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»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

«Разработка базы данных для хранения и обработки данных магазина одежды»

Студент ИУ7-63Б _____ И. А. Гринкевич
(Группа) (Подпись, дата) (И. О. Фамилия)

Руководитель курсовой работы _____ А. Л. Исаев
(Подпись, дата) (И. О. Фамилия)

2023 г.

РЕФЕРАТ

Расчетно-пояснительная записка 53 с., 12 рис., 9 табл., 18 источн., 1 прил.
АВИАБИЛЕТЫ, АВИАПЕРЕЛЕТЫ, БАЗЫ ДАННЫХ, ВЕБ-ПРИЛОЖЕНИЕ, РЕЛЯЦИОННАЯ МОДЕЛЬ ДАННЫХ, SQL

Объектом разработки является база данных и приложение к ней.

Объектом исследования являются кластеризованные и некластеризованные индексы.

Цель работы: разработка базы данных для хранения и обработки данных авиакомпании и веб-приложения, которое будет ее использовать.

В результате выполнения работы была разработана база данных для хранения и обработки данных авиакомпании и веб-приложение, использующее эту базу данных.

В ходе выполнения конструкторской части были выделены шесть сущностей: пользователь, заказ, рейс, билет, самолет и услуга. А также три роли на уровне базы данных: клиент, модератор и администратор.

При выполнении технологической части были выбраны средства реализации программного обеспечения (язык программирования C# и система управления базами данных Microsoft SQL Server), реализован интерфейс доступа к базе данных и проведено тестирование разработанного функционала (mock-тесты, интеграционные тесты и тесты по сценариям использования).

В ходе проведения исследования было установлено, что при 250 тысячах строк в таблице использование и кластеризованного, и некластеризованного индекса увеличивает скорость выполнения запроса в 21 раз при поиске первой записи и в 97 раз при поиске последней на примере одной из таблиц спроектированной базы данных.

Область применения результатов — дальнейшее развитие и расширение приложения для поиска и покупки билетов на любые виды транспорта.

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящей расчетно-пояснительной записке к курсовой работе применяют следующие сокращения и обозначения:

БД	База данных
ПО	Программное обеспечение
СУБД	Система управления базами данных
ER-диаграмма	Диаграмма «сущность-связь» (от англ. entity-relationship)
GPS	Система глобального позиционирования (Global Positioning System)
ID	Идентификатор
MPA	Многостраничное приложение (Multi Page Application)
SPA	Одностраничное приложение (Single Page Application)
Use case диаграмма	Диаграмма вариантов использования

ВВЕДЕНИЕ

В современном мире информация является ключевым ресурсом для успешного ведения бизнеса. Эффективное управление данными становится все более важным для компаний, особенно в сфере розничной торговли. Магазины одежды не являются исключением, их успешная деятельность напрямую зависит от умения эффективно хранить, управлять разнообразными данными, связанными с продажами, ассортиментом, клиентами и многими другими аспектами бизнеса.

Целью данной курсовой работы является разработка базы данных для хранения и обработки данных магазина одежды.

Для достижения поставленной цели необходимо решить следующие задачи:

1) проанализировать предметную область;
2) сформулировать требования и ограничения к разрабатываемой базе данных;
3) формализовать информацию, хранимую в БД;
4) разработать структуру базы данных и определить ролевую модель в контексте БД;
5) избрать инструменты для разработки и реализовать спроектированную базу данных.;
6) исследовать характеристики разработанного программного обеспечения.

1 Аналитическая часть

1.1 Анализ предметной области

В настоящее время магазины одежды играют важную роль в розничной торговле России. Они предоставляют широкий ассортимент товаров и являются основными местами для покупки одежды и аксессуаров для населения. Среди наиболее популярных онлайн-магазинов одежды в России следует выделить такие платформы, как stockmann.ru и lamoda.ru. Они зарекомендовали себя как надежные и удобные ресурсы для поиска и приобретения модной одежды и обуви.

Однако, несмотря на популярность и широкий выбор товаров, часто пользователи сталкиваются с проблемами в использовании этих онлайн-платформ. Один из основных аспектов, на который обращают внимание, является перегруженный интерфейс и сложность использования. Слишком много информации, разнообразные дополнительные функции и не всегда интуитивно понятный дизайн могут создавать неудобства для покупателей. По этой причине одной из задач курсовой работы заключается в разработке базы данных, которая позволит магазину одежды предоставить простой и понятный функционал для своих клиентов.

Разработка простого и понятного функционала для магазина одежды позволит повысить удовлетворенность клиентов и укрепить позиции магазина на рынке.

1.2 Требования к базе данных и приложению

Приложение должно предоставлять пользователям возможность искать товары по следующим параметрам:

1) пол человека;

2) категория;

3) бренд.

Также должна быть возможность сортировки товаров по возрастанию или убыванию цены.

Пользователи должны иметь возможность зарегистрироваться, авторизоваться, просмотреть информацию о товарах и брендах, добавить и удалить товары из корзины, совершить заказ, просмотреть историю заказов.

Администратор должен иметь возможность просматривать информацию о заказах пользователей, изменять информацию о пользователях, заказах, товарах и брендах, а также загружать и удалять новые товары и бренды.

1.3 Модели баз данных

1.3.1 Дореляционные базы данных

Дореляционные базы данных - это тип баз данных, который не использует традиционные таблицы и связи, характерные для реляционных баз данных. Вместо этого они организуют данные в более гибких структурах, таких как документы, графы или временные ряды. Это позволяет более эффективно хранить и обрабатывать данные с учетом их специфики. Примерами дореляционных баз данных могут служить NoSQL базы данных, такие как MongoDB, CouchDB, Cassandra.

Преимущества дореляционных баз данных включают более гибкую структуру данных, что делает их подходящими для работы с неструктурированными или полуструктурированными данными. Они также могут быть более масштабируемыми в определенных сценариях, таких как обработка больших объемов данных.

Однако дореляционные базы данных могут потребовать более сложных запросов и могут не обеспечивать те же уровни нормализации данных, что и реляционные базы данных.

1.3.2 Реляционные базы данных

Реляционные базы данных - это тип баз данных, который использует табличную структуру для хранения данных и устанавливает связи между этими таблицами. Каждая таблица представляет собой набор записей с атрибутами,

а связи между таблицами обеспечивают возможность объединения данных из разных таблиц при выполнении запросов.

Преимущества реляционных баз данных включают структурированный подход к хранению данных, что обеспечивает эффективность при работе с нормализованными данными. Они также обладают мощным языком запросов SQL, который позволяет легко извлекать и модифицировать данные.

Однако реляционные базы данных могут столкнуться с проблемами при работе с неструктурированными данными или данными, требующими гибких схем.

1.3.3 Постреляционные базы данных

Постреляционные базы данных - это новый подход к хранению данных, который пытается объединить преимущества как дореляционных, так и реляционных баз данных. Они предлагают гибкую схему данных, что делает их более способными к работе с разнообразными данными. Постреляционные базы данных также предлагают улучшенные механизмы обработки больших объемов данных.

Примеры постреляционных баз данных включают ArangoDB, Amazon DynamoDB и другие. Эти системы стремятся объединить гибкость дореляционных баз данных с мощью реляционных систем.

1.4 Информация, подлежащая хранению в базе данных

В базе данных нужно будет хранить информацию о пяти сущностях: пользователь, бренд, товар, заказ и позиция заказа.

1. Пользователь. Информация о зарегистрированных пользователях: логин, пароль, имя, пол, роль.

2. Бренд. Информация о брендах товаров: название, год основания, идентификатор логотипа, название компании владельца.

3. Товар. Информация о товаре: категория, размер, цена, пол, идентификатор

изображения, бренд, наличие.

4. Заказ. Информация о заказе: дата совершения, заказчик, цена, статус.

5. Позиция заказа. Информация о позиции в заказе: заказ, товар, количество.

Идентификаторы логотипа бренда и изображения товара представляют собой ID на сторонних сервисах для хранения изображений.

Так как база данных не имеет большого количества связей и хранимые в ней данные структурированы, для достижения поставленной цели была выбрана реляционная модель баз данных.

1.5 ER-диаграмма сущностей базы данных

На рисунке 1.1 показана диаграмма «сущность-связь» проектируемой базы данных в нотации Чена. Представлены пять сущностей и их свойства.

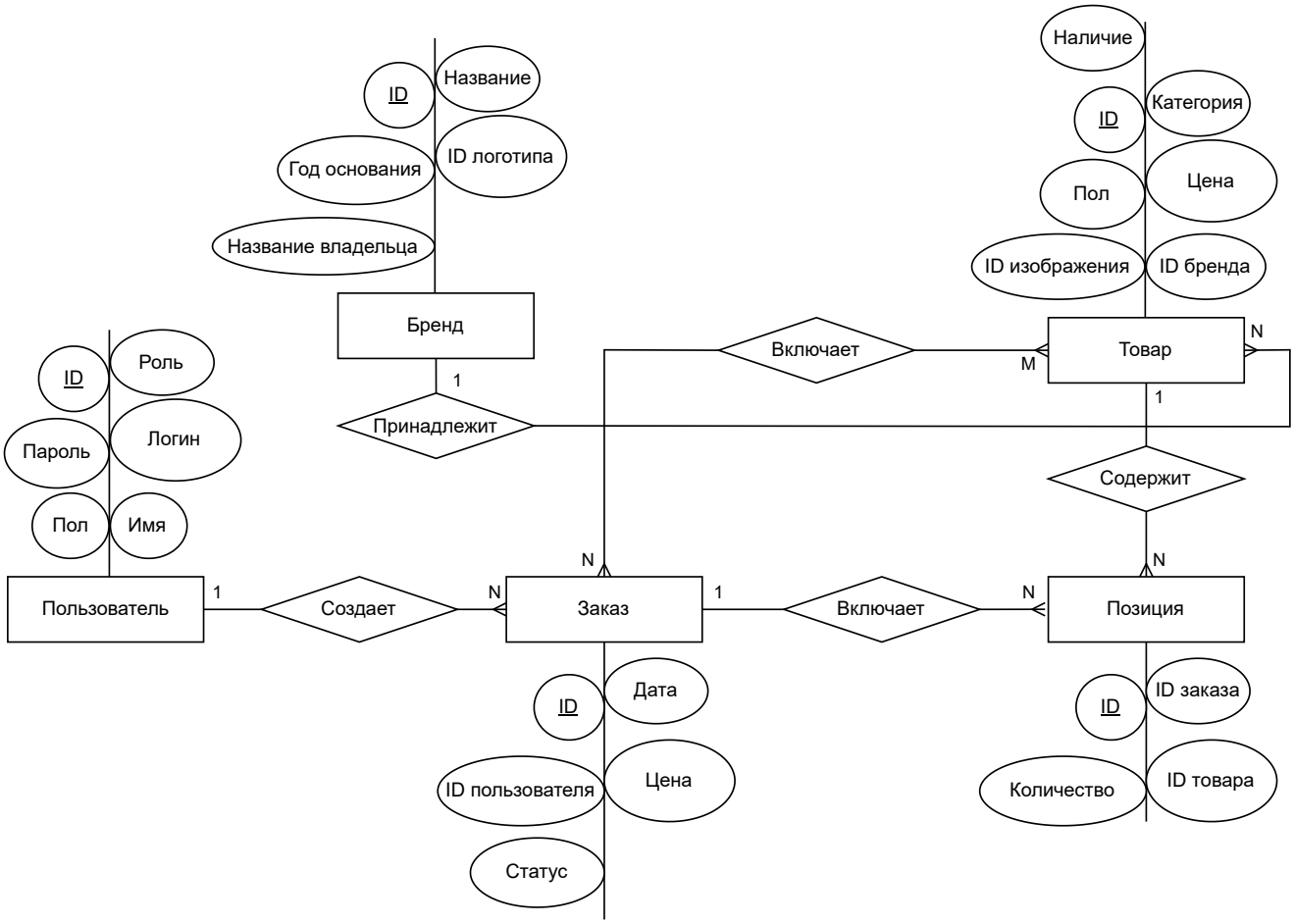

Рисунок 1.1 – ER-диаграмма

1.6 Пользователи приложения

Пользователи могут иметь одну из трех ролей: посетитель, клиент и администратор.

Посетитель — неавторизованный пользователь. Он может смотреть товары и бренды, авторизоваться и зарегистрироваться. На рисунке 1.2 представлена use-case диаграмма для роли посетитель.

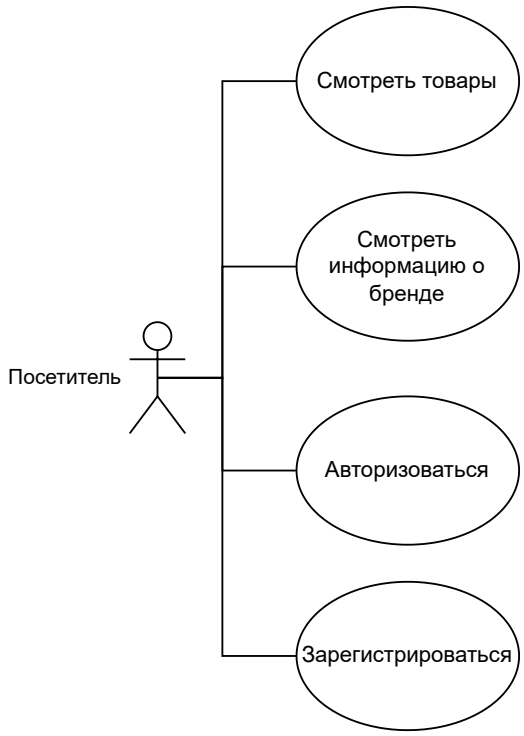

Рисунок 1.2 – Use-case диаграмма для роли посетитель

Клиент — зарегистрированный и авторизованный пользователь. Он имеет те же возможности что и посетитель (кроме регистрации), но в добавок он может добавить или удалить товары из корзины, посмотреть добавленные в корзину позиции, совершить заказ, просмотреть историю заказов. На рисунке 1.3 представлена use-case диаграмма для роли клиент.

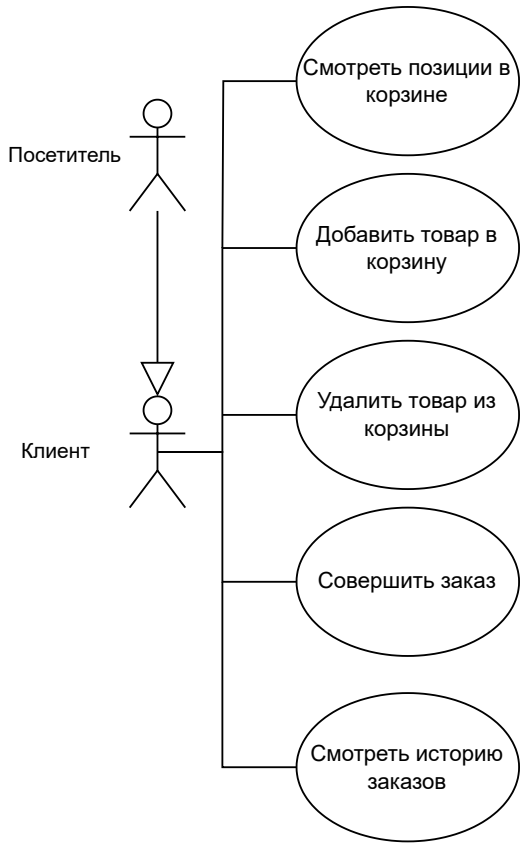

Рисунок 1.3 – Use-case диаграмма для роли клиент

Администратор имеет те же возможности что и клиент, но вдобавок может просматривать информацию о заказах пользователей, изменять информацию о пользователях, заказах, товарах и брендах, а также загружать и удалять новые товары и бренды. На рисунке 1.4 представлена use-case диаграмма для роли администратор.

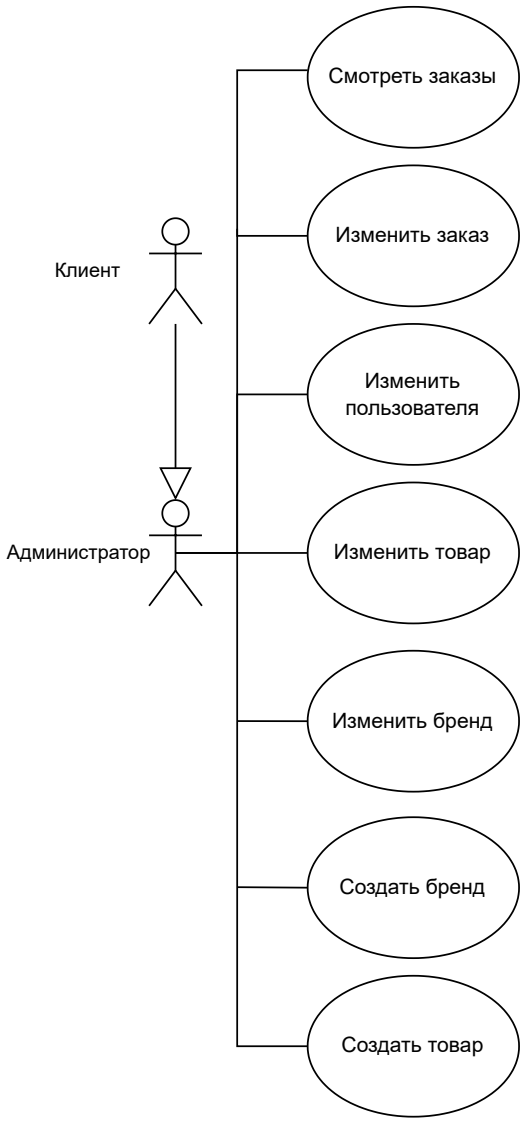

Рисунок 1.4 – Use-case диаграмма для роли администратор

Вывод из аналитической части

По ходу выполнения аналитической части данной курсовой работы была проанализирована рассматриваемая предметная область, а так же сформулированы требования и ограничения к разрабатываемой БД. Были рассмотрены модели баз данных и выбрана подходящая под поставленные задачи. Также была формализована информация, хранящиеся в БД и представлены диаграммы сущностей разрабатываемой базы данных и вариантов использования пользователями.

2 Конструкторская часть

2.1 Описание сущностей базы данных

На рисунке 2.1 представлена диаграмма базы данных.

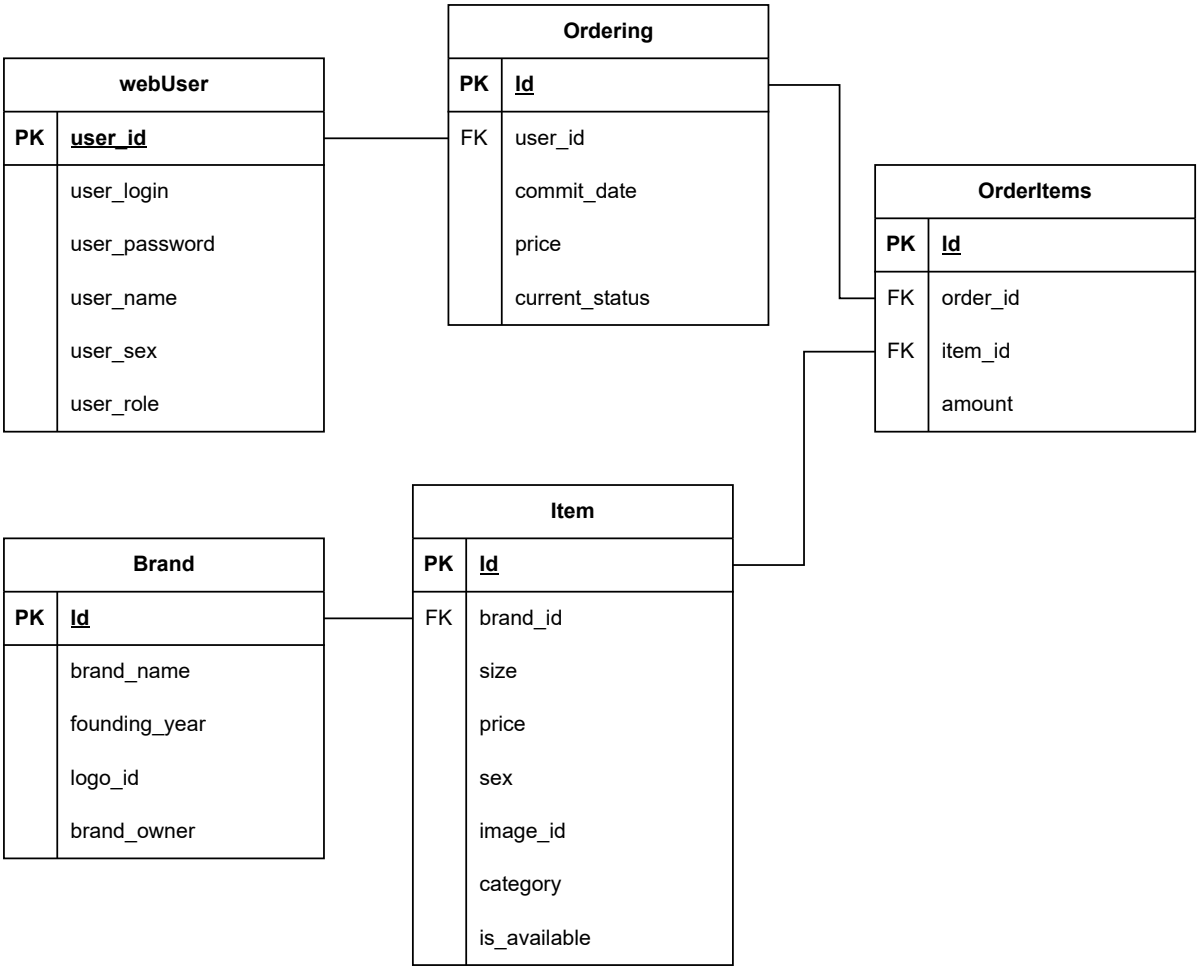

Рисунок 2.1 – Диаграмма базы данных

В таблицах 2.1–2.5 описаны атрибуты таблиц базы данных.

Таблица 2.1 – Атрибуты таблицы «Пользователи»

Имя атрибута	**Тип атрибута**	**Описание**
user_id	целое	Идентификатор — первичный ключ
user_login	строка	Логин пользователя
user_password	строка	Пароль пользователя
user_name	строка	Имя пользователя
user_sex	строка	Пол пользователя
user_role	строка	Роль пользователя

Таблица 2.2 – Атрибуты таблицы «Бренды»

Имя атрибута	**Тип атрибута**	**Описание**
id	целое	Идентификатор — первичный ключ
brand_name	строка	Название бренда
founding_year	целое	Год основания
logo_id	целое	Идентификатор логотипа в стороннем сервисе
brand_owner	строка	Название владельца бренда

Таблица 2.3 – Атрибуты таблицы-связки «Товары»

Имя атрибута	**Тип атрибута**	**Описание**
id	целое	Идентификатор — первичный ключ
category	строка	Категория товара
size	строка	Размер товара
price	целое	Цена
sex	строка	Пол товара
image_id	целое	Идентификатор изображения товара в стороннем сервисе
brand_id	целое	Идентификатор бренда — внешний ключ
is_available	логический тип	Наличие

Таблица 2.4 – Атрибуты таблицы «Заказы»

Имя атрибута	**Тип атрибута**	**Описание**
id	целое	Идентификатор — первичный ключ
commit_date	дата	Дата совершения заказа
user_id	целое	Идентификатор пользователя — внешний ключ
price	целое	Суммарная цена товаров в заказе
current_status	строка	Статус заказа

Таблица 2.5 – Атрибуты таблицы «Позиции заказов»

Имя атрибута	**Тип атрибута**	**Описание**
id	целое	Идентификатор — первичный ключ
order_id	целое	Идентификатор заказа — внешний ключ
item_id	целое	Идентификатор товара — внешний ключ
amount	целое	Количество товаров

2.2 Описание ограничений целостности базы данных

Для обеспечения целостности базы данных, на хранящуюся в ней информацию должны быть наложены ограничения.

Ограничения для таблицы «Пользователи». Идентификатор — уникальное положительное число, первичный ключ. Логин, пароль, имя, пол и роль пользователя не должны отсутствовать.

Ограничения для таблицы «Бренды». Идентификатор — уникальное положительное число, первичный ключ. Имя бренда, год основания, идентификатор логотипа и название владельца не должны отсутствовать, также год основания должен находиться в интервале от 1500 до 2024.

Ограничения для таблицы «Товары». Идентификатор — уникальное положительное число, первичный ключ. Идентификатор бренда — положительное число, внешний ключ, не должен отсутствовать. Категория, размер, цена, пол, идентификатор изображения, наличие не должны отсутствовать, также цена должна быть положительной.

Ограничения для таблицы «Заказы». Идентификатор — уникальное положительное число, первичный ключ. Идентификатор пользователя — положительное число, внешний ключ, не должен отсутствовать. Дата совершения, цена и статус не должны отсутствовать. Цена должна быть положительной.

Ограничения для таблицы «Позиции заказов». Идентификатор — уникальное положительное число, первичный ключ. Идентификаторы заказа и товара —

положительные числа, внешние ключи, не должны отсутствовать. Количество не должно отсутствовать и быть положительным.

2.3 Описание проектируемой функции на уровне базы данных

Пользователь может смотреть позиции, добавленные в корзину, для этой цели удобно реализовать функцию, которая по идентификатору пользователя будет возвращать позиции в корзине.

На рисунке 2.2 представлена схема проектируемой функции.

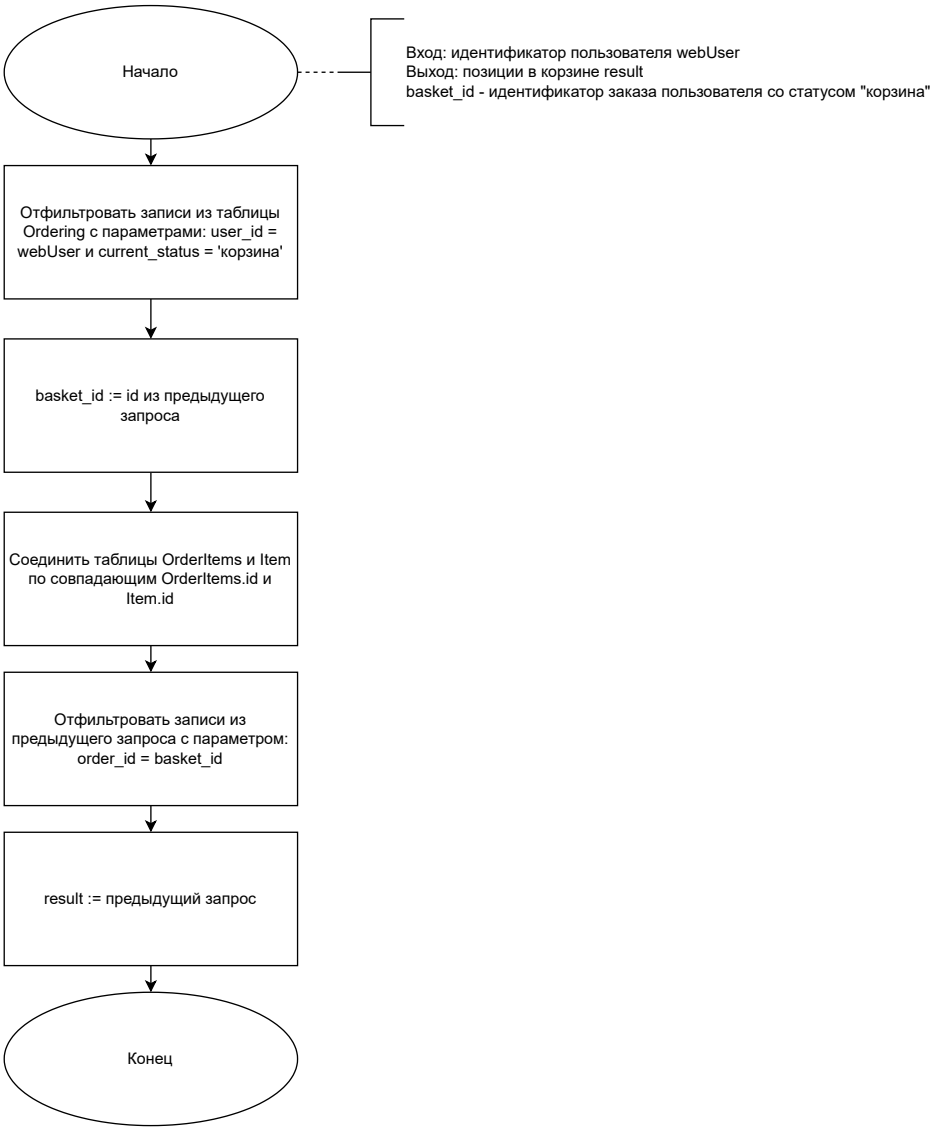

Рисунок 2.2 – Функция, возвращающая позиции в корзине

В функции используется переменная basket_id — идентификатор заказа пользователя со статусом «корзина».

2.4 Описание ролевой модели на уровне базы данных

Для работы с базой данных было предусмотрено создание трех ролей: посетитель, клиент и администратор.

Посетитель имеет доступ к просмотру информации о таких объектах базы данных, как товар и бренд.

Клиент имеет доступ к просмотру информации о товарах, брендах, заказах и позициях в заказах.

Администратор имеет возможность просматривать, добавлять, изменять и удалять информацию о любом объекта БД.

Вывод из конструкторской части

В ходе выполнения конструкторской части данной курсовой работы было представлено описание сущностей разрабатываемой базы данных и представлена соответствующая диаграмма; описаны атрибуты таблиц рассматриваемой БД; описаны ограничения накладываемые на информацию хранимую базе данных для обеспечения ее целостности; описана проектируемая функция и ролевая модель на уровне базы данных.

3 Технологическая часть

3.1 Средства реализации

Для разработки веб-сервера в рамках данного проекта был выбран язык программирования Golang, и данный выбор был обоснован по следующим причинам:

1. Простота и легкость в освоении: Go обладает синтаксисом, который легко понимать и осваивать, что способствует быстрой адаптации разработчиков к этому языку программирования;

2. Высокая производительность: Go предоставляет возможность создания высокопроизводительных приложений за счет низкого уровня абстракции;

3. Разнообразная экосистема: Go обладает обширной библиотекой стандартных пакетов и доступом к множеству сторонних библиотек и фреймворков;

4. Надежность: Go был создан Google для разработки надежных и стабильных приложений, что обеспечивает его высокую надежность.

В качестве системы управления базами данных была выбрана PostgreSQL по причине того, что данная СУБД часто используется в проектах на Golang, в следствие чего существует большое разнообразие библиотек и документации для работы с PostgreSQL на Go.

3.2 Реализация сущностей базы данных

В листингах 3.1–3.5 показана реализация сущностей базы данных.

Листинг 3.1 – Сущность «Пользователь»

```
type User struct {
    ID       int
    Login    string
    Password string
    Name     string
    Sex      string
    Role     string
```

```
}
```

Листинг 3.2 – Сущность «Бренд»

```
type Brand struct {
    ID    int
    Name  string
    Year  int
    Logo  int
    Owner string
}
```

Листинг 3.3 – Сущность «Товар»

```
type Item struct {
    ID          int
    Category    string
    Size        string
    Price       int
    Sex         string
    ImageID     int
    BrandID     int
    IsAvailable bool
}
```

Листинг 3.4 – Сущность «Заказ»

```
type Order struct {
    ID     int
    Date   time.Time
    UserID int
    Items  []OrderItem
    Price  int
    Status string
}
```

Листинг 3.5 – Сущность «Позиция заказа»

```
type OrderItem struct {
    Item
    Amount int
}
```

3.3 Реализация ограничений целостности базы данных

В листингах 3.6–3.10 показана реализация ограничений целостности базы данных (ограничения задаются при создании таблиц).

Листинг 3.6 – Ограничения для таблицы «Пользователи»

```
create table public.webUser(
  user_id serial not null primary key,
  user_login text not null unique,
  user_password text not null,
  user_name text not null,
  user_sex text not null,
  user_role text not null
);
```

Листинг 3.7 – Ограничения для таблицы «Бренды»

```
create table public.Brand(
  id serial not null primary key,
  brand_name text not null,
  founding_year int not null check (
    founding_year > 1500
    and founding_year < 2024
  ),
  logo_id int not null,
  brand_owner text not null
);
```

Листинг 3.8 – Ограничения для таблицы «Товары»

```
create table public.Item(
  id serial not null primary key,
  category text not null,
  size text not null,
  price int not null check (price > 0),
```

```
  sex text not null,
  image_id int not null,
  brand_id int not null,
  foreign key (brand_id) references public.Brand(id),
  is_available boolean
);
```

Листинг 3.9 – Ограничения для таблицы «Заказы»

```
create table public.Ordering(
  id serial not null primary key,
  commit_date date,
  user_id int not null,
  foreign key (user_id) references public.webUser(user_id),
  price int check (price > 0),
  current_status text not null
);
```

Листинг 3.10 – Ограничения для таблицы «Позиции заказов»

```
create table public.OrderItems(
  id serial not null primary key,
  order_id int not null,
  foreign key (order_id) references public.Ordering(id),
  item_id int not null,
  foreign key (item_id) references public.Item(id),
  amount int not null check (amount > 0)
);
```

3.4 Реализация функции на уровне базы данных

В листинге 3.11 показана реализация функции на уровне базы данных, которая по идентификатору пользователя возвращает позиции в его корзине.

Листинг 3.11 – Функция, возвращающая позиции в корзине пользователя по его идентификатору

```
CREATE
OR REPLACE FUNCTION ItemsInUsersBasket(webUser int) RETURNS TABLE (
  id int,
  category text,
  size text,
  price int,
  sex text,
  image_id int,
  brand_id int,
  is_available boolean,
  amount int
) AS $$ declare basket_id int;
BEGIN
select
  o.id into basket_id
from
  Ordering o
where
  o.user_id = webUser
  and current_status = 'корзина';
return query
SELECT
  i.id,
  i.category,
  i.size,
  i.price,
  i.sex,
  i.image_id,
  i.brand_id,
  i.is_available,
  o.amount
FROM
```

```
  OrderItems o
  JOIN Item i ON o.item_id = i.id
where
  o.order_id = basket_id;
END $$ LANGUAGE plpgsql;
```

3.5 Реализация ролевой модели на уровне базы данных

В листинге 3.12 представлена реализация ролевой модели на уровне базы данных.

Листинг 3.12 – Ролевая модель на уровне базы данных

```
create user "default_guest";
create user "default_user";
create user "default_admin";
alter role "default_guest" password '00000000';
alter role "default_user" password '11111111';
alter role "default_admin" password '22222222';
grant
select
  on table Item to "default_guest";
grant
select
  on table Brand to "default_guest";
grant
select
  on table Item to "default_user";
grant
select
  on table Ordering to "default_user";
grant
select
  on table Brand to "default_user";
grant
select
  on table OrderItems to "default_user";
alter role "default_admin" superuser;
```

3.6 Примеры работы

Для примера работы разработанного ПО были выполнены следующие запросы:

1. регистрация пользователя;
2. авторизация пользователя;
3. получение списка товаров;
4. добавление товара в корзину;
5. попытка создать новый товар клиентом.

На рисунках 3.1–3.5 представлены примеры запросов к серверу и результаты их обработки.

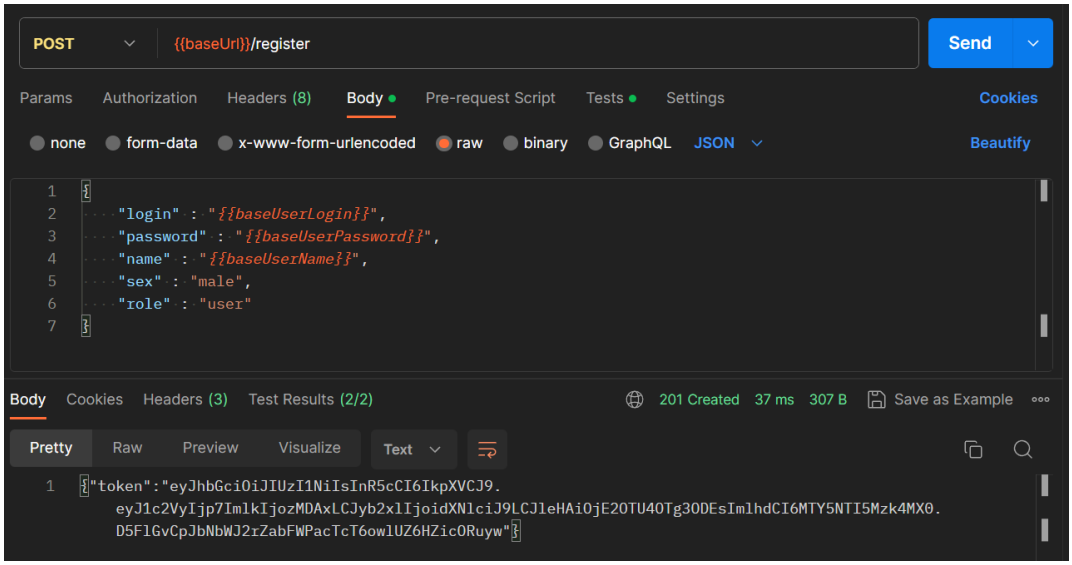

Рисунок 3.1 – Пример регистрации пользователя

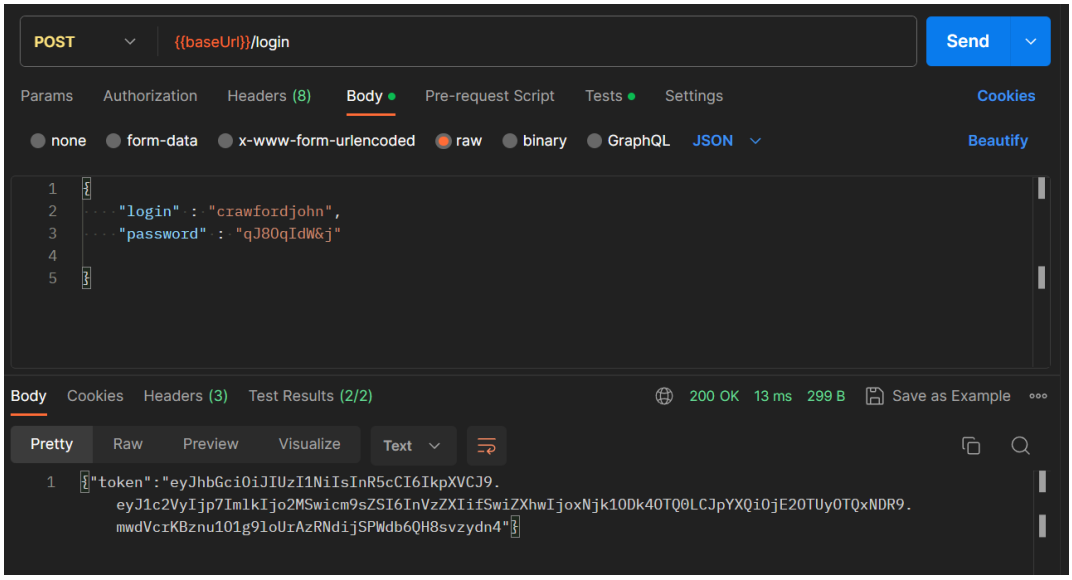

Рисунок 3.2 – Пример авторизации пользователя

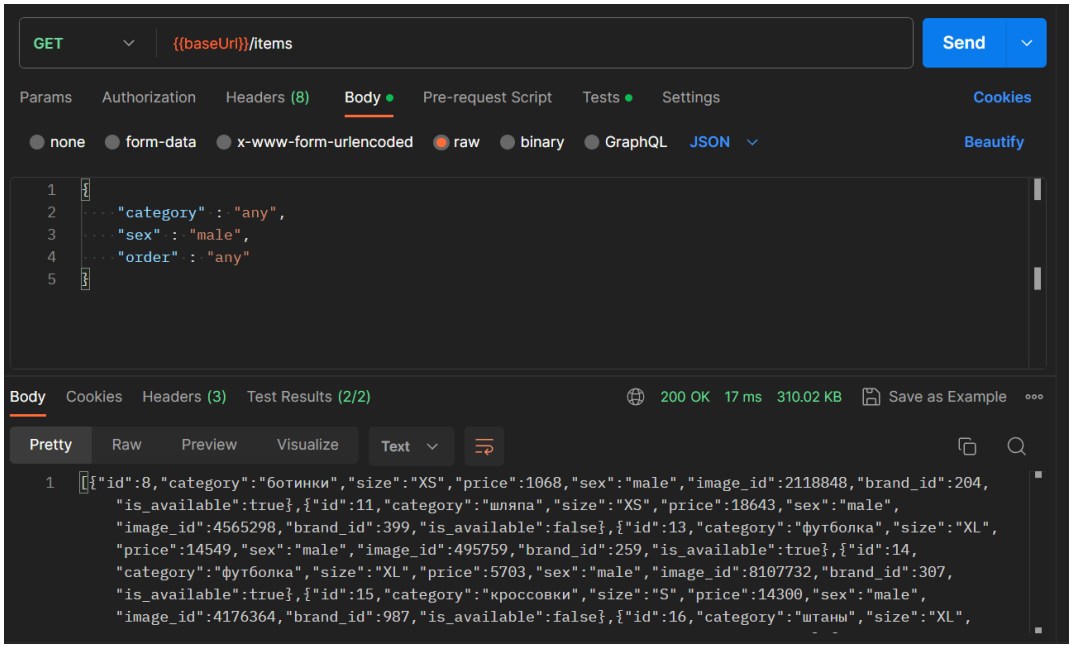

Рисунок 3.3 – Пример запроса на получение списка товаров

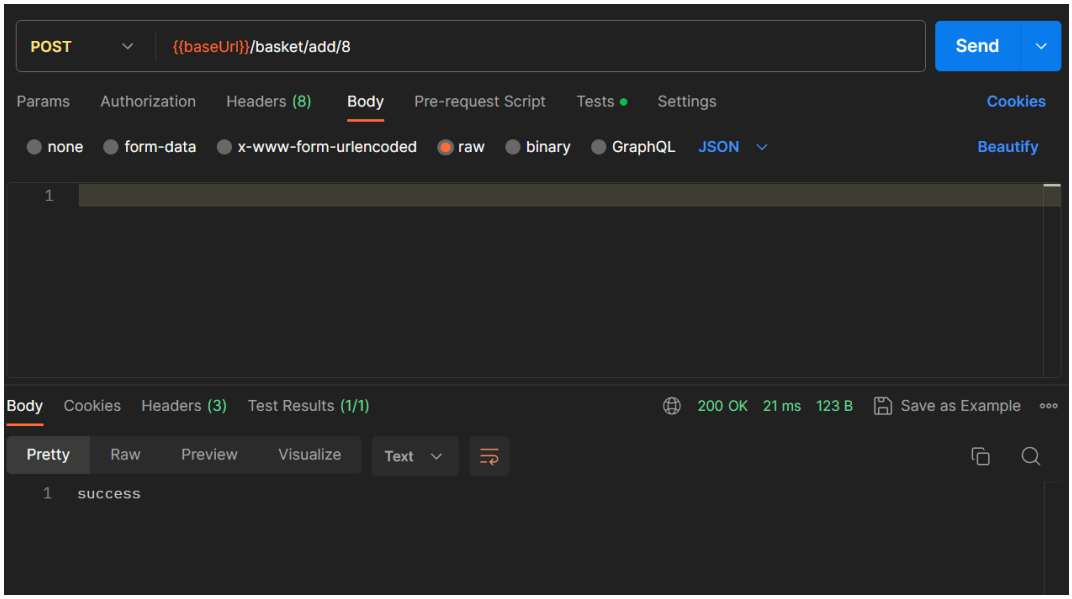

Рисунок 3.4 – Пример запроса на добавление товара в корзину

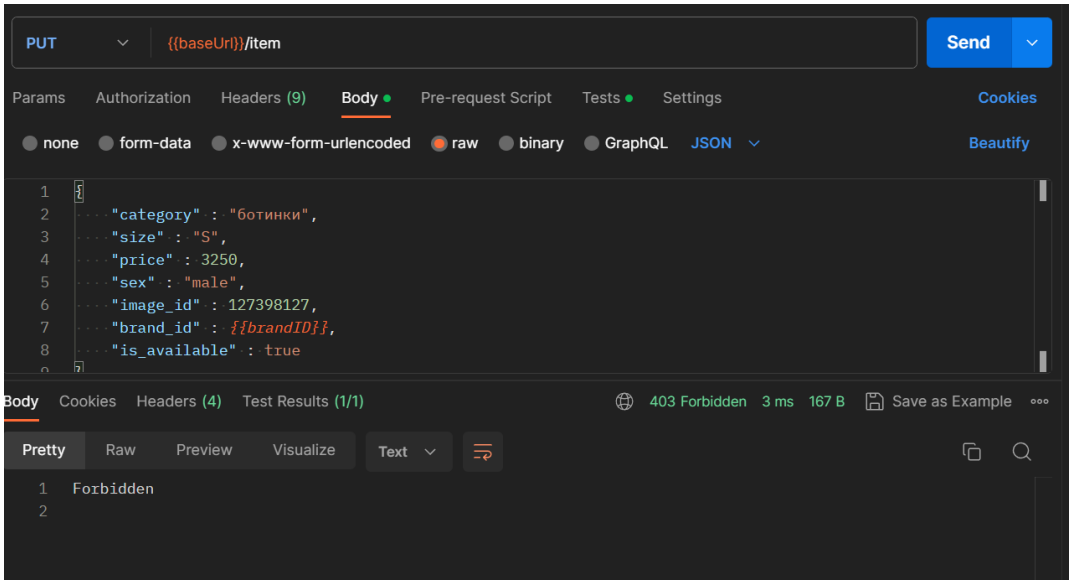

Рисунок 3.5 – Пример запроса на создание нового товара клиентом

Вывод из технологической части

В ходе выполнения технологической части данной курсовой работы были выбраны средства реализации разрабатываемого ПО; представлены листинги реализаций сущностей и ограничений целостности БД; представлены листинги реализации функции и ролевой модели на уровне базы данных, а также продемонстрированы примеры работы программы на различных запросах.

4 Исследовательская часть

4.1 Технические характеристики устройства

Технические характеристики устройства, на котором было проведено измерение времени выполнения запросов:

1) операционная система Windows 10 Pro x64;
2) оперативная память 16 ГБ;
3) процессор Intel® Core™ i7-4790K @ 4.00 ГГц.

Измерение времени выполнения запросов проводилось на стороне сервера с помощью класса Stopwatch, который предоставляет набор методов и свойств, которые можно использовать для точного измерения затраченного времени [**Stopwatch**].

4.2 Исследование характеристик разработанного программного обеспечения

Индекс — это объект базы данных, обеспечивающий дополнительные способы быстрого поиска и извлечения данных. Индекс может создаваться на одном (простой индекс) или нескольких (составной индекс) атрибутах. Если в таблице нет индекса, то поиск нужных строк выполняется простым сканированием по всей таблице. При наличии индекса время поиска нужных строк можно существенно уменьшить [**Gavrilova2022**].

Индексирование не влияет на размещение данных какой-либо таблицы в базе данных. Ускорение поиска данных через индекс обеспечивается за счет упорядочивания значений индексируемого атрибута (что позволяет просматривать в среднем половину индекса при линейном поиске) и за счет того, что индекс занимает меньше страниц памяти, чем сама таблица, поэтому система тратит меньше времени на чтение индекса, чем на чтение таблицы [**Karpova2009**].

В Microsoft SQL Server индексы хранятся в виде сбалансированных деревьев. Представление индекса в виде сбалансированного дерева означает, что стоимость

поиска любой строки остается относительно постоянной, независимо от того, где находится эта строка [**Gavrilova2022**].

Таблица или представление может иметь индексы кластеризованные и некластеризованные. Кластеризованные индексы сортируют и хранят строки данных в таблицах или представлениях на основе их ключевых значений. Этими значениями являются столбцы, включенные в определение индекса. Существует только один кластеризованный индекс для каждой таблицы, так как строки данных могут храниться в единственном порядке. Строки данных в таблице хранятся в порядке сортировки только в том случае, если таблица содержит кластеризованный индекс. Некластеризованные индексы имеют структуру, отдельную от строк данных. В некластеризованном индексе содержатся значения ключа некластеризованного индекса, и каждая запись значения ключа содержит указатель на строку данных, содержащую значение ключа. В Microsoft SQL Server индексы создаются автоматически при определении ограничений PRIMARY KEY или UNIQUE на основе столбцов таблицы [**Indices**].

Во всех таблицах базы данных, которую использует реализованное в ходе выполнения курсовой работы приложение, есть ограничение PRIMARY KEY, соответственно, Microsoft SQL Server автоматически создает кластеризованные индексы. В таблице «Пользователи» есть уникальный атрибут «почта», для которого СУБД создает некластеризованный индекс. Имеет смысл вручную создать некластеризованный составной индекс по атрибутам «пункт вылета», «пункт прибытия» и «дата вылета» для таблицы «Рейсы» для увеличения скорости поиска рейсов.

Кроме того, можно вручную создать кластеризованный и некластеризованный индекс по атрибуту «ID пользователя» в таблице «Заказы» и измерить время выполнения запроса поиска заказа по идентификатору пользователя без индекса, с кластеризованным и некластеризованным индексом.

4.3 Время выполнения запроса

В листинге 4.1 приведены команды создания кластеризованного и некластеризованного индекса для таблицы «Заказы».

Листинг 4.1 – Создание кластеризованного и некластеризованного индекса

```
CREATE CLUSTERED INDEX user_id_index ON Orders (UserId);
CREATE NONCLUSTERED INDEX user_id_index ON Orders (UserId);
```

В таблицах 4.1–4.2 приведено время выполнения в миллисекундах запроса без индекса, с кластеризованным и некластеризованным индексом при поиске первой и последней записи в таблице соответственно.

Таблица 4.1 – Время выполнения запроса при поиске первой записи в таблице

Кол-во строк	**Время выполнения запроса без индекса, мс**	**Время выполнения запроса с кластеризованным индексом, мс**	**Время выполнения запроса с некластеризованным индексом, мс**
100	468	440	450
1000	471	421	439
10000	510	425	445
50000	757	414	453
100000	1642	472	460
250000	10701	498	510

Таблица 4.2 – Время выполнения запроса при поиске последней записи в таблице

Кол-во строк	**Время выполнения запроса без индекса, мс**	**Время выполнения запроса с кластеризованным индексом, мс**	**Время выполнения запроса с некластеризованным индексом, мс**
100	495	442	461
1000	644	448	452
10000	715	457	470
50000	2136	440	445
100000	5978	498	510
250000	50571	524	520

На рисунках 4.1–4.2 показана зависимость времени выполнения запроса в миллисекундах от количества строк в таблице.

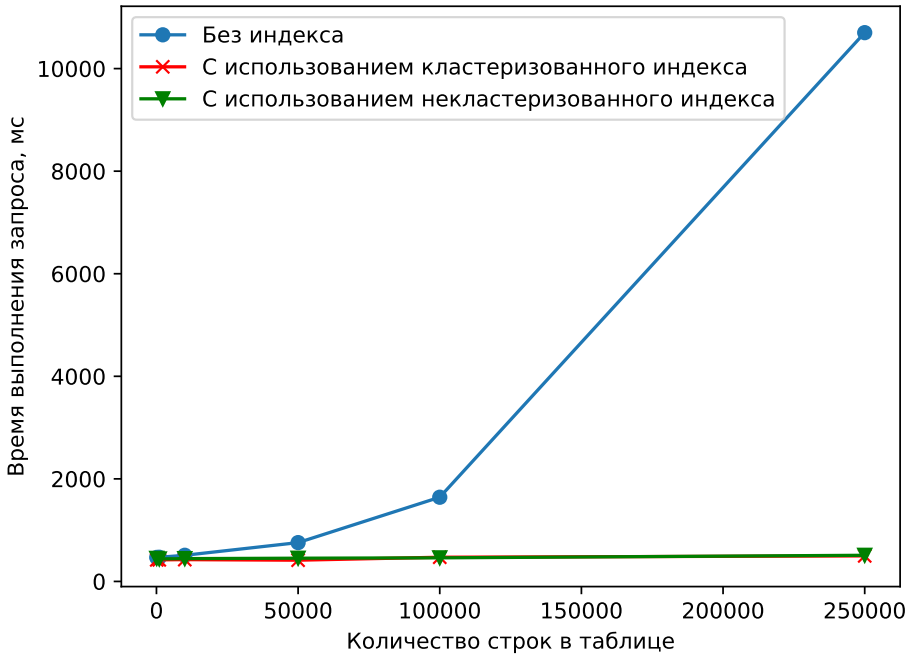

Рисунок 4.1 – Зависимость времени выполнения запроса от количества строк в таблице при поиске первой записи

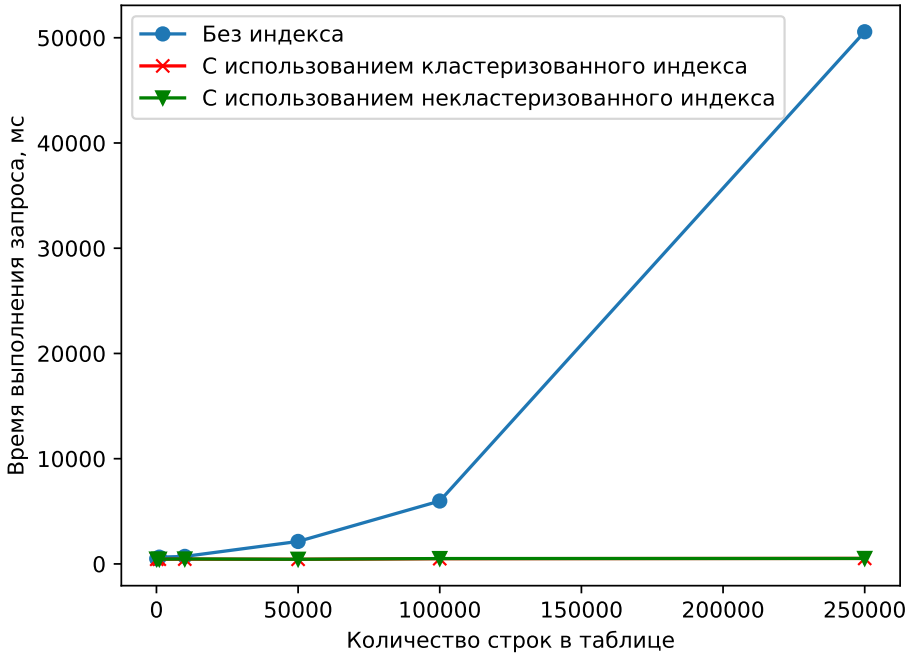

Рисунок 4.2 – Зависимость времени выполнения запроса от количества строк в таблице при поиске последней записи

Вывод из исследовательской части

В ходе выполнения исследовательской части курсовой работы был проведен краткий обзор кластеризованных и некластеризованных индексов; определены таблицы и атрибуты, для которых имеет смысл создавать индексы, и проведен эксперимент по измерению времени выполнения запроса без индекса, с кластеризованным и некластеризованным индексом. Согласно полученным при проведении эксперимента данным, при количестве строк в таблице до 10 тысяч существенной разницы во времени выполнения запроса не наблюдается. При большем количестве записей время выполнения запроса без использования индекса растет в разы быстрее, а время выполнения запроса с использованием двух типов индексов остается практически неизменным. Разница во времени выполнения запроса с кластеризованным и некластеризованным индексом практически отсутствует. При 250 тысячах строк в таблице использование индекса увеличивает скорость выполнения запроса в 21 раз при поиске первой записи и в 97 раз при поиске последней.

ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы была разработана база данных для хранения и обработки данных авиакомпании, а также веб-приложение, которое ее использует.

Были выполнены следующие задачи:

1) проведен обзор существующих веб-приложений для покупки авиабилетов и сформулированы требования и ограничения к разрабатываемой базе данных и приложению;

2) спроектирована архитектура базы данных, ограничения целостности и ролевая модель на уровне базы данных;

3) выбраны средства реализации и реализованы спроектированная база данных и необходимый интерфейс для взаимодействия с ней;

4) исследованы характеристики разработанного программного обеспечения.

В ходе проведения эксперимента при выполнении исследовательской части курсовой работы было установлено, что при 250 тысячах строк в таблице использование и кластеризованного, и некластеризованного индекса увеличивает скорость выполнения запроса в 21 раз при поиске первой записи и в 97 раз при поиске последней.

ПРИЛОЖЕНИЕ А

Презентация