Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ (ИУ)

КАФЕДРА ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ (ИУ7)

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

База данных сети магазинов

Студент группы **ИУ7-66Б** _____ **Ковель А. Д.**

Руководитель курсовой работы _____ **Степанов В. П.**

2023 г.

РЕФЕРАТ

Расчётно-пояснительная записка содержит 34 с., 6 рис., 2 табл., 16 ист.

Целью работы: создание базы данных для отслеживания посетителей в сети магазинов.

Ключевые слова: базы данных, PostgreSQL, реляционная модель, OLAP, OLTP.

В данной работе проводится изучение принципов работы баз данных.

Объектом исследования является модель представления данных в посетителей в сетях магазинов.

Результаты: разработана программа, предназначенная для работы с базами данных. Проанализированы разыные системы управления базами данных.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

OLAP — online analytical processing — интекративная аналитическая обработка.

OLTP — online transaction processing — транзакционная система.

БД — база данных.

СУБД — система управления базами данных.

ВВЕДЕНИЕ

В настоящее время сетевые магазины предоставляют своим клиентам широкий ассортимент товаров и услуг, а также внедряют новые технологии для улучшения качества обслуживания. Одной из таких технологий является отслеживание посетителей в сети магазинов, которое позволяет собирать информацию о перемещениях клиентов внутри магазина и анализировать ее для бизнес целей.

Целью работы: создание базы данных для отслеживания посетителей в сетях магазинов.

Для достижения поставленной цели, необходимо решить следующие задачи:

1. формализовать задачу и определить необходимый функционал;
2. описать структуру объектов БД;
3. выбрать СУБД для хранения данных;
4. спроектировать и реализовать программу для обработки заявок, которая будет взаимодействовать с описанной базой данных;
5. провести исследование времени обработки операций от количества запросов в СУБД.

1 Аналитическая часть

В данном разделе проведен анализ предметной области, формализированы данные, а также проведен анализ существующих решений.

Были выявлены основные проблемы, которые возникают при работе с данными, а также потребности пользователей в области управления и хранения информации.

Для более точного и структурированного анализа предметной области были формализированы данные, собранные из различных источников. Это позволило установить зависимости между различными факторами и определить ключевые требования к будущей системе управления базами данных.

Кроме того, в рамках данной работы был проведен анализ существующих решений на рынке, что позволило определить наиболее эффективные подходы к решению задач в области управления данными. Были выявлены преимущества и недостатки различных СУБД, а также проанализированы различные методы и подходы к созданию структуры БД.

1.1 Анализ предметной области

Термин "база данных"не имеет точного определения, но стоит отметить несколько из них.

База данных [1] — это совокупность данных, хранимых в упорядоченной форме, с целью обеспечения доступа к этим данным и их использования каким-либо организационными или прикладными процессам.

База данных — это самодокументирования собрание интегрированных записей.

- Запись — это события, которые надо где-то хранить;
- интегрированных — записи, которые имеют некоторую структуру.

Также необходимо определиться с типом базы данных. Всего существует два основных применения баз данных

1. **OLAP** [2] — это метод обработки данных, который используется для

анализа больших объемов данных.

2. **OLTP** [3] — это метод обработки транзакций, который используется для выполнения операция в режиме реального времени.

Из этих данных определений следует, что для поставленной задачи больше подойдет метод OLTP, так как для обработки посетителей в магазине, требуется обработка в реальном времени.

Для выполнения курсовой работы, также необходимо выбрать систему управления базами данных.

СУБД — это приложение обеспечивающее создание, хранение, обновление и поиск информации.

У систем управления базами данных существует классификация:

1.2 Классификация СУБД

1.2.1 По модели данных

Модель данных базы данных определяет, каким образом данные будут храниться и организовываться внутри системы управления базами данных (СУБД). Существуют две основные модели данных: реляционная и дореляционная. Каждая из них имеет свои особенности и может быть более или менее подходящей для конкретных задач. Давайте подробнее рассмотрим каждую из этих моделей.

1.2.2 Дореляционные

1. **Инвертированные списки** (файлы). БД на основе инвертированных списков представляет собой совокупность файлов, содержащих записи (таблиц). Для записей в файле определен некоторый порядок, диктуемый физической организацией данных. Для каждого файла может быть определено произвольное число других упорядочений на основании значений некоторых полей записей (инвертированных списков). Обычно для этого используются индексы. В такой модели данных отсутствуют ограничения целостности как таковые. Все ограничения на возможные экземпляры БД задаются теми программами, которые работают с БД.

Одно из немногих ограничений, которое все-таки может присутствовать — это ограничение, задаваемое уникальным индексом.

2. **Иерархическая модель** данных подразумевает что элементы, организованные в структуры, объединены иерархической или древовидной связью. В таком представлении родительский элемент может иметь несколько дочерних, а дочерний — только один родительский.
3. **Сетевые** — могут быть представлены в виде графа; логика выборки зависит от физической организации данных.

1.2.3 Реляционные

В отличие от вышеописанных, в данной модели не существует физических отношений между сущностями. Хранение информации осуществляется в виде таблиц (отношений), состоящих из рядов и столбцов. Отношение имеет имя, которое отличает его от имён всех других отношений.

Существует несколько типов реляционных моделей:

- **Структурный** — данные — набор отношений.
- **Целостностный** — отношения (таблицы) отвечают определенным условиям целостности.
- **Манипуляционный** — манипулирования отношениями осуществляется средствами реляционной алгебры и/или реляционного исчисления.

1.2.4 По архитектуре организации хранения данных

Различаются несколько типов, таких как: "Локальные"и "Распределенные".

1. **Локальные** — все части локальной СУБД размещаются на одном компьютере. То есть вся информция будет хранится на одном устройстве.
2. **Распределенные** — части СУБД могут размещаться на 2-х и более компьютерах. Такое хранение позволяет сократить риски, и предоставляет возможность возврата данных при их утери.

1.2.5 По способу доступа к БД

1. **Файл-серверные** — при работе с базой, данные отправляются приложению, которое с ней работает, вне зависимости от того, сколько их нужно. Все операции — на стороне клиента. Файловый сервер периодически обновляется тем же клиентом.
2. **Клиент-серверные** — вся работа на сервере, по сети передаются результаты запросов, гораздо меньше информации. Обеспечивается безопасность данных, потому что все происходит на стороне сервера.
3. **Встраиваемые** — библиотека, которая позволяет унифицированным образом хранить; большие объемы данных на локальной машине. Доступ к данным может происходить через SQL либо через особые функции СУБД. Встраиваемые СУБД быстрее обычных клиент-серверных и не требуют установки сервера, поэтому востребованы в локальном ПО, которое имеет дело с большими объемами данных.
4. **Сервисно-ориентированные** — БД является хранилищем сообщений, промежуточных состояний, метаинформации об очередях сообщений и сервисах;
5. Прочие — пространственная, временная и пространственно-временная.

1.3 Пользователи системы

В системе присуствуют три уровня пользователей.

1. **Сотрудник магазина** — пользователь, обладающим возможностями только просматривать сущность посетителей.
2. **Охрана** — пользователь, обладающий возможностями просматривать сущности: камер, украденных товаров, полок, посетителей.
3. **Администратор** — пользователь, обладающий возможностями изменения сущностей и полей базы данных, также есть доступен просмотр всех сущностей.

На рисунке 1 представлена диаграмма использования приложения.

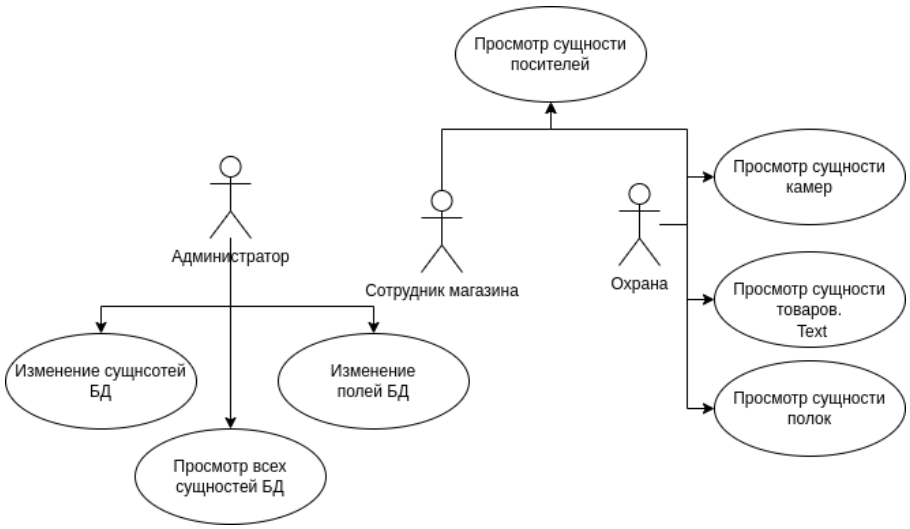

Рисунок 1 – Use-case диаграмма

1.4 Формализация данных

Для эффективного управления информацией необходимо формализовать ее представление. В базе данных, используемой для учета товаров в магазине, данные представлены в виде таблиц.

База данных состоит из нескольких таблиц:

1. таблица посетителей Visitor;
2. таблица камер Camera;
3. таблица полок Shelf;
4. талица товаров Product;
5. таблица сетей магазинов Chain store;

Каждая таблица содержит определенную информацию. Таблица посетителей хранит данные о посетителях магазина, такие как имя, фамилия, возраст и т.д. Таблица камер содержит информацию о расположении камер видеонаблюдения внутри магазина. Таблица полок содержит данные о количестве и расположении полок для товаров. Таблица товаров содержит информацию о товарах, такую как название, производитель, цена и т.д. Таблица сетей магазинов содержит данные о магазинах, входящих в сеть.

На рисунке представлена ER-диаграмма сущностей в нотации Чена.

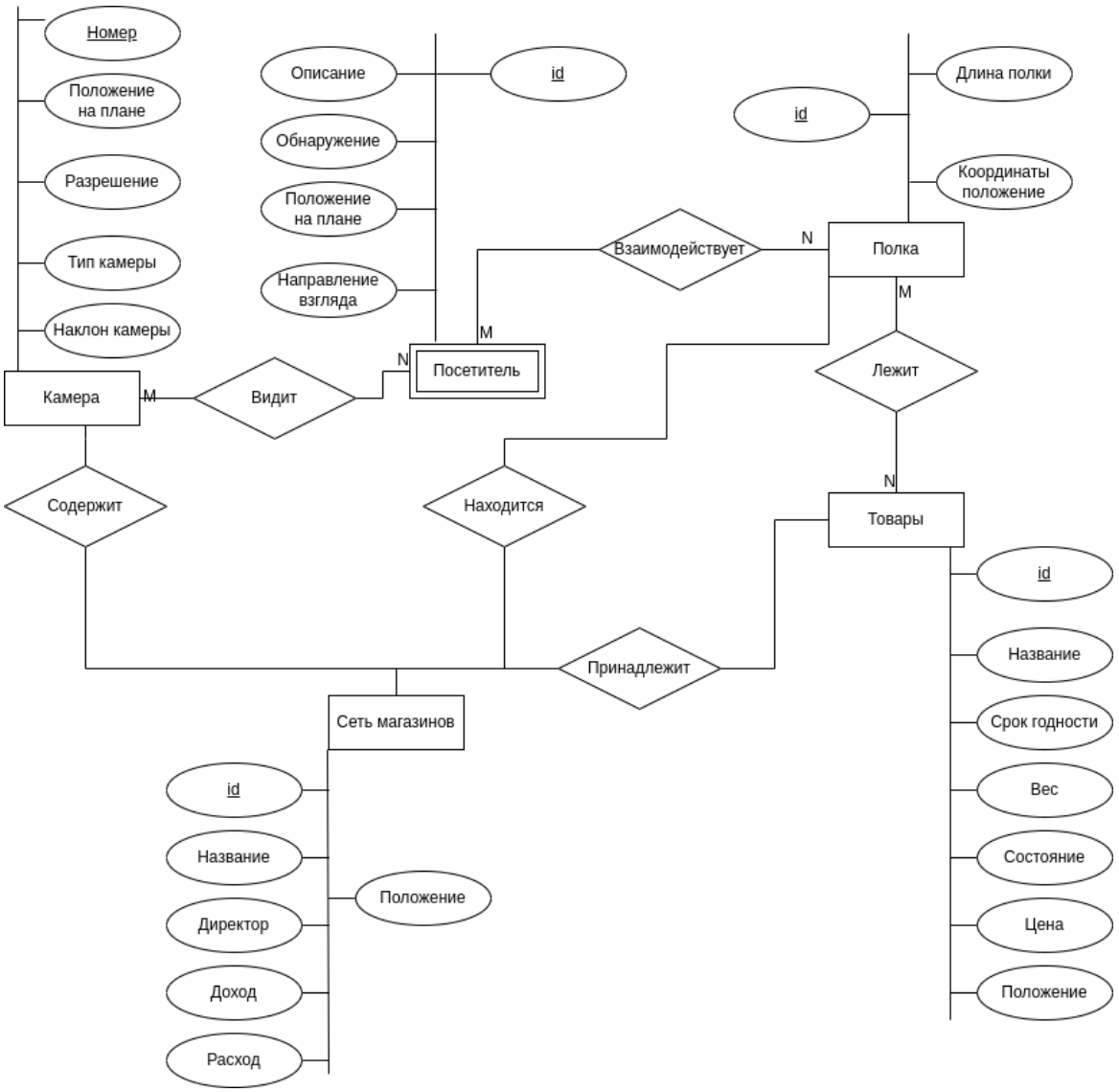

Рисунок 2 – ER-диаграмма в нотации Чена

1.5 Анализ существующих решений

Среди уже имеющихся проектов, решающих поставленную задачу, были выделены 3 аналога. Сравнение проводилось по ряду критериев, а именно наличие определения местоположения, не требующее дополнительного оборудования, определение характеристик посетителей.

В таблице 1 представлено сравнение по вышеупомянутым критериям.

Из таблицы можно сделать вывод, что каждая система требует большое количество дополнительного оборудования. Также не во всех присутствует определение характеристик посетителей.

Создаваемое программное обеспечение предоставляет дополнительный

Таблица 1 – Существующие решения поставленной задачи

Название проекта	**Местоположение**	**Доп. оборудование**	**Характеристики**
Антивор [4]	Определение местоположения может проводится при помощи дополнительного оборудования	Требуется большое количество дополнительного оборудования для работы системы	Не предусмотренно
Воролов [5]	Нет	Оборудование не образует систему	Не предусмотренно
Navigine [6]	Определение с помощью дополнительного оборудования	Браслет и стационарный датчик	С помощью браслета

функционал, без использования дополнительного оборудования.

Такими особенностями являются:

1. определение местоположения на основе положения камер;
2. задание характеристик посетителей;
3. проверка взаимодействия посетителя и товара.

1.6 Выбор модели базы данных

В данном проекте будет использоваться структурная реляционная модель данных, так как в рамках проектах она обладает следующими преимуществами:

1. представление информации осуществляется с помощью таблиц;
2. позволяет работать со структурированными данными;
3. имеет возможность произвольного доступа к записям сущностей;
4. исключает дублирование, при помощи реализации связи между отно-

шениями посредством внешнего ключа.

Вывод

В данном разделе рассмотрены ролевые модели системы, конкретизированны хранимые данные и их связь между собой, построенны соответствующие диграммы. Также представлено анализ существующих решений. Был осуществлен выбор модели данных.

2 Конструкторская часть

Конструкторская часть проектирования базы данных включает в себя несколько этапов, начиная от анализа требований и разработки концептуальной модели, заканчивая созданием физического дизайна БД. Один из самых важных этапов — это определение правил и триггеров, которые должны контролировать целостность данных в базе и предоставляют широкую возможность модификации базы данных.

2.1 Формализация сущностей системы

При проектировании базы данных необходимо определить все сущности, которые будут использоваться в системе. Каждая сущность представляет собой объект, который имеет свои атрибуты и отношения с другими сущностями.

На основе выделенных ранее сущностей 1 спроектированы таблицы базы данных, где содержаться название полей, которые будут представлены в базе данных.

2.1.1 Таблица Visitor

Содержит информацию о посетителях магазинов, которые включает следующие поля:

1. id — индетификатор пользователя, который является первичным ключом;
2. description — описание пользователя, вещественный тип;
3. location — положение пользователя на плане магазина, символьный тип который состои из двух вещественных чисел;
4. view — вектор взгляда, символьный тип, состоящий из шести вещественных чисел, которые задают вектор взгляда;
5. detection — описание пользователя, символьный тип, состоящий из 4 вещественных чисел, которые задают прямоугольник обнаружения посетителя.

2.1.2 Таблица Camera

Содержит информацию о камерах в магазине, которые включает следующая поля:

1. id — номер камеры в магазине, первичный ключ;
2. location — положение пользователя на плане магазина, символьный тип который состои из трех вещественных чисел;
3. resolution — разрешение камеры, символьный тип состоящий из двух вещественных чисел;
4. rotation — наклон камеры, символьный тип. Является матрицей состоящих из 9 вещественных чисел;
5. type — тип камеры, символьный тип.

2.1.3 Таблица Shelf

Таблица полок необходима для представления местоположения товаров, и дает представление с какими товарами взаимодействуют посетители магазина. Она содержит информацию о полках с товарами в магазине, которые включает следующая поля:

1. id — номер полки в магазине, первичный ключ;
2. location — положение пользователя на плане магазина, символьный тип который состои из трех вещественных чисел;
3. length — длина полки, вещественный тип.

2.1.4 Таблица Product

Таблица продуктов необходима для представления товаров в магазине, данная информация поможет структурировать доходы и расходы магазинов. Она содержит информацию о товарах в магазине, которые включает следующая поля:

1. id — номер товара в магазине, первичный ключ;
2. location — положение товара на плане магазина, символьный тип который состои из трех вещественных чисел;

3. name — имя товара, символьный тип;
4. dataEnd — срок годности, тип дата;
5. weight — вес товара, вещественный тип;
6. status — статус товара, целочисленный тип;
7. price — цена товара, вещественный тип.

2.1.5 Таблица ChainStore

Данная таблица объединяет разные магазины в одну общую таблицу, что позволит получать общую информацию сетей магазинов. Она содержит информацию о сетях магазинов, которые включает следующая поля:

1. id — номер магазина, первичный ключ;
2. location — положение магазина в городе, символьный тип который состои из двух вещественных чисел (широта и долгота);
3. name — имя магазина, символьный тип;
4. nameDir — имя директора магазина, символьный тип;
5. income — доход магазина, вещественный тип;
6. consumption — расходы магазина, вещественный тип.

2.2 Ролевая модель

Для эффективного взаимодействия пользователей с системой управления базами данных была разработана ролевая модель, которая позволит определить доступные функциональные возможности каждому пользователю в соответствии с его ролями и ответственностями. Это значительно повышает безопасность и защищенность системы, так как пользователи не имеют доступа к чувствительным данным, которые не относятся к их компетенции.

Кроме того, ролевая модель облегчает работу администраторов, так как она позволяет быстро и удобно настраивать доступ к базам данных для конкретных пользователей или групп пользователей. Также благодаря ролевой модели упрощается процесс мониторинга и аудита действий пользователей в системе.

Таким образом, ролевая модель является важной составляющей систе-

мы управления базами данных, которая обеспечивает ее надежную и бесперебойную работу в соответствии с требованиями пользователям и бизнес-процессам.

2.2.1 Сотрудник Employee

В данном случае, роль пользователей имеет право выполнения операции SELECT над таблицей Visitor.

Операция SELECT позволяет выбирать данные из указанной таблицы. В данном случае, роль имеет доступ только к таблице Visitor и может выбирать из нее информацию, но не может изменять, удалять или добавлять новые данные.

Такая ролевая модель может быть полезна для обеспечения безопасности данных в системе управления базами данных (СУБД). Она позволяет ограничить доступ пользователей к чувствительным данным и установить строгий контроль за использованием данных.

1. SELECT — над таблицей Visitor.

2.2.2 Охрана Security

Это описание также относится к ролевой модели управления доступом к данным в базе данных. Роль пользователей, которой предоставлены данные привилегии, имеет право выполнения операции SELECT над таблицами Visitor, Camera, Product и Shelf.

Таким образом, данная роль имеет более широкий доступ к данным, чем в предыдущем описании. Она может выбирать информацию из четырех таблиц: Visitor, Camera, Product и Shelf. Операция SELECT позволяет выбирать данные из указанных таблиц, но не изменять или удалять их.

1. SELECT — над таблицей Visitor;
2. SELECT — над таблицами Camera, Product, Shelf.

2.2.3 Администратор Administrator

Роль администратора имеет полный доступ ко всем таблицам в базе данных и обладает всеми правами на выполнение операций с данными.

1. все права над таблицами Visitor;
2. все права над таблицами Camera, Product, Shelf;
3. все права над таблицами ChainStore.

Таким образом, администратор - это самый привилегированный пользователь в системе управления базами данных. Ему предоставляется полный контроль над всеми аспектами работы с данными в базе данных и он может выполнять любые необходимые операции.

2.3 Разработка триггера и функции

В СУБД предусмотрена функция, которая проверяет местоположение пользователя относительно выхода. Данная функция возвращает статус посетителя (находится внутри или вне магазина).

Также системе представлен ALTER триггер, который оповещает систему о том, что посетитель вышел из магазина.

Вывод

В данном разделе были формализированны сущности системы, представлен рисунок диаграммы сущности системы, выделены ролевые модели, спроектированы триггер и функция.

3 Технологическая часть

В данном разделе выбирается СУБД, средства реализации приложения, описаны создание базы данных, триггера, функции и ролей, а также спроектирован пользовательский интерфейс. Описание процесса создания базы данных, триггеров, функций и ролей также является важной частью разработки приложения. Создание базы данных предполагает определение ее структуры и основных таблиц, а также правил связывания и обработки данных. Триггеры и функции позволяют автоматизировать процессы обработки данных и повысить эффективность работы с базой данных. Роли, в свою очередь, обеспечивают более гибкое управление доступом к данным и повышают безопасность системы.

3.1 Выбор СУБД

Существует множество различных СУБД, работающих на основе реляционной модели, каждая из которых имеет свои сильные и слабые стороны. Среди самых распространенных [7] выделяют MySQL [8], PostgreSQL [9] и SQLite [10]. Рассмотрим особенности каждой из них.

1. MySQL. Среди достоинств данной СУБД можно выделить высокую безопасность и масштабируемость, поддержку большей части функционала SQL. Однако, несмотря на перечисленные положительные аспекты, MySQL не сопровождается бесплатной технической поддержкой.
2. PostgreSQL. В рамках использования этой СУБД имеется возможность помимо встроенного SQL использовать различные дополнения, отличается поддержкой форматов csv и json, но оперирует большим объемом ресурсов.
3. SQLite. Очевидными достоинствами является компактность базы данных, которая состоит из одного файла, и переносимость. Но данная СУБД совершенно не подходит для больших БД, а также не поддерживает управление пользователями.

При реализации проекта использован PostgreSQL, поскольку эта СУБД обладает достаточным набором инструментов для поставленной задачи.

3.2 Требования к программе

Для того чтобы программное обеспечение удовлетворяло требованиям, необходимо определить их заранее и придерживаться их в процессе разработки. Программное обеспечение должно удовлетворять требованиям, которые необходимы для работы спроектированной системы:

1. возможно подключение к бд;
2. программа позволяет определять время запросов;
3. возможно создание программного интерфейса для работы с бд.

3.3 Средства реализации

Для реализации ПО был выбран язык программирования Python[11]. Для управления зависимостями в проекте можно использовать менеджер пакетов pip, с помощью которого можно устанавливать сторонние библиотеки и модули. В данном языке есть все требующиеся инструменты для данной курсовой работы. В качестве среды разработки была выбрана среда VS Code[12], запуск происходил через команду python back.py.

3.4 Создание базы данных

В соответствии с выбранной СУБД и спроектированной базой данных было осуществлено создание БД и ее сущностей представлено в листинге 1.

Листинг 1 – Создание БД

```
class Visitor(BASE):
    __tablename__ = 'Visitor'
    id = Column(Integer, Identity(always=True), primary_key=True)
    description = Column(Text)
    location = Column(Text)
    view = Column(Text)
    detection = Column(Text)
class Camera(BASE):
    __tablename__ = 'Camera'
    id = Column(Integer, Identity(always=True), primary_key=True)
    location = Column(Text)
```

```
    resolution = Column(Text)
    rotation = Column(Text)
    cam_type = Column(Text)
class CameraVisitor(BASE):
    __tablename__ = 'CameraVisitor'
    id_vis = Column(Integer, ForeignKey('Visitor.id'), primary_key=True)
    id_cam = Column(Integer, ForeignKey('Camera.id'), primary_key=True)
class Shelf(BASE):
    __tablename__ = 'Shelf'
    id = Column(Integer, Identity(always=True), primary_key=True)
    location = Column(Text)
    length = Column(Float)
class ShelfVisitor(BASE):
    __tablename__ = 'ShelfVisitor'
    id_shelf = Column(Integer, ForeignKey('Shelf.id'), primary_key=True)
    id_cam = Column(Integer, ForeignKey('Camera.id'), primary_key=True)
class Product(BASE):
    __tablename__ = 'Product'
    id = Column(Integer, Identity(always=True), primary_key=True)
    name = Column(Text)
    location = Column(Text)
    dataEnd = Column(Date)
    weight = Column(Float)
    status = Column(Integer)
    price = Column(Integer)
class ShelfProduct(BASE):
    __tablename__ = 'ShelfProduct'
    id_shelf = Column(Integer, ForeignKey('Shelf.id'), primary_key=True)
    id_product = Column(Integer, ForeignKey('Product.id'), primary_key=True)
class ChainStore(BASE):
    __tablename__ = 'ChainStore'
    id = Column(Integer, Identity(always=True), primary_key=True)
    name = Column(Text)
    location = Column(Text)
    nameDir = Column(Text)
    income = Column(Float)
    consumption = Column(Float)
```

3.5 Создание триггера

В предыдущем разделе был спроектирован триггер AFTER. Код его создания представлен в листинге 2

Листинг 2 – Реализация триггера AFTER

```
CREATE TRIGGER check_location_trigger
AFTER UPDATE ON "Visitor"
FOR EACH ROW
EXECUTE FUNCTION check_location();
```

Для этого триггера была написана соответствующая функция.

Код функции представлен в листинге 3.

Листинг 3 – Реализация функции

```
CREATE FUNCTION check_location() RETURNS TRIGGER AS $$
BEGIN
    IF NEW.location = '0␣0' THEN
    RAISE NOTICE 'New␣visitor␣added␣with␣location:␣%', NEW.location;
    Return NEW;
    ELSE
    RETURN NULL;
    END IF;
END;
$$ LANGUAGE plpgsql;
```

3.6 Создание ролей и выделение им прав

В конструкторском разделе была разработана ролевая модель, в которой выделены следующие роли:

1. employee — сотрудник;
2. security — охрана;
3. administrator — администратор.

Соответствующий этой ролевой модели сценарий создания ролей и выделения им прав представлен на листинге 4.

Листинг 4 – Создание ролей и выделение им прав

```
CREATE ROLE employee LOGIN PASSWORD 'postgres';
```

```
GRANT SELECT ON TABLE "Visitor" TO employee;

CREATE ROLE security LOGIN PASSWORD 'postgres';
GRANT SELECT ON TABLE "Visitor", "Product", "Shelf" TO security;

CREATE ROLE administrator LOGIN PASSWORD 'postgres';
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO administrator;
```

3.7 Интерфейс программы

Для работы с БД был разработан интерфейс взаимодействия в виде API [13]. Для реализации API была использована библиотека fastapi [14].

В программного интерфейсе реализованы методы: добавления, чтения, обновления и удаления для каждой таблицы. Методы представлены на рисунке 3.

Для использования чтения необходимо указать количество элементов, которые будут выведены. Чтобы добавить элемент в таблицу необходимо указать, необходимые поля в таблице. Для обновления и удаления элемента таблицы, необходимо указать его номер.

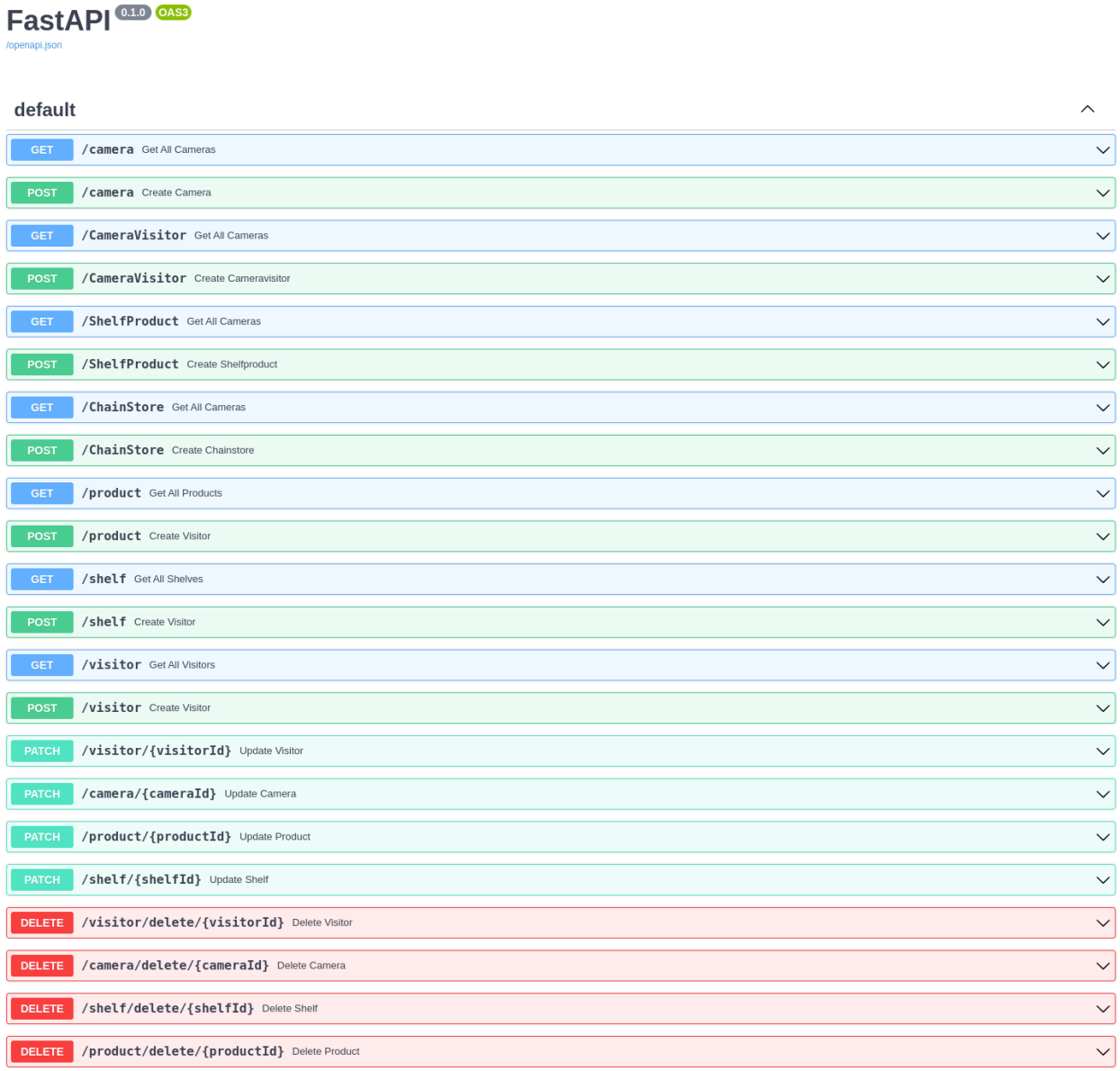

Рисунок 3 – Программный интерфейс

Вывод

В данном разделе выбраны СУБД и средств реализации, описано создание БД, триггера, ролей с выделением прав. Также представлен пользовательский интерфейс.

4 Исследовательская часть

В данном разделе произведено постановка задачи исследования и представлены результаты данного исследования. Целью исследования было изучение возможностей использования баз данных в различных сферах деятельности, выявление преимуществ и недостатков различных подходов к проектированию и реализации БД. Таким образом, результаты данного исследования могут быть использованы для выбора наиболее подходящего варианта базы данных для конкретных задач, а также для определения основных требований к БД при проектировании систем, которые будут использовать данную базу данных.

4.1 Технические характеристики

При тестировании программного обеспечения очень важно учитывать технические характеристики устройства, на котором происходят испытания. Это позволяет получить более точные результаты и оценить реальную производительность ПО. Тестирование выполнялось на устройстве, которое было подключено в сеть и не находилось под нагрузкой другими приложениями, со следующими техническими характеристиками:

- операционная система Pop!_OS 22.04 LTS [15] Linux [16];
- оперативная память 16 Гбайт;
- процессор AMD® Ryzen 7 2700 eight-core processor × 16 [**?**].

4.2 Демонстрация работы

На рисунке 4 демонстрируется запрос, который показывает каких посетителей видит камера, что поволяет определить местоположение всех людей в магазине. Системы видеонаблюдения широко используются в различных сферах деятельности, включая торговые центры, аэропорты, стадионы и другие общественные места. Они позволяют контролировать происходящее в режиме реального времени и быстро реагировать на возможные угрозы и непредвиденные ситуации.

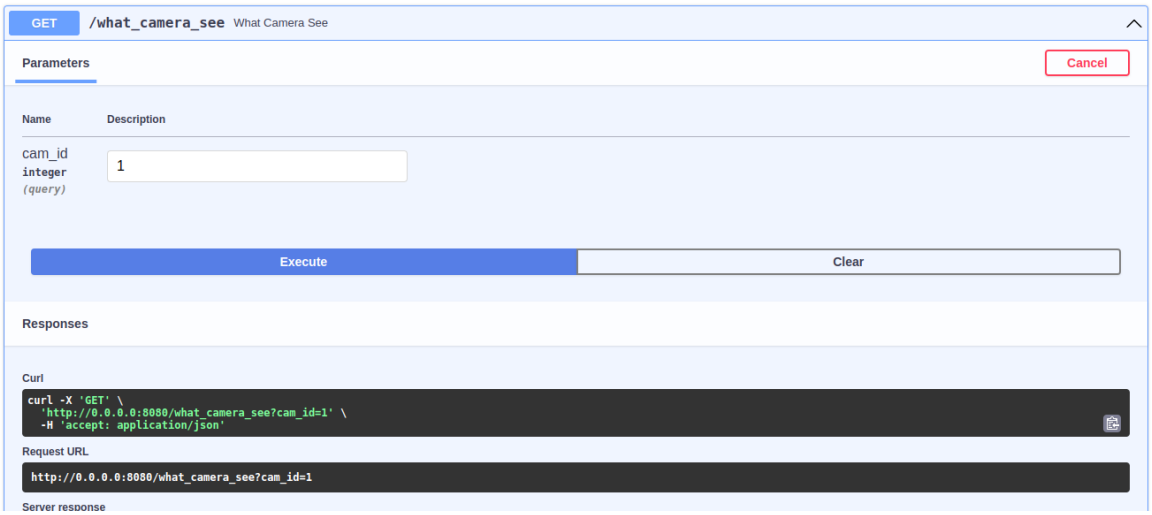

Рисунок 4 – Демонстрация запроса

На рисунке 5 демонстрируется результат запроса, из которого видно, что камера видит трех посетителей, в виде json документа.

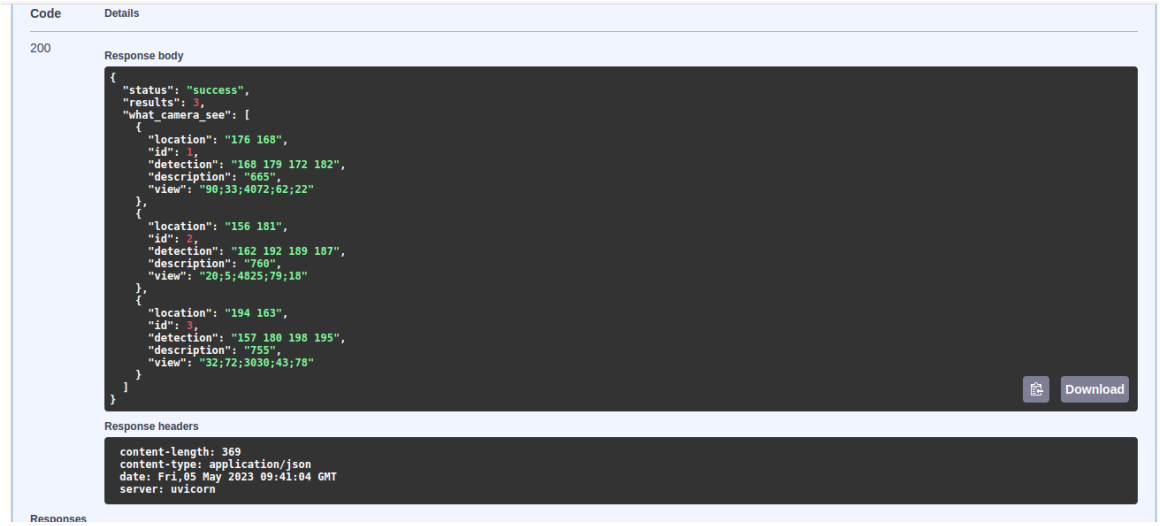

Рисунок 5 – Демонстрация результата запроса

4.3 Постановка исследования

Постановка исследования - это важный этап любого исследовательского проекта, который позволяет определить цели и задачи исследования, а также выбрать методы и инструменты для достижения поставленной цели.

В данном случае, целью исследования является изучение времени обработки операции от количества запросов в системе управления базами данных (СУБД). Эта цель может быть актуальна для предприятий и организаций,

которые используют БД для хранения и обработки большого объема информации.

Исследование времени обработки операции от количества запросов в СУБД может быть полезным для оптимизации работы систем управления базами данных, а также для повышения эффективности работы с БД в целом. Полученные результаты могут помочь организациям выбрать наиболее подходящую СУБД и оптимальную структуру таблиц, а также определить оптимальное количество запросов для ускорения работы СУБД без потери качества данных.

Целью является исследование времени обработки операции от количества запросов в СУБД.

4.4 Результаты исследования

В таблице 2 продемонстрировано пользовательское время программы при разном количестве полей таблиц.

Таблица 2 – Время работы программы при разном количестве полей

Количетво полей	Время в с.
100	0.006
1000	0.009
10000	0.097
100000	1.170
1000000	11.118

На рисунке 6 представлено время запроса к БД.

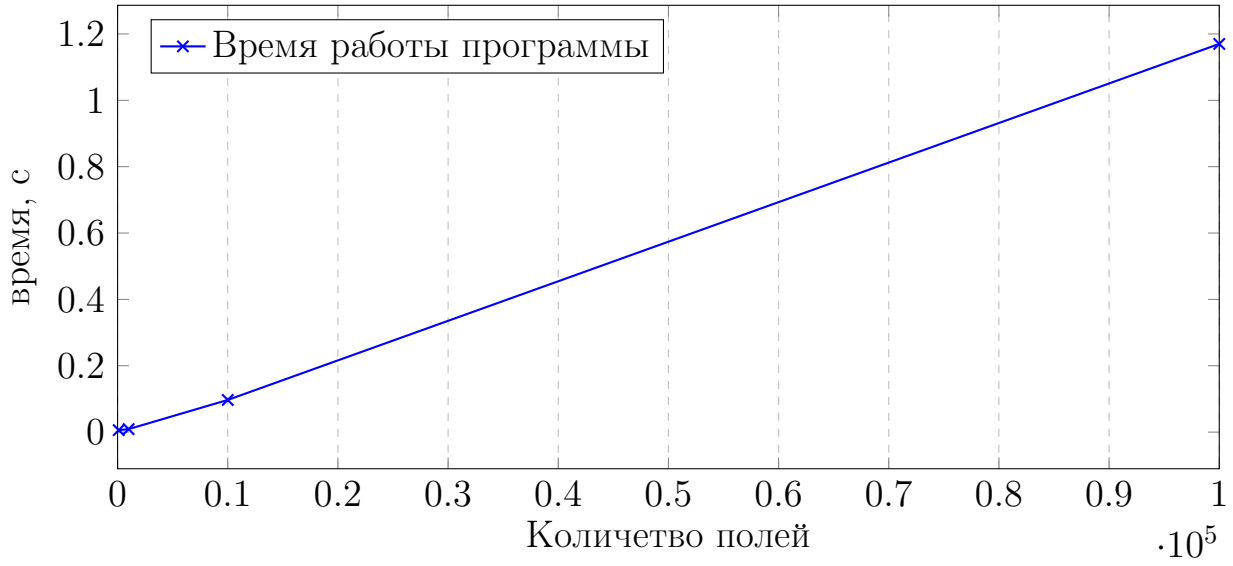

Рисунок 6 – Результаты время запроса к БД

Из результатов исследования можно сделать вывод, что зависимость времени запроса от количества полей линейно.

Вывод

В данном разделе постановлена задачи исследования и представлены результаты данного исследования.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной работы были выполнены следующее задачи:

1) формализована задача и определен необходимый функционал;
2) описана структуру объектов БД;
3) выбрана СУБД для хранения данных;
4) спроектирована и реализована программу для обработки заявок, которая будет взаимодействовать с описанной базой данных;
5) проведено исследование времени обработки операций от количества запросов в СУБД.

Цель курсового проекта достигнута.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Роб П., Коронелл К. Базы данных: концепции, технологии, применение. БХВ-Петербург: Вильямс, 2004. с. 15.

2. Архипенков С. Я., Голубев Д. В., Максименко О. Б. Хранилища данных. Москва: Диалог-МИФИ, 2002. с. 105.

3. Архипенков С. Я., Голубев Д. В., Максименко О. Б. Хранилища данных. Москва: Диалог-МИФИ, 2002. с. 106.

4. Антивор. `https://antivor.ru/`. дата обращения: 31.03.2023.

5. Антивор. `https://vorolov.ru/schetchiki-posetitelej/`. дата обращения: 31.03.2023.

6. Navigine. `https://nvgn.ru/platform/tracking/`. дата обращения: 31.03.2023.

7. LearnSQL. The Most Popular Databases for 2022. `https://learnsql.com/blog/most-popular-databases-2022/`. дата обращения: 01.05.2023.

8. MySQL. `https://www.mysql.com/`. дата обращения: 01.05.2023.

9. Лузанов П., Рогов Е., Лёвшин И. Postgres: первое знакомство. Москва: Диалог-МИФИ, 2002. с. 90.

10. SQLite. `https://www.sqlite.org/`. дата обращения: 01.05.2023.

11. Python. `https://www.python.org/`. дата обращения: 01.05.2023.

12. Vscode. `https://code.visualstudio.com/insiders/`. дата обращения: 01.05.2023.

13. API. `https://www.ibm.com/topics/api`. дата обращения: 01.05.2023.

14. FastAPI. https://fastapi.tiangolo.com/. дата обращения: 01.05.2023.

15. PopOs. https://pop.system76.com/. дата обращения: 01.05.2023.

16. Linux – Документация [Электронный ресурс]. Режим доступа: https://docs.kernel.org (дата обращения: 24.09.2022).

ПРИЛОЖЕНИЕ А

Презентация состоит из 16 страниц.