

Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра проектирования информационно-компьютерных систем

Дисциплина: Операционные системы, базы данных

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе
на тему

ПРОГРАММНОЕ СРЕДСТВО АВТОСАЛОНА

Студент:
гр. 113802 Разумов Д.А.

Руководитель:
Василькова А. Н.
Медведев О. С.

Минск 2023

СОДЕРЖАНИЕ

Введение.....	4
1 Анализ предметной области.....	5
1.1 Описание предметной области	5
1.2 Группы пользователей, их основные задачи и запросы к базе данных ...	6
1.3 Цели и задачи курсового проектирования	7
2 Проектирование базы данных	8
2.1 Инфологический этап проектирования	8
2.2 Логический этап проектирования	10
2.3 Обоснование выбора технологий и описание среды реализации	21
3 Программная реализация.....	22
3.1 Физическая структура базы данных	22
3.2 Реализация проекта базы данных.....	28
3.2.1 Создание и заполнение таблиц базы данных.....	28
3.2.2 Создание представлений.....	38
3.2.3 Назначение прав доступа.....	42
3.2.4 Создание индексов	44
3.2.5 Создание триггеров	46
3.3 Структура приложения.....	49
4 Тестирование.....	54
5 Описание применения	63
Заключение	64
Список использованных источников	65
Приложение А (обязательное) Листинг кода с комментариями.....	66

ВВЕДЕНИЕ

В этой курсовой работе будет спроектирована база данных для автосалона. В пояснительной записке будут представлены пять разделов: анализ предметной области, проектирование базы данных, программная реализация, тестирование, описание применения.

В первом разделе «Анализ предметной области» будут раскрыты следующие аспекты курсовой работы: описание предметной области, группы пользователей, их основные задачи и запросы к базе данных, цели и задачи курсового проектирования.

Во втором разделе «Проектирование базы данных» будут раскрыты следующие аспекты курсовой работы: инфологический этап проектирования, логический этап проектирования, обоснование выбора технологий и описание среды реализации.

В третьем разделе «Программная реализация» будут раскрыты следующие аспекты курсовой работы: Физическая структура базы данных, Реализация проекта базы данных, Структура приложения.

В четвертом разделе «Тестирование» будет продемонстрирована работа сайта.

В пятом разделе «Описание применения» будут представлены инструкции и рекомендации к использованию.

Перечень графического материала данной курсовой работы будет включать в себя две диаграммы: диаграмму деятельности и диаграмму вариантов использования.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Описание предметной области

В рамках данной курсовой работы рассматривается предметная область «Программное средство автосалона». В современном мире, в условиях, когда автомобиль является не роскошью, а постоянным средством передвижения, то практически каждый человек нуждается в нем. За частую, если человек не испытывает финансовые трудности, то он обращается в автосалон за покупкой нового автомобиля. Эта информация обширна и разрознена. Чтобы вести учет всех автомобилей, которые можно купить, в организации имеется потребность в структурировании данных об авто и заказах. Отсутствие такой возможности приводит к проблеме утери данных и большим временным затратам на выборку данных.

В функции автосалона входит не только продажа автомобилей, но и регистрация новых авто, обмен поддержанных и старых автомобилей на утилизацию, осуществление своих обязательств перед покупателем в виде ремонта и обслуживания автомобиля, осуществление сделок по поставке и продаже автомобилей, за частую и на международном уровне, профессиональная помощь и консультация, возможно, своим будущим покупателям.

Функционирование автосалона без автоматизации процесса достаточно трудоемкая работа. При автоматизации значительно сократиться время и осуществление основных операций автосалона. Автосалон располагает современной материальной базой, включающей вычислительную технику.

Автосалон – важная часть структуры каждого автомобильного бренда. Они представляют собой специальные магазины, офисы, дилеров, осуществляющие посреднические функции для каждого бренда. В большинстве случаев автосалоны являются импортерами каждого бренда автомобиля, в отдельной стране, а также в тех странах, которым принадлежит данный бренд автомобиля, автосалоны являются государственными учреждениями, но при этом имеют большое число посреднических фирм, эффективность деятельности которых очень высока.

Данная категория включает в себя исторически сформировавшийся экономический механизм, который осуществляет определенный комплекс услуг и отношений, содействует установлению и соблюдению баланса интересов между клиентами, автоконцернами (заводами), бизнесом и государством.

Автосалон - юридическое лицо, имеющее зарегистрированный товарный знак, фирменный знак, расчетный и иные счета в рублях и иностранной валюте в учреждениях банков. Автосалон имеет в своем составе администрацию, бухгалтерию, отдел кадров, отдел маркетинга и менеджмента, магазин.

Автомобильный бренд может быть представлен разными «фигурами» в зависимости от того, в чье имущество входит бренд. Им могут быть: отдельный человек или компания, которая является создателем данного бренда и

имеет на него большую часть прав (Tesla Илона Маска), но в большинстве случаев это государственные большие автоконцерны (BMW Германия).

Клиенты – это люди, которые включены в процесс приобретения нового автомобиля по своим личным целям. В свою очередь автосалоны и автоконцерны являются организаторами процесса продажи и производства автомобилей. Клиенты, соответственно, люди разного пола и возраста, с абсолютно разным уровнем заработка, с собственным вкусом и предпочтением, для которых необходим широкий выбор как в ценовом диапазоне, так и в большом выборе моделей каждого бренда. Для каждого покупателя автомобиль имеет разную ценность, поскольку все люди различаются по признакам возраста, пола, личного заработка и уровня жизни, и у каждого эксплуатация автомобиля разная.

Ограничениями каждого автосалона являются:

- продажа авто в кредит или рассрочку людям неспособным выплатить данную сумму в срок;

- продажа авто лицам младше 18 лет;

- оформление авто на людей, не имеющих водительского удостоверения;

Главными мерками успешной покупки автомобиля из салона является:

- высококлассная подготовка, позволяющая менеджеру правильно изложить всю информацию, чтобы облегчить выбор покупателю;

- большой выбор автомобилей;

- минимальное представление, что он ожидает от каждого автомобиля.

В каждом автосалоне развернута локальная вычислительная сеть (ЛВС), которая позволяет осуществлять централизованное хранение и обработку информации. Сеть охватывает все отделы.

1.2 Группы пользователей, их основные задачи и запросы к базе данных

Для определения информационных потребностей необходимо знать кто будет являться пользователем базы данных, и какая информация может потребоваться.

Пользователь базы данных – это физическое лицо, которое имеет доступ к базе данных и пользуется услугами информационной системы для получения информации.

Параметрические (конечные) пользователи – это наиболее многочисленная группа лиц, для удовлетворения информационных потребностей которых и создается БД. Это специалисты в своей области деятельности (руководители подразделений предприятия, работники медицинских учреждений, читатели тематических библиотек, кассиры в сберегательных кассах и т. д.), которые обычно не имеют специальной подготовки по программированию. Они подразделяются на прямых конечных пользователей и косвенных конечных пользователей.

Прямые конечные пользователи обращаются с базой данных в диалоговом режиме. Часть из них умеет обращаться к заранее составленным приложениям и интерпретировать ответы информационной системы. Другие умеют самостоятельно разрабатывать новые приложения.

С данным программным средством работают следующие группы пользователей:

- клиенты;
- посетители;
- менеджеры;

– администраторы;

«Посетитель» может:

- получить полный перечень автомобилей, находящихся в автосалоне;
- получить полную характеристику каждого автомобиля;
- получить все адреса и контакты всех филиалов данного автосалона;
- зарегистрироваться на сайте, чтобы стать клиентом;

«Клиент» может:

- получить полный перечень автомобилей, находящихся в автосалоне;
- получить полную характеристику каждого автомобиля;
- получить все адреса и контакты всех филиалов данного автосалона;
- просмотреть свои заказы;
- совершить заказ;
- управлять своими личными данными;

«Менеджеры» может:

- просматривать данные в таблице «Атомобили», «Модели», «Заказы»;
- вносить изменения (добавлять, редактировать, удалять) в таблицах:

«Автомобили», «Модели», «Заказы»;

«Администратор» может:

– просматривать данные в таблицах: «Клиенты», «Сотрудники», «Контакты»;

– вносить изменения (добавлять, редактировать, удалять) в таблицах: «Клиенты», «Сотрудники», «Контакты»;

–

1.3 Цели и задачи курсового проектирования

Целью курсовой работы является разработка базы данных «Автосалон». Для реализации поставленной цели необходимо решить следующие задачи:

- Изучить литературу по теме курсового проекта.
- Сделать анализ предметной области.
- Спроецировать и реализовать базу данных в среде конкретной СУБД.
- Подготовить графическую часть.

2 ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

2.1 Инфологический этап проектирования

Первым этапом и самым главным этапом в процессе проектирования и создания базы данных, является разработка инфологической модели.

Цель инфологического моделирования – обеспечение наиболее естественных для человека способов сбора и представления той информации, которую предполагается хранить в создаваемой базе данных. Основными конструктивными элементами инфологических моделей являются сущности, связи между ними и их свойства (атрибуты).

В данном курсовом проекте база данных создается для систематизации данных в автосалоне. База должна содержать данные о сотрудниках, клиентах, заказах, автомобилях и контактную информацию для связи.

В соответствии с предметной областью база данных проектируется с учётом следующих особенностей:

1. все авто оформляются в рамках заказа;
2. каждое авто содержит только одну модель, но каждая модель может содержать несколько авто.
3. каждый заказ совершается только одним клиентом, но каждый клиент может совершить несколько заказов;
4. каждый заказ совершается только одним сотрудником, но каждый сотрудник может совершить несколько заказов;
5. каждый авто может заказать несколько клиентов;
6. каждой модели авто может принадлежать несколько автомобилей;

Для базы данных «Автосалон» на основании проведенного анализа предметной области выделены следующие сущности:

Сотрудники. Атрибуты: ФИО; Номер телефона; Должность; Дата рождения; Логин; Пароль; Доступ; Роль; Email; Дата регистрации; Паспортные данные; Адресные данные.

Клиенты. Атрибуты: ФИО; Номер телефона; Дата рождения; Логин; Пароль; Доступ; Роль; Email; Количество заказов; Дата регистрации; Паспортные данные; Адресные данные.

Заказы. Атрибуты: Дата заказа; Клиент; Авто.

Автомобили. Атрибуты: Название; Двигатель; Год выпуска; Цена; Цвет; Комплектация; Фото; Модель; Наличие; Состояние;

Контакты. Атрибуты: Название; Номер телефона; Email; Время работы.

Анализ выявленных сущностей позволяет построить ER–диаграмму как показано на рисунке 1.

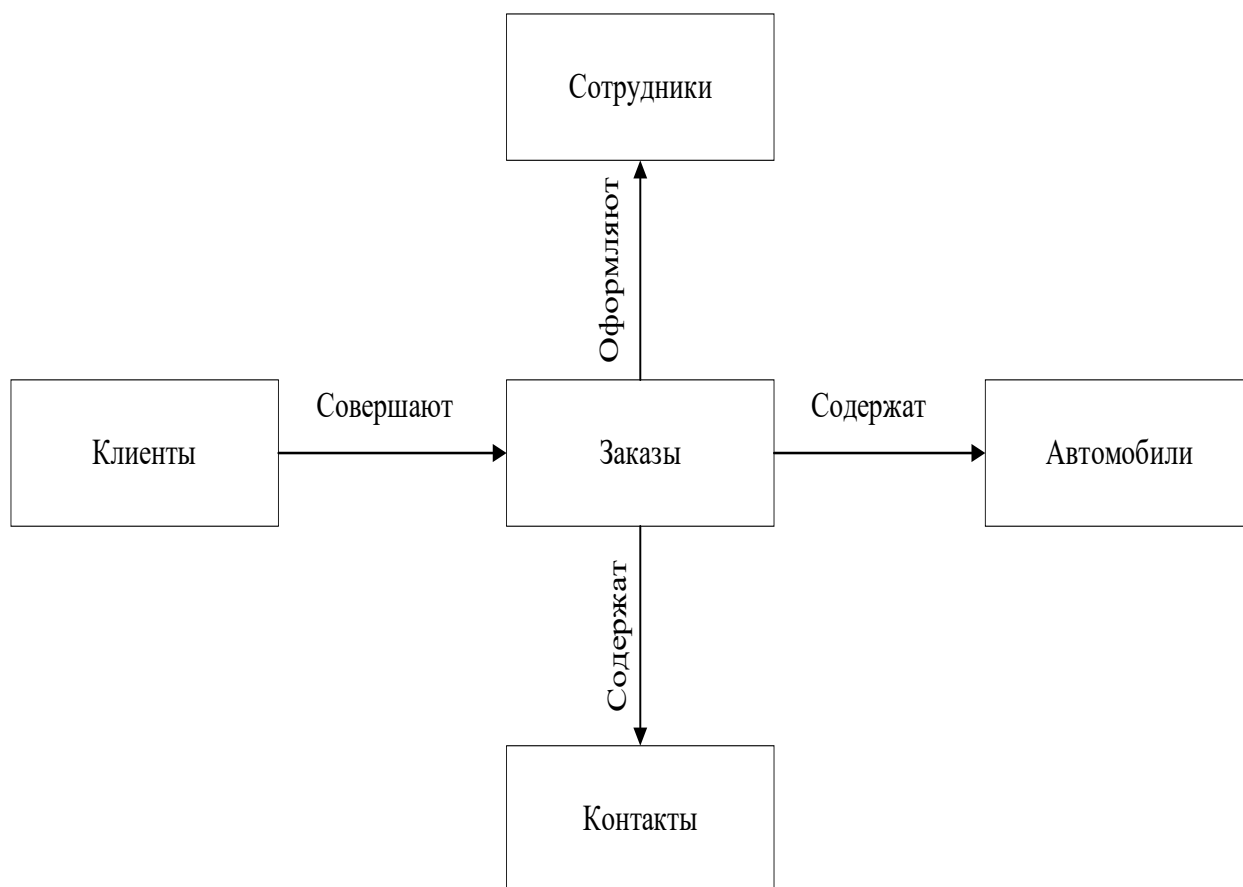


Рисунок 1 – ER-диаграмма предметной области

Для преобразования ER-диаграммы в схему базы данных приведём уточнённую ER-диаграмму, содержащую атрибуты сущностей как показано на рисунке 2.

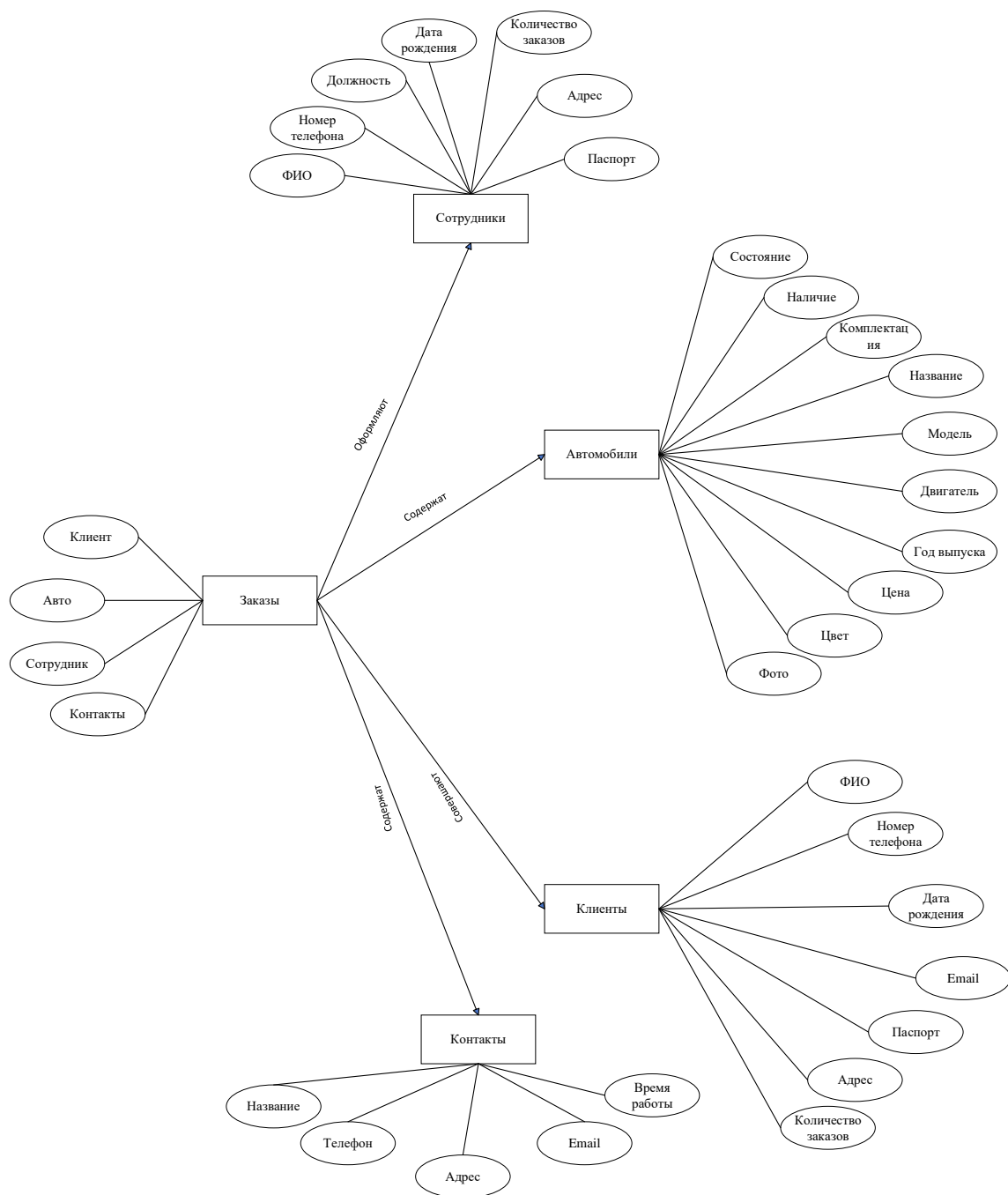


Рисунок 2 – Уточненная ER-диаграмма предметной области

2.2 Логический этап проектирования

Процесс проектирования БД должен опираться на определенную модель данных (реляционная, сетевая, иерархическая), которая определяется типом предполагаемой для реализации информационной системы СУБД.

Логическое проектирование базы данных – это процесс создания модели используемой на предприятии информации на основе выбранной модели организации данных, но без учета типа целевой СУБД и других физических аспектов реализации.

На основе созданных сущностей составим реляционные отношения.

В сущности, «Заказы» первичным ключом является атрибут «Код заказа». Описание сущности «Заказы», предназначенное для хранения данных о всех совершенных заказах, представлено в таблице 1.

Таблица 1 – Описание отношения «Заказы»

Название атрибута	Заполнение	Описание
Код заказа	Обязательное (ключевое поле)	Идентификационный номер заказа
Дата	Обязательное	Дата оформления заказа
Клиент	Обязательное	Клиент оформивший заказ
Сотрудник	Обязательное	Сотрудник
Автомобиль	Обязательное	Автомобиль, который купили
Контакты	Обязательное	Контактные данные

В сущности, «Сотрудники» первичным ключом является атрибут «Код сотрудники». Описание сущности «Сотрудники», предназначенное для хранения данных обо всех сотрудниках, представлено в таблице 2.

Таблица 2 – Описание отношения «Сотрудники»

Название атрибута	Заполнение	Описание
Код сотрудника	Обязательное (ключевое поле)	Идентификационный номер сотрудника
ФИО	Обязательное	Фамилия, имя, отчество сотрудника
Номер телефона	Необязательное	Номер телефона сотрудника
Дата рождения	Обязательное	Дата рождения сотрудника
Должность	Обязательное	Админ или менеджер
Количество заказов	Обязательное	Количество заказов данного сотрудника

Продолжение таблицы 2

Адрес	Необязательное	Адресные данные: город, улица, дом, квартира
Паспорт	Необязательное	Паспортные данные: серия и номер паспорта, кем выдан

В сущности, «Клиенты» первичным ключом является атрибут «Код клиента». Описание сущности «Клиенты», предназначенное для хранения данных обо всех клиентах, представлено в таблице 3.

Таблица 3 – Описание отношения «Клиенты»

Название атрибута	Заполнение	Описание
Код клиента	Обязательное (ключевое поле)	Идентификационный номер клиента
ФИО	Обязательное	Фамилия, имя, отчество клиента
Номер телефона	Необязательное	Номер телефона клиента
Дата рождения	Обязательное	Дата рождения клиента
Количество заказов	Обязательное	Количество заказов клиента
Адрес	Необязательное	Адресные данные: город, улица, дом, квартира
Паспорт	Необязательное	Паспортные данные: серия и номер паспорта, кем выдан

В сущности, «Контакты» первичным ключом является атрибут «Код контакты». Описание сущности «Контакты», предназначенное для хранения контактных данных автосалона, представлено в таблице 4.

Таблица 4 – Описание отношения «Контакты»

Название атрибута	Заполнение	Описание
Код контакта	Обязательное (ключевое поле)	Идентификационный номер контакта
Название	Обязательное	Название
Номер телефона	Обязательное	Номер телефона
Email	Обязательное	Email контактного центра

Продолжение таблицы 4

Время работы	Обязательное	Время работы контактного центра
Адрес	Обязательное	Адресные данные: город, улица, дом

В сущности, «Автомобили» первичным ключом является атрибут «Код автомобиля». Описание сущности «Автомобили», предназначенное для хранения данных обо всех автомобилях автосалона, представлено в таблице 5.

Таблица 5 – Описание отношения «Автомобили»

Название атрибута	Заполнение	Описание
Код автомобиля	Обязательное (ключевое поле)	Идентификационный номер автомобиля
Название	Обязательное	Название автомобиля
Модель	Обязательное	Модель автомобиля
Двигатель	Обязательное	Электрический или бензиновый
Год выпуска	Обязательное	Год выпуска автомобиля
Цена	Обязательное	Цена автомобиля
Цвет	Обязательное	Цвет автомобиля
Комплектация	Обязательное	Базовая, полная или средняя
Фото	Обязательное	Фотография автомобиля
Состояние	Обязательное	Новое или БУ
Статус	Обязательное	Есть в наличии или нету в наличии

Общий вид логической схемы базы данных приведен на рисунке 3.

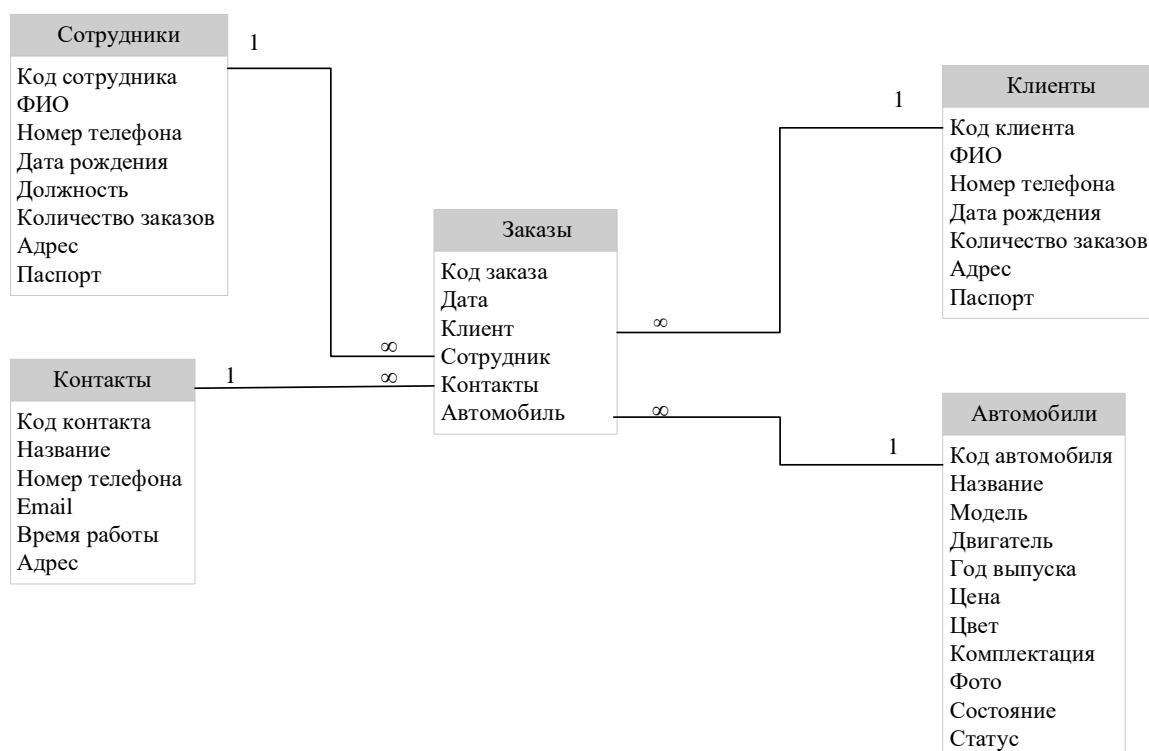


Рисунок 3 – Схема базы данных

Нормализация отношений (таблиц) – одна из основополагающих частей теории реляционных баз данных. Нормализация имеет своей целью избавиться от избыточности в отношениях и модифицировать их структуру таким образом, чтобы процесс работы с ними не был обременён различными посторонними сложностями. При игнорировании такого подхода эффективность проектирования стремительно снижается, что вкупе с прочими подобными вольностями может привести к критическим последствиям.

Нормальная форма – свойство отношения в реляционной модели данных, характеризующее его с точки зрения избыточности, потенциально приводящей к логически ошибочным результатам выборки или изменения данных. Нормальная форма определяется как совокупность требований, которым должно удовлетворять отношение.

Приведем разработанную базу данных к 3НФ.

Первая нормальная форма (1НФ) предполагает, что каждое сохраняемое данное должно представлять скалярное значение, а отношение не должно содержать повторяющихся строк.

Нормализации подлежат все отношения.

Вынесем сложный атрибут Адрес у клиентов и сотрудников на «Город», «Улица», «Дом», «Квартира» в отдельные отношения «Адреса клиентов» и «Адреса сотрудников». Вынесем сложный атрибут Адрес у контактов на «Город», «Улица», «Дом» в отдельное отношение «Адреса контактов». Вынесем сложный атрибут Паспорт у клиентов и сотрудников на «Серия», «Номер»,

«Кем выдан» в отдельные отношения «Паспортные данные клиентов» и «Паспортные данные сотрудников». Разобьем поле «ФИО» на «Фамилия», «Имя», «Отчество».

Затем создадим отношение «Пользователи», которая будет связана с отношениями «Клиенты» и «Сотрудники».

После приведения к первой нормальной форме отношений «Заказы», «Клиенты», «Сотрудники», «Контакты», «Автомобили» примут вид, представленный в таблицах 6–17 соответственно.

Таблица 6 – Описание отношения «Заказы»

Название атрибуты	Заполнение	Описание
Код заказа	Обязательное (ключевое поле)	Идентификационный номер заказа
Дата	Обязательное	Дата продажи
Код сотрудника	Обязательное	Идентификационный номер сотрудника
Код клиента	Обязательное	Идентификационный номер клиента
Код контакта	Обязательное	Идентификационный номер контактов
Код автомобиля	Обязательное	Идентификационных номер автомобиля

Таблица 7 – Описание отношения «Контакты»

Название атрибута	Заполнение	Описание
Код контакта	Обязательное (ключевое поле)	Идентификационный номер
Название	Обязательное	Название
Номер телефона	Обязательное	Номер телефона
Email	Обязательное	Email
Время работ	Обязательное	Время работы контактного центра
Код адреса	Обязательное	Идентификационный номер адресных данных

Таблица 8 – Описание отношения «Адреса контактов»

Название атрибута	Заполнение	Описание
Код адреса	Обязательное (ключевое поле)	Идентификационный номер адреса
Город	Обязательное	Город контактного центра
Улица	Обязательное	Улица контактного центра
Дом	Обязательное	Номер дома контактного центра

Таблица 9 – Описание отношения «Клиенты»

Название атрибуты	Заполнение	Описание
Код клиента	Обязательное (ключевое поле)	Идентификационный номер
Фамилия	Обязательное	Фамилия клиента
Имя	Обязательное	Имя клиента
Отчество	Необязательное	Отчество клиента
Дата рождения	Обязательное	Дата рождения клиента
Номер телефона	Необязательное	Номер телефона клиента
Количество заказов	Обязательное	Количество заказов клиента
Код адреса	Обязательное	Идентификационный номер адресных данных
Код паспорта	Обязательное	Идентификационный номер паспортных данных
Код авторизации	Обязательное	Идентификационный номер данных для авторизации

Таблица 10 – Описание отношения «Адреса клиентов»

Название атрибута	Заполнение	Описание
Код адреса	Обязательное (ключевое поле)	Идентификационный номер адресных данных

Продолжение таблицы 10

Город	Необязательное	Город клиента
Улица	Необязательное	Улица клиента
Дом	Необязательное	Номер дома клиента
Квартира	Необязательное	Номер квартиры клиента

Таблица 11 – Описание отношения «Паспортные данные клиентов»

Название атрибута	Заполнение	Описание
Код паспорта	Обязательное (ключевое поле)	Идентификационный номер паспортных данных
Серия	Необязательное	Серия паспорта клиента
Номер	Необязательное	Номер паспорта клиента
Кем выдан	Необязательное	Кем выдан паспорт

Таблица 12 – Описание отношения «Сотрудники»

Название атрибуты	Заполнение	Описание
Код сотрудника	Обязательное (ключевое поле)	Идентификационный номер
Фамилия	Обязательное	Фамилия сотрудника
Имя	Обязательное	Имя сотрудника
Отчество	Необязательное	Отчество сотрудника
Дата рождения	Обязательное	Дата рождения сотрудника
Номер телефона	Необязательное	Номер телефона сотрудника
Должность	Обязательное	Админ или менеджер
Количество заказов	Обязательное	Количество заказов сотрудника
Код адреса	Обязательное	Идентификационный номер адресных данных

Продолжение таблицы 12

Код паспорта	Обязательное	Идентификационный номер паспортных данных
Код авторизации	Обязательное	Идентификационный номер данных для авторизации

Таблица 13 – Описание отношения «Адреса сотрудников»

Название атрибута	Заполнение	Описание
Код адреса	Обязательное (ключевое поле)	Идентификационный номер адресных данных
Город	Необязательное	Город сотрудника
Улица	Необязательное	Улица сотрудника
Дом	Необязательное	Номер дома сотрудника
Квартира	Необязательное	Номер квартиры сотрудника

Таблица 14 – Описание отношения «Паспортные данные сотрудников»

Название атрибута	Заполнение	Описание
Код паспорта	Обязательное (ключевое поле)	Идентификационный номер паспортных данных
Серия	Необязательное	Серия паспорта сотрудника
Номер	Необязательное	Номер паспорта сотрудника
Кем выдан	Необязательное	Кем выдан паспорт

Таблица 15 – Описание сущности «Авторизация»

Название атрибута	Заполнение	Описание
Код авторизации	Обязательное (ключевое поле)	Идентификационный номер данных для авторизации
Логин	Обязательное	Логин
Пароль	Обязательное	Пароль
Доступ	Обязательное	Доступ имеется или не имеется

Продолжение таблицы 15

Роль	Обязательное	Клиент или сотрудник
Дата регистрации	Обязательное	Дата регистрации
Email	Обязательное	Email

2НФ. В нашем случае составные первичные ключи отсутствуют, так имеют простой первичный ключ и поэтому они сразу находятся во второй нормальной форме.

3НФ. В отношении «Автомобили» атрибут «Модель» зависит не от первичного ключа, поэтому его следует вынести в отдельное отношение «Модели».

Таблица 16 – Описание сущности «Автомобили»

Название атрибута	Заполнение	Описание
Код автомобиля	Обязательное (ключевое поле)	Идентификационный номер автомобиля
Название	Обязательное	Название автомобиля
Двигатель	Обязательное	Электрический или бензиновый
Год выпуска	Обязательное	Год выпуска автомобиля
Цена	Обязательное	Стоимость автомобиля
Цвет	Обязательное	Цвет автомобиля
Комплектация	Обязательное	Базовая, средняя или полная
Фотография	Обязательное	Фотография автомобиля
Состояние	Обязательное	Новое или БУ
Статус	Обязательное	Есть в наличии или нету
Код модели	Обязательное	Идентификационный номер модели автомобиля

Таблица 17 – Описание сущности «Модели»

Название атрибута	Заполнение	Описание
Код модели	Обязательное (ключевое поле)	Идентификационный номер модели автомобиля
Модель	Обязательное	Название модели
Фотография	Обязательное	Общая фотография для данной модели автомобиля
Количество	Обязательное	Количество машин данной модели в автосалоне

Схема базы данных после нормализации приведена на рисунке 4.

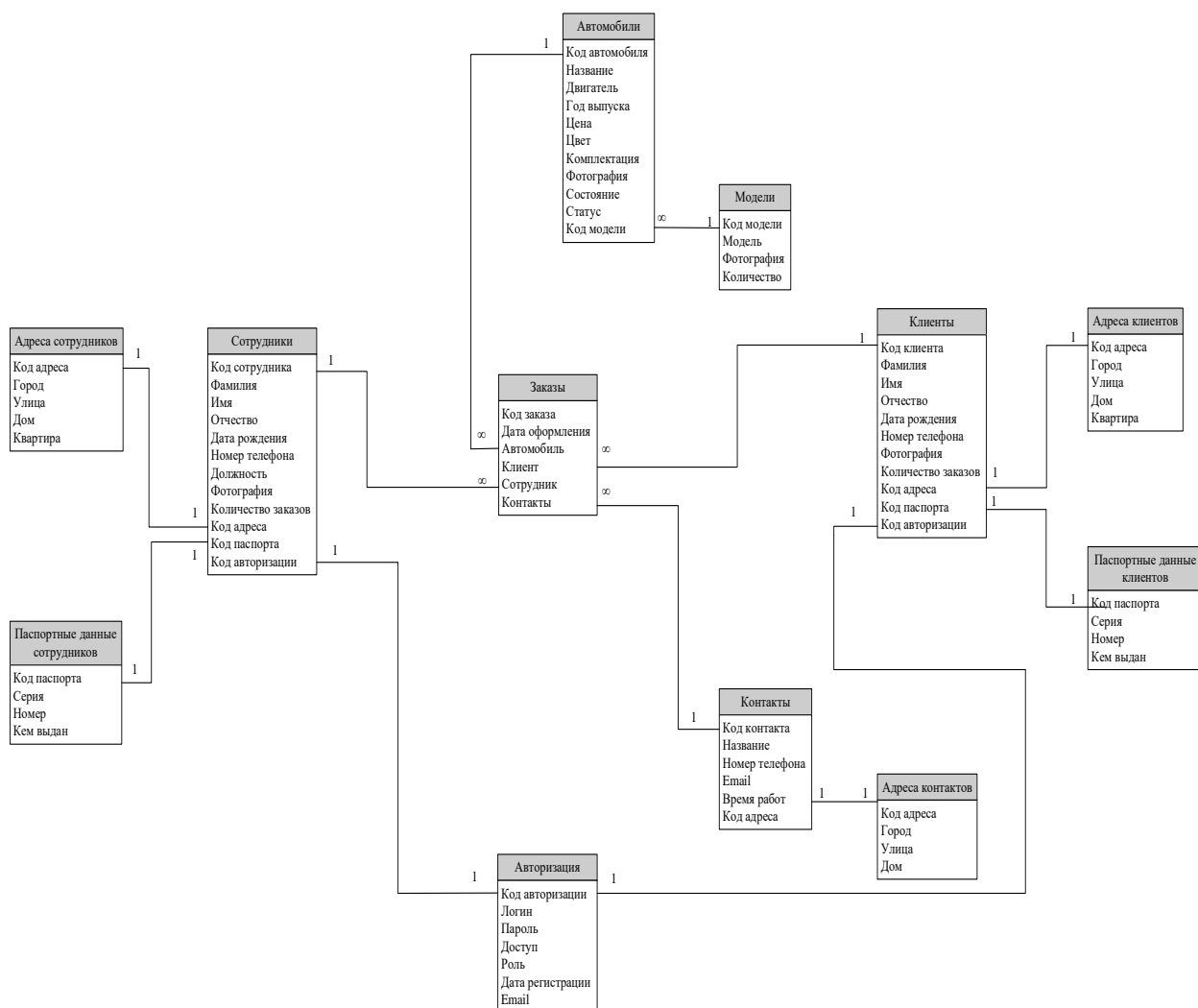


Рисунок 4 – Схема данных после нормализации

2.3 Обоснование выбора технологий и описание среды реализации

В качестве среды проектирования базы данных был выбран MySQL 8.0.31 – система управления реляционными базами данных.

Разработку и поддержку MySQL осуществляет корпорация Oracle, получившая права на торговую марку вместе с поглощённой Sun Microsystems, которая ранее приобрела шведскую компанию GNU General Public License, так и под собственной коммерческой лицензией.

Улучшения MySQL 8.0.31:

1. Использование по умолчанию движка InnoDB.
2. Улучшение функций по секционированию данных. Расширенный синтаксис для разбиения больших таблиц на несколько частей, размещенных в файловых системах.
3. Новый механизм оптимизации вложенных запросов и JOIN-операций.
4. Переработана система внутренних блокировок.

Для визуального проектирования был использован продукт PHPMysqlAdmin. PHPMysqlAdmin – это унифицированный визуальный инструмент для архитекторов баз данных, разработчиков и администраторов баз данных. PHPMysqlAdmin предоставляет моделирование данных, разработку SQL-кода и комплексные инструменты администрирования для настройки сервера, администрирования пользователей, резервного копирования и многого другого. PHPMysqlAdmin доступен в Windows, Linux и Mac OS X.

Ключевые особенности включают:

- позволяет создавать модели и управлять ими, преобразовывать динамическую базу данных в модель, а также создавать и редактировать таблицы, и вставлять данные;
- преобразовывать диаграммы ER в операторы SQL и отправлять их на сервер SQL, создавать модели из целевой базы данных или даже из импортированных файлов SQL;
- позволяет создавать, управлять и настраивать подключения и параметры подключения к серверам баз данных MySQL, также позволяет выполнять SQL-запросы к этим соединениям с помощью встроенного редактора;
- позволяет создавать, управлять и организовывать подключения к базе данных, визуально выбирать таблицы и столбцы, легко перемещаться между схемами, выбирать таблицы и поля, создавать новые или отбрасывать их;
- значительно упрощает управление пользователями: просмотр учетных записей все пользователей сервера, добавление и удаление пользователей, предоставлять и отзывать привилегии, изменять глобальные разрешения и разрешения базы данных, изменять пароли, проводить аудит;

Все эти функции и интуитивно понятный интерфейс делают PHPMysqlAdmin незаменимым инструментом для MySQL на локальном рабочем столе.

3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

3.1 Физическая структура базы данных

Физическая структура базы данных определяет тип и свойства данных, которые будут записаны в память компьютера.

На стадии физического проектирования происходит непосредственная реализация базы данных. Были созданы следующие таблицы: работодатели, вакансии, пол, должности, агент, образование, соискатели, сделки. Определены первичные и внешние ключи, типы данных полей таблиц.

Типы данных разделяют на числовые, дробные, символьные, даты и времени. Основные типы данных:

1. Числовые – TINYINT, SMALLINT, MEDIUMINT, INT, BIGINT;
 2. Дробные – FLOAT, DOUBLE, DECIMAL;
 3. Символьные – CHAR, VARCHAR, TINYTEXT, TEXT и т.д.;
 4. Даты и времени – DATE, DATETIME, TIMESTAMP, TIME, YEAR.
- В базе данных были использованы: INT, VARCHAR, DATE, DOUBLE:
- INT – хранит числа в диапазоне от -2147483648 до 2147483647;
 - VARCHAR – может хранить не более 255 символов;
 - DATE – дата в формате ГГГГ-ММ-ДД;
 - TEXT – может хранить не более 65 535 символов.

Таблица «Заказы» хранит информацию о заказах. Структура приведена в таблице 18.

Таблица 18 – Структура таблицы «Заказы».

Название поля	Тип	Ключ	Описание
id	Int	Первичный	Идентификационный номер заказа
date	Timestamp		Дата оформления заказа
id_client	Int	Внешний	Идентификационный номер клиента
id_employee	Int	Внешний	Идентификационный номер сотрудника
id_auto	Int	Внешний	Идентификационный номер автомобиля
id_contact	Int	Внешний	Идентификационный номер контактов

Таблица «Контакты» хранит информацию о контактах. Структура приведена в таблице 19.

Таблица 19 – Структура таблицы «Контакты».

Название поля	Тип	Ключ	Описание
id	Int	Первичный	Идентификационный номер заказа
name	Varchar(60)		Название контактного центра
phone	Varchar(20)		Номер телефона
email	Varchar(60)		Email
work_time	Text		Время работы
id_address	Int	Внешний	Идентификационный номер адресных данных

Таблица «Адреса контактов» хранит информацию об адресах контактов. Структура приведена в таблице 20.

Таблица 20 – Структура таблицы «Адреса контактов».

Название поля	Тип	Ключ	Описание
id	Int	Первичный	Идентификационный номер адреса
city	Varchar(30)		Город
street	Varchar(30)		Улица
house	Varchar(30)		Дом

Таблица «Сотрудники» хранит информацию о сотрудниках автосалона. Структура приведена в таблице 21.

Таблица 21 – Структура таблицы «Сотрудники».

Название поля	Тип	Ключ	Описание
id	Int	Первичный	Идентификационный номер сотрудника
last_name	Varchar(60)		Фамилия сотрудника
first_name	Varchar(60)		Имя сотрудника
surname	Varchar(60)		Отчество сотрудника

Продолжение таблицы 21

phone	Varchar(20)		Номер телефона сотрудника
date_birth	Date		Дата рождения сотрудника
job	Varchar(255)		Должность
count_orders	Int		Количество заказов
id_address	Int	Внешний	Идентификационный номер адреса
id_auth	Int	Внешний	Идентификационный номер авторизации
id_passport	Int	Внешний	Идентификационный номер паспорта

Таблица «Адреса сотрудников» хранит информацию об адресных данных сотрудников автосалона. Структура приведена в таблице 22.

Таблица 22 – Структура таблицы «Адреса сотрудников».

Название поля	Тип	Ключ	Описание
id	Int	Первичный	Идентификационный номер сотрудника
city	Varchar(60)		Город сотрудника
street	Varchar(60)		Улица сотрудника
house	Varchar(5)		Номер дома сотрудника
apartment	Int		Номер квартиры сотрудника

Таблица «Паспортные данные сотрудников» хранит информацию об паспортных данных сотрудников. Структура приведена в таблице 23.

Таблица 23 – Структура таблицы «Паспортные данные сотрудников».

Название поля	Тип	Ключ	Описание
id	Int	Первичный	Идентификационный номер сотрудника
series	Varchar(5)		Серия паспорта сотрудника
number	Varchar(10)		Номер паспорта сотрудника

Таблица «Клиенты» хранит информацию о клиентах автосалона. Структура приведена в таблице 24.

Таблица 24 – Структура таблицы «Клиента».

Название поля	Тип	Ключ	Описание
id	Int	Первичный	Идентификационный номер клиента
last_name	Varchar(60)		Фамилия клиента
first_name	Varchar(60)		Имя клиента
surname	Varchar(60)		Отчество клиента
phone	Varchat(30)		Номер телефона клиента
date_birth	Date		Дата рождения клиента
id_address	Int	Внешний	Идентификационный номер адреса
id_auth	Int	Внешний	Идентификационный номер авторизации
id_passport	Int	Внешний	Идентификационный номер паспорта

Таблица «Адреса клиентов» хранит информацию об адресных данных клиентов автосалона. Структура приведена в таблице 25.

Таблица 25 – Структура таблицы «Адреса клиентов».

Название поля	Тип	Ключ	Описание
id	Int	Первичный	Идентификационный номер клиента
city	Varchar(60)		Город клиента
street	Varchar(50)		Улица клиента
house	Varchar(5)		Номер дома клиента
apartment	Varchar(5)		Номер квартиры клиента

Таблица «Паспортные данные клиентов» хранит информацию об паспортных данных клиентов автосалона. Структура приведена в таблице 26.

Таблица 26 – Структура таблицы «Паспортные данные клиентов».

Название поля	Тип	Ключ	Описание
id	Int	Первичный	Идентификационный номер клиента
series	Varchar(5)		Серия паспорта клиента
number	Varchar(10)		Номер паспорта клиента
issued_by	Varchar(60)		Кем выдан клиента

Таблица «Авторизации» хранит информацию обо всех пользователях автосалона. Структура приведена в таблице 27.

Таблица 27 – Структура таблицы «Авторизация».

Название поля	Тип	Ключ	Описание
id	Int	Первичный	Идентификационный номер пользователя
login	Varchar(45)		Логин пользователя
password	Varchar(255)		Пароль пользователя
access	Tinyint		Доступ имеется или не имеется
role	Tinyint		Клиент или сотрудник
date_regist	Timestampt		Дата регистрации пользователя
email	Varchar(60)		Email пользователя

Таблица «Автомобили» хранит информацию обо всех автомобилях в автосалоне. Структура приведена в таблице 28.

Таблица 28 – Структура таблицы «Автомобили».

Название поля	Тип	Ключ	Описание
id	Int	Первичный	Идентификационный номер автомобиля
name	Varchar(30)		Название автомобиля
engine	Varchar(30)		Двигатель автомобиля

Продолжение таблицы 28

year	Int		Год выпуска автомобиля
price	Varchar(12)		Стоимость автомобиля
color	Varchar(12)		Цвет автомобиля
complexion	Varchar(20)		Комплектация базовая средняя или полная
img	Varchar(255)		Фотография автомобиля
status	Tinyint		Есть в наличии или нет
state	Varchar(12)		Состояние
id_model	Int	Внешний	Идентификационный номер модели автомобиля

Таблица «Модели» хранит информацию обо всех моделях автомобилей автосалона. Структура приведена в таблице 29.

Таблица 29 – Структура таблицы «Модели».

Название поля	Тип	Ключ	Описание
id	Int	Первичный	Идентификационный номер
model	Varchar(30)		Название модели
main_foto	Varchar(255)		Главное фото модели
counts	Int		Количество автомобилей данной модели

Структура базы данных показана на рисунке 5.

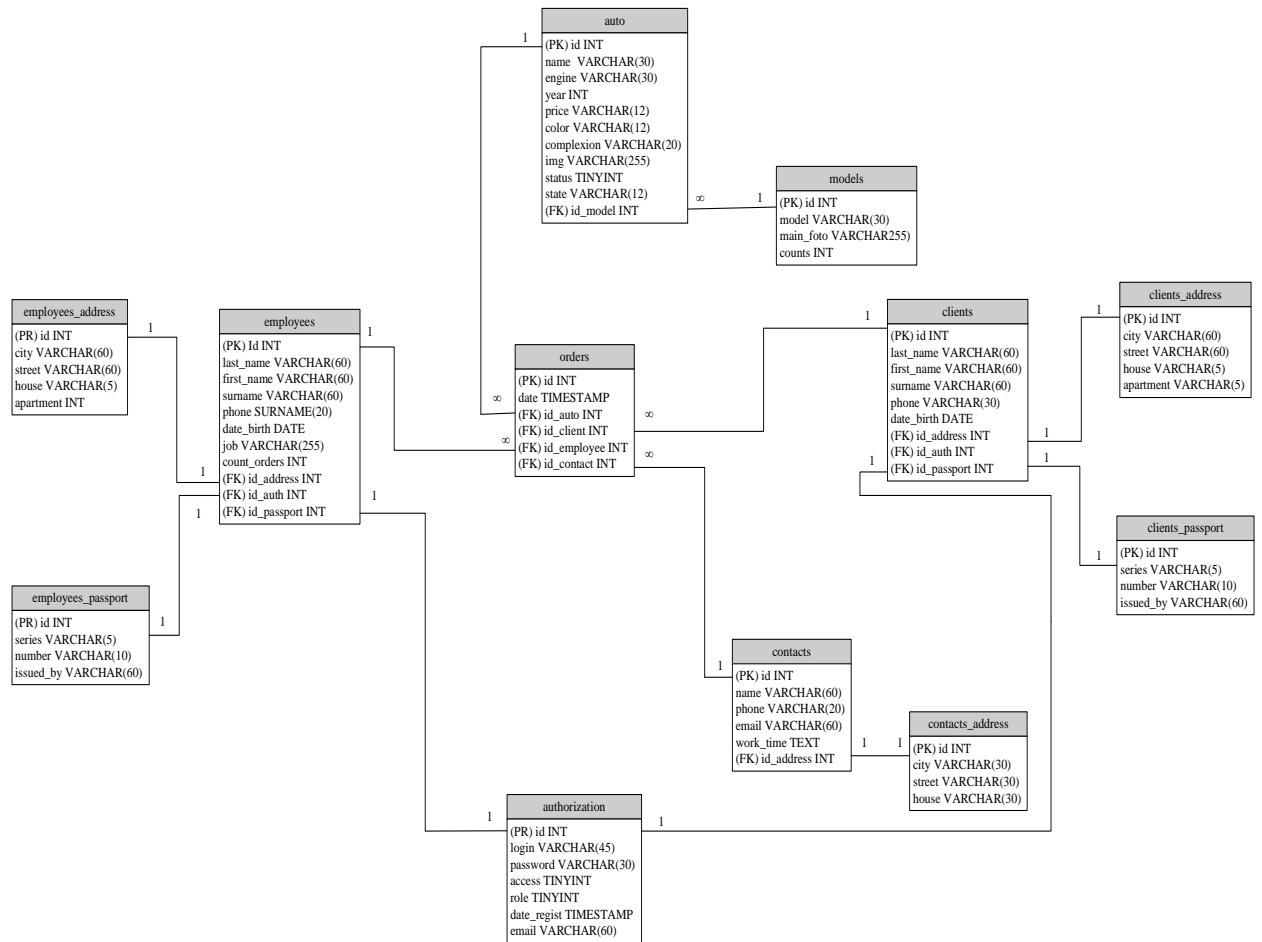


Рисунок 5 – Физическая структура базы данных

3.2 Реализация проекта базы данных

3.2.1 Создание и заполнение таблиц базы данных

Для создания таблиц используется оператор CREATE TABLE. Таблицы создаются по следующим правилам:

- Имя таблицы указывается после ключевого слова CREATE TABLE (если имя состоит из нескольких слов, то его следует заключить в одинарные кавычки).
- Далее в круглых скобках следуют имена столбцов таблицы (полей), после которых указывается тип данных, которому будет принадлежать поле.
- Не обязательно: затем указывается может ли поле содержать пустые значения (NULL — может быть пустым или NOT NULL — обязательно для заполнения).
- Одно из полей назначается первичным ключом (Primary key).
- Поля отделяются запятыми.

Общая команда создания таблицы представлена на рисунке 6:

```
CREATE TABLE название_таблицы  
(название_столбца1 тип_данных атрибуты_столбца1,  
название_столбца2 тип_данных атрибуты_столбца2,  
.....  
название_столбцаN тип_данных атрибуты_столбцаN,  
атрибуты_таблицы);
```

Рисунок 6 – Общая команда создания таблицы

Создание таблиц представлено на рисунках 7–18.

Создание таблицы «Заказы» представлено на рисунке 7.

```
CREATE TABLE `bmv_dealership`.`orders` (  
  `id` INT NOT NULL AUTO_INCREMENT ,  
  `date` TIMESTAMP NOT NULL,  
  `id_client` INT NULL,  
  `id_employee` INT NULL,  
  `id_auto` INT NULL,  
  `id_contact` INT NULL,  
  PRIMARY KEY (`id`)
```

Рисунок 7 – Создание таблицы «Заказы»

Создание таблицы «Контакты» представлено на рисунке 8.

```
CREATE TABLE `bmv_dealership`.`contacts` (  
  `id` INT NOT NULL AUTO_INCREMENT ,  
  `name` VARCHAR(60) NOT NULL ,  
  `phone` VARCHAR(20) NOT NULL ,  
  `work_time` TEXT NOT NULL ,  
  `email` VARCHAR(60) NOT NULL ,  
  `id_address` INT NOT NULL , PRIMARY KEY (`id`));
```

Рисунок 8 – Создание таблицы «Контакты»

Создание таблицы «Адреса контактов» представлено на рисунке 9.

```
CREATE TABLE `bmv_dealership`.`contacts_address` (  
  `id` INT NOT NULL AUTO_INCREMENT ,  
  city` VARCHAR(30) NOT NULL ,  
  `street` VARCHAR(30) NOT NULL ,  
  `house` VARCHAR(30) NOT NULL , PRIMARY KEY (`id`));
```

Рисунок 9 – Создание таблицы «Адреса контактов»

Создание таблицы «Автомобили» представлено на рисунке 10.

```
CREATE TABLE `bmv_dealership`.`auto` (  
  `id` INT NOT NULL AUTO_INCREMENT ,  
  `name` VARCHAR(30) NOT NULL ,  
  `engine` VARCHAR(30) NOT NULL ,  
  `year` INT NOT NULL ,  
  `price` VARCHAR(12) NOT NULL ,  
  `color` VARCHAR(20) NOT NULL ,  
  `complexion` VARCHAR(20) NOT NULL ,  
  `img` VARCHAR(255) NOT NULL ,  
  `status` TINYINT NOT NULL ,  
  `state` VARCHAR(12) NOT NULL ,  
  `id_model` INT NOT NULL ,  
  PRIMARY KEY (`id`));
```

Рисунок 10 – Создание таблицы «Автомобили»

Создание таблицы «Модели» представлено на рисунке 11.

```
CREATE TABLE `bmv_dealership`.`models` (  
  `id` INT NOT NULL AUTO_INCREMENT ,  
  `model` VARCHAR(30) NOT NULL ,  
  `main_foto` VARCHAR(255) NOT NULL ,  
  `counts` INT NOT NULL DEFAULT '0' ,  
  PRIMARY KEY (`id`));
```

Рисунок 11 – Создание таблицы «Автомобили»

Создание таблицы «Авторизация» представлено на рисунке 12.

```
CREATE TABLE `bmv_dealership`.`authorization` (  
  `id` INT NOT NULL AUTO_INCREMENT ,  
  `login` VARCHAR(45) NOT NULL ,  
  `password` VARCHAR(255) NOT NULL ,  
  `access` TINYINT NOT NULL ,  
  `role` TINYINT NOT NULL ,  
  `date_regist` TIMESTAMP NOT NULL ,  
  `email` VARCHAR(60) NOT NULL ,  
  PRIMARY KEY (`id`));
```

Рисунок 12 – Создание таблицы «Авторизация»

Создание таблицы «Клиенты» представлено на рисунке 13.

```
CREATE TABLE `bmv_dealership`.`clients` (  
  `id` INT NOT NULL AUTO_INCREMENT ,  
  `last_name` VARCHAR(60) NOT NULL ,  
  `first_name` VARCHAR(60) NOT NULL ,  
  `surname` VARCHAR(60) NULL ,  
  `date_birth` DATE NOT NULL ,  
  `phone` VARCHAR(30) NULL ,  
  `count_orders` INT NOT NULL DEFAULT '0' ,  
  `id_address` INT NOT NULL ,  
  `id_auth` INT NOT NULL ,  
  `id_passport` INT NOT NULL ,  
  PRIMARY KEY (`id`));
```

Рисунок 13 – Создание таблицы «Клиенты»

Создание таблицы «Адреса клиентов» представлено на рисунке 14.

```
CREATE TABLE `bmv_dealership`.`clients_address` (  
  `id` INT NOT NULL AUTO_INCREMENT ,  
  `city` VARCHAR(60) NULL ,  
  `street` VARCHAR(50) NULL ,  
  `house` VARCHAR(5) NULL ,  
  `apartment` VARCHAR(5) NULL ,  
  PRIMARY KEY (`id`));
```

Рисунок 14 – Создание таблицы «Адреса клиентов»

Создание таблицы «Паспортные данные клиентов» представлено на рисунке 15.

```
CREATE TABLE `bmv_dealership`.`clients_address` (  
  `id` INT NOT NULL AUTO_INCREMENT ,  
  `series` VARCHAR(5) NULL ,  
  `number` VARCHAR(10) NULL ,  
  `issued_by` VARCHAR(60) NULL ,  
  PRIMARY KEY (`id`));
```

Рисунок 15 – Создание таблицы «Паспортные данные клиентов»

Создание таблицы «Сотрудники» представлено на рисунке 16.

```
CREATE TABLE `bmv_dealership`.`employees` (  
  `id` INT NOT NULL AUTO_INCREMENT ,  
  `last_name` VARCHAR(60) NOT NULL ,  
  `first_name` VARCHAR(60) NOT NULL ,  
  `surname` VARCHAR(60) NULL ,  
  `date_birth` DATE NULL ,  
  `job` VARCHAR(255) NOT NULL ,  
  `id_address` INT NOT NULL ,  
  `id_auth` INT NOT NULL ,  
  `id_passport` INT NOT NULL , PRIMARY KEY (`id`));
```

Рисунок 16 – Создание таблицы «Сотрудники»

Создание таблицы «Адреса сотрудников» представлено на рисунке 17.

```
CREATE TABLE `bmv_dealership`.`employees_address` (  
  `id` INT NOT NULL AUTO_INCREMENT ,  
  `city` VARCHAR(60) NULL ,  
  `street` VARCHAR(60) NULL ,  
  `house` VARCHAR(5) NULL ,  
  `apartment` INT NULL ,  
  PRIMARY KEY (`id`));
```

Рисунок 17 – Создание таблицы «Адреса сотрудников»

Создание таблицы «Паспортные данные сотрудников» представлено на рисунке 18.

```
CREATE TABLE `bmv_dealership`.`employees_passports` (  
  `id` INT NOT NULL AUTO_INCREMENT ,  
  `series` VARCHAR(5) NULL ,  
  `number` VARCHAR(10) NULL ,  
  `issued_by` VARCHAR(60) NULL ,  
  PRIMARY KEY (`id`));
```

Рисунок 18 – Создание таблицы «Паспортные данные сотрудников»

Создание связей для таблиц представлено на рисунках 19–23.

Создание связей для таблицы «Заказы» представлено на рисунке 19.

```
ALTER TABLE `orders` ADD FOREIGN KEY (`id_auto`) REFERENCES `auto` (`id`) ON DE-  
LETE SET NULL ON UPDATE SET NULL;  
  
ALTER TABLE `orders` ADD FOREIGN KEY (`id_client`) REFERENCES `clients` (`id`) ON DE-  
LETE SET NULL ON UPDATE SET NULL;  
  
ALTER TABLE `orders` ADD FOREIGN KEY (`id_contact`) REFERENCES `contacts` (`id`) ON DE-  
LETE SET NULL ON UPDATE SET NULL;  
  
ALTER TABLE `orders` ADD FOREIGN KEY (`id_employee`) REFERENCES `employ-  
ees` (`id`) ON DELETE SET NULL ON UPDATE SET NULL;
```

Рисунок 19 – Создание связей для таблицы «Заказы»

Создание связи для таблицы «Контакты» представлено на рисунке 20.

```
ALTER TABLE `contacts` ADD FOREIGN KEY (`id_address`) REFERENCES `contacts_ad-  
dress`(`id`) ON DELETE CASCADE ON UPDATE CASCADE;
```

Рисунок 20 – Создание связей для таблицы «Контакты»

Создание связи для таблицы «Автомобили» представлено на рисунке 21.

```
ALTER TABLE `auto` ADD FOREIGN KEY (`id_model`) REFERENCES `models`(`id`) ON DE-  
LETE CASCADE ON UPDATE CASCADE;
```

Рисунок 21 – Создание связей для таблицы «Автомобили»

Создание связей для таблицы «Клиенты» представлено на рисунке 22.

```
ALTER TABLE `clientss` ADD FOREIGN KEY (`id_address`) REFERENCES `clients_ad-  
dress`(`id`) ON DELETE CASCADE ON UPDATE CASCADE;
```

```
ALTER TABLE `clientss` ADD FOREIGN KEY (`id_auth`) REFERENCES `authorization`(`id`) ON DE-  
LETE CASCADE ON UPDATE CASCADE;
```

```
ALTER TABLE `clientss` ADD FOREIGN KEY (`id_passport`) REFERENCES `clients_pass-  
port`(`id`) ON DELETE CASCADE ON UPDATE CASCADE;
```

Рисунок 22 – Создание связей для таблицы «Автомобили»

Создание связей для таблицы «Сотрудники» представлено на рисунке 23.

```
ALTER TABLE `employees` ADD FOREIGN KEY (`id_address`) REFERENCES `employees_ad-  
dress`(`id`) ON DELETE CASCADE ON UPDATE CASCADE;
```

```
ALTER TABLE `employeeess` ADD FOREIGN KEY (`id_auth`) REFERENCES `authoriza-  
tion`(`id`) ON DELETE CASCADE ON UPDATE CASCADE;
```

```
ALTER TABLE `employeeess` ADD FOREIGN KEY (`id_passport`) REFERENCES `employees_pass-  
port`(`id`) ON DELETE CASCADE ON UPDATE CASCADE;
```

Рисунок 23 – Создание связей для таблицы «Сотрудники»

Заполнение таблиц происходит с помощью оператора INSERT языка SQL. Общая команда для заполнения таблиц представлена на рисунке 24:

```
INSERT INTO название_таблицы (название_столбца_1, название_столбца_2, название_столбца_3,
...)  
VALUES (значение1, значение2, значение3, ...);
```

Рисунок 24 – Общая команда для заполнения таблиц

Примеры частичного заполнения таблиц показаны на рисунках 25–36. Заполнения таблицы «Адреса контактов» представлено на рисунке 25.

```
INSERT INTO `contacts_address` (`id`, `city`, `street`, `house`)  
VALUES (NULL, 'Минск', 'Машерова', '76'),  
        (NULL, 'Минск', 'Долгиновский тракт', '51'),  
        (NULL, 'Минск', 'Партизанский проспект', '91')
```

Рисунок 25 – Заполнение таблицы «Адреса контактов»

Заполнения таблицы «Контакты» представлено на рисунке 26.

```
INSERT INTO `contacts` (`id`, `name`, `phone`, `email`, `work_time`, `id_address`)  
VALUES (NULL, 'Отдел продаж автомобиля', '+375294572912', 'info@autoidea.by', 'Пн-Вс: 8:00 -  
20:00', '3'), (NULL, 'Отдел сервисного обслуживания', '+37529500081', 'service@autoidea.by', 'Пн-Вс: 8:00 -  
20:00', '5')
```

Рисунок 26 – Заполнение таблицы «Контакты»

Заполнения таблицы «Модели» представлено на рисунке 27.

```
INSERT INTO `models` (`id`, `model`, `main_foto`, `counts`)  
VALUES (NULL, 'X', '1681385693_X.jpg', '0'),  
        (NULL, 'M', '1681385756_M.jpg', '0')
```

Рисунок 27 – Заполнение таблицы «Модели»

Заполнения таблицы «Автомобили» представлено на рисунке 28.

```
INSERT INTO `auto` (`id`, `name`, `engine`, `year`, `price`, `color`, `complexion`, `img`, `status`, `state`, `id_model`)
VALUES
(NULL, 'X', 'Бензиновый', '2021', '110000', 'Красный', 'Базвая', '1681642646_iX.webp', '1', 'Новое', '65'),
(NULL, 'X
M60', 'Бензиновый', '2023', '160000', 'Тифани', 'Базвая', '1681642715_iX_M60.webp', '1', 'Новое', '65')
```

Рисунок 28 – Заполнение таблицы «Автомобили»

Заполнения таблицы «Авторизации» представлено на рисунке 29.

```
INSERT INTO `authorization` (`id`, `login`, `password`, `access`, `role`, `date_regist`, `email`)
VALUES
(NULL, 'andrei', '$2y$10$QtGuxF0YC14vvM4w1LEWDOo-
DoS.Wup7uF5e4BNy3fCRf7E3Oqg7bi', '1', '0', CURRENT_TIMESTAMP, 'andrei@mail.ru'),
(NULL, 'sava', '$2y$10$7.Cu7pQPylz86clTjUxKie/sWMS66u95d35ehgLJq16YmFh-
vAhtFS', '1', '1', CURRENT_TIMESTAMP, 'sava@mail.ru')
```

Рисунок 29 – Заполнение таблицы «Авторизации»

Заполнения таблицы «Заказы» представлено на рисунке 30.

```
INSERT INTO `orders` (`id`, `date`, `id_client`, `id_employee`, `id_auto`, `id_contact`)
VALUES (NULL, CURRENT_TIMESTAMP, '87', '11', '131', '20'),
(NULL, CURRENT_TIMESTAMP, '88', '10', '107', '20')
```

Рисунок 30 – Заполнение таблицы «Заказы»

Заполнения таблицы «Клиенты» представлено на рисунке 31.

```
INSERT INTO `clients` (`id`, `last_name`, `first_name`, `surname`, `date_birth`, `phone`, `count_orders`, `id_address`, `id_auth`, `id_passport`)
VALUES (NULL, 'Шемерей', 'Андрей', 'Алексеевич', '2003-01-01', '80295454546', '0', '53', '137', '49'),
(NULL, 'Рощупкин', 'Артур', 'Максимович', '2003-09-04', '80295671321', '0', '54', '139', '51')
```

Рисунок 31 – Заполнение таблицы «Клиенты»

Заполнения таблицы «Адреса клиентов» представлено на рисунке 32.

```
INSERT INTO `clients_address` (`id`, `city`, `street`, `house`, `apartment`)
VALUES (NULL, 'Ивацевичи', 'Набережная', '21', '21'),
       (NULL, 'Жлобин', 'Багратиона', '23', '1')
```

Рисунок 32 – Заполнение таблицы «Адреса клиентов»

Заполнение таблицы «Паспортные данные клиентов» представлено на рисунке 33.

```
INSERT INTO `clients_passport` (`id`, `series`, `number`, `issued_by`)
VALUES (NULL, 'MP', '2020201', 'Ивацевичским РОВД'),
       (NULL, 'МК', '6567678', 'Жлобинским РОВ')
```

Рисунок 33 – Заполнение таблицы «Паспортные данные клиентов»

Заполнение таблицы «Сотрудники» представлено на рисунке 34.

```
INSERT INTO `employees` (`id`, `last_name`, `first_name`, `surname`, `phone`, `date_birth`,
`job`, `id_address`, `id_auth`, `id_passport`)
VALUES (NULL, 'Чернова', 'Яна', 'Викторовна', '80338541948', '2005-05-
04' 'Менеджер', '23', '142', '32'),
       (NULL, 'Титов', 'Игорь', 'Владимирович', '80294443434', '2004-01-31', 'Ад-
мин', '21', '140', '29')
```

Рисунок 34 – Заполнение таблицы «Сотрудники»

Заполнение таблицы «Адреса сотрудников» показано на рисунке 35.

```
INSERT INTO `employees_address` (`id`, `city`, `street`, `house`, `apartment`)
VALUES (NULL, 'Калинковичи', 'Спортивная', '34', '31'),
       (NULL, 'Барановичи', 'Строителей', '76', '75')
```

Рисунок 35 – Заполнение таблицы «Адреса сотрудников»

Заполнение таблицы «Паспортные данные сотрудников» представлено на рисунке 36.

```
INSERT INTO `employees_passport` (`id`, `series`, `number`, `issued_by`)
VALUES (NULL, 'МО', '3892044', 'Калинковичским РОВД'),
       (NULL, 'МС', '2030405', 'Солигорским РОВД')
```

Рисунок 36 – Заполнение таблицы «Паспортные данные сотрудников»

3.2.2 Создание представлений

Представление (VIEW) – объект базы данных, являющийся результатом выполнения запроса к базе данных, определенного с помощью оператора SELECT, в момент обращения к представлению.

Представления иногда называют «виртуальными таблицами». Такое название связано с тем, что представление доступно для пользователя как таблица, но само оно не содержит данных, а извлекает их из таблиц в момент обращения к нему. Если данные изменены в базовой таблице, то пользователь получит актуальные данные при обращении к представлению, использующему данную таблицу; кэширования результатов выборки из таблицы при работе представлений не производится. При этом, механизм кэширования запросов работает на уровне запросов пользователя безотносительно к тому, обращается ли пользователь к таблицам или представлениям.

Представления могут основываться как на таблицах, так и на других представлениях, т.е. могут быть вложенными (до 32 уровней вложенности).

Общая команда создания представления представлена на рисунке 37:

```
CREATE VIEW название_представления
AS SELECT столбцы_таблиц
FROM таблицы
WHERE условие;
```

Рисунок 37 – Общая команда создания представления

В данной работе были созданы следующие представления:

1. Представления с запросом на выборку клиентов.
2. Представления с запросом на выборку сотрудников.
3. Представление с запросом на выборку заказов.
4. Представление с запросом на выборку автомобилей.
5. Представления с запросом о сотрудниках менеджерах.

В первом представлении демонстрируем полную информацию о каждом клиенте: ФИО клиента, дата рождения, номер телефона, фотография, город, улица, номер дома, номер квартиры, серия и номер паспорта, кем выдан паспорт, логин, роль, email, дата регистрации.

Создание представления обо всех клиентах представлено на рисунке 38.

```
CREATE VIEW clientsView AS
SELECT clients.id, last_name, first_name, surname, date_birth, phone, city, street, house, apartment, number, series, issued_by, login, access, role, email, date_regist
FROM clients INNER JOIN clients_passport ON clients.id_passport = clients_passport.id
INNER JOIN clients_address ON clients.id_address = clients_address.id
INNER JOIN authorization ON clients.id_auth = authorization.id;
```

Рисунок 38 – Создания представления «Клиенты»

Результат выполнения представлен на рисунке 39.

	id	last_name	first_name	surname	date_birth	count_orders	phone	city	street	house	apartment	number	series	issued_by	login	acc
86	Шемерей	Андрей	Олегович	2003-01-01	3	80295454546	Ивацевич	Набережная	21	21	2020201	МР	Ивацевичским	andrei	РОВД	
87	Соболь	Георгий	Николаевич	2004-04-04	4	80448102123	Жлобин	Багратиона	23	1	6567678	МК	Жлобинским	gosha	РОВ	

Рисунок 39 – Результат создания представления «clientsview»

Во втором представлении храниться полная информация о каждом сотруднике: ФИО сотрудника, дата рождения, номер телефона, фотография, должность, город, улица, номер дома, номер квартиры, серия и номер паспорта, кем выдан паспорт, логин, роль, email, дата регистрации.

Создание представления обо всех сотрудниках представлено на рисунке 40.

```
CREATE VIEW
employeesView AS
SELECT last_name, first_name, surname, date_birth, phone, job, city, street, house, apartment, number, series, issued_by, issued_when, validity, login, access, role, email, date_regist
FROM employees INNER JOIN employees_passport ON employees.id_passport = employees_passport.id
INNER JOIN employees_address ON employees.id_address = employees_address.id
INNER JOIN authorization ON employees.id_auth = authorization.id;
```

Рисунок 40 – Создания представления «Сотрудники»

Результат выполнения представлен на рисунке 41.

id	last_name	first_name	surname	date_birth	phone	job	city	street	house	apartment	number	series	issued_by	login	access
12	Чернова	Яна	Викторовна	2005-05-04	80338541948	Админ	Могилев	Новицкого	82	21	2626343	MP	Жлобинским РОВ	yana	
20	Соколова	Кристина	Юрьевна	1986-05-04		Админ	Столбцы	Можевского	12	2	8023721	МК	Столбцовским РОВД	sokol	
21	Мухопад	Даниил	Михайлович	1998-03-01	80445345657	Менеджер	Минск	Партизанский проспект	91	98	3467291	МК	Минским РОВД	mukha	

Рисунок 41 – Результат создания представления «employeesview»

В третьем представлении храниться полная информация о каждом заказе: дата оформления заказа, стоимость, название автомобиля, модель автомобиля, ФИО клиента, номер телефона и email клиента.

Создание представления обо всей информации каждого заказа представлено на рисунке 42.

```
CREATE VIEW ordersView AS
SELECT orders.id, orders.date, price, name, model, clients.last_name AS client_last_name,
CREATE VIEW ordersView AS
SELECT orders.id,
orders.id_client AS id_client,
orders.date, price, name, model,
clients.last_name AS client_last_name,
clients.first_name AS client_first_name,
clients.surname AS client_surname,
clients.phone AS phone,
authorization.email
FROM orders INNER JOIN clients ON orders.id_client = clients.id
INNER JOIN auto ON orders.id_auto = auto.id
INNER JOIN authorization ON authorization.id = clients.id_auth
INNER JOIN models ON auto.id_model = models.id ;
```

Рисунок 42 – Создания представления «Заказы»

Результат выполнения представлен на рисунке 43.

id	id_client	date	price	name	model	client_last_name	client_first_name	client_surname	phone	email
27	86	2023-04-27 18:52:25	90000	5	X	Шемерей	Андрей	Алексеевич	80295454546	andrei@mail.ru
21	87	2023-04-27 17:23:47	120000	60i xDrive	M	Соболев	Георгий	Николаевич	80448102123	gosha@mail.ru
25	87	2023-04-27 17:29:38	110000	7	X	Соболев	Георгий	Николаевич	80448102123	gosha@mail.ru
33	87	2023-04-27 19:29:04	110000	7	X	Соболев	Георгий	Николаевич	80448102123	gosha@mail.ru
23	88	2023-04-27 17:26:55	90000	5	X	Рощупкин	Артур	Максимович	80295671321	artur@mail.ru

Рисунок 43 – Результат создания представления «ordersview»

В четвертом представлении храниться полная информация о каждом автомобиле: модель, название, двигатель, год выпуска, стоимость, цвет, комплектация, фото, наличие и состояние.

Создание представления обо всей информации каждого автомобиля представлено на рисунке 44.

```
CREATE VIEW autosview AS
SELECT auto.id, model, name, `engine`, `year`, price, color, complexion, img, `status`, state
FROM auto INNER JOIN models ON auto.id_model = models.id;
```

Рисунок 44 – Создание представления «Автомобилей»

Результат выполнения представлен на рисунке 45.

<u>id</u>	<u>model</u>	<u>name</u>	<u>engine</u>	<u>year</u>	<u>price</u>	<u>color</u>	<u>complexion</u>	<u>img</u>	<u>status</u>	<u>state</u>
102	X	7	Бензиновый	2023	110000	Серый	Базовая	1681643228_X7.webp	1	Новое
104	X	6	Электрический	2020	95000	Голубой	Полная	1681643316_X6.webp	0	Новое
105	X	6	Бензиновый	2020	100000	Болотный	Полная	1681643351_X6_BROWN.webp	1	Новое
106	X	5	Бензиновый	2019	85000	Серебристый	Базовая	1681643387_X5.webp	0	Новое
107	X	5	Бензиновый	2022	90000	Белый	Средняя	1681643417_X5_WHITE.webp	1	Бу
108	X	4	Бензиновый	2023	105000	Серебристый	Полная	1681643450_X4.webp	0	Новое

Рисунок 45 – Результат создания представления «autosview»

В пятом представлении соберем информацию обо всех менеджерах автосалона, для оформления заказов. Представление содержит: ФИО, номер телефона и его количество заказов.

Создание представления обо всех менеджерах автосалона представлено на рисунке 46.

```
CREATE VIEW managersview
AS SELECT employees.id, last_name, first_name, surname, phone, email, count_orders
FROM employees INNER JOIN authorization ON employees.id_auth = authorization.id
WHERE job = 'Менеджер';
```

Рисунок 46 – Создание представления «Менеджеры»

Результат выполнения представлен на рисунке 47.

<u>id</u>	<u>last_name</u>	<u>first_name</u>	<u>surname</u>	<u>phone</u>	<u>email</u>	<u>count_orders</u>
10	Титов	Игорь	Владимирович	80294443434	igor@mail.ru	1
12	Чернова	Яна	Викторовна	80338541948	yana@mail.ru	3

Рисунок 47 – Результат создания представления «managersview»

3.2.3 Назначение прав доступа

Рано или поздно, в любой многопользовательской базе данных возникает вопрос о разграничении прав на различные объекты, которые хранятся в этой базе. Причем разновидности этих прав могут быть самыми разными.

Вариант, когда пользователю необходимо дать право на изменение только заданного атрибута, у некоторых объектов, лишь на какие-то допустимые значения, может выглядеть вполне заурядным.

Поэтому, необходимо иметь достаточно гибкую систему прав, позволяющую легко раздавать пользователям и группам самые разнообразные права на любые объекты.

Для создания нового пользователя базы данных используют команду представленную на рисунке 48.

```
CREATE USER 'имя_пользователя'@'localhost'  
IDENTIFIED BY 'пароль';
```

Рисунок 48 – Общая команда создания пользователя

Создадим трех пользователей для наглядности: boss, admin и manager.

Boss – это самый босс нашего автосалона, admin – это главный администратор, а manager – главный менеджер автосалона. В дальнейшем мы подробно рассмотрим их полномочия и назначим соответствующие права доступа.

Код создания всех трех пользователей представлен на рисунке 49.

```
CREATE USER 'boss'@'localhost' IDENTIFIED BY 'boss';  
CREATE USER 'admin'@'localhost' IDENTIFIED BY 'admin';  
CREATE USER 'manager'@'localhost' IDENTIFIED BY 'manager';
```

Рисунок 49 – Создание пользователей базы данных

Права доступа пользователей предоставляются с помощью команды GRANT:

GRANT права ON база_данных.таблица TO 'логин'@'localhost';

Права доступа могут быть следующие:

– ALL – дает полный доступ к базе данных. Кстати, если база данных не определена в команде, то данный запрос даст полный доступ ко всему в MySQL, что делать не рекомендуется;

- CREATE – позволяет пользователю создавать базы данных и таблицы;
- DELETE – дает право пользователю удалять строки из таблиц;
- DROP – дает право удалять базы данных и таблица целиком;
- EXECUTE – дает право пользователю выполнять хранимые процедуры;
- GRANT OPTION – с этой опцией пользователь сможет давать права (или удалять) другим пользователям;
- INSERT – это право на добавление новых строк в таблицу;
- SELECT – самое распространенное право – парсить (извлекать) данные из SQL для чтения;
- SHOW DATABASES – этому пользователю можно будет посмотреть на список баз данных;
- UPDATE – дает право пользователю изменять текущие строки в таблице;

Итак, назначим теперь права доступа трем нашим пользователям. У пользователя boss будут все полномочия, поскольку он на правах главного администратора может просматривать информацию об автомобилях, сотрудниках, клиентах, заказах, а также контакты автосалона.

Назначение прав доступа для босса автосалон показано на рисунке 50.

```
GRANT ALL
ON bmw_dealership.*
TO 'boss'@'localhost';
```

Рисунок 50 – Назначение прав доступа для босса автосалона

Далее рассмотрим пользователя admin: человек с такой должностью ведет оперативный о контактной информации, а также ведет учет всех сотрудников и клиентах автосалона. Поэтому у пользователя admin должен быть доступ ко всем таблицам, но без возможности удаления таблиц и базы данных, а также создания новых сотрудников.

Назначение прав доступа для главного администратора представлено на рисунке 51.

```
GRANT SELECT, INSERT, DELETE, UPDATE
ON bmw_dealership*
to 'admin'@'localhost';
```

Рисунок 51 – Назначение прав доступа для главного администратора

И, наконец, пришла очередь менеджера. В его обязанности входит продажа автомобилей. Это значит, что человек с такой должностью должен иметь право просматривать, менять, заполнять и удалять записи из таблиц заказы, модели и автомобили.

Назначение прав доступа для менеджера представлено на рисунке 52.

```
GRANT SELECT
ON autosview
TO 'manager'@'localhost';
GRANT SELECT
ON ordersview
TO 'manager'@'localhost';
GRANT INSERT,SELECT,UPDATE, DELETE
ON auto
TO 'manager'@'localhost';
GRANT INSERT,SELECT,UPDATE, DELETE
ON models
TO 'manager'@'localhost';
GRANT INSERT,SELECT,UPDATE, DELETE
ON orders
TO 'manager'@'localhost';
```

Рисунок 52 – Назначение прав доступа для менеджера

Итоговая таблица пользователей представлена на рисунке 53.

Host	User	Select_priv	Insert_priv	Update_priv	Delete_priv	Create_priv	Drop_priv	Reload_priv	Shutdown_priv	Process_priv	File_priv	Grant_priv	References_priv
%	root	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
localhost	admin	N	N	N	N	N	N	N	N	N	N	N	N
localhost	boss	N	N	N	N	N	N	N	N	N	N	N	N
localhost	manager	N	N	N	N	N	N	N	N	N	N	N	N
localhost	mysql.infoschema	Y	N	N	N	N	N	N	N	N	N	N	N

Рисунок 53 – Итоговая таблица пользователей

3.2.4 Создание индексов

Индекс – объект базы данных, создаваемый с целью повышения производительности поиска данных. Таблицы в базе данных могут иметь большое количество строк, которые хранятся в произвольном порядке, и их поиск по заданному критерию путём последовательного просмотра таблицы строка за строкой может занимать много времени. Для повышения производительности поиска данных могут быть проиндексированы некоторые столбцы таблиц.

Для создания индекса в таблице используется следующая команда (не допускаются повторяющиеся значения) представленная на рисунке 54.

```
CREATE UNIQUE INDEX index_name  
ON table_name (column1, column2, ...);
```

Рисунок 54 – Общая команда создания индексов

Типы MySQL индексов:

PRIMARY KEY - Первичный ключ: основной ключ, который позволяет хранить свои значения как уникальные записи таблицы. Данный тип ключа должен присутствовать в таблице в единственном экземпляре. Обычно данный тип ключа определяют колонке с наименованием id.

UNIQUE - Уникальный ключ: Частично похож на первичный ключ, за счет того, что значения колонки таблицы должны быть уникальными (не должны повторяться) и не должны быть равны NULL.

В работе были проиндексированы следующие поля:

1. Фамилия клиента в таблице «Clients». Код индексирования представления представлен на рисунке 55:

```
CREATE INDEX last_name  
ON clients(last_name)
```

Рисунок 55 – Код индексирования в таблице «Clients»

2. Фамилия сотрудника в таблице «Employees». Код индексирования представлен на рисунке 56:

```
CREATE INDEX last_name  
ON employees(last_name)
```

Рисунок 56 – Код индексирования в таблице «Employees»

3. Цена за авто в таблице «Auto». Код индексирования представлен на рисунке 57:

```
CREATE INDEX price  
ON auto(price)
```

Рисунок 57 – Код индексирования в таблице «Auto»

4. Количество автомобилей данной модели в таблице «Models». Код индексирования представлен на рисунке 58:

```
CREATE INDEX model  
ON models(model)
```

Рисунок 58 – Код индексирования в таблице «Models»

3.2.5 Создание триггеров

Для полноценного функционирования базы данных были созданы несколько триггеров.

Создание триггера «insert_auto_update_counts». При добавлении нового автомобиля в таблицу «auto», исходя из его модели в таблице «models» в поле соответствующей модели, в столбце «counts» прибавляется +1. Код создания триггера «insert_auto_update_counts» представлен на рисунке 59.

```
CREATE TRIGGER insert_auto_update_count  
AFTER INSERT ON auto  
FOR EACH ROW  
UPDATE models  
SET models.counts = models.counts + 1  
WHERE models.id = NEW.id_model
```

Рисунок 59 – Код создания триггера «insert_auto_update_counts»

Создание триггера «delete_auto_update_counts». При удалении автомобиля из таблицы «auto», исходя из его модели в таблице «models» в поле соответствующей модели, в столбце «counts» отнимается -1. Код создания триггера «delete_auto_update_counts» представлен на рисунке 60.

```
CREATE TRIGGER delete_auto_update_count  
AFTER DELETE ON auto  
FOR EACH ROW UPDATE models  
SET models.counts = models.counts - 1  
WHERE models.id = OLD.id_model
```

Рисунок 60 – Код создания триггера «delete_auto_update_counts»

Создание триггера «delete_auto_update_counts». При удалении автомобиля из таблицы «auto», исходя из его модели в таблице «models» в поле соответствующей модели, в столбце «counts» отнимается -1. Код создания триггера «delete_auto_update_counts» представлен на рисунке 61.

```
CREATE TRIGGER delete_auto_update_count
AFTER DELETE ON auto
FOR EACH ROW
UPDATE models
SET models.counts = models.counts - 1
WHERE models.id = OLD.id_model
```

Рисунок 61 – Код создания триггера «delete_auto_update_counts»

Создание триггера «insert_update_price». При добавлении нового автомобиля с фиксированной ценой, к цене автоматически добавляется 3000 как налог. Код создания триггера «insert_update_counts» представлен на рисунке 62.

```
CREATE TRIGGER insert_update_price
BEFORE INSERT ON auto
FOR EACH ROW
SET NEW.price = NEW.price + 3000
```

Рисунок 62 – Код создания триггера «insert_update_counts»

Создание триггера «update_update_price». При добавлении нового автомобиля с фиксированной ценой, к цене автоматически добавляется 3000 как налог. Код создания триггера «update_update_counts» представлен на рисунке 63.

```
CREATE TRIGGER update_update_price
BEFORE UPDATE ON auto
FOR EACH ROW
SET NEW.price = NEW.price + 3000
```

Рисунок 63 – Код создания триггера «update_update_counts»

Создадим триггеры «delete_client_address», «delete_client_passport», «delete_client_auth» для удаления всех данных о клиенте, чтобы при удалении клиента, также удалялись его адресные данные, паспортные данные и данные для авторизации. Код создания триггеров представлен на рисунках 64–66.

```
CREATE TRIGGER delete_client_address
BEFORE DELETE
ON clients
FOR EACH ROW
DELETE FROM clients_address
WHERE id = OLD.id_address
```

Рисунок 64 – Код создания триггера «delete_client_address»

```
CREATE TRIGGER delete_client_passport
BEFORE DELETE
ON clients
FOR EACH ROW
DELETE FROM clients_passport
WHERE id = OLD.id_passport
```

Рисунок 65 – Код создания триггера «delete_client_passport»

```
CREATE TRIGGER delete_client_auth
BEFORE DELETE
ON clients
FOR EACH ROW
DELETE FROM authorization
WHERE id = OLD.id_auth
```

Рисунок 66 – Код создания триггера «delete_client_auth»

Создадим триггеры «delete_employee_address», «delete_employee_passport», «delete_employee_auth» для удаления всех данных о клиенте, чтобы при удалении клиента, также удалялись его адресные данные, паспортные данные и данные для авторизации. Код создания триггеров представлен на рисунках 67–69.

```
CREATE TRIGGER delete_employee_address
BEFORE DELETE ON employees
FOR EACH ROW
DELETE FROM employees_address
WHERE id = OLD.id_address
```

Рисунок 67 – Код создания триггера «delete_employee_address»

```
CREATE TRIGGER delete_employee_passport
BEFORE DELETE ON employees
FOR EACH ROW
DELETE FROM employees_passport
WHERE id = OLD.id_passport
```

Рисунок 68 – Код создания триггера «delete_employee_passport»

```
CREATE TRIGGER delete_employee_auth
BEFORE DELETE ON employees
FOR EACH ROW
DELETE FROM authorization
WHERE id = OLD.id_auth
```

Рисунок 69 – Код создания триггера «delete_employee_auth»

Создадим триггер «delete_contact_address» для удаления контактных данных, чтобы при удалении контакты, также удалялись его адресные данные. Код создания триггера представлен на рисунках 70.

```
CREATE TRIGGER delete_contact_address
BEFORE DELETE ON contacts
FOR EACH ROW
DELETE FROM contacts_address
WHERE id = OLD.id_address
```

Рисунок 70 – Код создания триггера «delete_contact_address»

3.3 Структура приложения

У многих организаций есть свой веб-сайт, с помощью которого сотрудники получают доступ к базам данных своей компании, а администрация может легко и удобно работать с информацией о сотрудниках. В нашем случае веб-сайт будет создан на языке программирования PHP, а также с использованием HTML, SCSS, GULP, JavaScript с помощью средства разработки Visual Code.

Для начала создадим структуру проектных папок, для удобства работы с большим количеством файлов. Затем настроим GULP-сборку, для оптимизации файлов и улучшения загрузки веб-сайта в будущем.

Напишем для начала frontend часть веб-сайта, для этого создадим необходимое количество файлов для каждой страницы, отрисуем разметку, пропишем стили и настроим визуальные эффекты с помощью JS.

Далее перейдем в backend часть. Для связи базы данных с веб-сайтом, запустим OpenServer и произведем подключение к базе данных. Код подключения к базе данных представлен на рисунке 71.

```
$driver = "mysql";
$host = "localhost";
$db_name = "bmvc_dealership";
$db_user = "root";
$db_pass = "";
$charset = "utf8";
$options = [PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC];

try {
    $pdo = new PDO(
        "$driver:host=$host; dbname=$db_name; charset=$charset",
        $db_user, $db_pass, $options
    );
} catch(PDOException $e) die("Ошибка подключения к базе данных");
```

Рисунок 71 – Подключение к базе данных

В классе Order описываются функции, которые будут срабатывать по определенным ссылкам при запуске приложения. Это такие функции как добавление заказа сотрудником, добавление заказа клиентом, удаление заказа. На рисунке 72 показан класс Order.

```
class Order {  
    public $NEW_AUTO = 19;  
    public $OLD_AUTO = 20;  
    public $CLIENT = 0;  
    public $NEW = 'Новое';  
    //Добавление заказа клиентом  
    public function addOrderClient(){...}  
    //Добавление заказа сотрудником  
    public function addOrderEmployee(){...}  
    //Удаление заказа  
    public function deleteOrder($id) {...}  
}
```

Рисунок 72 – Класс Order

В классе User описываются функции, которые будут срабатывать по определенным ссылкам при запуске приложения. Это такие функции как создание пользовательской сессии для авторизации, регистрация, авторизация, изменение пароля, редактирование персональных данных, удаление аккаунта, добавление заказа, удаление заказа. На рисунке 73 показан класс User.

```
class User {  
    public $ACCESS = 1;  
    public $NO_ACCESS = 0;  
    public $CLIENT = 0;  
    public $EMPLOYEE = 1;  
    //Создаем сессию для авторизации  
    public function userAuth($arr) {...}  
    //Регистрация  
    public function registration() {...}  
    //Авторизация  
    public function authorization() {...}  
    //Изменение пароля  
    public function editPassword($id) {...}  
    //Редактирование персональных данных  
    public function updateUser() {...}  
    //Удаление аккаунта  
    public function deleteUser($id) {...}  
    //Добавление заказа  
    public function addOrder() {...}  
}
```

Рисунок 73 – Класс User

В классе `Models` описываются функции, которые будут срабатывать по определенным ссылкам при запуске приложения. Это такие функции как добавление, удаление и редактирование модели. На рисунке 74 показан класс `Model`.

```
class Model {  
  
    //Добавление модели  
    public function addModel() {...}  
  
    //Редактирование модели  
    public function updateModel() {...}  
  
    //Удаление модели  
    public function deleteModel($id) {...}  
  
}
```

Рисунок 74 – Класс `Model`

В классе `Contact` описываются функции, которые будут срабатывать по определенным ссылкам при запуске приложения. Это такие функции как добавление, удаление и редактирование модели. На рисунке 75 показан класс `Contact`.

```
class Contact {  
  
    //Добавление контакта  
    public function addContact() {...}  
  
    //Редактирование контакта  
    public function updateContact() {...}  
  
    //Удаление контакта  
    public function deleteContact($id) {...}  
  
}
```

Рисунок 75 – Класс `Contact`

В классе Auto описываются функции, которые будут срабатывать по определенным ссылкам при запуске приложения. Это такие функции как добавление, удаление, редактирование модели и редактирование статуса модели. На рисунке 76 показан класс Auto.

```
class Auto {  
  
    public $AVAILABLE = 1;  
    public $NO_AVAILABLE = 0;  
  
    //Добавление авто  
    public function addAuto() {...}  
  
    //Редактирование авто  
    public function updateAuto() {...}  
  
    //Редактирование статуса авто  
    public function updateStatusAuto($id) {...}  
  
    //Удаление авто  
    public function deleteAuto($id) {...}  
  
}
```

Рисунок 76 – Класс Auto

В классе Client описываются функции, которые будут срабатывать по определенным ссылкам при запуске приложения. Это такие функции как добавление, удаление, редактирование клиента и редактирование доступа клиента. На рисунке 77 показан класс Client.

```
class Client {  
    public $ACCESS = 1;  
    public $NO_ACCESS = 0;  
    public $CLIENT = 0;  
    public $errMsg = [];  
    //Добавление клиента  
    public function addClient() {...}  
    //Редактирование клиента  
    public function updateClient() {...}  
    //Редактирование доступа клиента  
    public function updateStatusClient($id) {...}  
    //Удаление клиента  
    public function deleteClient($id) {...}  
}
```

Рисунок 77 – Класс Client

В классе Employee описываются функции, которые будут срабатывать по определенным ссылкам при запуске приложения. Это такие функции как добавление, удаление, редактирование сотрудника и редактирование доступа сотрудника. На рисунке 78 показан класс Employee.

```
class Employee {  
  
    public $ACCESS = 1;  
    public $NO_ACCESS = 0;  
    public $ADMIN = 1;  
    public $errMsg = [];  
  
    //Добавление сотрудника  
    public function addEmployee() {...}  
  
    //Редактирование сотрудника  
    public function updateEmployee() {...}  
  
    //Редактирование доступа сотрудника  
    public function updateStatusEmployee($id) {...}  
  
    //Удаление сотрудника  
    public function deleteEmployee($id) {...}  
  
}
```

Рисунок 78 – Класс Employee

После создания классов и файлов .php для остальных отношений БД, приложение готово к запуску. Всякий запуск приложения начинается с авторизации.

Функция авторизации пользователя заключается в проверке существования в БД пользователей введенного email и соответствующего ему пароля. Также авторизация пользователя подразумевает получение его роли из отношения User.

После успешной авторизации пользователя создаётся пользовательская сессия с соответствующими привилегиями, согласно роли пользователя.

4 ТЕСТИРОВАНИЕ

Рассмотрим несколько исключительных ситуаций, показанных на рисунках 79–86, которые могут быть вызваны при некорректном вводе данных.

Рассмотрим случаи, когда пользователь которого нет в базе данных, пытается войти. Результат обработки данной ситуации показан на рисунках 87–96.

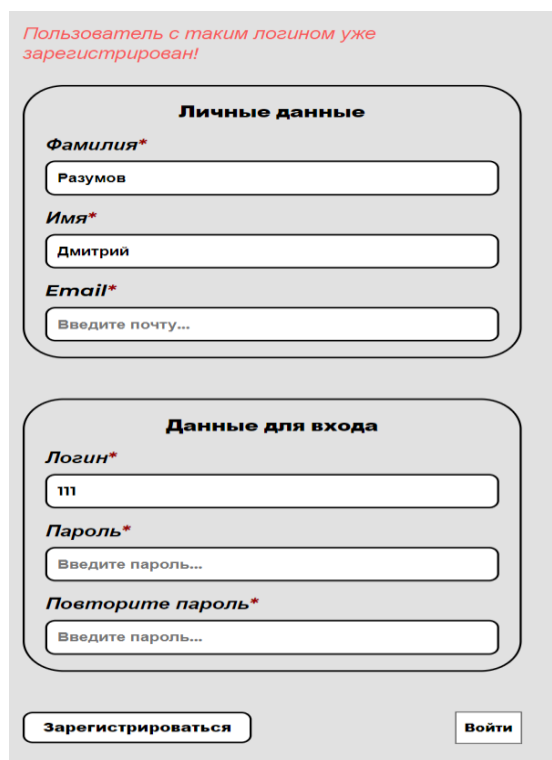
The screenshot shows a login form titled "АВТОРИЗАЦИЯ". Below the title is a link: "У вас еще нет аккаунта? Вы можете зарегистрироваться [здесь](#)". There are two input fields: "Email (при регистрации)" containing "1@mail.ru" and "Пароль" containing "Введите пароль...". Below the password field is a tooltip with an orange exclamation mark icon and the text "Заполните это поле.". At the bottom are two buttons: "Войти" and "Зарегистрироваться".

Рисунок 79 – Пример входа, когда не все поля заполнены

The screenshot shows the same login form as Figure 79, but with an error message: "Неправильный логин или пароль!" in red text above the input fields. The "Email (при регистрации)" field contains "111@mail.ru" and the "Пароль" field contains "Введите пароль...". The buttons "Войти" and "Зарегистрироваться" are still present at the bottom.

Рисунок 80 – Пример входа, когда неправильно введен логин или пароль

Рассмотрим случай, когда пользователь пытается зарегистрироваться. Результат обработки данной ситуации показан на рисунках 81–82.



Пользователь с таким логином уже зарегистрирован!

Личные данные

Фамилия*
Разумов

Имя*
Дмитрий

Email*
Введите почту...

Данные для входа

Логин*
m

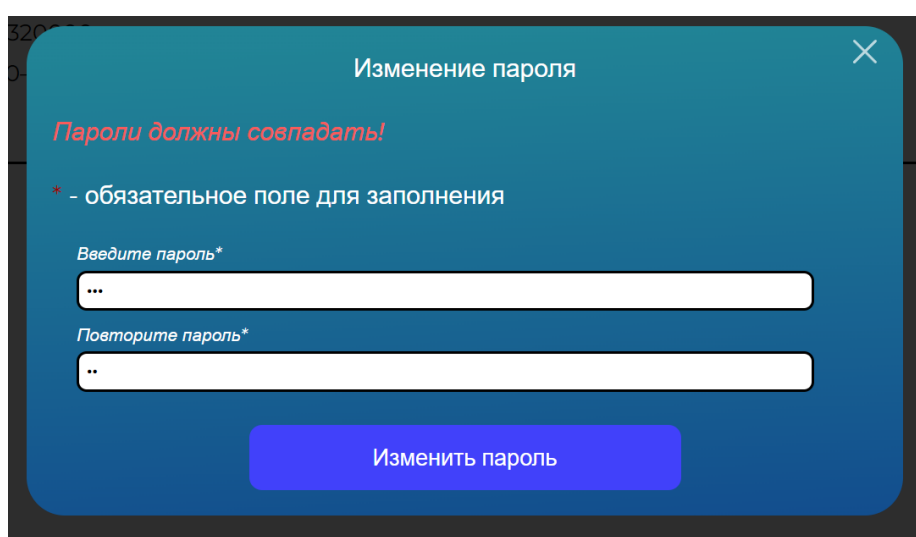
Пароль*
Введите пароль...

Повторите пароль*
Введите пароль...

Зарегистрироваться Войти

Рисунок 81 – Пример регистрации, когда пользователь с таким логином уже существует в базе данных

Рассмотрим случай, когда пользователь меняет пароль, но пароли не совпадают. Результат обработки данной ситуации представлен на рисунке 82.



Изменение пароля

Пароли должны совпадать!

* - обязательное поле для заполнения

Введите пароль*
...

Повторите пароль*
..

Изменить пароль

Рисунок 82 – Пример, когда пользователь меняет пароль

Пример поиска автомобиля, когда по данному запросу ничего не найдено. Результат обработки данной ситуации представлен на рисунке 83.

The screenshot shows a web form titled "Выберите свой автомобиль" (Choose your car). Below the title is a section "Подберите авто по вашим запросам" (Select car by your requests) containing various filters: "Название" (Name) with a dropdown set to "7", "Цвет" (Color) with a dropdown set to "Тифани" (Tiffany), "Комплектация" (Configuration) with a dropdown set to "Базовая" (Basic), "Наличие" (Availability) with a checkbox "Есть" (Yes) checked, "Двигатель" (Engine) with checkboxes for "Бензиновый" (Gasoline) and "Электрический" (Electric), "Состояние" (Condition) with checkboxes for "Новое" (New) and "Бу" (Used), and "Год выпуска" (Year of release) with "От" (From) and "До" (To) dropdowns. There is also a "Цена" (Price) section with "От" and "До" dropdowns. A blue "Найти" (Find) button is at the bottom right. Below the form is a breadcrumb trail: "ALL > X > M > 8 > 7 > 5 > 4 > 3 > 2 > 1 > Z4". At the bottom, a red message reads: "По данному поиску ничего не найдено! Повторите поиск!" (No results found for this search! Repeat the search!).

Рисунок 83 – Пример неудачного поиска

Пример неудачного поиска в админ панели. Результат обработки данной ситуации представлен на рисунке 84.

The screenshot shows an admin panel interface. On the left is a sidebar with three buttons: "Клиенты" (Clients), "Сотрудники" (Employees), and "Контакты" (Contacts). The "Клиенты" button is highlighted. In the main area, there is a green "Добавить" (Add) button and a search input field containing the text "Дима". Below the search field, a red message reads: "По данному поиску ничего не найдено! Повторите поиск!" (No results found for this search! Repeat the search!).

Рисунок 84 – Пример неудачного поиска в админ панели

Пример добавления клиента или сотрудника при вводе логина или email, которые уже имеются в базе данных. Результат обработки данной ситуации представлен на рисунке 85.

The screenshot shows a registration form titled "Добавление клиента" (Add client). On the left is a sidebar with three buttons: "Клиенты" (Clients), "Сотрудники" (Employees), and "Контакты" (Contacts). The "Клиенты" button is highlighted. In the main area, there is a red message: "Пользователь с таким логином уже зарегистрирован!" (User with this login is already registered!). Below this message is a note: "* - обязательное поле для заполнения" (* - required field for filling). The form itself is titled "Личные данные" (Personal data) and contains two input fields: "Фамилия*" (Surname*) and "Имя*" (Name*), both containing the letter "1".

Рисунок 85 – Пример неудачного добавления пользователя

Пример заполнения каких-либо данных, когда поле является обязательным, но оно не заполнено. Результат обработки данной ситуации представлен на рисунке 86.

Добавление заказа

* - обязательное поле для заполнения

Автомобиль*

105 - 6

Клиент*

Выберите клиента:

Выберите один из пунктов списка.

Добавить

Рисунок 86 – Пример незаполненного обязательного поля

Если у пользователя не заполнены все персональные данные, то он не может совершить заказ, до тех пор, пока не заполнит всю личные данные. Результат обработки данной ситуации представлен на рисунке 87.

Оформление заказа на автомобиль **BMW X серии 7**

Ваши персональные данные:

ФИО: Чернова Яна Викторовна
Дата рождения: 2005-05-04
Номер телефона: 80338541948
Логин: yana
Email: yana@mail.ru
Место жительства: г.Могилев, ул., д., кв.21
Серия и номер паспорта:
Кем выдан:

Данные автомобиля:

Модель: BMW X серии 7
Коробка передач: Бензиновый
Цвет: Серый
Год выпуска: 2023
Состояние: Новое
Комплектация: Базовая

СТОИМОСТЬ: 110000

Вы не можете оформить заказ, так как у вас введены не все личные данные. Перейдите в личный кабинет и отредактируйте персональные данные

Рисунок 87 – Пример оформления заказа, когда заполнены не все персональные данные

Далее рассмотрим несколько ситуаций работы веб-приложения в обычном режиме, представленных на рисунках 88–96.

Рассмотрим, как реализовано редактирование данных клиента. Данные таблицы клиенты показано на рисунке 88.

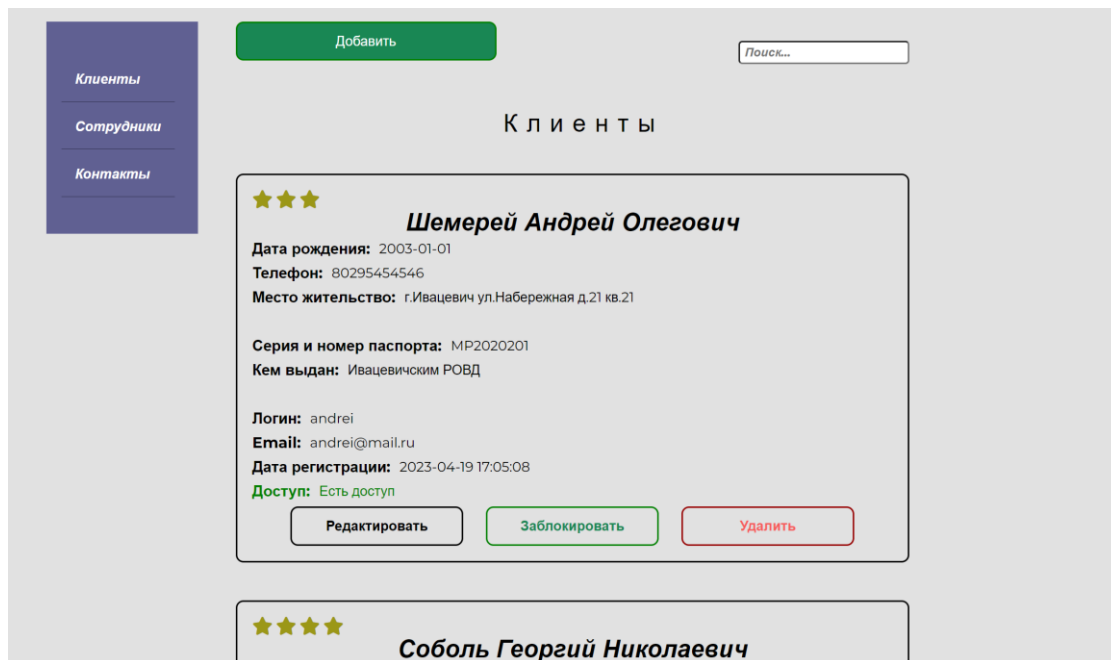


Рисунок 88 – Пример вывода клиентов

После перехода на данную страницу пользователь выбирает клиента и нажимает кнопку «Редактировать». После нажатия переходим на страницу редактирования данного клиента, показанную на рисунке 89.

Рисунок 89 – Редактирование данных о клиенте

После перехода на данную страницу в полях ввода по умолчанию стоят старые данные, пользователь выбирает, что хочет изменить и нажимает кнопку «Сохранить». Изменим, например, отчество. Результат показан на рисунке 90.

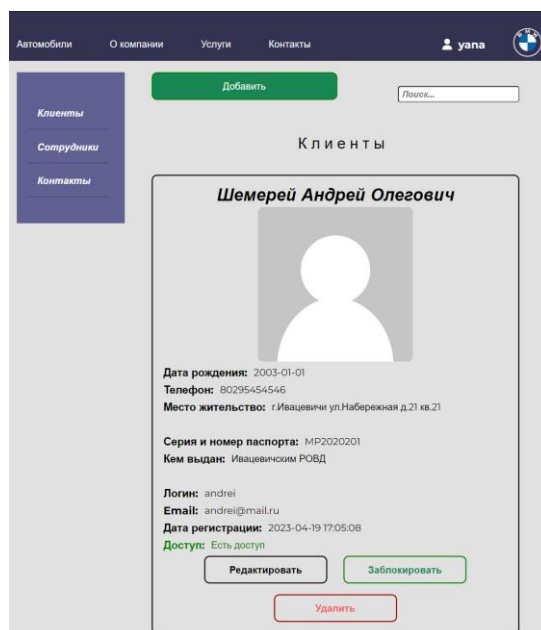


Рисунок 90 – Результат редактирования данных о клиенте

Аналогичный вывод подробной информации о работниках, контактах, моделях, авто, заказах, но в заказах не предусмотрено редактирование.

Рассмотрим, как реализовано оформление заказа клиентов. Данные таблицы автомобили показано на рисунке 91.

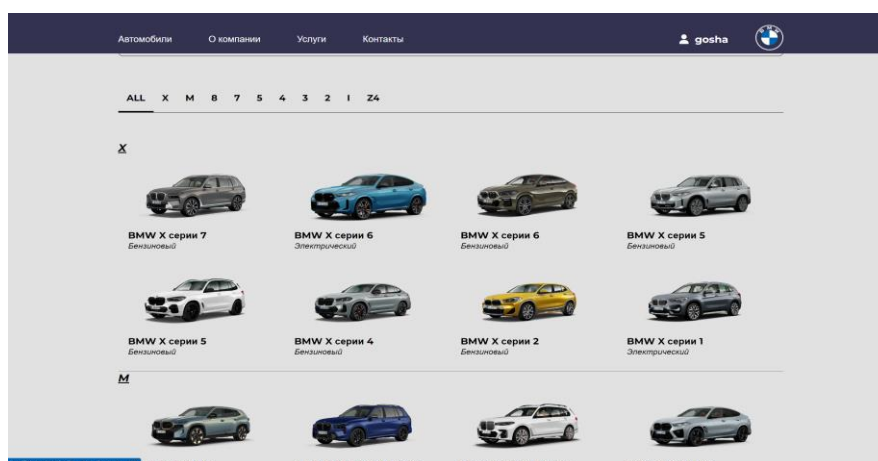


Рисунок 91 – Пример вывода таблицы автомобилей

При выборе какого-либо автомобиля мы переходим на персональную страницу этого автомобиля, где находится главное фото данной марки авто, данные о данном автомобиле, маленькое фото, и кнопки для оформления заказа. Пример персональной страницы автомобиля показан на рисунке 92.

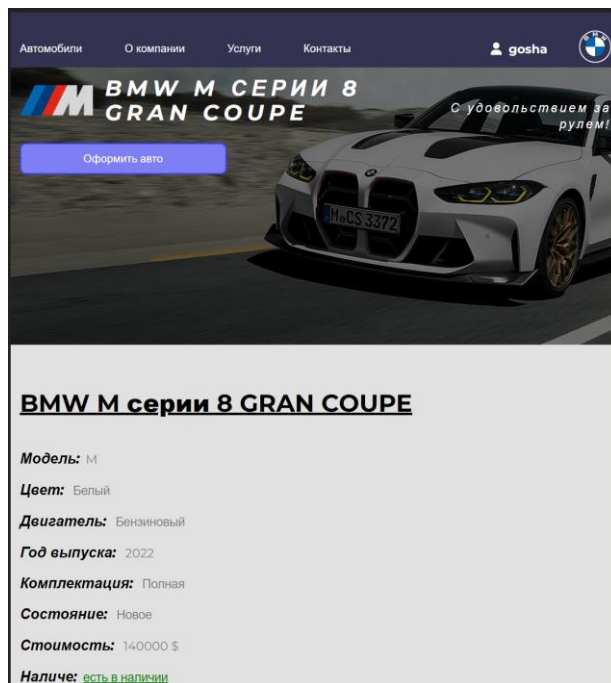


Рисунок 92 – Пример вывода персональной страницы каждого автомобиля

При нажатии на кнопку «Оформить заказ», всплывает окно для оформления заказа, показанное на рисунке 93.

Оформление заказа на автомобиль **BMW M серии 8 GRAN COUPE**

Серия и номер паспорта: MK6567678
Кем выдан: Жлобинским РОВ

Данные автомобиля:
Модель: BMW M серии 8 GRAN COUPE
Коробка передач: Бензиновый
Цвет: Белый
Год выпуска: 2022
Состояние: Новое
Комплектация: Полная

СТОИМОСТЬ: 140000

Оформление заказа

Ваша почта
Введите почту...

Ваш логин
Введите логин...

Введите пароль для оформления заказа
Введите пароль...

Повторите пароль для подтверждения заказа
Введите пароль...

Оформить заказ

Рисунок 93 – Пример оформления заказа

Пример заполнения поля email, когда email не соответствует email. Результат обработки данной ситуации представлен на рисунке 94.

Email*

1

⚠ Адрес электронной почты должен содержать символ "@". В адресе "1" отсутствует символ "@".

Рисунок 94 – Пример неправильного ввода email

После успешного оформления заказа, в личном кабинете пользователя, а также в «Панели менеджера» добавиться данный заказ. Пример заказов каждого пользователя представлен на рисунке 95.

Ваши заказы:

Заказ №42

Авто: X 6
Двигатель: Бензиновый
Год выпуска: 2023-05-05 18:48:04
Стоимость: 100000
Комплектация: Полная
Цвет: Болотный
Состояние: Новое
Дата оформления: 2023-05-05 18:48:04

Менеджер: Титов Игорь
Номер телефона: 80294443434
Email: igor@mail.ru

Контакты: Отдел продаж автомобиля
Email: info@autoidea.by
Номер телефона: +375294572912
Время работы: Пн-Вт: 8:00 - 20:00

Отменить

Заказ №44

Авто: M X
Двигатель: Бензиновый

Рисунок 95 – Вывод заказов каждого клиента в личном кабинете

На рисунке 96 показан пример поиска авто, по выбранным критериям.

Подберите авто по вашим запросам


Название	Цвет	Комплектация	Наличие	Двигатель	Состояние	Год выпуска
X	Красный	Комплектация:	<input type="checkbox"/> Есть	<input type="checkbox"/> Бензиновый <input type="checkbox"/> Электрический	<input type="checkbox"/> Новое <input type="checkbox"/> Бу	От До

Цена
От
До

Найти

ALL X M 8 7 5 4 3 2 1 Z4

1



BMW I серии X
Бензиновый

Рисунок 96 – Пример поиска автомобиля

5 ОПИСАНИЕ ПРИМЕНЕНИЯ

Данный сайт (интернет-магазин) был создан на языке программирования PHP, JavaScript, а также с использованием SCSS, HTML, а также сборщика проектов GULP, поэтому для его запуска на компьютере необходимо установить и запустить сервер Open Server Panel. Определенный тип операционной системы не требуется.

Сайт прост в использовании: в нем имеется хедер-меню со ссылками, при нажатии на которые будет осуществляться переход на нужную страницу или кнопка в меню которого можно зайти в свой аккаунт или зарегистрироваться.

Данное приложение может быть полезно работникам, а также клиентам или просто посетителям сайта данного автосалона.

Администраторы могут со всеми удобствами и комфортом работать с данными о сотрудниках, клиентах и контактными данными автосалона. Он может при необходимости отредактировать их личные и персональные данные, добавить, удалить, а также просмотреть всю необходимую личную и персональную информацию, получить данные об их заказах с помощью поиска и сортировки.

Менеджер, работающий в автосалоне, сможет с помощью данного сайта добавлять, удалять, искать, редактировать автомобили и их модели. А также оформлять или удалять заказы с особой скоростью.

Клиенты могут получить полный список имеющихся автомобилей в данном автосалоне, произвести сортировку или поиск по своим предпочтениям. А также оформить заказ в один клик.

Также каждый посетитель, вне зависимости от его статуса и регистрации, может просмотреть всю необходимую контактную информацию о нашем автосалоне, ознакомиться с сервисными услугами, просмотреть всю информацию о каждом автомобиле, и зарегистрироваться для дальнейших действий.

Каждый пользователь нашего автосалона, имеет возможность изменить свои личные и персональные данные, удалить аккаунт или выйти из него в случае необходимости, изменить пароль для входа.

Таким образом, данный сайт в перспективе сможет принести много пользы множеству работников автосалона, клиентам и посетителям, поскольку оно создавалось специально для упрощения взаимодействия с данными в этой области.

ЗАКЛЮЧЕНИЕ

В результате выполнения работы была достигнута цель – разработана база данных и веб-сайт автосалона. В ходе разработки были решены следующие задачи:

Был проведен анализ информационных потребностей пользователей и сформулирована задача, реализация которой позволила удовлетворить их потребности, описана предметная область, инфологический, логический и физические этапы проектирования, описана среда и язык реализации.

Были разработаны инфологическая и физическая модели БД, определены типы данных и ограничения. Были освоены некоторые приемы написания запросов и представлений на языке SQL.

В данной курсовой работе была спроектирована (до 3Н) и реализована реляционная база данных (СУБД MySQL SERVER), создали программное средство, позволяющее осуществлять вывод данных и их изменение. В базе данных имеются ограничения: на некоторых полях установлено ограничение not null, во всех таблицах выбрано ключевое поле, на него наложено ограничение auto_increment.

В дальнейшем разработанную систему можно усовершенствовать до полноценного интернет-магазина автосалона какого-либо бренда и использовать каркас и структуру для любого автомобильного бренда.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Кузнецов Максим Валерьевич, Симдянов Игорь Вячеславович. MySQL на примерах.
2. А. Т. Доманов, Н. И. Сорока, БГУИР. Стандарт предприятия. Дипломные проекты (работы). Общие требования. СТП 01–2017.
3. Wiki-учебник по веб-технологиям: MySQL [Электронный ресурс] / Режим доступа: <https://www.webmasterwiki.ru/MySQL>. - Дата доступа: 21.04.2023.
4. Обучение основам MySQL [Электронный ресурс] / Режим доступа: <https://itproger.com/course/php-mysql/21>. - Дата доступа: 29.04.2023.
5. Основные команды sql [Электронный ресурс]. – Режим доступа: <https://tproger.ru/translations/sql-recap/>. – Дата доступа: 20.02.2023.
6. Sql теория и практика [Электронный ресурс]. – Режим доступа: <https://proglib.io/p/sql-practice-sites>– Дата доступа: 29.04.2023.
7. Основы sql. [Электронный ресурс]. – Режим доступа: <https://htmlacademy.ru/tutorial/php/sql>. – Дата доступа: 16.03.2023.
8. Gulp - сборка. [Электронный ресурс]. – Режим доступа: <https://www.youtube.com/watch?v=jU88mLuLWlk>. – Дата доступа: 12.02.2023.
9. Динамический веб-сайт на HTML/CSS/JS/PHP. [Электронный ресурс]. – Режим доступа: <https://www.youtube.com/@andrievskii/playlists>. – Дата доступа: 17.04.2023.
10. Официальный сайт BMW. [Электронный ресурс]. – Режим доступа: <https://www.bmw.by/>. – Дата доступа: 12.03.2023.

ПРИЛОЖЕНИЕ А

(обязательное)

Листинг кода с комментариями

Файл database.php

```
<?php

session_start();
require('connect.php');

//Проверяем выполнение запроса к БД
function dbCheckErr($query) {
    $errInfo = $query->errorInfo(); //Получаем ошибки в массив
    if($errInfo[0] !== PDO::ERR_NONE){
        echo $errInfo[2]; //Вывод ошибки
        exit(); //Завершаем функцию
    }
    return true;
}

//Запрос на получение всех данных из одной таблицы
function selectAll($table, $params = []) {
    global $pdo; //Глобальная переменная

    //Формируем sql запрос (Пример: SELECT * FROM models)
    $sql = "SELECT * FROM $table";

    if(!empty($params)) { //Если массив параметров не пустой, то разбираем данные
        $i = 0; //Если i=0, то where иначе and
        foreach($params as $key => $value){ //Разбираем данные
            if(!is_numeric($value)) { //Если параметр строка, то оборачиваем ее в кавычки
                $value = "'" . $value . "'";
            }
            if($i === 0) { //Если первый параметр добавляем впереди "WHERE
                $sql = $sql . " WHERE $key = $value";
            } else { //Если параметр не первый добавляем впередм "AND
                $sql = $sql . " AND $key = $value";
            }
            $i++;
        }
    }

    //Подготовка sql запроса для отправки на сервер
    $query = $pdo->prepare($sql);
    $query->execute();
}
```

```

//Проверка запроса на ошибки
dbCheckErr($query);

//Возвращаем полученный результат
return $query->fetchAll();
}

//Запрос на получение одной строки из одной выбранной таблицы
function selectOne($table, $params = []) {
    global $pdo; //Глобальная переменная

    //Формируем sql запрос (Пример: SELECT * FROM clients)
    $sql = "SELECT * FROM $table";

    if(!empty($params)) { //Если массив параметров не пустой, то разбираем данные
        $i = 0; //Если i=0, то where иначе and
        foreach($params as $key => $value){ //Разбираем данные
            if(!is_numeric($value)) { //Если параметр строка, то оборачиваем ее в кавычки
                $value = "'" . $value . "'";
            }
            if($i === 0) { //Если первый параметр добавляем впереди "WHERE
                $sql = $sql . " WHERE $key = $value";
            } else { //Если параметр не первый добавляем впереди "AND
                $sql = $sql . " AND $key = $value";
            }
            $i++;
        }
    }

    //Подготовка sql запроса для отправки на сервер
    $query = $pdo->prepare($sql);
    $query->execute();

    //Проверка запроса на ошибки
    dbCheckErr($query);

    //Возвращаем полученный результат
    return $query->fetch();
}

//Добавление записи в таблицу
function insert($table, $params) {
    global $pdo; //Глобальная переменная

    $i = 0; //Если 0, то запятую в sql запросе не ставим
    $coll = "; //Ключи
    $mask = "; //Значения

    //Разбираем параметры на данные для запроса (названия столбцов и значение)

```

```
foreach($params as $key => $value) {
    if($i === 0) {
        $coll = $coll . $key;
        $mask = $mask . " " . $value . " ";
    } else {
        $coll = $coll . ", $key";
        $mask = $mask . ", '$value'";
    }
    $i++;
}
```

//Формируем sql запрос (Пример: INSERT INTO contacts (name, phone, work_time, email) VALUES ('Call-center', '80447104585', '8-21', 'call@autoidea.by'))

```
$sql = "INSERT INTO $table ($coll) VALUES ($mask)";
```

//Подготовка sql запроса для отправки на сервер

```
$query = $pdo->prepare($sql);
```

```
$query->execute($params);
```

//Проверка запроса на ошибки

```
dbCheckErr($query);
```

//Возвращаем полученный результат

```
return $pdo->lastInsertId();
```

```
}
```

//Обновление данных в таблице БД

```
function update($table, $id, $params) {
```

```
    global $pdo; //Глобальная переменная
```

```
    $i = 0; //Если 0, то запятую в sql запросе не ставим
```

```
    $str = ""; //Ключи и параметры для обновления
```

//Разбираем параметры на данные для запроса (названия столбцов и значение)

```
foreach($params as $key => $value) {
```

```
    if($i === 0) {
```

```
        $str = $str . $key . " = " . $value . " ";
```

```
    } else {
```

```
        $str = $str . ", " . $key . " = " . $value . " ";
```

```
    }
```

```
    $i++;
```

```
}
```

//Формируем sql запрос (Пример: UPDATE contacts SET name = 'Отдел продаж автомобиля', phone = '+375447104585', work_time = 'Пн-Вс: 8:00 - 20:00', email = 'info@autoidea.by' WHERE id = 4)

```
$sql = "UPDATE $table SET $str WHERE id = $id";
```

//Подготовка sql запроса для отправки на сервер

```
$query = $pdo->prepare($sql);
```

```

$query->execute($params);

//Проверка запроса на ошибки
dbCheckErr($query);
}

//Удаление данных из таблицы БД
function delete($table, $id) {
    global $pdo; //Глобальная переменная

    //Формируем sql запрос (Пример: DELETE FROM contacts WHERE id =4)
    $sql = "DELETE FROM $table WHERE id =" . $id;

    //Подготовка sql запроса для отправки на сервер
    $query = $pdo->prepare($sql);
    $query->execute();

    //Проверка запроса на ошибки
    dbCheckErr($query);
}

function selectAutoFromAutosWithModelsOnSingle($table1, $table2, $id) {
    global $pdo;
    $sql = "SELECT t1.*, t2.model, t2.main_foto FROM $table1 AS t1 JOIN $table2 AS t2 ON t1.id_model = t2.id
    WHERE t1.id = $id";
    $query = $pdo->prepare($sql);
    $query->execute();
    dbCheckErr($query); //Проверка запроса на ошибки
    return $query->fetch();
}

//Получить количество авто каждой модели
function getCountModel($idModel) {
    global $pdo; //Глобальная переменная

    //Формируем sql запрос (Пример: SELECT COUNT(id_model) AS count FROM auto JOIN models ON
    auto.id_model = models.id WHERE id_model = 56)
    $sql = "SELECT COUNT(id_model) AS count FROM auto JOIN models ON auto.id_model = models.id WHERE
    id_model = $idModel";

    //Подготовка sql запроса для отправки на сервер
    $query = $pdo->prepare($sql);
    $query->execute();

    //Проверка запроса на ошибки
    dbCheckErr($query);

    //Возвращаем полученный результат

```

```

return $query->fetchAll();
}

//Выборка личных данных из сессии (того кто сейчас авторизован)
function getPersonalData($table1, $table2, $table3, $table4, $id) {
    global $pdo; //Глобальная переменная

    //Формируем sql запрос
    $sql = "SELECT
        t1.id,
        t1.last_name,
        t1.first_name,
        t1.surname,
        t1.date_birth,
        t1.phone,
        t2.city,
        t2.street,
        t2.house,
        t2.apartment,
        t3.series,
        t3.number,
        t3.issued_by,
        t4.login,
        t4.email,
        t4.date_regist
    FROM $table1 AS t1 JOIN $table2 AS t2 ON t1.id_address = t2.id
        JOIN $table3 AS t3 ON t1.id_passport = t3.id
        JOIN $table4 AS t4 ON t1.id_auth = t4.id
    WHERE t4.id = $id";

    //Подготовка sql запроса для отправки на сервер
    $query = $pdo->prepare($sql);
    $query->execute();

    //Проверка запроса на ошибки
    dbCheckErr($query);

    //Возвращаем полученный результат
    return $query->fetch();
}

//Выборка заказов определенного клиента
function getOrders($id) {
    global $pdo; //Глобальная переменная

    //Формируем sql запрос
    $sql = "SELECT
        t1.id,
        t1.date,

```

```

t2.last_name,
t2.first_name,
t2.phone,
t5.email AS emailEmployee,
t4.name AS nameContact,
t4.email AS emailContact,
t4.phone AS phoneContact,
t4.work_time,
t3.name,
t6.model,
t3.engine,
t3.year,
t3.price,
t3.color,
t3.complexion,
t3.state
FROM orders AS t1 JOIN employees AS t2 ON t1.id_employee = t2.id
      JOIN `auto` AS t3 ON t1.id_auto = t3.id
      JOIN contacts AS t4 ON t1.id_contact = t4.id
      JOIN authorization AS t5 ON t2.id_auth = t5.id
      JOIN models AS t6 ON t3.id_model = t6.id
WHERE t1.id_client = $id";

```

```

//Подготовка sql запроса для отправки на сервер
$query = $pdo->prepare($sql);
$query->execute();

```

```

//Проверка запроса на ошибки
dbCheckErr($query);

```

```

//Возвращаем полученный результат
return $query->fetchAll();
}

```

```

//Получаем уникальные значения цветов автомобиля
function getColorsAutos() {
    global $pdo; //Глобальная переменная

```

```

//Формируем sql запрос
$sql = "SELECT DISTINCT color FROM `auto`";

```

```

//Подготовка sql запроса для отправки на сервер
$query = $pdo->prepare($sql);
$query->execute();

```

```

//Проверка запроса на ошибки
dbCheckErr($query);

```

```

//Возвращаем полученный результат

```

```

return $query->fetchAll();
}

//Поиск авто по выбранным критериям
function searchAutos($params, $paramsPrice, $paramsYear) {
    global $pdo; //Глобальная переменная

    //Получаем строку для поиска цены
    $i = 0;
    $price = "";
    if($paramsPrice['price__from'] == "" && $paramsPrice['price__to'] != "") { // Если введена только сумма "до" =>
price < 70000
        $price = ' price' . ' < ' . $paramsPrice['price__to'];
    } elseif($paramsPrice['price__from'] != "" && $paramsPrice['price__to'] == "") { // Если введена только сумма
"от" => price > 50000
        $price = ' price' . ' > ' . $paramsPrice['price__from'];
    } elseif($paramsPrice['price__from'] != "" && $paramsPrice['price__to'] != "") { // Если введена сумма и "от" и
"до" => price BETWEEN 50000 AND 70000
        $price = ' price' . ' BETWEEN ' . $paramsPrice['price__from'] . ' AND ' . $paramsPrice['price__to'];
    }

    //Получаем строку для поиска года выпуска
    $year = "";
    if($paramsYear['year__from'] == "" && $paramsYear['year__to'] != "") { // Если введен год только "до" => year <
2022
        $year = ' year' . ' < ' . $paramsYear['year__to'];
    } elseif($paramsYear['year__from'] != "" && $paramsYear['year__to'] == "") { // Если введен год только "от" =>
year > 2019
        $year = ' year' . ' > ' . $paramsYear['year__from'];
    } elseif($paramsYear['year__from'] != "" && $paramsYear['year__to'] != "") { // Если введен год и "от" и "до" =>
year BETWEEN 2019 AND 2022
        $year = ' year' . ' BETWEEN ' . $paramsYear['year__from'] . ' AND ' . $paramsYear['year__to'];
    }

    //Получаем строку для поиска остальных данных
    $str = "";
    $i = 0;
    foreach($params as $key => $value) {
        if(!empty($value) && $value != "" && $i == 0) { //Если значение первое в строке то не ставим впереди
"AND"
            $str = $str . "`" . $key . "` = '" . $value . "'"; // `name` = '7' AND `color` = 'Серый'
            $i++;
        } else if(!empty($value) && $value != "" && $i != 0) { //Если значение первое в строке то ставим впереди
"AND"
            $str = $str . ' AND ' . "`" . $key . "` = '" . $value . "'";
        }
    }

    //Формируем итоговую строку запроса (проверка для правильности написания "WHERE" и "AND")

```



```

if($price != " && $year != " && $str != ") { //Если заполнены все поля
    $result = 'WHERE ' . $price . ' AND ' . $year . ' AND ' . $str; //WHERE price BETWEEN 50000 AND 70000
    AND year BETWEEN 2019 AND 2023 AND `name` = '7' AND `complexion` = 'Средняя' AND `color` =
    'Серый' AND `status` = '1' AND `engine` = 'Бензиновый'
} else if($price != " && $year != " && $str == ") { //Если асполнены поля цены и гола, но не заполнены дру-
гие данные
    $result = 'WHERE ' . $price . ' AND ' . $year; // WHERE price BETWEEN 50000 AND 70000 AND year BE-
TWEEN 2020 AND 2023
} else if($price != " && $year == " && $str != ") { //Если заполнены поля цены и других данных, но не заплот-
нен год
    $result = 'WHERE ' . $price . ' AND ' . $str; //WHERE price BETWEEN 55000 AND 65000 AND `name` =
'7' AND `color` = 'Синий'
} else if($price == " && $year != " && $str != ") { //Если заполнены год и другие данные, но не заполнена
цена
    $result = 'WHERE ' . $year . ' AND ' . $str; //WHERE year BETWEEN 2019 AND 2023 AND `name` =
'7' AND `complexion` = 'Полная' AND `color` = 'Голубой' AND `status` = '1' AND `engine` = 'Бензиновый'
} else if($price != " && $year == " && $str == ") { //Если заполнена только цена, а год и другие данные не
заполнены
    $result = 'WHERE ' . $price; //WHERE price BETWEEN 50000 AND 65000
} else if($price == " && $year != " && $str == ") { //Если заполнены только год, а цена и другие данные не
заполнены
    $result = 'WHERE ' . $year; //WHERE year BETWEEN 2020 AND 2022
} else if($price == " && $year == " && $str != ") { //Если заполнены только другие данные, а цена и год не
заполнены
    $result = 'WHERE ' . $str; //WHERE `complexion` = 'Полная' AND `color` = 'Черный'
}

//Формируем sql запрос
$sql = "SELECT
    auto.id,
    auto.name,
    auto.engine,
    auto.year,
    auto.price,
    auto.color,
    auto.complexion,
    auto.img,
    auto.status,
    models.model
FROM `auto` JOIN `models` ON auto.id_model = models.id $result";

//Подготовка sql запроса для отправки на сервер
$query = $pdo->prepare($sql);
$query->execute();

//Проверка запроса на ошибки
dbCheckErr($query);

//Возвращаем полученный результат
return $query->fetchAll();

```

```

}

//Поиск в панели админа по почти всем данным
function searchAdmin($search, $table) {
    $search = trim(strip_tags(stripslashes(htmlspecialchars($search)))); //Проверка вводимой строки
    global $pdo; //Глобальная переменная

    //Формируем sql запрос
    $sql = "SELECT * FROM $table WHERE
        last_name LIKE '%$search%' OR
        first_name LIKE '%$search%' OR
        surname LIKE '%$search%' OR
        date_birth LIKE '%$search%' OR
        phone LIKE '%$search%' OR
        city LIKE '%$search%' OR
        `number` LIKE '%$search%' OR
        `login` LIKE '%$search%' OR
        email LIKE '%$search%' OR
        series LIKE '%$search%' OR
        issued_by LIKE '%$search%'";

    //Подготовка sql запроса для отправки на сервер
    $query = $pdo->prepare($sql);
    $query->execute();

    //Проверка запроса на ошибки
    dbCheckErr($query);

    //Возвращаем полученный результат
    return $query->fetchAll();
}
?>

```