МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1–40 05 01 «Информационные системы и технологии»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОЙ РАБОТЕ НА ТЕМУ:**

«Реализация базы данных для онлайн магазина свечей с использованием технологии аналитики данных»

Выполнил студент Рудяк В.В.

(Ф.И.О.)

Руководитель проекта асс. Н.И. Уласевич

(учен. степень, звание, должность, Ф.И.О., подпись)

Консультант: асс. Н.И. Уласевич.

(учен. степень, звание, должность, Ф.И.О., подпись)

Нормоконтролер: асс. Н.И. Уласевич.

(учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

**Содержание**

[Введение 3](#_Toc167093848)

[1 Постановка задачи 4](#_Toc167093849)

[1.1 Обзор аналогичных решений 4](#_Toc167093850)

[1.2 Определение цели и задачи 7](#_Toc167093851)

[2 Проектирование базы данных 8](#_Toc167093852)

[2.1 Описание структуры базы данных 8](#_Toc167093853)

[2.2 Логическая схема базы данных 9](#_Toc167093854)

[3 Разработка объектов базы данных 11](#_Toc167093855)

[3.1 Разработка таблиц базы данных 11](#_Toc167093856)

[3.2 Процедуры 13](#_Toc167093857)

[3.3 Индексы 14](#_Toc167093858)

[3.4 Функции 15](#_Toc167093859)

[3.5 Триггеры 16](#_Toc167093860)

[4 Описание процедур импорта и экспорта 20](#_Toc167093861)

[5 Тестирование производительности 23](#_Toc167093862)

[6 Технология «Аналитика данных» 24](#_Toc167093863)

[Заключение 26](#_Toc167093864)

[Список используемых источников 27](#_Toc167093865)

[Приложение А Создание таблиц 28](#_Toc167093866)

[Приложение Б Создание процедур 30](#_Toc167093867)

[Приложение В Создание функций 35](#_Toc167093868)

[Приложение Г Создание триггеров 38](#_Toc167093869)

[Приложение Д Аналитика данных 42](#_Toc167093870)

[Приложение Ж Реализация ролей 46](#_Toc167093871)

# **Введение**

Темой проекта является создание интернет-магазина, который будет заказчику предоставлять возможность продавать товары, а пользователям — ознакомиться с ассортиментом товаров, найти нужные товары и оформить заказ через интернет.

Онлайн-магазины свечей сейчас пользуются высокой актуальностью, поскольку в наше время интерьерные свечи приобретают все большую популярность как элемент декора и подарков, а возможность выбрать и приобрести понравившийся товар онлайн значительно упрощает задачу и экономит время. Благодаря высокой актуальности данных товаров, сейчас это является так же довольно прибыльным бизнесом.

Основной целью проекта является создание интернет-магазина, который будет полностью соответствовать требованиям заказчика, и предоставлять пользователям возможность ознакомиться с ассортиментом магазина свечей, выбрать понравившиеся товары и оформить заказ через интернет.

Целевая аудитория: Заказчик, которому необходим интернет-магазин для продажи товаров. Любители свечей и декора, организаторы мероприятий (свадьбы, праздники, вечеринки), покупатели, ищущие подарки, владельцы интерьерных салонов и магазинов.

Для разработки проекта будет использована система управления реляционными базами данных MSSQL.

Целью данного проекта является разработка удобного функционала, а так же создание базы данных для онлайн-магазина свечей.

1. **Постановка задачи**
   1. **Обзор аналогичных решений**

Для успешной реализации проекта необходимо ознакомиться и проанализировать аналогичные проекты, чтобы подметить фрагменты, которые можно использовать в качестве примера, а так же для того, чтобы избежать возможные ошибки.

В качестве первого аналога рассмотрим интернет-магазин свечей «Сandlesbox», который представлен на рисунке 1.1.

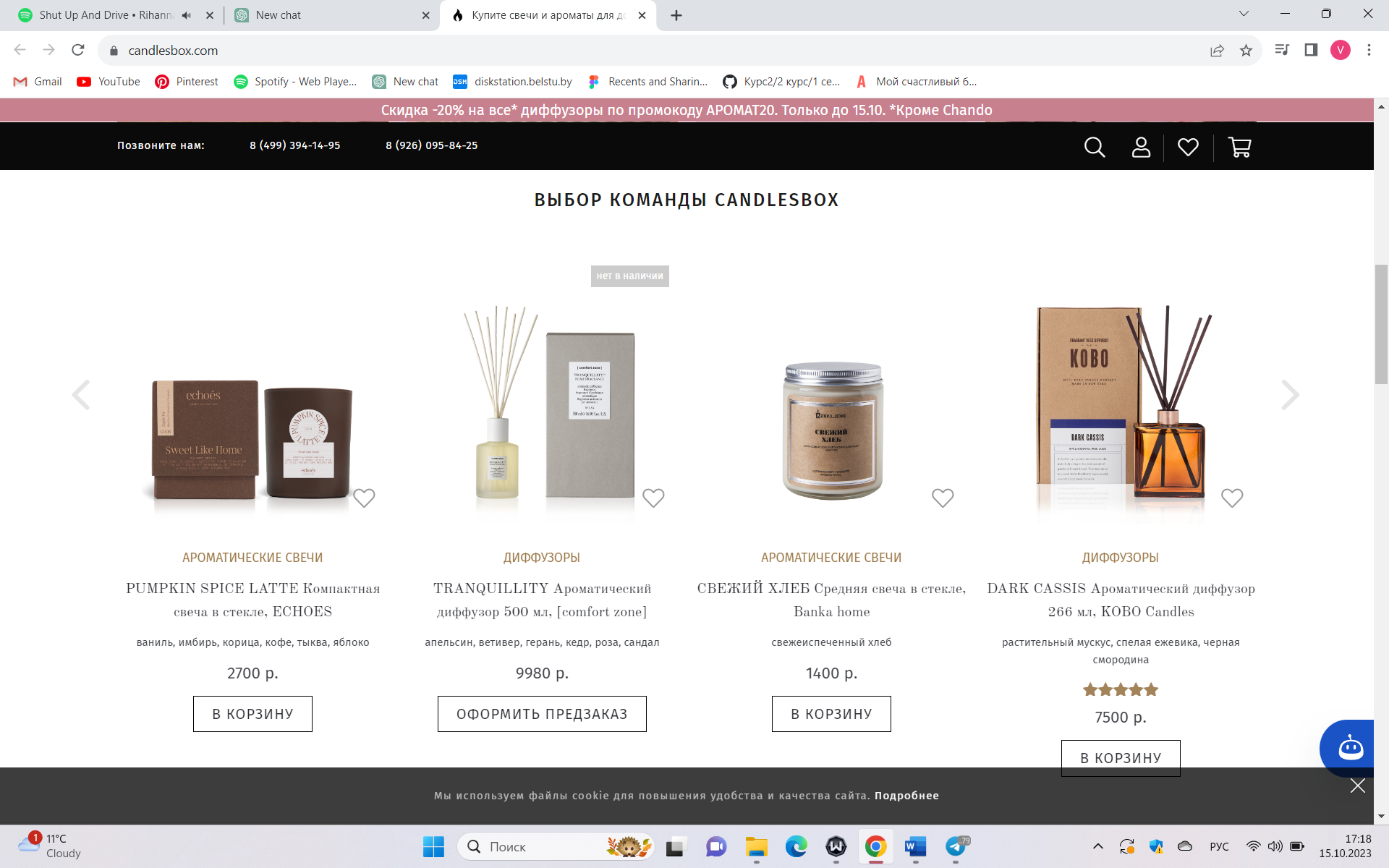


Рисунок 1.1 – Сайт интернет-магазина «Сandlesbox»

У данного интернет-магазина очень удобный и лаконичный интерфейс, интересный дизайн, а так же хорошо подобраны фотографии, которые отлично сочетаются со стилистикой веб-сайта. Из минусов можно выделить отсутствие разделов товаров и возможности сортировать товар по критериям.

Следующим аналогом является сайт интернет-магазина свечей «Свечной двор», который представлен на рисунке 1.2.

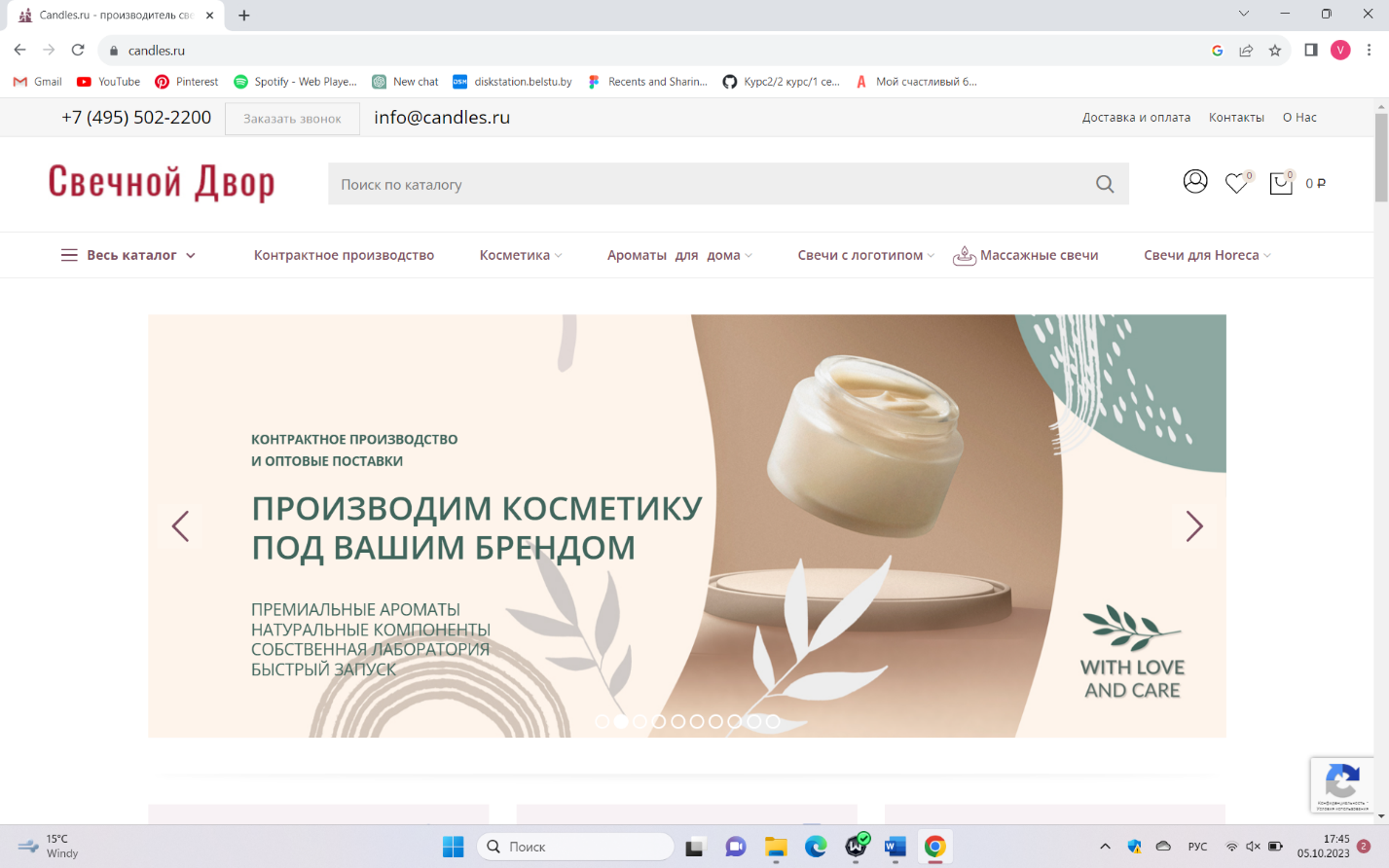


Рисунок 1.2 – Сайт интернет-магазина «Свечной двор»

Можно заметить, что у данного сайта есть разделы, а так же есть возможность отсортировать товары по критериям. Наличие разделов на сайте играет важную роль в организации информации и облегчении навигации для пользователей. Каждый раздел имеет своё предназначение и помогает удовлетворить потребности клиентов, будь то покупка, получение информации или связь с магазином. А возможность сортировки товаров упрощает поиск нужных товаров. На рисунке 1.3 изображена реализация сортировки товаров по критериям.

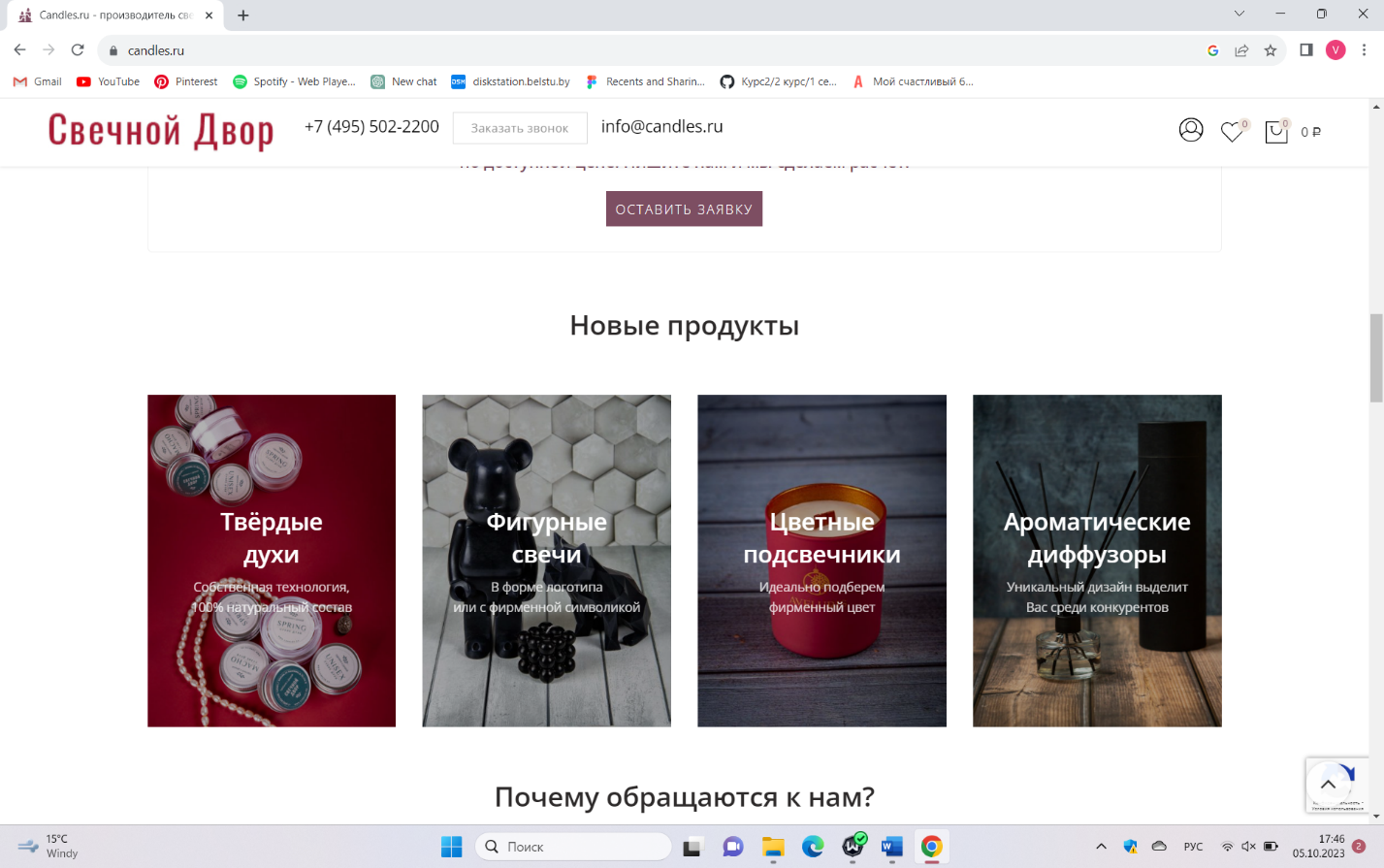


Рисунок 1.3 – Реализация сортировки товаров по критериям

У данного сайта из плюсов удобный интерфейс, где есть разделы, что можно позаимствовать. Однако можно выделить отсутствие уникального и запоминающегося логотипа и дизайна, что делает его мало отличимым на фоне других интернет-ресурсов.

В качестве третьего аналога рассмотрим сайт интернет-магазина свечей «Сandleworld». Сайт представлен на рисунке 1.4.

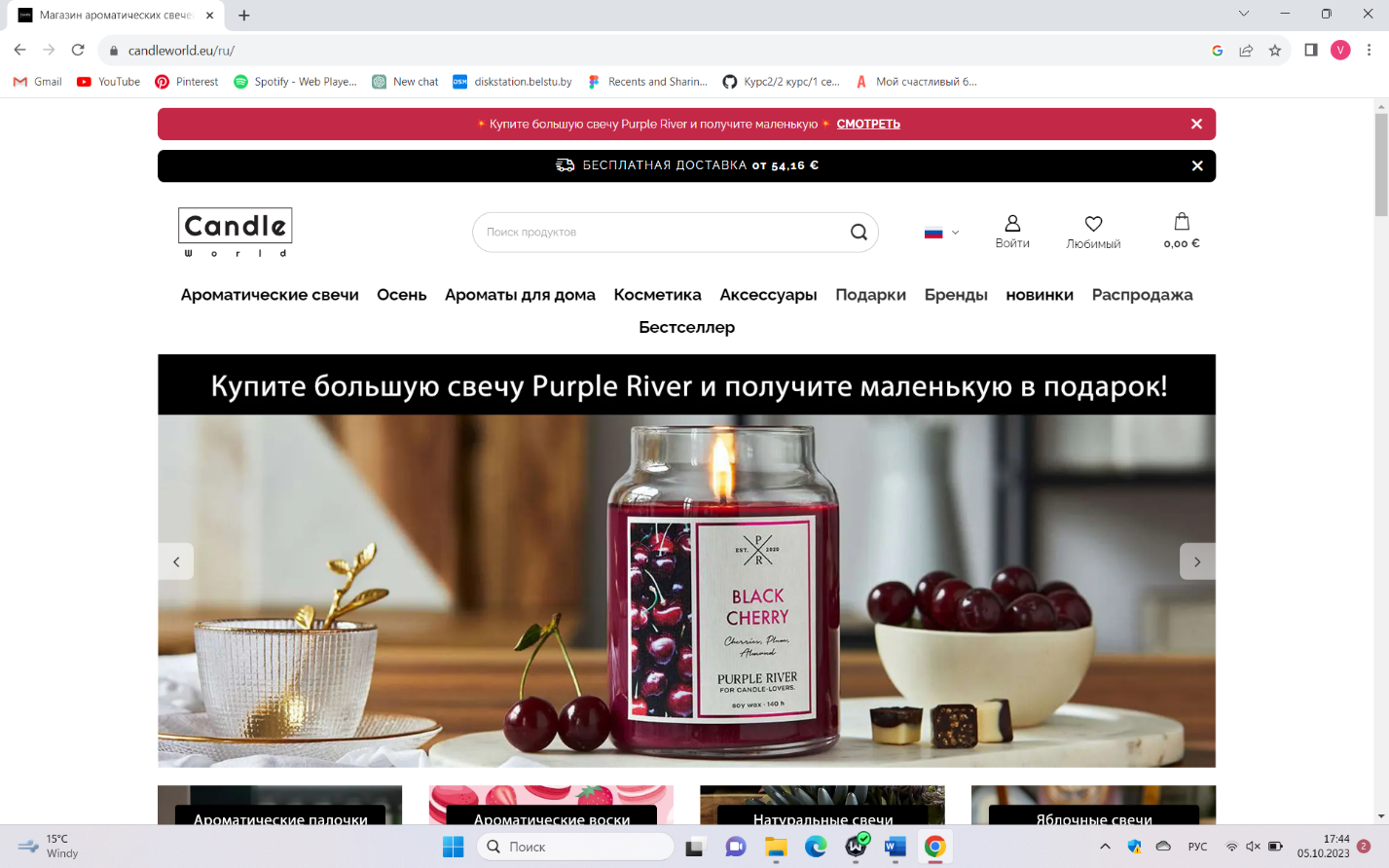
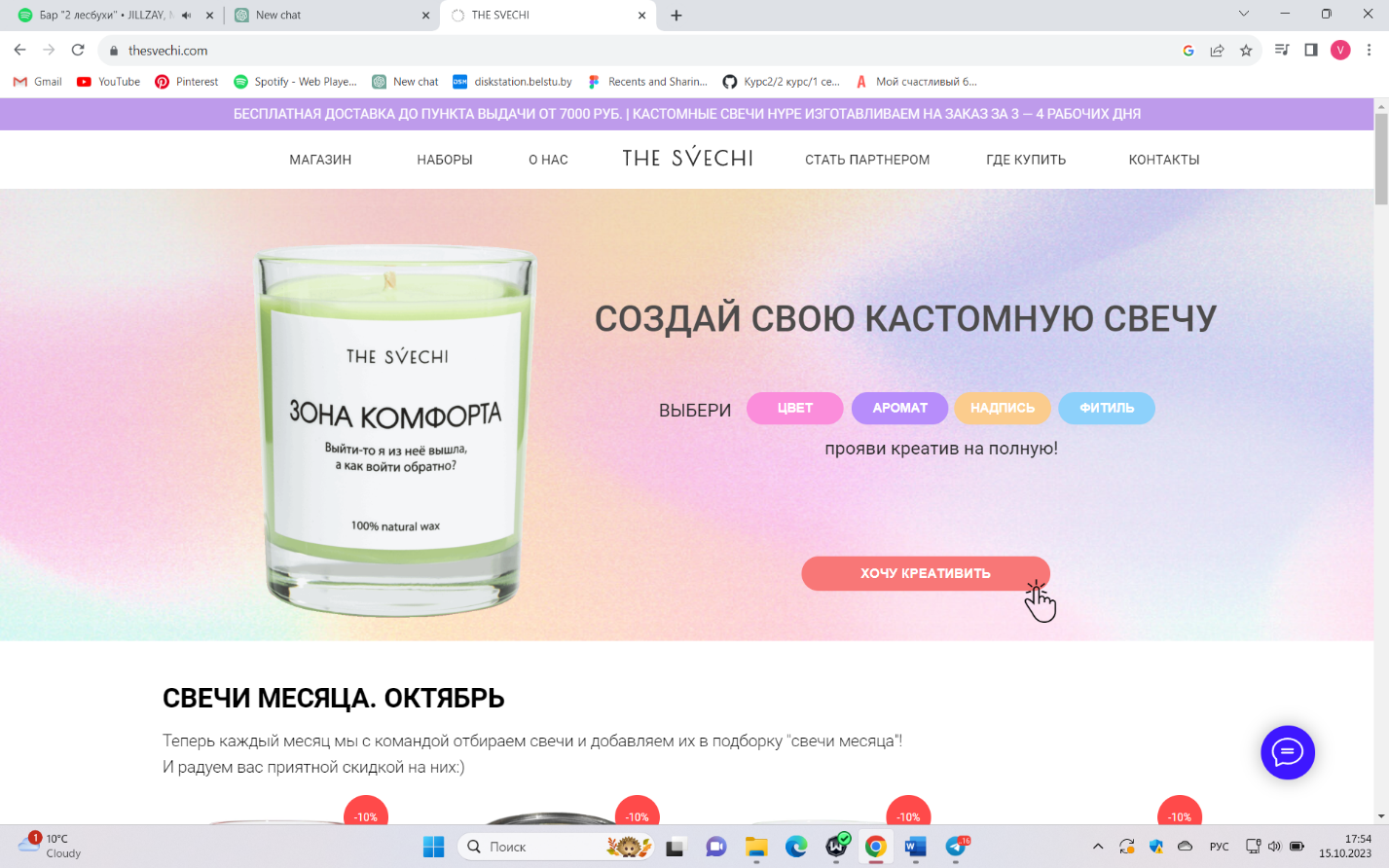


Рисунок 1.4 – Сайт интернет-магазина «Сandleworld»

Этот сайт предлагает удобную навигацию по разделам, что упрощает поиск нужных товаров. Интерфейс сайта интуитивно понятен и позволяет легко ориентироваться даже новым пользователям. Однако, несмотря на функциональность, сайту не хватает уникального и привлекательного дизайна, который мог бы выделить его среди конкурентов.

Последним аналогом рассмотрим интерне-магазин свечей «THESVECHI».

Рисунок 1.5 – Сайт интернет-магазина «THESVECHI»

Данный сайт однозначно можно считать примером во всём. У него хорошо продуманный удобный интерфейс и функционал, уникальный дизайн и логотип. Каталог товаров удобен в использовании, фотографии товаров подобраны качественные и в одном стиле, а так же присутствует отличительная черта у каждого товара в качестве интересной надписи. На рисунке 1.6 представлен каталог товаров магазина «THESVECHI»

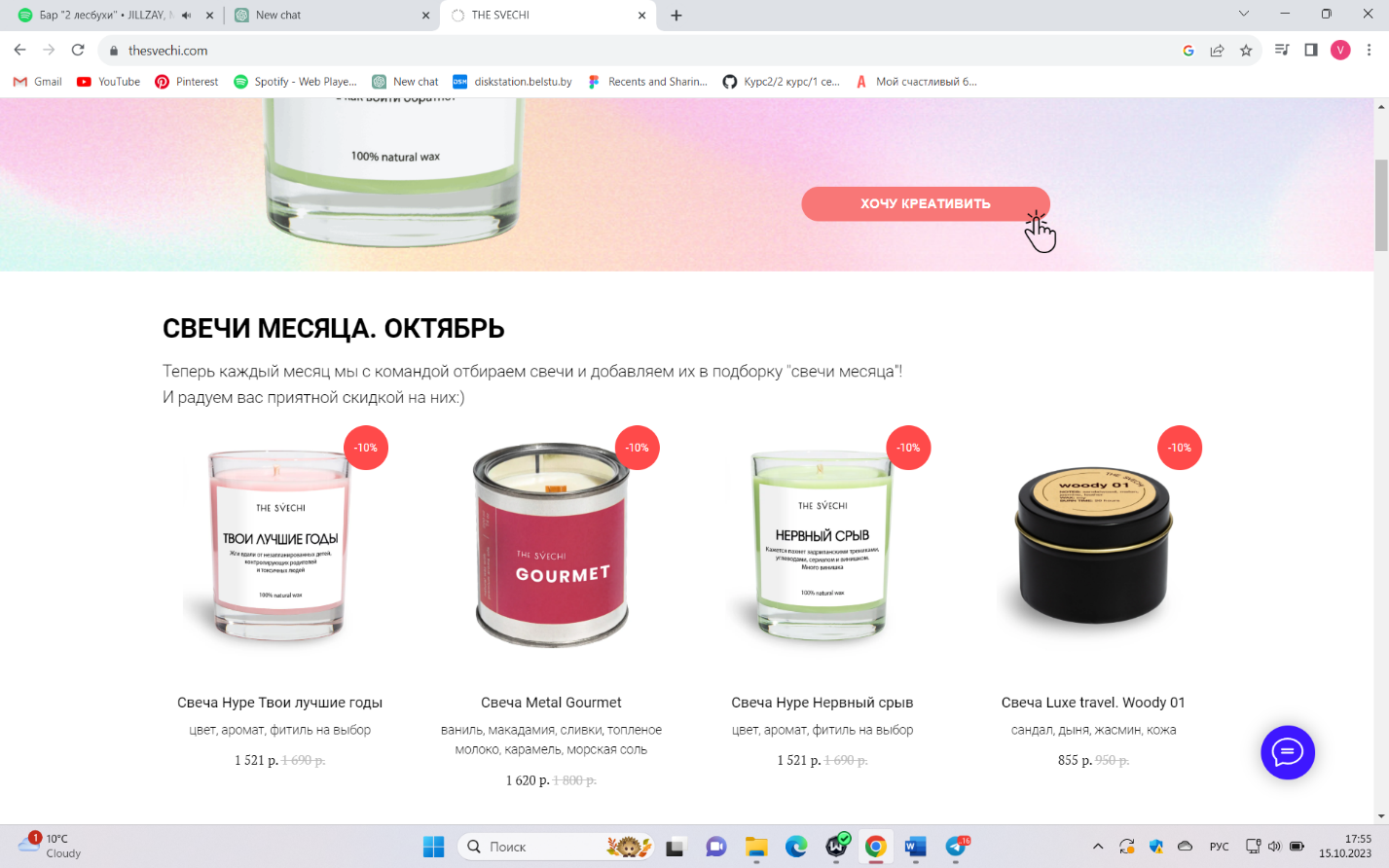


Рисунок 1.6 – Каталог товаров интернет-магазина «THESVECHI»

Благодаря всему вышеперечисленному, интернет-магазин «THESVECHI» явно выделяется среди других интернет-магазинов данной тематики. Из минусов можно выделить только отсутствие доставки в другие страны.

В результате анализа можно подвести следующий итог. Было выявлено, что дизайн сайта, логотип и качество фотографий играет важную роль для привлечения клиентов, поэтому нам тоже стоит уделить этому большое внимание.

## 1.2 Определение цели и задачи

На текущий момент рынок инструментов для управления магазина по продаже свечей представляет собой множество решений с различным уровнем функционала. Однако, многие из них не могут предложить полноценного комплекса инструментов, сочетающего в себе удобство и гибкость, что оставляет простор для новых разработок в этой сфере. Задача проекта: создать интерфейс, взаимодействие с которым будет понятно любому пользователю. Построить базу данных и выполнить тестирование готового продукта.

Должны быть выполнены следующие требования:

˗ база данных должна быть реализована в СУБД MSSQL.

˗ должен быть проведен импорт данных из JSON файлов, экспорт данных в формат JSON;

˗ необходимо протестировать производительность базы данных на таблице, содержащей не менее 100 000 строк, и внести изменения в структуру в случае необходимости. Необходимо проанализировать планы запросов к таблице;

˗ применить технологию аналитики в своей базе данных

# **2 Проектирование базы данных**

## 2.1 Описание структуры базы данных

Интернет-сервис "Любовь и свечи" предоставляет пользователям широкий набор возможностей, среди которых:

˗ Просмотр информации о товарах

˗ Добавление товаров в корзину

˗ Поиск товара по критериям и по названию

˗ Оформление заказов ­

Посетители сайта "Любовь и свечи" начинают с роли гостя, позволяющей ознакомиться с товарами. Сортировка по критериям упрощает поиск. Инструмент поиска обеспечивает быструю навигацию. Каждая свеча представлена качественным фото, информацией о материалах и ароматах для осознанного выбора.

После регистрации на сайте "Любовь и свечи" пользователь становится клиентом, получая дополнительные функции. В режиме клиента легко добавлять товары в корзину, что обеспечивает удобство при выборе. Оформление заказа включает согласие на обработку личных данных для оплаты, доставки и обратной связи. Магазин гарантирует безопасность и конфиденциальность данных в соответствии с современными стандартами.

Администратор магазина управляет ассортиментом, добавляя новые и удаляя неактуальные товары.

Функциональное наполнение пользовательских ролей представлено на UML-диаграмме на рисунке 2.1.



Рисунок 2.1 – Функциональное наполнение пользовательских ролей

Таким образом оценивая основные функции сервиса, можно заключить, что сайт обладает функционалом, который позволяет пользователям осуществлять не только поиск и просмотр информации, но и управлять своими заказами. Предоставление такой возможности укрепляет доверие клиентов к сервису, поскольку появляется ощущение контроля над ситуацией, которое сопровождается комфортом при пользовании сайтом. Реализация ролей представлена в приложении Ж.

## 2.2 Логическая схема базы данных

Схема базы данных представляет собой логическую конфигурацию либо целой реляционной базы данных, либо ее части. Схема может существовать как в виде наглядного представления базы данных, так и в виде набора формул (также именуемых «условиями целостности»), которые регулируют ее устройство. Эти формулы выражаются с помощью языка описания данных, например, SQL. Будучи частью словаря данных, схема показывает, как связаны между собой сущности, из которых состоит база данных (таблицы, представления, хранимые процедуры и так далее). Диаграмма базы данных представлена на рисунке 2.4.

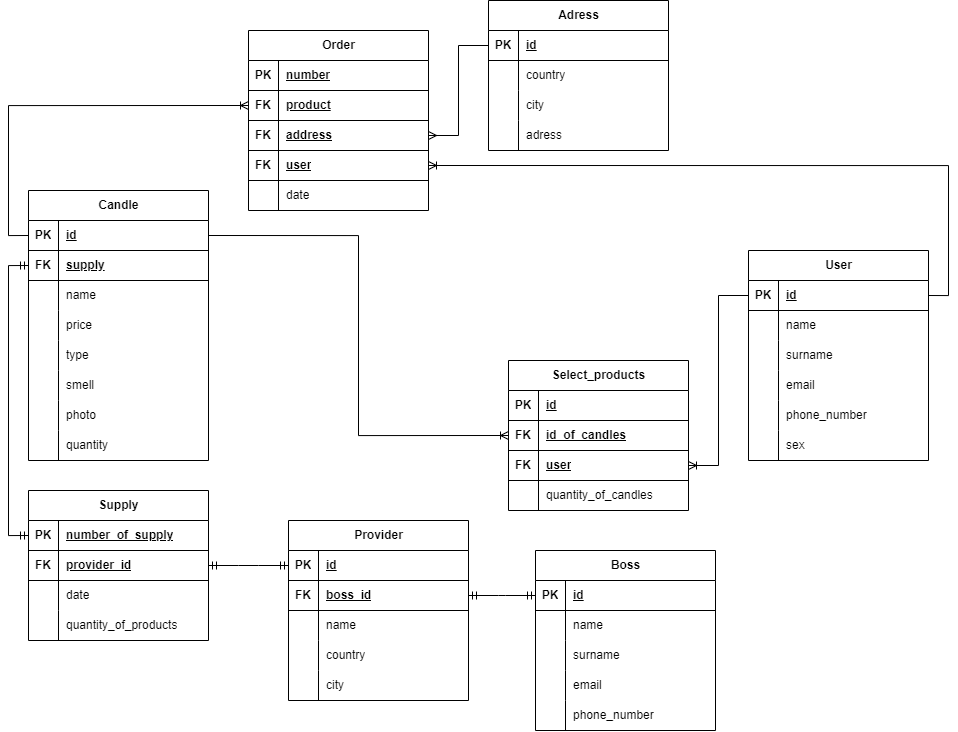


Рисунок 2.4 – Диаграмма базы данных

Для реализации базы данных было разработано 8 таблиц. Пояснение назначений каждой таблицы представлено в таблице 2.1.

Таблица 2.1 – Назначение таблиц в базе данных

|  |  |
| --- | --- |
| Имя таблицы | Назначение таблицы |
| Candle | Содержит информацию о каждом товаре |
| User | Содержит информацию о каждом пользователе |
| Order | Содержит информацию о заказах |
| Supply | Содержит информацию о поставках |
| Provider | Содержит информацию о компаниях-поставщиках |
| Boss | Содержит информацию о начальниках компаний-поставщиков |
| Select\_product | Содержит информацию о выбранных товарах для заказа |
| Adress | Содержит информацию о всех адресах пользователя |

Описанные таблицы представляют собой основные сущности, используемые в базе данных Интернет-сервиса "Любовь и свечи". Каждая таблица играет важную роль в функционировании сервиса, отражая различные аспекты его работы. Листинг SQL-кода для создания таблиц находится в [приложении А](#_Приложение_А).

**2.3 Вывод**

Таким образом, диаграмма показывает связи между таблицами и полями, а также отношения между ними, что значительно упрощает и улучшает разработку базы данных. Такая архитектура базы данных предоставляет гибкость в управлении данными и высокую производительность. Понимание структуры и функционала этих таблиц помогает обеспечить корректное хранение и обработку данных, а также оптимизировать процессы взаимодействия с пользователями и управления контентом сервиса.

# **3 Разработка объектов базы данных**

## 3.1 Разработка таблиц базы данных

При разработке таблиц в базе данных важно следовать принципам нормализации данных, чтобы избежать избыточности и обеспечить эффективное хранение информации. Это позволяет сократить объем хранимых данных и предотвратить аномалии при их обновлении. Для реализации базы данных было разработано 8 таблиц: Candle, User, Order, Supply, Provider, Boss, Select\_product, Adress.

Таблица Candle хранит информацию о товарах и состоит из столбцов, назначения которых представлены в таблице 3.1.

Таблица 3.1 – Столбцы таблицы Candle

|  |  |
| --- | --- |
| Столбцы таблицы | Назначение |
| Id | Хранит уникальный идентификатор товара |
| Supply | Хранит уникальный идентификатор поставщика |
| Name | Хранит название товара |
| Price | Хранит значение цены |
| Type\_smell | Хранит значение типа запаха |
| Smell | Хранит значение запаха |
| photo | Хранит фото |
| quantity | Хранит значение количества товаров |

Таблица Order хранит информацию о заказах и состоит из столбцов назначения которых представлены в таблице 3.2.

Таблица 3.2 – Столбцы таблицы Order

|  |  |
| --- | --- |
| Столбцы таблицы | Назначение |
| Number\_of\_order | Хранит номер заказа |
| Id\_user | Хранит уникальный идентификатор пользователя |
| Date | Хранит дату оформления заказа |
| Adress | Хранит адрес доставки |
| Products | Хранит информацию о выбранных товаарх |

Таблица User хранит информацию о клиентах и состоит из столбцов назначения которых представлены в таблице 3.3.

Таблица 3.3 – Столбцы таблицы User

|  |  |
| --- | --- |
| Столбцы таблицы | Назначение |
| Id | Хранит уникальный идентификатор пользователя |
| Name | Хранит имя пользователя |
| Surname | Хранит фамилию пользователя |
| Email | Хранит адрес электронной почты пользователя |
| Phone\_number | Хранит номер телефона пользователя |
| Sex | Хранит пол пользователя |

Таблица Select\_product хранит информацию о выбранных товарах и состоит из столбцов назначения которых представлены в таблице 3.4.

Таблица 3.4 – Столбцы таблицы Select\_product

|  |  |
| --- | --- |
| Столбцы таблицы | Назначение |
| Id | Хранит уникальный идентификатор выбранных товаров |
| Id\_of\_candles | Хранит уникальный идентификатор товара |
| Quantity\_if\_candles | Хранит количество товара |

Таблица Supply хранит информацию о поставках и состоит из столбцов назначения которых представлены в таблице 3.5.

Таблица 3.5 – Столбцы таблицы Supply

|  |  |
| --- | --- |
| Столбцы таблицы | Назначение |
| Number\_of\_supply | Хранит номер поставки |
| Provider\_id | Хранит уникальный идентификатор компании-поставщика |
| Date | Хранит дату поставки |
| Quantity\_of\_product | Хранит количество товара |

Таблица Provider хранит информацию о поставщиках и состоит из столбцов назначения которых представлены в таблице 3.6.

Таблица 3.6 – Столбцы таблицы Provider

|  |  |
| --- | --- |
| Столбцы таблицы | Назначение |
| Id | Хранит уникальный идентификатор компании-поставщика |
| Boss\_id | Хранит уникальный идентификатор начальника компании-поставщика |
| Name | Хранит название компании-поставщика |
| Country | Хранит страна компании-поставщика |
| city | Хранит город компании-поставщика |

Таблица Boss хранит информацию о начальниках компаний-поставщиков и состоит из столбцов назначения которых представлены в таблице 3.7.

Таблица 3.7 – Столбцы таблицы Boss

|  |  |
| --- | --- |
| Столбцы таблицы | Назначение |
| Id | Хранит уникальный идентификатор начальника компании-поставщика |
| Name | Хранит имя начальника компании-поставщика |
| Surname | Хранит фамилию начальника компании-поставщика |
| Email | Хранит адрес электронной почты начальника компании-поставщика |
| Phone\_number | Хранит номер телефона начальника компании-поставщика |

Связи в базе данных играют ключевую роль в связности между различными таблицами. Правильное проектирование связей между таблицами имеет важное значение для обеспечения целостности и эффективности работы базы данных. Это позволяет избежать дублирования данных, упрощает выполнение сложных запросов и обеспечивает согласованность информации. Связи между таблицами представлены в таблице 3.8.

Таблица 3.8 – Связи между таблицами

|  |  |  |
| --- | --- | --- |
| Таблица PK | Таблица FK | Описание связи |
| Select\_products | Order | Один к одному |
| Adress | Order | Один к многим |
| User | Order | Один к многим |
| Supply | Candle | Один к многим |
| Provider | Supply | Один к одному |
| Boss | Provider | Один к одному |

## 3.2 Процедуры

Процедура – это набор инструкций SQL, которые выполняют определенную операцию или набор операций. Она хранится в базе данных и может быть вызвана для выполнения этих инструкций. В процедуре можно использовать переменные, условные операторы, циклы, операторы выбора и другие SQL-конструкции для выполнения различных задач, таких как добавление данных, обновление таблицы, выборка данных и многое другое.

Процедуры, разработанные в рамках курсового проекта:

˗ AddToCart – добавление товара в корзину;

˗ ClearCart – очистка корзины;

˗ SearchCandlesByName – поиск товаров по названию;

˗ AddProduct – добавление товара в каталог;

˗ DeleteProduct – удаление товара из каталога;

˗ PlaceOrder – оформление заказа;

˗ AddSupply – управление поставками;

˗ ManageProvider – управление поставщиками;

˗ UpdateCandle– изменение товаров;

˗ DeleteUserByLogin– удаление пользователя по логину;

В качестве примера в листинге 3.1 представлена разработка процедуры, которая выполняет функцию оформления заказа, а разработка остальных процедур представлена в приложении Б.

|  |
| --- |
| CREATE PROCEDURE ADD\_ORDER  @login\_user VARCHAR(50),  @address\_id INT,  @order\_date DATE  AS  BEGIN  DECLARE @id\_of\_candles INT;  DECLARE @quantity\_of\_candles INT;  DECLARE cart\_cursor CURSOR FOR  SELECT id\_of\_candles, quantity\_of\_candles  FROM Select\_products  WHERE login\_user = @login\_user;  OPEN cart\_cursor;  FETCH NEXT FROM cart\_cursor INTO @id\_of\_candles, @quantity\_of\_candles;  WHILE @@FETCH\_STATUS = 0  BEGIN  -- Вставка записи в таблицу Orders  INSERT INTO Orders (adress, products, login\_user, date)  VALUES (@address\_id, @id\_of\_candles, @login\_user, @order\_date);  FETCH NEXT FROM cart\_cursor INTO @id\_of\_candles, @quantity\_of\_candles;  END;  CLOSE cart\_cursor;  DEALLOCATE cart\_cursor;  -- Очистка корзины после оформления заказа  DELETE FROM Select\_products WHERE login\_user = @login\_user;  END;  GO |

Листинг 3.1 – Разработка процедуры оформления заказа

Хранимые процедуры имеют много преимуществ. Он уменьшают сетевой трафик, поскольку только имя передается по сети от приложения на сервер. Правда, мы можем передавать еще несколько параметров, но передача имени хранимой процедуры и нескольких параметров не так сильно загружает сеть, как передача всего кода всякий раз, когда нужно этот код выполнить.

Хранимые процедуры усиливают безопасность. Код в хранимой процедуре определяет, какие действия производятся над объектами базы данных. Это означает, что при всяком вызове хранимой процедуры один и тот же блок кода выполняется одним и тем же способом. То есть мы не должны беспокоиться об ошибках в коде, повторяя набор одного и того же кода всякий раз, когда вам нужно выполнить его.

## 3.3 Индексы

Индексы в контексте баз данных – это структуры данных, создаваемые для оптимизации производительности запросов. Они позволяют быстро находить и получать доступ к данным в таблицах по определенным столбцам или комбинациям столбцов.

Индексы функционируют как предметные указатели в книге, позволяя быстро находить информацию без прочтения всего текста. Они представляют собой специальные таблицы, строки которых, в отличие от обычных таблиц данных, расположены в строго определённом порядке. Но вместо того, чтобы содержать все данные о некоторой записи, индекс содержит только столбец (или столбцы), используемый, чтобы найти строки в таблице данных, вместе с информацией, описывающей, где физически расположена эта строка. Таким образом, роль индексов состоит в том, чтобы облегчить поиск подмножества строк и столбцов таблицы без необходимости сканировать каждую строку в таблице

Индексаторы, разработанные в рамках курсового проекта, представлены в листинге 3.2.

|  |
| --- |
| CRCREATE INDEX idx\_CandleName ON Candle (name);  CREATE INDEX idx\_Candlequantity ON Candle(quantity);  CREATE INDEX idx\_Candletype ON Candle(type);  CREATE INDEX idx\_Candlesmell ON Candle(smell);  CREATE INDEX idx\_Candleprice ON Candle(price);  CREATE INDEX idx\_Address ON Address(adress);  CREATE INDEX idx\_NameUser ON Users(name);  CREATE INDEX idx\_loginUser ON Users(login);  CREATE INDEX idx\_dateOrder ON Orders(date); |

Листинг 3.2 – Разработка индексаторов

Важно помнить, что применять индексы нужно избирательно и только для тех полей, по которым действительно выполняются частые и важные запросы. Необходимо найти баланс между ускорением запросов и издержками на поддержание индексов.

## 3.4 Функции

Функции в контексте баз данных – это программные конструкции, которые принимают входные параметры, выполняют определенные вычисления или операции над данными, и возвращают результат. Они облегчают повторное использование кода, улучшают читаемость запросов и позволяют создавать более сложные и гибкие запросы.

Функции, разработанные в рамках курсового проекта:

˗ AddToCart – поиск по типу товара;

˗ ClearCart – поиск по названию товара;

˗ SearchCandlesByName – поиск товара по запаху;

˗ AddProduct – поиск заказов, оформленных одним пользователем;

˗ DeleteProduct – вычисление общей стоимости товаров в корзине;

˗ GetProductsDeliveredOnDate – поиск товаров по дате поставки;

˗ GetProductsByProvider– поиск товаров по поставщику;

В листинге 3.3 для примера представлена реализация функции, которая выполняет поиск товаров, заказанных конкретным пользователем. Остальные функции, разработанные в рамках курсового проекта, представлены в приложении В.

|  |
| --- |
| CREATE FUNCTION GetOrdersByUser  (  @login\_user VARCHAR(50)  )  RETURNS TABLE  AS  RETURN  (  SELECT  o.number,  o.adress,  a.country,  a.city,  a.adress AS address\_details,  o.products,  c.name AS product\_name,  o.login\_user,  o.date  FROM  Orders o  JOIN  Address a ON o.adress = a.ID  JOIN  Candle c ON o.products = c.ID  WHERE  o.login\_user = @login\_user  );  GO |

Листинг 3.3 – Разработка функции, выполняющей поиск товаров, заказанных конкретным пользователем

Функция принимает один параметр @login\_user, который представляет собой имя пользователя и имеет тип VARCHAR с максимальной длиной 50 символов. Функция возвращает таблицу и состоит из запроса, который выбирает данные из таблицы Orders, объединяет ее с таблицей Address по столбцу adress, а затем объединяет результат с таблицей Candle по столбцу products. Результаты фильтруются по значению login\_user, которое передается в функцию через параметр @login\_user. Таким образом, функция возвращает данные о заказах указанного пользователя, включая номер заказа, адрес доставки, страну и город доставки, детали адреса, продукты в заказе, название продукта и дату заказа.

## 3.5 Триггеры

Триггер SQL Server – это часть процедурного кода, как и хранимая процедура, но который выполняется только при наступлении определенного события. Имеются различные типы событий, которые могут вызвать срабатывание триггера. Например, вставка строки в таблицу, изменение структуры таблицы или авторизация пользователя в экземпляре SQL Server.

Функции, разработанные в рамках курсового проекта:

˗ OrderSuccessTrigger – триггер на оформление заказа;

˗ CheckProductQuantity– триггер на количество товаров в корзине;

˗ Trigger\_ClearCart\_EmptyTable– триггер на очистку корзины;

˗ PreventDuplicateCandle – триггер на одинаковые названия товаров при добавлении товаров в каталог;

˗ DeleteCandleTrigger – триггер на удаление товара из каталога;

˗ trg\_CheckDuplicateLogin – триггер на логин пользователя;

˗ UserDeletedTrigger– триггер на удаление пользователя;

В листинге 3.4 представлен триггер, который проверяет количество товаров перед добавлением товара в корзину, остальные представлены в приложении Г.

|  |
| --- |
| CREATE TRIGGER CheckProductQuantity  ON Select\_products  INSTEAD OF INSERT  AS  BEGIN  DECLARE @id\_of\_candles INT;  DECLARE @quantity\_of\_candles INT;  DECLARE @available\_quantity INT;  SELECT @id\_of\_candles = inserted.id\_of\_candles,  @quantity\_of\_candles = inserted.quantity\_of\_candles  FROM inserted;  SELECT @available\_quantity = quantity  FROM Candle  WHERE ID = @id\_of\_candles;  IF @quantity\_of\_candles > @available\_quantity  BEGIN  PRINT 'Ошибка: Недостаточное количество товаров на складе.';  END  ELSE  BEGIN  INSERT INTO Select\_products (id\_of\_candles, quantity\_of\_candles, login\_user)  SELECT id\_of\_candles, quantity\_of\_candles, login\_user  FROM inserted;  PRINT 'Товар успешно добавлен в корзину.';  END  END;  GO |

Листинг 3.4 – Создание триггера CheckProductQuantity

Этот триггер контролирует количество товаров при добавлении записей в таблицу Select\_products, предотвращая добавление товаров, если их количество превышает доступное на складе. Триггер активируется при операции вставки новых записей в таблицу Select\_products. Если количество вставляемых товаров превышает доступное количество на складе, выводится сообщение об ошибке. В противном случае, товары успешно добавляются в корзину. Таким образом, этот триггер предотвращает возможность добавления товаров в количестве, превышающем доступное количество на момент операции.

**3.6 Представления**

В SQL представления (views) содержат строки и столбцы, аналогичные таблицам, однако без фактических данных. Представление можно рассматривать как виртуальную таблицу, созданную из одной или нескольких таблиц, чтобы упростить работу с данными.

Представления создаются в случаях, когда необходимо дать пользователю возможность просматривать не всю таблицу, а только ее часть, а также для объединения двух или более таблиц.

Представления, разработанные в рамках курсового проекта:

˗ CandleCatalog– просмотр каталога;

˗ AllCandles– просмотр всей информации о товарах;

˗ UserOrders– просмотр заказов пользователей;

˗ SupplyDetails– просмотр поставок товаров;

˗ ProviderAndBossInfo– информация о поставщиках и их начальниках;

В листинге 3.5 представлены все представления.

|  |
| --- |
| --просмотр каталога  CREATE VIEW CandleCatalog AS  SELECT name, price, smell, photo  FROM Candle;  --просмотр информации о товарах  CREATE VIEW AllCandles AS  SELECT  c.ID,  c.name,  c.price,  c.type,  c.smell,  c.photo,  c.quantity,  s.date AS supply\_date,  p.name AS provider\_name  FROM  Candle c  INNER JOIN Supply s ON c.supply = s.number\_of\_supply  INNER JOIN Provider p ON s.provider\_id = p.ID;  --просмотр заказов пользователей  CREATE VIEW UserOrders AS  SELECT  o.number AS order\_number,  u.login AS user\_login,  u.name AS user\_name,  u.surname AS user\_surname,  a.country AS delivery\_country,  a.city AS delivery\_city,  a.adress AS delivery\_address,  o.date AS order\_date  FROM  Orders o  INNER JOIN Users u ON o.login\_user = u.login  INNER JOIN Address a ON o.adress = a.ID;  --просмотр поставок товаров  CREATE VIEW SupplyDetails AS  SELECT  s.number\_of\_supply AS supply\_id,  s.date AS supply\_date,  s.quantity\_of\_products AS total\_products,  p.name AS provider\_name,  b.name AS boss\_name,  b.surname AS boss\_surname,  b.email AS boss\_email,  b.phone\_number AS boss\_phone  FROM  Supply s  INNER JOIN Provider p ON s.provider\_id = p.ID  INNER JOIN Boss b ON p.boss\_id = b.ID;  --информация о поставщиках и их начальниках  CREATE VIEW ProviderAndBossInfo  AS  SELECT  p.ID AS ProviderID,  p.name AS ProviderName,  p.country AS ProviderCountry,  p.city AS ProviderCity,  b.name AS BossName,  b.surname AS BossSurname,  b.email AS BossEmail,  b.phone\_number AS BossPhoneNumber  FROM  Provider p  INNER JOIN  Boss b ON p.boss\_id = b.ID |

Листинг 3.5 – Представления

Используя эти представления, можно получить нужные данные при работе с базой данных.

# **4 Описание процедур импорта и экспорта**

Для таблицы Candle в базе данных доступна процедура экспорта данных в формате JSON. Это может пригодиться, если необходимо переместить данные на другой сервер или создать резервную копию.

Для реализации экспорта данных в JSON, была разработана функция, результатом которой стало создание JSON файла, с записанными строками из определенной таблицы в JSON формат. Код представлен в листинге 4.1.

|  |
| --- |
| CREATE OR ALTER PROCEDURE ExportToJson  @path NVARCHAR(500)  AS  BEGIN  DECLARE @output NVARCHAR(MAX);  DECLARE @cmd NVARCHAR(4000);  DECLARE @psCmd NVARCHAR(4000);  SELECT @output = (  SELECT  ID,  name,  price,  [type],  smell,  photo,  quantity,  supply  FROM  Candle  FOR JSON AUTO, ROOT('Candle')  );  SET @output = REPLACE(@output, '"', '\"');  SET @psCmd = 'powershell.exe -Command "$jsonContent = ''' + @output + '''; $filePath = ''' + @path + '''; $jsonContent | Out-File -FilePath $filePath -Encoding default"';  IF LEN(@psCmd) > 2000  BEGIN  RAISERROR('Команда слишком длинная', 16, 1);  RETURN;  END  EXEC xp\_cmdshell @psCmd;  END;  EXEC ExportToJson @path = 'C:\Users\user\Desktop\course3\file.json'; |

Листинг 4.1 – Процедура, выполняющая экспорт данных в json

А для реализации импорта была реализована процедура, представленная в листинге 4.2.

|  |
| --- |
| CREATE OR ALTER PROCEDURE ImportGenreDataFromJson  AS  BEGIN  CREATE TABLE #TempGenre (  id INT IDENTITY(1,1) PRIMARY KEY,  genre VARCHAR(255) NOT NULL UNIQUE  );  DECLARE @jsonInput NVARCHAR(MAX);  SELECT @jsonInput = BulkColumn  FROM OPENROWSET (BULK 'C:\localhost\export and import\genre.json', SINGLE\_CLOB) AS j;  INSERT INTO #TempGenre (genre)  SELECT genre  FROM OPENJSON(@jsonInput, '$.Genre')  WITH (  genre VARCHAR(255) '$.genre'  );  MERGE genre AS target  USING #TempGenre AS source  ON target.id = source.id  WHEN MATCHED THEN  UPDATE SET  target.genre = source.genre  WHEN NOT MATCHED BY TARGET THEN  INSERT (genre)  VALUES (source.genre);  DROP TABLE #TempGenre;  END;  GO  --импорт  CREATE OR ALTER PROCEDURE ImportFromJson  AS  BEGIN  CREATE TABLE #TempCandle (  ID INT IDENTITY(1,1) PRIMARY KEY,  name NVARCHAR(255) NOT NULL,  price INT,  [type] NVARCHAR(255),  smell NVARCHAR(255),  photo NVARCHAR(255),  quantity INT,  supply INT  );  DECLARE @Input NVARCHAR(MAX);  SELECT @Input = BulkColumn  FROM OPENROWSET (BULK 'C:\Users\user\Desktop\course3\file.json', SINGLE\_CLOB) AS j;  INSERT INTO #TempCandle (name, price, [type], smell, photo, quantity, supply)  SELECT  name,  price,  [type],  smell,  photo,  quantity,  supply  FROM OPENJSON(@Input, '$.Candle')  WITH (  name NVARCHAR(255) '$.name',  price INT '$.price',  [type] NVARCHAR(255) '$.type',  smell NVARCHAR(255) '$.smell',  photo NVARCHAR(255) '$.photo',  quantity INT '$.quantity',  supply INT '$.supply'  );  MERGE Candle AS target  USING #TempCandle AS source  ON target.ID = source.ID  WHEN MATCHED THEN  UPDATE SET  target.name = source.name,  target.price = source.price,  target.[type] = source.[type],  target.smell = source.smell,  target.photo = source.photo,  target.quantity = source.quantity,  target.supply = source.supply  WHEN NOT MATCHED BY TARGET THEN  INSERT (name, price, [type], smell, photo, quantity, supply)  VALUES (source.name, source.price, source.[type], source.smell, source.photo, source.quantity, source.supply);  DROP TABLE #TempCandle;  END; |

Листинг 4.1 – Процедура, выполняющая импорт данных из json

Таким образом, пользователи базы данных могут легко экспортировать в формате JSON, что делает управление базой данных более удобным и эффективным.

# **5 Тестирование производительности**

Тестирование базы данных – важный шаг, который разработчики должны предпринять, чтобы обеспечить оптимизацию производительности. Чем качественнее тестирование, тем лучше в итоге должен выйти конечный продукт. Выявление проблем с производительностью до того, как они повлияют на конечного пользователя, имеет жизненно важное значение для успеха любого приложения в разработке.

Для проверки производительности базы данных необходимо заполнить ее большим количеством различных данных. Для данной задачи была разработана процедура, которая за одно выполнение позволяет вставить в таблицу 100 000 строк. Код данной процедуры представлен в листинге 5.1.

|  |
| --- |
| CREATE OR ALTER PROCEDURE PopulateAddress  AS  BEGIN  DECLARE @i INT = 1  WHILE @i <= 100000  BEGIN  INSERT INTO Address (country, city, adress)  VALUES ('Country ' + CAST(@i AS NVARCHAR(50)), 'City ' + CAST(@i AS NVARCHAR(50)), 'Address ' + CAST(@i AS NVARCHAR(50)))  SET @i = @i + 1  END  END |

Листинг 5.1 – Процедура тестирования

План выполнения с представлен на рисунке 5.1.

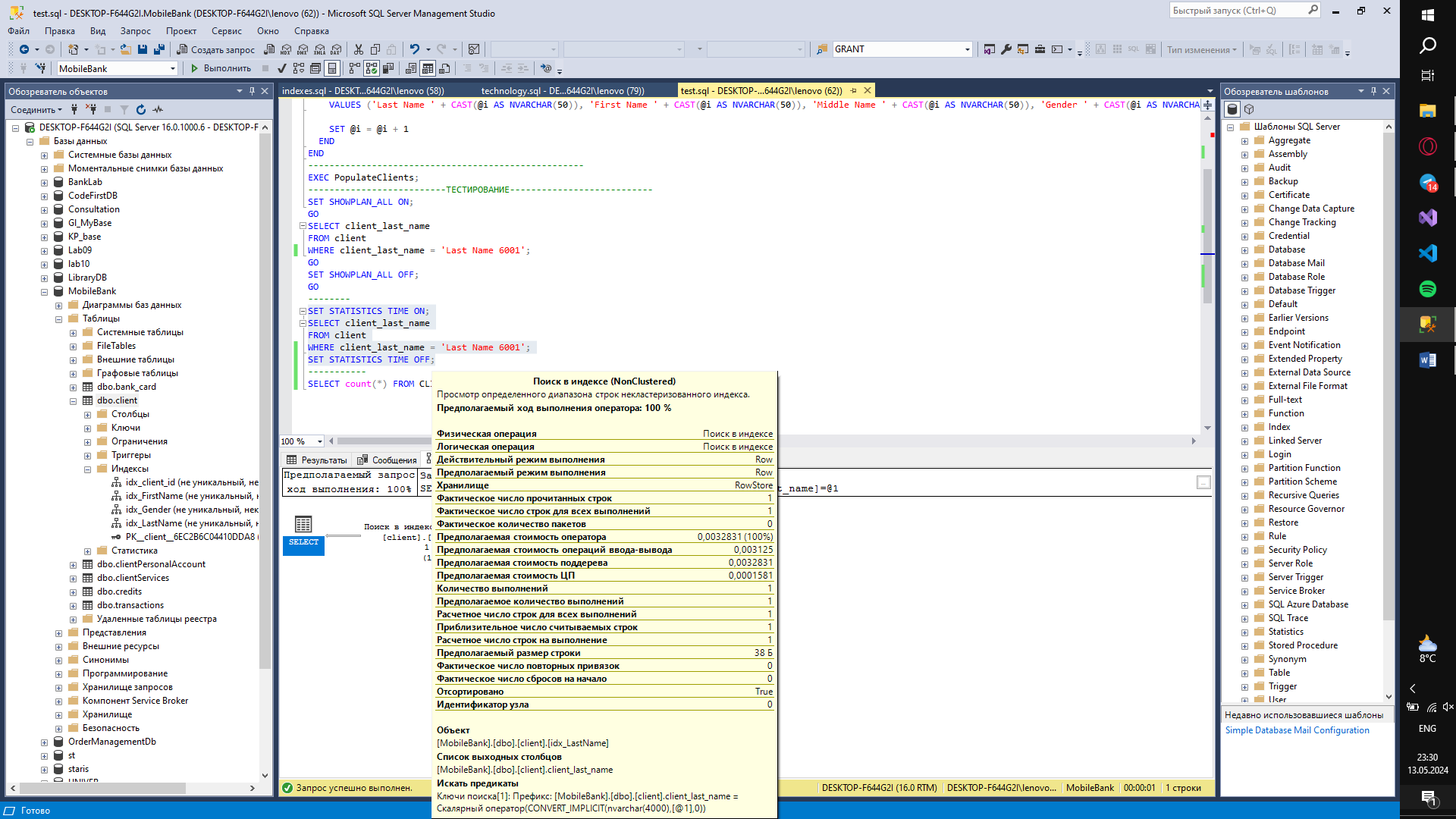


Рисунок 5.1 – План выполнения

# **6 Технология «Аналитика данных»**

Аналитика данных представляет собой процесс систематического анализа и интерпретации данных с целью извлечения полезной информации, выявления закономерностей и поддержания принятия обоснованных решений. В контексте бизнес-аналитики это часто включает сбор данных из различных источников, их обработку и анализ для понимания текущего состояния бизнеса и прогнозирования будущих тенденций.

В моем случае была разработана хранимая процедура T-SQL для анализа данных в базе данных свечного магазина. Процедура AnalyzeCandles выполняет несколько аналитических запросов, выводя результаты в текстовом формате с использованием курсоров. Код процедуры представлен в приложении Д.

Сначала объявляются переменные и курсор для выборки топ-5 самых продаваемых свечей. Курсор открывается, данные извлекаются построчно, и результаты выводятся с помощью команды PRINT. Затем курсор закрывается и освобождается.

Следующий запрос подсчитывает количество поставок по месяцам. Объявляются переменные и курсор для этой выборки, курсор открывается, данные извлекаются и выводятся, после чего курсор закрывается и освобождается.

Далее выполняется запрос для выбора поставщиков с ценами выше среднего. Объявляются переменные и курсор для этой выборки, курсор открывается, данные извлекаются и выводятся, затем курсор закрывается и освобождается.

Затем выбираются топ-5 пользователей по количеству заказов. Объявляются переменные и курсор для этой выборки, курсор открывается, данные извлекаются и выводятся, после чего курсор закрывается и освобождается.

Следующий запрос выбирает топ-5 популярных типов и ароматов свечей. Объявляются переменные и курсор для этой выборки, курсор открывается, данные извлекаются и выводятся, затем курсор закрывается и освобождается.

Наконец, выбираются топ-5 городов по количеству заказов. Объявляются переменные и курсор для этой выборки, курсор открывается, данные извлекаются и выводятся, после чего курсор закрывается и освобождается.

Процедура завершается. Для вызова процедуры используется команда EXEC AnalyzeCandles, которая выполняет процедуру и выводит результаты построчно в виде сообщений.

Результат выполнения процедуры представлен на рисунке 7.1.

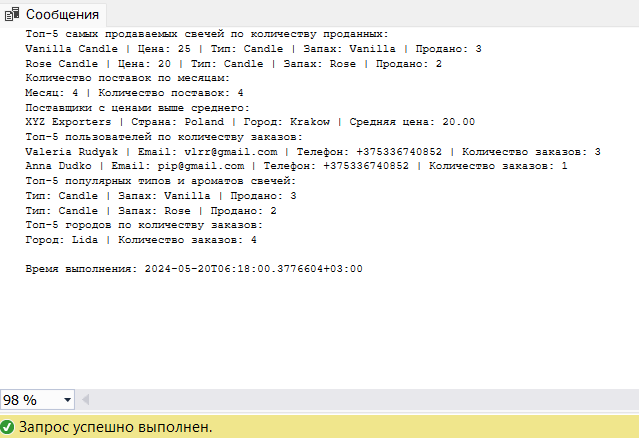


Рисунок 7.1 – Результат выполнения процедуры аналитики данных

Аналитика данных является ключевым элементом современного бизнеса, позволяющим принимать обоснованные решения и повышать эффективность работы. Использование мощных инструментов и методов анализа данных позволяет компаниям глубже понимать свою деятельность и лучше адаптироваться к изменениям на рынке.

# **Заключение**

В результате выполнения данного курсового проекта была реализована база данных для онлайн-магазина свечей «Любовь и свечи». Разработанная база данных для магазина свечей обеспечивает эффективное управление заказами, пользователями, продуктами и адресами доставки. Она включает в себя различные компоненты, такие как процедуры, функции, представления, триггеры и индексы, для обеспечения целостности данных и повышения производительности. Разработанные процедуры обеспечивают базовые операции управления данными, а также функционал поиска, сортировки и фильтрации, что сделает взаимодействие с системой интуитивно понятным и удобным для пользователей. Внедрение аналитики данных позволяет получать ценную информацию для улучшения бизнеса. Также были созданы пользователи базы данных с различными уровнями доступа – пользователь и администратор, что обеспечивает отделение прав доступа и повышает безопасность системы.

# **Список используемых источников**

1. Таблицы в SQL Server [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/sql/relational-databases/tables/tables?view=sql-server-ver16 – Дата доступа: 01.05.2024.
2. Триггеры в SQL Server [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/sql/relational-databases/triggers/create-dml-triggers?view=sql-server-ver16 – Дата доступа: 2.05.2024
3. Индексы в SQL Server [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/sql/relational-databases/indexes/indexes?view=sql-server-ver16 – Дата доступа: 10.05.2024
4. Оптимизация производительности в SQL Server [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/sql/relational-databases/performance/performance-center-for-sql-server?view=sql-server-ver16 – Дата доступа: 15.05.2024
5. Представления в SQL Server [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/sql/relational-databases/views/create-views?view=sql-server-ver16 – Дата доступа: 15.05.2024

# **Приложение А Создание таблиц**

|  |
| --- |
| CREATE TABLE Boss (  ID INT IDENTITY(1,1) PRIMARY KEY,  name nvarchar(255),  surname nvarchar(255),  email varchar(100),  phone\_number varchar(13),  );  CREATE TABLE Provider (  ID INT IDENTITY(1,1) PRIMARY KEY,  name nvarchar(255),  country nvarchar(255),  city nvarchar(255),  boss\_id int,  FOREIGN KEY (boss\_id) REFERENCES Boss(id),  );  CREATE TABLE Supply (  number\_of\_supply INT IDENTITY(1,1) PRIMARY KEY,  date date,  quantity\_of\_products int,  provider\_id int,  FOREIGN KEY (provider\_id) REFERENCES Provider(id),  );  CREATE TABLE Candle (  ID INT IDENTITY(1,1) PRIMARY KEY,  name nvarchar(255),  price int,  type nvarchar(255),  smell nvarchar(255),  photo nvarchar(255),  quantity int,  supply int,  FOREIGN KEY (supply) REFERENCES Supply(number\_of\_supply)  );  CREATE TABLE Address (  ID INT IDENTITY(1,1) PRIMARY KEY,  country nvarchar(255),  city nvarchar(255),  adress nvarchar(255),  );  CREATE TABLE Users (  login varchar(50) PRIMARY KEY,  name nvarchar(255),  surname nvarchar(255),  email varchar(100),  phone\_number varchar(13),  sex nvarchar(3),  );  CREATE TABLE Select\_products (  ID INT IDENTITY(1,1) PRIMARY KEY,  id\_of\_candles INT,  FOREIGN KEY (id\_of\_candles) REFERENCES Candle(ID),  quantity\_of\_candles INT,  login\_user varchar(50),  FOREIGN KEY (login\_user) REFERENCES Users(login),  );  CREATE TABLE Orders (  number INT IDENTITY(1,1) PRIMARY KEY,  adress INT,  FOREIGN KEY (adress) REFERENCES Address(id),  products INT,  FOREIGN KEY (products) REFERENCES Candle(ID),  login\_user varchar(50),  FOREIGN KEY (login\_user) REFERENCES Users(login),  date DATE,  ); |

# **Приложение Б Создание процедур**

|  |
| --- |
| -- Создание процедуры для оформления заказа  CREATE PROCEDURE ADD\_ORDER  @login\_user VARCHAR(50),  @address\_id INT,  @order\_date DATE  AS  BEGIN  DECLARE @id\_of\_candles INT;  DECLARE @quantity\_of\_candles INT;  DECLARE cart\_cursor CURSOR FOR  SELECT id\_of\_candles, quantity\_of\_candles  FROM Select\_products  WHERE login\_user = @login\_user;  OPEN cart\_cursor;  FETCH NEXT FROM cart\_cursor INTO @id\_of\_candles, @quantity\_of\_candles;  WHILE @@FETCH\_STATUS = 0  BEGIN  -- Вставка записи в таблицу Orders  INSERT INTO Orders (adress, products, login\_user, date)  VALUES (@address\_id, @id\_of\_candles, @login\_user, @order\_date);  FETCH NEXT FROM cart\_cursor INTO @id\_of\_candles, @quantity\_of\_candles;  END;  CLOSE cart\_cursor;  DEALLOCATE cart\_cursor;  -- Очистка корзины после оформления заказа  DELETE FROM Select\_products WHERE login\_user = @login\_user;  END;  GO  -- Пример вызова процедуры  EXEC ADD\_ORDER 'rr.valeri', 1, '2024-05-18';  EXEC ADD\_ORDER 'papita', 1, '2024-05-18';  select \* from Orders;  select \* from Select\_products;  select \* from Users;  drop PROCEDURE ADD\_ORDER  ---------------------------------------------  -- Процедура для добавления товаров в корзину  CREATE PROCEDURE ADD\_TO\_CART  @login\_user VARCHAR(50),  @id\_of\_candles INT,  @quantity\_of\_candles INT  AS  BEGIN  INSERT INTO Select\_products (id\_of\_candles, quantity\_of\_candles, login\_user)  VALUES (@id\_of\_candles, @quantity\_of\_candles, @login\_user);  END;  GO  --  EXEC ADD\_TO\_CART 'rr.valeri', 1, 3;  EXEC ADD\_TO\_CART 'rr.valeri', 2, 2;  EXEC ADD\_TO\_CART 'papita', 3, 2;  select \* from Select\_products;  select \* from Candle  select \* from Users  ----изм товаров--  CREATE PROCEDURE UpdateCandle  @CandleID INT,  @Name NVARCHAR(255),  @Price INT,  @Type NVARCHAR(255),  @Smell NVARCHAR(255),  @Photo NVARCHAR(255),  @Quantity INT,  @SupplyID INT  AS  BEGIN  UPDATE Candle  SET  name = @Name,  price = @Price,  type = @Type,  smell = @Smell,  photo = @Photo,  quantity = @Quantity,  supply = @SupplyID  WHERE  ID = @CandleID;  END;  -------------------------------  EXEC UpdateCandle  @CandleID = 4,  @Name = 'Lavender Incense',  @Price = 150,  @Type = 'Incense',  @Smell = 'Lavender',  @Photo = 'lavender.jpg',  @Quantity = 100,  @SupplyID = 2;  select \* from Candle  ---------------------------  --очистка корзины---  CREATE PROCEDURE ClearCart  @login\_user VARCHAR(50)  AS  BEGIN  DELETE FROM Select\_products WHERE login\_user = @login\_user;  END;  ---------------------  EXEC ClearCart 'papita';  select \* from Select\_products  ---------------------  ---Добавление товара в каталог  CREATE PROCEDURE AddCandle  @name NVARCHAR(255),  @price INT,  @type NVARCHAR(255),  @smell NVARCHAR(255),  @photo NVARCHAR(255),  @quantity INT,  @supply INT  AS  BEGIN  INSERT INTO Candle (name, price, type, smell, photo, quantity, supply)  VALUES (@name, @price, @type, @smell, @photo, @quantity, @supply);  END;  ------  EXEC AddCandle 'Orange Candle', 25, 'Candle', 'Orange', 'Orange\_candle.jpg', 100, 1;  select \* from Candle;  -----------------------------  --удаление товара из каталога  CREATE PROCEDURE DeleteCandle  @candleID INT  AS  BEGIN  DELETE FROM Candle WHERE ID = @candleID;  END;  --------------  EXEC DeleteCandle @candleID = 35;  -------------  ---процедура управления поставками---  CREATE PROCEDURE AddSupply  @date DATE,  @quantity\_of\_products INT,  @provider\_id INT  AS  BEGIN  INSERT INTO Supply (date, quantity\_of\_products, provider\_id)  VALUES (@date, @quantity\_of\_products, @provider\_id);  END;  ----  EXEC AddSupply @date = '2024-04-30', @quantity\_of\_products = 100, @provider\_id = 2;  --------------------------  ---процедура управления поставщиками  CREATE PROCEDURE ManageProvider  @Action VARCHAR(10), -- Возможные значения: 'INSERT', 'UPDATE', 'DELETE'  @ProviderID INT, -- ID поставщика  @Name NVARCHAR(MAX), -- Новое имя поставщика (для INSERT и UPDATE)  @Country NVARCHAR(MAX), -- Новая страна (для INSERT и UPDATE)  @City NVARCHAR(MAX), -- Новый город (для INSERT и UPDATE)  @BossID INT -- ID босса (для INSERT и UPDATE)  AS  BEGIN  IF @Action = 'INSERT'  BEGIN  INSERT INTO Provider (name, country, city, boss\_id)  VALUES (@Name, @Country, @City, @BossID);  END  ELSE IF @Action = 'UPDATE'  BEGIN  UPDATE Provider  SET name = @Name, country = @Country, city = @City, boss\_id = @BossID  WHERE ID = @ProviderID;  END  ELSE IF @Action = 'DELETE'  BEGIN  DELETE FROM Provider  WHERE ID = @ProviderID;  END  END;  ----  EXEC ManageProvider 'INSERT', NULL, 'Prov', 'Belarus', 'Lida', 2;  EXEC ManageProvider 'UPDATE', @ProviderID = 4, @Name = 'AAA', @Country = 'Poland', @City = 'Krakow', @BossID = 1;  EXEC ManageProvider 'DELETE', @ProviderID = 4;  -----  CREATE PROCEDURE AddUser  @login VARCHAR(50),  @name NVARCHAR(255),  @surname NVARCHAR(255),  @email VARCHAR(100),  @phone\_number VARCHAR(13),  @sex NVARCHAR(3)  AS  BEGIN  -- Вставка данных в таблицу Users  INSERT INTO Users (login, name, surname, email, phone\_number, sex)  VALUES (@login, @name, @surname, @email, @phone\_number, @sex);  END;  GO  ----------------------  EXEC AddUser  @login = 'john\_doe',  @name = 'John',  @surname = 'Doe',  @email = 'john.doe@example.com',  @phone\_number = '1234567890',  @sex = 'M';  select \* from Users  ----------------------  --удаление пользователя  CREATE OR ALTER PROCEDURE DeleteUserByLogin  @login varchar(50)  AS  BEGIN  DELETE FROM Users  WHERE login = @login;  END;  GO |

# **Приложение В Создание функций**

|  |
| --- |
| ---функция поиска по типу продукта---  CREATE FUNCTION GetCandlesByType  (  @type NVARCHAR(MAX)  )  RETURNS TABLE  AS  RETURN  (  SELECT \*  FROM Candle  WHERE type = @type  );  ----  SELECT \* FROM GetCandlesByType('candle');  ----------------------------------------  ---функция поиска по запаху---  CREATE FUNCTION GetCandlesBySmell  (  @Smell nvarchar(MAX)  )  RETURNS TABLE  AS  RETURN  (  SELECT \*  FROM Candle  WHERE smell = @Smell  );  -----  SELECT \* FROM dbo.GetCandlesBySmell('rose');  ---------------------------------------  --поиск по названию  CREATE FUNCTION FindCandleByName  (  @name NVARCHAR(MAX)  )  RETURNS TABLE  AS  RETURN  (  SELECT \*  FROM Candle  WHERE name LIKE '%' + @name + '%'  );  -------------------  SELECT \* FROM FindCandleByName('nil');  --поиск заказа по юзеру  CREATE FUNCTION GetOrdersByUser  (  @login\_user VARCHAR(50)  )  RETURNS TABLE  AS  RETURN  (  SELECT  o.number,  o.adress,  a.country,  a.city,  a.adress AS address\_details,  o.products,  c.name AS product\_name,  o.login\_user,  o.date  FROM  Orders o  JOIN  Address a ON o.adress = a.ID  JOIN  Candle c ON o.products = c.ID  WHERE  o.login\_user = @login\_user  );  GO  ---------------------------  SELECT \* FROM GetOrdersByUser('rr.valeri');  ---функция подсчета общей стоимости в корзине  CREATE FUNCTION CalculateTotalCost()  RETURNS INT  AS  BEGIN  DECLARE @totalCost INT;  SELECT @totalCost = SUM(sp.quantity\_of\_candles \* c.price)  FROM Select\_products sp  JOIN Candle c ON sp.id\_of\_candles = c.ID;  RETURN @totalCost;  END;  --------------------------  SELECT dbo.CalculateTotalCost() AS TotalCost;  --поиск товаров по дню поставки  CREATE FUNCTION GetProductsDeliveredOnDate  (  @deliveryDate DATE  )  RETURNS TABLE  AS  RETURN  (  SELECT c.name AS ProductName, s.date AS DeliveryDate  FROM Candle c  INNER JOIN Supply s ON c.supply = s.number\_of\_supply  WHERE s.date = @deliveryDate  );  ----------------------------  SELECT \* FROM dbo.GetProductsDeliveredOnDate('2024-04-04');  SELECT \* FROM Supply  --поиск товаров по поставщику  CREATE FUNCTION GetProductsByProvider  (  @providerID INT  )  RETURNS TABLE  AS  RETURN  (  SELECT c.name AS ProductName  FROM Candle c  INNER JOIN Supply s ON c.supply = s.number\_of\_supply  WHERE s.provider\_id = @providerID  );  -------------------------  SELECT \* FROM dbo.GetProductsByProvider(1); |

# **Приложение Г Создание триггеров**

|  |
| --- |
| --триггер для проверки количества товаров  CREATE TRIGGER CheckProductQuantity  ON Select\_products  INSTEAD OF INSERT  AS  BEGIN  DECLARE @id\_of\_candles INT;  DECLARE @quantity\_of\_candles INT;  DECLARE @available\_quantity INT;  SELECT @id\_of\_candles = inserted.id\_of\_candles,  @quantity\_of\_candles = inserted.quantity\_of\_candles  FROM inserted;  SELECT @available\_quantity = quantity  FROM Candle  WHERE ID = @id\_of\_candles;  IF @quantity\_of\_candles > @available\_quantity  BEGIN  PRINT 'Ошибка: Недостаточное количество товаров на складе.';  END  ELSE  BEGIN  INSERT INTO Select\_products (id\_of\_candles, quantity\_of\_candles, login\_user)  SELECT id\_of\_candles, quantity\_of\_candles, login\_user  FROM inserted;  PRINT 'Товар успешно добавлен в корзину.';  END  END;  GO  ------------------------------------  EXEC ADD\_TO\_CART 'papita', 3, 10000;  select \* from Select\_products  ------------------------------------  --триггер на оформление заказа  CREATE TRIGGER AfterOrderInsert  ON Orders  AFTER INSERT  AS  BEGIN  DECLARE @OrderID INT;  DECLARE @UserName NVARCHAR(255);  -- Получение информации о новом заказе  SELECT TOP 1  @OrderID = inserted.number,  @UserName = u.name  FROM inserted  JOIN Users u ON inserted.login\_user = u.login;  -- Вывод сообщения об успешном оформлении заказа  PRINT 'Заказ №' + CAST(@OrderID AS NVARCHAR(10)) + ' для пользователя ' + @UserName + ' успешно оформлен.';  END;  GO  -----------------------------  ---триггер на очистку корзины  CREATE TRIGGER Trigger\_ClearCart\_EmptyTable  ON Select\_products  INSTEAD OF DELETE  AS  BEGIN  IF NOT EXISTS (SELECT TOP 1 1 FROM Select\_products)  BEGIN  PRINT 'Таблица корзины уже пуста.';  END  ELSE  BEGIN  DELETE FROM Select\_products;  PRINT 'Корзина успешно очищена.';  END  END;  --триггер на одинаковые названия товаров  CREATE TRIGGER PreventDuplicateCandle  ON Candle  INSTEAD OF INSERT  AS  BEGIN  IF EXISTS (SELECT 1 FROM Candle c INNER JOIN inserted i ON c.name = i.name)  BEGIN  PRINT 'Ошибка: товар с таким названием уже существует в каталоге.';  END  ELSE  BEGIN  INSERT INTO Candle (name, price, type, smell, photo, quantity, supply)  SELECT name, price, type, smell, photo, quantity, supply  FROM inserted;  END  END;  ---------  --триггер на удаление товара из каталога  CREATE TRIGGER DeleteCandleTrigger  ON Candle  AFTER DELETE  AS  BEGIN  DECLARE @DeletedCandleID INT;  SELECT @DeletedCandleID = deleted.ID FROM deleted;  IF NOT EXISTS (SELECT \* FROM Candle WHERE ID = @DeletedCandleID)  BEGIN  PRINT 'Товар успешно удален из каталога.';  END  END;  ---триггер на добавление информации о пользователе  CREATE TRIGGER trg\_CheckDuplicateLogin  ON Users  INSTEAD OF INSERT  AS  BEGIN  DECLARE @login VARCHAR(50);    -- Получаем логин из вставляемых данных  SELECT @login = login FROM inserted;  -- Проверка на существование пользователя с таким же логином  IF EXISTS (SELECT 1 FROM Users WHERE login = @login)  BEGIN  -- Вывод сообщения об ошибке, если логин уже существует  RAISERROR('Пользователь с таким логином уже существует', 16, 1);  END  ELSE  BEGIN  -- Если логин уникален, вставка данных в таблицу Users  INSERT INTO Users (login, name, surname, email, phone\_number, sex)  SELECT login, name, surname, email, phone\_number, sex FROM inserted;  -- Вывод сообщения об успешном добавлении пользователя  PRINT 'Информация успешно добавлена!';  END  END;  GO  -------------------------  --триггер на оформление заказа  drop TRIGGER OrderSuccessTrigger  ON Orders  AFTER INSERT  AS  BEGIN  DECLARE @OrderID INT;  DECLARE @UserName NVARCHAR(255);  -- Получение информации о новом заказе  SELECT @OrderID = inserted.number, @UserName = users.name  FROM inserted  JOIN Users ON inserted.login\_user = Users.login;  -- Вывод сообщения об успешном оформлении заказа  PRINT 'Заказ №' + CAST(@OrderID AS NVARCHAR(10)) + ' для пользователя ' + @UserName + ' успешно оформлен.';  END;  GO  CREATE OR ALTER TRIGGER UserDeletedTrigger  ON Users  AFTER DELETE  AS  BEGIN  DECLARE @deletedLogin VARCHAR(50);  SELECT @deletedLogin = login FROM deleted;  PRINT 'Пользователь с логином ' + @deletedLogin + ' успешно удален.';  END;  GO |

# **Приложение Д Аналитика данных**

|  |
| --- |
| CREATE PROCEDURE AnalyzeCandles  AS  BEGIN  -- Топ-5 самых продаваемых свечей по количеству проданных  DECLARE @name NVARCHAR(255), @price INT, @type NVARCHAR(255), @smell NVARCHAR(255), @total\_sold INT;  DECLARE top5\_cursor CURSOR FOR  SELECT TOP 5  c.name, c.price, c.type, c.smell, SUM(sp.quantity\_of\_candles) AS total\_sold  FROM  Candle c  JOIN  Select\_products sp ON c.ID = sp.id\_of\_candles  GROUP BY  c.name, c.price, c.type, c.smell  ORDER BY  total\_sold DESC;  OPEN top5\_cursor;  FETCH NEXT FROM top5\_cursor INTO @name, @price, @type, @smell, @total\_sold;  PRINT 'Топ-5 самых продаваемых свечей по количеству проданных:';  WHILE @@FETCH\_STATUS = 0  BEGIN  PRINT @name + ' | Цена: ' + CAST(@price AS NVARCHAR(10)) + ' | Тип: ' + @type + ' | Запах: ' + @smell + ' | Продано: ' + CAST(@total\_sold AS NVARCHAR(10));  FETCH NEXT FROM top5\_cursor INTO @name, @price, @type, @smell, @total\_sold;  END  CLOSE top5\_cursor;  DEALLOCATE top5\_cursor;  -- Количество поставок по месяцам  DECLARE @month INT, @num\_supplies INT;  DECLARE supplies\_cursor CURSOR FOR  SELECT  MONTH(s.date) AS month,  COUNT(\*) AS num\_supplies  FROM  Supply s  GROUP BY  MONTH(s.date)  ORDER BY  month;  OPEN supplies\_cursor;  FETCH NEXT FROM supplies\_cursor INTO @month, @num\_supplies;  PRINT 'Количество поставок по месяцам:';  WHILE @@FETCH\_STATUS = 0  BEGIN  PRINT 'Месяц: ' + CAST(@month AS NVARCHAR(2)) + ' | Количество поставок: ' + CAST(@num\_supplies AS NVARCHAR(10));  FETCH NEXT FROM supplies\_cursor INTO @month, @num\_supplies;  END  CLOSE supplies\_cursor;  DEALLOCATE supplies\_cursor;  -- Поставщики с ценами выше среднего  DECLARE @provider\_name NVARCHAR(255), @country NVARCHAR(255), @provider\_city NVARCHAR(255), @avg\_price DECIMAL(10,2);  DECLARE suppliers\_cursor CURSOR FOR  SELECT  p.name AS provider\_name, p.country, p.city, AVG(c.price) AS avg\_price  FROM  Provider p  JOIN  Supply s ON p.ID = s.provider\_id  JOIN  Candle c ON s.number\_of\_supply = c.supply  GROUP BY  p.name, p.country, p.city  HAVING  AVG(c.price) > (SELECT AVG(price) FROM Candle)  ORDER BY  avg\_price DESC;  OPEN suppliers\_cursor;  FETCH NEXT FROM suppliers\_cursor INTO @provider\_name, @country, @provider\_city, @avg\_price;  PRINT 'Поставщики с ценами выше среднего:';  WHILE @@FETCH\_STATUS = 0  BEGIN  PRINT @provider\_name + ' | Страна: ' + @country + ' | Город: ' + @provider\_city + ' | Средняя цена: ' + CAST(@avg\_price AS NVARCHAR(10));  FETCH NEXT FROM suppliers\_cursor INTO @provider\_name, @country, @provider\_city, @avg\_price;  END  CLOSE suppliers\_cursor;  DEALLOCATE suppliers\_cursor;  -- Топ-5 пользователей по количеству заказов  DECLARE @user\_name NVARCHAR(255), @user\_surname NVARCHAR(255), @user\_email VARCHAR(100), @user\_phone\_number VARCHAR(13), @total\_orders INT;  DECLARE users\_cursor CURSOR FOR  SELECT TOP 5  u.name, u.surname, u.email, u.phone\_number, COUNT(\*) AS total\_orders  FROM  Users u  JOIN  Orders o ON u.login = o.login\_user  GROUP BY  u.name, u.surname, u.email, u.phone\_number  ORDER BY  total\_orders DESC;  OPEN users\_cursor;  FETCH NEXT FROM users\_cursor INTO @user\_name, @user\_surname, @user\_email, @user\_phone\_number, @total\_orders;  PRINT 'Топ-5 пользователей по количеству заказов:';  WHILE @@FETCH\_STATUS = 0  BEGIN  PRINT @user\_name + ' ' + @user\_surname + ' | Email: ' + @user\_email + ' | Телефон: ' + @user\_phone\_number + ' | Количество заказов: ' + CAST(@total\_orders AS NVARCHAR(10));  FETCH NEXT FROM users\_cursor INTO @user\_name, @user\_surname, @user\_email, @user\_phone\_number, @total\_orders;  END  CLOSE users\_cursor;  DEALLOCATE users\_cursor;  -- Топ-5 популярных типов и ароматов свечей  DECLARE popular\_candles\_cursor CURSOR FOR  SELECT TOP 5  c.type, c.smell, SUM(sp.quantity\_of\_candles) AS total\_sold  FROM  Candle c  JOIN  Select\_products sp ON c.ID = sp.id\_of\_candles  GROUP BY  c.type, c.smell  ORDER BY  total\_sold DESC;  OPEN popular\_candles\_cursor;  FETCH NEXT FROM popular\_candles\_cursor INTO @type, @smell, @total\_sold;  PRINT 'Топ-5 популярных типов и ароматов свечей:';  WHILE @@FETCH\_STATUS = 0  BEGIN  PRINT 'Тип: ' + @type + ' | Запах: ' + @smell + ' | Продано: ' + CAST(@total\_sold AS NVARCHAR(10));  FETCH NEXT FROM popular\_candles\_cursor INTO @type, @smell, @total\_sold;  END  CLOSE popular\_candles\_cursor;  DEALLOCATE popular\_candles\_cursor;  -- Топ-5 городов по количеству заказов  DECLARE @order\_city NVARCHAR(255);  DECLARE cities\_cursor CURSOR FOR  SELECT TOP 5  a.city, COUNT(\*) AS total\_orders  FROM  Orders o  JOIN  Address a ON o.adress = a.ID  GROUP BY  a.city  ORDER BY  total\_orders DESC;  OPEN cities\_cursor;  FETCH NEXT FROM cities\_cursor INTO @order\_city, @total\_orders;  PRINT 'Топ-5 городов по количеству заказов:';  WHILE @@FETCH\_STATUS = 0  BEGIN  PRINT 'Город: ' + @order\_city + ' | Количество заказов: ' + CAST(@total\_orders AS NVARCHAR(10));  FETCH NEXT FROM cities\_cursor INTO @order\_city, @total\_orders;  END  CLOSE cities\_cursor;  DEALLOCATE cities\_cursor;  END;  -----------------------------  EXEC AnalyzeCandles; |

# **Приложение Ж Реализация ролей**

|  |
| --- |
| CREATE LOGIN AdministratorCandle WITH PASSWORD = 'admin\_for\_candle';  CREATE LOGIN ClientCandle WITH PASSWORD = 'client\_for\_candle';  CREATE USER admin\_for\_candle FOR LOGIN AdministratorCandle;  CREATE USER client\_for\_candle FOR LOGIN ClientCandle;  CREATE ROLE AdministratorRole;  CREATE ROLE ClientRole;  ALTER ROLE AdministratorRole ADD MEMBER admin\_for\_candle;  ALTER ROLE ClientRole ADD MEMBER client\_for\_candle;  CREATE LOGIN Guestt WITH PASSWORD = 'Guestt';  CREATE USER Guestt FOR LOGIN Guestt;  CREATE ROLE GuestRole;  ALTER ROLE GuestRole ADD MEMBER Guestt;  GRANT SELECT, UPDATE, INSERT, DELETE ON dbo.Users TO AdministratorRole;  GRANT SELECT, UPDATE, INSERT, DELETE ON dbo.Candle TO AdministratorRole;  GRANT SELECT, UPDATE, INSERT, DELETE ON dbo.Adress TO AdministratorRole;  GRANT SELECT, UPDATE, INSERT, DELETE ON dbo.Orders TO AdministratorRole;  GRANT SELECT, UPDATE, INSERT, DELETE ON dbo.Boss TO AdministratorRole;  GRANT SELECT, UPDATE, INSERT, DELETE ON dbo.Supply TO AdministratorRole;  GRANT SELECT, UPDATE, INSERT, DELETE ON dbo.Provider TO AdministratorRole;  GRANT SELECT, UPDATE, INSERT, DELETE ON dbo.Select\_products TO AdministratorRole;  GRANT EXECUTE ON dbo.ADD\_TO\_CART TO AdministratorRole;  GRANT EXECUTE ON dbo.ClearCart TO AdministratorRole;  GRANT EXECUTE ON dbo.AddCandle TO AdministratorRole;  GRANT EXECUTE ON dbo.DeleteCandle TO AdministratorRole;  GRANT EXECUTE ON dbo.ADD\_ORDER TO AdministratorRole;  GRANT EXECUTE ON dbo.AddSupply TO AdministratorRole;  GRANT EXECUTE ON dbo.ManageProvider TO AdministratorRole;  GRANT EXECUTE ON dbo.UpdateCandle TO AdministratorRole;  GRANT EXECUTE ON dbo.AddUser TO AdministratorRole;  GRANT EXECUTE ON dbo.AnalyzeCandles TO AdministratorRole;  GRANT EXECUTE ON dbo.ADD\_TO\_CART TO ClientnRole;  GRANT EXECUTE ON dbo.ClearCart TO ClientnRole;  GRANT EXECUTE ON dbo.ADD\_ORDER TO ClientnRole;  GRANT EXECUTE ON dbo.AddUser TO ClientnRole;  GRANT EXECUTE ON dbo.AddUser TO GuestRole;  --ему доступны функции  --проверовчка  execute as user = 'admin\_for\_candle';  exec dbo.AnalyzeCandles  revert;  execute as user = 'client\_for\_candle';  exec dbo.AnalyzeCandles  revert;  execute as user = 'Guestt';  exec dbo.AnalyzeCandles  revert; |