**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc469493859)

[1 АНАЛИЗ ПРОТОТИПОВ И ИСТОЧНИКОВ 5](#_Toc469493860)

[1.1 MySql Workbench 5](#_Toc469493861)

[1.2 СУБД MySql 5](#_Toc469493862)

[1.3 CarService 6](#_Toc469493863)

[1.4 Конфигурация «Автосервис» 7](#_Toc469493864)

[2 РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ 9](#_Toc469493865)

[2.1 Выбор средств разработки 9](#_Toc469493866)

[2.2 Спецификация функциональных требований 10](#_Toc469493867)

[2.3 Ключевые особенности базы данных 10](#_Toc469493868)

[3 РАЗРАБОТКА БАЗЫ ДАННЫХ 11](#_Toc469493869)

[3.1 Сущности и связи 11](#_Toc469493870)

[3.2 Особенности нормализации 13](#_Toc469493871)

[ЗАКЛЮЧЕНИЕ 15](#_Toc469493872)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 16](#_Toc469493873)

[ПРИЛОЖЕНИЕ A. ДАМП БАЗЫ ДАННЫХ 17](#_Toc469493874)

# ВВЕДЕНИЕ

Программное обеспечение - программа или множество программ, используемых для управления компьютером.

Длительное время наиболее распространенными способами передачи данных между ЭВМ являлись твердотельные накопители. На данных носителях информации передавались как данные, используемые человеком для своих целей, так и отдельные экземпляры программного обеспечения, которые таким образом распространялись. Практически единственным способом приобрести определенное программное средство была покупка твердотельного накопителя (CD-диск, дискеты) с записанным на нем программным средством. Это объясняется отсутствием ныне развитой глобальной сети Интернет. Однако с появлением такой развитой сети стало возможным не только получать запрашиваемую информацию по сети, но и оплачивать её также используя сеть Интернет. Это дает возможность исключить необходимость приобретения твердотельных накопителей для покупки программного средства, как и необходимость доставлять данный накопитель заказчику при помощи транспортных средств.

У каждого из способов распространения программных средств есть свои плюсы и минусы. Способ распространения на твердотельных накопителях будет существовать за счет своей надежности, однако в настоящее время все более популярными становятся сервисы распространения программных средств по сети Интернет. Преимуществом такого способа является скорость, отсутствие необходимости совершать какие-либо перемещения с целью приобретения, а также возможность восстановить программное средство даже в случае полной утраты аппаратуры на которой было установлено данное программное обеспечение.

В данном курсовом проекте разработана база данных для веб-ориентированного программного средства «Автосервис». Она включает в себя данные о автомобилях и их владельцах, механиках и поставленные для их задачи, а также данные о заказах на услуги автосервиса и статус их выполнения.

**1 АНАЛИЗ ПРОТОТИПОВ И ИСТОЧНИКОВ**

* 1. **MySql Workbench**

Первое, что стоит отметить — Workbench дает возможность визуально проектировать вашу базу данных, т.е. составлять схему БД. Визуальное представление вашей базы данных всегда дает куда большую информацию, чем сухой список таблиц. В таком варианте вы сразу видите, каким образом связаны между собой таблицы, можете группировать таблицы по каким-либо параметрами и отражать это на схеме. При этом визуальное проектирование удобно не только для того, чтобы кому-то рассказывать о проектируемой БД, но и для личного использования.

Во-вторых, программа имеет встроенный редактор SQL-кода, с помощью которого можно быстро внести любые правки в SQL-запросы. При этом возможно строить запросы любой сложности, получать различные выборки из таблиц, связывать их, создавать новые таблицы и редактировать существующие, работать с ключами, полями, связями. Одним словом — полноценный SQL-редактор.

В-третьих, MySQL Workbench позволяет осуществлять синхронизацию локальной схемы БД с реальной базой на вашем локальном или рабочем сервере. Благодаря этому после проектирования не требуется вручную создавать таблицы в базе на вашем сервере, достаточно сделать несколько простых действий в программе, после чего на рабочем сервере будет создана полноценная база данных со всеми указанными связями и параметрами.

Так же программа дает подробную статистику по работе базы данных.[9]

* 1. **СУБД MySql**

MySQL - свободная реляционная система управления базами данных. Разработку и поддержку MySQL осуществляет корпорация Oracle, получившая права на торговую марку вместе с поглощённой Sun Microsystems, которая ранее приобрела шведскую компанию MySQL AB. Продукт распространяется как под GNU General Public License, так и под собственной коммерческой лицензией. Помимо этого, разработчики создают функциональность по заказу лицензионных пользователей. Именно благодаря такому заказу почти в самых ранних версиях появился механизм репликации.

MySQL является решением для малых и средних приложений. Входит в состав серверов WAMP, AppServ, LAMP и в портативные сборки серверов Денвер, XAMPP, VertrigoServ. Обычно MySQL используется в качестве сервера, к которому обращаются локальные или удалённые клиенты, однако в дистрибутив входит библиотека внутреннего сервера, позволяющая включать MySQL в автономные программы.

Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа MyISAM, поддерживающие полнотекстовый поиск, так и таблицы InnoDB, поддерживающие транзакции на уровне отдельных записей. Более того, СУБД MySQL поставляется со специальным типом таблиц EXAMPLE, демонстрирующим принципы создания новых типов таблиц. Благодаря открытой архитектуре и GPL-лицензированию, в СУБД MySQL постоянно появляются новые типы таблиц.

26 февраля 2008 года Sun Microsystems приобрела MySQL AB за $1 млрд, 27 января 2010 года Oracle приобрела Sun Microsystems за $7,4 млрд и включила MySQL в свою линейку СУБД.

Сообществом разработчиков MySQL созданы различные ответвления кода, такие как Drizzle (англ.), OurDelta, Percona Server и MariaDB. Все эти

ответвления уже существовали на момент поглощения компании Sun корпорацией Oracle.

MySQL возникла как попытка применить mSQL к собственным разработкам компании: таблицам, для которых использовались ISAM — подпрограммы низкого уровня. В результате был выработан новый SQL-интерфейс, но API-интерфейс остался в наследство от mSQL. Откуда происходит название «MySQL» — доподлинно неизвестно. Разработчики дают два варианта: либо потому, что практически все наработки компании начинались с префикса My, либо в честь девочки по имени My, дочери Майкла Монти Видениуса, одного из разработчиков системы.

Логотип MySQL в виде дельфина носит имя «Sakila». Он был выбран из большого списка предложенных пользователями «имён дельфина». Имя «Sakila» было отправлено Open Source-разработчиком Ambrose Twebaze.

MySQL имеет API для языков Delphi, C, C++, Эйфель, Java, Лисп, Perl, PHP, Python, Ruby, Smalltalk, Компонентный Паскаль и Tcl, библиотеки для языков платформы .NET, а также обеспечивает поддержку для ODBC посредством ODBC-драйвера MyODBC.[11]

* 1. **CarService**

«CarService» - решение для владельцев Автосервисов, стремящихся сделать обслуживание своих клиентов удобным и современным. Модуль обеспечивает обслуживание пользователей автосервиса в режиме 24/7, предоставляя актуальную информацию о свободном/занятом времени. Модуль легко встраивается на любой сайт и позволяет клиентам записаться в автосервис онлайн без предварительного звонка по телефону.[13]

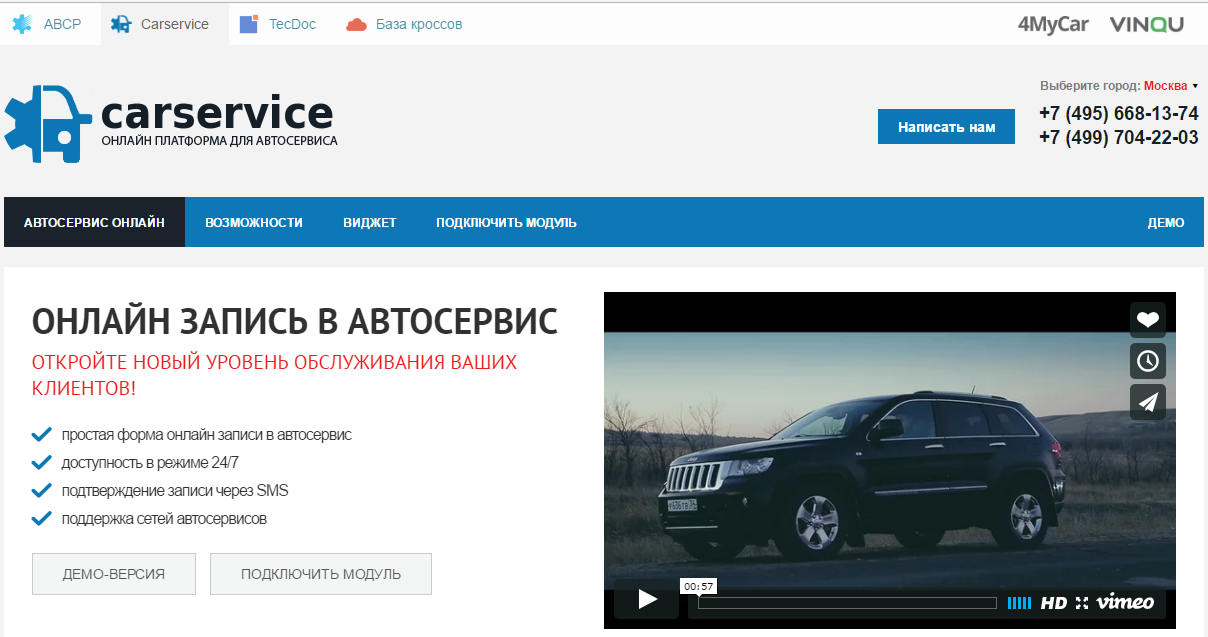


Рисунок 1.1 – Вид главной страницы модуля CarService

* 1. **Конфигурация «Автосервис»**

**Конфигурация "Автосервис"** - предназначена для автоматизации работы автосервиса. Данная программа позволит усовершенствовать процесс обслуживания клиентов благодаря ведению всестороннего учета. Имеется возможность ведения базы клиентов, их контактной информации, ведения справочников запчастей и услуг. Учет заявок клиентов, ремонтных работ и материалов, потраченных при ремонте. Регистрация продаж, блок складского учета (контроль поступления и списания запчастей, учет остатков на складе). Конфигурация легко и быстро настраивается под конкретные требования заказчика. Гибкая структура базы данных дает возможность создавать новые таблицы, отчеты, графики, добавлять поля, задавать списки и многое другое. Программа интуитивно проста для пользователей и не требует квалифицируемых ИТ-ресурсов.

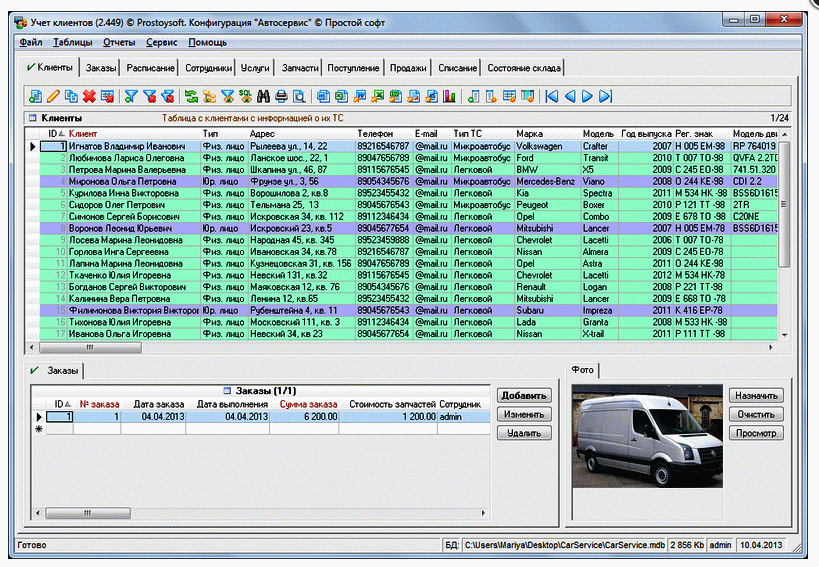


Рисунок 1.2 – Внешний вид программы

«**Конфигурация «Автосервис»**»

# РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ

## **2.1** **Выбор средств разработки**

Для хранения информации наиболее часто используются реляционные системы управления базами данными. Информация в них представлена в виде таблиц. Они содержат простые типы данных и, при необходимости, могут быть взаимосвязаны между собой.

Приложения, как правило, оперируют экземплярами классов (объектами), которые являются абстракциями объектов реального мира. Такой подход более удобен с точки зрения бизнес-логики. Кроме того, это дает такие преимущества при разработке как проверки типов, скорость работы кода и пр.

Таким образом, для использования реляционной базы данных необходимо использовать конвертер, который будет преобразовывать объекты в табличный вид и наоборот. При этом он должен учитывать различия в организации информации в обоих форматах. Например, в таблицах все строки отличаются друг от друга.

Для использования в разработке базы данных были выбраны 2 базы данных – MS SQL Server (режим разработки) и MySQL Workbench (режим реального использования). MS SQL Server была выбрана для режима разработки, т.к. она имеет прекрасные характеристики по производительности на малом объеме данных, не требует настройки и занимает мало места на диске. Выбор в пользу MySQL Workbench в режиме реального использования был сделан из-за ее высокой производительности при большом объеме данных, к тому же она имеет большое количество настроек для оптимизации.

В качестве языка программирования был выбран язык запросов SQL, т.к. этот язык отличается высокой скоростью разработки, легкой читаемостью и переносимостью программ. В качестве основной платформы был выбран MySQL Workbench. Данная платформа позволяет быстро и легко создавать базы данных, к тому же она предоставляет удобные механизмы работы с различными базами данных.

* 1. **Спецификация функциональных требований**

С учетом вышеприведенных требований сформулирована следующая спецификация функциональных требований:

* База данных должна учитывать все параметры предметной области.
* База данных должна иметь 3-ю нормальную форму.
* Количество таблиц должно составлять не менее 20 штук.
* Должны быть сделаны триггеры на удаление.
* Должны быть сделаны каскадные операции.
* База данных должна быть спроектирована в MySQL Workbench.
* База данных должна иметь целостность данных, то есть соблюдать логические условия, в соответствии с которыми данные считаются правильными.
* Должны быть верно соблюдены типы данных в таблицах.
* Должны быть соблюдены законы баз данных.
* Для обеспечения целостности на данные, хранящиеся в БД, должны быть реализованы ограничения.
* Все таблицы должны храниться в базе данных.
  1. **Ключевые особенности базы данных**

К ключевым особенностям разрабатываемого программного средства можно отнести следующие:

* низкие аппаратные требования;
* работа с большим количеством запросов на просмотр данных и небольшим числом запросов на редактирование;
* совмещение служебной и общедоступной информации;
* модульность и простота расширения.

# 3 РАЗРАБОТКА БАЗЫ ДАННЫХ

## 

## **3.1 Сущности и связи**

Предметная область содержит следующие сущности:

**Car**

Сущность Car является информацией об автомобиле владельца. Содержит идентификационный номер и ссылки на объекты, которые являются технической составляющей автомобиля.

**Engine**

Сущность Engine является информацией о двигателе автомобиля. Содержит тип двигателя, его объём, идентификационный номер, ссылку на объект комплектацию двигателя.

**Suspension**

Сущность Suspension является информацией о подвеске автомобиля и содержит идентификационный номер, названия передней и задней оси и ссылки на объект Wheels и объект BrakeSystem.

**Model**

Сущность Model является информацией о модели автомобиля и содержит идентификационный номер, название модели, её год выпуска и ссылку на производителя модели автомобиля.

**Manufacturers**

Сущность Manufacturers является информацией о производителе автомобиля и содержит идентификационный номер, название производителя.

В отдельные сущности были вынесены самые главные части автомобиля. Это сделано для того, чтобы упростить работу механиков над отдельными частями автомобиля. На рисунке 2.1 показаны связи между сущностью автомобиля и его комплектующими.

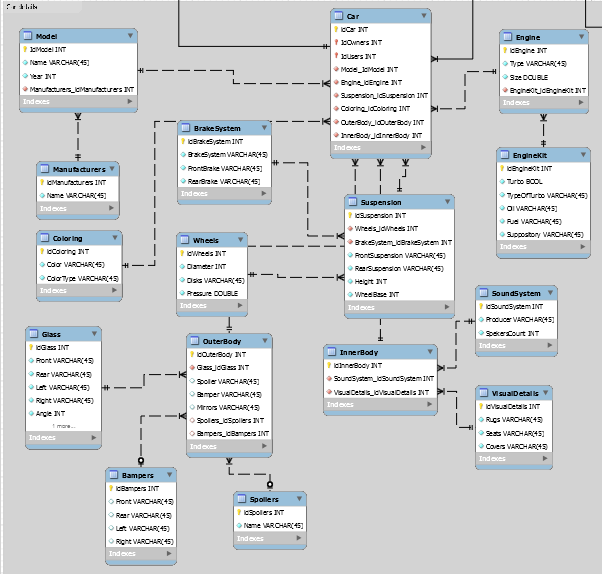


Рисунок 3.1 – Связь автомобиля с его запчастями

**Users**

Сущность Users является информацией о пользователе, осуществляющем заказ на предоставление услуг автосервиса и прочие действия в процессе пользования услугами сервиса. Содержит идентификационный номер пользователя, его логин, email-адрес, фамилию, имя, отчество, пароль в зашифрованном виде.

**Roles**

Сущность Roles является информацией о роли пользователя для разграничения доступа к элементам системы и содержит идентификационный номер роли и её название.

**Feedback**

Сущность Feedback является информацией об отзывах оставляемых пользователями на сайте автосервиса. Содержит идентификационный номер, идентификационный номер пользователя и содержание сделанного отзыва.

**Notifications**

Сущность Notifications является информацией об уведомлениях, которые пользователь может установить на сайте. Содержит идентификационный номер и тип уведомления.

**Owners**

Сущность Owners является информацией о собственниках, которые обращаются за услугами автосервиса. Содержит идентификационный номер и номер пользователя.

**Mechanics**

Сущность Mechanics является информацией о механиках, которые работают на автосервисе. Содержит идентификационный номер, номер пользователя, номер должности и номер сотрудника.

**Services**

Сущность Services является информацией об услугах предоставляемых автосервисом. Содержит идентификационный номер, название услуги и её описание.

**RepairOrder**

Сущность RepairOrder является информацией об услуге заказанной владельцем у автосервиса. Содержит идентификационный номер, номер пользователя, номер оказываемой услуги, дату создания услуги, её предполагаемое время окончания, цена, скидка.

**Labors**

Сущность Labors является информацией об итоговом счёте, который должен заплатить владелец автосервису за оказанные услуги. Содержит идентификационный номер и итоговую цену.

**RepairStatus**

Сущность RepairStatus является информацией о количестве проделанной работы механиками на автомобиле владельца. Содержит идентификационный номер, номер пользователя, номер оказываемой услуги, дату создания услуги, дату обновления услуги, номер статуса.

## **3.2 Особенности нормализации**

Проверка на первую нормальную форму. Просмотрев отношения можно заметить, что отсутствуют сложные атрибуты, а также не наблюдаются многозначные атрибуты, т.е. таблицы приведены к первой нормальной форме [14].

Проверка на вторую нормальную форму. Просмотрев все отношения, описанные выше, можно сказать, что неключевые атрибуты этих отношений функционально полно зависят от первичных ключей, что является доказательством того, что отношения приведены ко второй нормальной форме.

Проверка на третью нормальную форму. Транзитивные функциональные зависимости отсутствуют, поэтому отношения соответствуют третьей нормальной форме [15].

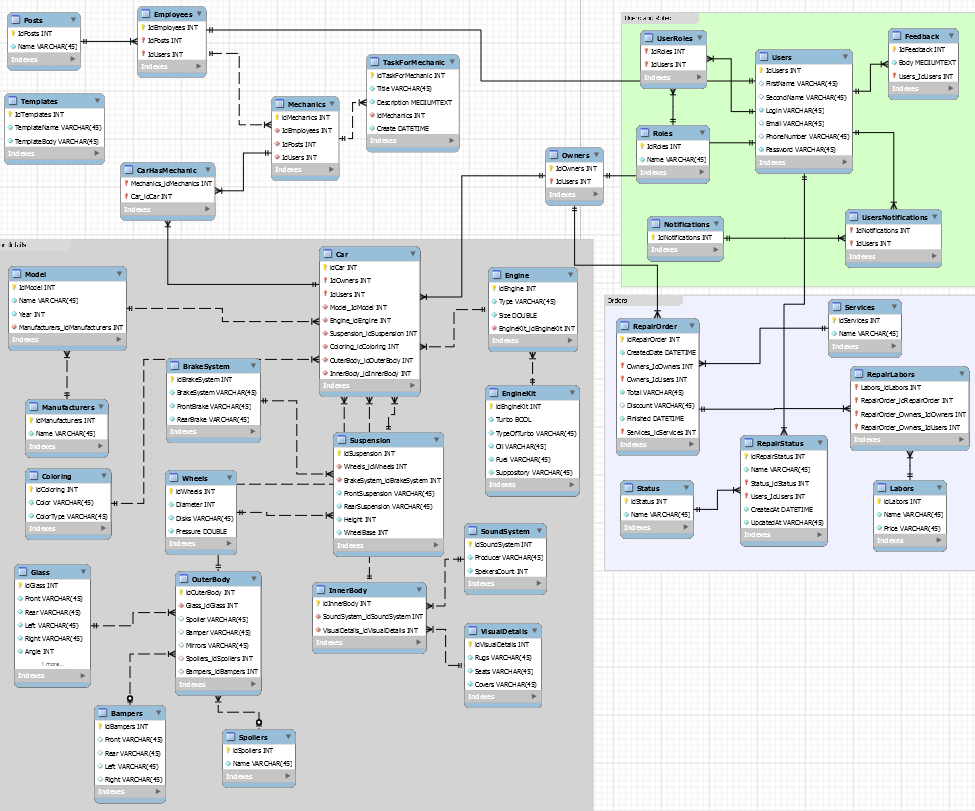


Рисунок 3.2 - Структура базы данных

# ЗАКЛЮЧЕНИЕ

Разработанная в ходе выполнения курсового проекта база данных «Автосервис», является актуальной на сегодняшний день и имеет большую практическую значимость. Она помогает в работе сотрудников поликлиники по сбору данных, необходимых при лечении, а также по сбору данных о самих сотрудниках.

В спроектированной базе заложены следующие возможности:

* Отображение всех доступных поликлиник и возможности заказ талонов на прием.
* Отображение оказываемых автосервисом услуг;
* Отображение список всех механиков;
* Отображение список всех владельцев автомобилей;
* Отображение список всех пользователей;
* Отображение список автомобилей на сервисе
* Заказ на оказание услуг;
* Просмотр заказов;
* Редактирование профиля пользователя;
* Возможность распределения пользователей по ролям;
* Отслеживание статуса проведения технического обслуживания;
* История оказания услуг для конкретного клиента.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] MSDN [Электронный ресурс] – Электронные данные: http://msdn.microsoft.com/en-us/library.

[2] http://carservice.abcp.ru/carservice-widget

[3] Фримен, А. ASP.NET MVC 4 с примерами на C# 5.0 для профессионалов.: Пер. с англ. / А. Фримен. – М.: Изд. Дом «Вильямс», 2013. – 688 с.

[4] Албахари, Д. C# 5.0. Справочник. Полное описание языка.: Пер. с англ. / Д. [5] Албахари, Б. Албахари. – М.: Изд. Дом «Вильямс», 2013. – 1008 с.

[6] Кнут, Д.Э. Искусство программирования: в 3 т. / Д. Э. Кнут. – 3-изд. – М.: Изд. Дом «Вильямс», 2006. – Т. 1: Основные алгоритмы. – 720 c.

[7] Станек, У. Microsoft SQL Server 2008. Справочник администратора.: Пер. с англ. / У. Станек – М.: Русская Редакция, 2013 – 720 с.

[8] Роберт Мартин – Чистый код. Создание, анализ и рефакторинг. – Питер, 2014.

[9] <https://ru.wikipedia.org/wiki/MySQL_Workbench>.

[10] Куликов С. С. Базы данных. – Мн.: БГУИР – 248с.

[11] <https://ru.wikipedia.org/wiki/MySQL>

[12] Карпова Т.С. Базы данных. Модели, разработка, реализация/СПб.: Питер, 2002. - 304 с.

[13] К.Дж.Дейт, Хью Дарвен. Основы будущих систем баз данных. Третий манифест. – Москва: Издательство Янус-К, 2004 г. – 656с.

[14] Карпова Т.С. Базы данных. Модели, разработка, реализация.СПб.: Питер,2002. – 304 с.

[15] Хансен Г.,Хансен Д.Базы данных: разработка и управление, пер. с англ., М.: ЗАО«Издательство БИНОМ», 1999. – 704с.: ил.

# ПРИЛОЖЕНИЕ A. ДАМП БАЗЫ ДАННЫХ

-- MySQL Script generated by MySQL Workbench

-- 12/13/16 23:41:25

-- Model: New Model Version: 1.0

-- MySQL Workbench Forward Engineering

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='TRADITIONAL,ALLOW\_INVALID\_DATES';

-- -----------------------------------------------------

-- Schema mydb

-- -----------------------------------------------------

DROP SCHEMA IF EXISTS `mydb` ;

-- -----------------------------------------------------

-- Schema mydb

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `mydb` DEFAULT CHARACTER SET utf8 ;

USE `mydb` ;

-- -----------------------------------------------------

-- Table `mydb`.`Posts`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`Posts` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Posts` (

`IdPosts` INT NOT NULL,

`Name` VARCHAR(45) NOT NULL,

PRIMARY KEY (`IdPosts`),

UNIQUE INDEX `Name\_UNIQUE` (`Name` ASC))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Users`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`Users` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Users` (

`IdUsers` INT NOT NULL,

`FirstName` VARCHAR(45) NULL,

`SecondName` VARCHAR(45) NULL,

`Login` VARCHAR(45) NOT NULL,

`Email` VARCHAR(45) NULL,

`PhoneNumber` VARCHAR(45) NULL,

`Password` VARCHAR(45) NOT NULL,

PRIMARY KEY (`IdUsers`),

UNIQUE INDEX `IdUsers\_UNIQUE` (`IdUsers` ASC),

UNIQUE INDEX `Login\_UNIQUE` (`Login` ASC))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Employees`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`Employees` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Employees` (

`IdEmployees` INT NOT NULL,

`IdPosts` INT NOT NULL,

`IdUsers` INT NOT NULL,

PRIMARY KEY (`IdEmployees`, `IdPosts`, `IdUsers`),

INDEX `fk\_Employees\_Posts1\_idx` (`IdPosts` ASC),

INDEX `fk\_Employees\_Users1\_idx` (`IdUsers` ASC),

CONSTRAINT `fk\_Employees\_Posts1`

FOREIGN KEY (`IdPosts`)

REFERENCES `mydb`.`Posts` (`IdPosts`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Employees\_Users1`

FOREIGN KEY (`IdUsers`)

REFERENCES `mydb`.`Users` (`IdUsers`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Owners`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`Owners` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Owners` (

`IdOwners` INT NOT NULL,

`IdUsers` INT NOT NULL,

PRIMARY KEY (`IdOwners`, `IdUsers`),

INDEX `fk\_Owners\_Users1\_idx` (`IdUsers` ASC),

CONSTRAINT `fk\_Owners\_Users1`

FOREIGN KEY (`IdUsers`)

REFERENCES `mydb`.`Users` (`IdUsers`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Manufacturers`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`Manufacturers` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Manufacturers` (

`idManufacturers` INT NOT NULL,

`Name` VARCHAR(45) NOT NULL,

PRIMARY KEY (`idManufacturers`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Model`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`Model` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Model` (

`IdModel` INT NOT NULL,

`Name` VARCHAR(45) NOT NULL,

`Year` INT NOT NULL,

`Manufacturers\_idManufacturers` INT NOT NULL,

PRIMARY KEY (`IdModel`),

INDEX `fk\_Model\_Manufacturers1\_idx` (`Manufacturers\_idManufacturers` ASC),

CONSTRAINT `fk\_Model\_Manufacturers1`

FOREIGN KEY (`Manufacturers\_idManufacturers`)

REFERENCES `mydb`.`Manufacturers` (`idManufacturers`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`EngineKit`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`EngineKit` ;

CREATE TABLE IF NOT EXISTS `mydb`.`EngineKit` (

`idEngineKit` INT NOT NULL,

`Turbo` TINYINT(1) NOT NULL,

`TypeOfTurbo` VARCHAR(45) NOT NULL,

`Oil` VARCHAR(45) NOT NULL,

`Fuel` VARCHAR(45) NOT NULL,

`Suppository` VARCHAR(45) NOT NULL,

PRIMARY KEY (`idEngineKit`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Engine`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`Engine` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Engine` (

`idEngine` INT NOT NULL,

`Type` VARCHAR(45) NOT NULL,

`Size` DOUBLE NOT NULL,

`EngineKit\_idEngineKit` INT NOT NULL,

PRIMARY KEY (`idEngine`),

INDEX `fk\_Engine\_EngineKit1\_idx` (`EngineKit\_idEngineKit` ASC),

CONSTRAINT `fk\_Engine\_EngineKit1`

FOREIGN KEY (`EngineKit\_idEngineKit`)

REFERENCES `mydb`.`EngineKit` (`idEngineKit`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Wheels`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`Wheels` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Wheels` (

`idWheels` INT NOT NULL,

`Diameter` INT NOT NULL,

`Disks` VARCHAR(45) NOT NULL,

`Pressure` DOUBLE NOT NULL,

PRIMARY KEY (`idWheels`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`BrakeSystem`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`BrakeSystem` ;

CREATE TABLE IF NOT EXISTS `mydb`.`BrakeSystem` (

`idBrakeSystem` INT NOT NULL,

`BrakeSystem` VARCHAR(45) NOT NULL,

`FrontBrake` VARCHAR(45) NOT NULL,

`RearBrake` VARCHAR(45) NOT NULL,

PRIMARY KEY (`idBrakeSystem`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Suspension`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`Suspension` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Suspension` (

`idSuspension` INT NOT NULL,

`Wheels\_idWheels` INT NOT NULL,

`BrakeSystem\_idBrakeSystem` INT NOT NULL,

`FrontSuspension` VARCHAR(45) NOT NULL,

`RearSuspension` VARCHAR(45) NOT NULL,

`Height` INT NOT NULL,

`WheelBase` INT NOT NULL,

PRIMARY KEY (`idSuspension`),

INDEX `fk\_Suspension\_Wheels1\_idx` (`Wheels\_idWheels` ASC),

INDEX `fk\_Suspension\_BrakeSystem1\_idx` (`BrakeSystem\_idBrakeSystem` ASC),

CONSTRAINT `fk\_Suspension\_Wheels1`

FOREIGN KEY (`Wheels\_idWheels`)

REFERENCES `mydb`.`Wheels` (`idWheels`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Suspension\_BrakeSystem1`

FOREIGN KEY (`BrakeSystem\_idBrakeSystem`)

REFERENCES `mydb`.`BrakeSystem` (`idBrakeSystem`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Coloring`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`Coloring` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Coloring` (

`idColoring` INT NOT NULL,

`Color` VARCHAR(45) NOT NULL,

`ColorType` VARCHAR(45) NOT NULL,

PRIMARY KEY (`idColoring`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Glass`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`Glass` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Glass` (

`idGlass` INT NOT NULL,

`Front` VARCHAR(45) NOT NULL,

`Rear` VARCHAR(45) NOT NULL,

`Left` VARCHAR(45) NOT NULL,

`Right` VARCHAR(45) NOT NULL,

`Angle` INT NOT NULL,

`Toning` VARCHAR(45) NULL,

PRIMARY KEY (`idGlass`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Spoilers`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`Spoilers` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Spoilers` (

`idSpoilers` INT NOT NULL,

`Name` VARCHAR(45) NOT NULL,

PRIMARY KEY (`idSpoilers`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Bampers`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`Bampers` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Bampers` (

`idBampers` INT NOT NULL,

`Front` VARCHAR(45) NULL,

`Rear` VARCHAR(45) NULL,

`Left` VARCHAR(45) NULL,

`Right` VARCHAR(45) NULL,

PRIMARY KEY (`idBampers`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`OuterBody`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`OuterBody` ;

CREATE TABLE IF NOT EXISTS `mydb`.`OuterBody` (

`idOuterBody` INT NOT NULL,

`Glass\_idGlass` INT NOT NULL,

`Spoiler` VARCHAR(45) NULL,

`Bamper` VARCHAR(45) NULL,

`Mirrors` VARCHAR(45) NULL,

`Spoilers\_idSpoilers` INT NULL,

`Bampers\_idBampers` INT NULL,

PRIMARY KEY (`idOuterBody`),

INDEX `fk\_OuterBody\_Glass1\_idx` (`Glass\_idGlass` ASC),

INDEX `fk\_OuterBody\_Spoilers1\_idx` (`Spoilers\_idSpoilers` ASC),

INDEX `fk\_OuterBody\_Bampers1\_idx` (`Bampers\_idBampers` ASC),

CONSTRAINT `fk\_OuterBody\_Glass1`

FOREIGN KEY (`Glass\_idGlass`)

REFERENCES `mydb`.`Glass` (`idGlass`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_OuterBody\_Spoilers1`

FOREIGN KEY (`Spoilers\_idSpoilers`)

REFERENCES `mydb`.`Spoilers` (`idSpoilers`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_OuterBody\_Bampers1`

FOREIGN KEY (`Bampers\_idBampers`)

REFERENCES `mydb`.`Bampers` (`idBampers`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`SoundSystem`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`SoundSystem` ;

CREATE TABLE IF NOT EXISTS `mydb`.`SoundSystem` (

`idSoundSystem` INT NOT NULL,

`Producer` VARCHAR(45) NOT NULL,

`SpekersCount` INT NOT NULL,

PRIMARY KEY (`idSoundSystem`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`VisualDetails`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`VisualDetails` ;

CREATE TABLE IF NOT EXISTS `mydb`.`VisualDetails` (

`idVisualDetails` INT NOT NULL,

`Rugs` VARCHAR(45) NOT NULL,

`Seats` VARCHAR(45) NOT NULL,

`Covers` VARCHAR(45) NOT NULL,

PRIMARY KEY (`idVisualDetails`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`InnerBody`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`InnerBody` ;

CREATE TABLE IF NOT EXISTS `mydb`.`InnerBody` (

`idInnerBody` INT NOT NULL,

`SoundSystem\_idSoundSystem` INT NOT NULL,

`VisualDetails\_idVisualDetails` INT NOT NULL,

PRIMARY KEY (`idInnerBody`),

INDEX `fk\_InnerBody\_SoundSystem1\_idx` (`SoundSystem\_idSoundSystem` ASC),

INDEX `fk\_InnerBody\_VisualDetails1\_idx` (`VisualDetails\_idVisualDetails` ASC),

CONSTRAINT `fk\_InnerBody\_SoundSystem1`

FOREIGN KEY (`SoundSystem\_idSoundSystem`)

REFERENCES `mydb`.`SoundSystem` (`idSoundSystem`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_InnerBody\_VisualDetails1`

FOREIGN KEY (`VisualDetails\_idVisualDetails`)

REFERENCES `mydb`.`VisualDetails` (`idVisualDetails`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Car`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`Car` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Car` (

`idCar` INT NOT NULL,

`IdOwners` INT NOT NULL,

`IdUsers` INT NOT NULL,

`Model\_IdModel` INT NOT NULL,

`Engine\_idEngine` INT NOT NULL,

`Suspension\_idSuspension` INT NOT NULL,

`Coloring\_idColoring` INT NOT NULL,

`OuterBody\_idOuterBody` INT NOT NULL,

`InnerBody\_idInnerBody` INT NOT NULL,

PRIMARY KEY (`idCar`, `IdOwners`, `IdUsers`),

INDEX `fk\_Car\_Owners1\_idx` (`IdOwners` ASC, `IdUsers` ASC),

INDEX `fk\_Car\_Model1\_idx` (`Model\_IdModel` ASC),

INDEX `fk\_Car\_Engine1\_idx` (`Engine\_idEngine` ASC),

INDEX `fk\_Car\_Suspension1\_idx` (`Suspension\_idSuspension` ASC),

INDEX `fk\_Car\_Coloring1\_idx` (`Coloring\_idColoring` ASC),

INDEX `fk\_Car\_OuterBody1\_idx` (`OuterBody\_idOuterBody` ASC),

INDEX `fk\_Car\_InnerBody1\_idx` (`InnerBody\_idInnerBody` ASC),

CONSTRAINT `fk\_Car\_Owners1`

FOREIGN KEY (`IdOwners` , `IdUsers`)

REFERENCES `mydb`.`Owners` (`IdOwners` , `IdUsers`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Car\_Model1`

FOREIGN KEY (`Model\_IdModel`)

REFERENCES `mydb`.`Model` (`IdModel`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Car\_Engine1`

FOREIGN KEY (`Engine\_idEngine`)

REFERENCES `mydb`.`Engine` (`idEngine`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Car\_Suspension1`

FOREIGN KEY (`Suspension\_idSuspension`)

REFERENCES `mydb`.`Suspension` (`idSuspension`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Car\_Coloring1`

FOREIGN KEY (`Coloring\_idColoring`)

REFERENCES `mydb`.`Coloring` (`idColoring`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Car\_OuterBody1`

FOREIGN KEY (`OuterBody\_idOuterBody`)

REFERENCES `mydb`.`OuterBody` (`idOuterBody`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Car\_InnerBody1`

FOREIGN KEY (`InnerBody\_idInnerBody`)

REFERENCES `mydb`.`InnerBody` (`idInnerBody`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Roles`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`Roles` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Roles` (

`IdRoles` INT NOT NULL,

`Name` VARCHAR(45) NOT NULL,

PRIMARY KEY (`IdRoles`),

UNIQUE INDEX `IdRoles\_UNIQUE` (`IdRoles` ASC))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Notifications`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`Notifications` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Notifications` (

`IdNotifications` INT NOT NULL,

PRIMARY KEY (`IdNotifications`),

UNIQUE INDEX `IdNotifications\_UNIQUE` (`IdNotifications` ASC))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`UserRoles`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`UserRoles` ;

CREATE TABLE IF NOT EXISTS `mydb`.`UserRoles` (

`IdRoles` INT NOT NULL,

`IdUsers` INT NOT NULL,

PRIMARY KEY (`IdRoles`, `IdUsers`),

INDEX `fk\_UserRoles\_Users1\_idx` (`IdUsers` ASC),

CONSTRAINT `fk\_UserRoles\_UserRoles1`

FOREIGN KEY (`IdRoles`)

REFERENCES `mydb`.`Roles` (`IdRoles`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_UserRoles\_Users1`

FOREIGN KEY (`IdUsers`)

REFERENCES `mydb`.`Users` (`IdUsers`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`UsersNotifications`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`UsersNotifications` ;

CREATE TABLE IF NOT EXISTS `mydb`.`UsersNotifications` (

`IdNotifications` INT NOT NULL,

`IdUsers` INT NOT NULL,

PRIMARY KEY (`IdNotifications`, `IdUsers`),

INDEX `fk\_UsersNotifications\_Users1\_idx` (`IdUsers` ASC),

CONSTRAINT `fk\_UsersNotifications\_Notifications1`

FOREIGN KEY (`IdNotifications`)

REFERENCES `mydb`.`Notifications` (`IdNotifications`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_UsersNotifications\_Users1`

FOREIGN KEY (`IdUsers`)

REFERENCES `mydb`.`Users` (`IdUsers`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Feedback`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`Feedback` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Feedback` (

`IdFeedback` INT NOT NULL,

`Body` MEDIUMTEXT NOT NULL,

`Users\_IdUsers` INT NOT NULL,

PRIMARY KEY (`IdFeedback`, `Users\_IdUsers`),

INDEX `fk\_Feedback\_Users1\_idx` (`Users\_IdUsers` ASC),

CONSTRAINT `fk\_Feedback\_Users1`

FOREIGN KEY (`Users\_IdUsers`)

REFERENCES `mydb`.`Users` (`IdUsers`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Templates`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`Templates` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Templates` (

`IdTemplates` INT NOT NULL,

`TemplateName` VARCHAR(45) NOT NULL,

`TemplateBody` VARCHAR(45) NOT NULL,

PRIMARY KEY (`IdTemplates`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Mechanics`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`Mechanics` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Mechanics` (

`idMechanics` INT NOT NULL,

`IdEmployees` INT NOT NULL,

`IdPosts` INT NOT NULL,

`IdUsers` INT NOT NULL,

PRIMARY KEY (`idMechanics`),

INDEX `fk\_Mechanics\_Employees1\_idx` (`IdEmployees` ASC, `IdPosts` ASC, `IdUsers` ASC),

CONSTRAINT `fk\_Mechanics\_Employees1`

FOREIGN KEY (`IdEmployees` , `IdPosts` , `IdUsers`)

REFERENCES `mydb`.`Employees` (`IdEmployees` , `IdPosts` , `IdUsers`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`CarHasMechanic`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`CarHasMechanic` ;

CREATE TABLE IF NOT EXISTS `mydb`.`CarHasMechanic` (

`Mechanics\_idMechanics` INT NOT NULL,

`Car\_idCar` INT NOT NULL,

PRIMARY KEY (`Mechanics\_idMechanics`, `Car\_idCar`),

INDEX `fk\_CarHasMechanic\_Car1\_idx` (`Car\_idCar` ASC),

CONSTRAINT `fk\_CarHasMechanic\_Mechanics1`

FOREIGN KEY (`Mechanics\_idMechanics`)

REFERENCES `mydb`.`Mechanics` (`idMechanics`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_CarHasMechanic\_Car1`

FOREIGN KEY (`Car\_idCar`)

REFERENCES `mydb`.`Car` (`idCar`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`TaskForMechanic`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`TaskForMechanic` ;

CREATE TABLE IF NOT EXISTS `mydb`.`TaskForMechanic` (

`idTaskForMechanic` INT NOT NULL,

`Title` VARCHAR(45) NOT NULL,

`Description` MEDIUMTEXT NOT NULL,

`idMechanics` INT NOT NULL,

`Create` DATETIME NOT NULL,

PRIMARY KEY (`idTaskForMechanic`),

INDEX `fk\_TaskForMechanic\_Mechanics1\_idx` (`idMechanics` ASC),

CONSTRAINT `fk\_TaskForMechanic\_Mechanics1`

FOREIGN KEY (`idMechanics`)

REFERENCES `mydb`.`Mechanics` (`idMechanics`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Labors`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`Labors` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Labors` (

`idLabors` INT NOT NULL,

`Name` VARCHAR(45) NOT NULL,

`Price` VARCHAR(45) NOT NULL,

PRIMARY KEY (`idLabors`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Services`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`Services` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Services` (

`idServices` INT NOT NULL,

`Name` VARCHAR(45) NOT NULL,

PRIMARY KEY (`idServices`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`RepairOrder`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`RepairOrder` ;

CREATE TABLE IF NOT EXISTS `mydb`.`RepairOrder` (

`idRepairOrder` INT NOT NULL,

`CreatedDate` DATETIME NOT NULL,

`Owners\_IdOwners` INT NOT NULL,

`Owners\_IdUsers` INT NOT NULL,

`Total` VARCHAR(45) NOT NULL,

`Discount` VARCHAR(45) NULL,

`Finished` DATETIME NOT NULL,

`Services\_idServices` INT NOT NULL,

PRIMARY KEY (`idRepairOrder`, `Owners\_IdOwners`, `Owners\_IdUsers`, `Services\_idServices`),

INDEX `fk\_RepairOrder\_Owners1\_idx` (`Owners\_IdOwners` ASC, `Owners\_IdUsers` ASC),

INDEX `fk\_RepairOrder\_Services1\_idx` (`Services\_idServices` ASC),

CONSTRAINT `fk\_RepairOrder\_Owners1`

FOREIGN KEY (`Owners\_IdOwners` , `Owners\_IdUsers`)

REFERENCES `mydb`.`Owners` (`IdOwners` , `IdUsers`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_RepairOrder\_Services1`

FOREIGN KEY (`Services\_idServices`)

REFERENCES `mydb`.`Services` (`idServices`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`RepairLabors`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`RepairLabors` ;

CREATE TABLE IF NOT EXISTS `mydb`.`RepairLabors` (

`Labors\_idLabors` INT NOT NULL,

`RepairOrder\_idRepairOrder` INT NOT NULL,

`RepairOrder\_Owners\_IdOwners` INT NOT NULL,

`RepairOrder\_Owners\_IdUsers` INT NOT NULL,

PRIMARY KEY (`Labors\_idLabors`, `RepairOrder\_idRepairOrder`, `RepairOrder\_Owners\_IdOwners`, `RepairOrder\_Owners\_IdUsers`),

INDEX `fk\_RepairJob\_RepairOrder1\_idx` (`RepairOrder\_idRepairOrder` ASC, `RepairOrder\_Owners\_IdOwners` ASC, `RepairOrder\_Owners\_IdUsers` ASC),

CONSTRAINT `fk\_RepairJob\_Labors1`

FOREIGN KEY (`Labors\_idLabors`)

REFERENCES `mydb`.`Labors` (`idLabors`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_RepairJob\_RepairOrder1`

FOREIGN KEY (`RepairOrder\_idRepairOrder` , `RepairOrder\_Owners\_IdOwners` , `RepairOrder\_Owners\_IdUsers`)

REFERENCES `mydb`.`RepairOrder` (`idRepairOrder` , `Owners\_IdOwners` , `Owners\_IdUsers`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`Status`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`Status` ;

CREATE TABLE IF NOT EXISTS `mydb`.`Status` (

`idStatus` INT NOT NULL,

`Name` VARCHAR(45) NOT NULL,

PRIMARY KEY (`idStatus`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `mydb`.`RepairStatus`

-- -----------------------------------------------------

DROP TABLE IF EXISTS `mydb`.`RepairStatus` ;

CREATE TABLE IF NOT EXISTS `mydb`.`RepairStatus` (

`idRepairStatus` INT NOT NULL,

`Name` VARCHAR(45) NOT NULL,

`Status\_idStatus` INT NOT NULL,

`Users\_IdUsers` INT NOT NULL,

`CreatedAt` DATETIME NOT NULL,

`UpdatedAt` VARCHAR(45) NOT NULL,

PRIMARY KEY (`idRepairStatus`, `Status\_idStatus`, `Users\_IdUsers`),

INDEX `fk\_RepairStatus\_Status1\_idx` (`Status\_idStatus` ASC),

INDEX `fk\_RepairStatus\_Users1\_idx` (`Users\_IdUsers` ASC),

CONSTRAINT `fk\_RepairStatus\_Status1`

FOREIGN KEY (`Status\_idStatus`)

REFERENCES `mydb`.`Status` (`idStatus`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_RepairStatus\_Users1`

FOREIGN KEY (`Users\_IdUsers`)

REFERENCES `mydb`.`Users` (`IdUsers`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;