|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |

Институт искусственного интеллекта

Кафедра автоматических систем

**Отчёт**

по дисциплине: **Проектирование информационных систем и баз данных реального времени**

|  |  |
| --- | --- |
| **Студент группы:** КВБО-05-21 | Обухова А. М.**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
| **Преподаватель:** | Савиных Н. В.**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |

|  |  |
| --- | --- |
| Работа представлена к защите: | «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2023 г. |

Москва 2023

Содержание

Введение

1. Создание логической модели данных
2. Создание физической модели данных
3. Подключение к серверу и работа с БД(Практика SQL 1)
4. Разработка процедур, функций, тригеров(Практика SQL 3)
5. Создание пользовательского приложения

Заключение

ПРИЛОЖЕНИЕ А

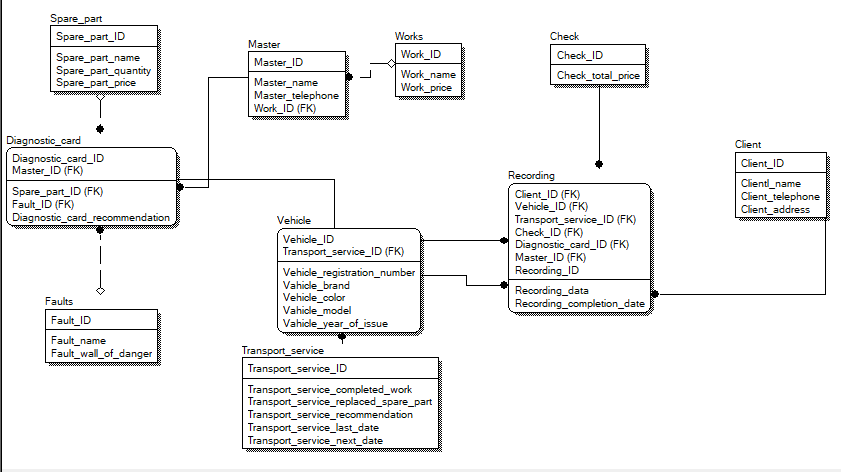
ПРИЛОЖЕНИЕ Б

**Введение**

В качестве темы для выполнения работы была выбрынна база данных станции технического обслуживания автомобилей(далее СТО). Основными компонентами базы данных СТО будут клиент, его транспортное средство, карта технического обслуживания транспортного средства, мастер. Карта диагностики для мастера, чек, запись, запчасти, поломки. Остольные компоненты буду дополнять или связывать предыдущие. Расмотрим их подробнее на логической модели данных.

1. **Создание логической модели данных**

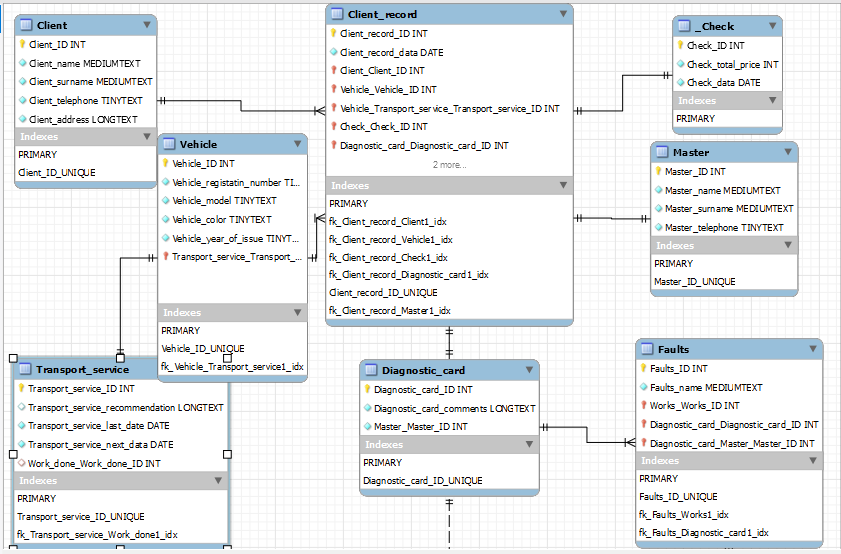
Создание логической модели данных было произвено в приложении Erwin и имеет следующий вид

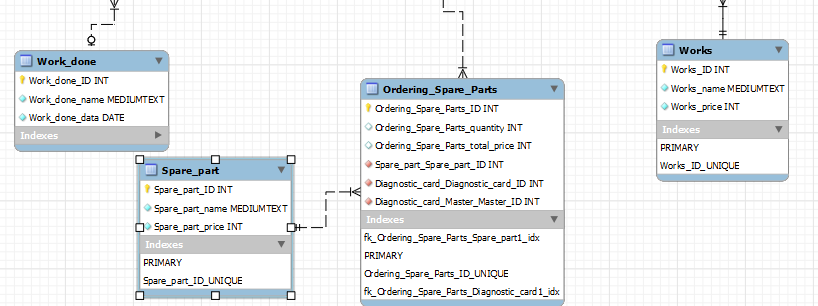


Полученная логическая модель имеет 3 нормальную форму. Для дальнейшего удобства и выполнения предстоящей работы, данная логическая модель данных была перенесена в программу MySQL Workbench и разнаботана на ее основе физическая модель данных.

1. **Создание физической модели данных**

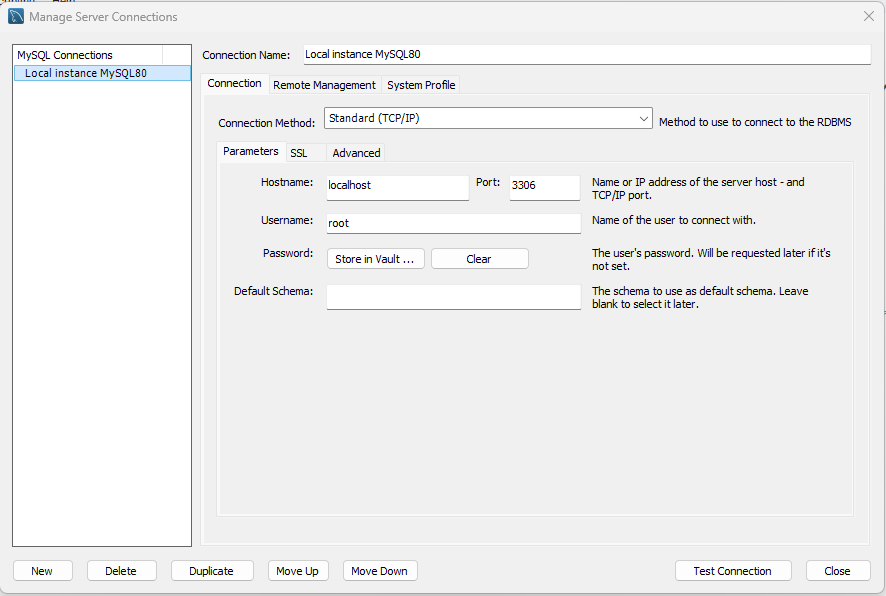
В ходе разроботки физической модели данных была немного изменена структура схемы для дальнейшего удобства работы с базой данных





1. **Подключение к серверу и работа с БД**

Подключение к локальному серверу производилось также на основе MySQL Workbench. Создадим локальный сервер:



Импортируем в него сгенерированный Workbench скрипт физической модели данных (ПРИЛОЖЕНИЕ А).

1. **Разработка процедур, функций, тригеров(Практика SQL 3)**

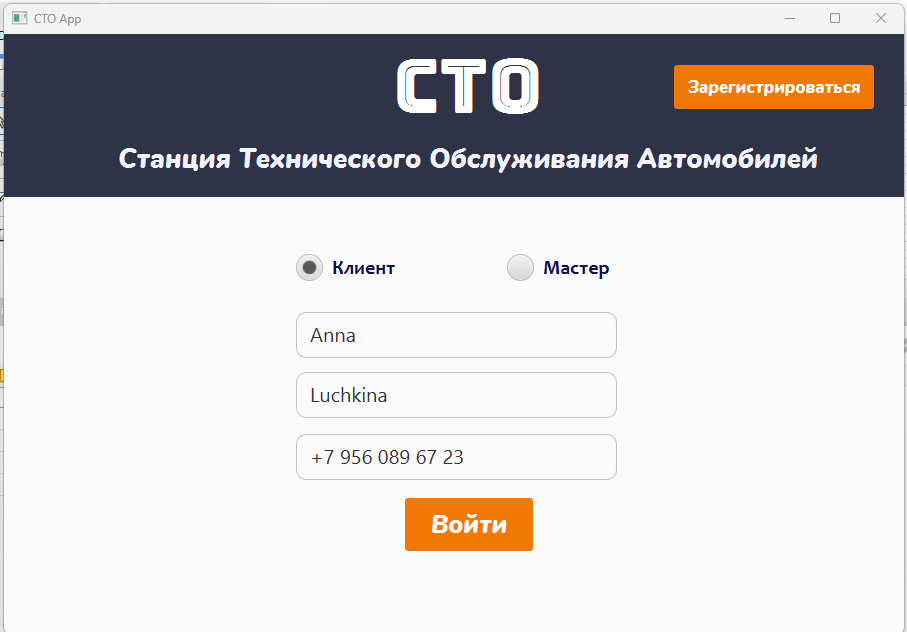
В ходе работы были разработаны следующие функции, процедуры и тригеры:

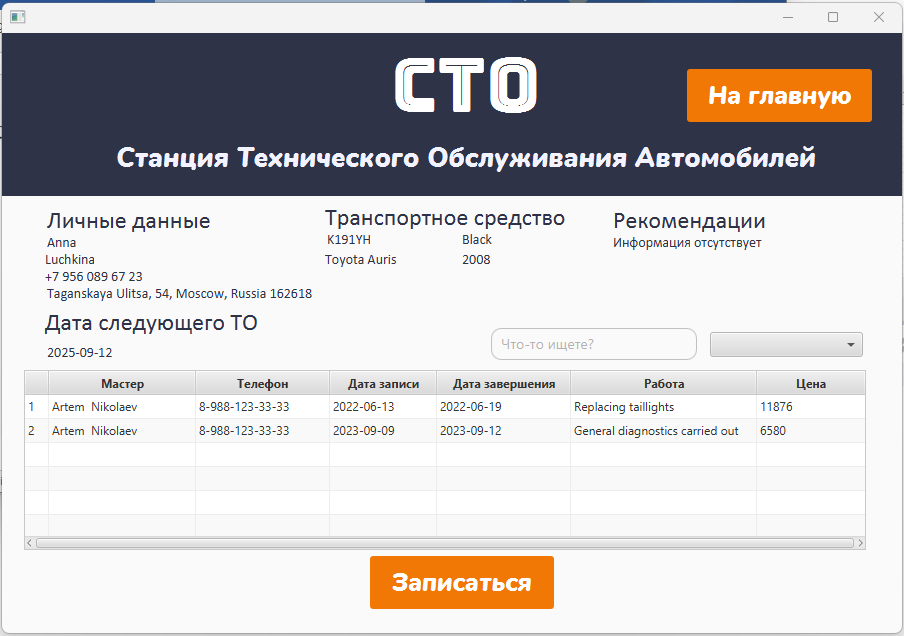
* Функция SparePartTotalPrice – вычисляет полную стоимость всех используемых запчастей с условием используемого количества и цены за одну деталь
* Функция TransportServiceNextData – определяет дату следующего технического обслуживания с учётом даты предыдущего ТО.
* Функция AllMaster – возвращает полную информацию о каждом мастере
* Процедура get\_transport\_service\_diagnostic – обновляет данные таблицы “Транспортное обслуживание”, если мастер провёл диагностическое обследование
* Процедура get\_work\_done – обновляет таблицу “Выполненые работы”, если мастер выполнил таковую(любую другую процедуры, кроме диагностики)
* Процедура get\_check\_price – вычисляет общую стоимость обслуживания клиента
* Процедура del – удаляет последних добавленных клиента и его транспортное средство
* Тригер SetNextData – вызывает функцию TransportServiceNextData
* Тригер SetCheck – вычисляет дату составления чека и вызывает процедуру get\_check\_price
* Тригер TransportServiceDiagnostic – вызывает процедуру get\_transport\_service\_diagnostic
* Тригер WorkDone – вызывает процедуру get\_work\_done

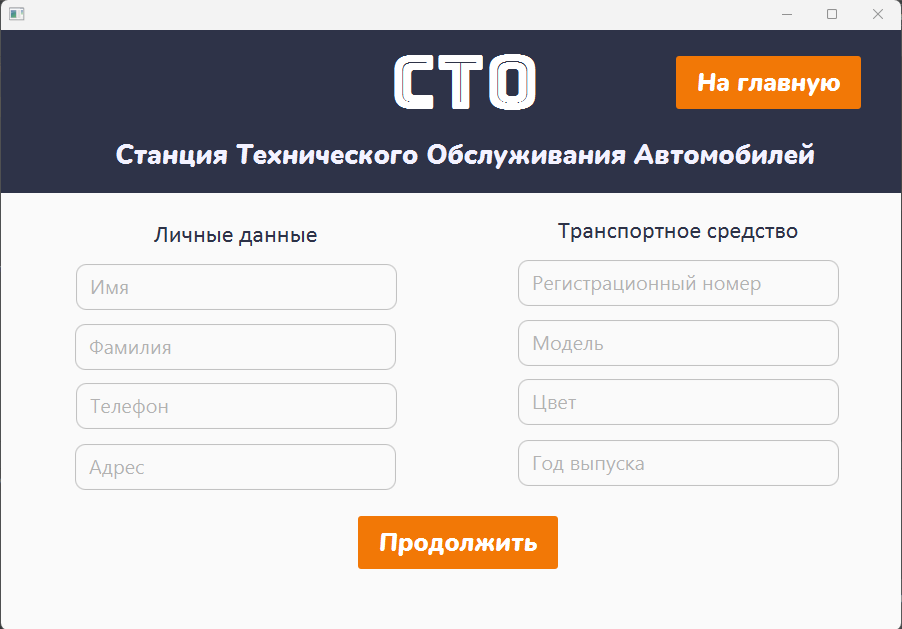
Полный представление функций, процедур и тригеров представлен в ПРИЛОЖЕНИИ Б

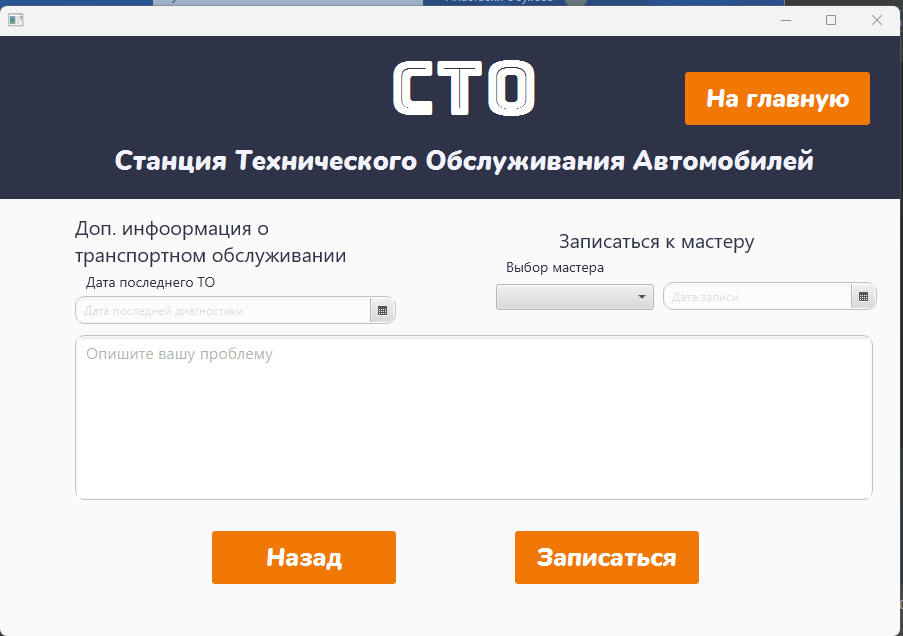
1. **Создание пользовательского приложение.**

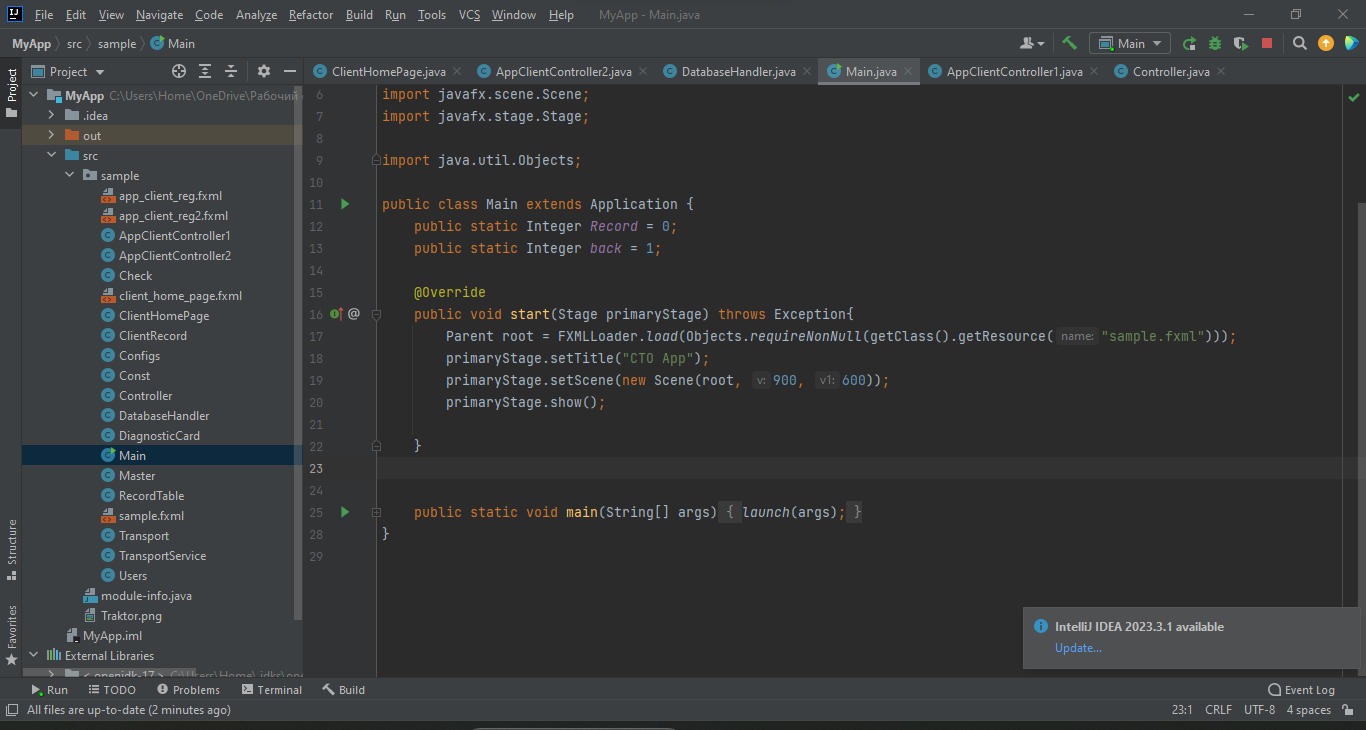
Приложение было написано на языке JAVA с помощью библиотеки JavaFX. Интерфейс был разработа в SceneBuilder. В приложении реализованны основные функции: вход пользователя, регистрация пользователя, поиск информации пользователя, добавление новой записи уже существующего пользователя.











**Заключение**

В ходе проделанной работы была разработана база данных СТО, создана логическая и физеческая модель данных, подключён сервер в MySQL WorkBench, написаны процедуры, функции тригеры и разработано пользовательское приложение с основным функционалом.

ПРИЛОЖЕНИЕ А

-- MySQL Script generated by MySQL Workbench

-- Fri Sep 29 09:32:58 2023

-- 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='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

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

-- Schema MyBD

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

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

-- Schema MyBD

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

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

USE `MyBD` ;

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

-- Table `MyBD`.`Check`

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

CREATE TABLE IF NOT EXISTS `MyBD`.`Check` (

`Check\_ID` INT NOT NULL,

`Check\_total\_price` INT NOT NULL,

`Check\_data` DATETIME NOT NULL,

PRIMARY KEY (`Check\_ID`))

ENGINE = InnoDB;

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

-- Table `MyBD`.`Client`

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

CREATE TABLE IF NOT EXISTS `MyBD`.`Client` (

`Client\_ID` INT NOT NULL,

`Client\_name` MEDIUMTEXT NOT NULL,

`Client\_telephone` INT NOT NULL,

`Client\_address` LONGTEXT NOT NULL,

PRIMARY KEY (`Client\_ID`),

UNIQUE INDEX `Client\_ID\_UNIQUE` (`Client\_ID` ASC) VISIBLE)

ENGINE = InnoDB;

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

-- Table `MyBD`.`Transport\_service`

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

CREATE TABLE IF NOT EXISTS `MyBD`.`Transport\_service` (

`Transport\_service\_ID` INT NOT NULL,

`Transport\_service\_completed\_work` LONGTEXT NOT NULL,

`Transport\_service\_replaced\_spare\_part` LONGTEXT NULL,

`Transport\_service\_recommendation` LONGTEXT NULL,

`Transport\_service\_last\_date` DATETIME NOT NULL,

`Transport\_service\_next\_data` DATETIME NOT NULL,

PRIMARY KEY (`Transport\_service\_ID`),

UNIQUE INDEX `Transport\_service\_ID\_UNIQUE` (`Transport\_service\_ID` ASC) VISIBLE)

ENGINE = InnoDB;

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

-- Table `MyBD`.`Vehicle`

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

CREATE TABLE IF NOT EXISTS `MyBD`.`Vehicle` (

`Vehicle\_ID` INT NOT NULL,

`Vehicle\_registatin\_number` TINYTEXT NOT NULL,

`Vehicle\_brand` TINYTEXT NOT NULL,

`Vehicle\_color` TINYTEXT NOT NULL,

`Vehicle\_model` TINYTEXT NOT NULL,

`Vehicle\_year\_of\_issue` TINYTEXT NOT NULL,

`Transport\_service\_Transport\_service\_ID` INT NOT NULL,

PRIMARY KEY (`Vehicle\_ID`, `Transport\_service\_Transport\_service\_ID`),

UNIQUE INDEX `Vehicle\_registatin\_number\_UNIQUE` (`Vehicle\_registatin\_number` ASC) VISIBLE,

UNIQUE INDEX `Vehicle\_ID\_UNIQUE` (`Vehicle\_ID` ASC) VISIBLE,

INDEX `fk\_Vehicle\_Transport\_service1\_idx` (`Transport\_service\_Transport\_service\_ID` ASC) VISIBLE,

CONSTRAINT `fk\_Vehicle\_Transport\_service1`

FOREIGN KEY (`Transport\_service\_Transport\_service\_ID`)

REFERENCES `MyBD`.`Transport\_service` (`Transport\_service\_ID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

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

-- Table `MyBD`.`Diagnostic\_card`

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

CREATE TABLE IF NOT EXISTS `MyBD`.`Diagnostic\_card` (

`Diagnostic\_card\_ID` INT NOT NULL,

`Diagnostic\_card\_quanity\_spare\_part` INT NULL,

`Master\_Master\_ID` INT NOT NULL,

PRIMARY KEY (`Diagnostic\_card\_ID`, `Master\_Master\_ID`),

UNIQUE INDEX `Diagnostic\_card\_ID\_UNIQUE` (`Diagnostic\_card\_ID` ASC) VISIBLE)

ENGINE = InnoDB;

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

-- Table `MyBD`.`Master`

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

CREATE TABLE IF NOT EXISTS `MyBD`.`Master` (

`Master\_ID` INT NOT NULL,

`Master\_name` MEDIUMTEXT NOT NULL,

`Master\_telephone` INT NOT NULL,

PRIMARY KEY (`Master\_ID`),

UNIQUE INDEX `Master\_ID\_UNIQUE` (`Master\_ID` ASC) VISIBLE)

ENGINE = InnoDB;

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

-- Table `MyBD`.`Client\_record`

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

CREATE TABLE IF NOT EXISTS `MyBD`.`Client\_record` (

`Client\_record\_ID` INT NOT NULL,

`Client\_record\_data` DATETIME NOT NULL,

`Client\_completion\_date` DATETIME NOT NULL,

`Client\_Client\_ID` INT NOT NULL,

`Vehicle\_Vehicle\_ID` INT NOT NULL,

`Vehicle\_Transport\_service\_Transport\_service\_ID` INT NOT NULL,

`Check\_Check\_ID` INT NOT NULL,

`Diagnostic\_card\_Diagnostic\_card\_ID` INT NOT NULL,

`Master\_Master\_ID` INT NOT NULL,

PRIMARY KEY (`Client\_record\_ID`, `Client\_Client\_ID`, `Vehicle\_Vehicle\_ID`, `Vehicle\_Transport\_service\_Transport\_service\_ID`, `Check\_Check\_ID`, `Diagnostic\_card\_Diagnostic\_card\_ID`, `Master\_Master\_ID`),

INDEX `fk\_Client\_record\_Client1\_idx` (`Client\_Client\_ID` ASC) VISIBLE,

INDEX `fk\_Client\_record\_Vehicle1\_idx` (`Vehicle\_Vehicle\_ID` ASC, `Vehicle\_Transport\_service\_Transport\_service\_ID` ASC) VISIBLE,

INDEX `fk\_Client\_record\_Check1\_idx` (`Check\_Check\_ID` ASC) VISIBLE,

INDEX `fk\_Client\_record\_Diagnostic\_card1\_idx` (`Diagnostic\_card\_Diagnostic\_card\_ID` ASC) VISIBLE,

UNIQUE INDEX `Client\_record\_ID\_UNIQUE` (`Client\_record\_ID` ASC) VISIBLE,

INDEX `fk\_Client\_record\_Master1\_idx` (`Master\_Master\_ID` ASC) VISIBLE,

CONSTRAINT `fk\_Client\_record\_Client1`

FOREIGN KEY (`Client\_Client\_ID`)

REFERENCES `MyBD`.`Client` (`Client\_ID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Client\_record\_Vehicle1`

FOREIGN KEY (`Vehicle\_Vehicle\_ID` , `Vehicle\_Transport\_service\_Transport\_service\_ID`)

REFERENCES `MyBD`.`Vehicle` (`Vehicle\_ID` , `Transport\_service\_Transport\_service\_ID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Client\_record\_Check1`

FOREIGN KEY (`Check\_Check\_ID`)

REFERENCES `MyBD`.`Check` (`Check\_ID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Client\_record\_Diagnostic\_card1`

FOREIGN KEY (`Diagnostic\_card\_Diagnostic\_card\_ID`)

REFERENCES `MyBD`.`Diagnostic\_card` (`Diagnostic\_card\_ID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Client\_record\_Master1`

FOREIGN KEY (`Master\_Master\_ID`)

REFERENCES `MyBD`.`Master` (`Master\_ID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

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

-- Table `MyBD`.`Works`

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

CREATE TABLE IF NOT EXISTS `MyBD`.`Works` (

`Works\_ID` INT NOT NULL,

`Works\_name` MEDIUMTEXT NOT NULL,

`Works\_price` INT NOT NULL,

PRIMARY KEY (`Works\_ID`),

UNIQUE INDEX `Works\_ID\_UNIQUE` (`Works\_ID` ASC) VISIBLE)

ENGINE = InnoDB;

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

-- Table `MyBD`.`Faults`

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

CREATE TABLE IF NOT EXISTS `MyBD`.`Faults` (

`Faults\_ID` INT NOT NULL,

`Faults\_name` MEDIUMTEXT NOT NULL,

`Faults\_wall\_of\_danger` INT NOT NULL,

`Works\_Works\_ID` INT NOT NULL,

`Diagnostic\_card\_Diagnostic\_card\_ID` INT NOT NULL,

`Diagnostic\_card\_Master\_Master\_ID` INT NOT NULL,

PRIMARY KEY (`Faults\_ID`, `Works\_Works\_ID`, `Diagnostic\_card\_Diagnostic\_card\_ID`, `Diagnostic\_card\_Master\_Master\_ID`),

UNIQUE INDEX `Faults\_ID\_UNIQUE` (`Faults\_ID` ASC) VISIBLE,

INDEX `fk\_Faults\_Works1\_idx` (`Works\_Works\_ID` ASC) VISIBLE,

INDEX `fk\_Faults\_Diagnostic\_card1\_idx` (`Diagnostic\_card\_Diagnostic\_card\_ID` ASC, `Diagnostic\_card\_Master\_Master\_ID` ASC) VISIBLE,

CONSTRAINT `fk\_Faults\_Works1`

FOREIGN KEY (`Works\_Works\_ID`)

REFERENCES `MyBD`.`Works` (`Works\_ID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Faults\_Diagnostic\_card1`

FOREIGN KEY (`Diagnostic\_card\_Diagnostic\_card\_ID` , `Diagnostic\_card\_Master\_Master\_ID`)

REFERENCES `MyBD`.`Diagnostic\_card` (`Diagnostic\_card\_ID` , `Master\_Master\_ID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

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

-- Table `MyBD`.`Spare\_part`

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

CREATE TABLE IF NOT EXISTS `MyBD`.`Spare\_part` (

`Spare\_part\_ID` INT NOT NULL,

`Spare\_part\_name` MEDIUMTEXT NOT NULL,

`Spare\_part\_price` DOUBLE NOT NULL,

PRIMARY KEY (`Spare\_part\_ID`),

UNIQUE INDEX `Spare\_part\_ID\_UNIQUE` (`Spare\_part\_ID` ASC) VISIBLE,

UNIQUE INDEX `Spare\_part\_name\_UNIQUE` (`Spare\_part\_name` ASC) VISIBLE)

ENGINE = InnoDB;

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

-- Table `MyBD`.`Ordering\_Spare\_Parts`

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

CREATE TABLE IF NOT EXISTS `MyBD`.`Ordering\_Spare\_Parts` (

`Ordering\_Spare\_Parts\_ID` INT NOT NULL,

`Ordering\_Spare\_Parts\_quantity` INT NULL,

`Ordering\_Spare\_Parts\_total\_price` INT NULL,

`Spare\_part\_Spare\_part\_ID` INT NOT NULL,

`Diagnostic\_card\_Diagnostic\_card\_ID` INT NOT NULL,

`Diagnostic\_card\_Master\_Master\_ID` INT NOT NULL,

INDEX `fk\_Ordering\_Spare\_Parts\_Spare\_part1\_idx` (`Spare\_part\_Spare\_part\_ID` ASC) VISIBLE,

PRIMARY KEY (`Ordering\_Spare\_Parts\_ID`),

UNIQUE INDEX `Ordering\_Spare\_Parts\_ID\_UNIQUE` (`Ordering\_Spare\_Parts\_ID` ASC) VISIBLE,

INDEX `fk\_Ordering\_Spare\_Parts\_Diagnostic\_card1\_idx` (`Diagnostic\_card\_Diagnostic\_card\_ID` ASC, `Diagnostic\_card\_Master\_Master\_ID` ASC) VISIBLE,

CONSTRAINT `fk\_Ordering\_Spare\_Parts\_Spare\_part1`

FOREIGN KEY (`Spare\_part\_Spare\_part\_ID`)

REFERENCES `MyBD`.`Spare\_part` (`Spare\_part\_ID`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_Ordering\_Spare\_Parts\_Diagnostic\_card1`

FOREIGN KEY (`Diagnostic\_card\_Diagnostic\_card\_ID` , `Diagnostic\_card\_Master\_Master\_ID`)

REFERENCES `MyBD`.`Diagnostic\_card` (`Diagnostic\_card\_ID` , `Master\_Master\_ID`)

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;

ПРИЛОЖЕНИЕ Б

DELIMITER //

CREATE TRIGGER SetNextData before INSERT ON transport\_service

FOR EACH ROW

BEGIN

call TransportServiceNextData(transport\_service\_last\_date);

END//

DELIMITER ;

DELIMITER //

CREATE TRIGGER SetCheck before INSERT ON \_check

FOR EACH ROW

BEGIN

SET New.check\_data = NOW();

call get\_check\_price();

END//

DELIMITER ;

DELIMITER //

CREATE TRIGGER TransportServiceDiagnostic before INSERT ON transport\_service

FOR EACH ROW

BEGIN

call get\_transport\_service\_diagnostic;

END//

DELIMITER ;

DELIMITER //

CREATE TRIGGER WorkDone before INSERT ON work\_done

FOR EACH ROW

BEGIN

call get\_work\_done;

END//

DELIMITER ;

DELIMITER //

CREATE PROCEDURE get\_transport\_service\_diagnostic()

BEGIN

update transport\_service

join client\_record on Transport\_service\_ID = Vehicle\_Transport\_service\_Transport\_service\_ID

join diagnostic\_card on Diagnostic\_card\_ID = client\_record.Diagnostic\_card\_Diagnostic\_card\_ID

join faults on Diagnostic\_card\_ID =faults.Diagnostic\_card\_Diagnostic\_card\_ID

set transport\_service\_last\_date = NOW(), transport\_service\_next\_data = TransportServiceNextData (NOW()) where works\_works\_id = 40001;

END//

DELIMITER ;

DELIMITER //

CREATE PROCEDURE get\_work\_done()

BEGIN

update work\_done

join transport\_service on work\_done\_id = work\_done\_work\_done\_id

join client\_record on Transport\_service\_ID = Vehicle\_Transport\_service\_Transport\_service\_ID

join diagnostic\_card on Diagnostic\_card\_ID = client\_record.Diagnostic\_card\_Diagnostic\_card\_ID

join faults on Diagnostic\_card\_ID =faults.Diagnostic\_card\_Diagnostic\_card\_ID

join works on works.works\_id = faults.works\_works\_id

set work\_done\_data = NOW(), work\_done.Work\_done\_name = works.works\_name where works\_works\_id != 40001;

END//

DELIMITER ;

DELIMITER //

CREATE PROCEDURE get\_check\_price()

BEGIN

update \_check

join client\_record on check\_id = check\_check\_id

join diagnostic\_card on Diagnostic\_card\_ID = client\_record.Diagnostic\_card\_Diagnostic\_card\_ID

join faults on Diagnostic\_card\_ID =faults.Diagnostic\_card\_Diagnostic\_card\_ID

join works on works\_id = works\_works\_id

join ordering\_spare\_parts on Diagnostic\_card\_ID = ordering\_spare\_parts.Diagnostic\_card\_diagnostic\_card\_id

join spare\_part on spare\_part\_id = spare\_part\_spare\_part\_id

set check\_total\_price = works\_price + SparePartTotalPrice(Ordering\_Spare\_Parts\_quantity, spare\_part\_price);

END//

DELIMITER ;

DELIMITER //

CREATE FUNCTION SparePartTotalPrice ( Quantity int, PartPrice int) #Общая цена запластей

returns int

deterministic

begin

declare TotalPrice int;

set TotalPrice = Quantity\*PartPrice;

return TotalPrice;

end //

DELIMITER ;

DELIMITER //

CREATE FUNCTION TransportServiceNextData (last\_data DATE)#Дата следующего ТО

returns int

deterministic

begin

declare next\_data DATE;

set next\_data = TIMESTAMPADD(year,3,last\_data);

return next\_data;

end //

DELIMITER ;

DELIMITER //

CREATE FUNCTION AllMaster (Master\_name text, Master\_surname text, Master\_telephone text)

returns text

deterministic

begin

declare All\_Master text;

set All\_Master= Master\_name+" "+Master\_surname+" "+Master\_telephone;

return All\_Master;

end //

DELIMITER ;

DELIMITER //

CREATE PROCEDURE del()

BEGIN

DELETE FROM vehicle ORDER BY vehicle\_id DESC LIMIT 1 ;

DELETE FROM client ORDER BY client\_id DESC LIMIT 1 ;

END//

DELIMITER ;