Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Базы данных (БД)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему:

«База данных интернет-магазина мебели»

БГУИР КП 1-40 01 01 423 ПЗ

Студент: гр. 951004 Турок К.А.

Руководитель: асс. Марина И.М.

Минск 2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Сегодня информационные технологии прочно вошли в нашу жизнь. При этом несмотря на столь стремительный прогресс, компьютер и сопутствующие ему технологии продолжают развиваться стремительными темпами. Быстрое развитие вычислительной техники привело к прочному входу компьютера в жизнь человека. С появлением новых, более современных и быстродействующих компьютеров все больше увеличиваются области их применения. Сейчас трудно найти сферы деятельности, где бы не применялась вычислительная техника. Использование компьютеров сильно изменило многие профессии человека. Огромное увеличение объёма информационных потоков в конце XX века привело к тому, что успешность любого предприятия очень сильно зависит от способности быстрого получения и обработки информации. Внедрение ЭВМ, вычислительных систем и средств автоматизации в различные сферы производства позволяет оперировать большими объемами информации. Применение ЭВМ становится необходимым условием успешного выполнения производственных задач. Наглядность и полнота представления большого объема информации на экране монитора способствует более легкому ее восприятию и быстрому анализу.

Постепенно развивались персональные компьютеры. Их популярность можно объяснить тем, что они просты в эксплуатации, потребляют мало энергии и обладают высоким быстродействием. Такие компьютеры можно легко размещать на рабочих местах. Организация автоматизированных рабочих мест на производстве способствует сокращению потерь времени, связанных с ежедневными операциями по учету и хранению информации, контролю протекания производственного процесса. Такие системы позволяют обеспечивать:

- быстрый доступ к любой информации;
- анализ состояния производственного процесса и своевременное принятие решений в случае обнаружения сбоев;
- контроль исполнения решений и т.д.

С развитием персональных компьютеров развивается и индустрия интернет-магазинов. Интернет-магазины создаются для большого числа пользователей с разными предпочтениями.

Выбранная тема считается актуальной на сегодняшний день, так как сегодня миллионы людей ежедневно, не выходя из дома, покупают различные товары в электронных магазинах. В мире, огромными темпами растет количество пользователей Internet и, как следствие, количество онлайн покупателей.

Онлайн-магазины существенно уменьшают издержки производителя, сэкономив на содержании обычного магазина, расширяют рынки сбыта. Это дает онлайн-магазинам преимущество перед обычными магазинами. Этот

момент является существенным при переходе производителей с «обычной» торговли на «онлайн».

Высокое качество продукции, умение донести информацию о продукте до потребителя и эффективная система сбыта, делает предприятие успешным на рынке. Во многих компаниях встречаются проблемы сбыта, которые мешают эффективно работать отделу продаж, и не исчезают даже с подбором хороших продавцов. Решить их можно только путем автоматизации процесса продаж. Более широкий, или концептуальный, подход рассматривает электронный бизнес как способ предпринимательства, способствующий достижению стратегического успеха в новую информационную эпоху. При таком понимании электронный бизнес отнюдь не сводится к информационным технологиям или активности в Интернете. Электронная коммерция затрагивает все аспекты бизнеса, включая стратегию, процессы, организацию и технологию, и выводит его далеко за сложившиеся границы. Online shop или e-shop — веб-сайт, рекламирующий товар или услугу, принимающий заказы на покупку, предлагающий пользователю выбор варианта расчета, способа получения заказа и выписывающий счет на оплату.

Онлайн-магазины стали важной частью нашей современной жизни, поэтому за последнее время можно заметить появления множества онлайн магазинов. Для упрощения части их работы и достижения необходимой автоматизации им необходимо соответствующее программное обеспечение. В совокупности с неспадающей потребностью в мебели, онлайн-магазины этой отрасли, пожалуй, достигли пика популярности. По этой причине выбор темы курсового проекта остановился на онлайн-магазине мебели.

Целью курсового проектирования является создание базы данных программного средства интернет-магазина. База данных ползволяет быстро и с минимальным количеством ошибок оперировать данными. В случае курсового проекта – с данными интернет-магазина мебели.

Первый раздел содержит сравнительный анализ существующих по данной тематике технических научных решений. Также в разделе ставится общая задача курсового проектирования и формулируются конкретные задачи.

Во втором разделе описываются требования и функциональные требования. Задачей является создание основных требований к проектируемой базе данных.

Третий раздел затрагивает модель базы данных.

В четвертом разделе обосновывается бизнес-логика.

Пятый раздел содержит способы тестирования базы данных и работоспособности её отдельных функций.

В заключении описываются результаты курсового проектирования и приводится список использованной литературы.

1. АНАЛИЗ ПРОТОТИПОВ И ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ

1.1. Анализ актуальности программного средства

В наше время стремительно развиваются сети онлайн-магазинов. Пользователи все чаще предпочитают посещать интернет-магазины, чем осуществлять покупки ранее популярным походом в магазин. Это делает более удобным выбор и заказ товаров. У покупателей нет необходимости искать ближайший магазин, так как просмотреть необходимые товары можно, просто перейдя по ссылке. Рынок мебели всегда является одним из самых стабильных и распространенных, по сравнению с некоторыми другими отраслями: у людей всегда есть необходимость приобретать товары домашнего интерьера. Современные вычислительные средства позволяют обеспечить доступ большого числа пользователей к каталогу товаров большого числа магазинов. Для более быстрого и качественного доступа к информации, необходимо создать нормально функционирующую базу данных, обеспечивающую необходимый доступ к информации интернет-магазина.

1.2. Анализ существующих программных средств

1.2.1. «Программа для автоматизации мебельного производства» от компании Maxtarget.

Программа предназначена для комплексной автоматизации компаний, занимающихся производством и продажей мебели. Данная программа позволяет совершенствовать процесс обслуживания клиентов благодаря ведению всестороннего учета. Имеется возможность ведения справочников товаров, материалов и услуг, учет заказов клиентов, контроль производства мебели, контроль поступления, списания и перемещения материалов, учет остатков. Регистрация продаж. Хранение контактной информации клиентов. Конфигурация легко и быстро настраивается под конкретные требования заказчика. Интерфейс программы изображен на рисунке 1.1

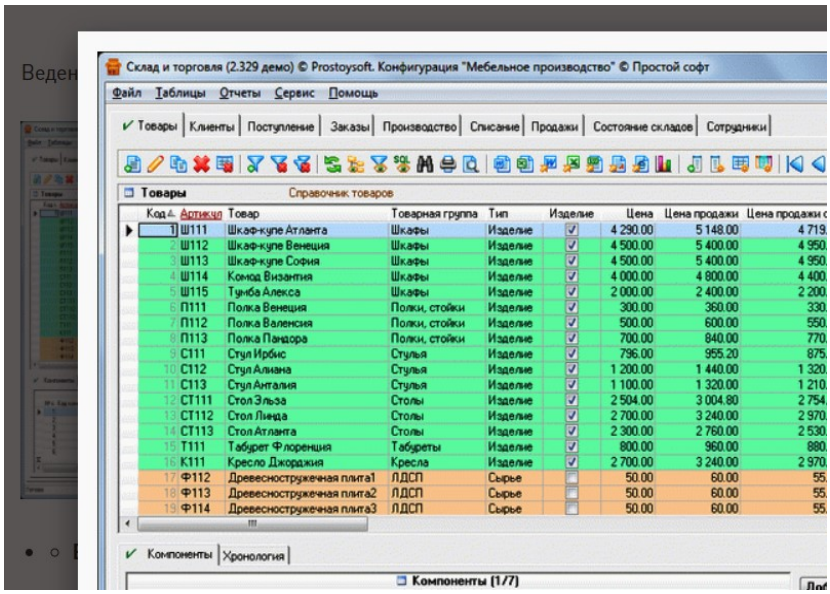

Рисунок 1.1 – Интерфейс программы от Maxtarget.by

Достоинства программы:

- наличие справочника товаров;
- ведение базы клиентов;
- отображение поступления материалов;
- учет заказов;
- наличие блока производства;
- возможность списания материалов;
- отображение состояния склада.

Недостатки программы:

- невозможность вывода информации в файл.

1.2.2 “Программа мебели” от usu.kz

Данная программа позволяет работать как со специализированным торговым оборудованием, так и без него, распечатывая чеки и накладные через обычный принтер. Это может совершать качественный и упорядоченный учет мебели. Управление мебельным магазином позволяет осуществлять контроль мебели, а также персонала. В конце дня или месяца можно сделать свод по продавцам за определенный период.

Программу мебели можно настроить индивидуально под каждый вид деятельности. Причем, отталкиваясь от базовой комплектации программы мебели, каждый мебельный магазин будет иметь индивидуально настроенную под себя программу мебели. Интерфейс программы изображен на рисунке 1.2.

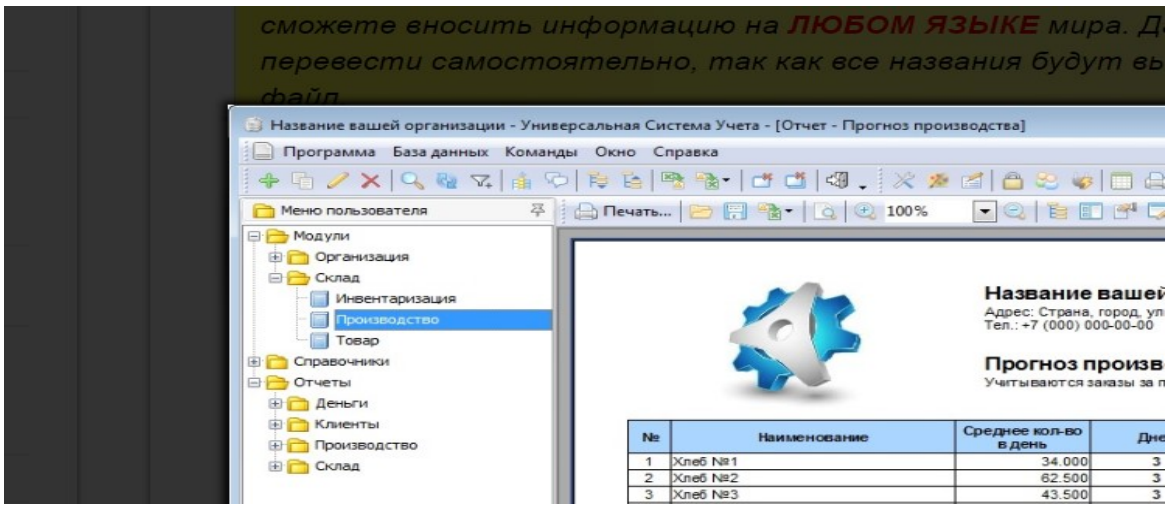

Рисунок 1.2 – Интерфейс программы от usu.kz

Достоинства программы:

- возможность вывода в файл или на чек;
- настройка программы «под себя»;
- контроль персонала;
- заполнение бланков;
- создание маршрута для водителя;
- возможность списания товара;
- выбор типа древесины.

Недостатки программы:

- невозможность учета поступления товара.

1.2.3 Программное средство от компании 1С

Программа предназначена для автоматизации процесса у компаний, занимающихся производством и продажей мебели. Программа работает с базой данных клиентов, работников и товаров предприятия. Данное программное обеспечение предназначено не только для учета услуг по продаже мебели, но и для контроля учета производства. Программа предназначена для работников предприятия и продавцов. Интерфейс программы изображен на рисунке 1.3.

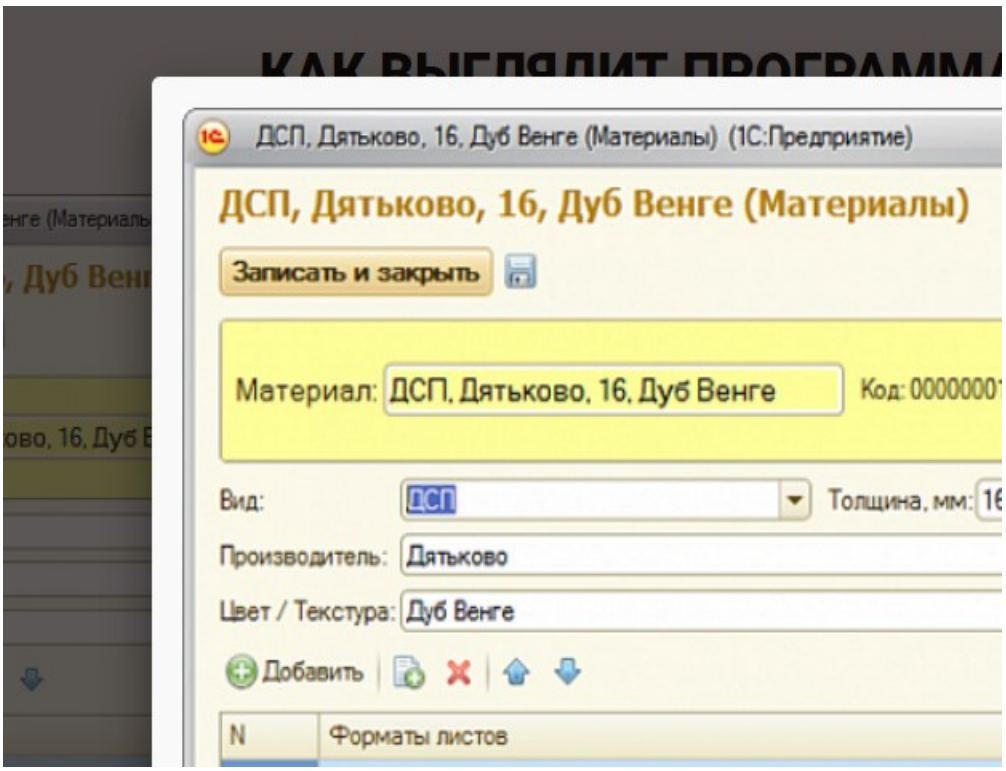

Рисунок 1.3 – Интерфейс программы от 1С

Достоинства программы:

- работа с базой данных клиентов;
- возможность контроля производства;
- информация о работниках компании;
- вывод информации в файл;
- возможность выбора материала товара;
- возможность выбора цвета товара;
- сортировка заказов.

Недостатки программы:

- нет возможности списания материалов;
- нет возможности отображения состояния склада.

1.2.4 Программное средство от ekam.ru

Данное программное средство предназначено для автоматизации учета услуг по продаже мебели. Программа позволяет продавать товары под заказ. Программа может выводить статистику о проданных товарах, что позволяет отслеживать наиболее значимые товары и уменьшать поставку менее популярных. Программное средство предназначено как для магазинов, реализующих продажу мебели, так и для производств. Интерфейс программы изображен на рисунке 1.4.

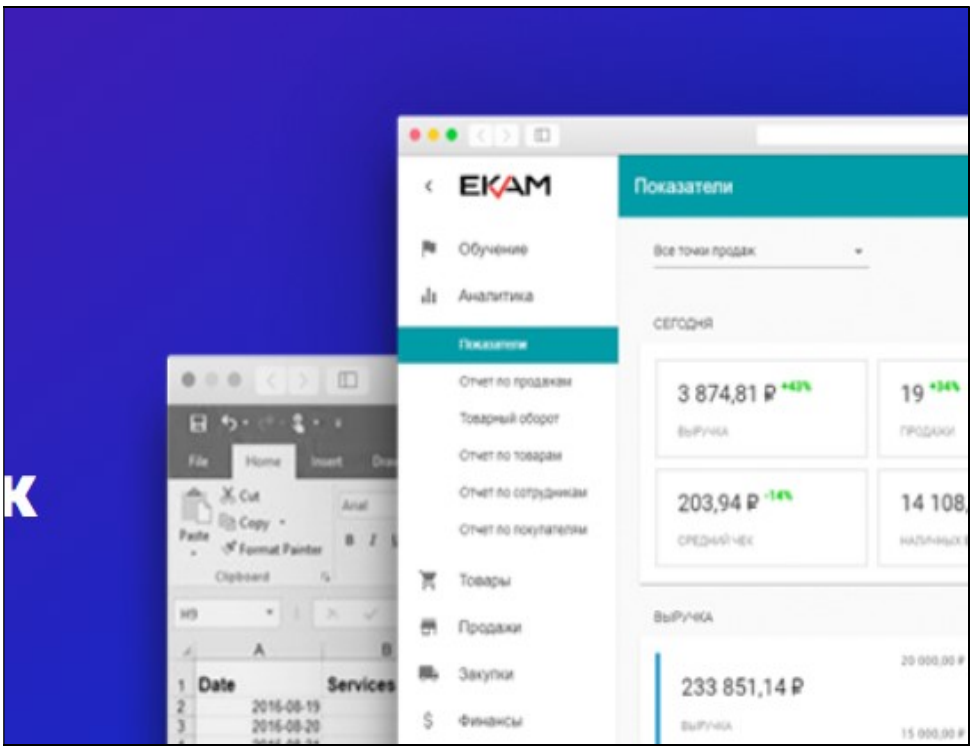

Рисунок 1.4 – Интерфейс программы от ekam.ru

Достоинства программы

- возможность изменения ассортимента каталогов;
- возможность связи между поставщиками и покупателями;
- возможность оценивать предоставляемые услуги;
- работа с базой данных клиентов;
- возможность объединения всего оборудования в одну систему (кассовый аппарат, весы, фискальный регистратор, платежный терминал, принтер чеков и др.).

Недостатки программы

- невозможность отслеживания пути товара;
- невозможность выора материала товара;
- нет возможности списания материалов.

1.2.5 Программное средство от Prostoysoft

Программа предназначена для автоматизации ведения учета в мебельных магазинах. Данная конфигурация является универсальным средством, с помощью которого вести учет продаж и остатков. Практически любая мебель при продаже требует дальнейшей сборки, в данной программе предназначена комплектация мебели по упаковкам, каждая упаковка имеет уникальный артикул, что позволяет легко и быстро собрать полный комплект на складе с последующей отгрузкой покупателю. Имеется возможность ведения справочников товаров и услуг, клиентов, учет продаж и оплат, складской учет, ведение кассы магазина, учет расходных операций.

Также в программе настроен блок логистики, куда при продаже поступают заказы на доставку с последующим подбором даты отправки и перевозчика. Для каждого перевозчика можно легко формировать лист доставки на определенную дату. Интерфейс программы изображен на рисунке 1.5.

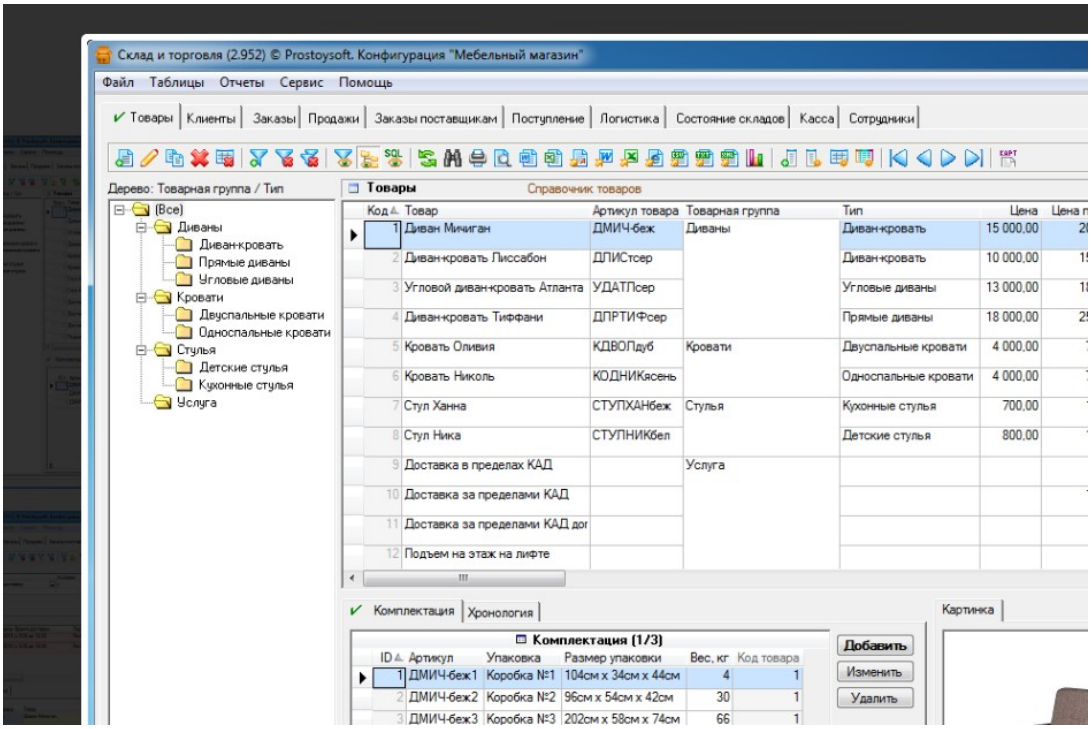

Рисунок 1.5 – Интерфейс программы от ПростойСофт

Достоинства программы:

- ведение справочника товаров;
- учет клиентов;
- складской учет;
- ведение кассы;
- импорт и экспорт данных;
- автоматизация работы отдела логистики;
- хранение информации о сотрудниках;
- набор отчетов с возможностью добавления.

Недостатки программы:

- невозможность отслеживания нахождения товара;
- невозможность списания товара;
- нет возможности работы с базой данных клиентов.

1.2.6 Программное средство CRM

CRM – это программа, которая автоматизирует работу с клиентами и облегчает процесс продажи товара. Она хранит документацию; ведет учет клиентов, помогает планировать задачи и отслеживать их выполнение. Про-

граммное средство предназначено как для магазинов, реализующих продажу мебели, так и для производств. Интерфейс программы приеден на рисунке 1.6.

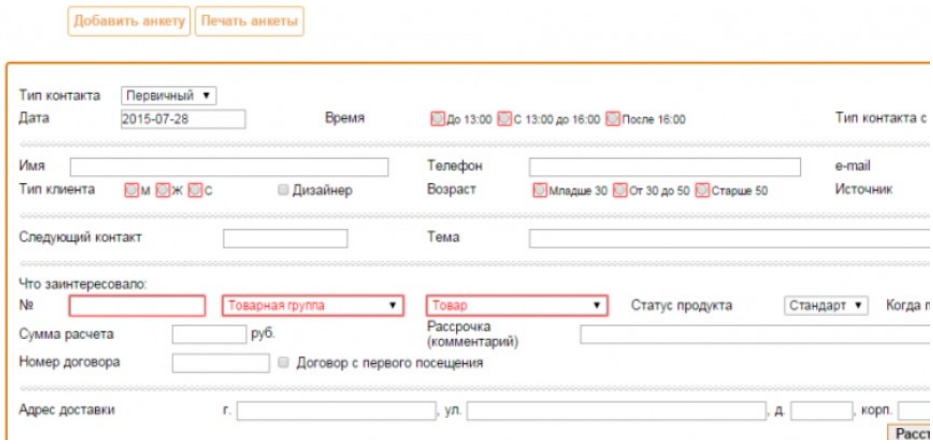

Рисунок 1.6 – Интерфейс программы CRM

Достоинства программы:

- объединение всех торговых точек;
- работа с базой данных клиентов;
- отображение статистики;
- сохранение клиентов и отображение их предыдущих покупок;
- составление договора;
- оформление рассрочки.

Недостатки программы:

- невозможность отслеживания нахождения товара;
- невозможность составления маршрута для доставки товара;
- невозможность подключения другого оборудования (кассовый аппарат, банковский терминал и т. д.).

1.3.Постановка задачи

На основании проанализированных схожих продуктов были выведены основные требования к проектируемой базе данных:

– база данных должна корректно обрабатывать входные данные;

– база данных должна предусмотреть добавление, обновление корректных данных и реакцию на триггеры добавления, обновления;

– база данных должна предусмотреть удаление данных и реакцию на триггеры удаления.

– база данных должна удалять данные в соответствии с запросами к БД;

– база данных должна обеспечить обобщение и систематизацию: товаров, заказов, доставок, для последующей эффективной работы над ними в системе;

Также для функционирования системы необходимо предусмотреть следующие элементы в базе данных:

– предоставление информации для каждого продукта (представление);

– предоставление информации о самых популярных продуктов по количеству заказов (представление);

– предоставление информации о самых эффективных доставщиках по количеству доставок (представление);

– предоставление информации о самых дорогих заказах (представление);

– предоставление информации о пользователях (представление);

– предоставление информации о заказах (представление);

– получение ID характеристики продукта (функция);

– получение значения характер характеристики продукта (функция);

– создание заказа (функция);

– получение времени проведенным пользователем на сайте (функция);

– создание пользователя (функция);

– обновление информации о пользователе (процедура);

– очистка статистики времени проведенным на сайте пользователями (процедура);

– установка времени создания пользователя (триггер);

– установка времени создания заказа (триггер);

– очистка статистики времени проведенным на сайте пользователями при достижении максимальной величины записей (триггер).

Для полноты представления системы необходимо учитывать существующие характеристики, типы товаров, а также к каким автомобилям относятся те или иные предметы интерьера.

В качестве языка базы данных была выбрана СУБД MySQL, поскольку она сочетает в себе широкие возможности, и простоту написания реляционных баз данных.

2. АНАЛИЗ ТРЕБОВАНИЙ И РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ

2.1 Требования к оборудованию

В качестве СУБД была выбрана MySQL (рисунок 2.1) т.к. она имеет необходимую простоту использования, наглядность, гибкость, низкую стоимость владения (относительно платных СУБД), а также масштабируемость и производительность.

MySQL позволяет хранить целочисленные значения со знаком и беззнаковые, длиной в 1, 2, 3, 4 и 8 байтов, работает со строковыми и текстовыми данными фиксированной и переменной длины, позволяет осуществлять SQL-команды SELECT, DELETE, INSERT, REPLACE и UPDATE, обеспечивает полную поддержку операторов и функций в SELECT- и WHERE- частях запросов, работает с GROUP BY и ORDER BY, поддерживает групповые функции COUNT(), AVG(), STD(), SUM(), MAX() и MIN(), позволяет использовать JOIN в запросах, в т.ч. LEFT OUTER JOIN и RIGHT OUTER JOIN, поддерживает репликацию, транзакции, работу с внешними ключами и каскадные изменения на их основе, а также обеспечивает многие другие функциональные возможности[7].

Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа MyISAM, поддерживающие полнотекстовый поиск, так и таблицы InnoDB, поддерживающие транзакции на уровне отдельных записей. Есть и другие типы таблиц, разработанные сообществом.

СУБД MySQL появилась в 1995. Написана на C и C++, протестирована на множестве различных компиляторов и работает на различных платформах. С 2010 года разработку и поддержку MySQL осуществляет корпорация Oracle. Продукт распространяется как под GNU GPL, так и под собственной коммерческой лицензией. Однако по условиям GPL, если какая-либо программа включает исходные коды MySQL, то и эта программа тоже должна распространяться по лицензии GPL. Для нежелающих открывать исходные тексты своих программ как раз предусмотрена коммерческая лицензия, которая, в дополнение к возможности разработки под «закрытой» лицензией, обеспечивает качественную сервисную поддержку. Сообществом разработчиков MySQL созданы различные ответвления — Drizzle, OurDelta, Percona Server и MariaDB, все эти ответвления уже существовали на момент получения прав на MySQL корпорацией Oracle[2].

Рисунок 2.1 – Логотип MySQL

В качестве среды проектирования баз данных, была выбрана Sparx Enterprise Architect (рисунок 2.2), поскольку она предоставляет удобный и мощный инструментарий для проектирования моделей баз-данных разных уровней (от логического до визического). Кроме того, предоставляется инструментарий кодогенерации, что значительно ускоряет разработку и снижает вероятность ошибок.

Sparx Enterprise Architect - это инструмент моделирования полного жизненного цикла на основе UML, который используется для планирования, проектирования и создания программно-интенсивных систем и бизнес-процессов. Разработанный Sparx Systems, австралийской компанией-разработчиком программного обеспечения, основанной Джеффри Спарксом в 1996 году, Enterprise Architect доступен в четырех различных редакциях (вводная профессиональная версия, корпоративная командная версия, многофункциональная унифицированная версия и, наконец, версия Ultimate), каждая настроены для различных сценариев использования.

На данный момент у Enterprise Architect более 850,000 XNUMX пользователей по всему миру. Его пользователи охватывают широкий спектр отраслей, включая аэрокосмическую и оборонную, автомобильную, банковскую и финансовую, электротехническую, медицинскую, исследовательскую и академическую сферы, розничную торговлю, транспорт и коммунальные

услуги. С момента своего первого выпуска Enterprise Architect стал предпочтительным инструментом моделирования UML для разработчиков, консультантов и аналитиков, которые используют его не только для моделирования архитектуры своих систем, но и для обработки реализации этих моделей на протяжении всего жизненного цикла разработки приложений.

Набор инструментов Sparx Enterprise Architect разработан на базе UML, что помогает создать простые и легко обслуживаемые системы[1].

Он обладает рядом преимуществ:

– скорость — загружает огромные модели за секунды;

– стабильность и исполнительность;

– программа обеспечивает доступ огромному количеству пользователей, которым доступен один и тот же вид на предприятие, как на локализованной территории, так и по всему Земному шару;

– позволяет создавать динамические модели симуляции предприятий;

– предоставляет возможность полного слежения за процессами: от требований, анализа и дизайна модели до реализации и развертывания;

– позволяет группе лиц управлять комплексной информацией;

– поддерживает много популярных языков, таких как C++, Visual Basic, PHP, Delphi и многие другие. Это позволяет сгенерировать или обратно спроектировать исходный код;

– позволяет визуализировать Ваши приложения;

– позволяет смоделировать базы данных, а также бизнес-процессы;

Рисунок 2.2 – Логотип Sparx Enterprise Architect

2.2 Пользователи системы и их роли

В базе данных существуют следующие типы пользователей:

– системный администратор – пользователи, обеспечивающие функционирование системы, в том числе обслуживание БД (СУБД);

– администратор процесса – пользователи, осуществляющие настройку системы и регулирование прав доступа пользователей к функциональным подсистемам. Основная задача администратора процессов заключается в обеспечении работы остальных пользователей системы путем предоставления им прав доступа и распределения ролей. Администратор процессов также выполняет операции по контролю и исправлению ошибок в работе пользователей, мониторингу системы, заполнению, дополнению и изменению содержимого справочников системы;

– директор – сотрудник, который имеет полный доступ к системе. Также он может назначать права роли сотрудникам;

– доставщик – сотрудники, непосредственно осуществляющие доставку заказов на дом или пункты самовывозов с помощью системы;

– менеджер – сотрудники, которые: управляют заказами, назначают автомобили доставщикам, управляют продуктами, а также автомобилями и скидками, управляют точками самовывоза;

– покупатель – пользователь системы, который может сформировать заказ;

– управляющий складом – сотрудник, который управляет складом;

– управляющий пунктом самовывоза – сотрудник, который управляет пунктом самовывоза.

Основной поиск информации будет осуществляться с помощью запросов в базу данных по определенным критериям товаров. На стороне пользовательского интерфейса ожидается выбор интересуемых типов товара и их критериев. Затем серверная сторона обращается в базу данных, получая список товаров, соответствующих заданным критериям. После чего они должны быть отображены пользователю.

3. МОДЕЛЬ ПРЕДМЕТНОЙ ОБЛАСТИ

3.1 Сущности и связи

Рассмотрим даталогическую модель базы данных. Представим ее основные сущности в текстовом формате в виде таблиц.

Таблица 3.1 – Описание таблицы User

Имя поля	Тип	Описание
User_id	Integer	Идентификатор таблицы.
Uin_id	Bigint	Внешний ключ для дочерних таблиц.
User_registration_time	DATETIME	Дата и время регистрации пользователя.

Таблица 3.2 – Описание таблицы Message

Имя поля	Тип	Описание
Mes_id	Integer	Идентификатор таблицы.
User_id	Integer	Идентификатор пользователя, отправившего сообщение.
Mes_context	TEXT	Текст сообщения.
Mes_time	DATETIME	Дата и врем отправки сообщения.

Таблица 3.3 – Описание таблицы Order

Имя поля	Тип	Описание
Ord_id	Integer	Идентификатор таблицы.
User_id	Integer	Внешний ключ для пользователя, сделавшего заказ.
Ord_price	Integer	Стоимость заказа.
Ord_created_time	DATETIME	Дата и время оформления заказа.
Ost_id	Integer	Внешний ключ для таблицы со статусом заказа.
Rpo_id	Integer	Внешний ключ для таблицы с информацией о месте самовывоза.

Таблица 3.4 – Описание таблицы Warehouse

Имя поля	Тип	Описание
war_id	Integer	Идентификатор таблицы.
Add_id	Integer	Внешний ключ для таблицы адреса.
Cap_id	Integer	Внешний ключ для таблицы с информацией о вместимости.
User_id	Integer	Внешний ключ для таблицы пользователя.
War_name	VARCHAR(150)	Имя склада

Таблица 3.6 – Описание таблицы Product

Имя поля	Тип	Описание
Pro_id	Integer	Идентификатор таблицы.
Pty_id	Integer	Внешний ключ для таблицы типа продукта.
Fac_id	Integer	Внешний ключ для таблицы фабрики.
Col_id	Integer	Внешний ключ для таблицы цвета.
Cty_id	Integer	Внешний ключ для таблицы города производства продукта.
Pro_name	VARCHAR(100)	Имя продукта.
Pro_desription	TEXT	Текстовое описание продукта.
Pro_price	Integer	Цена продукта.
Pro_release_year	Integer	Год выпуска продукта.

3.2 Особенности нормализации

Таблица product может содержать очень много узкоспециализированных полей, вследствие чего было принято решение вынести все эти поля в узкоспециализированные таблицы, уникальность достигается за счет применения связи один к одному. Уникальность многих полей (полей типов) достигается за счет получения их из таблицы product_value, где каждому id соответствует уникальный тип.

Таблица user_info была нормализована таким образом, поскольку пользователь может довольно часто изменять информацию о своих личных данных, а таблица user является одной из ключевых.

Таблица message была нормализована так, поскольку пользователи могут добавлять или создавать группы для общения.

Таблица capacity нормализована данным образом, потому что она используется не только для определения размерности товара, но также и для определения автомобильной вместимости и вместимости складов.

Таблица order нормализована так, поскольку не каждый заказ отправляется на пункт самовывоза.

Специализированные таблицы товаров в большинстве случаев имеют значительное количество полей, и при расширении функционала, их стоит вынести в отдельную таблицу.

4. ОПИСАНИЕ БИЗНЕС-ЛОГИКИ

4.1 Функции базы данных

В базе данных имеются 5 функций:

– получение уникального идентификатора значения продукта;

– получение значения продукта по уникальному идентификатору;

– получение информации о проведенном времени пользователем на сайте;

– создание заказа;

– создание пользователя.

Функция получения уникального идентификатора значения продукта получает на вход значение продукта, а также его тип. Данная функция необходима для обеспечения безопасности и целостности при вводе различных значений (рисунок 4.1).

```
CREATE FUNCTION GET_ID_BY_PRODUCT_VALUE(
    `val` VARCHAR(140),
    `prod_type` INTEGER UNSIGNED
)
RETURNS INTEGER UNSIGNED DETERMINISTIC
    RETURN (
        SELECT `pva_id`
        FROM `product_value` `pv`
        WHERE `val` = `pv`.`pva_value` AND `prod_type` = `pv`.`pty_id`
    );

```

Рисунок 4.1 – Код функции получения уникального идентификатора значения продукта

Функция получения значения продукта по уникальному идентификатору (рисунок 4.2).

```
CREATE FUNCTION GET_PRODUCT_VALUE_BY_ID(`product_val_id` INTEGER UNSIGNED)
RETURNS VARCHAR(140) DETERMINISTIC
    RETURN (
        SELECT `pva_value`
        FROM `product_value`
        WHERE `product_val_id` = `pva_id`
    );
```

Рисунок 4.2 – Код функции получения значения продукта по уникальному идентификатору

Функция получения информации о проведенном времени пользователем на сайте. Принимает на вход идентификатор пользователя (рисунок 4.3).

```
CREATE FUNCTION GET_USER_TOTAL_TIME_ON_SITE(`user_id` INTEGER UNSIGNED)
RETURNS INTEGER UNSIGNED DETERMINISTIC
    RETURN (
        SELECT
            IFNULL(
                SUM(
                    TIME_TO_SEC(`uon_time_out`) -
                    TIME_TO_SEC(`uon_time_in`)
                ), 0
            )
        FROM `user_online`
        WHERE `use_id` = `user_id`
    );

```

Рисунок 4.3 – Код функции получения информации о проведенном времени пользователем на сайте

Функция создания заказа принимает на вход: идентификатор покупателя, идентификатор доставщика, идентификатор статуса, цену, идентификатор адреса, с какого времени осуществлять доставку, по какое время необходимо осуществить доставку и дополнительную информацию. После выполнения функция возвращает идентификатор созданного заказа (рисунок 4.4).

```
CREATE FUNCTION CREATE_ORDER(
    `user` INTEGER UNSIGNED,
    `delivery_user` INTEGER UNSIGNED,
    `status` INTEGER UNSIGNED,
    `price` INTEGER UNSIGNED,
    `address` INTEGER UNSIGNED,
    `time_from` DATETIME,
    `time_to` DATETIME,
    `additional_info` TEXT
)
RETURNS INTEGER UNSIGNED DETERMINISTIC
BEGIN
    INSERT INTO `order`
    VALUES(NULL, `user`, `status`, `price`, CURDATE());

    INSERT INTO `delivery`
    VALUES(NULL, `user`, LAST_INSERT_ID(), `address`, `time_from`, `time_to`, NULL, `additional_info`);

    RETURN LAST_INSERT_ID();
END;
```

Рисунок 4.4 – Код функции создания заказа

Функция создания пользователя принимает на вход: идентификатор типа пользователя, электронную почту, пароль, прозвище, имя, отчество, фа-

милию, телефон. После выполнения функция возвращает идентификатор созданного пользователя (рисунок 4.5).

```
CREATE FUNCTION CREATE_USER(
    `user_type` INTEGER UNSIGNED,
    `email` VARCHAR(100),
    `password` VARCHAR(100),
    `nickname` VARCHAR(100),
    `first_name` VARCHAR(100),
    `middle_name` VARCHAR(100),
    `last_name` VARCHAR(100),
    `phone` VARCHAR(20)
)
RETURNS INTEGER UNSIGNED DETERMINISTIC
BEGIN
    INSERT INTO `user_info`
    VALUES(
        NULL,
        NULL,
        `user_type`,
        `email`,
        `password`,
        `nickname`,
        `first_name`,
        `middle_name`,
        `last_name`,
        `phone`,
        CURDATE()
    );

    SET @user_info_id = LAST_INSERT_ID();
    INSERT INTO `user` VALUES(NULL, user_info_id, CURDATE());
    UPDATE `user_info` SET `use_id` = LAST_INSERT_ID() WHERE `uin_id` = @user_info_id;

    RETURN LAST_INSERT_ID();
END;
```

Рисунок 4.5 – Код функции создания пользователя

4.2 Процедуры базы данных

В базе данных имеются 2 процедуры:

– обновление информации пользователя;

– очистка статистики посещения пользователей.

Процедура обновления информации пользователя. В данную процедуру передаются следующие параметры: идентификатор пользователя, идентификатор типа, электронная почта, пароль, прозвище, имя, отчество, фамилия, телефон (рисунок 4.6).

```sql
CREATE PROCEDURE UPDATE_USER_INFO(
    `user_id` INTEGER UNSIGNED,
    `user_type` INTEGER UNSIGNED,
    `email` VARCHAR(100),
    `password` VARCHAR(100),
    `nickname` VARCHAR(100),
    `first_name` VARCHAR(100),
    `middle_name` VARCHAR(100),
    `last_name` VARCHAR(100),
    `phone` VARCHAR(20)
)
BEGIN
    INSERT INTO `user_info`
    VALUES(
        NULL,
        `user_id`,
        `user_type`,
        `email`,
        `password`,
        `nickname`,
        `first_name`,
        `middle_name`,
        `last_name`,
        `phone`,
        CURDATE()
    );
    UPDATE `user_info` SET `use_id` = `user_id` WHERE `uin_id` = LAST_INSERT_ID();
END;


```

Рисунок 4.6 – Код процедуры обновления информации пользователя

Процедура очистки статистики посещения пользователей (рисунок 4.7).

```sql
CREATE PROCEDURE CLEAR_INFORMATION_ONLINE()
BEGIN
    DELETE FROM `user_online`
    WHERE `uon_out` IS NOT NULL;
END;

```

Рисунок 4.7 – Код процедуры очистки статистики посещения пользователей

4.3 Представления базы данных

В базе данных имеется 19 представлений:
– получить все напольные лампы;
– получить все полки;

– получить все комоды;
– получить все раковины;
– получить все ванны;
– получить все двери;
– получить все шкафы;
– получить все кровати;
– получить все стулья;
– получить все столы;
– получить все кресла;
– получить все диваны;
– получить все зеркала;
– получить все матрасы;
– получить топ 5 продуктов по заказам;
– получить топ 5 лучших доставщиков по количеству заказов;
– получить топ 5 заказов по цене;
– получить информацию о пользователях;
– получить заказы.

Представление всех заказов, которое используется для формирования статистки на сервере (рисунок 4.8).

```
1 CREATE OR REPLACE VIEW GET_ORDERS AS
2     SELECT `ord_price`, `ord_created_time`, `product`.*,
3            `count`, `address`.*, `del_time_from`, `del_time_to`,
4            `del_time_done`, `del_additional_info`
5     FROM `order`
6     JOIN `m2m_order_product` USING(`ord_id`)
7     JOIN `product` USING(`pro_id`)
8     JOIN `delivery` USING(`ord_id`)
9     JOIN `address` USING(`add_id`);
10
```

Рисунок 4.8 – Код представления всех заказов

Представление всей информации пользователей. Используется для поиска пользователей (рисунок 4.9).

```
1 CREATE OR REPLACE VIEW GET_USERS_INFO AS
2     SELECT `use_registration_time`, `user_info`.*
3     FROM `user`
4     JOIN `user_info` USING(`use_id`);
5
```

Рисунок 4.9 – Код представления всей информации пользователей

Представление 5 самых дорогих заказов. Данное представление используется для отображения статистики заказов на странице директора (рисунок 4.10).

```
CREATE OR REPLACE VIEW GET_TOP_5_ORDERS_BY_PRICE AS
    SELECT * FROM `order`
    ORDER BY `ord_price` DESC LIMIT 5;

```

Рисунок 4.10 – Код представления 5 самых дорогих заказа

Представление 5 лучших доставщиков по количеству заказов. Это представление помогает определить лучших доставщиков (рисунок 4.11).

```
CREATE OR REPLACE VIEW GET_TOP_5_DELIVERY_USER_BY_NUM_OF_ORDERS AS
    SELECT `use_id`, COUNT(`use_id`) AS `count`
    FROM `order`
    GROUP BY `use_id`
    ORDER BY COUNT(`count`) DESC LIMIT 5;
```

Рисунок 4.11 – Код представления 5 лучших доставщиков по количеству заказов

Представление 5 лучших продуктов по количеству заказов. С помощью данного представления эти продукты отображаются на главной странице магазина (рисунок 4.12).

```
CREATE OR REPLACE VIEW GET_TOP_5_PRODUCT_BY_ORDER AS
    SELECT `pro_id`, SUM(`count`) AS `count`
    FROM `m2m_order_product`
    GROUP BY `pro_id`
    ORDER BY COUNT(`count`) DESC LIMIT 5;

```

Рисунок 4.12 – Код представления 5 лучших продуктов по количеству заказов

4.4 Триггеры базы данных

В базе данных имеется 3 триггера:

– установка времени регистрации пользователя;

– установка времени создания заказа;

– очистка статистики посещений сайта пользователями.

Триггер установки времени регистрации пользователя. Нужен для точной установки времени создания новой записи в таблице пользователей (рисунок 4.13).

```
CREATE TRIGGER `TRG_udpate_user_registration_time_before_ins`
BEFORE INSERT
ON `user`
    FOR EACH ROW
    BEGIN
        SET NEW.`use_registration_time` = CURDATE();
    END;
```

Рисунок 4.13 – Код триггера установки времени регистрации пользователя

Триггер установки времени создания заказа. Нужен для точной установки времени создания новой записи в таблице заказов (рисунок 4.14).

```
CREATE TRIGGER `TRG_udpate_order_time_created_after_ins`
BEFORE INSERT
ON `order`
    FOR EACH ROW
    BEGIN
        SET NEW.`ord_created_time` = CURDATE();
    END;
```

Рисунок 4.14 – Код триггера установки времени создания заказа

Триггер очистки статистики посещений сайта пользователями. Нужен для очистки от записей в таблице посещений при переполнении (рисунок 4.15).

```
CREATE TRIGGER `TRG_user_online_clear_before_insert`
    AFTER INSERT
    ON `user_online` FOR EACH ROW
BEGIN
    SET @count = (SELECT COUNT(*) FROM `user_online`);
    IF @count > 18446744073709551610 THEN
        CALL CLEAR_INFORMATION_ONLINE();
    END IF;
END;
```

Рисунок 4.15 – Код триггера статистики посещений сайта пользователями

5. ТЕСТИРОВАНИЕ

Завершающим этапом проектирования базы данных является тестирование, которое представляет собой процесс исследования базы данных с целью получения информации о качестве разработанного продукта.

В ходе тестирования были проверены основные функциональные действия, производимые с базой данных в рамках взятой предметной области.

Таблица 5.1 – Тестовый сценарий регистрации пользователя

Тест	Ожидаемый результат	Результат
Регистрация пользователя Добавить кортеж в таблицы user и user_info	Успешное добавление	Успех

Таблица 5.2 – Тестовый сценарий регистрации пользователя с существующим email

Тест	Ожидаемый результат	Результат
Регистрация пользователя с существующим email Добавление кортежа в таблицу user_info с существующим email	Запрет создания	Успех

Таблица 5.3 – Тестовый сценарий добавления склада

Тест	Ожидаемый результат	Результат
Добавление склада Добавление кортежа в таблицу warehouse	Успешное добавление	Успех

Таблица 5.4 – Тестовый сценарий добавления пункта самовывоза

Тест	Ожидаемый результат	Результат
Добавление пункта самовывоза Добавление кортежа в таблицу reception_point	Успешное добавление	Успех

Таблица 5.5 – Тестовый сценарий добавления транспорта

Тест	Ожидаемый результат	Результат
Добавление транспорта Добавление кортежа в таблицу car	Успешное добавление	Успех

Таблица 5.6 – Тестовый сценарий создания пользователя с помощью функции CREATE_USER

Тест	Ожидаемый результат	Результат
Создание пользователя с помощью функции CREATE_USER Добавление кортежа в таблицы, связанные с информацией о пользователях, то есть user и user_info	Успешное добавление	Успех

Таблица 5.7 – Тестовый сценарий обновления информации с помощью функции UPDATE_USER

Тест	Ожидаемый результат	Результат
Обновление информации с помощью функции UPDATE_USER Добавление кортежа в таблицу user_info, а также обновление uin_id в таблице user	Успешное добавление	Успех

Таблица 5.8 – Тестовый сценарий добавления адреса

Тест	Ожидаемый результат	Результат
Добавление адреса Добавление кортежа в таблицу address	Успешное добавление	Успех

Таблица 5.9 – Тестовый сценарий добавления скидки на продукт

Тест	Ожидаемый результат	Результат
Добавление скидки на продукт Добавление кортежа в таблицу discount и m2m_product_discount	Успешное добавление	Успех

Таблица 5.10 – Тестовый сценарий добавления нового предприятия

Тест	Ожидаемый результат	Результат
Добавление предприятия Добавление кортежа в таблицу factory	Успешное добавление	Успех

Таблица 5.11 – Тестовый сценарий удаления пользователя

Тест	Ожидаемый результат	Результат
Удаление пользователя Удаление пользователя кортежа из таблиц, связанных с информацией о пользователе, а также каскадное удаление связанных с ним кортежей	Удаление пользователя и его зависимых кортежей	Успех

Таблица 5.12 – Тестовый сценарий нахождения цены продукта по идентификатору

Тест	Ожидаемый результат	Результат
Нахождение цены продукта по идентификатору Использование функции GET_PRODUCT_ VALUE _BY_ ID	Получение цены	Успех

ЗАКЛЮЧЕНИЕ

Развитие аппаратных и программных технологий привело к большой популярности сети Интернет и позволило ей занять лидирующее положение среди основных инструментов ведения бизнеса, в частности, электронной торговли. Присутствие торговой компании в Интернете необходимо не только с целью получения и наращивания желаемой прибыли, но и для успешной конкурентной борьбы в современных условиях.

Онлайн-магазин – наиболее популярный вид онлайновой торговли. В процессе создания данного веб-ресурса важно грамотно выстроить стратегию ведения бизнеса. В ряд важнейших вопросов, которые предстоит решить торговой компании, входят разработка организационной структуры и ассортиментной политики, выбор способа построения и дальнейшего сопровождения информационной системы интернет-магазина, организация службы доставки, маркетинговая деятельность и, что немаловажно, разработка качественного веб-дизайна предоставляемого ресурса.

В начале разработки были поставлены следующие цели:

– создать качественную базу данных;

– получить опыт написания баз данных;

– изучить различные подходы к проектированию реляционных баз данных;

– закрепить знания различных аспектов СУБД MySQL;

– реализовать возможность последующего совершенствования базы данных в зависимости от требований потребителя;

– получить опыт проектирования бизнес-логики.

В ходе работы решалось множество проблем связанных с реализацией задач проекта, что позволило приобрести ценный опыт. Наиболее важными навыками, полученными во время проектирования, стали:

– закрепление знаний о СУБД MySQL;

– огромный опыт в проектировании баз данных;

– опыт в обработке событий.

.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Куликов С. С. Реляционные базы данных в примерах – Минск, 2021 – 422 с.

[2] Куликов С. С. Работа с MySQL, MSSQL Server и Oracle – Минск, 2021 – 602 с.

[3] Интернет-ресурс с официальной документацией MySQL [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.mysql.com/

[4] Интернет-ресурс для онлайн-обучения базам данных [Электронный ресурс]. – Электронные данные. – Режим доступа: https://site-do.ru/db/db1.php

[5] Интернет-ресурс конструирования SQL-запросов [Электронный ресурс]. – Электронные данные. – Режим доступа: https://web-creator.ru/articles/mysql

[6] Паутов, Алексей. Документация на Mysql. СПб: «Символ», 2007. – 452 с.

[7] Аткинсон, Леон. Mysql для профессионалов. СПб: «Питер», 2010. – 573 с.

[8] Молинаро, Энтони. SQL. Сборник рецептов. СПб: «Символ», 2009. – 769 с.

[9] Официальное руководство по MySQL [Электронный ресурс]. – Электронные данные. – Режим доступа: http://www.mysql.com/.

[10] Форум MySQL [Электронный ресурс]. – Электронные данные. – Режим доступа: http://softtime.ru/forum/read.php?id_forum=3

ПРИЛОЖЕНИЕ А
(обязательное)
Исходный текст программы

```
/* Create Functions */

DELIMITER //
CREATE FUNCTION CREATE_ORDER(
    `user` INTEGER UNSIGNED,
    `delivery_user` INTEGER UNSIGNED,
    `status` INTEGER UNSIGNED,
      `price` INTEGER UNSIGNED,
    `address` INTEGER UNSIGNED,
    `time_from` DATETIME,
    `time_to` DATETIME,
    `additional_info` TEXT
)
RETURNS INTEGER UNSIGNED DETERMINISTIC
BEGIN
      INSERT INTO `order`
    VALUES(NULL, `user`, `status`, `price`, CURDATE());

      INSERT INTO `delivery`
    VALUES(NULL, `user`, LAST_INSERT_ID(), `address`, `time_from`, `time_to`,
NULL, `additional_info`);

    RETURN LAST_INSERT_ID();
END;
//
DELIMITER ;
;
DELIMITER //
CREATE FUNCTION CREATE_USER(
      `user_type` INTEGER UNSIGNED,
    `email` VARCHAR(100),
    `password` VARCHAR(100),
    `nickname` VARCHAR(100),
    `first_name` VARCHAR(100),
    `middle_name` VARCHAR(100),
    `last_name` VARCHAR(100),
    `phone` VARCHAR(20)
)
RETURNS INTEGER UNSIGNED DETERMINISTIC
BEGIN
      INSERT INTO `user_info`
    VALUES(
        NULL,
        NULL,
        `user_type`,
        `email`,
        `password`,
        `nickname`,
        `first_name`,
        `middle_name`,
        `last_name`,
        `phone`,
        CURDATE()
    );
    SET @user_info_id = LAST_INSERT_ID();
    INSERT INTO `user` VALUES(NULL, user_info_id, CURDATE());
```

```
    UPDATE `user_info` SET `use_id` = LAST_INSERT_ID() WHERE `uin_id` =
@user_info_id;
    RETURN LAST_INSERT_ID();
END;
//
DELIMITER ;
;
DELIMITER //
CREATE FUNCTION GET_ID_BY_PRODUCT_VALUE(`val` VARCHAR(140), `prod_type`
INTEGER UNSIGNED)
RETURNS INTEGER UNSIGNED DETERMINISTIC
      RETURN (
            SELECT `pva_id`
            FROM `product_value` `pv`
            WHERE `val` = `pv`.`pva_value` AND `prod_type` = `pv`.`pty_id`
    );
//
DELIMITER ;
;
DELIMITER //
CREATE FUNCTION GET_PRODUCT_VALUE_BY_ID(`product_val_id` INTEGER UNSIGNED)
RETURNS VARCHAR(140) DETERMINISTIC
      RETURN (
            SELECT `pva_value`
            FROM `product_value`
            WHERE `product_val_id` = `pva_id`
    );
//
DELIMITER ;
;
DELIMITER //
CREATE FUNCTION GET_USER_TOTAL_TIME_ON_SITE(`user_id` INTEGER UNSIGNED)
RETURNS INTEGER UNSIGNED DETERMINISTIC
      RETURN (
            SELECT
                  IFNULL(
                        SUM(
                              TIME_TO_SEC(`uon_time_out`) -
                              TIME_TO_SEC(`uon_time_in`)
                        ), 0
                  )
            FROM `user_online`
            WHERE `use_id` = `user_id`
    );
//
DELIMITER ;
;
/* Create Tables */
CREATE TABLE `address`
(
      `add_id` INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
      `add_country` VARCHAR(50) NOT NULL,
      `add_city` VARCHAR(100) NOT NULL,
      `add_region` VARCHAR(150)      NULL,
      `add_street` VARCHAR(150) NOT NULL,
      `add_building` VARCHAR(50)     NULL,
      `add_home` VARCHAR(50) NOT NULL,
      `add_apartment` VARCHAR(50) NOT NULL,
      `add_postcode` VARCHAR(50)     NULL,
      CONSTRAINT `PK_address` PRIMARY KEY (`add_id` ASC)
)
;
```

```
CREATE TABLE `armchair`
(
      `armchair_id` INTEGER UNSIGNED NOT NULL,
      `armchair_weight` INTEGER UNSIGNED NOT NULL,
      `armchair_height` INTEGER UNSIGNED NOT NULL,
      `armchair_depth` INTEGER UNSIGNED NOT NULL,
      `armchair_covering` VARCHAR(50) NOT NULL,
      CONSTRAINT `PK_wiper_blades` PRIMARY KEY (`armchair_id` ASC)
);
CREATE TABLE `bath`
(
      `bath_id` INTEGER UNSIGNED NOT NULL,
      `bath_volume` INTEGER UNSIGNED NOT NULL,
      `bath_color` VARCHAR(50) NOT NULL,
      `bath_height` INTEGER UNSIGNED NOT NULL,
      `bath_width` INTEGER UNSIGNED NOT NULL,
      CONSTRAINT `PK_accumulator` PRIMARY KEY (`bath_id` ASC)
);
CREATE TABLE `bed`
(
      `bed_id` INTEGER UNSIGNED NOT NULL,
      `bed_num_of_places` INTEGER UNSIGNED NOT NULL,
      `bed_length` INTEGER UNSIGNED NOT NULL,
      `bed_width` INTEGER UNSIGNED NOT NULL,
      `bed_material` VARCHAR(50)     NULL,
      CONSTRAINT `PK_filters` PRIMARY KEY (`bed_id` ASC)
);
CREATE TABLE `capacity`
(
      `cap_id` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
      `cap_width` INTEGER UNSIGNED NOT NULL,
      `cap_height` INTEGER UNSIGNED NOT NULL,
      `cap_length` INTEGER UNSIGNED NOT NULL,
      `cap_weight` INTEGER     UNSIGNED NULL,
      CONSTRAINT `PK_capacity` PRIMARY KEY (`cap_id` ASC)
);
CREATE TABLE `car_brand`
(
      `cba_id` INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
      `cba_name` VARCHAR(150) NOT NULL,
      CONSTRAINT `PK_car_brand` PRIMARY KEY (`cba_id` ASC)
);
CREATE TABLE `car_type`
(
      `cty_id` INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
      `cap_id` BIGINT UNSIGNED NOT NULL,
      `fac_id` INTEGER UNSIGNED NOT NULL,
      `col_id` INTEGER UNSIGNED NOT NULL,
      `cba_id` INTEGER UNSIGNED NOT NULL,
      `cty_release_year` INTEGER UNSIGNED NOT NULL,
      `cty_fuel_consumption` INTEGER UNSIGNED NOT NULL,
      `cty_name` VARCHAR(100) NOT NULL,
      `cty_vin_num` CHAR(17) NOT NULL,
      CONSTRAINT `PK_car_type` PRIMARY KEY (`cty_id` ASC)
);
CREATE TABLE `chair`
(
      `chair_id` INTEGER UNSIGNED NOT NULL,
      `chair_height` INTEGER UNSIGNED NOT NULL,
      `chair_upholstery` VARCHAR(50) NOT NULL,
      `chair_material` VARCHAR(50)   NULL,
      CONSTRAINT `PK_oil_seal` PRIMARY KEY (`chair_id` ASC)
```

```
);
CREATE TABLE `color`
(
      `col_id` INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
      `col_name` VARCHAR(60) NOT NULL,
      CONSTRAINT `PK_color` PRIMARY KEY (`col_id` ASC)
);
CREATE TABLE `comments`
(
      `com_id` INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
      `pro_id` INTEGER UNSIGNED NOT NULL,
      `use_id` INTEGER UNSIGNED NOT NULL,
      `com_content` TEXT NOT NULL,
      CONSTRAINT `PK_comments` PRIMARY KEY (`com_id` ASC)
);
CREATE TABLE `cupboard`
(
      `cupboard_id` INTEGER UNSIGNED NOT NULL,
      `cupboard_type` VARCHAR(50) NOT NULL,
      `cupboard_height` INTEGER UNSIGNED NOT NULL,
      `cupboard_depth` INTEGER UNSIGNED NOT NULL,
      `cupboard_num_of_sections` INTEGER UNSIGNED NOT NULL,
      CONSTRAINT `PK_starter` PRIMARY KEY (`cupboard_id` ASC)
);
CREATE TABLE `delivery`
(
      `del_id` INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
      `use_id` INTEGER  UNSIGNED NULL,
      `ord_id` INTEGER UNSIGNED NOT NULL,
      `add_id` INTEGER UNSIGNED NOT NULL,
      `del_time_from` DATETIME NOT NULL,
      `del_time_to` DATETIME NOT NULL,
      `del_time_done` DATETIME      NULL,
      `del_additional_info` TEXT    NULL,
      CONSTRAINT `PK_delivery` PRIMARY KEY (`del_id` ASC)
);
CREATE TABLE `delivery_truck`
(
      `car_id` INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
      `cty_id` INTEGER UNSIGNED NOT NULL,
      `car_number` VARCHAR(30) NOT NULL,
      `use_id` INTEGER  UNSIGNED NULL,
      `war_id` INTEGER UNSIGNED NOT NULL,
      CONSTRAINT `PK_car` PRIMARY KEY (`car_id` ASC)
);
CREATE TABLE `discount`
(
      `dis_id` INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
      `dis_percent` INTEGER UNSIGNED NOT NULL,
      `dis_code` VARCHAR(16) NOT NULL,
      `dis_start_date` DATETIME     NULL,
      `dis_end_date` DATETIME       NULL,
      CONSTRAINT `PK_discount` PRIMARY KEY (`dis_id` ASC)
);
CREATE TABLE `door`
(
      `door_id` INTEGER UNSIGNED NOT NULL,
      `door_color` VARCHAR(50) NOT NULL,
      `door_height` INTEGER UNSIGNED NOT NULL,
      `door_width` INTEGER UNSIGNED NOT NULL,
      `door_lock` VARCHAR(50) NOT NULL,
      CONSTRAINT `PK_pump` PRIMARY KEY (`door_id` ASC)
```

```
);
CREATE TABLE `dresser`
(
      `dresser_id` INTEGER UNSIGNED NOT NULL,
      `dresser_number_of_shelfs` INTEGER UNSIGNED NOT NULL,
      `dresser_height` INTEGER UNSIGNED NOT NULL,
      `dresser_depth` INTEGER UNSIGNED NOT NULL,
      `dresser_width` INTEGER UNSIGNED NOT NULL,
      `dresser_color` VARCHAR(50) NOT NULL,
      `dresser_material` VARCHAR(50) NOT NULL,
      CONSTRAINT `PK_brake_columns` PRIMARY KEY (`dresser_id` ASC)
);
CREATE TABLE `factory`
(
      `fac_id` INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
      `add_id` INTEGER UNSIGNED NOT NULL,
      `man_id` INTEGER UNSIGNED NOT NULL,
      `fac_name` VARCHAR(200) NOT NULL,
      CONSTRAINT `PK_manufacturer` PRIMARY KEY (`fac_id` ASC)
);
CREATE TABLE `floor_lamp`
(
      `lmp_id` INTEGER UNSIGNED NOT NULL,
      `lmp_height` INTEGER UNSIGNED NOT NULL,
      `lmp_num_of_lamps` INTEGER UNSIGNED NOT NULL,
      `lmp_color` VARCHAR(50) NOT NULL,
      `lmp_material` VARCHAR(50) NOT NULL,
      CONSTRAINT `PK_wheel_disc` PRIMARY KEY (`lmp_id` ASC)
);
CREATE TABLE `image_url`
(
      `iur_id` INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
      `iur_url` VARCHAR(2048) NOT NULL,
      `iur_name` VARCHAR(250)        NULL,
      CONSTRAINT `PK_image_url` PRIMARY KEY (`iur_id` ASC)
);
CREATE TABLE `m2m_mesage_user`
(
      `mes_id` INTEGER UNSIGNED NOT NULL,
      `use_id` INTEGER UNSIGNED NOT NULL,
      CONSTRAINT `PK_message_recipient` PRIMARY KEY (`use_id` ASC, `mes_id`
ASC)
);
CREATE TABLE `m2m_news_image_url`
(
      `new_id` INTEGER UNSIGNED NOT NULL,
      `iur_id` INTEGER UNSIGNED NOT NULL,
      CONSTRAINT `PK_news_image` PRIMARY KEY (`new_id` ASC, `iur_id` ASC)
);
CREATE TABLE `m2m_order_product`
(
      `ord_id` INTEGER UNSIGNED NOT NULL,
      `pro_id` INTEGER UNSIGNED NOT NULL,
      `count` INTEGER UNSIGNED NOT NULL,
      CONSTRAINT `PK_m2m_order_product` PRIMARY KEY (`ord_id` ASC, `pro_id`
ASC));
CREATE TABLE `m2m_product_discount`
(
      `pro_id` INTEGER UNSIGNED NOT NULL,
      `dis_id` INTEGER UNSIGNED NOT NULL,
      CONSTRAINT `PK_m2m_product_discount` PRIMARY KEY (`pro_id` ASC,
`dis_id` ASC)
```

```
);
CREATE TABLE `m2m_product_image_url`
(
      `pro_id` INTEGER UNSIGNED NOT NULL,
      `iur_id` INTEGER UNSIGNED NOT NULL,
      CONSTRAINT `PK_m2m_product_image_url` PRIMARY KEY (`pro_id` ASC,
`iur_id` ASC)
);
CREATE TABLE `m2m_user_type_permission`
(
      `uty_id` INTEGER UNSIGNED NOT NULL,
      `per_id` INTEGER UNSIGNED NOT NULL,
      CONSTRAINT `PK_m2m_user_type_permission` PRIMARY KEY (`per_id` ASC,
`uty_id` ASC)
);
CREATE TABLE `m2m_warehouse_product`
(
      `war_id` INTEGER UNSIGNED NOT NULL,
      `pro_id` INTEGER UNSIGNED NOT NULL,
      `count` INTEGER UNSIGNED NOT NULL,
      CONSTRAINT `PK_m2m_warehouse_product` PRIMARY KEY (`war_id` ASC,
`pro_id` ASC)
);
CREATE TABLE `manufacturer`
(
      `man_id` INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
      `man_name` VARCHAR(200) NOT NULL,
      CONSTRAINT `PK_manufacturer` PRIMARY KEY (`man_id` ASC)
);
CREATE TABLE `matress`
(
      `matress_id` INTEGER UNSIGNED NOT NULL,
      `matress_type` VARCHAR(50) NOT NULL,
      `matress_rigidity` VARCHAR(50) NOT NULL,
      `matress_color` VARCHAR(50)    NULL,
      CONSTRAINT `PK_oil` PRIMARY KEY (`matress_id` ASC)
);
CREATE TABLE `message`
(
      `mes_id` INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
      `use_id` INTEGER UNSIGNED NOT NULL,
      `mes_content` TEXT       NULL,
      `mes_time` DATETIME NOT NULL,
      CONSTRAINT `PK_messages` PRIMARY KEY (`mes_id` ASC)
);
CREATE TABLE `mirror`
(
      `mirror_id` INTEGER UNSIGNED NOT NULL,
      `mirror_width` INTEGER UNSIGNED NOT NULL,
      `mirror_height` INTEGER UNSIGNED NOT NULL,
      `mirrior_frame` VARCHAR(50)    NULL,
      CONSTRAINT `PK_steering_rack` PRIMARY KEY (`mirror_id` ASC)
);
CREATE TABLE `news`
(
      `new_id` INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
      `use_id` INTEGER UNSIGNED NOT NULL,
      `new_header` VARCHAR(500) NOT NULL,
      `new_content` TEXT NOT NULL,
      `new_time` DATETIME NOT NULL,
      CONSTRAINT `PK_news` PRIMARY KEY (`new_id` ASC)
);
```

```
CREATE TABLE `order`
(
      `ord_id` INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
      `use_id` INTEGER UNSIGNED NOT NULL,
      `ost_id` INTEGER UNSIGNED NOT NULL,
      `ord_price` INTEGER UNSIGNED NOT NULL,
      `ord_created_time` DATETIME NOT NULL,
      `rpo_id` INTEGER   UNSIGNED NULL,
      CONSTRAINT `PK_order` PRIMARY KEY (`ord_id` ASC)
);
CREATE TABLE `order_status`
(
      `ost_id` INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
      `ost_name` VARCHAR(250) NOT NULL,
      CONSTRAINT `PK_order_status` PRIMARY KEY (`ost_id` ASC)
);
CREATE TABLE `permission`
(
      `per_id` INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
      `per_name` VARCHAR(150) NOT NULL,
      CONSTRAINT `PK_permission` PRIMARY KEY (`per_id` ASC)
);
CREATE TABLE `product`
(
      `pro_id` INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
      `pty_id` INTEGER UNSIGNED NOT NULL,
      `cap_id` BIGINT UNSIGNED NOT NULL,
      `fac_id` INTEGER   UNSIGNED NULL,
      `col_id` INTEGER   UNSIGNED NULL,
      `cty_id` INTEGER   UNSIGNED NULL,
      `pro_name` VARCHAR(100) NOT NULL,
      `pro_description` TEXT   NULL,
      `pro_price` INTEGER NOT NULL,
      `pro_release_year` INTEGER    UNSIGNED NULL,
      CONSTRAINT `PK_product` PRIMARY KEY (`pro_id` ASC)
);
CREATE TABLE `product_type`
(
      `pty_id` INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
      `pty_name` VARCHAR(140) NOT NULL,
      CONSTRAINT `PK_product_type` PRIMARY KEY (`pty_id` ASC)
);
CREATE TABLE `product_value`
(
      `pva_id` INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
      `pva_value` VARCHAR(140) NOT NULL,
      `pty_id` INTEGER UNSIGNED NOT NULL,
      CONSTRAINT `PK_product_value` PRIMARY KEY (`pva_id` ASC)
);
CREATE TABLE `reception_point`
(
      `rpo_id` INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
      `add_id` INTEGER UNSIGNED NOT NULL,
      `use_id` INTEGER   UNSIGNED NULL,
      `cap_id` BIGINT UNSIGNED NOT NULL,
      `rec_name` VARCHAR(150) NOT NULL,
      CONSTRAINT `PK_reception_point` PRIMARY KEY (`rpo_id` ASC)
);
CREATE TABLE `shelf`
(
      `shelf_id` INTEGER UNSIGNED NOT NULL,
      `shelf_width` INTEGER UNSIGNED NOT NULL,
```

```
      `shelf_depth` INTEGER UNSIGNED NOT NULL,
      `shelf_color` VARCHAR(50) NOT NULL,
      `shelf_material` VARCHAR(50) NOT NULL,
      CONSTRAINT `PK_timing_belt` PRIMARY KEY (`shelf_id` ASC)
);
CREATE TABLE `sink`
(
      `sink_id` INTEGER UNSIGNED NOT NULL,
      `sink_volume` INT        NULL,
      `sink_color` VARCHAR(50)       NULL,
      `sink_depth` INTEGER     UNSIGNED NULL,
      `sink_width` INTEGER UNSIGNED NOT NULL,
      `sink_height` INTEGER UNSIGNED NOT NULL,
      CONSTRAINT `PK_brake_disc` PRIMARY KEY (`sink_id` ASC)
);
CREATE TABLE `sofa`
(
      `sofa_id` INTEGER UNSIGNED NOT NULL,
      `sofa_number_of_seats` INTEGER UNSIGNED NOT NULL,
      `sofa_length` INT NOT NULL,
      `sofa_height` INT NOT NULL,
      `sofa_depth` INTEGER UNSIGNED NOT NULL,
      `sofa_covering` VARCHAR(50) NOT NULL,
      `sro_tip` INTEGER UNSIGNED NOT NULL,
      CONSTRAINT `PK_steering_rods` PRIMARY KEY (`sofa_id` ASC)
);
CREATE TABLE `tables`
(
      `table_id` INTEGER UNSIGNED NOT NULL,
      `table_height` INTEGER UNSIGNED NOT NULL,
      `table_num_of_seats` INTEGER UNSIGNED NOT NULL,
      `table_material` VARCHAR(50) NOT NULL,
      CONSTRAINT `PK_candle` PRIMARY KEY (`table_id` ASC)
);
CREATE TABLE `user`
(
      `use_id` INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
      `uin_id` BIGINT UNSIGNED NOT NULL,
      `use_registration_time` DATETIME NOT NULL,
      CONSTRAINT `PK_user` PRIMARY KEY (`use_id` ASC)
);
CREATE TABLE `user_info`
(
      `uin_id` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
      `use_id` INTEGER   UNSIGNED NULL,
      `uty_id` INTEGER UNSIGNED NOT NULL,
      `uin_email` VARCHAR(100) NOT NULL,
      `uin_password` VARCHAR(100) NOT NULL,
      `uin_nickname` VARCHAR(100)    NULL,
      `uin_first_name` VARCHAR(100)       NULL,
      `uin_middle_name` VARCHAR(100)       NULL,
      `uin_last_name` VARCHAR(100)   NULL,
      `uin_phone` VARCHAR(20)        NULL,
      `uin_time_update` DATETIME NOT NULL,
      CONSTRAINT `PK_user_info` PRIMARY KEY (`uin_id` ASC)
);
CREATE TABLE `user_online`
(
      `uon_id` BIGINT UNSIGNED NOT NULL AUTO_INCREMENT,
      `use_id` INTEGER UNSIGNED NOT NULL,
      `uon_time_in` DATETIME NOT NULL,
      `uon_time_out` DATETIME NOT NULL,
```

```
      CONSTRAINT `PK_user_online` PRIMARY KEY (`uon_id` ASC)
);
CREATE TABLE `user_type`
(
      `uty_id` INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
      `uty_name` VARCHAR(150) NOT NULL,
      CONSTRAINT `PK_user_type` PRIMARY KEY (`uty_id` ASC)
);
CREATE TABLE `warehouse`
(
      `war_id` INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
      `add_id` INTEGER UNSIGNED NOT NULL,
      `cap_id` BIGINT UNSIGNED NOT NULL,
      `use_id` INTEGER  UNSIGNED NULL,
      `war_name` VARCHAR(150) NOT NULL,
      CONSTRAINT `PK_warehouse` PRIMARY KEY (`war_id` ASC)
);
CREATE TABLE `wishlist`
(
      `pro_id` INTEGER UNSIGNED NOT NULL,
      `use_id` INTEGER UNSIGNED NOT NULL,
      `count` INTEGER UNSIGNED NOT NULL,
      CONSTRAINT `PK_wishlist` PRIMARY KEY (`pro_id` ASC, `use_id` ASC)
);
/* Create Primary Keys, Indexes, Uniques, Checks */
ALTER TABLE `armchair`
 ADD INDEX `IXFK_wiper_blades_product` (`armchair_id` ASC)
;
ALTER TABLE `bath`
 ADD INDEX `IXFK_accumulator_product` (`bath_id` ASC)
;
ALTER TABLE `bed`
 ADD INDEX `IXFK_filters_product` (`bed_id` ASC)
;
ALTER TABLE `car_brand`
 ADD CONSTRAINT `UNX_cba_name` UNIQUE (`cba_name` ASC)
;
ALTER TABLE `car_brand`
 ADD INDEX `IX_cba_name` (`cba_name` ASC)
;

ALTER TABLE `car_type`
 ADD INDEX `IX_cty_name` (`cty_name` ASC)
;
ALTER TABLE `chair`
 ADD INDEX `IXFK_oil_seal_product` (`chair_id` ASC)
;
ALTER TABLE `color`
 ADD CONSTRAINT `UNX_col_name` UNIQUE (`col_name` ASC)
;
ALTER TABLE `color`
 ADD INDEX `IX_col_name` (`col_name` ASC)
;
ALTER TABLE `cupboard`
 ADD INDEX `IXFK_starter_product` (`cupboard_id` ASC)
;
ALTER TABLE `door`
 ADD INDEX `IXFK_pump_product` (`door_id` ASC)
;
ALTER TABLE `dresser`
 ADD INDEX `IXFK_brake_columns_product` (`dresser_id` ASC)
;
```

```
ALTER TABLE `floor_lamp`
 ADD INDEX `IXFK_wheel_disc_product` (`lmp_id` ASC)
;
ALTER TABLE `matress`
 ADD INDEX `IXFK_oil_product` (`matress_id` ASC)
;
ALTER TABLE `mirror`
 ADD INDEX `IXFK_steering_rack_product` (`mirror_id` ASC)
;
ALTER TABLE `news`
 ADD INDEX `IX_new_header` (`new_header` ASC)
;
DELIMITER //
CREATE TRIGGER `TRG_udpate_order_time_created_after_ins`
BEFORE INSERT
ON `order`
      FOR EACH ROW
      BEGIN
            SET NEW.`ord_created_time` = CURDATE();
      END;
//
DELIMITER ;
;
ALTER TABLE `order_status`
 ADD CONSTRAINT `UNX_ost_name` UNIQUE (`ost_name` ASC)
;
ALTER TABLE `order_status`
 ADD INDEX `IX_ost_name` (`ost_name` ASC)
;
ALTER TABLE `permission`
 ADD CONSTRAINT `UNX_per_name` UNIQUE (`per_name` ASC)
;
ALTER TABLE `product_type`
 ADD CONSTRAINT `UNX_pty_name` UNIQUE (`pty_name` ASC)
;

ALTER TABLE `product_type`
 ADD INDEX `IX_pty_name` (`pty_name` ASC)
;

ALTER TABLE `product_value`
 ADD CONSTRAINT `UNX_pva_value` UNIQUE (`pva_value` ASC)
;
ALTER TABLE `shelf`
 ADD INDEX `IXFK_timing_belt_product` (`shelf_id` ASC)
;
ALTER TABLE `sink`
 ADD INDEX `IXFK_brake_disc_product` (`sink_id` ASC)
;
ALTER TABLE `sofa`
 ADD INDEX `IXFK_steering_rods_product` (`sofa_id` ASC)
;
ALTER TABLE `tables`
 ADD INDEX `IXFK_candle_product` (`table_id` ASC)
;
DELIMITER //
CREATE TRIGGER `TRG_udpate_user_registration_time_before_ins`
BEFORE INSERT
ON `user`
      FOR EACH ROW
      BEGIN
            SET NEW.`use_registration_time` = CURDATE();
```

```
      END;
//
DELIMITER ;
;
ALTER TABLE `user_info`
 ADD CONSTRAINT `UNX_uin_email` UNIQUE (`uin_email` ASC)
;
ALTER TABLE `user_info`
 ADD CONSTRAINT `UNX_uin_nickname` UNIQUE (`uin_nickname` ASC)
;
ALTER TABLE `user_info`
 ADD CONSTRAINT `UNX_uin_phone` UNIQUE (`uin_phone` ASC)
;
DELIMITER //
CREATE TRIGGER `TRG_user_online_clear_before_insert`
    AFTER INSERT
    ON `user_online` FOR EACH ROW
BEGIN
    SET @count = (SELECT COUNT(*) FROM `user_online`);
      IF @count > 18446744073709551610 THEN
            CALL CLEAR_INFORMATION_ONLINE();
      END IF;
END;
//
DELIMITER ;
;
ALTER TABLE `user_type`
 ADD CONSTRAINT `UNX_uty_name` UNIQUE (`uty_name` ASC)
;
ALTER TABLE `warehouse`
 ADD INDEX `IX_war_name` (`war_name` ASC)
;
/* Create Foreign Key Constraints */
ALTER TABLE `armchair`
 ADD CONSTRAINT `FK_wiper_blades_product`
      FOREIGN KEY (`armchair_id`) REFERENCES `product` (`pro_id`) ON DELETE
Cascade ON UPDATE Cascade
;
ALTER TABLE `bath`
 ADD CONSTRAINT `FK_accumulator_product`
      FOREIGN KEY (`bath_id`) REFERENCES `product` (`pro_id`) ON DELETE
Cascade ON UPDATE Cascade
;
ALTER TABLE `bed`
 ADD CONSTRAINT `FK_filters_product`
      FOREIGN KEY (`bed_id`) REFERENCES `product` (`pro_id`) ON DELETE
Cascade ON UPDATE Cascade
;
ALTER TABLE `car_type`
 ADD CONSTRAINT `FK_car_type_capacity`
      FOREIGN KEY (`cap_id`) REFERENCES `capacity` (`cap_id`) ON DELETE
Restrict ON UPDATE Cascade
;
ALTER TABLE `car_type`
 ADD CONSTRAINT `FK_car_type_car_brand`
      FOREIGN KEY (`cba_id`) REFERENCES `car_brand` (`cba_id`) ON DELETE
Restrict ON UPDATE Cascade
;
ALTER TABLE `car_type`
 ADD CONSTRAINT `FK_car_type_color`
      FOREIGN KEY (`col_id`) REFERENCES `color` (`col_id`) ON DELETE Restrict
ON UPDATE Cascade
```

```
;
ALTER TABLE `car_type`
 ADD CONSTRAINT `FK_car_type_factory`
      FOREIGN KEY (`fac_id`) REFERENCES `factory` (`fac_id`) ON DELETE
Restrict ON UPDATE Cascade
;
ALTER TABLE `chair`
 ADD CONSTRAINT `FK_oil_seal_product`
      FOREIGN KEY (`chair_id`) REFERENCES `product` (`pro_id`) ON DELETE
Cascade ON UPDATE Cascade
;
ALTER TABLE `comments`
 ADD CONSTRAINT `FK_comments_product`
      FOREIGN KEY (`pro_id`) REFERENCES `product` (`pro_id`) ON DELETE No
Action ON UPDATE No Action
;
ALTER TABLE `comments`
 ADD CONSTRAINT `FK_comments_user`
      FOREIGN KEY (`use_id`) REFERENCES `user` (`use_id`) ON DELETE No Action
ON UPDATE Cascade
;
ALTER TABLE `cupboard`
 ADD CONSTRAINT `FK_starter_product`
      FOREIGN KEY (`cupboard_id`) REFERENCES `product` (`pro_id`) ON DELETE
Cascade ON UPDATE Cascade
;

ALTER TABLE `delivery`
 ADD CONSTRAINT `FK_delivery_address`
      FOREIGN KEY (`add_id`) REFERENCES `address` (`add_id`) ON DELETE
Restrict ON UPDATE Cascade
;
ALTER TABLE `delivery`
 ADD CONSTRAINT `FK_delivery_order`
      FOREIGN KEY (`ord_id`) REFERENCES `order` (`ord_id`) ON DELETE Cascade
ON UPDATE Cascade
;
ALTER TABLE `delivery`
 ADD CONSTRAINT `FK_delivery_user`
      FOREIGN KEY (`use_id`) REFERENCES `user` (`use_id`) ON DELETE Restrict
ON UPDATE Restrict
;
ALTER TABLE `delivery_truck`
 ADD CONSTRAINT `FK_car_car_type`
      FOREIGN KEY (`cty_id`) REFERENCES `car_type` (`cty_id`) ON DELETE
Restrict ON UPDATE Cascade
;

ALTER TABLE `delivery_truck`
 ADD CONSTRAINT `FK_car_user`
      FOREIGN KEY (`use_id`) REFERENCES `user` (`use_id`) ON DELETE Set Null
ON UPDATE Cascade
;
ALTER TABLE `delivery_truck`
 ADD CONSTRAINT `FK_car_warehouse`
      FOREIGN KEY (`war_id`) REFERENCES `warehouse` (`war_id`) ON DELETE
Restrict ON UPDATE Restrict
;
ALTER TABLE `door`
 ADD CONSTRAINT `FK_pump_product`
      FOREIGN KEY (`door_id`) REFERENCES `product` (`pro_id`) ON DELETE
Cascade ON UPDATE Cascade
```

```
;
ALTER TABLE `dresser`
 ADD CONSTRAINT `FK_brake_columns_product`
      FOREIGN KEY (`dresser_id`) REFERENCES `product` (`pro_id`) ON DELETE
Cascade ON UPDATE Cascade
;
ALTER TABLE `factory`
 ADD CONSTRAINT `FK_factory_manufacturer`
      FOREIGN KEY (`man_id`) REFERENCES `manufacturer` (`man_id`) ON DELETE
Restrict ON UPDATE Cascade
;
ALTER TABLE `factory`
 ADD CONSTRAINT `FK_manufacturer_address`
      FOREIGN KEY (`add_id`) REFERENCES `address` (`add_id`) ON DELETE
Restrict ON UPDATE Cascade
;
ALTER TABLE `floor_lamp`
 ADD CONSTRAINT `FK_wheel_disc_product`
      FOREIGN KEY (`lmp_id`) REFERENCES `product` (`pro_id`) ON DELETE
Cascade ON UPDATE Cascade
;
ALTER TABLE `m2m_mesage_user`
 ADD CONSTRAINT `FK_message_recipient_message`
      FOREIGN KEY (`mes_id`) REFERENCES `message` (`mes_id`) ON DELETE
Cascade ON UPDATE Cascade
;
ALTER TABLE `m2m_mesage_user`
 ADD CONSTRAINT `FK_message_recipient_user`
      FOREIGN KEY (`use_id`) REFERENCES `user` (`use_id`) ON DELETE Cascade
ON UPDATE Cascade
;
ALTER TABLE `m2m_news_image_url`
 ADD CONSTRAINT `FK_m2m_news_image_url_image_url`
      FOREIGN KEY (`iur_id`) REFERENCES `image_url` (`iur_id`) ON DELETE
Cascade ON UPDATE Cascade
;
ALTER TABLE `m2m_news_image_url`
 ADD CONSTRAINT `FK_m2m_news_image_url_news`
      FOREIGN KEY (`new_id`) REFERENCES `news` (`new_id`) ON DELETE Cascade
ON UPDATE Cascade
;

ALTER TABLE `m2m_order_product`
 ADD CONSTRAINT `FK_m2m_order_product_order`
      FOREIGN KEY (`ord_id`) REFERENCES `order` (`ord_id`) ON DELETE Cascade
ON UPDATE Cascade
;
ALTER TABLE `m2m_order_product`
 ADD CONSTRAINT `FK_m2m_order_product_product`
      FOREIGN KEY (`pro_id`) REFERENCES `product` (`pro_id`) ON DELETE
Restrict ON UPDATE Cascade
;
ALTER TABLE `m2m_product_discount`
 ADD CONSTRAINT `FK_m2m_product_discount_discount`
      FOREIGN KEY (`dis_id`) REFERENCES `discount` (`dis_id`) ON DELETE
Cascade ON UPDATE Cascade
;
ALTER TABLE `m2m_product_discount`
 ADD CONSTRAINT `FK_m2m_product_discount_product`
      FOREIGN KEY (`pro_id`) REFERENCES `product` (`pro_id`) ON DELETE
Cascade ON UPDATE Cascade
;
```

```
ALTER TABLE `m2m_product_image_url`
 ADD CONSTRAINT `FK_m2m_product_image_url_image_url`
      FOREIGN KEY (`iur_id`) REFERENCES `image_url` (`iur_id`) ON DELETE
Cascade ON UPDATE Cascade
;
ALTER TABLE `m2m_product_image_url`
 ADD CONSTRAINT `FK_m2m_product_image_url_product`
      FOREIGN KEY (`pro_id`) REFERENCES `product` (`pro_id`) ON DELETE
Cascade ON UPDATE Cascade
;
ALTER TABLE `m2m_user_type_permission`
 ADD CONSTRAINT `FK_m2m_user_type_permission_permission`
      FOREIGN KEY (`per_id`) REFERENCES `permission` (`per_id`) ON DELETE
Cascade ON UPDATE Cascade
;
ALTER TABLE `m2m_user_type_permission`
 ADD CONSTRAINT `FK_m2m_user_type_permission_user_type`
      FOREIGN KEY (`uty_id`) REFERENCES `user_type` (`uty_id`) ON DELETE
Cascade ON UPDATE Cascade
;
ALTER TABLE `m2m_warehouse_product`
 ADD CONSTRAINT `FK_m2m_warehouse_product_product`
      FOREIGN KEY (`pro_id`) REFERENCES `product` (`pro_id`) ON DELETE
Restrict ON UPDATE Cascade
;
ALTER TABLE `m2m_warehouse_product`
 ADD CONSTRAINT `FK_m2m_warehouse_product_warehouse`
      FOREIGN KEY (`war_id`) REFERENCES `warehouse` (`war_id`) ON DELETE
Restrict ON UPDATE Cascade
;
ALTER TABLE `matress`
 ADD CONSTRAINT `FK_oil_product`
      FOREIGN KEY (`matress_id`) REFERENCES `product` (`pro_id`) ON DELETE
Cascade ON UPDATE Cascade
;
ALTER TABLE `message`
 ADD CONSTRAINT `FK_message_user`
      FOREIGN KEY (`use_id`) REFERENCES `user` (`use_id`) ON DELETE No Action
ON UPDATE No Action
;
ALTER TABLE `mirror`
 ADD CONSTRAINT `FK_steering_rack_product`
      FOREIGN KEY (`mirror_id`) REFERENCES `product` (`pro_id`) ON DELETE
Cascade ON UPDATE Cascade
;
ALTER TABLE `news`
 ADD CONSTRAINT `FK_news_user`
      FOREIGN KEY (`use_id`) REFERENCES `user` (`use_id`) ON DELETE No Action
ON UPDATE Cascade
;
ALTER TABLE `order`
 ADD CONSTRAINT `FK_order_order_status`
      FOREIGN KEY (`ost_id`) REFERENCES `order_status` (`ost_id`) ON DELETE
Restrict ON UPDATE Restrict
;
ALTER TABLE `order`
 ADD CONSTRAINT `FK_order_reception_point`
      FOREIGN KEY (`rpo_id`) REFERENCES `reception_point` (`rpo_id`) ON
DELETE Restrict ON UPDATE Restrict
;
ALTER TABLE `order`
 ADD CONSTRAINT `FK_order_user`
```

```
      FOREIGN KEY (`use_id`) REFERENCES `user` (`use_id`) ON DELETE Cascade
ON UPDATE Cascade
;
ALTER TABLE `product`
 ADD CONSTRAINT `FK_product_capacity`
      FOREIGN KEY (`cap_id`) REFERENCES `capacity` (`cap_id`) ON DELETE
Restrict ON UPDATE Cascade
;
ALTER TABLE `product`
 ADD CONSTRAINT `FK_product_car_type`
      FOREIGN KEY (`cty_id`) REFERENCES `car_type` (`cty_id`) ON DELETE
Restrict ON UPDATE Restrict
;
ALTER TABLE `product`
 ADD CONSTRAINT `FK_product_color`
      FOREIGN KEY (`col_id`) REFERENCES `color` (`col_id`) ON DELETE Restrict
ON UPDATE Cascade
;
ALTER TABLE `product`
 ADD CONSTRAINT `FK_product_factory`
      FOREIGN KEY (`fac_id`) REFERENCES `factory` (`fac_id`) ON DELETE No
Action ON UPDATE No Action
;
ALTER TABLE `product`
 ADD CONSTRAINT `FK_product_product_type`
      FOREIGN KEY (`pty_id`) REFERENCES `product_type` (`pty_id`) ON DELETE
Restrict ON UPDATE Restrict
;
ALTER TABLE `product_value`
 ADD CONSTRAINT `FK_product_value_product_type`
      FOREIGN KEY (`pty_id`) REFERENCES `product_type` (`pty_id`) ON DELETE
Restrict ON UPDATE Restrict
;
ALTER TABLE `reception_point`
 ADD CONSTRAINT `FK_reception_point_address`
      FOREIGN KEY (`add_id`) REFERENCES `address` (`add_id`) ON DELETE
Restrict ON UPDATE Cascade
;
ALTER TABLE `reception_point`
 ADD CONSTRAINT `FK_reception_point_capacity`
      FOREIGN KEY (`cap_id`) REFERENCES `capacity` (`cap_id`) ON DELETE
Restrict ON UPDATE Restrict
;
ALTER TABLE `reception_point`
 ADD CONSTRAINT `FK_reception_point_user`
      FOREIGN KEY (`use_id`) REFERENCES `user` (`use_id`) ON DELETE Restrict
ON UPDATE Restrict
;
ALTER TABLE `shelf`
 ADD CONSTRAINT `FK_timing_belt_product`
      FOREIGN KEY (`shelf_id`) REFERENCES `product` (`pro_id`) ON DELETE
Cascade ON UPDATE Cascade
;
ALTER TABLE `sink`
 ADD CONSTRAINT `FK_brake_disc_product`
      FOREIGN KEY (`sink_id`) REFERENCES `product` (`pro_id`) ON DELETE
Cascade ON UPDATE Cascade
;
ALTER TABLE `sofa`
 ADD CONSTRAINT `FK_steering_rods_product`
      FOREIGN KEY (`sofa_id`) REFERENCES `product` (`pro_id`) ON DELETE
Cascade ON UPDATE Cascade
```

```
;
ALTER TABLE `tables`
 ADD CONSTRAINT `FK_candle_product`
      FOREIGN KEY (`table_id`) REFERENCES `product` (`pro_id`) ON DELETE
Cascade ON UPDATE Cascade
;
ALTER TABLE `user`
 ADD CONSTRAINT `FK_user_user_info`
      FOREIGN KEY (`uin_id`) REFERENCES `user_info` (`uin_id`) ON DELETE
Restrict ON UPDATE Cascade
;
ALTER TABLE `user_info`
 ADD CONSTRAINT `FK_user_info_user`
      FOREIGN KEY (`use_id`) REFERENCES `user` (`use_id`) ON DELETE Cascade
ON UPDATE Cascade
;
ALTER TABLE `user_info`
 ADD CONSTRAINT `FK_user_info_user_type`
      FOREIGN KEY (`uty_id`) REFERENCES `user_type` (`uty_id`) ON DELETE
Restrict ON UPDATE No Action
;
ALTER TABLE `user_online`
 ADD CONSTRAINT `FK_user_online_user`
      FOREIGN KEY (`use_id`) REFERENCES `user` (`use_id`) ON DELETE Cascade
ON UPDATE Cascade
;
ALTER TABLE `warehouse`
 ADD CONSTRAINT `FK_warehouse_address`
      FOREIGN KEY (`add_id`) REFERENCES `address` (`add_id`) ON DELETE No
Action ON UPDATE No Action
;
ALTER TABLE `warehouse`
 ADD CONSTRAINT `FK_warehouse_capacity`
      FOREIGN KEY (`cap_id`) REFERENCES `capacity` (`cap_id`) ON DELETE No
Action ON UPDATE No Action
;
ALTER TABLE `warehouse`
 ADD CONSTRAINT `FK_warehouse_user`
      FOREIGN KEY (`use_id`) REFERENCES `user` (`use_id`) ON DELETE Restrict
ON UPDATE Restrict
;
ALTER TABLE `wishlist`
 ADD CONSTRAINT `FK_wishlist_product`
      FOREIGN KEY (`pro_id`) REFERENCES `product` (`pro_id`) ON DELETE
Cascade ON UPDATE Cascade
;
ALTER TABLE `wishlist`
 ADD CONSTRAINT `FK_wishlist_user`
      FOREIGN KEY (`use_id`) REFERENCES `user` (`use_id`) ON DELETE Cascade
ON UPDATE Cascade
;
SET FOREIGN_KEY_CHECKS=1
;
/* Create Views */
CREATE OR REPLACE VIEW GET_ARMCHAIRS AS
    SELECT `product`.*, `col_name`, `wbl_size`,
                `cty_release_year`, `cty_fuel_consumption`, `cty_name`,
`cty_vin_num`,
                GET_PRODUCT_VALUE_BY_ID(`wbl_type`) AS `wbl_type`,
            `wbl_mounting_type`,
                GET_PRODUCT_VALUE_BY_ID(`wbl_material`) AS `wbl_material`,
            `fac_name`, `man_name`, `add_country`, `add_city`, `add_region`,
```

```
            `add_street`, `add_building`, `add_home`, `add_apartment`,
`add_postcode`,
            `cap_width`, `cap_height`, `cap_length`, `cap_weight`
    FROM `product`
    JOIN `armchair` ON `armchair_id` = `pro_id`
    JOIN `color` USING(`col_id`)
    JOIN `factory` USING(`fac_id`)
    JOIN `manufacturer` USING(`man_id`)
    JOIN `address` USING(`add_id`)
    JOIN `capacity` USING(`cap_id`)
    JOIN `car_type` USING(`cty_id`);
;
CREATE OR REPLACE VIEW GET_BATHS AS
    SELECT `product`.*, `col_name`,
                `cty_release_year`, `cty_fuel_consumption`, `cty_name`,
`cty_vin_num`,
                GET_PRODUCT_VALUE_BY_ID(`acc_type`) AS `acc_type`,
            `acc_capacity`, `acc_voltage`, `acc_efficiency`,
            `fac_name`, `man_name`, `add_country`, `add_city`, `add_region`,
            `add_street`, `add_building`, `add_home`, `add_apartment`,
`add_postcode`,
            `cap_width`, `cap_height`, `cap_length`, `cap_weight`
    FROM `product`
    JOIN `bath` ON `bath_id` = `pro_id`
    JOIN `color` USING(`col_id`)
    JOIN `factory` USING(`fac_id`)
    JOIN `manufacturer` USING(`man_id`)
    JOIN `address` USING(`add_id`)
    JOIN `capacity` USING(`cap_id`)
    JOIN `car_type` USING(`cty_id`);
;
CREATE OR REPLACE VIEW GET_BEDS AS
    SELECT `product`.*, `col_name`,
                `cty_release_year`, `cty_fuel_consumption`, `cty_name`,
`cty_vin_num`,
                GET_PRODUCT_VALUE_BY_ID(`fil_type`) AS `fil_type`,
            `fil_efficiency`,
                GET_PRODUCT_VALUE_BY_ID(`fil_material`) AS `fil_material`,
            `fac_name`, `man_name`, `add_country`, `add_city`, `add_region`,
            `add_street`, `add_building`, `add_home`, `add_apartment`,
`add_postcode`,
            `cap_width`, `cap_height`, `cap_length`, `cap_weight`
    FROM `product`
    JOIN `bed` ON `bed_id` = `pro_id`
    JOIN `color` USING(`col_id`)
    JOIN `factory` USING(`fac_id`)
    JOIN `manufacturer` USING(`man_id`)
    JOIN `address` USING(`add_id`)
    JOIN `capacity` USING(`cap_id`)
    JOIN `car_type` USING(`cty_id`);
;
CREATE OR REPLACE VIEW GET_CHAIRS AS
    SELECT `product`.*, `col_name`, `chair`.*,
                `cty_release_year`, `cty_fuel_consumption`, `cty_name`,
`cty_vin_num`,
            `fac_name`, `man_name`, `add_country`, `add_city`, `add_region`,
            `add_street`, `add_building`, `add_home`, `add_apartment`,
`add_postcode`,
            `cap_width`, `cap_height`, `cap_length`, `cap_weight`
    FROM `product`
    JOIN `chair` ON `chair_id` = `pro_id`
    JOIN `color` USING(`col_id`)
```

```
    JOIN `factory` USING(`fac_id`)
    JOIN `manufacturer` USING(`man_id`)
    JOIN `address` USING(`add_id`)
    JOIN `capacity` USING(`cap_id`)
    JOIN `car_type` USING(`cty_id`);
;
CREATE OR REPLACE VIEW GET_CUPBOARDS AS
    SELECT `product`.*, `col_name`,
                `cty_release_year`, `cty_fuel_consumption`, `cty_name`,
`cty_vin_num`,
                GET_PRODUCT_VALUE_BY_ID(`sta_type`) AS `sta_type`,
            `sta_voltage`, `sta_power`, `sta_turnovers`,
            `sta_current_strength`, `sta_tension_force`,
            `fac_name`, `man_name`, `add_country`, `add_city`, `add_region`,
            `add_street`, `add_building`, `add_home`, `add_apartment`,
`add_postcode`,
            `cap_width`, `cap_height`, `cap_length`, `cap_weight`
    FROM `product`
    JOIN `cupboard` ON `cupboard_id` = `pro_id`
    JOIN `color` USING(`col_id`)
    JOIN `factory` USING(`fac_id`)
    JOIN `manufacturer` USING(`man_id`)
    JOIN `address` USING(`add_id`)
    JOIN `capacity` USING(`cap_id`)
    JOIN `car_type` USING(`cty_id`);
;
CREATE OR REPLACE VIEW GET_DOORS AS
    SELECT `product`.*, `col_name`, `door`.*,
                `cty_release_year`, `cty_fuel_consumption`, `cty_name`,
`cty_vin_num`,
            `fac_name`, `man_name`, `add_country`, `add_city`, `add_region`,
            `add_street`, `add_building`, `add_home`, `add_apartment`,
`add_postcode`,
            `cap_width`, `cap_height`, `cap_length`, `cap_weight`
    FROM `product`
    JOIN `door` ON `door_id` = `pro_id`
    JOIN `color` USING(`col_id`)
    JOIN `factory` USING(`fac_id`)
    JOIN `manufacturer` USING(`man_id`)
    JOIN `address` USING(`add_id`)
    JOIN `capacity` USING(`cap_id`)
    JOIN `car_type` USING(`cty_id`);
;
CREATE OR REPLACE VIEW GET_DRESSERS AS
    SELECT `product`.*, `col_name`,
                `cty_release_year`, `cty_fuel_consumption`, `cty_name`,
`cty_vin_num`,
                    GET_PRODUCT_VALUE_BY_ID(`bco_frame_material`) AS
`bco_frame_material`,
                    GET_PRODUCT_VALUE_BY_ID(`bco_internal_material`) AS
`bco_internal_material`,
             `bco_column_width`, `bco_column_length`,
`bco_friction_coefficient`, `bco_max_load`,
            `fac_name`, `man_name`, `add_country`, `add_city`, `add_region`,
            `add_street`, `add_building`, `add_home`, `add_apartment`,
            `add_postcode`, `cap_width`, `cap_height`, `cap_length`,
`cap_weight`
    FROM `product`
    JOIN `shelf` ON `shelf_id` = `pro_id`
    JOIN `color` USING(`col_id`)
    JOIN `factory` USING(`fac_id`)
    JOIN `manufacturer` USING(`man_id`)
```

```
    JOIN `address` USING(`add_id`)
    JOIN `capacity` USING(`cap_id`)
    JOIN `car_type` USING(`cty_id`);
;
CREATE OR REPLACE VIEW GET_FLOOR_LAMPS AS
    SELECT `product`.*, `floor_lamp`.*, `col_name`,
                `cty_release_year`, `cty_fuel_consumption`, `cty_name`,
`cty_vin_num`,
                `fac_name`, `man_name`, `add_country`, `add_city`,
`add_region`,
            `add_street`, `add_building`, `add_home`, `add_apartment`,
            `add_postcode`, `cap_width`, `cap_height`, `cap_length`,
`cap_weight`
    FROM `product`
    JOIN `floor_lamp` ON `lmp_id` = `pro_id`
    JOIN `color` USING(`col_id`)
    JOIN `factory` USING(`fac_id`)
    JOIN `manufacturer` USING(`man_id`)
    JOIN `address` USING(`add_id`)
    JOIN `capacity` USING(`cap_id`)
    JOIN `car_type` USING(`cty_id`);
;
CREATE OR REPLACE VIEW GET_MATRESSES AS
    SELECT `product`.*, `col_name`,
                `cty_release_year`, `cty_fuel_consumption`, `cty_name`,
`cty_vin_num`,
                GET_PRODUCT_VALUE_BY_ID(`oil_type`) AS `oil_type`,
            `oil_capacity`,
                GET_PRODUCT_VALUE_BY_ID(`oil_specification`) AS
`oil_specification`,
            `fac_name`, `man_name`, `add_country`, `add_city`, `add_region`,
            `add_street`, `add_building`, `add_home`, `add_apartment`,
`add_postcode`,
            `cap_width`, `cap_height`, `cap_length`, `cap_weight`
    FROM `product`
    JOIN `matress` ON `matress_id` = `pro_id`
    JOIN `color` USING(`col_id`)
    JOIN `factory` USING(`fac_id`)
    JOIN `manufacturer` USING(`man_id`)
    JOIN `address` USING(`add_id`)
    JOIN `capacity` USING(`cap_id`)
    JOIN `car_type` USING(`cty_id`);
;
CREATE OR REPLACE VIEW GET_MIRRORS AS
    SELECT `product`.*, `col_name`,
                `cty_release_year`, `cty_fuel_consumption`, `cty_name`,
`cty_vin_num`,
                GET_PRODUCT_VALUE_BY_ID(`sra_mechanism_type`) AS
`sra_mechanism_type`,
                GET_PRODUCT_VALUE_BY_ID(`sra_operation_type`) AS
`sra_operation_type`,
            `fac_name`, `man_name`, `add_country`, `add_city`, `add_region`,
            `add_street`, `add_building`, `add_home`, `add_apartment`,
`add_postcode`,
            `cap_width`, `cap_height`, `cap_length`, `cap_weight`
    FROM `product`
    JOIN `mirror` ON `mirror_id` = `pro_id`
    JOIN `color` USING(`col_id`)
    JOIN `factory` USING(`fac_id`)
    JOIN `manufacturer` USING(`man_id`)
    JOIN `address` USING(`add_id`)
    JOIN `capacity` USING(`cap_id`)
```

```
    JOIN `car_type` USING(`cty_id`);
;

CREATE OR REPLACE VIEW GET_ORDERS AS
      SELECT `ord_price`, `ord_created_time`, `product`.*,
               `count`, `address`.*, `del_time_from`, `del_time_to`,
            `del_time_done`, `del_additional_info`
    FROM `order`
      JOIN `m2m_order_product` USING(`ord_id`)
      JOIN `product` USING(`pro_id`)
      JOIN `delivery` USING(`ord_id`)
      JOIN `address` USING(`add_id`);
;
CREATE OR REPLACE VIEW GET_SHELFS AS
    SELECT `product`.*, `col_name`,
                  `cty_release_year`, `cty_fuel_consumption`, `cty_name`,
`cty_vin_num`,
                  GET_PRODUCT_VALUE_BY_ID(`tbe_transport_type`) AS
`tbe_transport_type`,
            `tbe_teeth_number`,
                  GET_PRODUCT_VALUE_BY_ID(`tbe_profile_code`) AS
`tbe_profile_code`,
                  GET_PRODUCT_VALUE_BY_ID(`tbe_teeth_type`) AS
`tbe_teeth_type`,
            `tbe_material`, `fac_name`, `man_name`, `add_country`,
`add_city`, `add_region`,
           `add_street`, `add_building`, `add_home`, `add_apartment`,
           `add_postcode`, `cap_width`, `cap_height`, `cap_length`,
`cap_weight`
    FROM `product`
    JOIN `shelf` ON `shelf_id` = `pro_id`
    JOIN `color` USING(`col_id`)
    JOIN `factory` USING(`fac_id`)
    JOIN `manufacturer` USING(`man_id`)
    JOIN `address` USING(`add_id`)
    JOIN `capacity` USING(`cap_id`)
    JOIN `car_type` USING(`cty_id`);
;
CREATE OR REPLACE VIEW GET_SINKS AS
    SELECT `product`.*, `col_name`,
              `cty_release_year`, `cty_fuel_consumption`, `cty_name`,
`cty_vin_num`,
              `bdi_is_ventilated`,
              GET_PRODUCT_VALUE_BY_ID(`bdi_perforation_type`) AS
`bdi_perforation_type`,
              GET_PRODUCT_VALUE_BY_ID(`bdi_material`) AS `bdi_material`,
              GET_PRODUCT_VALUE_BY_ID(`bdi_installation_side`) AS
`bdi_installation_side`,
              GET_PRODUCT_VALUE_BY_ID(`bdi_type`) AS `bdi_type`,
           `bdi_inner_diameter`, `bdi_outer_diameter`, `bdi_disk_width`,
           `bdi_holes_number`, `bdi_hole_width`, `fac_name`, `man_name`,
           `add_country`, `add_city`, `add_region`, `add_street`,
`add_building`,
           `add_home`, `add_apartment`, `add_postcode`, `cap_width`,
`cap_height`,
           `cap_length`, `cap_weight`
    FROM `product`
    JOIN `sink` ON `sink_id` = `pro_id`
    JOIN `color` USING(`col_id`)
    JOIN `factory` USING(`fac_id`)
    JOIN `manufacturer` USING(`man_id`)
    JOIN `address` USING(`add_id`)
```

```
    JOIN `capacity` USING(`cap_id`)
    JOIN `car_type` USING(`cty_id`);
;
CREATE OR REPLACE VIEW GET_SOFAS AS
    SELECT `product`.*, `col_name`, `sofa`.*,
               `cty_release_year`, `cty_fuel_consumption`, `cty_name`,
`cty_vin_num`,
           `fac_name`, `man_name`, `add_country`, `add_city`, `add_region`,
           `add_street`, `add_building`, `add_home`, `add_apartment`,
`add_postcode`,
           `cap_width`, `cap_height`, `cap_length`, `cap_weight`
    FROM `product`
    JOIN `sofa` ON `sofa_id` = `pro_id`
    JOIN `color` USING(`col_id`)
    JOIN `factory` USING(`fac_id`)
    JOIN `manufacturer` USING(`man_id`)
    JOIN `address` USING(`add_id`)
    JOIN `capacity` USING(`cap_id`)
    JOIN `car_type` USING(`cty_id`);
;
CREATE OR REPLACE VIEW GET_TABLES AS
    SELECT `product`.*, `col_name`, `can_spark_gap`,
               `cty_release_year`, `cty_fuel_consumption`, `cty_name`,
`cty_vin_num`,
               `can_caloric_num`, `can_connecting_dimensions`,
              GET_PRODUCT_VALUE_BY_ID(`can_material`) AS `can_material`,
           `can_num_of_side_electrodes`, `can_is_purify`,
           `fac_name`, `man_name`, `add_country`, `add_city`, `add_region`,
           `add_street`, `add_building`, `add_home`, `add_apartment`,
`add_postcode`,
           `cap_width`, `cap_height`, `cap_length`, `cap_weight`
    FROM `product`
    JOIN `tables` ON `table_id` = `pro_id`
    JOIN `color` USING(`col_id`)
    JOIN `factory` USING(`fac_id`)
    JOIN `manufacturer` USING(`man_id`)
    JOIN `address` USING(`add_id`)
    JOIN `capacity` USING(`cap_id`)
    JOIN `car_type` USING(`cty_id`);
;
CREATE OR REPLACE VIEW GET_TOP_5_DELIVERY_USER_BY_NUM_OF_ORDERS AS
      SELECT `use_id`, COUNT(`use_id`) AS `count`
      FROM `order`
      GROUP BY `use_id`
      ORDER BY COUNT(`count`) DESC LIMIT 5;
;
CREATE OR REPLACE VIEW GET_TOP_5_ORDERS_BY_PRICE AS
      SELECT * FROM `order`
      ORDER BY `ord_price` DESC LIMIT 5;
;
CREATE OR REPLACE VIEW GET_TOP_5_PRODUCT_BY_ORDER AS
      SELECT `pro_id`, SUM(`count`) AS `count`
      FROM `m2m_order_product`
      GROUP BY `pro_id`
      ORDER BY COUNT(`count`) DESC LIMIT 5;
;
CREATE OR REPLACE VIEW GET_USERS_INFO AS
      SELECT `use_registration_time`, `user_info`.*
    FROM `user`
      JOIN `user_info` USING(`use_id`);
;
/* Create Stored Procedures */
```

```
DELIMITER //
CREATE PROCEDURE CLEAR_INFORMATION_ONLINE()
BEGIN
      DELETE FROM `user_online`
      WHERE `uon_out` IS NOT NULL;
END;
//
DELIMITER ;
;
DELIMITER //
CREATE PROCEDURE UPDATE_USER_INFO(
    `user_id` INTEGER UNSIGNED,
      `user_type` INTEGER UNSIGNED,
    `email` VARCHAR(100),
    `password` VARCHAR(100),
    `nickname` VARCHAR(100),
    `first_name` VARCHAR(100),
    `middle_name` VARCHAR(100),
    `last_name` VARCHAR(100),
    `phone` VARCHAR(20)
)
BEGIN
      INSERT INTO `user_info`
    VALUES(
        NULL,
        `user_id`,
        `user_type`,
        `email`,
        `password`,
        `nickname`,
        `first_name`,
        `middle_name`,
        `last_name`,
        `phone`,
        CURDATE()
    );
    UPDATE `user_info` SET `use_id` = `user_id` WHERE `uin_id` =
LAST_INSERT_ID();
END;


//
DELIMITER ;
;
```

Обозначение	Наименование	Дополнительные сведения
	Текстовые документы	
БГУИР КП 1–40 01 01 423 ПЗ	Пояснительная записка	54 с.
	Графические документы	
ГУИР 951004 423 Пл	Логическая модель БД. Плакат	Формат А1

					БГУИР КП 1-40 01 01 423 Д1		
Изм.	Л.	№ докум.	Подп.	Дата		Лист	Листов
Разраб.		Турок К.А.			База данных интернет-магазина мебели Ведомость курсового проекта	Г	54 54
Пров.		Марина И.М.				Кафедра ПОИТ гр. 951004	