметную область, собрать о ней информацию и формализовать её в виде структурированного описания.

2. Задание к лабораторной работе

Выберите одну из предложенных предметных областей или предложите свою (согласовав её с преподавателем):

- Система автоматизации для небольшой библиотеки.
- База данных для учёта успеваемости студентов в учебной части.
- Система для управления заказами в интернет-магазине.
- База данных для учёта оборудования и заявок в IT-отделе (Help Desk).
- Система для ведения кадрового учёта в небольшой компании.

На основе выбранного варианта выполните следующие действия:

1. Сбор информации. Определите основных пользователей будущей системы и их роли. Выявите основные бизнес-процессы (например, «регистрация нового студента», «оформление заказа», «выдача книги»).

2. Анализ и формализация. На основе собранной информации выделите: Основные сущности (например, «Студент», «Дисциплина», «Заказ», «Товар»). Атрибуты каждой сущности (например, для сущности «Студент»: ФИО, номер студенческого билета, группа).

Связи между сущностями (например, «Студент» сдает «Экзамен» по «Дисциплине»). Документирование. Представьте результаты анализа в виде структурированного документа, описывающего все сущности, их атрибуты и связи.

Порядок выполнения работы

1. Выбор и описание предметной области

Выберите тему из предложенного списка или предложите свою.

Кратко опишите предметную область (2-3 предложения). Укажите цель создания информационной системы для этой области.

2. Определение ролей пользователей и бизнес-процессов

Определите, кто будет пользоваться системой. Опишите роли (например, Администратор, Оператор, Конечный пользователь).

Для каждой роли опишите основные функции, которые она будет выполнять в системе.

Составьте список из 5-7 ключевых бизнес-процессов, которые должна автоматизировать система. Опишите каждый процесс в формате: «Сущность 1» выполняет «Действие» над «Сущностью 2» (например, «Клиент» оформляет «Заказ»).

3. Выделение сущностей и их атрибутов

На основе анализа бизнес-процессов выделите 5-8 основных сущностей предметной области.

Для каждой сущности составьте таблицу со списком её атрибутов (полей). Обязательно выделите среди них:

Первичный ключ (Primary Key): атрибут или группа атрибутов, однозначно идентифицирующих каждую запись в сущности.

Обязательные атрибуты: поля, которые должны быть заполнены всегда.

Форма представления результата:

Оформленная схема базы данных в тетради.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 1.1 Основы хранения и обработки данных.

Лабораторное занятие №2. Проектирование реляционной схемы базы данных в среде СУБД.

Цель: получение практических навыков проектирования базы данных

Выполнив работу, вы будете уметь:

У 1.1.1 анализировать предметную область и выделять основные сущности;

Уо 02.01 определять необходимые источники информации; эффективно искать информацию, необходимую для решения задачи и/или проблемы, оформлять результаты поиска.

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: Postgre SQL.

Задание:

В соответствии со своим вариантом проанализируйте предметную область решаемой задачи и разработайте логическую структуру соответствующей базы данных

Краткие теоретические сведения:

Проектирование базы данных заключается в ее многоступенчатом описании с различной степенью детализации и формализации, в ходе которого производится уточнение и оптимизация структуры базы данных. Проектирование начинается с описания предметной области и задач информационной системы, идет к более абстрактному уровню логического описания данных и далее – к схеме физической (внутренней) модели базы данных. Трем основным уровням моделирования системы – **концептуальному, логическому и физическому** соответствуют три последовательных этапа детализации описания объектов базы данных и их взаимосвязей.

На **концептуальном уровне** проектирования производится смысловое описание информации предметной области, определяются ее границы, производится абстрагирование от несущественных деталей. В результате определяются моделируемые объекты, и их свойства, и связи. Выполняется структуризация знаний о предметной области, стандартизируется терминология. Затем строится концептуальная модель, описываемая на естественном языке. Для описания свойств и связей объектов применяют различные диаграммы.

На следующем шаге принимается решение о том, в какой конкретно СУБД будет реализована база данных. **Выбор СУБД** является сложной задачей и должен основываться на потребностях с точки зрения информационной системы и пользователей. Определяющими здесь являются вид программного продукта и категория пользователей (или профессиональные программисты, или конечные пользователи, или то и другое).

Другими показателями, влияющими на выбор СУБД, являются:

- Удобство и простота использования;
- Качество средств разработки, защиты и контроля базы данных;
- Уровень коммуникационных средств (в случае применения ее в сетях);
- Фирма-разработчик;
- Стоимость.

Каждая конкретная СУБД работает с определенной моделью данных. Под моделью данных понимается способ их взаимосвязи: в виде иерархического дерева, сложной сетевой структуры или связанных таблиц. В настоящее время большинство СУБД использует табличную модель данных, называемую *реляционной*.

На **логическом уровне** производится отображение данных концептуальной модели в логическую модель в рамках той структуры данных, которая поддерживается выбранной СУБД. Логическая модель не зависит от конкретной СУБД и может быть реализована на любой СУБД реляционного типа.

На **физическом уровне** производится выбор рациональной структуры хранения данных и методов доступа к ним, которые обеспечивает выбранная СУБД. На этом уровне решаются вопросы эффективного выполнения запросов к БД, для чего строятся дополнительные структуры, например индексы. В физической модели содержится информация обо всех объектах базы данных (таблицах, индексах, процедурах и др.) и используемых типах данных. Физическая модель *зависит* от конкретной СУБД. Одной и той же логической модели может соответствовать несколько разных физических моделей. Физическое проектирование является начальным этапом реализации базы данных.

Варианты заданий:

1. *БД «Зарплата».* Таблица «Должности»: код должности; должность. Таблица «Тарифы»: код должности; разряд; ставка, р./ч. Таблица «Работники»: отдел; код должности; разряд; табельный номер; Ф.И.О. Таблица «Табель»: год; месяц; табельный номер; количество отработанных часов; дата начисления зарплаты
2. *БД «Оптовая база».* Таблица «Товары»: код товара; товар; единица измерения; цена. Таблица «Склад»: дата поступления, код товара; количество. Таблица «Заявки»: номер заявки; организация; код товара; требуемое количество товара. Таблица «Отпуск товаров»: номер заявки; код товара; дата отпуска товара; количество отпущенного товара
3. *БД «Библиотека».* Таблица «Книги»: жанр; шифр книги; автор; название; год издания; количество экземпляров. Таблица «Читатели»: номер читательского билета; Ф.И.О.; адрес. Таблица «Выдачи»: дата выдачи; номер читательского билета; шифр книги; количество экземпляров; срок возврата; фактическая дата возврата
4. *БД «ГИБДД».* Таблица «Автомобили»: модель автомобиля; номер двигателя; номер кузова; серия и номер технического паспорта; государственный номер автомобиля. Таблица «Владельцы»: государственный номер автомобиля; Ф.И.О.; адрес, серия и номер водительского удостоверения. Таблица «Виды нарушений»: код нарушения; вид нарушения. Таблица «Нарушители»: дата нарушения; код нарушения; государственный номер автомобиля; размер штрафа, р.
5. *БД «Соревнования».* Таблица «Команда»: спортивная организация; шифр команды, название команды. Таблица «Участники»: шифр команды; номер участника; Ф.И.О. Таблица «Старт»: стартовый номер; номер участника; время старта; отметка о невыходе на старт. Таблица «Финиш»: порядковый номер финиширования; стартовый номер; время финиша; отметка о сходе с дистанции
6. *БД «Перевозки».* Таблица «Транспорт»: модель автомобиля; государственный номер автомобиля; удельный расход топлива, л/100 км. Таблица «Заявки»: номер заявки;
7. дата; пункт отправления; пункт назначения; груз; единица измерения; количество груза. Таблица «Доставка»: номер заявки; государственный номер автомобиля; дата отправления; дата возвращения; пройденное расстояние; расход топлива
8. *БД «Сессия».* Таблица «Кафедры и дисциплины»: номер кафедры; кафедра; шифр дисциплины; наименование дисциплины; вид аттестации (зачет или экзамен). Таблица «Преподаватели»: номер кафедры; табельный номер преподавателя; Ф.И.О. Таблица «Студенты»: шифр студента; Ф.И.О. Таблица «Оценки»: дата; шифр дисциплины; табельный номер преподавателя; шифр студента; оценка
9. *БД «Телефонная компания».* Таблица «Тарифы»: код тарифа; вид тарифа; цена, р./мин. Таблица «Виды льгот»: код льготы; вид льготы; размер, %. Таблица «Абоненты»: лицевой счет; телефон; Ф.И.О.; адрес; код льготы. Таблица «Платежи»: лицевой счет; код тарифа; дата оплаты; сумма платежа; дата отключения за неуплату

Порядок выполнения работы:

1. Выполнить анализ предметной области.
2. Выполнить анализ данных.
3. Определить набор атрибутов для данной предметной области.
4. Определить набор таблиц.

При проектировании таблиц рекомендуется руководствоваться следующими основными принципами:

- каждая таблица должна содержать данные только на одну тему;
 - данные не должны дублироваться.
5. Создать словарь имен.
 6. Определить состав и типы полей.
 7. Создать связи между таблицами.

Форма представления результата:

Оформленная схема базы данных в тетради.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 1.2. Проектирование базы данных**Лабораторное занятие №3****Приведение базы данных к нормальной форме 3НФ**

Цель: получение практических навыков по нормализации базы данных

Выполнив работу, вы будете уметь:

У 1.1.2 разрабатывать концептуальную, логическую и физическую модели баз данных;

Уо 02.01 определять необходимые источники информации; эффективно искать информацию, необходимую для решения задачи и/или проблемы, оформлять результаты поиска.

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение.

Задание:

В соответствии со своим вариантом задания, выданным на предыдущем занятии привести базу данных к 3НФ. Реализовать схему базы данных в среде PostgreSQL.

Краткие теоретические сведения:**Нормализация баз данных**

Нормальные формы – это рекомендации по проектированию баз данных. Рекомендуется нормализовать базу данных в некоторой степени потому, что этот процесс имеет ряд существенных преимуществ с точки зрения эффективности и удобства обращения с базой данных.

- ☐ ☐ В нормализованной структуре базы данных можно производить сложные выборки данных относительно простыми SQL-запросами.
- ☐ ☐ Целостность данных. Нормализованная база данных позволяет надежно хранить данные.
- ☐ ☐ Нормализация предотвращает появление избыточности хранимых данных. Данные всегда хранятся только в одном месте, что делает легким процесс вставки, обновления и удаления данных. Есть исключение из этого правила. Ключи, сами по себе, хранятся в нескольких местах потому, что они копируются как внешние ключи в другие таблицы.
- ☐ ☐ Масштабируемость – это возможность системы справляться с будущим ростом. Для базы данных это значит, что она должна быть способна работать быстро, когда число пользователей и объемы данных возрастают. Масштабируемость – это очень важная

характеристика любой модели базы данных и для РСУБД.

Нормализация баз данных заключается в приведении структуры хранения данных к нормальным формам (NF). Всего таких форм существует 8, но часто достаточным является соблюдение первых трех. Рассмотрим их более подробно на примере учебной базы данных. Примеры будут строиться по принципу «что было бы, если было иначе, чем сейчас».

Первая нормальная форма

Основным правилом первой формы является необходимость неделимости значения в каждом поле (столбце) строки – **атомарность** значений.

Рассмотрим таблицы сотрудников и телефонных линий.

Чтобы избавиться от связывающей таблицы «Сотрудники_Линии», мы могли бы записать идентификаторы сотрудников для каждой линии в виде перечня в дополнительном столбце:

Но подобная структура не является надежной. Представьте, что Вам необходимо поменять некоторым сотрудникам подключенные линии. Потребуется осуществить разбор составного поля, чтобы определить наличие id сотрудника в каждой записи линий, затем скорректировать перечень. Получается слишком сложный и долгий процесс для такой простой операции.

Организации структуры таблиц с применением дополнительной связывающей избавляет от подобных проблем.

Помимо атомарности к первой нормальной форме относятся следующие правила:

- Строки таблиц не должны зависеть друг от друга, т.е. первая запись не должна влиять на вторую и наоборот, вторая на третью и т.д. Размещение записей в таблице не имеет никакого значения.
- Аналогичная ситуация со столбцами записей. Их порядок не должен влиять на понимание информации.
- Каждая строка должна быть уникальна, поэтому для нее определяется **первичный ключ**, состоящий из одного либо нескольких полей (**составной ключ**). Первичный ключ не может повторяться в пределах таблицы и служит идентификатором записи.

Вторая нормальная форма

Условием этой формы является отсутствие зависимости неключевых полей от части составного ключа.

Так как составной ключ в учебной базе наблюдается только в таблице «Сотрудники_Линии», то рассмотрим пример на ней.

В нашем примере столбцы описания и приоритета зависят от столбца «Линия», входящего в составной ключ. Это значит, что для каждой линии, подключенной разным сотрудникам, потребуется повторно указывать описание и приоритетность. Подобная структура приводит к **избыточности данных**.

Также велика вероятность возникновения противоречивой информации. Изменяя приоритет или описание для линии, можно по ошибке оставить некоторые строки не обработанными. В таком случае, для одного и того же идентификатора линии значения зависимых полей будут различными.

Если соблюдены правила первой нормальной формы, то создание таблицы «Линии» и перенос в нее зависимых столбцов удовлетворяет второй нормальной форме.

Все риски, которые были рассмотрены для 2NF, так же относятся к 3NF и устраняются переносом зависимых полей в отдельную таблицу.

Третья нормальная форма

3NF схожа по логике с 2NF, но с некоторым отличием. Если 2 форма ликвидирует зависимости неключевых полей от части ключа, то третья нормальная форма исключает зависимость неключевых полей от других неключевых полей.

На приведенном примере таблицы сотрудников видно, что столбец «Супервайзер»

имеет зависимость от столбца «Группа», а это значит, что при изменении значения поля группы, потребуется изменить значение поля супервайзера.

Денормализация базы данных

Теория нормальных форм не всегда применима на практике. Например, неатомарные значения не всегда являются «злом», а иногда наоборот. Связано это с необходимостью дополнительного объединения (следовательно, затрат производительности системы) при выполнении запросов, особенно когда производится обработка большого массива информации.

Порядок выполнения работы:

Используя метод нормальных форм спроектировать базу данных. Процесс проектирования должен быть представлен в форме отчета, показан процесс формирования таблиц под выделенные сущности и их последовательная нормализация методом нормальных форм.

Форма представления результата:

представление отчета на образовательном портале (в соответствующем курсе).

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 1.2. Проектирование базы данных

Лабораторное занятие №4

Установка и настройка PostgreSQL

Цель: Формирование умений устанавливать и настраивать систему управления баз данных PostgreSQL

Выполнив работу, вы будете уметь:

У 1.3.1 разрабатывать объекты базы данных, такие как таблицы, индексы и связи между ними;

Уо 02.01 определять необходимые источники информации; эффективно искать информацию, необходимую для решения задачи и/или проблемы, оформлять результаты поиска.

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: PostgreSQL.

Задание:

PostgreSQL — это мощная, кроссплатформенная объектно-реляционная система управления базами данных (ОРСУБД) с открытым исходным кодом. Она отличается высокой надёжностью, соответствием стандартам SQL и расширяемостью. В рамках данной работы вы установите PostgreSQL, создадите своего первого пользователя и базу данных, а также настроите основные параметры для обеспечения базовой безопасности и производительности.

2. Задание к лабораторной работе

1. Установить СУБД PostgreSQL последней стабильной версии на локальную машину (или в виртуальную среду).
2. Выполнить базовую конфигурацию сервера, изменив стандартные параметры для повышения безопасности.
3. Создать нового пользователя (роль) с паролем и ограниченными правами.
4. Создать новую базу данных, владельцем которой будет созданный пользователь.
5. Проверить возможность подключения к новой базе данных от имени нового пользователя.
6. Настроить `pg_hba.conf` для переключения метода аутентификации с `peer/trust` на `md5`.

Порядок выполнения работы

Примечание: Все команды, которые необходимо вводить в терминале (командной строке), выделены моноширинным шрифтом. Команды, вводимые в оболочке `psql`, начинаются с символа `=>`.

Шаг 1. Установка PostgreSQL

Для дистрибутивов на базе Debian/Ubuntu (Linux):

- 1.1. Обновите список пакетов:

```
sudo apt update
```

- 1.2. Установите PostgreSQL и дополнительные утилиты:

```
sudo apt install postgresql postgresql-contrib
```

Для дистрибутивов на базе RedHat/CentOS (Linux):

- 1.1. Установите репозиторий PostgreSQL:

```
sudo yum install -y https://download.postgresql.org/pub/repos/yum/repos/EL-8-x86_64/pgdg-redhat-repo-latest.noarch.rpm
```

- 1.2. Установите сервер и клиент:

```
sudo yum install postgresql13-server postgresql13
```

Для Windows:

- 1.1. Скачайте установщик с официального сайта: <https://www.postgresql.org/download/windows/>

- 1.2. Запустите установку, следуя инструкциям мастера. Обязательно запомните пароль суперпользователя `postgres`, который вы зададите при установке.

Шаг 2. Проверка установки и базовая структура

- 2.1. Переключитесь на системного пользователя `postgres`, под которым по умолчанию работает СУБД:

```
sudo -i -u postgres
```

- 2.2. Войдите в интерактивную оболочку PostgreSQL:

```
psql
```

- 2.3. Выведите список баз данных, чтобы убедиться, что сервер работает:

```
=> \l
```

- 2.4. Выведите список пользователей (ролей):

```
=> \du
```

2.5. Выйдите из оболочки psql:

=> \q

2.6. Выйдите из пользователя postgres (в Linux):

exit

Шаг 3. Создание нового пользователя и базы данных

3.1. Создайте нового пользователя (роль) с паролем и правом на создание баз данных:

```
sudo -u postgres createuser -P --createdb my_user
```

Система запросит пароль для нового пользователя.

3.2. Создайте новую базу данных, владельцем которой будет my_user:

```
sudo -u postgres createdb -O my_user my_database
```

Шаг 4. Подключение к базе данных от имени нового пользователя

4.1. Попробуйте подключиться к созданной базе данных, используя нового пользователя:

```
psql -U my_user -d my_database -h 127.0.0.1 -W
```

Флаг -W заставит psql запросить пароль.

4.2. Введите пароль, созданный на шаге 3.1.

4.3. Убедитесь, что вы подключились к нужной базе данных:

```
=> \conninfo
```

Форма представления результата:

Отчет по выполненной лабораторной работе.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 1.3 Разработка и администрирование базы данных

Лабораторное занятие №5

Создание базы данных в среде разработки.

Цель: получение практических навыков по освоению операций создания таблиц

Выполнив работу, вы будете уметь:

У 1.3.1 разрабатывать объекты базы данных, такие как таблицы, индексы и связи между ними;

Уо 02.02 использовать современное программное обеспечение, различные цифровые средства для решения профессиональных задач; проявлять культуру информационной безопасности при использовании информационно-коммуникационных технологий.

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: Postgre SQL.

Задание 1:

Создать базу данных Туризм и таблицы базы данных в Postgre SQL.

Порядок выполнения работы:

1. Создайте в своей рабочей папке папку с именем *База данных*.
2. Запустите Postgre SQL. Создайте новую базу данных с именем *Туризм*.
3. Создайте таблицу **Сотрудники**. Наименования и типы полей представлены в приведенной таблице.

Название поля	Тип данных
---------------	------------

<u>Код сотрудника</u>	Числовой
-----------------------	----------

ФИО	Текст
-----	-------

<u>Должность</u>	Текст
------------------	-------

Дата найма	Дата/Время
------------	------------

<u>Дата рождения</u>	Дата/Время
----------------------	------------

Домашний телефон	Текст
------------------	-------

<u>Адрес</u>	Текст
--------------	-------

Размер оклада	Числовой
---------------	----------

4. Для поля **Оклад** задайте условие, что он больше 5000 р., но не больше 10000.
5. Создайте первичный ключ.
6. Создайте новые таблицы **Клиенты** и **Страны**. В качестве первичного ключа задайте **Код Клиента** и **Код Тура**.
7. Создайте таблицу **Договоры**, которая должна иметь следующие поля:

Название поля	Тип данных
---------------	------------

Номер договора	Число
----------------	-------

<u>Код клиента</u>	Числовой
--------------------	----------

Код тура	Число
----------	-------

Дата начала тура	Дата/Время
------------------	------------

Дата окончания тура	Дата/Время
---------------------	------------

Число туристов	Числовой
----------------	----------

Цена тура	Денежный
-----------	----------

Дата платежа	Дата/Время
--------------	------------

Код Сотрудника	Числовой
----------------	----------

8. Введите в таблицы 10 разнообразных записей. В таблице **Сотрудники** осуществите ввод заведомо некорректных данных для проверки работоспособности условия на значение.

Форма представления результата:

Отчет по выполненной лабораторной работе.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 1.3 Разработка и администрирование базы данных

Лабораторное занятие №6

Организация SQL запросов на выборку данных.

Цель: Формирование умений создавать запросы на выборку данных из таблиц в pgadmin4

Выполнив работу, вы будете уметь:

У 1.2.2 оптимизировать запросы к базе данных для повышения производительности

Уо 02.02 использовать современное программное обеспечение, различные цифровые средства для решения профессиональных задач; проявлять культуру информационной безопасности при использовании информационно-коммуникационных технологий.

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: Postgre SQL.

Задание 1:

По своему варианту базы данных в Postgre SQL для каждой таблицы создайте первичные и внешние ключи, установите связи и укажите индексируемые поля.

Краткие теоретические сведения:

Индексы обеспечивают поиск и сортировку записей в Workbench. В индексе хранится местоположение записей на основе одного или нескольких полей, которые были выбраны для индексирования. После того как Workbench получает сведения о позиции в индексе, он может извлечь данные путем перемещения непосредственно к нужной позиции. Благодаря этому использование индекса гораздо эффективнее просмотра всех записей для поиска необходимых данных.

Выбор полей для индексирования

Вы можете создавать индексы, основанные на одном или нескольких полях. В основном требуется индексировать поля, в которых часто осуществляется поиск, сортируемые поля и поля, объединенные с полями в других таблицах, что часто используется в запросах по нескольким таблицам. Индексы ускоряют поиск и выполнение запросов, однако они могут привести к снижению производительности при добавлении или обновлении данных. Каждый раз, когда вы добавляете или изменяете запись в таблице, содержащей один или несколько индексов, Workbench приходится обновлять индексы. Добавление записей с помощью запроса на добавление или с помощью импортирования записей также, скорее всего, будет происходить медленнее, если таблица-получатель содержит индексы.

Примечание: Первичный ключ таблицы индексируется автоматически.

Составные индексы

Если предполагается, что необходимо будет часто выполнять поиск или сортировку по нескольким полям, вы можете создать индекс для этого сочетания полей. Например, если в одном запросе часто задаются условия для полей "Поставщик" и "Наименование_продукта", имеет смысл создать для этих полей составной индекс.

При сортировке таблицы по составному индексу Workbench сначала выполняет сортировку по первому полю, заданному для индекса. Последовательность полей определяется при создании составного индекса. Если в первом поле содержатся записи с повторяющимися значениями, затем выполняется сортировка по второму полю, заданному для индекса, и т. д.

В составной индекс можно включить до 10 полей.

Создание индекса

Перед созданием индекса необходимо решить, следует ли создать индекс для одного поля или составной индекс. Индекс для одного поля создается с помощью установки свойства

При создании уникального индекса невозможно ввести новое значение в определенном поле, если такое значение уже существует в том же поле другой записи. Workbench автоматически создает уникальный индекс для первичных ключей, однако может потребоваться запретить создание повторяющихся значений и в других полях. Например, вы можете создать уникальный индекс для поля, в котором содержатся серийные номера, чтобы двум продуктам нельзя было присвоить один и тот же серийный номер.

Удаление индекса

Если индекс становится ненужным или приводит к значительному снижению производительности, его можно удалить. При этом удаляется только сам индекс, а не поля, на которых он основан.

Просмотр или редактирование индексов

Чтобы оценить влияние индексов на производительность или убедиться, что необходимые поля проиндексированы, просмотрите индексы в таблице.

Автоматическое создание индексов

В некоторых случаях индексы создаются автоматически. Например, индексы создаются для любых полей, которые определяются пользователем в качестве первичного ключа таблицы.

Так как каждый индекс требует дополнительной обработки, производительность при добавлении или обновлении данных снижается. Поэтому рекомендуется уменьшить их число, чтобы сократить количество создаваемых индексов.

Порядок выполнения работы:

1. Используя рассмотренный ниже пример, установить для своей базы данных первичные ключи.

2. Установить необходимы индексированные поля.

3. Установить связи между таблицами на схеме данных.

Пример.

Установка ключевых полей.

Отдельные таблицы, содержащие информацию по определенной теме, необходимо связать в единую структуру базы данных. Для связывания таблиц следует задать **ключевые поля**.

Ключ состоит из одного или нескольких полей, значения которых однозначно определяют каждую запись в таблице. Наиболее подходящим в качестве ключевого поля является *Счетчик*, так как значения в данном поле являются уникальными (т. е. исключают повторы).

2) в таблице *Клиенты* ключевое поле *Код клиента*

3) в таблице *Заказы* ключевое поле *Код заказа*

заказы	
Имя поля	Тип данных
Код заказа	Счетчик
Код клиента	Числовой
Код сотрудника	Числовой
Дата размещения	Дата/время
Дата исполнения	Дата/время
Сумма	Денежный
Отметка о выполнении	Логический

Создание связей между таблицами.

Существует несколько типов отношений между таблицами:

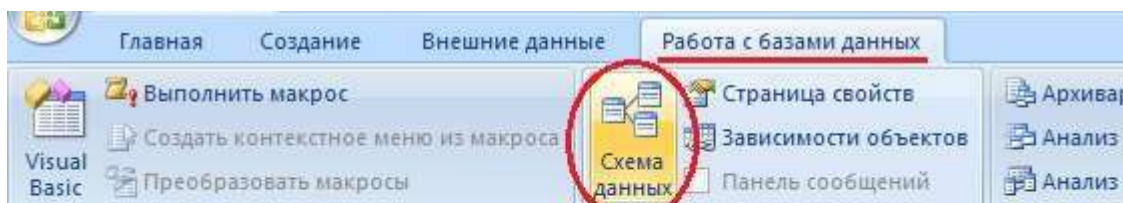
- при отношении «один-к-одному» каждой записи ключевого поля в первой таблице соответствует только одна запись в связанном поле другой таблицы, и наоборот. Отношения такого типа используются не очень часто. Иногда их можно использовать для разделения таблиц, содержащих много полей, для отделения части таблицы по соображениям безопасности;

- при отношении «один-к-многим» каждой записи в первой таблице соответствует несколько записей во второй, но запись во второй таблице не может иметь более одной связанной записи в первой таблице;

- при отношении «многие-к-многим» одной записи в первой таблице могут соответствовать несколько записей во второй таблице, а одной записи во второй таблице могут соответствовать несколько записей в первой.

Создайте связь между таблицами Клиенты и Заказы.

В результате должна получиться схема данных, представленная на рисунке.



На схеме данных связи отображаются в виде соединительных линий со специальными значками около таблиц. Связь «один-к-многим» помечается «1» вблизи главной таблицы (имеющей первичный ключ) и «∞» вблизи подчиненной таблицы (имеющей внешний ключ). Связь «один-к-одному» помечается двумя «1» (оба поля таблиц имеют первичные ключи). Неопределенная связь не имеет никаких знаков. Если установлено объединение, то его направление отмечается стрелкой на конце соединительной линии (ни одно из объединенных полей не является ключевым и не имеет уникального индекса).

Форма представления результата:

Отчет по выполненной лабораторной работе.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 1.3 Разработка и администрирование базы данных

Лабораторное занятие №7

Организация SQL запросов на выборку данных. Операция конкатенации. Использование строковых функций.

Цель: Формирование умений создавать запросы на выборку данных с использованием конкатенации и строковых функций.

Выполнив работу, вы будете уметь:

У 1.2.2 оптимизировать запросы к базе данных для повышения производительности

Уо 02.02 использовать современное программное обеспечение, различные цифровые средства для решения профессиональных задач; проявлять культуру информационной безопасности при использовании информационно-коммуникационных технологий.

Материальное обеспечение:

Методические указания для выполнения практических работ, компьютер, программное обеспечение: СУБД PostgreSQL

Оператор `SELECT` является основным инструментом для извлечения данных из таблиц. Помимо простого извлечения столбцов, SQL предоставляет мощные средства для преобразования данных "на лету" прямо в тексте запроса. Это позволяет:

* ****Форматировать вывод:**** объединять строки, изменять регистр, обрезать или дополнять текст.

* ****Создавать вычисляемые поля:**** формировать новые значения на основе существующих данных.

* ****Улучшать читаемость:**** давать столбцам в результате запроса понятные псевдонимы (алиасы).

В данной лабораторной работе вы научитесь использовать операцию конкатенации

(объединения строк) и основные строковые функции, которые поддерживаются большинством СУБД (PostgreSQL, PostgreSQL, Oracle, SQL Server).

2. Задание к лабораторной работе

В качестве рабочей базы данных используется учебная база данных магазина (например, `dvd-rental` для PostgreSQL или любая другая с таблицами `customer`, `film`, `actor`).

Вам необходимо составить и выполнить SQL-запросы для решения следующих задач:

1. ****Конкатенация:**** Сформировать для каждого клиента из таблицы `customer` полное имя в формате "Фамилия, Имя" и вывести его вместе с адресом электронной почты.

2. ****Строковые функции (форматирование):**** Вывести список всех фильмов из таблицы `film`. Для каждого фильма сформировать название в верхнем регистре и описание в нижнем регистре.

3. ****Строковые функции (обрезка и длина):**** Вывести список фильмов, название которых длиннее 15 символов. Для каждого такого фильма вывести первые 15 символов названия и полную длину названия.

4. ****Сложное форматирование:**** Вывести список актеров из таблицы `actor`. Для каждого актера сформировать строку вида: "Актер: [Фамилия], [Имя] | Длина полного имени: [N] символов".

Порядок выполнения работы

Шаг 1. Подготовка

1.1. Убедитесь, что СУБД PostgreSQL (или другая) запущена и вы можете подключиться к учебной базе данных.

1.2. Откройте инструмент для выполнения SQL-запросов (например, `psql`, pgAdmin, DBeaver).

Шаг 2. Выполнение задания 1: Конкатенация

2.1. Задача — объединить фамилию и имя клиента.

2.2. В PostgreSQL для конкатенации используется оператор `||`.

2.3. Используйте ключевое слово `AS` для присвоения столбцу с результатом понятного имени.

2.4. Примерный вид запроса:

```
```sql
SELECT last_name || ', ' || first_name AS "Полное имя", email
FROM customer;
```
```

***Примечание:** В других СУБД оператор конкатенации может отличаться (например, `+` в SQL Server или функция `CONCAT()`).

Шаг 3. Выполнение задания 2: Строковые функции

3.1. Задача — изменить регистр текста.

3.2. Используйте функции `UPPER()` для перевода в верхний регистр и `LOWER()` для нижнего.

3.3. Примерный вид запроса:

```
```sql
```

```
SELECT UPPER(title) AS "Название (ВЕРХНИЙ РЕГИСТР)", LOWER(description) AS
"Описание (нижний регистр)"
```

```
FROM film;
```

Шаг 4. Выполнение задания 3: Обрезка и длина строки

4.1. Задача — получить длину строки и её часть.

4.2. Используйте функцию `LENGTH()` для определения длины строки и `SUBSTRING()` для извлечения её части.

4.3. Примерный вид запроса:

```
```sql
```

```
SELECT title,
```

```
LENGTH(title) AS "Длина названия",
```

```
SUBSTRING(title FROM 1 FOR 15) AS "Первые 15 символов"
```

```
FROM film
```

```
WHERE LENGTH(title) > 15;
```

Форма представления результата:

Отчет по выполненной лабораторной работе.

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 1.3 Разработка и администрирование базы данных

Лабораторное занятие №8

Организация SQL запросов на выборку данных. Операции с данными.

Цель: Формирование умений создавать запросы на выборку данных, изменять и удалять записи в таблице

Выполнив работу, вы будете уметь:

У 1.2.2 оптимизировать запросы к базе данных для повышения производительности;

Уо 02.02 использовать современное программное обеспечение, различные цифровые средства для решения профессиональных задач; проявлять культуру информационной безопасности при использовании информационно-коммуникационных технологий.

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: Postgre SQL, MS Office

Задание 1:

В СУБД Postgre SQL заполнить базу данных (свой вариант) значениями.

Краткие теоретические сведения:

Для добавления одной или нескольких строк в таблицу используется команда:

```
INSERT [INTO] <Имя таблицы>  
[(<Список столбцов>)]
```

```
VALUES
```

```
(<Список значений 1>),  
(<Список значений 2>),  
...  
(<Список значений N>);
```

В команде INSERT используются следующие основные параметры.

- Имя таблицы, в которую добавляются строки.
- Список имен столбцов, для которых будут заданы значения. Если значения будут заданы для всех столбцов таблицы, то приводить список столбцов необязательно. Если столбец таблицы не включен в список, то в этом столбце при добавлении строки будет автоматически установлено значение по умолчанию.
- Значения, которые нужно добавить в таблицу. Значения могут указываться в одном из следующих форматов:
 - набор значений для каждой добавляемой строки заключается в скобки. Набор значений внутри каждой пары скобок должен соответствовать указанному списку столбцов, а если список столбцов не указан, то упорядоченному списку всех столбцов, составляющих таблицу (список столбцов таблицы можно просмотреть с помощью команды DESCRIBE. Значения внутри набора, а также сами наборы отделяются друг от друга запятыми;
 - символьные значения, а также значения даты и времени приводятся в одинарных кавычках. Для числовых значений кавычки необязательны. Десятичным разделителем для чисел с дробной частью служит точка. Время и даты вводятся, соответственно, в формате «YYYY-MM-DD» и «HH:MM:SS»;
 - чтобы ввести в столбец неопределенное значение, то необходимо указать вместо значения ключевое слово NULL *без кавычек* (слово в кавычках рассматривается как обычная символьная строка);
 - вместо значения можно указать ключевое слово DEFAULT *без кавычек*, тогда в столбец будет введено значение по умолчанию (если оно задано для этого столбца).

Например, добавьте в таблицу Product сведения о продукции компании.

```
INSERT INTO Product (name_pr, description, price, qty)
VALUES
```

```
(' Инфракрасный обогреватель', '3 режима нагрева: 400 Вт, 800 Вт, 1200 Вт', 1445.00, 4),
('Триль', 'Мощность 1440 Вт. Быстрый нагрев', 4115.00, 6),
```

```
('Кофеварка', 'Цвет: черный. Мощность: 450 Вт', 1710.00, 10),
('Чайник', 'Цвет: белый. Мощность: 2200 Вт. Объем: 2 л', 1725.00, 14),
('Утюг', 'Цвет: фиолетовый. Мощность: 1400 Вт', 5300.00, 16);
```

Эта команда добавляет значения в столбцы name_pr (наименование), description (описание), price (цена) и qty (количество) таблицы Product. При этом в столбец id (идентификатор) автоматически вносятся порядковые номера строк, поскольку этот столбец имеет тип данных SERIAL.

Порядок выполнения работы:

Используя команду INSERT вставить данные в таблицы базы данных.

Задание 2:

Таблицы базы данных Students заполнить начальными значениями.

Порядок выполнения работы:

Заполним таблицу **"Специальности"**. Для заполнения этой таблицы в обозревателе объектов щелкните правой кнопкой мыши по таблице **"Специальности"** и в появившемся меню выберите пункт **"Изменить первые 200 строк"**. В рабочей области "Microsoft SQL Server Management Studio" проявится окно заполнения таблиц. Заполните таблицу **"Специальности"**.

Так как поле **"Код специальности"** является первичным полем связи и ключевым числовым счетчиком, то оно заполняется автоматически (заполнять его не нужно).

Закройте окно заполнения таблицы **"Специальность"** щелкнув по кнопке закрытия окна в верхнем правом углу, над таблицей.

После заполнения таблицы **"Специальности"** заполним таблицу **"Предметы"**.

Откройте ее для заполнения как описано выше, и заполните.

Закройте окно заполнения таблицы **"Предметы"** и перейдите к заполнению таблицы

"Студенты". Откройте таблицу **"Студенты"** для заполнения и заполните ее.

Для заполнения дат в качестве разделителя можно использовать знак ".". Даты можно заполнять в формате "день.месяц.год".

Поле **"Код специальности"** является вторичным полем связи (для связи с таблицей **"Специальности"**). Следовательно, значения этого поля необходимо заполнять значениями поля **"Код специальности"** таблицы **"Специальности"**. В нашем случае это значения от 1

до 5. Если у Вас коды специальностей в таблице **"Специальности"** имеют другие значения, то внесите их в таблицу **"Студенты"**.

По окончании заполнения, закройте окно заполнения таблицы **"Студенты"**.

Наконец заполним таблицу **"Оценки"**.

Поля с датами заполняются, как и в таблице **"Студенты"**. Поля **"Код предмета 1"**, **"Код предмета 2"** и **"Код предмета 3"** являются внешними полями связи с таблицей **"Предметы"**. Поэтому они должны быть заполнены значениями поля **"Код предмета из этой таблицы"**, то есть значениями от 1 до 5.

Закройте окно заполнения таблицы **"Оценки"**.

Задание3:

Выполнить импорт данных в базу данных.

Краткие теоретические сведения:

Если требуется добавить в таблицу большой массив данных, удобно использовать для этого команду загрузки данных из файла. Загрузка из файла выполняется программой PostgreSQL значительно быстрее, чем вставка строк с помощью команды INSERT.

Например, чтобы загрузить данные в таблицу Customers, выполните следующие действия.

1. Запустите стандартную программу Windows Блокнот (Пуск → Все программы → Стандартные → Блокнот).

2. В окне программы Блокнот введите данные, используя для отделения значений друг от друга клавишу Tab, а для перехода на следующую строку – клавишу Enter.

Вместо отсутствующего значения необходимо при заполнении файла ввести символы «\N». Тогда в базу данных будет загружено неопределенное значение (NULL).

3. Для сохранения файла с данными нажмите комбинацию клавиш Ctrl+S. В стандартном окне Windows *Сохранить как* выберите папку, в которую нужно поместить файл (например, C: \data). Введите имя файла (например, Customers. txt) и нажмите кнопку Сохранить.

4. Для загрузки данных из созданного файла выполните команду

```
LOAD DATA LOCAL INFILE 'C:/data/Customers.txt'
```

```
INTO TABLE Customers
```

```
CHARACTER SET utf8;
```

Обратите внимание, что в пути к файлу необходимо использовать прямую косую черту, а не обратную.

Порядок выполнения работы:

1. Создайте базу данных на сервере PostgreSQL.
2. Сценарий SQL предоставлен так, чтобы создать большинство таблиц и вставки данных в них. Все, что нужно сделать, это импортировать сценарий SQL в вашу базу данных. database.sql.
3. Таблица Сотрудники (персонал, должности) не включены в этот сценарий SQL. Обратитесь к диаграмме базы данных (ERD) и словарю данных.
4. Создайте таблицы сотрудников (персонал, положение и расписаний) согласно спецификации.
5. Все данные сотрудников представлены в файле employee-import.
6. Эти данные не отформатированы для импортирования непосредственно в базу данных, необходимо отформатировать данные и загрузить их в таблицы, которые вы только что создали.
7. В поле " Full Name" в формате "Имя Фамилия" используются разные символы

- разделителя.
8. Убедитесь, что адреса электронной почты в правильном формате

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 1.3 Разработка и администрирование базы данных

Лабораторное занятие №9

Вычисления в SQL запросах. Упорядочение выборки.

Цель: Формирование умений создавать запросы на выборку данных с использованием различных вычислений, применением прямой и обратной сортировки.

Выполнив работу, вы будете уметь:

У 1.2.2 оптимизировать запросы к базе данных для повышения производительности;
Уо 02.02 использовать современное программное обеспечение, различные цифровые средства для решения профессиональных задач; проявлять культуру информационной безопасности при использовании информационно-коммуникационных технологий.

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: Postgre SQL, MS Office

Задание:

Разработать модифицирующие запросы к базе данных Туризм.

Краткие теоретические сведения:

Модификация базы данных с помощью запросов на изменение

Запрос на обновление

Запрос этого типа используется при необходимости внесения изменений во множество записей базы данных, поэтому целесообразно сделать резервную копию таблицы.

Выполняется этот вид запроса в два этапа: сначала проверяется правильность отбора обновляемых записей с помощью запроса на выборку, затем он преобразуется в запрос на обновление и выполняется повторно.

Периодически убирая в архивные таблицы «старые» записи, можно увеличить быстродействие основных частей и улучшить обзорность базы данных.

Кроме того, при необходимости добавить данные в таблицу базы данных из другой базы можно также использовать запросы на добавление.

Запрос на удаление

«Старые» или неиспользуемые записи таблиц можно удалить, но обязательно сначала произвести выборку и проверить ее. Целесообразно сделать копию.

Порядок выполнения работы:

Запрос на создание

1. Создайте обобщенную таблицу **Договоры по странам**, включив в нее следующие поля:

Из таблицы **Договоры**: *Номер договора; Название клиента*

Из таблицы **Страны**: *Название страны; Регион.*

2. Увеличьте **Размер оклада** у менеджеров по продажам на 15%.
3. Создайте путем копирования дубликат таблицы **Договоры** без данных, назвав ее **Договоры 2023 года**. Просмотрите таблицу **Договоры 2023 года** – она должна быть пустой и иметь такую же структуру, как и таблица **Договоры**.
4. Отберите в таблицу **Договоры 2023 года** записи обо всех договорах этого года.

Выполните запрос и подтвердите добавление; просмотрите результаты архивации и сохраните запрос, обратив внимание на значок у его имени.

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 1.3 Разработка и администрирование базы данных

Лабораторное занятие №10

Организация SQL запросов на выборку данных с использованием составных условий.

Цель: получение практических навыков по освоению операций создания запросов с помощью языка SQL

Выполнив работу, вы будете уметь:

У 1.2.2 оптимизировать запросы к базе данных для повышения производительности;

Уо 02.02 использовать современное программное обеспечение, различные цифровые средства для решения профессиональных задач; проявлять культуру информационной безопасности при использовании информационно-коммуникационных технологий

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: PostgreSQL, PostgreSQL.

Задание :

Разработать запросы для базы данных Туризм с помощью конструктора в PostgreSQL.

Краткие теоретические сведения:

Запрос является объектом базы данных. Он представляет собой сформулированную информационную потребность.

При работе с запросом можно выделить два этапа: формирование (проектирование) и выполнение. При выполнении запроса выбирается информация из всех таблиц базы данных в соответствии с критерием запроса.

Порядок выполнения работы:

Запрос на выборку

1. Сделайте текущей базу данных **Туризм** и создайте следующие запросы:

- список всех путешествий в определенную страну (например, Испанию);
- список всех регионов в конкретной стране (например, Англии).
- все туры, проданные в 2023 году. Сохраните запрос с именем «*Туры 2023*»;
- список сотрудников, работающих с 1999 года и раньше. Сохраните запрос с именем «Ветераны». Добавьте в запрос строку «Сортировка» и установите сортировку по фамилиям.
- сотрудникам, которые родились в 1973 г., используя в качестве критерия выражение: **Between... and**, а затем повторите запрос, построив выражение с помощью знаков «<» и «>»;
- сотрудникам, фамилии которых с «Г» по «Я»;
- сотрудникам, фамилии которых начинаются с «Н» по «Я» и с «А» по «В»; индивидуальным клиентам, фамилии которых имеют вторую букву «о»;
- пяти фамилиям сотрудников, которые начинаются с букв «А» или «В». постоянным клиентам, количество договоров с которыми больше 3.

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 1.3 Разработка и администрирование базы данных

Лабораторное занятие №11

Использование агрегированных функций для организации SQL запросов на выборку данных.

Цель: получение практических навыков по освоению операции конкатенации и строковых функций в запросах на языке SQL

Выполнив работу, вы будете уметь:

У 1.2.2 оптимизировать запросы к базе данных для повышения производительности;

Уо 02.02 использовать современное программное обеспечение, различные цифровые средства для решения профессиональных задач; проявлять культуру информационной безопасности при использовании информационно-коммуникационных технологий

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания,

компьютер, программное обеспечение: Postgre SQL, Postgre SQL.

Задание: Составить запросы для Учебной базы данных

Краткие теоретические сведения:

Операция конкатенации «&» в Postgre SQL позволяет соединять («склеивать») значения двух или более столбцов символьного типа или символьных констант в одну строку.

Пример:

```
SELECT SURNAME&'__'&NAME, STIPEND  
FROM STUDENT
```

```
WHERE KURS = 4 AND STIPEND> 0;
```

Результат выполнения запроса:

Функции преобразования символов в строке

LCASE(строка)-перевод в строчные символы

UCASE(строка)-перевод в прописные символы

Пример:

```
SELECT LCASE (SURNAME), UCASE (NAME)  
FROM STUDENT  
WHERE KURS = 4 AND STIPEND> 0;
```

Строковые функции

LEN (строка) –определение длины строки

□тип возвращаемого значения INT

□функция возвращает NULL, если строка имеет NULL значение

LEFT (строка, длина) – выделение левой подстроки

RIGHT (строка, длина) – выделение правой подстроки

MID (строка, начало, количество) – выделение подстроки

LTRIM (строка) – удаляет пробелы в начале строки

RTRIM (строка) – удаляет пробелы в конце строки

STR (число) – выполняет конвертирование числового значения в символьный формат

INSTR (строка, подстрока [, <начало поиска> [, <номер вхождения>]]) -

возвращает позицию найденной подстроки (тип возвращаемого значения- INT), где

<начало поиска> задает начальную позицию в строке для поиска<подстроки>. Если не задано, то по умолчанию применяется значение 1;

<номер вхождения> задает порядковый номер искомой подстроки. Если не задан, то по умолчанию применяется значение 1.

Значение выражений в <начале поиска> или <номера вхождения> должны иметь беззнаковый целый тип или приводиться к этому типу

Примеры: Результат выполнения запросов:

1. SELECT SURNAME,LEN (SURNAME),KURS
FROM STUDENT
WHERE KURS = 3;

SURNAME	Expr1001	KURS
Петров	5	3
Павлов	5	3
Лукин	5	3

2. SELECT SURNAME, Right(SURNAME,2)
FROM STUDENT
WHERE KURS IN (2, 3, 4);

SURNAME	Expr1001
Петров	ов
Сидоров	ов
Кузнецов	ов
Вайков	ов
Павлов	ов
Лукин	ин
Петров	ов

Порядок выполнения работы:

1. Составьте запрос для таблицы STUDENT «Учебной базы данных» таким образом, чтобы выходная таблица была представлена одним столбцом, содержащим последовательность, разделенных символом «;» значений всех столбцов этой таблицы; результат необходимо представить прописными символами (например, **ИВАН;ИВАНОВ;150;1;ОРЕЛ;3/12/82;10**).

2. Составьте запрос для таблицы STUDENT «Учебной базы данных» таким образом, чтобы выходная таблица содержала один столбец в следующем виде: **Б.КУЗНЕЦОВ; место жительства – БРЯНСК; родился – 8.12.81**.

3. Составьте запрос для таблицы STUDENT «Учебной базы данных» таким образом, чтобы столбец выходной таблицы был представлен в следующем виде: **Б;Кузнецов;место жительства-Брянск;родился-08.12.1981**

4. Составьте запрос для таблицы STUDENT «Учебной базы данных» таким образом, чтобы выходная таблица содержала один столбец в следующем виде: **Борис Кузнецов родился в 1981 году**.

5. Выведите имена и фамилии студентов и объем получаемой ими стипендии, увеличенной в 10 раз.

6. Составьте запрос для таблицы STUDENT «Учебной базы данных» следующим образом: **БОРИС КУЗНЕЦОВ родился в 1981 году**, только для студентов 1, 2 и 4-го курсов.

7. Составьте запрос для таблицы UNIVERSITY «Учебной базы данных» таким образом, чтобы выходная таблица содержала один столбец в следующем виде: **Код – 10;ВГУ – г.ВОРОНЕЖ;Рейтинг=296**.

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 1.3 Разработка и администрирование базы данных

Лабораторное занятие №12

Использование группировки данных.

Цель: Формирование умений создавать запросы на выборку данных с применением группировки

Выполнив работу, вы будете уметь:

У 1.2.2 оптимизировать запросы к базе данных для повышения производительности;

Уо 02.02 использовать современное программное обеспечение, различные цифровые средства для решения профессиональных задач; проявлять культуру информационной безопасности при использовании информационно-коммуникационных технологий

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: Postgre SQL

Задание 1:

Разработать запросы для базы данных Туризм с помощью конструктора в Postgre SQL.

Краткие теоретические сведения:

Запрос является объектом базы данных. Он представляет собой сформулированную информационную потребность.

При работе с запросом можно выделить два этапа: формирование (проектирование) и выполнение. При выполнении запроса выбирается информация из всех таблиц базы данных в соответствии с критерием запроса.

Критерии запроса устанавливаются в строке «Условия отбора» и последующих строках, связанных логическим оператором OR. Все критерии отбора, указанные в одной строке, объединяются оператором AND.

Порядок выполнения работы:

1. Создайте запрос для расчета ведомости заработной платы для сотрудников агентства, включив в нее следующие поля: **Фамилия сотрудника, Размер оклада, Стаж, Надбавка, Налог, На руки.**

Для поля **Стаж** нужно использовать формулу, в которой учитывается сегодняшняя дата и **Дата найма** на работу:

$$\text{Стаж} = (\text{Date()} - \text{Сотрудники!ДатаНайма}) / 365$$

Для поля **Надбавка** нужно исходить из того, что она составляет 20% от **Размера оклада**, если **Стаж** меньше 5 лет, и 30% — если стаж больше 5 лет:

Поле **Налог** рассчитывается как 13% от **Размера оклада**:

Поле **На руки** рассчитывается как:

$$[\text{Размер оклада}] + [\text{надбавка}] - [\text{налог}].$$

В результате выполнения запроса будет получена новая ведомость.

2. Создайте запрос для определения стоимости путевок корпоративных клиентов, включив в него поля **Клиент, Стоимость путевки**
3. Сформируйте запрос для выборки всех туров по названию страны.

4. Создайте запрос для получения данных на сотрудников, работающих по турам в конкретную страну.
5. Создайте запрос по всем клиентам, оформившим договоры в определенную страну и регион.

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 1.3 Разработка и администрирование базы данных

Лабораторное занятие №13

Использование условий при модификации данных.

Цель: Формирование умений записи в таблице при соблюдении одного или нескольких условий

Выполнив работу, вы будете уметь:

У 1.2.2 оптимизировать запросы к базе данных для повышения производительности;

Уо 02.02 использовать современное программное обеспечение, различные цифровые средства для решения профессиональных задач; проявлять культуру информационной безопасности при использовании информационно-коммуникационных технологий

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: PostgreSQL.

Задание 1:

Разработать запросы для Учебной базы данных

Краткие теоретические сведения:

Оператор SELECT (выбрать) языка SQL является самым важным и самым часто используемым оператором. Он предназначен для выборки информации из таблиц базы данных. Упрощенный синтаксис оператора SELECT выглядит следующим образом:

```
SELECT [DISTINCT] <список атрибутов>
FROM <список таблиц>
[WHERE <условие выборки>]
[ORDER BY <список атрибутов>]
[GROUP BY <список атрибутов>]
[HAVING <условие>]
[UNION <выражение с оператором SELECT>];
```

В квадратных скобках указаны элементы, которые могут отсутствовать в запросе.

Ключевое слово SELECT сообщает базе данных, что данное предложение является запросом на извлечение информации. После слова SELECT через запятую перечисляются наименования полей, содержимое которых запрашивается.

Обязательным ключевым словом в предложении-запросе SELECT является слово

FROM (из). За ключевым словом FROM указывается список разделенных запятыми имен таблиц, из которых извлекается информация.

Например,

```
SELECT NAME, SURNAME  
FROM STUDENT;
```

Любой SQL-запрос должен заканчиваться символом «;».

Если необходимо вывести значения всех столбцов таблицы, то можно вместо перечисления их имен использовать символ «*/»

```
SELECT *  
FROM STUDENT;
```

Для исключения из результата SELECT-запроса повторяющихся записей используется ключевое слово DISTINCT (отличный). Если запрос SELECT извлекает множество полей, то DISTINCT исключает дубликаты строк, в которых значения всех выбранных полей идентичны.

Использование в операторе SELECT предложения, определяемого ключевым словом WHERE (где), позволяет задавать выражение условия, принимающее значение истина или ложь для значений полей строк таблиц, к которым обращается оператор SELECT. Предложение WHERE определяет, какие строки указанных таблиц должны быть выбраны.

Упорядочение выходных полей (ORDER BY) .

ORDER BY позволяет упорядочивать записи в соответствии со значениями одного или нескольких выбранных полей. При этом можно задать возрастающую (ASC) и убывающую (DESC) последовательность сортировки для каждого из столбцов.

По умолчанию принята возрастающая последовательность сортировки.

Порядок выполнения работы:

1. Выбрать все данные из таблицы предметов обучения SUBJECT с упорядочением по наименованиям предметов.

```
SELECT * FROM  
SUBJECT  
  
ORDER BY SUBJ_NAME;
```

Тот же список, но упорядоченный в обратном порядке.

```
SELECT * FROM  
SUBJECT  
ORDER BY SUBJ_NAME DESC;
```

3. Упорядочить выводимый список предметов по семестрам, а внутри семестров – по наименованиям предметов.

```
SELECT * FROM
```

```
SUBJECT
ORDER BY SEMESTER, SUBJ_NAME;
```

4. При упорядочении вместо наименований столбцов можно указывать их номера, имея, однако, в виду, что данном случае это номера столбцов указанные при определении выходных данных в запросе, а не номера столбцов в таблице. Полем с номером 1 является первое поле, указанное в предложении ORDER BY- независимо от его расположения в таблице.

```
SELECT SUBJ_ID, SEMESTER
FROM SUBJECT
ORDER BY 2 DESC;
```

Результат выполнения запроса:

Если в поле, которое используется для упорядочения, существуют NULL- значения, то все они размещаются в конце или предшествуют всем остальным значениям этого поля.

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 1.3 Разработка и администрирование базы данных

Лабораторное занятие №14

Организация SQL запросов на выборку данных с использованием подзапросов.

Цель: Формирование умений создавать сложные запросы на выборку данных, используя подзапросы

Выполнив работу, вы будете уметь:

У 1.2.2 оптимизировать запросы к базе данных для повышения производительности;

Уо 02.02 использовать современное программное обеспечение, различные цифровые средства для решения профессиональных задач; проявлять культуру информационной безопасности при использовании информационно-коммуникационных технологий

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: Postgre SQL.

Задание 1:

Разработать запросы для Учебной базы данных

Краткие теоретические сведения:

Использование в операторе SELECT предложения, определяемого ключевым словом

WHERE(где), позволяет задавать выражения условия (предикат), принимающее значение *истина* или *ложь* для значений полей строк таблиц, к которому обращается оператор SELECT. Предложение WHERE определяет, какие строки *указанных таблиц должны быть выбраны*. В таблицу, являющуюся результатом запроса, включается только те строки, для которых условие(предикат), указанное в предложении WHERE, принимает значение *истина*.

Операторы **IN** (равен любому из списка) **NOT IN** (не равен ни одному из списка) используется для сравнения проверяемого значения поля с заданным списком. Этот список значений указывается в скобках справа от оператора IN. Построенный с использованием IN предикат (условие) считается истинным, если значения поля, имя которого указано слева от IN, *совпадает* (подразумевается точное совпадение) с одним из значений, перечисленных в списке, указанном в скобках справа от IN.

Предикат, построенный с использованием NOT IN, считается истинным, если значение поля, имя которого указано слева от NOT IN, не совпадает ни с одним из значений, перечисленных в списке, указанном в скобках от NOT IN.

Оператор **BETWEEN** используется для проверки условия вхождения значения поля в заданный интервал, то есть вместо списка значений атрибута этот оператор задает границы его изменения.

Оператор **LIKE** применим только к символьным полям типа CHAR или VARCHAR. Этот оператор просматривает строковые значения полей с целью определения, входит ли заданная в операцию LIKE подстрока (образец поиска) в символьную строку-значение проверяемого поля. Можно применять шаблон:

Символ подчеркивания «?» указанный в шаблоне, определяет возможность наличия в указанном месте одного любого символа;

Символ «*» допускает присутствие в указанном месте проверяемой строки последовательности любых символов произвольной длины.

Like `_[A-K]*` – значения атрибутов символьного типа, начинающиеся с А по К

Like `_[!A-K]*` - значения атрибутов символьного типа, начинающиеся на любые символы кроме символов с А по К (с Л по Я)

Like `_[A,K]*` - значения атрибутов символьного типа, начинающиеся с А или К

Оператор NULL используется для проверки содержимого поля на наличие в нем пустого значения. **IS NULL** (является пустым), **IS NOT NULL**(является не пустым).

Порядок выполнения работы:

1. Написать запрос, выполняющий выборку имен всех студентов с фамилией Петров, сведения о которых находятся в таблице STUDENT

```
SELECT NAME, SURNAME  
FROM STUDENT  
WHERE SURNAME = 'Петров';
```

Результат выполнения запроса:

2. В задаваемых предложениях WHERE условиях могут использоваться операции сравнения, определяемые операторами =(равно), >(больше), <(меньше), >=(больше или равно), <=(меньше или равно), <>(не равно), а также логические операторы AND, OR и NOT. Запрос для получения имен и фамилий студентов, обучающихся на третьем курсе и получающих стипендию(размер стипендии больше нуля), будет выглядеть таким образом:

```
SELECT NAME, SURNAME
FROM STUDENT
WHERE KURS = 3 AND STIPEND > 0;
```

Результат выполнения запроса:

3. Получить из таблицы EXAM_MARKS сведения о студентах, имеющих экзаменационные оценки только 4 и 5.

```
SELECT *
FROM EXAM_MARKS
WHERE MARK IN (4, 5);
```

4. Получить из таблицы EXAM_MARKS сведения о студентах, не имеющих ни одной экзаменационной оценки только 4 и 5.

```
SELECT *
FROM EXAM_MARKS WHERE
MARK NOT IN (4, 5);
```

5. Запрос на вывод записей о предметах, на изучение которых отводится количество часов, находящееся в пределах между 30 и 40, имеет вид:

```
SELECT * FROM
SUBJECT
WHERE HOUR BETWEEN 30 AND 40;
```

Результат выполнения запроса:

Запрос1			
SUBJ_ID	SUBJ_NAME	HOUR	SEMESTER
22	Физика	34	1
56	История	34	4
73	Физкультура	34	5
*	0	0	0

6. Написать запрос, выбирающий из таблицы STUDENT сведения о студентах, фамилии которых начинаются на букву «К».

```
SELECT *
FROM STUDENT
WHERE SURNAME LIKE —K*|;
```

Результат выполнения запроса:

Запрос1							
STUDENT_ID	SURNAME	NAME	STIPEND	KURS	CITY	BIRTHDAY	UNIV_ID
10	Кузнецов	Борис	0	2	Брянск	08.12.1981	10
32	Котов	Павел	150	5	Белгород		14
*	0		0	0			0

7. Вывести все данные студента с номером 265;
8. Вывести информацию о студентах с именем Вадим;
9. Выбрать фамилии студентов со 2-го и 3-го курсов получающих стипендию;
10. Вывести имена, фамилии и даты рождения студентов 5 курса;
11. Вывести информацию о студентах из Воронежа;
12. Выбрать фамилию, имя, курс, город студентов получающих стипендию от 100 до 200 рублей;
13. Вывести список идентификаторов университетов, исключая повторения.

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 1.3 Разработка и администрирование базы данных

Лабораторное занятие №15

Импорт данных пользователя в базу данных.

Цель: Формирование умений импортировать данные в базу данных, обрабатывать csv-файлы, менять кодировку.

Выполнив работу, вы будете уметь:

У 1.4.3 создавать и восстанавливать резервные копии данных

Уо 04.01 организовывать работу коллектива и команды

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: PostgreSQL.

Задание

Общие сведения

В реальных проектах данные редко вводятся вручную через SQL-запросы. Чаще всего возникает необходимость загрузить в базу данных большие объемы информации из внешних источников: отчеты Excel, выгрузки из других систем, логи и т.д. PostgreSQL предоставляет для этого мощный и эффективный инструмент — команду COPY. Она позволяет быстро перенести данные из файла, расположенного на сервере, в таблицу базы данных.

В рамках этой работы вы создадите таблицу, подготовите файл с данными в формате CSV и выполните его импорт, изучив основные параметры команды и способы решения проблем, возникающих при загрузке.

2. Задание к лабораторной работе

1. Подготовка данных: Создать текстовый файл с данными о товарах в формате CSV (Comma-Separated Values).

2. Создание структуры: В базе данных создать пустую таблицу с соответствующей структурой для приема данных.

3. Импорт данных: Выполнить загрузку данных из файла в таблицу с помощью SQL-команды COPY.

4. Верификация: Проверить корректность загруженных данных с помощью SQL-запросов.

5. Обработка ошибок: Модифицировать файл или команду COPY для устранения типичных ошибок импорта (например, несоответствие порядка столбцов).

3. Порядок выполнения работы

Примечание: Предполагается, что у вас есть доступ к базе данных PostgreSQL и вы можете выполнять SQL-команды (например, через psql или pgAdmin).

Шаг 1. Подготовка CSV-файла

1.1. Создайте на своем компьютере текстовый файл с именем products.csv.

1.2. Откройте его в любом текстовом редакторе (например, Notepad++ или VS Code) и заполните данными. Первая строка должна содержать заголовки столбцов.

Пример содержимого products.csv:

```
product_id,product_name,price,category
1,Ноутбук Lenovo IdeaPad,55000,Электроника
2,Кофемолка Bosch MKM 6003,3500,Бытовая техника
3,Книга "1984" Джорджа Оруэлл,800,Книги
```

* Важно: Используйте запятую (,) в качестве разделителя.

* Важно: Если данные содержат запятые или кавычки, их следует заключать в двойные кавычки ("), как показано в примере с названием книги.

Шаг 2. Создание таблицы в базе данных

2.1. Подключитесь к вашей базе данных.

2.2. Выполните SQL-команду для создания таблицы products, структура которой соответствует вашему CSV-файлу.

```
CREATE TABLE products (
    product_id SERIAL PRIMARY KEY,
    product_name VARCHAR(255) NOT NULL,
    price NUMERIC(10, 2) NOT NULL,
    category VARCHAR(100)
);
```

* SERIAL для product_id создаст автоинкрементируемый первичный ключ.

* Типы данных (VARCHAR, NUMERIC) должны соответствовать данным в вашем файле.

Шаг 3. Выполнение импорта с помощью команды COPY

3.1. Команда COPY выполняется на сервере базы данных, поэтому путь к файлу products.csv должен быть доступен именно серверу. Для локального тестирования часто используют \copy — аналог COPY, который читает файл с компьютера клиента.

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 1.3 Разработка и администрирование базы данных

Лабораторное занятие №16

Экспорт данных базы в документы пользователя.

Цель: Формирование умений экспортировать данные из таблицы в csv-файлы

Выполнив работу, вы будете уметь:

У 1.4.3 создавать и восстанавливать резервные копии данных;

Уо 04.01 организовывать работу коллектива и команды

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания,

компьютер, программное обеспечение: Postgre SQL

Задание :

1. Общие сведения

Экспорт данных — это процесс, обратный импорту. Он необходим для решения множества задач: создания резервных копий отдельных таблиц, обмена данными с другими системами, подготовки отчетов для пользователей, не имеющих прямого доступа к базе данных, или для последующего анализа данных в электронных таблицах (например, Microsoft Excel).

Основным инструментом для этой задачи в PostgreSQL является команда COPY. Она позволяет выгружать данные из таблицы или результата выполнения SQL-запроса в файл, расположенный на сервере. Для пользователей, работающих с клиентского компьютера, удобнее использовать её аналог \copy.

2. Задание к лабораторной работе

Подготовка к экспорту: Создать в базе данных таблицу и заполнить её тестовыми данными (можно использовать результаты выполнения предыдущей лабораторной работы по импорту).

Экспорт всей таблицы: Выполнить выгрузку всех данных из созданной таблицы в файл формата CSV на локальный компьютер.

Экспорт по условию: Выполнить выгрузку только части данных (например, товаров определенной категории) в текстовый файл с другим разделителем.

Верификация: Проверить содержимое созданных файлов и убедиться, что данные экспортированы корректно.

Порядок выполнения работы

Примечание: Предполагается, что у вас есть доступ к базе данных PostgreSQL и вы можете выполнять SQL-команды.

Шаг 1. Подготовка данных в базе

1.1. Если у вас нет готовой таблицы, создайте её и добавьте несколько строк для теста.

```
CREATE TABLE IF NOT EXISTS employees (  
    id SERIAL PRIMARY KEY,  
    first_name VARCHAR(50),  
    last_name VARCHAR(50),  
    department VARCHAR(100),  
    salary NUMERIC(10,2)  
);  
  
-- Вставим несколько тестовых записей  
INSERT INTO employees (first_name, last_name, department, salary)  
VALUES  
( 'Иван', 'Иванов', 'Отдел разработки', 150000.00),  
( 'Петр', 'Петров', 'Отдел маркетинга', 95000.50),  
( 'Анна', 'Сидорова', 'Отдел разработки', 145000.75);
```

Шаг 2. Экспорт всей таблицы в CSV-файл (на клиентской машине)

2.1. Для экспорта данных с клиентской машины (вашего компьютера) используется команда `\copy`. Она работает аналогично `COPY`, но читает/пишет файлы на стороне клиента.

2.2. Выполните следующую команду в вашем SQL-клиенте (например, `psql`). Убедитесь, что у вас есть права на запись в указанную папку.

```
\copy (SELECT * FROM employees) TO 'C:\exports\employees_full.csv' WITH CSV  
HEADER;
```

(SELECT * FROM employees): Источник данных. Это может быть как целая таблица, так и результат любого SELECT-запроса.

TO 'C:\exports\employees_full.csv': Путь к файлу на вашем компьютере, куда будут записаны данные.

WITH CSV HEADER: Опции вывода.CSV — указывает, что файл должен быть в формате CSV (значения разделяются запятыми).

HEADER — добавляет первую строку с именами столбцов.

Шаг 3. Экспорт отфильтрованных данных с другим разделителем

3.1. Теперь выполним более сложную задачу: выгрузим только сотрудников из "Отдела разработки" и используем в качестве разделителя точку с запятой (;), что часто требуется для европейских версий Excel.

3.2. Выполните команду:

```
\copy (  
    SELECT first_name, last_name, salary  
    FROM employees  
    WHERE department = 'Отдел разработки'  
) TO 'C:\exports\developers_semicolon.txt' WITH (FORMAT 'csv', DELIMITER ';',  
HEADER);
```

WHERE department = 'Отдел разработки': Условие фильтрации данных.

WITH (FORMAT 'csv', DELIMITER ';', HEADER): Явное указание опций. FORMAT 'csv'

— формат файла.

DELIMITER ';' — заменяет стандартный разделитель (запятую) на точку с запятой.

HEADER — включает строку заголовка.

Шаг 4. Проверка результатов

4.1. Перейдите в папку C:\exports\ на вашем компьютере и откройте созданные файлы (employees_full.csv и developers_semicolon.txt) с помощью текстового редактора или Microsoft Excel.

4.2. Убедитесь, что:

Данные соответствуют тем, что находятся в таблице.

В первом файле разделителем является запятая, во втором — точка с запятой.

В обоих файлах присутствует строка с заголовками столбцов.

4. Требования к отчёту

Отчёт по лабораторной работе должен содержать:

Титульный лист с указанием названия работы и ваших данных.

SQL-код, использованный для создания и наполнения тестовой таблицы (CREATE TABLE, INSERT).

SQL-команды, использованные для экспорта данных (\copy).

Скриншоты содержимого созданных файлов (в текстовом редакторе или Excel), подтверждающие успешный экспорт.

Ответы на контрольные вопросы.

Вывод по проделанной работе.

5. Контрольные вопросы для самопроверки

В чём заключается ключевое различие между командами COPY и \copy?

Каким образом можно выгрузить не всю таблицу целиком, а только результат выполнения сложного запроса с условиями (WHERE, JOIN)?

Как изменить символ-разделитель в выходном файле при использовании команды COPY/\copy?

Для чего используется опция HEADER и что произойдёт, если её не указать?

Какой формат файла (FORMAT) следует указывать для создания стандартного CSV-файла?

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 1.3 Разработка и администрирование базы данных

Лабораторное занятие №17

Создание представлений.

Цель: Формирование умений создавать представления на основе созданных запросов к базе данных.

Выполнив работу, вы будете уметь:

У 1.2.2 оптимизировать запросы к базе данных для повышения производительности

Уо 02.02 использовать современное программное обеспечение, различные цифровые средства для решения профессиональных задач; проявлять культуру информационной безопасности при использовании информационно-коммуникационных технологий

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: PostgreSQL.

Задание 1:

Разработать представления в базе данных Туризм.

Представление (или вью, от англ. view) в базе данных — это виртуальная таблица, содержимое которой определяется результатом выполнения SQL-запроса.

Проще говоря, представление — это «сохранённый запрос», которому присваивается имя, и после этого его можно использовать в других запросах так же, как обычную таблицу. При этом сами данные в представлении не хранятся. Каждый раз, когда вы обращаетесь к представлению, СУБД выполняет соответствующий ему запрос, получает данные из базовых (реальных) таблиц и показывает их вам.

Ключевые характеристики и назначение представлений

1. Виртуальность: Представление не хранит данные физически. Это просто "окно" или "шаблон" для доступа к данным, лежащим в других таблицах.
2. Динамичность: Если данные в базовых таблицах изменятся, то при следующем обращении к представлению вы увидите уже обновлённые данные.
3. Упрощение: Позволяют скрыть сложную логику (например, соединения нескольких таблиц JOIN, вычисления, группировки) за простым и понятным именем.
4. Безопасность: Являются мощным инструментом для разграничения прав доступа. Можно дать пользователю доступ к представлению, но закрыть доступ к исходным таблицам, тем самым ограничив видимость только определёнными строками или столбцами.

Аналогия из жизни

Представьте, что у вас есть большая папка с документами (это ваша база данных), а внутри неё — несколько толстых папок с финансовыми отчётами за разные годы (это таблицы).

Вместо того чтобы каждый раз искать нужные отчёты, складывать их на стол и вручную суммировать показатели, вы создаёте "Сводный отчёт за год" (это представление). Этот сводный отчёт — не новая папка с бумагами, а просто лист с инструкцией: «взять данные из папки за 1-й квартал, из папки за 2-й квартал и т.д., и сложить их». Когда вам нужен годовой отчёт, вы просто смотрите на этот лист. Если кто-то изменит данные в исходных квартальных папках, то при следующем взгляде на "Сводный отчёт" вы увидите уже новые, актуальные цифры.

Как создать представление

Для создания представления используется команда CREATE VIEW. В её основе лежит любой валидный SELECT-запрос.

Пример:

Допустим, у нас есть две таблицы: orders (заказы) и customers (клиенты).

* orders: order_id, customer_id, order_date, total_amount

* customers: customer_id, first_name, last_name

Нам часто нужно получать список заказов с именами клиентов. Вместо того чтобы каждый раз писать сложный запрос с JOIN, мы создадим представление:

```
CREATE VIEW customer_orders AS
```

```
SELECT
```

```
    o.order_id,
```

```
    c.first_name,
```

```
    c.last_name,
```

```
    o.order_date,
```

```
    o.total_amount
```

```
FROM orders o
```

```
JOIN customers c ON o.customer_id = c.customer_id;
```

Теперь у нас есть виртуальная таблица customer_orders. Мы можем делать к ней запросы так же просто, как к обычной таблице:

```
SELECT * FROM customer_orders WHERE total_amount > 10000;
```

СУБД выполнит этот запрос, "раскроет" представление и фактически выполнит:

```
SELECT * FROM (
```

```
    SELECT ... JOIN ... WHERE ...
```

```
) AS customer_orders
```

```
WHERE total_amount > 10000;
```

Порядок выполнения работы:

1. Создайте обобщенную таблицу **Договоры по странам**, включив в нее следующие поля:
Из таблицы **Договоры**: *Номер договора; Название клиента*
Из таблицы **Страны**: *Название страны; Регион*.
2. Увеличьте **Размер оклада** у менеджеров по продажам на 15%.
Для этого:
☐ ☐ Составьте новый запрос на выборку, включив в него поля **Фамилия**, **Должность** и **Размер оклада**;
☐ ☐ Проверьте составленный запрос;
☐ ☐ Видоизмените запрос, установив ему статус «Обновление» (**Запрос – Обновление**). В появившейся в бланке запроса строке «Обновление» для поля **Размер оклада** внесите с помощью формулы: [Размер оклада]*1,15;
3. Создайте путем копирования дубликат таблицы **Договоры** без данных, назвав ее **Договоры 2023 года**. Для этого в контекстном меню для таблицы **Договоры** выберите

Копировать, затем выполните команду *Вставить*, в параметрах вставки укажите *«Только структуру»*. Просмотрите таблицу **Договоры 2023 года** – она должна быть пустой и иметь такую же структуру, как и таблица **Договоры**.

4. Отберите в таблицу **Договоры 2023 года** записи обо всех договорах этого года.

Для этого:

- a. ☐ ☐ Создайте запрос на выборку, включив в него все поля таблицы **Договоры** в любой последовательности, и критерий по дате, выполните его для проверки правильности;
- b. ☐ ☐ Измените статус запроса на *«Добавление»*, в появившемся окне задайте имя таблицы для добавления **Договоры 2023 года**, обратите внимание на появление строки *«Добавление»* в бланке запроса;
- c. ☐ ☐ Выполните запрос и подтвердите добавление; просмотрите результаты архивации и сохраните запрос, обратив внимание на значок у его имени.

Запрос на удаление

7. Удалите из таблицы **Договоры** записи о договорах 2023 года, используя копию сохраненного запроса на добавление в таблицу **Договоры 2023 года**, изменив его статус на *«Удаление»*.

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 1.3 Разработка и администрирование базы данных

Лабораторное занятие №18

Создание функций и процедур

Цель: Формирование умений создавать функции и процедуры в базе данных

Выполнив работу, вы будете уметь:

У 1.2.3 разрабатывать хранимые процедуры и триггеры для баз данных;

У 02.02 использовать современное программное обеспечение, различные цифровые средства для решения профессиональных задач; проявлять культуру информационной безопасности при использовании информационно-коммуникационных технологий;

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: PostgreSQL.

Задание 1:

Составить функции по Учебной базе данных

Краткие теоретические сведения:

В то время как стандартный SQL отлично подходит для запросов к данным, для

реализации сложной бизнес-логики, многократного выполнения операций или управления транзакциями на стороне сервера СУБД требуются более мощные инструменты. Для этого в PostgreSQL используется процедурный язык PL/pgSQL.

Ключевые понятия:

- * Функция (Function): Подпрограмма, которая всегда возвращает значение. Вызов функции является частью SQL-выражения (например, SELECT). Она не может управлять транзакциями (начинать, коммитить или откатывать их).

- * Процедура (Procedure): Подпрограмма, которая не возвращает значения. Вызывается с помощью команды CALL. Основное отличие от функции — возможность управлять транзакциями внутри своего тела (команды BEGIN, COMMIT, ROLLBACK).

Использование этих объектов позволяет:

- * Уменьшить нагрузку на сеть, выполняя логику на сервере.
- * Обеспечить целостность и безопасность данных.
- * Повысить производительность за счет кэширования планов выполнения.

2. Задание к лабораторной работе

1. Создание функции:

- * Создать функцию `get_customer_info(customer_id INT)`, которая принимает идентификатор клиента и возвращает его полное имя и адрес в виде одной отформатированной строки.

2. Создание процедуры:

- * Создать процедуру `transfer_funds(from_account_id INT, to_account_id INT, amount NUMERIC)`, которая переводит указанную сумму денег с одного счета на другой. Процедура должна проверять, достаточно ли средств на счете-отправителе, и выполнять операцию в рамках одной транзакции.

3. Вызов и тестирование:

- * Вызвать созданную функцию с помощью SELECT.
- * Вызвать созданную процедуру с помощью CALL.
- * Проверить результаты выполнения, сделав выборку данных из соответствующих таблиц до и после вызова.

3. Порядок выполнения работы

Примечание: Для выполнения работы используется учебная база данных (например, dvd-rental или любая другая с таблицами customer, payment).

Шаг 1. Подготовка тестовых данных

Для выполнения заданий нам понадобятся таблицы. Если их нет, можно создать упрощенные версии.

-- Упрощенная таблица клиентов

```
CREATE TABLE IF NOT EXISTS lab_customers (  
    customer_id SERIAL PRIMARY KEY,  
    first_name VARCHAR(50),  
    last_name VARCHAR(50),  
    address VARCHAR(255)  
);
```

```
-- Упрощенная таблица счетов
CREATE TABLE IF NOT EXISTS lab_accounts (
    account_id SERIAL PRIMARY KEY,
    customer_id INT,
    balance NUMERIC(15,2) DEFAULT 0.00,
    FOREIGN KEY (customer_id) REFERENCES lab_customers(customer_id)
);
```

```
-- Вставим тестовые данные
INSERT INTO lab_customers (first_name, last_name, address) VALUES
('Иван', 'Иванов', 'ул. Центральная, д. 1, кв. 10'),
('Петр', 'Петров', 'пр-т Мира, д. 55');
```

```
INSERT INTO lab_accounts (customer_id, balance) VALUES
(1, 10000.00), -- У Иванова 10000
(2, 5000.00); -- У Петрова 5000
```

Шаг 2. Создание и вызов функции

2.1. Создадим функцию get_customer_info. Она будет возвращать текст (RETURNS TEXT).

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 1.3 Разработка и администрирование базы данных

Лабораторное занятие №19

Создание триггеров.

Цель: Формирование умений создавать триггеры к базе данных

Выполнив работу, вы будете уметь:

У 1.2.3 разрабатывать хранимые процедуры и триггеры для баз данных

Уо 02.02 использовать современное программное обеспечение, различные цифровые средства для решения профессиональных задач; проявлять культуру информационной безопасности при использовании информационно-коммуникационных технологий

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: PostgreSQL.

Задание

Модифицировать схему базы данных «Библиотека» таким образом, чтобы таблица subscribers хранила актуальную информацию о дате последнего визита читателя в

библиотеку.

Краткие теоретические сведения:

Триггер - это хранимая процедура особого типа, которую пользователь не вызывает непосредственно, а исполнение которой обусловлено определенной модификацией данных в заданной таблице или столбце БД.

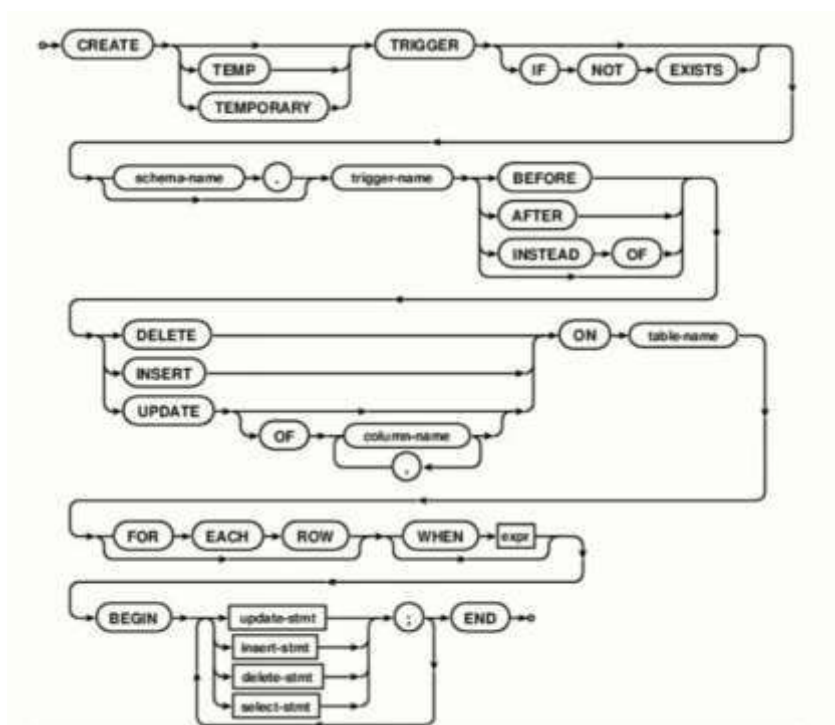
Триггер - это хранимая в базе данных процедура, автоматически вызываемая СУБД при возникновении соответствующих условий.

Триггеры применяются для обеспечения целостности данных и реализации сложной бизнес логики.

Триггер запускается сервером автоматически при попытке изменения данных в таблице, с которой он связан.

Все производимые им модификации данных рассматриваются как одна транзакция. В случае обнаружения ошибки или нарушения целостности данных происходит откат этой транзакции.

Момент запуска триггера определяется с помощью ключевых слов: BEFORE – триггер запускается до выполнения связанного с ним события, например, до добавления записи; AFTER – после события.



Синтаксис

BEFORE – триггер запускается до выполнения связанного с ним события;
AFTER – триггер запускается после выполнения связанного с ним события.
DELIMITER//

CREATE TRIGGER имя_триггера

BEFORE|AFTER триггерное_событие ON имя_таблицы
[FOR EACH {ROW|STATEMENT}]

```
BEGIN
тело_триггера;
END;
```

```
//
```

Где

FOR EACH ROW – выполняемые триггером действия задаются для каждой строки, охваченной данным событием.

FOR EACH STATEMENT – действия задаются только один раз для каждого события.

Порядок выполнения работы:

Ожидаемый результат

Таблица **subscribers** содержит дополнительное поле, хранящее актуальную информацию о дате последнего визита читателя в библиотеку.

Для решения этой задачи нужно будет выполнить три шага:

1. модифицировать таблицу **subscribers** (добавив туда поле для хранения даты последнего визита читателя);
2. проинициализировать значения последних визитов для всех читателей;
3. создать триггеры для поддержания этой информации в актуальном состоянии.

В Postgre SQL триггеры не активируются каскадными операциями, потому изменения в таблице **subscriptions**, вызванные удалением книг, останутся «незаметными» для триггеров на этой таблице.

1.

-- Модификация таблицы:

```
ALTER TABLE `subscribers`
```

```
ADD COLUMN `s_last_visit` DATE NULL DEFAULT NULL AFTER `s_name`;
```

-- Инициализация данных:

```
UPDATE `subscribers`
```

```
LEFT JOIN (SELECT `sb_subscriber`,
```

```
MAX(`sb_start`) AS `last_visit`
```

```
FROM `subscriptions`
```

```
GROUP BY `sb_subscriber`) AS `prepared_data`
```

```
ON `s_id` = `sb_subscriber`
```

```
SET `s_last_visit` = `last_visit`
```

2. Поскольку значение даты последнего визита читателя зависит от информации,

представленной в таблице **subscriptions** (конкретно — от значения поля **sb_start**),

именно на этой таблице нам и придётся создавать триггеры. В принципе, во всех трёх триггерах можно было использовать однотипное решение (**UPDATE** на основе **JOIN**),

но для разнообразия в **INSERT**-триггере мы реализуем более простой вариант: проверим, оказалась ли дата добавляемой выдачи больше, чем сохранённая дата последнего визита и, если это так, обновим дату последнего визита.

3. В **UPDATE**- и **DELETE**-триггерах такое решение не годится: если в результате этих операций окажется, что читатель ни разу не был в библиотеке, значением даты его последнего визита должен стать **NULL**. Такой результат проще всего достигается с помощью **UPDATE** на основе **JOIN**.

Для операций **UPDATE** и **DELETE** критически важно использовать именно **AFTER**-триггеры, т.к. **BEFORE**-триггеры будут работать со «старыми» данными, что приведёт к некорректному определению искомой даты.

Также обратите внимание, что в **UPDATE**-триггере мы должны обновлять дату последнего визита для двух потенциально разных читателей, т.к. при внесении изменения выдача может быть «передана» от одного читателя к другому (мы рассмотрим эту ситуацию в процессе проверки работоспособности полученного решения).

4. Удаление старых версий триггеров

-- (удобно в процессе разработки и отладки):

```
DROP TRIGGER `last_visit_on_subscriptions_ins`;
```

```
DROP TRIGGER `last_visit_on_subscriptions_upd`;
```

```
DROP TRIGGER `last_visit_on_subscriptions_del`;
```

-- Переключение разделителя завершения запроса,

-- т.к. сейчас запросом будет создание триггера,

-- внутри которого есть свои, классические запросы:

```
DELIMITER $$
```

5. Создание триггера, реагирующего на добавление выдачи книг:

```
CREATE TRIGGER `last_visit_on_subscriptions_ins`
```

```
AFTER INSERT
```

```
ON `subscriptions`
```

```
FOR EACH ROW
```

```
BEGIN
```

```
IF (SELECT IFNULL(`s_last_visit`, '1970-01-01')
```

```
FROM `subscribers`
```

```
WHERE `s_id` = NEW.`sb_subscriber`) < NEW.`sb_start`
```

```
THEN
```

```
UPDATE `subscribers`
```

```
SET `s_last_visit` = NEW.`sb_start`
```

```
WHERE `s_id` = NEW.`sb_subscriber`;
END IF;

END; $$
```

6. Создание триггера, реагирующего на обновление выдачи книг:

```
CREATE TRIGGER `last_visit_on_subscriptions_upd`

AFTER UPDATE ON
`subscriptions` FOR
EACH ROW BEGIN

UPDATE `subscribers`

LEFT JOIN (SELECT `sb_subscriber`,
MAX(`sb_start`) AS `last_visit` FROM
`subscriptions`

GROUP BY `sb_subscriber`) AS `prepared_data` ON
`s_id` = `sb_subscriber`

SET `s_last_visit` = `last_visit`

WHERE `s_id` IN (OLD.`sb_subscriber`, NEW.`sb_subscriber`);
END;

$$
```

7. Создание триггера, реагирующего на удаление выдачи книг:

```
CREATE TRIGGER `last_visit_on_subscriptions_del`

AFTER DELETE ON
`subscriptions` FOR
EACH ROW BEGIN

UPDATE `subscribers`

LEFT JOIN (SELECT `sb_subscriber`,
MAX(`sb_start`) AS `last_visit` FROM
`subscriptions`

GROUP BY `sb_subscriber`) AS `prepared_data` ON
`s_id` = `sb_subscriber`

SET `s_last_visit` = `last_visit` WHERE
`s_id` = OLD.`sb_subscriber`; END;

$$
```

-Восстановление разделителя завершения запросов:
DELIMITER ;

8. Проверим работоспособность полученного решения. Будем изменять данные в таблице subscriptions и отслеживать изменения данных в таблице subscribers. Для

начала добавим выдачу книги читателю с идентификатором 2 (ранее он никогда не был в библиотеке):

```
INSERT INTO `subscriptions` VALUES (200,
```

```
2, 1,
```

```
'2019-01-12', '2019-02-12', 'N')
```

1 Иванов И.И. 2 Петров П.П. 3 Сидоров С.С. 4 Сидоров С.С.

9. Теперь эмулируем ситуацию «книга ошибочно записана на другого читателя» и изменим идентификатор читателя в только что добавленной выдаче с 2 на 1.

NULL-значение даты последнего визита Петрова П.П. корректно восстановилось:

```
UPDATE `subscriptions`
```

```
SET `sb_subscriber` = 1
```

```
WHERE `sb_id` = 200
```

s_id s_name

1 Иванов И.И. 2 Петров П.П. 3 Сидоров
С.С. 4 Сидоров С.С.

s_last_visit

2019-01-12

NULL

2014-08-03

2015-10-08

10. Снова добавим выдачу книги Петрову П.П.:

```
INSERT INTO `subscriptions`
```

```
VALUES (201, 2,
```

```
1,
```

```
'2020-01-12',
```

```
'2020-02-12', 'N')
```

s_id s_name

1 Иванов И.И. 2 Петров П.П. 3 Сидоров С.С.
4 Сидоров С.С.

s_last_visit

2019-01-12

2020-01-12

2014-08-03

2015-10-08

11. Изменим значение даты ранее откорректированной выдачи книги (которую мы переписали с Петрова П.П. на Иванова И.И.):

```
UPDATE `subscriptions`
```

```
SET `sb_start` = '2018-01-12' WHERE
```

```
`sb_id` = 200
```

12. Удалим эту выдачу:

```
DELETE FROM `subscriptions` WHERE `sb_id` = 200
```

```
s_id s_name
```

```
1 Иванов И.И. 2 Петров П.П. 3 Сидоров С.С. 4 Сидоров С.С.
```

```
s_last_visit 2015-10-07 2020-01-12 2014-08-03 2015-10-08
```

13. Удалим единственную выдачу книги Петрову П.П.:

```
DELETE FROM `subscriptions`
```

```
WHERE `sb_id` = 201
```

14. Будут ли аннулированы изменения данных, вызванные работой **BEFORE**-триггера, если операция, активировавшая этот триггер, не сможет завершиться успешно?

Изменим вид **INSERT**-триггера с **AFTER** на **BEFORE**:

```
DROP TRIGGER `last_visit_on_subscriptions_ins`; DELIMITER $$
```

```
CREATE TRIGGER `last_visit_on_subscriptions_ins` BEFORE  
INSERT
```

```
ON `subscriptions` FOR  
EACH ROW
```

```
BEGIN
```

```
IF (SELECT IFNULL(`s_last_visit`, '1970-01-01') FROM  
`subscribers`
```

```
WHERE `s_id` = NEW.`sb_subscriber`) < NEW.`sb_start` THEN
```

```
UPDATE `subscribers`
```

```
SET `s_last_visit` = NEW.`sb_start` WHERE `s_id`  
= NEW.`sb_subscriber`; END IF;
```

```
END; $$
```

```
DELIMITER ;
```

15. Выполним последовательно добавление двух выдач книг с разными датами, но одинаковыми значениями первичных ключей:

```
INSERT INTO `subscriptions` VALUES  
(500,
```

```
2, 1,
```

```
'2020-01-12', '2020-  
02-12', 'N');
```

```
INSERT INTO `subscriptions` VALUES  
(500,
```

```
2, 1,
```

```
'2021-01-12', '2021-
```

02-12', 'N');

16. Проверим данные в таблице **subscribers**:

s_id s_name

1 Иванов И.И.

Итак, изменения аннулируются, если операция не может быть завершена успешно. В противном случае значением даты последнего визита Петрова П.П. было бы 2021-01-12.

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

МДК.01.02. Управление базами данных

Тема 2.1 Организация защиты данных в хранилищах

Лабораторное занятие №20

Транзакции и блокировки.

Цель: Формирование умений создавать процедуры с транзакциями и блокировками

Выполнив работу, вы будете уметь:

У 1.4.2 управлять транзакциями и контролировать целостность данных;

Уо 02.01 определять необходимые источники информации; эффективно искать информацию, необходимую для решения задачи и/или проблемы, оформлять результаты поиска.

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: Postgre SQL.

Задание

Создать хранимую функцию, автоматизирующую проверку условий задачи, т.е. возвращающую значение 1 (все условия выполнены) или -1, -2, -3 (если хотя бы одно условие нарушено, модуль числа соответствует номеру условия) в зависимости от того, выполняются ли следующие условия.

Порядок выполнения работы:

1. дата выдачи книги не может находиться в будущем;
2. дата возврата книги не может находиться в прошлом (только в случае вставки данных);
3. дата возврата книги не может быть меньше даты выдачи книги.

DELIMITER \$\$

**CREATE FUNCTION CHECK_SUBSCRIPTION_DATES(sb_start DATE,
sb_finish DATE,**

**is_insert INT)
RETURNS INT
DETERMINISTIC
BEGIN**

DECLARE result INT DEFAULT 1;

**-- Блокировка выдач книг с датой выдачи в будущем IF (sb_start >
CURDATE())**

THEN

SET result = -1; END IF;

**-- Блокировка выдач книг с датой возврата в прошлом. IF ((sb_finish <
CURDATE()) AND (is_insert = 1)) THEN**

SET result = -2; END IF;

**-- Блокировка выдач книг с датой возврата меньшей, чем дата выдачи. IF (sb_finish <
sb_start)**

THEN

SET result = -3; END IF;

RETURN result; END;

\$\$ DELIMITER ;

**SELECT CHECK_SUBSCRIPTION_DATES('2025-01-01', '2026-01-01', 1); SELECT
CHECK_SUBSCRIPTION_DATES('2025-01-01', '2026-01-01', 0); SELECT
CHECK_SUBSCRIPTION_DATES('2005-01-01', '2006-01-01', 1); SELECT
CHECK_SUBSCRIPTION_DATES('2005-01-01', '2006-01-01', 0); SELECT
CHECK_SUBSCRIPTION_DATES('2005-01-01', '2004-01-01', 1); SELECT
CHECK_SUBSCRIPTION_DATES('2005-01-01', '2004-01-01', 0);**

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 2.1 Организация защиты данных в хранилищах

Лабораторное занятие №21

Выполнение настроек для автоматизации обслуживания базы данных.

Цель: Формирование умений выполнять настройки для автоматизации обслуживания базы данных

Выполнив работу, вы будете уметь:

У 1.4.2 управлять транзакциями и контролировать целостность данных

Уо 02.01 определять необходимые источники информации; эффективно искать информацию, необходимую для решения задачи и/или проблемы, оформлять результаты поиска

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: Postgre SQL.

Задание1.

Создать на таблице **books** триггер, определяющий уровень изолированности транзакции, в котором сейчас проходит операция вставки, и отменяющий операцию, если уровень изолированности транзакции отличен от **SERIALIZABLE**.

Порядок выполнения работы:

1. Если операция вставки данных в таблицу **books** выполняется в транзакции с уровнем изолированности, отличным от **SERIALIZABLE**, триггер отменяет эту операцию и порождает исключительную ситуацию.
2. Для простоты (отсутствия необходимости вручную выполнять вставку) и единообразия (поддержки всеми тремя СУБД) используем **AFTER**-триггеры. Таким образом, представленные ниже решения будут отличаться только логикой определения уровня изолированности транзакций, т.к. в каждой СУБД соответствующий механизм реализован совершенно особенным, несовместимым с другими СУБД, образом.

DELIMITER \$\$

```
CREATE TRIGGER `books_ins_trans` AFTER  
INSERT
```

```
ON `books` FOR  
EACH ROW BEGIN
```

```
DECLARE isolation_level VARCHAR(50); SET  
isolation_level =
```

```
(
```

```
SELECT `VARIABLE_VALUE` FROM  
`information_schema`.  
`session_variables`
```

```
WHERE `VARIABLE_NAME` =  
'tx_isolation'
```

```
);
```

```
IF (isolation_level != 'SERIALIZABLE') THEN
```

```
SIGNAL SQLSTATE '45001' SET MESSAGE_TEXT = 'Please, switch your transaction to
SERIALIZABLE isolation level and rerun this INSERT again.', POSTGRE SQL_ERRNO =
1001;
```

```
END IF;
END;
```

```
$$ DELIMITER ;
```

3. Проверить работоспособность и корректность представленного решения можно выполнением следующего кода: первая попытка выполнить вставку закончится исключительной ситуацией, порождённой в триггере, а вторая попытка пройдёт успешно.

```
SET SESSION TRANSACTION ISOLATION
LEVEL READ COMMITTED; INSERT INTO
`books`
```

```
(`b_name`, `b_year`,
`b_quantity`)
```

```
VALUES ('И ещё одна книга', 1985,
```

```
2);
```

```
SET SESSION TRANSACTION
ISOLATION LEVEL SERIALIZABLE;
INSERT INTO `books` (`b_name`,
```

```
`b_year`, `b_quantity`)
```

```
VALUES ('И ещё одна книга', 1985,
```

```
2);
```

Задание2.

Создать хранимую процедуру, выполняющую подсчёт количества записей в указанной таблице таким образом, чтобы запрос выполнялся максимально быстро (вне зависимости от параллельно выполняемых запросов), даже если в итоге он вернёт не совсем корректные данные.

Порядок выполнения работы:

1. Хранимая процедура должна выполнять подсчёт записей в указанной таблице в транзакции с уровнем изолированности, обеспечивающим минимальную вероятность ожидания завершения конкурирующих транзакций или отдельных операций в них.
2. Идея решения данной задачи состоит в том, чтобы использовать такой уровень изолированности транзакций, который меньше всего подвержен влиянию со стороны блокировок, порождённых другими транзакциями.
3. В Postgre SQL таким уровнем является **READ UNCOMMITTED**, в MS SQL Server — тоже **READ UNCOMMITTED** или **SNAPSHOT** (но **SNAPSHOT** может приводить к дополнительным расходам ресурсов), в Oracle чтение данных всегда происходит в независимом режиме, потому в этой СУБД можно использовать **READ COMMITTED**

(тем более, что **READ UNCOMMITTED** в Oracle нет).

DELIMITER \$\$

**CREATE PROCEDURE COUNT_ROWS(IN table_name VARCHAR(150), OUT
rows_in_table INT)**

BEGIN

SET SESSION TRANSACTION

ISOLATION LEVEL READ UNCOMMITTED;

SET @count_query =

**CONCAT('SELECT COUNT(1) INTO @rows_found
FROM ', table_name);**

**PREPARE count_stmt FROM @count_query;
EXECUTE count_stmt;**

**DEALLOCATE PREPARE count_stmt; SET
rows_in_table := @rows_found; END;**

\$\$ DELIMITER ;

4. Проверить работоспособность и корректность представленного решения
можно выполнением следующего кода.

**CALL COUNT_ROWS('subscriptions', @rows_in_table); SELECT
@rows_in_table;**

5. Проверить работоспособность и корректность представленного решения
можно выполнением следующего кода.

DECLARE @res INT;

**EXECUTE COUNT_ROWS 'subscriptions', @res OUTPUT; SELECT
@res;**

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 2.1 Организация защиты данных в хранилищах

Лабораторное занятие №22

Выполнение резервного копирования и восстановления базы данных из резервной копии.

Цель: Формирование умений выполнять резервное копирование и восстановление базы данных из резервной копии.

Выполнив работу, вы будете уметь:

У 1.4.3 создавать и восстанавливать резервные копии данных

Уо 02.02 использовать современное программное обеспечение, различные цифровые средства для решения профессиональных задач; проявлять культуру информационной безопасности при использовании информационно-коммуникационных технологий.

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: PostgreSQL.

Задание

Резервное копирование (бэкап) — это критически важная процедура, обеспечивающая сохранность данных в случае аппаратных сбоев, программных ошибок или человеческого фактора. PostgreSQL предоставляет несколько инструментов для выполнения этой задачи.

* `pg_dump`: Утилита для создания резервной копии. Она генерирует файл, содержащий SQL-команды (CREATE, INSERT и т.д.), необходимые для восстановления базы данных.

* `pg_restore`: Утилита для восстановления базы из резервной копии, созданной в специальном пользовательском формате (.custom или .backup). Этот формат не является текстовым SQL-скриптом, но позволяет гибко управлять процессом восстановления.

* `psql`: Клиентская утилита для PostgreSQL, которая может использоваться для выполнения SQL-скриптов, в том числе и файлов, созданных `pg_dump`.

Существует два основных подхода к резервному копированию:

1. Логическое копирование (с помощью `pg_dump`): Копируется структура и данные в виде SQL-команд. Гибкий, но медленный для больших БД.

2. Физическое копирование (с помощью файловых системных утилит или `pg_basebackup`): Копируются файлы базы данных на уровне файловой системы. Очень быстрый, но менее гибкий.

В рамках данной работы рассматривается логическое копирование.

2. Задание к лабораторной работе

1. Подготовка: Создать в PostgreSQL тестовую базу данных и наполнить её данными.

2. Создание резервной копии:

* Создать полный бэкап базы данных в виде SQL-скрипта.

* Создать полный бэкап базы данных в пользовательском формате (.backup).

3. Восстановление данных:

* Удалить исходную базу данных.

* Восстановить базу данных из SQL-скрипта в новую базу.

* Восстановить базу данных из файла в пользовательском формате в другую новую базу.

4. Верификация: Проверить, что восстановленные базы данных идентичны исходной по структуре и содержанию.

Порядок выполнения работы

Примечание: Все команды выполняются в терминале (командной строке) вашей операционной системы, а не в клиенте psql.

Шаг 1. Подготовка исходной базы данных

1.1. Создайте новую базу данных для экспериментов:

```
createdb lab_backup_db
```

1.2. Подключитесь к ней через psql и создайте простую таблицу с данными:

```
CREATE TABLE products (  
    id SERIAL PRIMARY KEY,  
    name VARCHAR(100),  
    price NUMERIC(10,2)  
);
```

```
INSERT INTO products (name, price) VALUES  
( 'Ноутбук', 65000.00),  
( 'Смартфон', 35000.50),  
( 'Наушники', 5500.00);
```

1.3. Проверьте данные:

```
SELECT * FROM products;
```

Вы должны увидеть три строки.

Шаг 2. Создание резервной копии (бэкапа)

2.1. Создание бэкапа в виде SQL-скрипта:

Этот метод создает текстовый файл с SQL-командами. Он универсален и может быть выполнен на любой машине с установленным psql.

```
pg_dump -U postgres -F p -f lab_backup_db.sql lab_backup_db  
* -U postgres: Указывает имя пользователя для подключения к СУБД (стандартный суперпользователь).  
* -F p: Указывает формат вывода как "plain" (обычный текст, SQL-скрипт).  
* -f lab_backup_db.sql: Указывает имя выходного файла.  
* lab_backup_db: Имя базы данных, которую нужно забэкапить.
```

Создать нового пользователя СУБД и предоставить ему полный набор прав на базу данных «Библиотека».

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 2.1 Организация защиты данных в хранилищах

Лабораторное занятие №23

Управление привилегиями и доступом к данным.

Цель: Формирование умений выполнять резервное копирование и восстановление базы данных из резервной копии.

Выполнив работу, вы будете уметь:

У 1.4.1 создавать пользователей и назначать права доступа

Уо 04.03 использовать навыки управления проектами в распределении ресурсов и формировании графика выполнения задач

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: Postgre SQL.

Задание

Общие сведения

Безопасность данных — одна из ключевых задач администратора баз данных. В многопользовательской среде необходимо гарантировать, что каждый пользователь имеет доступ только к тем данным и операциям, которые необходимы ему для выполнения своих обязанностей. Этот принцип называется принципом наименьших привилегий.

В PostgreSQL управление доступом строится вокруг следующих понятий:

- * Роль (Role): Абстрактная сущность, которая может владеть объектами БД и иметь права. Роль может быть назначена пользователю или другой роли. По сути, "пользователь" в PostgreSQL — это роль с правом входа (LOGIN).
- * Привилегия (Privilege/Permission): Разрешение на выполнение определенного действия над объектом БД (например, SELECT, INSERT, UPDATE, DELETE).
- * Объект (Object): Таблица, представление, последовательность, функция, схема или сама база данных.

Управление доступом осуществляется с помощью двух основных команд:

- * GRANT: Предоставляет привилегии.
- * REVOKE: Отзывает ранее предоставленные привилегии.

2. Задание к лабораторной работе

1. Создание ролей:

- * Создать суперпользователя-администратора.
- * Создать обычного пользователя с правом входа в систему.

2. Настройка прав на уровне схемы:

- * Создать новую схему.
- * Настроить права так, чтобы другие пользователи не могли видеть объекты в этой схеме по умолчанию.

3. Управление правами на таблицу:

- * Создать таблицу внутри новой схемы от имени администратора.
- * Предоставить обычному пользователю право на чтение (SELECT) и добавление (INSERT) данных в эту таблицу.
- * Проверить, что пользователь не может изменять (UPDATE) или удалять (DELETE) данные.

4. Делегирование прав:

- * Предоставить обычному пользователю право передавать свои привилегии другим ролям (WITH GRANT OPTION).

5. Отзыв прав:

- * Отозвать у пользователя право на вставку данных.

Порядок выполнения работы

Примечание: Выполняйте команды последовательно, переключаясь между пользователями.

Шаг 1. Создание ролей и подключение

Сначала работаем от имени суперпользователя postgres.

1.1. Создайте нового администратора и обычного пользователя.

```
-- Создаем роль-администратора без права входа (для владения объектами)  
CREATE ROLE db_admin WITH CREATEDB CREATEROLE;
```

```
-- Создаем обычного пользователя с правом входа и паролем  
CREATE USER analyst WITH PASSWORD 'password123' LOGIN;
```

1.2. Подключитесь к базе данных под новым пользователем analyst, чтобы убедиться, что он существует.

```
psql -U analyst -d postgres
```

Вы должны успешно подключиться.

Шаг 2. Создание схемы и настройка прав по умолчанию

Вернитесь в сессию суперпользователя postgres.

2.1. Создайте новую схему, владельцем которой будет db_admin.

```
CREATE SCHEMA finance AUTHORIZATION db_admin;
```

2.2. Измените права доступа к этой схеме по умолчанию, чтобы никто, кроме владельца, не мог использовать её.

```
REVOKE ALL ON SCHEMA finance FROM PUBLIC;
```

(Команда REVOKE ... FROM PUBLIC отменяет все стандартные права, которые выдаются всем пользователям по умолчанию).

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 2.1 Организация защиты данных в хранилищах

Лабораторное занятие №24

Реализация доступа пользователей к базе данных.

Цель: Формирование умений создавать пользователей и роли в базе данных, назначать и запрещать права.

Выполнив работу, вы будете уметь:

У 1.5.2

Уо 04.03

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: PostgreSQL.

Задание

Общие сведения

Реализация доступа — это комплексный процесс, который включает не только создание учетной записи, но и её интеграцию с конкретной базой данных и её объектами. В реальных системах доступ предоставляется по принципу "минимальных привилегий": пользователь получает ровно те права, которые необходимы для выполнения его задач, и не более того.

Процесс обычно состоит из следующих этапов:

1. Создание роли (пользователя) в системе аутентификации СУБД.
2. Выделение ресурсов: Создание базы данных и назначение владельца.
3. Предоставление прав доступа: Разрешение пользователю подключаться к базе и работать с ее объектами (схемами, таблицами).
4. Настройка клиентского подключения: Проверка доступа с помощью различных инструментов (psql, графические клиенты) или программного кода.

В этой работе мы пройдем все эти шаги, создав нового пользователя и настроив ему доступ к специально созданной для него базе данных.

2. Задание к лабораторной работе

1. Создание инфраструктуры:

* Создать новую базу данных `company_db`.

* Создать нового пользователя (роль) `hr_manager` с паролем.

2. Настройка прав доступа:

* Назначить пользователя `hr_manager` владельцем базы данных `company_db`.

* Подключиться к базе данных под новой учетной записью и создать схему `employees`.

3. Разграничение доступа (дополнительно):

* Создать второго пользователя `intern`.

* Предоставить пользователю `intern` право на чтение данных из схемы `employees`, но запретить добавление и изменение данных.

4. Проверка доступа:

- * Убедиться, что оба пользователя могут успешно подключиться к своей базе данных.
- * Проверить, что их возможности соответствуют выданным правам (например, intern может делать SELECT, но не может INSERT).

Порядок выполнения работы

Шаг 1. Создание базы данных и основного пользователя

Все команды этого шага выполняются от имени суперпользователя (обычно postgres), например, через терминал командой `sudo -u postgres psql`.

1.1. Создайте новую базу данных:

```
CREATE DATABASE company_db;
```

1.2. Создайте нового пользователя с надежным паролем:

```
CREATE USER hr_manager WITH PASSWORD 'StrongP@ssw0rd!';
```

Примечание: Хорошей практикой является использование флага LOGIN, но команда CREATE USER делает это по умолчанию.

Шаг 2. Настройка прав доступа к базе данных

2.1. Сделайте пользователя hr_manager владельцем базы данных company_db. Это автоматически даст ему полные права в рамках этой БД.

```
ALTER DATABASE company_db OWNER TO hr_manager;
```

2.2. Теперь переключимся на нашу новую базу данных и создадим структуру. Для этого нужно выйти из текущей сессии (\q) и подключиться заново, используя новые данные.

```
psql -U hr_manager -d company_db
```

Вы вошли в систему как hr_manager.

2.3. Внутри базы company_db создайте схему и таблицу для дальнейшей работы.

```
CREATE SCHEMA employees;
```

```
CREATE TABLE employees.staff (  
    id SERIAL PRIMARY KEY,  
    full_name VARCHAR(150),  
    position VARCHAR(100)  
);
```

-- Добавим тестовые данные

```
INSERT INTO employees.staff (full_name, position) VALUES  
( 'Иван Иванов', 'Разработчик'),  
( 'Анна Петрова', 'Менеджер');
```

Шаг 3. Создание дополнительного пользователя и разграничение прав

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 2.1 Организация защиты данных в хранилищах

Лабораторное занятие №25

Мониторинг безопасности работы с базами данных.

Цель: Формирование умений осуществлять мониторинг безопасности работы с базами данных

Выполнив работу, вы будете уметь:

У 1.5.3 устанавливать и настраивать механизмы аутентификации и авторизации пользователей;

У 04.03 использовать навыки управления проектами в распределении ресурсов и формировании графика выполнения задач

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: PostgreSQL.

Задание

Общие сведения

Мониторинг безопасности — это непрерывный процесс наблюдения за действиями в системе для обнаружения аномалий, предотвращения утечек данных и расследования инцидентов. В контексте СУБД мониторинг позволяет ответить на вопросы:

- * Кто подключался к базе данных и когда?
- * Были ли попытки несанкционированного доступа (неудачные входы)?
- * Какие пользователи изменяли структуру базы данных (CREATE, ALTER, DROP)?
- * К каким данным обращались пользователи?

Основным инструментом для решения этих задач в PostgreSQL является журналирование (logging). Настроив соответствующие параметры в файле конфигурации, можно заставить сервер записывать практически все события, происходящие в системе, в специальный лог-файл.

2. Задание к лабораторной работе

1. Настройка аудита:

- * Включить журналирование всех успешных и неуспешных попыток авторизации.
- * Включить журналирование всех DDL-команд (команд изменения структуры БД).

2. Проведение тестовых действий:

- * Выполнить серию действий от имени разных пользователей: успешное подключение, попытка подключения с неверным паролем, создание таблицы, изменение таблицы, удаление таблицы.

3. Анализ журнала:

- * Найти в журнале событий записи, соответствующие выполненным действиям.

* Проанализировать содержимое записей: определить время события, пользователя, IP-адрес клиента и тип операции.

4. (Дополнительно) Использовать расширение pgaudit для детального аудита операций с данными (SELECT, INSERT, UPDATE, DELETE).

Порядок выполнения работы

Шаг 1. Подготовка и настройка логирования

Для выполнения этой части работы вам потребуются права администратора ОС для редактирования конфигурационного файла PostgreSQL.

1.1. Откройте файл конфигурации postgresql.conf. Его расположение зависит от вашей операционной системы и способа установки (например, /etc/postgresql/16/main/postgresql.conf или C:\Program Files\PostgreSQL\<версия>\data\postgresql.conf).

1.2. Найдите в файле следующие параметры и установите их значения:

```
# Управление общими настройками логирования
logging_collector = on      # Включает сборщик логов (обязательно)
log_destination = 'stderr' # Куда писать логи (стандартная настройка)
log_directory = 'pg_log'   # Папка для хранения файлов логов (относительно каталога данных)
log_filename = 'postgresql-%Y-%m-%d_%H%M%S.log' # Шаблон имени файла лога
```

```
# Что именно логировать
```

```
log_connections = on      # Логировать все попытки подключения к серверу
log_disconnections = on   # Логировать завершение сеансов
log_duration = on         # Логировать длительность выполнения всех SQL-запросов
log_statement = 'ddl'      # Логировать все DDL-команды (CREATE, ALTER, DROP)
```

1.3. Сохраните файл и перезапустите службу PostgreSQL, чтобы изменения вступили в силу.

```
sudo systemctl restart postgresql
```

Шаг 2. Проведение тестовых действий

Теперь выполним действия, которые должны быть отражены в логах.

2.1. Успешное подключение:

Подключитесь к вашей базе данных как суперпользователь:

```
psql -U postgres -d company_db
Сразу после этого выйдите из сессии (\q).
```

2.2. Неуспешная попытка подключения:

Попробуйте подключиться с неправильным паролем:

```
psql -U hr_manager -d company_db -W
Введите любой неверный пароль. Соединение не будет установлено.
```

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 2.1 Организация защиты данных в хранилищах

Лабораторное занятие №26

Установка приоритетов.

Цель: Формирование умений устанавливать приоритеты в базе данных для избегания перегрузки

Выполнив работу, вы будете уметь:

У 1.5.3 устанавливать и настраивать механизмы аутентификации и авторизации пользователей;

Уо 02.02 использовать современное программное обеспечение, различные цифровые средства для решения профессиональных задач; проявлять культуру информационной безопасности при использовании информационно-коммуникационных технологий

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: Postgre SQL.

Задание

Общие сведения

В многопользовательской среде несколько запросов могут выполняться одновременно, конкурируя за одни и те же ресурсы: процессорное время (CPU), дисковый ввод-вывод (I/O) и оперативную память. По умолчанию PostgreSQL использует стратегию "справедливого" распределения ресурсов, где все активные запросы имеют теоретически равный приоритет. Однако для обеспечения стабильной работы системы может потребоваться вмешаться в этот процесс:

* Понизить приоритет "тяжелых", долго выполняющихся или не критичных запросов, которые потребляют слишком много ресурсов.

* Повысить приоритет критически важных операций (например, отчетов для руководства или платежей).

Важно понимать, что прямое управление приоритетом отдельного SQL-запроса (как это делается в ОС) в PostgreSQL отсутствует. Вместо этого используются механизмы управления процессами, которые эти запросы выполняют.

2. Задание к лабораторной работе

1. Создание нагрузки: Запустить два длительных запроса от имени разных пользователей. Один запрос будет интенсивно использовать CPU, другой — выполнять много дисковых операций.

2. Мониторинг: Определить идентификаторы процессов (PID) этих запросов.

3. Управление приоритетом:

* Попытайтесь прервать один из запросов (CANCEL).

* Принудительно завершить другой запрос (TERMINATE).

4. Анализ: Проанализировать результат и убедиться, что управление процессом привело к остановке соответствующего SQL-запроса.

Порядок выполнения работы

Шаг 1. Подготовка тестовой среды

Для имитации длительной операции, нагружающей CPU, идеально подходит функция `generate_series()`. Для создания нагрузки на I/O мы будем создавать большую таблицу и заполнять её данными.

Подключитесь к вашей базе данных как суперпользователь (`postgres`) или любой пользователь с достаточными правами.

Шаг 2. Создание нагрузки (длительных запросов)

Откройте два отдельных окна терминала.

Окно 1 (Нагрузка на CPU):

Запустите следующий запрос. Он будет вычислять сумму логарифмов для большого диапазона чисел, эффективно загружая одно ядро процессора.

```
SELECT SUM(LOG(g)) FROM generate_series(1, 100000000) AS g;
```

Окно 2 (Нагрузка на I/O):

Сначала создайте большую таблицу, а затем начните вставку в неё данных. Этот процесс создаст значительную нагрузку на запись на диск.

```
-- Создаем таблицу
```

```
CREATE TABLE big_table (  
    id SERIAL PRIMARY KEY,  
    data TEXT,  
    filler CHAR(1000) -- Добавим поле для увеличения размера строки  
);
```

```
-- Начинаем вставку большого объема данных
```

```
INSERT INTO big_table (data, filler)  
SELECT 'some_data', REPEAT('x', 1000)  
FROM generate_series(1, 500000);
```

Шаг 3. Мониторинг и определение PID запросов

Пока запросы выполняются, откройте третье окно терминала и подключитесь к БД. Выполните запрос к системной таблице `pg_stat_activity`, чтобы увидеть текущие активные сессии.

```
SELECT pid, username, datname, state, query  
FROM pg_stat_activity  
WHERE state = 'active' AND query NOT LIKE '%pg_stat_activity%';
```

Этот запрос покажет вам список всех активных запросов, их состояние и, самое главное, их PID (столбец `pid`). Найдите в этом списке ваши два запущенных запроса и запишите их идентификаторы.

Форма представления результата:

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 2.1 Организация защиты данных в хранилищах

Лабораторное занятие №27

Реализация механизмов для защиты баз данных в PostgreSQL.

Цель: Формирование умений устанавливать механизмы для защиты баз данных

Выполнив работу, вы будете уметь:

У 1.5.1 использовать механизмы аудита для отслеживания доступа к данным;

Уо 02.02 использовать современное программное обеспечение, различные цифровые средства для решения профессиональных задач; проявлять культуру информационной безопасности при использовании информационно-коммуникационных технологий.

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: PostgreSQL.

Задание

Теоретические основы

Управление пользователями и ролями (Authentication and Authorization)

PostgreSQL использует систему ролей для управления доступом. Роль может быть пользователем или группой.

Аутентификация: Проверка подлинности пользователя при подключении к серверу. Настраивается в файле pg_hba.conf.

Авторизация: Определение прав доступа уже аутентифицированного пользователя к объектам БД.

Принцип наименьших привилегий

Пользователю должны предоставляться только те права, которые необходимы для выполнения его задач.

Шифрование соединений

Для защиты передаваемых данных от перехвата используется протокол TLS/SSL.

Аудит

Процесс регистрации событий безопасности (вход в систему, выполнение DDL/DML команд) для последующего анализа.

Порядок выполнения работы.

В рамках практической части необходимо выполнить следующие шаги. Все команды выполняются от имени суперпользователя (postgres) или другого пользователя с достаточными правами.

Задание 1: Создание ролей и разграничение доступа

Цель: Создать администратора БД, разработчика и аналитика с разными уровнями доступа.

Подключаемся к СУБД:

```
psql -U postgres
```

Создаем роли:

Администратор БД (может управлять объектами):
`CREATE ROLE db_admin WITH LOGIN
PASSWORD 'StrongP@ssw0rd!' NOSUPERUSER INHERIT NOCREATEDB NOCREATEROLE;`

Группа "Разработчики" (будем выдавать права группе):
`CREATE ROLE developers_group
NOLOGIN;`

Пользователь-разработчик:
`CREATE ROLE developer_user WITH LOGIN PASSWORD
'DevP@ssw0rd!' NOSUPERUSER INHERIT;`

-- Добавляем пользователя в группу
`GRANT developers_group TO developer_user;`

Создание тестовой базы данных и схемы:

```
CREATE DATABASE company_db OWNER db_admin;  
\c company_db  
CREATE SCHEMA hr AUTHORIZATION db_admin;
```

Разграничение доступа:

Выдадим администратору полные права на схему:
`GRANT ALL PRIVILEGES ON SCHEMA hr TO
db_admin;`

Выдадим группе разработчиков право на создание таблиц и других объектов в схеме:
`GRANT
USAGE, CREATE ON SCHEMA hr TO developers_group;`

Предоставим возможность работать с данными в существующих таблицах (если они есть):
`GRANT
SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA hr TO developers_group;`

-- Чтобы новые таблицы автоматически наследовали эти права:

```
ALTER DEFAULT PRIVILEGES IN SCHEMA hr GRANT SELECT, INSERT, UPDATE, DELETE  
ON TABLES TO developers_group;
```

Проверка:

Подключитесь под пользователем `developer_user` и попробуйте создать таблицу в схеме `hr`. Затем попытайтесь удалить всю базу данных — эта операция должна завершиться ошибкой из-за отсутствия прав.

Задание 2: Настройка шифрования соединений (SSL/TLS)

Цель: Обеспечить защищенное соединение клиентов с сервером PostgreSQL.

Генерация самоподписанных сертификатов (для теста):

На сервере выполните команды:

```
# Генерируем закрытый ключ
openssl genrsa -out server.key 2048
# Генерируем сертификат, используя ключ
openssl req -new -x509 -key server.key -out server.crt -days 365
```

Настройка PostgreSQL:

В файле postgresql.conf укажите пути к сертификатам:

```
ssl = on
ssl_cert_file = 'server.crt'
ssl_key_file = 'server.key'
```

Перезапустите службу PostgreSQL.

Настройка pg_hba.conf:

Для хостов, требующих SSL-соединения, добавьте строку:

```
hostssl all all 192.168.1.0/24 md5
```

Эта строка разрешает подключения по SSL для всех пользователей и баз данных из сети 192.168.1.0/24 с проверкой пароля.

Проверка:

Попробуйте подключиться к базе данных с клиента, указав требование SSL:

```
psql "host=localhost user=db_admin dbname=company_db sslmode=require"
```

Если подключение успешно, значит, шифрование работает.

Задание 3: Включение аудита

Цель: Настроить логирование операций создания, изменения и удаления объектов (DDL).

Настройка параметров в postgresql.conf:

Найдите и установите следующие параметры:

```
log_statement = 'ddl'    # Логировать все команды DDL (CREATE, ALTER, DROP)
log_destination = 'csvlog' # Формат лог-файлов (можно оставить 'stderr')
logging_collector = on   # Включить сборщик логов
log_directory = 'pg_log' # Директория для хранения логов
log_filename = 'audit-%Y-%m-%d.log' # Шаблон именования файлов
log_rotation_age = 1d    # Поворачивать файл раз в день
```

Перезапуск PostgreSQL:

После внесения изменений перезапустите сервер.

Проверка:

Выполните любую команду DDL, например:

```
CREATE TABLE hr.test_audit (id serial);
```

Проверьте содержимое файла лога в директории `pg_log`. Там должна появиться запись об этой операции с указанием времени, пользователя и IP-адреса.

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено.

Тема 2.1 Организация защиты данных в хранилищах

Лабораторное занятие №28

Мониторинг сетевого трафика.

Цель: Формирование умений осуществлять мониторинг сетевого трафика в PostgreSQL

Выполнив работу, вы будете уметь:

У 1.5.1 использовать механизмы аудита для отслеживания доступа к данным;

Уо 02.02 использовать современное программное обеспечение, различные цифровые средства для решения профессиональных задач; проявлять культуру информационной безопасности при использовании информационно-коммуникационных технологий.

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: PostgreSQL.

Задание

Общие сведения

Защита базы данных — это многоуровневый процесс, направленный на обеспечение конфиденциальности, целостности и доступности данных. В PostgreSQL реализован набор инструментов, позволяющих выстроить эшелонированную оборону:

* ****Сетевой уровень (*pg_hba.conf*):**** Первый рубеж обороны. Определяет, кто может подключаться к серверу, откуда и каким способом проходить аутентификацию.

* ****Уровень объектов (Привилегии *GRANT/REVOKE*):**** Позволяет детально разграничивать доступ к конкретным таблицам, представлениям, функциям уже внутри самой БД.

- * ****Шифрование соединений (*SSL*):**** Защищает данные от перехвата при передаче по сети между клиентом и сервером.
- * ****Маскировка данных (Представления *VIEW*):**** Скрывает конфиденциальные столбцы от пользователей, которым не нужен полный доступ.
- * ****Аудит (Логирование):**** Позволяет отслеживать действия пользователей для выявления подозрительной активности и расследования инцидентов.

В рамках данной работы вы последовательно настроите каждый из этих уровней защиты для учебной базы данных.

2. Задание к лабораторной работе

1. ****Настройка сетевого доступа:****

- * Изменить файл `pg_hba.conf` так, чтобы локальные подключения через Unix-сокеты требовали пароля (`md5`), а удаленное подключение с определенного IP-адреса было запрещено.

2. ****Реализация принципа наименьших привилегий:****

- * Создать пользователя `report_user`.

- * Создать представление, которое скрывает конфиденциальный столбец (например, `email`) в таблице клиентов.

- * Предоставить пользователю `report_user` доступ только к этому представлению, но не к исходной таблице.

3. ****Обеспечение шифрования трафика:****

- * Настроить PostgreSQL для использования SSL-соединений.

4. ****Включение базового аудита:****

- * Настроить параметры логирования для записи всех попыток подключения и всех DDL-команд.

Порядок выполнения работы

Шаг 1. Ограничение сетевого доступа (файл `pg_hba.conf`)

Файл `pg_hba.conf` (Host-Based Authentication) управляет правилами доступа к серверу.

1.1. Откройте файл `pg_hba.conf`, который находится в каталоге с данными PostgreSQL (обычно рядом с `postgresql.conf`).

1.2. Найдите строки, относящиеся к локальным подключениям (`local`) и IPv4-подключениям (`host`). Они могут выглядеть примерно так:

```

...
TYPE DATABASE      USER      ADDRESS      METHOD
local all          all                peer
host  all          all      127.0.0.1/32  md5
host  all          all      ::1/128       md5
...

```

1.3. Модифицируйте правила согласно заданию:

- * Для локальных подключений через Unix-сокеты измените метод с `peer` на `md5`. Это заставит даже локальных пользователей вводить пароль.

- * Добавьте новое правило, запрещающее все подключения с определенного IP-адреса (например, `192.168.1.100`). Используйте метод `reject`.

Пример конфигурации:

```

...

```

Требовать пароль для локальных подключений

local all all md5

Разрешить подключения с localhost

host all all 127.0.0.1/32 md5

Запретить подключения с конкретного IP

host all all 192.168.1.100/32 reject

Разрешить остальные IPv6 подключения

host all all ::1/128

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания. Оценка «неудовлетворительно» ставится, если задание не выполнено

МДК.01.03 Конфигурирование и программирование в системе 1С

Тема 3.1. Программирование в среде 1С Предприятие

Лабораторное занятие №29

Установка системы 1С:ПРЕДПРИЯТИЕ. Основные принципы работы с платформой.

Цель: Формирование умений устанавливать систему 1С:ПРЕДПРИЯТИЕ, осуществлять вход в режиме администратора и пользователя.

Выполнив работу, вы будете уметь:

У 1.3.1 разрабатывать объекты базы данных, такие как таблицы, индексы и связи между ними;

Уо 04.03 использовать навыки управления проектами в распределении ресурсов и формировании графика выполнения задач

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: 1С:ПРЕДПРИЯТИЕ.

Задание

Общие сведения

Система программ «1С:Предприятие» является технологической платформой, на базе которой создаются и работают прикладные решения (конфигурации), предназначенные для автоматизации различных видов деятельности предприятий.

Ключевые компоненты системы:

- * **Платформа «1С:Предприятие»**: Набор программных средств, "движок", который выполняет код конфигурации, управляет базами данных, обеспечивает пользовательский интерфейс. Платформа не работает сама по себе, ей нужна конфигурация.
- * **Конфигурация**: Готовое прикладное решение (например, «1С:Бухгалтерия 8», «1С:Управление торговлей 8»). Это набор алгоритмов, форм, отчетов, реализующих конкретную методологию учета.
- * **Информационная база (ИБ)**: Совокупность данных, которые платформа использует во время работы. ИБ включает в себя структуру таблиц (метаданные) и сами данные (документы, справочники). Каждая ИБ обязательно привязана к конкретной конфигурации.

Архитектура может быть двух видов:

1. **Файл-серверный вариант**: Файлы базы данных хранятся в виде файла на диске (*.1CD). Подходит для небольших рабочих групп (до ~10-15 пользователей).
2. **Клиент-серверный вариант**: Используется трехзвенная архитектура: Клиентское приложение -> Сервер «1С:Предприятия» -> Система управления базами данных (СУБД, например, PostgreSQL или MS SQL Server). Этот вариант предназначен для средних и крупных внедрений.

В рамках данной работы мы рассмотрим установку платформы и создание файловой информационной базы.

2. Задание к лабораторной работе

1. Установить на компьютер платформу «1С:Предприятие».
2. Запустить «Толстый клиент» (*1cv8.exe) и создать новую файловую информационную базу из шаблона типовой конфигурации (например, «Бухгалтерия предприятия»).
3. Выполнить первоначальное заполнение информационной базы (создать организацию).
4. Изучить основные элементы пользовательского интерфейса: панель разделов, панель навигации, панель действий.
5. Создать несколько элементов справочника «Номенклатура» (товары/услуги).
6. Сформировать один из стандартных отчетов.

Важное замечание: Для выполнения этой работы требуется дистрибутив технологической платформы «1С:Предприятие 8.x» и дистрибутив конфигурации (например, «Бухгалтерия предприятия»). Обычно они поставляются в виде единого установочного пакета.

Порядок выполнения работы

Шаг 1. Установка платформы «1С:Предприятие»

- 1.1. Запустите установочный файл `setup.exe`.
- 1.2. Следуйте инструкциям мастера установки:
 - * Выберите **«Установка 1С:Предприятия»**.
- 1.3. На этапе выбора компонентов выберите оба варианта клиента:
 - * **«1С:Предприятие»** (Толстый клиент) — для полноценной работы с конфигуратором и всеми функциями.
 - * **«1С:Предприятие — Тонкий клиент»** — для быстрой работы через сеть.
- 1.4. Дождитесь окончания установки.

Шаг 2. Создание новой информационной базы

- 2.1. Найдите в меню «Пуск» папку «1С Предприятие» и запустите ярлык ****«1С:Предприятие»**** (это запуск толстого клиента).
- 2.2. Откроется окно «Запуск 1С:Предприятия». Нажмите кнопку ****«Добавить...»****.
- 2.3. В окне добавления выберите пункт ****«Создание новой информационной базы»**** и нажмите «Далее >».
- 2.4. На следующем экране выберите ****«Создание информационной базы без конфигурации... (для разработки новой конфигурации или загрузки**

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания. Оценка «неудовлетворительно» ставится, если задание не выполнено

Тема 3.1. Программирование в среде 1С Предприятие

Лабораторное занятие №30

Разработка конфигурации для организации хранения информации о студентах и изучаемых ими предметах.

Цель: Формирование умений разработки конфигурации для организации хранения информации о студентах и изучаемых ими предметах.

Выполнив работу, вы будете уметь:

У 1.3.1 разрабатывать объекты базы данных, такие как таблицы, индексы и связи между ними;

Уо 04.03 использовать навыки управления проектами в распределении ресурсов и формировании графика выполнения задач

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: 1С:ПРЕДПРИЯТИЕ.

Задание

Общие сведения

В отличие от предыдущих лабораторных работ, где вы использовали готовые типовые конфигурации (как «Бухгалтерия»), эта работа предполагает создание собственной конфигурации с нуля. Это позволяет понять, как устроена система изнутри.

Ключевые понятия:

* Конфигуратор: Среда разработки, входящая в платформу «1С:Предприятие», предназначенная для создания и изменения структуры конфигурации.

* Метаданные («Дерево конфигурации»): Это «схема» или «чертеж» вашей базы данных. Здесь вы определяете, какие таблицы (справочники, документы) будут в базе, какие у них будут поля и как они будут связаны между собой.

* Справочник: Объект метаданных, предназначенный для хранения списков однотипных данных (например, список студентов, список предметов). Аналог таблицы-справочника в реляционной БД.

* Документ: Основной объект для регистрации хозяйственных операций или событий. Документ всегда имеет дату и время. В нашей задаче документом будет событие «Студент записался на курс».

* Регистр сведений: Объект, предназначенный для хранения информации, развернутой по комбинации измерений. Идеально подходит для связи «многие-ко-многим» (один студент может изучать много предметов, один предмет могут изучать много студентов).

2. Задание к лабораторной работе

Необходимо разработать конфигурацию, которая решает следующую задачу:

Хранить информацию о студентах, учебных предметах и связывать их друг с другом, фиксируя факт изучения предмета конкретным студентом.

Для этого необходимо:

1. Создать справочник Студенты для хранения информации о ФИО и дате рождения.
2. Создать справочник Предметы для хранения названий дисциплин.
3. Создать документ ЗаписьНаКурс, который будет связывать одного студента с одним предметом.
4. Создать регистр сведений УчебныйПлан для накопления всех записей о том, кто какой предмет изучает.
5. Настроить логику документа так, чтобы при его проведении данные записывались в регистр.
6. Проверить работоспособность созданного механизма.

Порядок выполнения работы

Шаг 1. Создание новой информационной базы

- 1.1. Запустите «1С:Предприятие» из меню «Пуск».
- 1.2. В окне запуска нажмите кнопку «Добавить...».
- 1.3. Выберите пункт «Создание новой информационной базы».
- 1.4. Выберите «Создание информационной базы без конфигурации...» (пустой базы для разработки).
- 1.5. Укажите имя (База_Студенты) и расположение базы. Нажмите «Готово».
- 1.6. После запуска базы откроется окно с предупреждением об отсутствии конфигурации. Закройте его. Система готова к открытию в режиме Конфигуратора.

Шаг 2. Проектирование структуры данных в Конфигураторе

Закройте информационную базу и запустите «1С:Предприятие» (Конфигуратор).

2.1. Создание справочников

2.1.1. В дереве конфигурации правой кнопкой мыши -> Добавить -> Справочник. Назовите его Студенты.

2.1.2. На закладке «Данные» создайте реквизиты:

* ФИО (Тип: Строка, Длина: 100).

* ДатаРождения (Тип: Дата).

2.1.3. Повторите процесс для второго справочника. Правая кнопка мыши -> Добавить -> Справочник. Назовите его Предметы.

2.1.4. На закладке «Данные» создайте реквизит:

* Название (Тип: Строка, Длина: 150).

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания. Оценка «неудовлетворительно» ставится, если задание не выполнено

Тема 3.1. Программирование в среде 1С Предприятие

Лабораторное занятие №31

Разработка информационной системы для хранения информации о сотрудниках предприятия.

Цель: Формирование умений разработки информационной системы для хранения информации о сотрудниках предприятия.

Выполнив работу, вы будете уметь:

У 1.3.1 разрабатывать объекты базы данных, такие как таблицы, индексы и связи между ними;

Уо 04.03 использовать навыки управления проектами в распределении ресурсов и формировании графика выполнения задач

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: 1С:ПРЕДПРИЯТИЕ.

Задание

Общие сведения

Информационная система по учету персонала должна не просто хранить список сотрудников, но и отражать динамику их состояния в компании: прием на работу, перевод на другую должность, изменение оклада, увольнение. Это накладывает определенные требования к структуре данных:

* Справочник "Сотрудники": Должен быть иерархическим, чтобы можно было группировать сотрудников по отделам или должностям.

* Документы: Каждое кадровое событие оформляется отдельным документом. Документ является первичным источником информации.

* Регистр сведений "Кадровая история": Служит для накопления данных о том, кем, когда и на каких условиях работал сотрудник. Регистр позволяет получить актуальное состояние сотрудника на любую дату и проследить всю историю его изменений.

В отличие от предыдущей работы, здесь мы будем работать с данными, которые изменяются во времени, что требует реализации механизма получения актуальных данных на текущую дату.

2. Задание к лабораторной работе

Необходимо разработать конфигурацию, позволяющую вести учет кадровой информации предприятия. Система должна поддерживать следующие операции:

1. Хранение списка сотрудников с основными реквизитами (ФИО, дата рождения).
2. Оформление приема сотрудника на работу с указанием должности, подразделения и оклада.
3. Оформление увольнения сотрудника с указанием причины и даты.
4. Автоматическое ведение истории кадровых изменений в регистре сведений.
5. Получение актуальной информации о сотруднике (текущая должность, оклад).

Порядок выполнения работы

Примечание: Работа выполняется в режиме Конфигуратора ("1С:Предприятие (Конфигуратор)") над той же базой, что и предыдущая работа, либо в новой базе.

Шаг 1. Проектирование структуры метаданных

1.1. Создание справочника "Сотрудники"

1.1. В дереве конфигурации (Общие -> Справочники) создайте новый объект с именем Сотрудники.

1.2. На закладке "Иерархия" установите флажок "Иерархический справочник" и выберите вид иерархии "Иерархия элементов и групп". Это позволит создавать группы (например, "Администрация", "Бухгалтерия") и помещать в них элементы-сотрудников.

1.3. На закладке "Данные" добавьте реквизиты:

* ДатаРождения (Тип: Дата)

* ФизическоеЛицо (Тип: СправочникСсылка.ФизическиеЛица) — это стандартная ссылка для связи с общими данными человека.

1.2. Создание документов "ПриемНаРаботу" и "Увольнение"

1.4. Создайте документ "ПриемНаРаботу".

1.5. На закладке "Данные" добавьте реквизиты шапки:

* Сотрудник (Тип: СправочникСсылка.Сотрудники)

* Подразделение (Тип: Строка, длина 100)

* Должность (Тип: Строка, длина 100)

* Оклад (Тип: Число, точность 2)

1.6. Аналогично создайте документ "Увольнение".

1.7. На закладке "Данные" добавьте реквизиты шапки:

* Сотрудник (Тип: СправочникСсылка.Сотрудники)

* ПричинаУвольнения (Тип: Строка, длина 200)

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания. Оценка «неудовлетворительно» ставится, если задание не выполнено

Тема 3.1. Программирование в среде 1С Предприятие

Лабораторное занятие №32

Разработка конфигурации для учета посещений клиентами экскурсий.

Цель: Формирование умений разработки конфигурации для учета посещений клиентами экскурсий.

Выполнив работу, вы будете уметь:

У 1.3.1 разрабатывать объекты базы данных, такие как таблицы, индексы и связи между ними;

Уо 04.03 использовать навыки управления проектами в распределении ресурсов и формировании графика выполнения задач.

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: 1С:ПРЕДПРИЯТИЕ.

Задание

Общие сведения

Данная задача является классическим примером автоматизации торговой или сервисной деятельности. Ключевым моментом здесь является не просто хранение информации о факте продажи экскурсии, а учет этой операции. Учет подразумевает регистрацию количественных и суммовых показателей (сколько билетов продано, на какую сумму), что позволяет формировать отчеты и анализировать деятельность предприятия.

Для реализации учета в «1С:Предприятие» используются регистры накопления. Они предназначены для хранения истории изменений остатков или оборотов по каким-либо ресурсам (например, количеству и сумме). В нашей задаче регистр накопления будет хранить информацию об уменьшении остатка доступных мест на экскурсию и увеличении объема продаж при каждом проведении документа "Продажа экскурсии".

2. Задание к лабораторной работе

Необходимо разработать конфигурацию, позволяющую вести учет продаж экскурсионных туров. Система должна поддерживать следующие функции:

1. Хранение каталога экскурсий (название, описание, стоимость, количество мест).
2. Хранение списка клиентов.
3. Регистрация факта продажи экскурсии клиенту через документ.
4. Автоматическое списание проданных мест из общего количества доступных мест на экскурсию.
5. Формирование отчета, показывающего остатки экскурсий и объем продаж за период.

Порядок выполнения работы

Примечание: Работа выполняется в режиме Конфигуратора ("1С:Предприятие (Конфигуратор)")

Шаг 1. Проектирование структуры метаданных

1.1. Создание справочников

* Справочник "Экскурсии"

* Создайте справочник Экскурсии.

* На закладке "Данные" добавьте реквизиты:

* Стоимость (Тип: Число, длина 10, точность 2) — цена одного билета.

* КоличествоМест (Тип: Число, длина 6) — общее количество мест в группе.

* Справочник "Клиенты"

* Создайте справочник Клиенты.

* На закладке "Данные" добавьте реквизиты:

* ФИО (Тип: Строка, длина 100).

* Телефон (Тип: Строка, длина 20).

1.2. Создание документа "ПродажаЭкскурсии"

* Создайте документ ПродажаЭкскурсии.

* На закладке "Данные" добавьте реквизиты шапки:

* Экскурсия (Тип: СправочникСсылка.Экскурсии).

* На закладке "Данные" создайте табличную часть Участники со следующими реквизитами:

* Клиент (Тип: СправочникСсылка.Клиенты).

* ЦенаБилета (Тип: Число, длина 10, точность 2).

1.3. Создание регистра накопления "ПродажиЭкскурсий"

* Создайте регистр накопления ПродажиЭкскурсий. Выберите вид регистра «Остатки».

* На закладке "Измерения" добавьте измерения:

* Экскурсия (Тип: СправочникСсылка.Экскурсии).

* Клиент (Тип: СправочникСсылка.Клиенты).

* На закладке "Ресурсы" добавьте ресурсы:

* Количество (Тип: Число, длина 6) — количество проданных билетов.

* Сумма (Тип: Число, длина 15, точность 2) — общая сумма продажи.

Шаг 2. Написание программного кода (проведение документа)

Это ключевой этап, где реализуется логика учета.

1. Откройте модуль объекта документа ПродажаЭкскурсии.

2. Найдите процедуру ОбработкаПроведения(Отказ, РежимПроведения).

1. Замените или дополните её следующим кодом:

// Получаем курс рубля (требуется для корректного расчета итоговой суммы)

КурсРубля = 1;

КратностьРубля = 1;

// Цикл по каждой строке табличной части "Участники"
Для Каждого ТекСтрока Из Участники Цикл

// Регистр "ПродажиЭкскурсий" - Приход

Движения.ПродажиЭкскурсий.Записывать = Истина;

Движение = Движения.ПродажиЭкскурсий.ДобавитьПриход();

Движение.Период = Дата;

Движение.Экскурсия = Экскурсия;

Движение.Клиент = ТекСтрока.Клиент;

Движение.Количество = 1; // Продаем один билет

Движение.Сумма = ТекСтрока.ЦенаБилета;

// Регистр "ПродажиЭкскурсий" - Расход (списываем место из общего пула)

// Это движение отражает уменьшение доступного остатка мест на самой экскурсии.

ДвижениеРасход = Движения.ПродажиЭкскурсий.ДобавитьРасход();

ДвижениеРасход.Период = Дата;

ДвижениеРасход.Экскурсия = Экскурсия;

// Клиент для расхода не важен, оставляем пустым

ДвижениеРасход.Клиент = Неопределено;

ДвижениеРасход.Количество = 1;

// Сумма для расходного движения обычно равна нулю, так как это не денежная операция,

// а операция контроля остатков.

ДвижениеРасход.Сумма = 0;

КонецЦикла;

Шаг 3. Создание отчета "ВедомостьПоПродажамЭкскурсий"

1. В дереве конфигурации -> **Общие** -> **Отчеты** -> Добавить.

2. Назовите отчет ВедомостьПоПродажамЭкскурсий.

3. Откройте основную схему компоновки данных (СКД) отчета.

4. Нажмите кнопку "Настроить схему компоновки данных".

5. Перейдите на вкладку "Наборы данных" и нажмите "Добавить" -> "Набор данных - Запрос".

6. Введите следующий текст запроса:

ВЫБРАТЬ

ПродажиЭкскурсийОстаткиИОбороты.Экскурсия КАК Экскурсия,

СУММА(ПродажиЭкскурсийОстаткиИОбороты.КоличествоПриход)

КАК

КоличествоПродано,

СУММА(ПродажиЭкскурсийОстаткиИОбороты.СуммаПриход) КАК СуммаПродаж

ИЗ

РегистрНакопления.ПродажиЭкскурсий.ОстаткиИОбороты(, , Авто, ,) КАК

ПродажиЭкскурсийОстаткиИОбороты

СГРУППИРОВАТЬ ПО

ПродажиЭкскурсийОстаткиИОбороты.Экскурсия

7. Сохраните изменения.

8. Настройте внешний вид отчета на вкладках "Настройки", "Макет". Для автоматического макета достаточно нажать "Заполнить".

Шаг 4. Сохранение и запуск конфигурации

Нажмите **F7** для обновления конфигурации базы данных и затем **F5** для запуска в пользовательском режиме.

Шаг 5. Тестирование работоспособности

1. **Подготовка:** В разделе "Экскурсии" создайте несколько элементов, например, "Обзорная экскурсия по городу" (стоимость 1000 руб., мест - 10) и "Музей космонавтики" (стоимость 500 руб., мест - 5).

2. В разделе "Клиенты" создайте карточки для нескольких клиентов.

3. **Тестовая продажа:**

- Создайте новый документ "Продажа экскурсии".
- Выберите экскурсию "Обзорная экскурсия по городу".
- Перейдите на вкладку "Участники" и добавьте двух клиентов.
- Проведите документ.

Проверка отчета:

- Сформируйте отчет "Ведомость по продажам экскурсий". Вы должны увидеть, что по "Обзорной экскурсии" было продано 2 билета на общую сумму 2000 руб.

Контроль остатков:

- Попробуйте создать еще один документ продажи той же экскурсии и добавить туда 9 клиентов.
- При попытке провести такой документ система выдаст ошибку "Не хватает свободного остатка", так как вы пытаетесь продать больше билетов, чем есть в наличии (2 уже проданы + 9 новых > 10 всего). Это работа встроенного механизма контроля остатков.

4. Требования к отчёту

Отчёт по лабораторной работе должен содержать:

1. **Титульный лист.**

2. **Скриншоты** следующих этапов:

- Дерево конфигурации со всеми созданными объектами.
- Окно редактирования модуля документа с написанным кодом.
- Список элементов справочника "Экскурсии".
- Форма документа "ПродажаЭкскурсии" в процессе заполнения.
- Результат сформированного отчета "Ведомость по продажам экскурсий".
- Сообщение системы о нехватке остатка при попытке перепродать билеты.

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания.

Оценка «неудовлетворительно» ставится, если задание не выполнено

Тема 3.1. Программирование в среде 1С Предприятие

Лабораторное занятие №33

Разработка учетной системы для ведения информации о кассовых операциях.

Цель: Формирование умений разработки учетной системы для ведения информации о кассовых операциях.

Выполнив работу, вы будете уметь:

У 1.3.1 разрабатывать объекты базы данных, такие как таблицы, индексы и связи между ними;

Уо 04.03 использовать навыки управления проектами в распределении ресурсов и формировании графика выполнения задач.

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: 1С:ПРЕДПРИЯТИЕ.

Задание

Общие сведения

Учет наличных денежных средств — одна из фундаментальных задач любой учетной системы. Ключевым принципом здесь является двойная запись: любое движение денег должно иметь источник. Деньги не могут появиться "из ниоткуда" или исчезнуть "в никуда". Каждое поступление (приход) должно быть уравновешено соответствующим расходом или наоборот.

Для реализации этого принципа в "1С:Предприятии" используется регистр накопления с видом «Остатки». Этот регистр будет хранить текущий остаток денег в кассе. Документ "Приходный кассовый ордер" (ПКО) будет делать приходную запись в регистр, увеличивая остаток, а документ "Расходный кассовый ордер" (РКО) — расходную, уменьшая его.

2. Задание к лабораторной работе

Необходимо разработать конфигурацию, позволяющую вести учет операций по кассе предприятия. Система должна поддерживать следующие функции:

1. Хранение информации о кассовых операциях через два документа: Поступление наличных и Выдача наличных.
2. Автоматическое обновление остатка денежных средств в кассе при проведении документов.
3. Формирование отчета "Ведомость по денежным средствам", показывающего начальный остаток, все движения за период и конечный остаток.
4. Контроль отрицательного остатка (недопустимость выдачи больше, чем есть в кассе).

Порядок выполнения работы

Примечание: Работа выполняется в режиме Конфигуратора ("1С:Предприятие (Конфигуратор)")

Шаг 1. Проектирование структуры метаданных

1.1. Создание регистра накопления "ДенежныеСредства"

Это ядро нашей системы учета.

1.1. В дереве конфигурации (Общие -> Регистры накопления) создайте новый объект ДенежныеСредства.

1.2. На закладке "Основные" установите вид регистра "Остатки".

1.3. На закладке "Данные":

* Добавьте измерение Касса (Тип: Строка, длина=50). Это позволит в будущем расширить систему для учета нескольких касс.

* Добавьте ресурс Сумма (Тип: Число, длина=15, точность=2). Именно этот ресурс мы будем увеличивать и уменьшать.

Шаг 2. Создание документов для движения денег

2.1. Документ "ПоступлениеНаличных" (ПКО)

2.1. Создайте документ ПоступлениеНаличных.

2.2. На закладке "Данные" добавьте реквизиты шапки:

* Касса (Тип: Строка, длина=50).

* СуммаДокумента (Тип: Число, длина=15, точность=2).

* Основание (Тип: Строка, длина=200) - например, "Выручка от продаж".

2.2. Документ "ВыдачаНаличных" (РКО)

2.3. Создайте документ ВыдачаНаличных.

2.4. На закладке "Данные" добавьте реквизиты шапки:

* Касса (Тип: Строка, длина=50).

* СуммаДокумента (Тип: Число, длина=15, точность=2).

* Основание (Тип: Строка, длина=200) - например, "Хозяйственные расходы".

Шаг 3. Написание программного кода (проведение документов)

3.1. Модуль документа "ПоступлениеНаличных"

3.1. Откройте модуль объекта документа "ПоступлениеНаличных".

3.2. Найдите процедуру ОбработкаПроведения.

3.3. Заполните её кодом для регистрации прихода денег.

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания. Оценка «неудовлетворительно» ставится, если задание не выполнено

Тема 3.1. Программирование в среде 1С Предприятие

Лабораторное занятие №34

Разработка информационной системы.

Цель: Формирование умений разработки информационной системы по заданию преподавателя.

Выполнив работу, вы будете уметь:

У 1.3.1 разрабатывать объекты базы данных, такие как таблицы, индексы и связи между ними;

Уо 04.03 использовать навыки управления проектами в распределении ресурсов и формировании графика выполнения задач.

Материальное обеспечение:

Методические указания для выполнения практических работ, вариант задания, компьютер, программное обеспечение: 1С:ПРЕДПРИЯТИЕ.

Задание

Эта лабораторная работа является итоговой и предполагает комплексный подход к разработке. В отличие от предыдущих работ с готовыми шаблонами задач (учет студентов, экскурсий), здесь вы сталкиваетесь с необходимостью самостоятельно формализовать требования и перевести их на язык объектов метаданных платформы "1С:Предприятие".

Процесс разработки будет включать следующие этапы:

1. Анализ требований: Детальное изучение задания преподавателя, выделение ключевых сущностей (объектов) предметной области.
2. Проектирование: Создание ER-диаграммы или просто схемы связей между объектами (какие справочники нужны, какие документы будут создавать движения, какие регистры потребуются).
3. Реализация: Непосредственное создание объектов в Конфигураторе, написание программного кода.
4. Тестирование: Проверка работоспособности системы путем ввода тестовых данных и анализа полученных результатов.
5. Документирование: Подготовка отчета о проделанной работе.

Задание к лабораторной работе

Задание формулируется преподавателем индивидуально. Ниже приведен универсальный алгоритм, который поможет вам выполнить любую подобную работу. В качестве примера будем исходить из гипотетического задания: «Разработать систему учета заявок на ремонт оборудования».

Примерное содержание типового задания:

- * Учет клиентов-заказчиков.
- * Учет парка оборудования у каждого клиента.
- * Регистрация заявок на ремонт с указанием вида поломки, срочности и ответственного мастера.
- * Отслеживание статуса заявки (Новая, В работе, Выполнена, Отменена).
- * Формирование отчетов: "Заявки в работе по мастерам", "Выполненные заявки за период".

Ваша задача — адаптировать этот алгоритм под ваше конкретное задание.

Порядок выполнения работы

Анализ задания и проектирование структуры данных

Прежде чем открывать Конфигуратор, необходимо продумать архитектуру вашей будущей системы.

1. Выделите основные сущности. Для нашего примера это: Клиенты, Оборудование, Заявки.
2. Определите типы объектов для каждой сущности:
 - * Клиенты -> Справочник Клиенты.
 - * Оборудование -> Справочник Оборудование (с реквизитом-владельцем Клиент).
 - * Заявки -> Документ ЗаявкаНаРемонт.
3. Продумайте связи и реквизиты:
 - * У справочника Клиенты будут реквизиты: ИНН, Адрес, Телефон.
 - * У справочника Оборудование: Инвентарный номер, Модель, Дата установки (реквизит-ссылка на Клиента).
 - * У документа ЗаявкаНаРемонт: Реквизиты шапки - Клиент, Оборудование, ДатаИсполнения. Табличная часть СписокРабот с полями ВидРаботы, Статус.
4. Определитесь с учетными механизмами. Если требуется накапливать количественные показатели (например, сколько заявок выполнил каждый мастер), понадобится регистр накопления. Если нужно хранить историю изменения статусов — регистр сведений.

Форма представления результата:

Отчет по выполненной лабораторной работе

Критерии оценки:

Оценка «отлично» ставится, если задание выполнено верно.

Оценка «хорошо» ставится, если ход выполнения задания верный, но была допущена одна или две ошибки, приведшие к неправильному результату.

Оценка «удовлетворительно» ставится, если приведено неполное выполнение задания. Оценка «неудовлетворительно» ставится, если задание не выполнено