РЕФЕРАТ

Отчет 29 с., 9 рис., 10 табл., 8 источн., 1 прил.

Ключевые слова: Базы данных, PostgreSQL, SQL, Java.

Цель работы — разработка базы данных для хранения и обработки данных музыкального магазина.

В данной работе проведено сравнение существующих моделей данных, формализованы данные, реализована ролевая модель на уровне базы данных, спроектирована и реализована база данных и интерфейс для взаимодействия с ней, проведено исследование влияния индексов на время выполнения запросов на выборку.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Российский рынок музыкальных инструментов потенциально является одним из самых крупных в Европе, о чем свидетельствует стабильный рост музыкальной индустрии в течение последних лет. Ежегодный рост объема продаж музыкальных инструментов и оборудования по данным агентства «Бизнес Монитор», составляет 20–30 % [1]. В связи с этим, появляется спрос на программное обеспечение, способное автоматизировать работу музыкальных магазинов.

Цель курсовой работы — разработка базы данных для хранения и обработки данных музыкального магазина.

Для достижения цели необходимо выполнить следующие задачи:

— провести анализ существующих решений;

— сформулировать описание пользователей приложения;

— формализовать данные;

— провести анализ моделей данных в базах данных;

— спроектировать архитектуру базы данных и ограничения целостности;

— спроектировать ролевую модель на уровне базы данных;

— выбрать средства реализации;

— реализовать спроектированную БД и необходимый интерфейс для взаимодействия с ней;

— сравнить время выполнения запросов на выборку при наличии индекса в базе данных и без него.

1 Аналитический раздел

1.1 Анализ существующих решений

Сформулируем критерии сравнения существующих решений:

— К1: наличие программы лояльности;

— К2: возможность доставки;

— К3: возможность отслеживать текущий статус заказа;

— К4: доступность в РФ.

Результаты сравнения существующих решений с разрабатываемым приведены в таблице 1.1.

Таблица 1.1 – Сравнение существующих решений

Решение	К1	К2	К3	К4
Музторг	+	+	+	+
Мир музыки	-	+	+	+
Music Store	+	+	+	-
Разрабатываемое решение	+	+	+	+

Таким образом, разрабатываемое решение не уступает существующим по всем критериям.

1.2 Описание пользователей приложения

Требуется реализовать четыре вида ролей — неавторизованный пользователь, заказчик, сотрудник магазина и администратор.

Действия неавторизованного пользователя:

— войти/зарегистрироваться;

— посмотреть ассортимент товаров;

— посмотреть информацию о конкретном товаре.

Действия заказчика:

— выйти;

— посмотреть личные данные;

— посмотреть ассортимент товаров;

— посмотреть информацию о конкретном товаре;

— добавить товар в корзину;

— оформить заказ;

— посмотреть историю заказов.

Действия сотрудника:

— выйти;

— посмотреть личные данные;

— посмотреть ассортимент товаров;

— посмотреть информацию о конкретном товаре;

— добавить товар в корзину;

— оформить заказ;

— изменить статус заказа;

— изменить количество товара на складе;

— посмотреть историю продаж.

Действия администратора:

— выйти;

— посмотреть ассортимент товаров;

— посмотреть информацию о конкретном товаре;

— изменить количество товара на складе;

— добавить товар;

— добавить сотрудника.

На рисунке 1.1 представлена диаграмма прецедентов для разных пользователей.

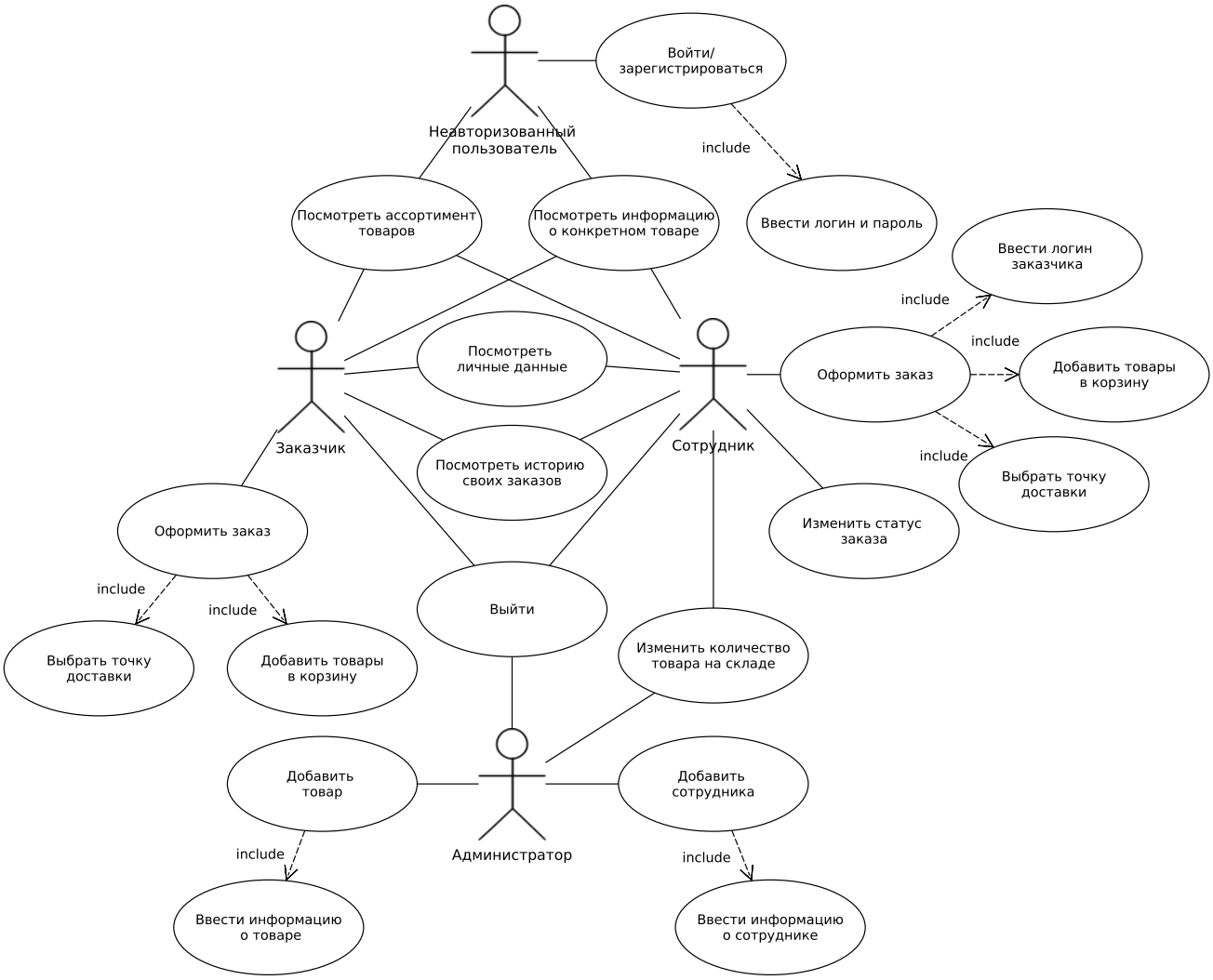

Рисунок 1.1 – Диаграмма прецедентов

1.3 Формализация данных

База данных должна хранить информацию о:

— пользователях;

— товарах;

— производителях;

— заказах;

— бонусных картах;

— точках доставки.

В таблице 1.2 приведена информация о необходимых сущностях и их характеристиках.

Таблица 1.2 – Сущности и характеристики

сущность	характеристики
пользователь	логин, пароль, email, ID бонусной карты, роль
товар	ID, название, цена, количество на складе, описание, цвет, производитель, ссылка на изображение
заказ	ID, логин заказчика, логин сотрудника, дата, статус, стоимость, количество потраченных бонусов, ID точки доставки
единица заказа	ID заказа, ID товара, количество товара, стоимость на момент заказа
точка доставки	ID, адрес
бонусная карта	ID, количество бонусов

На рисунке 1.2 отображена ER-диаграмма системы, основанная на приведенной выше таблице.

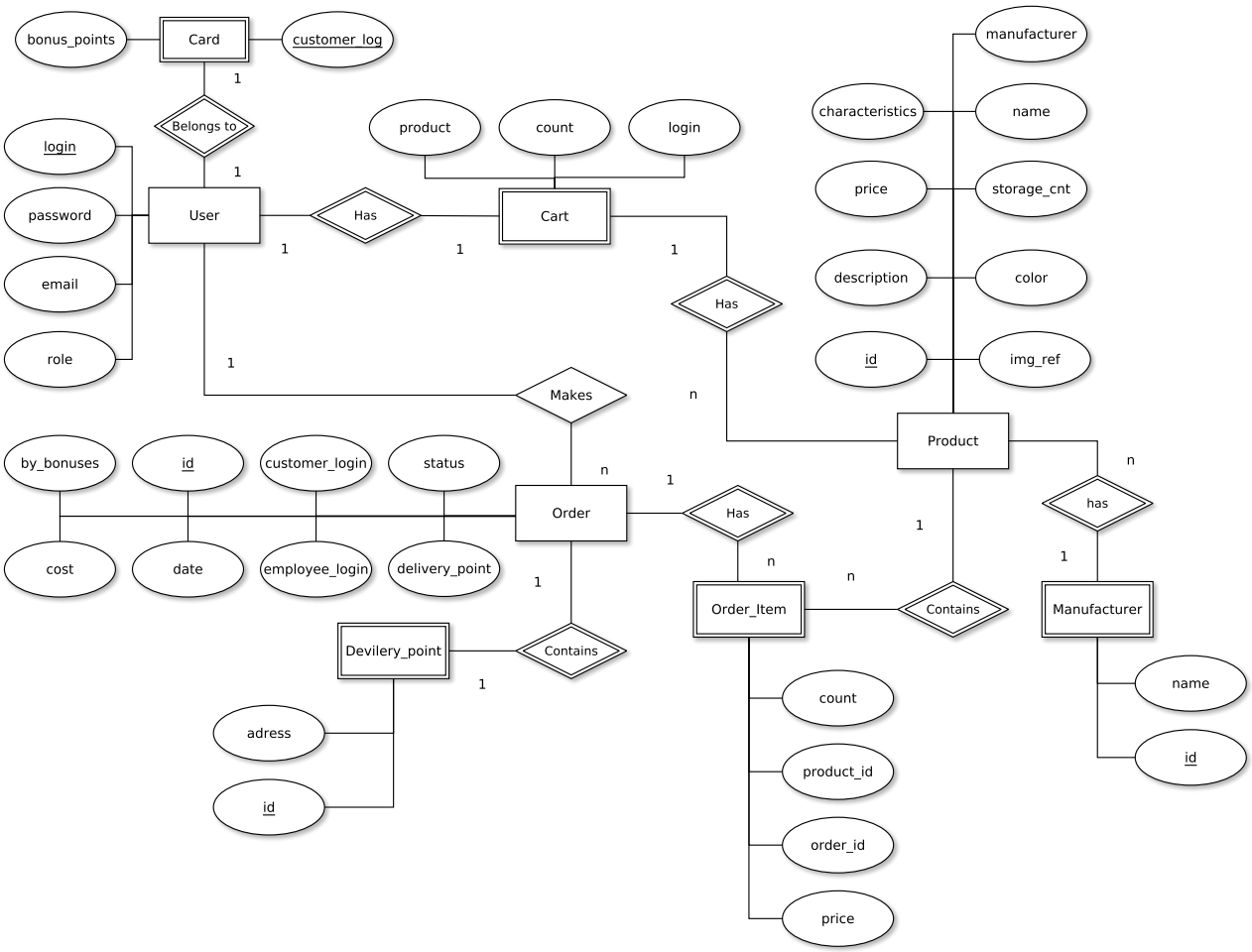

Рисунок 1.2 – ER-диаграмма

1.4 Выбор модели данных

1.4.1 Иерархическая модель

В иерархической модели данные представляются в виде древовидной структуры. Каждый объект в этой модели может являться предком по отношению к одному или нескольким дочерним узлам. При этом у дочернего узла может быть только один предок. Если удаляется предкок, то удаляются все дочерние узлы. Особенностями модели являются быстрый доступ к данным за счет древовидной структуры и невозможность реализовать связь «многие ко многим» [2].

1.4.2 Сетевая модель

Сетевая модель данных основана на графовой структуре. Отличие от иерархической модели состоит в том, что в сетевой структуре данных у дочернего узла может быть любое число предков. Данная модель требует заранее определенной структуры данных, что может быть неудобно при изменениях в схеме базы данных. К особенностям сетевой модели можно отнести возможность реализовать сложные связи между данными и зависимость времени доступа к данным от их физической организации [2].

1.4.3 Реляционная модель

Реляционная модель данных основана на понятии отношения — информационной модели реального объекта предметной области, формально представленной множеством однотипных кортежей. Кортеж отношения соответствует экземпляру объекта, свойства которого определяются значениями соответствующих атрибутов (полей) кортежа. Кортежи отношений могут быть связаны между собой с помощью внешних ключей — ссылок на соответствующие атрибуты [2].

Реляционная модель состоит из трех частей: структурной, целостностной и манипуляционной. Структурная часть реляционной модели описывает, из каких объектов состоит реляционная модель. Целостная часть определяет базовые требования целостности. Манипуляционная часть описывает способы манипулирования данными.

1.4.4 Модель ключ-значение

В этой модели данные хранятся как совокупность пар ключ-значение, в которых ключ является уникальным и используется для поиска, добавления или удаления соответствующего значения. При этом ключи могут являться сложными объектами. В данной модели отсутствуют взаимосвязи между объектами, ограничения целостности. Поиск по значениям неключевых атрибутов требуется выполнять на уровне приложения [3].

1.4.5 Документо-ориентированная модель

Базовая структурная единица документо-ориентированной модели — это документ. Схожие документы могут относиться к одной коллекции. Если в реляционной модели все кортежи обладают идентичной структурой, то в документо-ориентированной базе данных все документы могут быть произвольными. При использовании этой модели отсутствуют затраты на сборку данных из нескольких таблиц в единое целое для формирования результата запроса [3].

Вывод

Для решения поставленной задачи была выбрана реляционная модель данных, так как она обеспечивает целостность на уровне базы данных и позволяет организовать связь «многие ко многим».

2 Конструкторский раздел

2.1 Описание сущностей базы данных

Обозначения в таблицах: PK — первичный ключ, FK — внешний ключ, U — поле должно быть уникальным, NN — поле обязательно должно быть заполнено.

Описание сущностей базы данных приведено в таблицах 2.1–2.8.

Таблица 2.1 – Таблица User

поле	тип данных	ограничения	значение
login	text	PK	логин
password	text	NN	пароль
email	text	U	электронная почта
role	text	NN	роль

Таблица 2.2 – Таблица Card

поле	тип данных	ограничения	значение
customer_login	text	FK, NN	логин заказчика
bonuses	text	NN	количество бонусов

Таблица 2.3 – Таблица Manufacturer

поле	тип данных	ограничения	значение
id	uuid	PK	идентификатор
name	text	NN	имя

Таблица 2.4 – Таблица Product

поле	тип данных	ограничения	значение
id	text	PK	идентификатор
name	text	NN	название
price	int	NN	цена
description	text		описание
color	text		цвет
storage_cnt	int	NN	количество на складе
img_ref	text		ссылка на изображение
manufacturer_id	uuid	FK, NN	идентификатор производителя
characteristics	json		характеристики

Таблица 2.5 – Таблица DeliveryPoint

поле	тип данных	ограничения	значение
id	uuid	PK	идентификатор
address	text	U, NN	адрес

Таблица 2.6 – Таблица Order

поле	тип данных	ограничения	значение
id	uuid	PK	идентификатор
customer_login	text	FK	логин заказчика
employee_login	text	FK	логин сотрудника
date	timestamp	NN	дата
status	text	NN	статус
delivery_point_id	uuid	NN	идентификатор точки доставки
initial_cost	int	NN	начальная стоимость заказа
paid_by_bonuses	int	NN	оплачено бонусами

Таблица 2.7 – Таблица Order_Product

поле	тип данных	ограничения	значение
order_id	uuid	FK, NN	идентификатор заказа
product_id	uuid	FK, NN	идентификатор товара
price	int	NN	цена товара
cnt_products	int	NN	количество товара

Таблица 2.8 – Таблица Cart

поле	тип данных	ограничения	значение
product_id	uuid	FK, NN	идентификатор товара
login	text	FK, NN	логин заказчика
cnt_products	int	NN	количество товара

2.2 Ролевая модель

В данном разделе описаны права, которыми обладают пользователи на уровне базы данных.

Права неавторизованного пользователя:

— выборка из таблиц User, Product, Manufacturer, DeliveryPoint, Card;

— вставка в таблицы Card, User.

Права заказчика:

— выборка из таблиц User, Product, Manufacturer, DeliveryPoint, Card, Order_Product;

— вставка в таблицы Order, Order_Product, Cart.

— обновление таблиц Card, Product, Cart.

— удаление из таблицы Cart.

Права сотрудника:

— выборка из таблиц User, Product, Manufacturer, DeliveryPoint, Card, Order_Product;

— вставка в таблицы Order, Order_Product, Cart.

— обновление таблиц Card, Product, Cart, Order.

— удаление из таблицы Cart.

Администратор обладает доступом к выполнению любых операций с любыми таблицами.

2.3 Разработка триггера

При выполнении операции вставки записи в таблицу заказов, в системе автоматически уменьшается количество товара на складе. Для этого используется специальный триггер decrease_storage_cnt_trigger. Триггер decrease_storage_cnt_trigger относится к категории AFTER (после выполнения операции) и выполняет UPDATE запрос для изменения соответствующей записи в таблице Product.

На рисунке 2.1 представлена схема алгоритма работы триггера.

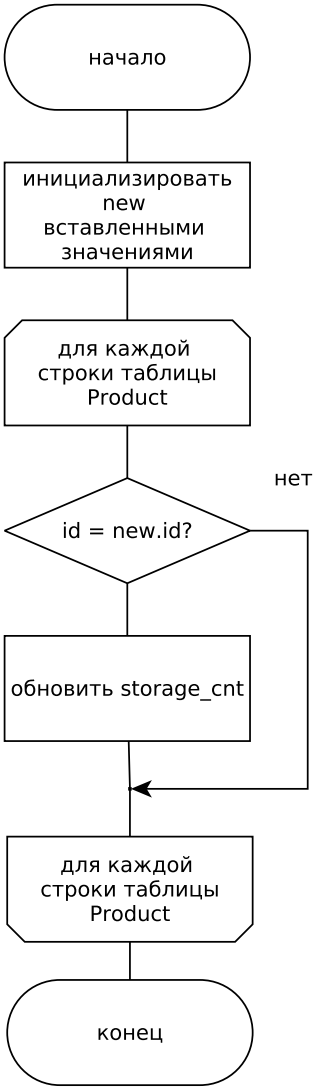

Рисунок 2.1 – Схема алгоритма работы триггера

3 Технологический раздел

3.1 Используемое программное обеспечение

В качестве СУБД была выбрана реляционная система управления базами данных PostgreSQL [4]. PostgreSQL является бесплатной и доступной в РФ, поддерживается большинством операционных систем, имеет открытый исходный код. Кроме того, данная СУБД позволяет обеспечить целостность данных с помощью различных ограничений.

Для реализации приложения был выбран платформонезависимымй язык программирования Java [5], система автоматической сборки Gradle [6] и набор программного обеспечения openjdk-20, содержащий библиотеку JDBC [7], позволяющую работать с реляционными базами данных.

3.2 Реализация сущностей системы

В листингах 3.1–3.3 приведены запросы для создания таблиц.

Листинг 3.1 – Запросы для создания таблиц (часть 1)

```
1  CREATE TABLE IF NOT EXISTS public.User (
2    login         text  PRIMARY KEY DEFAULT gen_random_uuid()
3    , password    bytea NOT NULL
4    , role_       text  NOT NULL
5    , first_name  text
6    , last_name   text
7    , birth_date  date
8    , email       text
9    , constraint check_role check (role_ in
       ('EMPLOYEE','CUSTOMER', 'ADMIN'))
10 );
11
12 CREATE TABLE IF NOT EXISTS public.Card (
13   user_login      text NOT NULL REFERENCES public.User(login) ON
       DELETE CASCADE
14   , bonuses       int  NOT NULL DEFAULT 0 CHECK (bonuses >= 0)
15 );
16 CREATE TABLE IF NOT EXISTS public.Manufacturer (
17   id              uuid PRIMARY KEY DEFAULT gen_random_uuid()
18   , name_         text UNIQUE NOT NULL
19 );
```

Листинг 3.2 – Запросы для создания таблиц (часть 2)

```
1  CREATE TABLE IF NOT EXISTS public.Product (
2    id                 uuid PRIMARY KEY DEFAULT gen_random_uuid()
3    , name_            text NOT NULL
4    , price            int  NOT NULL
5    , description      text
6    , color            text
7    , storage_cnt      int  NOT NULL DEFAULT 0 CHECK (storage_cnt
       >= 0)
8    , img_ref          text
9    , manufacturer_id  uuid NOT NULL REFERENCES
       public.Manufacturer(id) ON DELETE CASCADE
10   , characteristics json
11 );
12
13 CREATE TABLE IF NOT EXISTS public.DeliveryPoint (
14   id             uuid PRIMARY KEY DEFAULT gen_random_uuid()
15   , address      text UNIQUE NOT NULL
16 );
17
18 CREATE TABLE IF NOT EXISTS public.Order_(
19   id                   uuid        PRIMARY KEY DEFAULT
       gen_random_uuid()
20   , customer_login     text        REFERENCES public.User(login)
       ON DELETE CASCADE
21   , employee_login     text        REFERENCES public.User(login)
       ON DELETE CASCADE
22   , date_              timestamptz NOT NULL
23   , status             text        NOT NULL
24   , delivery_point_id  uuid        NOT NULL REFERENCES
       public.DeliveryPoint(id) ON DELETE CASCADE
25   , initial_cost       int         NOT NULL CHECK (initial_cost >
       0)
26   , paid_by_bonuses    int         NOT NULL CHECK
       (paid_by_bonuses <= initial_cost)
27   , constraint check_status check (status in ('formed','built',
       'delivered', 'received'))
28 );
```

Листинг 3.3 – Запросы для создания таблиц (часть 3)

```
1  CREATE TABLE IF NOT EXISTS public.Order_Product (
2    order_id            uuid NOT NULL REFERENCES public.Order_(id)
         ON DELETE CASCADE
3    , product_id        uuid NOT NULL REFERENCES
       public.Product(id) ON DELETE CASCADE
4    , price           int  NOT NULL CHECK (price > 0)
5    , cnt_products      int  NOT NULL CHECK (cnt_products > 0)
6  );
7
8  CREATE TABLE IF NOT EXISTS public.Cart (
9    login               text NOT NULL REFERENCES
       public.User(login) ON DELETE CASCADE
10   , product_id        uuid NOT NULL REFERENCES
       public.Product(id) ON DELETE CASCADE
11   , cnt_products      int  NOT NULL CHECK (cnt_products > 0)
12   , primary key(login, product_id)
13 );
```

3.3 Реализация триггера

В листинге 3.4 приведена реализация функции decrease_storage_cnt.

Листинг 3.4 – Реализация функции decrease_storage_cnt

```
1  CREATE OR REPLACE FUNCTION decrease_storage_cnt()
2    RETURNS TRIGGER AS
3  $$
4  BEGIN
5    update public.product
6    SET storage_cnt = storage_cnt - new.cnt_products
7    where id = new.product_id;
8    RETURN new;
9  END;
10 $$ language plpgsql;
```

В листинге 3.5 приведена реализация триггера decrease_storage_cnt_trigger.

Листинг 3.5 – Создание триггера decrease_storage_cnt_trigger

```
1 CREATE TRIGGER decrease_storage_cnt_trigger
2   AFTER INSERT
3   ON public.order_product
4   FOR EACH ROW
5 EXECUTE PROCEDURE decrease_storage_cnt();
```

3.4 Реализация ролевой модели

В листингах 3.6–3.9 приведена реализация ролевой модели.

Листинг 3.6 – Реализация роли заказчика

```
1 CREATE ROLE customer;
2 GRANT SELECT ON public.User TO customer;
3 GRANT SELECT ON public.Manufacturer TO customer;
4 GRANT SELECT, UPDATE ON public.Product TO customer;
5 GRANT SELECT ON public.DeliveryPoint TO customer;
6 GRANT SELECT, UPDATE ON public.Card TO customer;
7 GRANT SELECT, INSERT ON public.Order_ TO customer;
8 GRANT SELECT, INSERT ON public.Order_Product TO customer;
9 GRANT SELECT, INSERT, update, Delete ON public.Cart TO customer;
```

Листинг 3.7 – Реализация роли сотрудника

```
1 CREATE ROLE employee;
2 GRANT SELECT ON public.User TO employee;
3 GRANT SELECT ON public.Manufacturer TO employee;
4 GRANT SELECT, UPDATE ON public.Product TO employee;
5 GRANT SELECT ON public.DeliveryPoint TO employee;
6 GRANT SELECT, UPDATE ON public.Card TO employee;
7 GRANT SELECT, INSERT, UPDATE ON public.Order_ TO employee;
8 GRANT SELECT, INSERT ON public.Order_Product TO employee;
9 GRANT SELECT, INSERT, update, Delete ON public.Cart TO employee;
```

Листинг 3.8 – Реализация роли администратора

```
1 CREATE ROLE admin_;
2 grant all privileges ON all tables in schema public to admin_;
```

Листинг 3.9 – Реализация роли незарегистрированного пользователя

```
1 CREATE ROLE unregistered;
2 GRANT SELECT, INSERT ON public.User TO unregistered;
3 GRANT SELECT ON public.Manufacturer TO unregistered;
4 GRANT SELECT ON public.Product TO unregistered;
5 GRANT SELECT ON public.DeliveryPoint TO unregistered;
6 GRANT INSERT ON public.Card TO unregistered;
```

3.5 Демонстрация работы приложения

На рисунках 3.1–3.3 изображена главная страница, страница с детельной информацией о товаре, а также страница с информацией о пользователе. продемонстрирован интерфейс приложения.

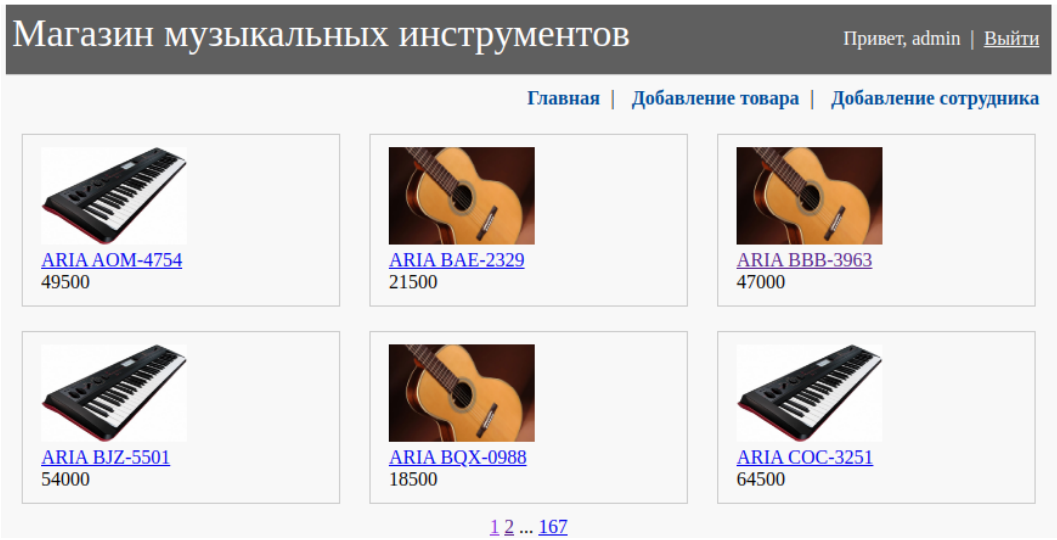

Рисунок 3.1 – Главная страница

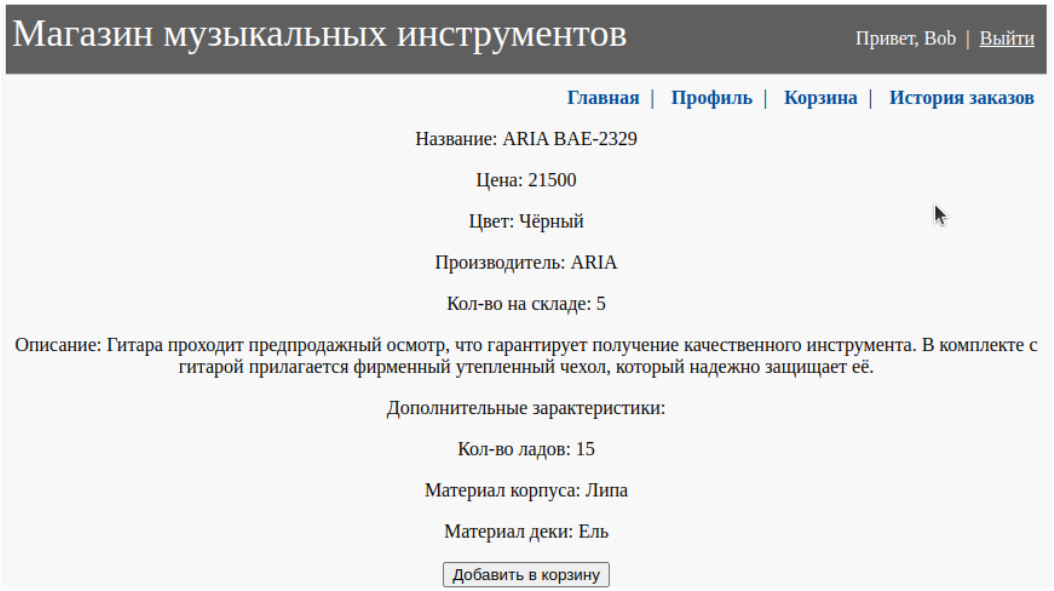

Рисунок 3.2 – Страница товара

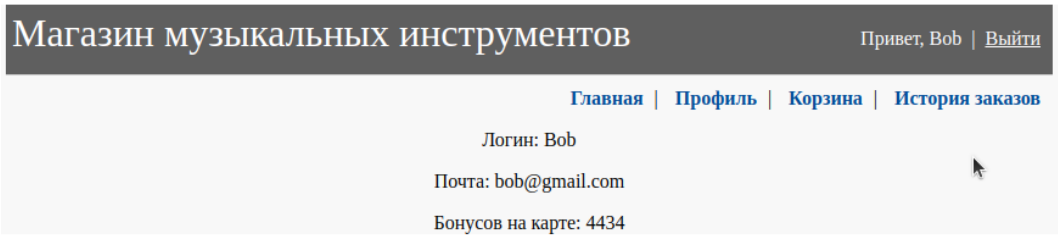

Рисунок 3.3 – Информация о пользователе

На рисунках 3.4–3.5 изображена корзина пользователя и его история заказов.

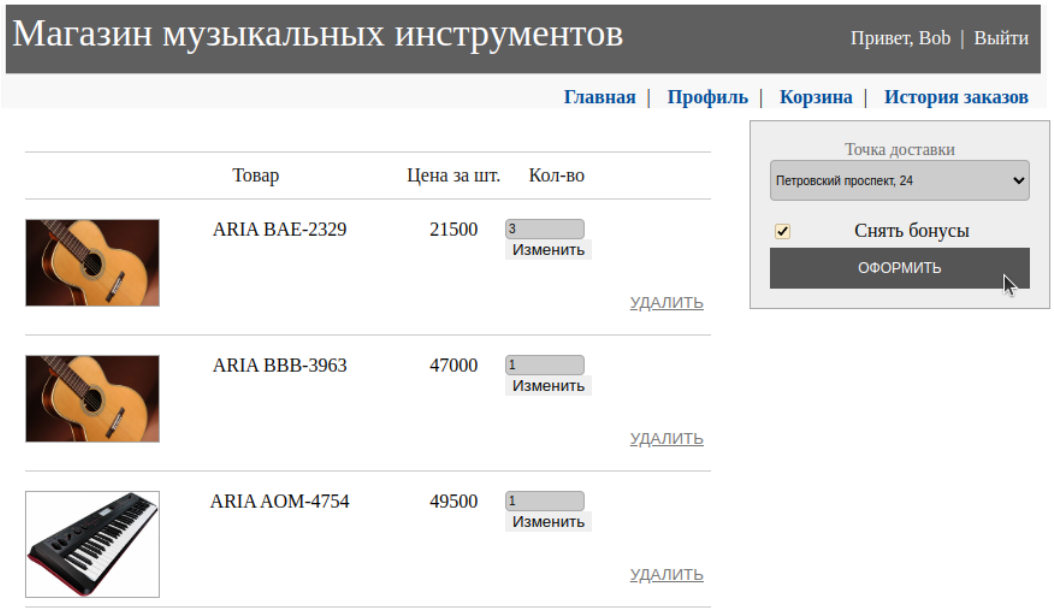

Рисунок 3.4 – Корзина пользователя

Рисунок 3.5 – История заказов

4 Исследовательская часть

4.1 Постановка исследования

Цель исследования — сравнение времени выполнения запросов на выборку при наличии индекса в базе данных и без него.

В листинге 4.1 приведен запрос на выборку из таблицы товаров по названию товара.

Листинг 4.1 – запрос на выборку из таблицы товаров

```
1 SELECT p.id
2 FROM public.product p
3 WHERE p.name_ = ?;
```

В листинге 4.2 приведен запрос для создания индекса на поле name_ таблицы товаров.

Листинг 4.2 – запрос для создания индекса

```
1 CREATE UNIQUE INDEX product_name_idx ON public.product (name_);
```

Технические характеристики устройства, на котором выполнялись замеры времени:

— операционная система — Ubuntu 22.04.1 Linux x86_64;

— оперативная память — 8 ГБ;

— процессор — AMD Ryzen 5 3550H [8].

Замеры проводились на ноутбуке, включенном в сеть электропитания. Во время замеров ноутбук не был нагружен сторонними приложениями.

4.2 Результаты исследования

На рисунке 4.1 представлен график, иллюстрирующий зависимость времени выполнения запроса на выборку из таблицы товаров от количества записей и от наличия индекса.

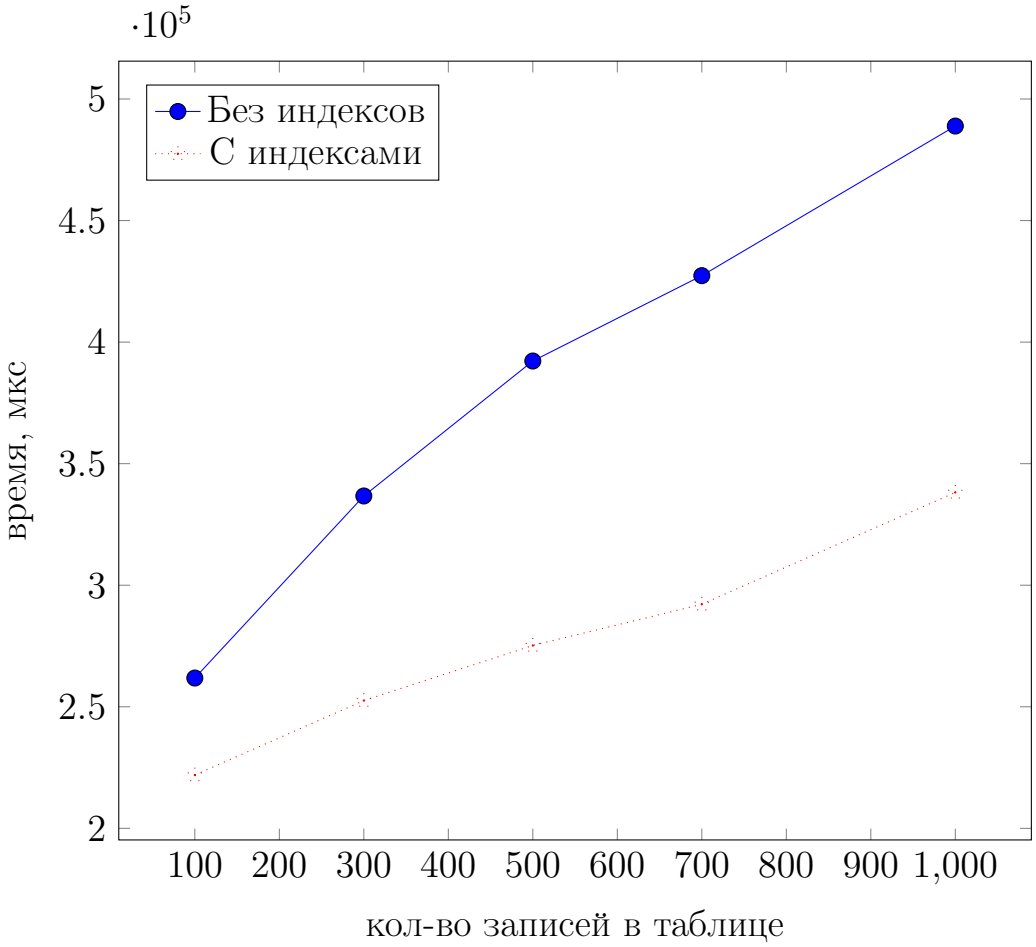

Рисунок 4.1 – Сравнение времени выполнения запросов на выборку при наличии индекса и без него

Вывод

При 100 записях в таблице время выполнения запроса при наличии индекса быстрее на 15 %, при 1000 записях — на 31 %. Таким образом, индексы значительно ускоряют операции выборки. Однако важно отметить, что индексы замедляют операции вставки, обновления и удаления данных, так как база данных должна поддерживать актуальность индекса.

ЗАКЛЮЧЕНИЕ

Цель достигнута — разработана база данных для хранения и обработки данных музыкального магазина.

Были выполнены все задачи:

— проведен анализ существующих решений;

— сформулировано описание пользователей приложения;

— формализованы данные;

— проведен анализ моделей данных;

— спроектирована архитектура базы данных и ограничения целостности;

— спроектирована ролевая модель на уровне базы данных;

— выбраны средства реализации;

— реализована спроектированная БД и необходимый интерфейс для взаимодействия с ней;

— проведено сравнение времени выполнения запросов на выборку при наличии индекса в базе данных и без него.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Исследование объективности связи ценообразующих факторов качества музыкальных инструментов с мнением их потенциальных покупателей / А. Е. Зверовщиков [и др.] // Известия высших учебных заведений. Поволжский регион. — 2018. № 1 (45). — С 190–206.
2. Основы проектирования баз данных: учебное пособие / Д.А. Попова-Коварцева, Е.В. Сопченко // Издательство Самарского университета, — 2019. — 112 с.
3. Основы технологий баз данных: учебное пособие / А. А. Агафонов, А. М. Белов // Издательство Самарского университета, — 2023. — 304 с.
4. PostgreSQL: The World's Most Advanced Open Source Relational Database [Электронный ресурс]. — Режим доступа: `https://www.postgresql.org/` — (Дата обращения: 2023-05-20).
5. Документация по Java [Электронный ресурс]. — Режим доступа: `https://www.java.com/ru/` — (Дата обращения: 2023-05-17).
6. Gradle Build Tool [Электронный ресурс]. — Режим доступа: `https://gradle.org/` — (Дата обращения: 2023-05-17).
7. JDBC Introduction [Электронный ресурс]. — Режим доступа: `https://docs.oracle.com/javase/tutorial/jdbc/overview/index.html` — (Дата обращения: 2023-05-18).
8. AMD Ryzen™ 5 3550H [Электронный ресурс]. — Режим доступа: `https://www.amd.com/en/products/apu/amd-ryzen-5-3550h` (дата обращения: 2023-05-17).

ПРИЛОЖЕНИЕ А

Презентация к курсовой работе

Презентация содержит 19 слайдов.