ГУАП

КАФЕДРА № 41

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

Старший преподаватель		Б.К. Акопян
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №2

СОЗДАНИЕ БАЗЫ ДАННЫХ СРЕДСТВАМИ ГРАФИЧЕСКОГО ИНТЕРФЕЙСА MYSQL WORKBENCH

по курсу: БАЗЫ ДАННЫХ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4217		Д.М. Никитин
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2024

1. **Цель работы:** даталогическое проектирование реляционной БД в среде MySQL Workbech, физическая реализация БД на основе модели данных, заполнение БД данными.

2. Последовательность выполнения работы:

1. Внимательно прочитать методические указания.

2. Выполнить упражнения: – по даталогическому проектированию БД в среде MySQL Workbench, – физической реализации БД на сервере на основе модели БД, – заполнению данными БД.

3. На основании своего варианта реализовать БД на сервере, ориентируясь на этапы разработки БД по упражнениям. Выполнение каждого этапа необходимо зафиксировать на 3-7 скриншотах. Количество записей в каждой таблице от 3 до 5. Задание по лабораторной работе выполняется на основании трех таблиц Приложения А. Структура первой и второй таблиц варианта вам знакома. По этим таблицам в первой лабораторной работе были реализованы две БД. В данной работе эти справочные таблицы входят в состав реляционной БД. Третья таблица соединяет их между собой связью «один ко многим» и, по сути, является журналом учета операций. 17

4. Выполните отчет в соответствие с требованиями ГОСТ 7.32-2017 и ГОСТ 2.105-2019: по оформлению отчетов (https://guap.ru/standart/doc).

3. **Вариант 24:**

1. Сведения о водителе (Код водителя, Фамилия, Имя, Номер водительского удостоверения).

2. Автомашина (Код машины, Марка, Номер машины, Год выпуска).

3. Дорожные происшествия (Код события, Код машины, Код водителя, Дата происшествия, Сумма ущерба).

4. **Ход выполнения работы:**

4.1. **Даталогическое проектирование БД:**

Первым делом открываем MySQL Workbench 8.0, подключимся к серверу и создадим новую модель БД и изменим её название на transportation.

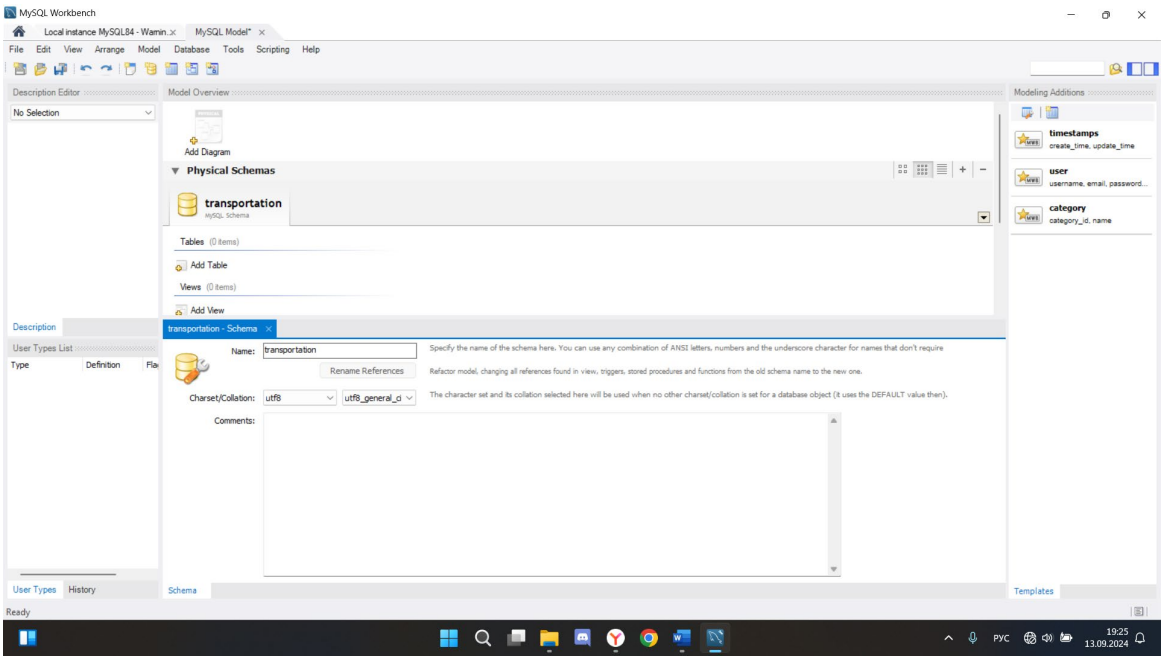

Рисунок 1 – Создание модели БД

После этого приступим к построению EER диаграммы.

Создадим таблицу drivers с полями driver_id, family, name, license_id.

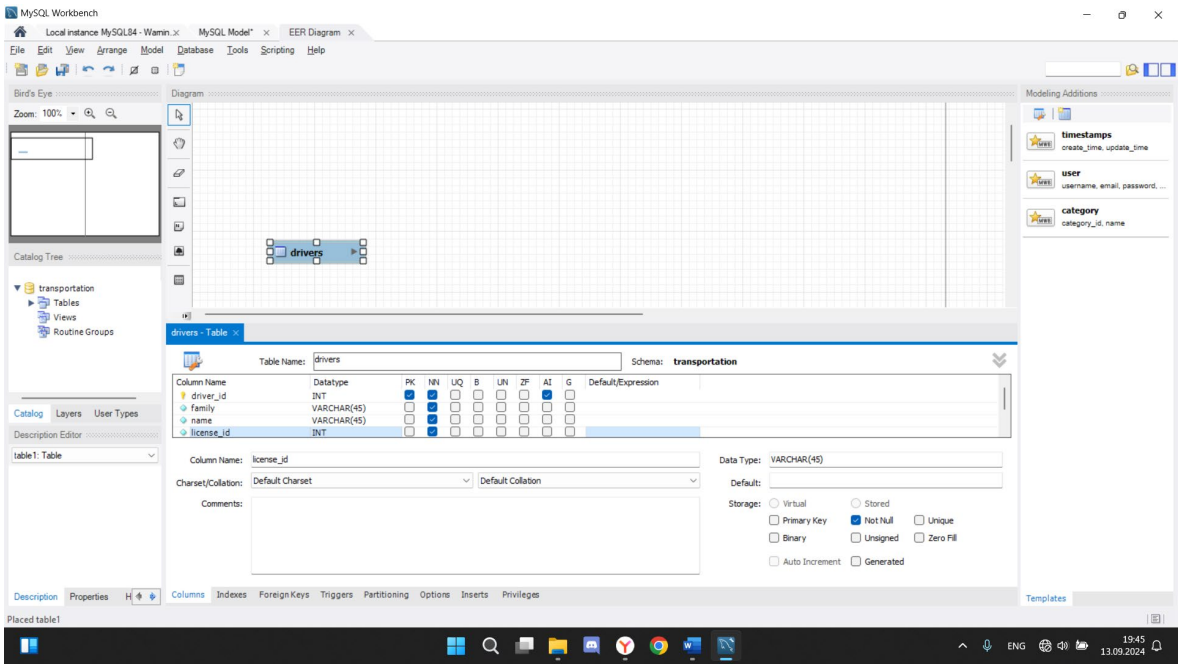

Рисунок 2 – Таблица drivers

Далее была создана таблица cars с полями car_id, brand, number и year.

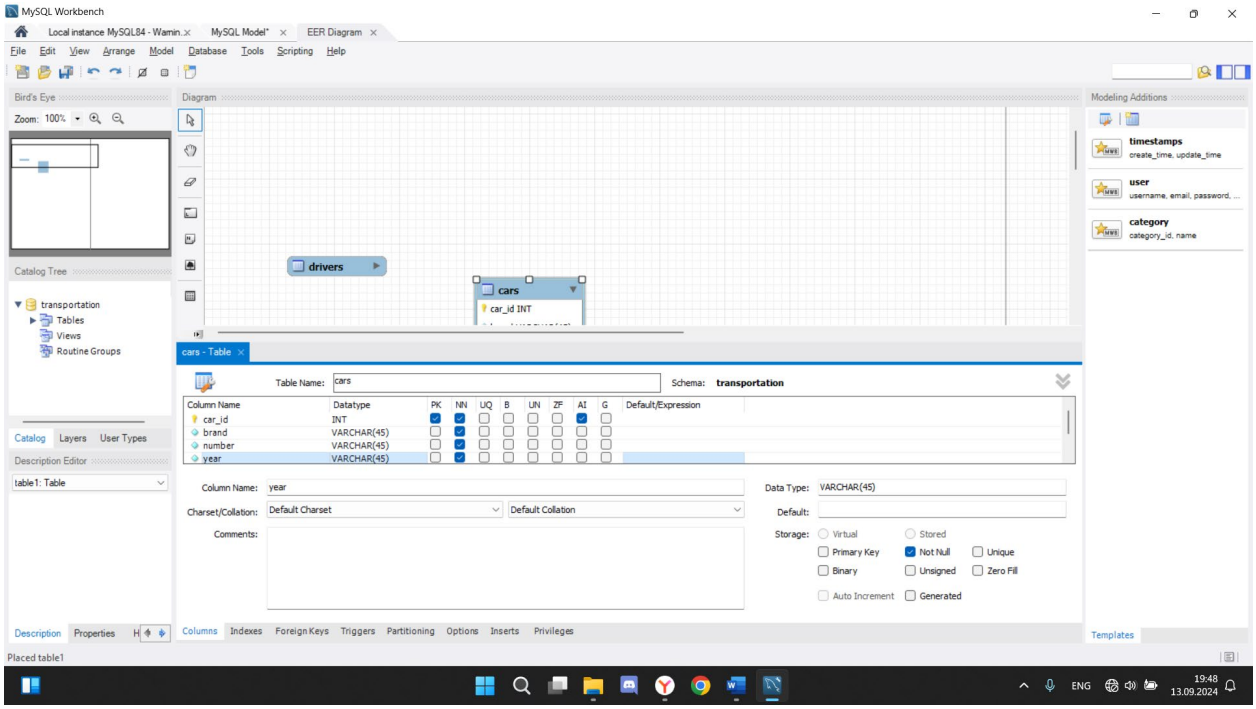

Рисунок 3 – Создание второй таблицы

После этого была создана третья таблица под названием accident с полями accident_id, car_id, driver_id, date и amount.

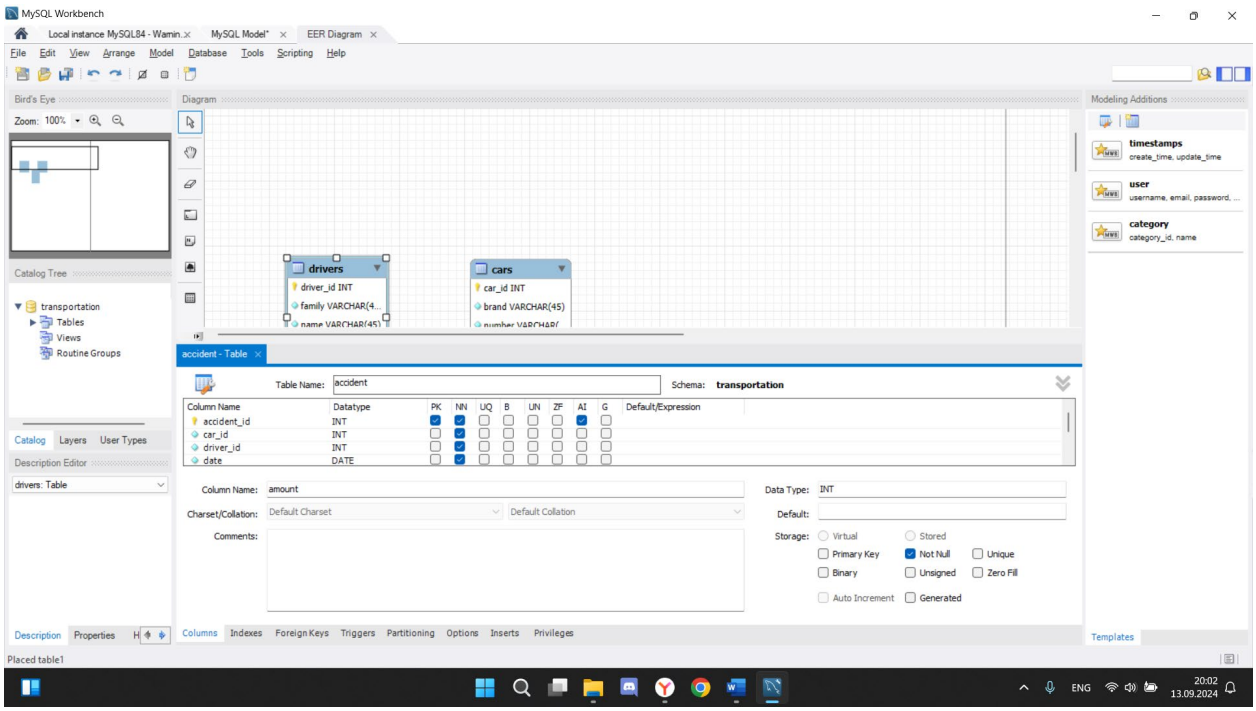

Рисунок 4 – Создание третьей таблицы

Далее были созданы внешние ключи. Поля car_id таблиц accidents и cars были связаны, а так же поля driver_id таблиц drivers и accidents.

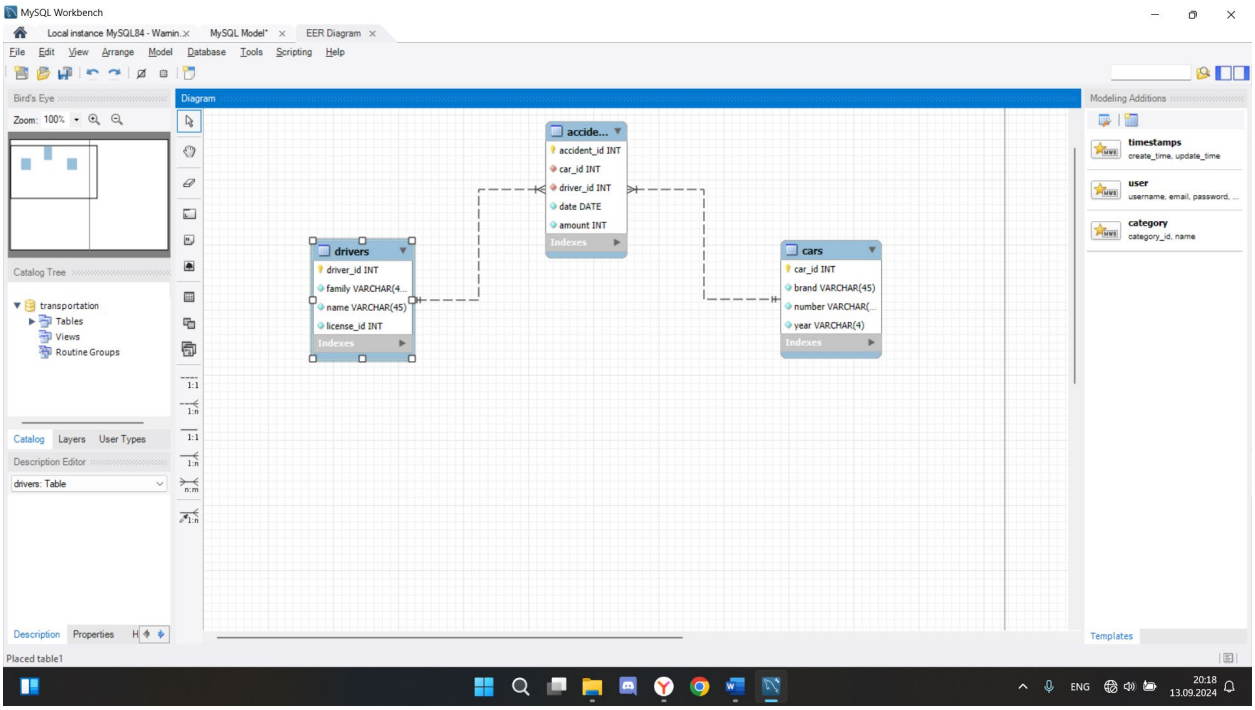

Рисунок 5 – Соединённые таблицы

После этого модель данных была сохранена в формате .mwb, .png и .svg.

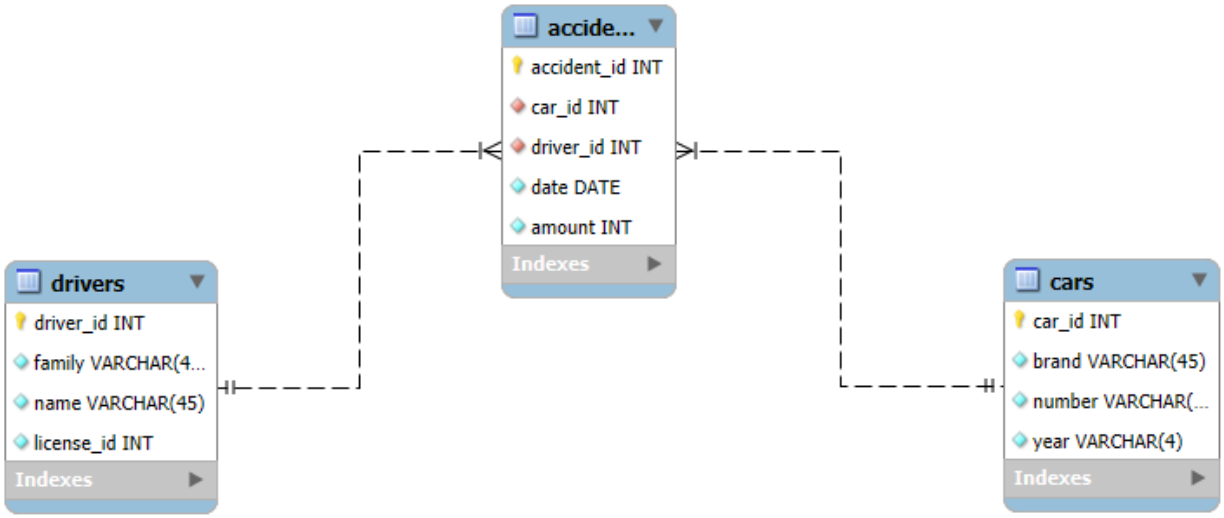

Рисунок 6 – Модель данных в формате .png

4.2. **Реализация БД на сервере на основе модели данных:**

Далее создадим SQL скрипт с помощью модели базы данных.

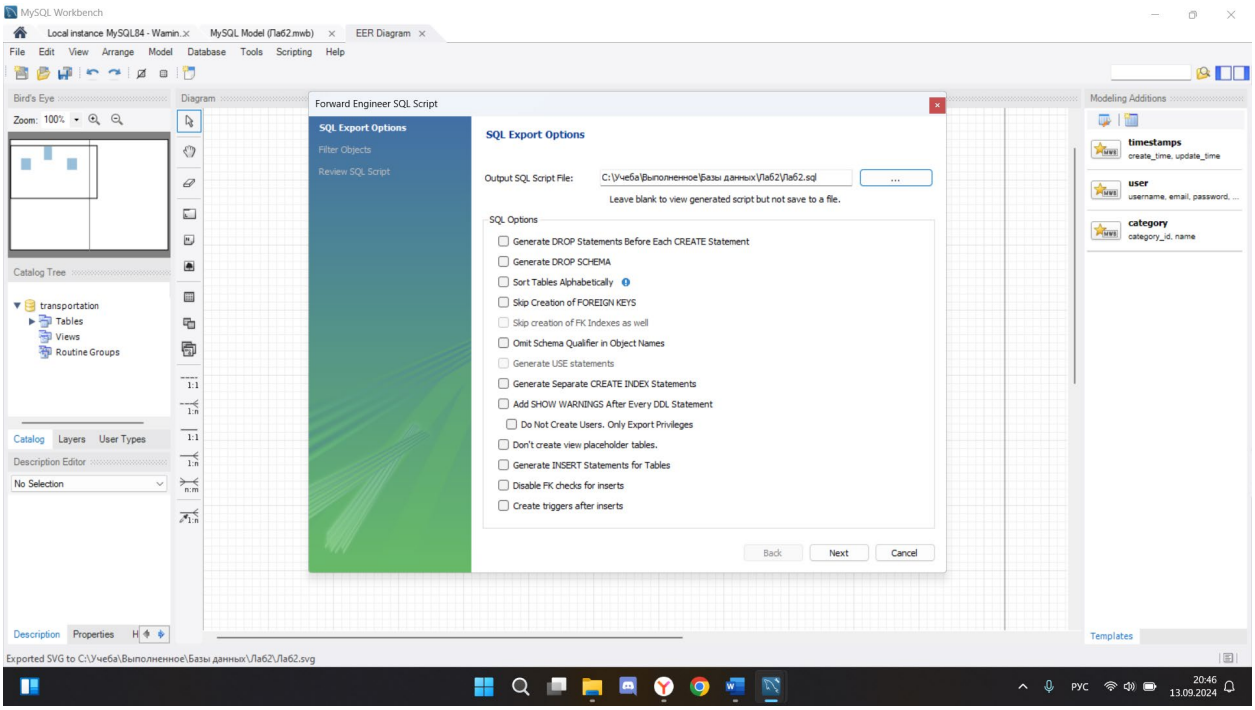

Рисунок 7 – Создание базы данных по модели

С помощью скрипта, сгенерированного автоматически по модели данных, создаём базу данных.

```
-- MySQL Script generated by MySQL Workbench
-- Fri Sep 13 20:48:29 2024
-- 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 transportation
-- -----------------------------------------------------

-- -----------------------------------------------------
-- Schema transportation
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `transportation` DEFAULT CHARACTER SET utf8
;
USE `transportation` ;

-- -----------------------------------------------------
-- Table `transportation`.`drivers`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `transportation`.`drivers` (
  `driver_id` INT NOT NULL AUTO_INCREMENT,
  `family` VARCHAR(45) NOT NULL,
  `name` VARCHAR(45) NOT NULL,
  `license_id` INT NOT NULL,
  PRIMARY KEY (`driver_id`))
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `transportation`.`cars`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `transportation`.`cars` (
  `car_id` INT NOT NULL AUTO_INCREMENT,
  `brand` VARCHAR(45) NOT NULL,
  `number` VARCHAR(9) NOT NULL,
  `year` VARCHAR(4) NOT NULL,
  PRIMARY KEY (`car_id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `transportation`.`accident`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `transportation`.`accident` (
  `accident_id` INT NOT NULL AUTO_INCREMENT,
  `car_id` INT NOT NULL,
  `driver_id` INT NOT NULL,
  `date` DATE NOT NULL,
  `amount` INT NOT NULL,
  PRIMARY KEY (`accident_id`),
  INDEX `driver_id_idx` (`driver_id` ASC) VISIBLE,
  INDEX `car_id_idx` (`car_id` ASC) VISIBLE,
  CONSTRAINT `driver_id`
    FOREIGN KEY (`driver_id`)
    REFERENCES `transportation`.`drivers` (`driver_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `car_id`
    FOREIGN KEY (`car_id`)
    REFERENCES `transportation`.`cars` (`car_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

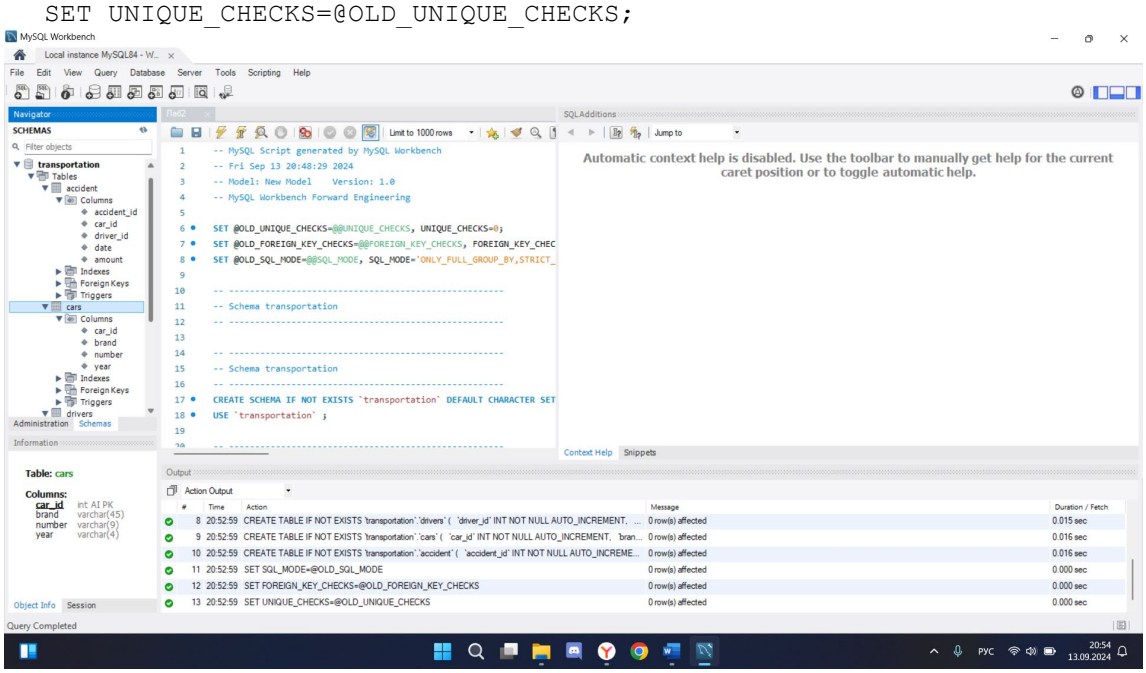

Рисунок 8 – Созданная база данных и часть скрипта

Проверяем таблицу с помощью Table Inspector.

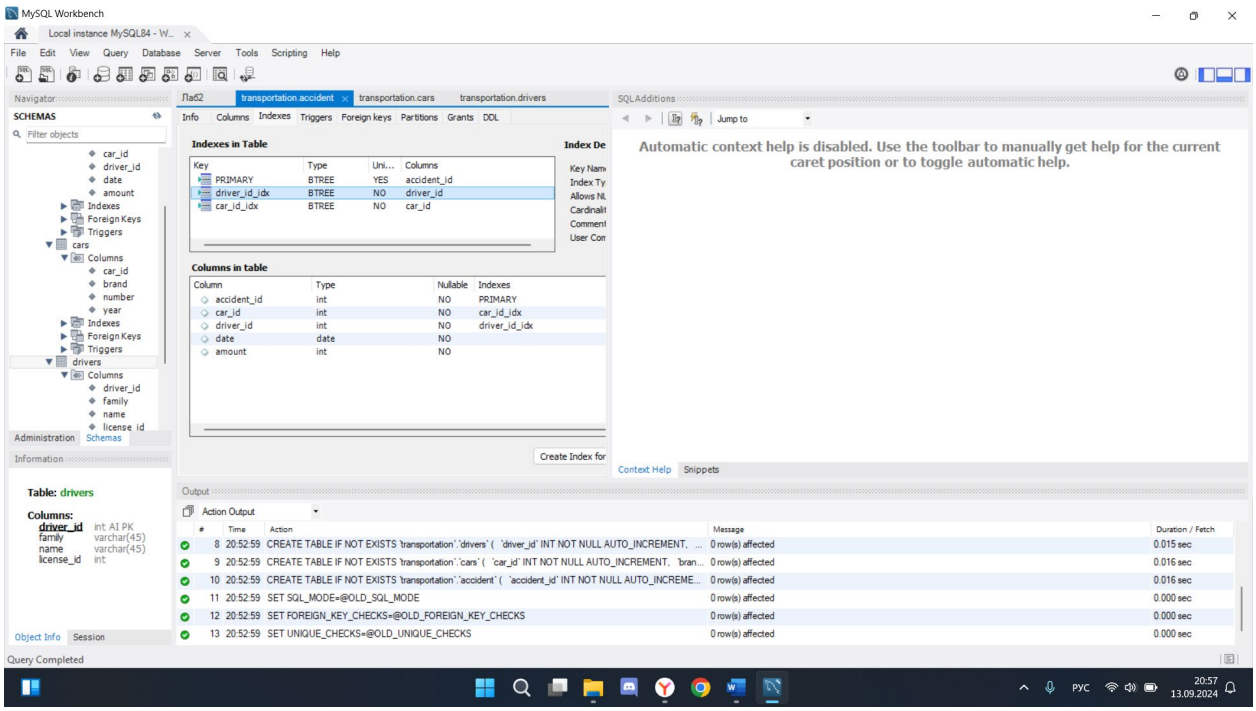

Рисунок 9 – Создана таблица, она верна

Далее просмотрим таблицу accident в табличном виде.

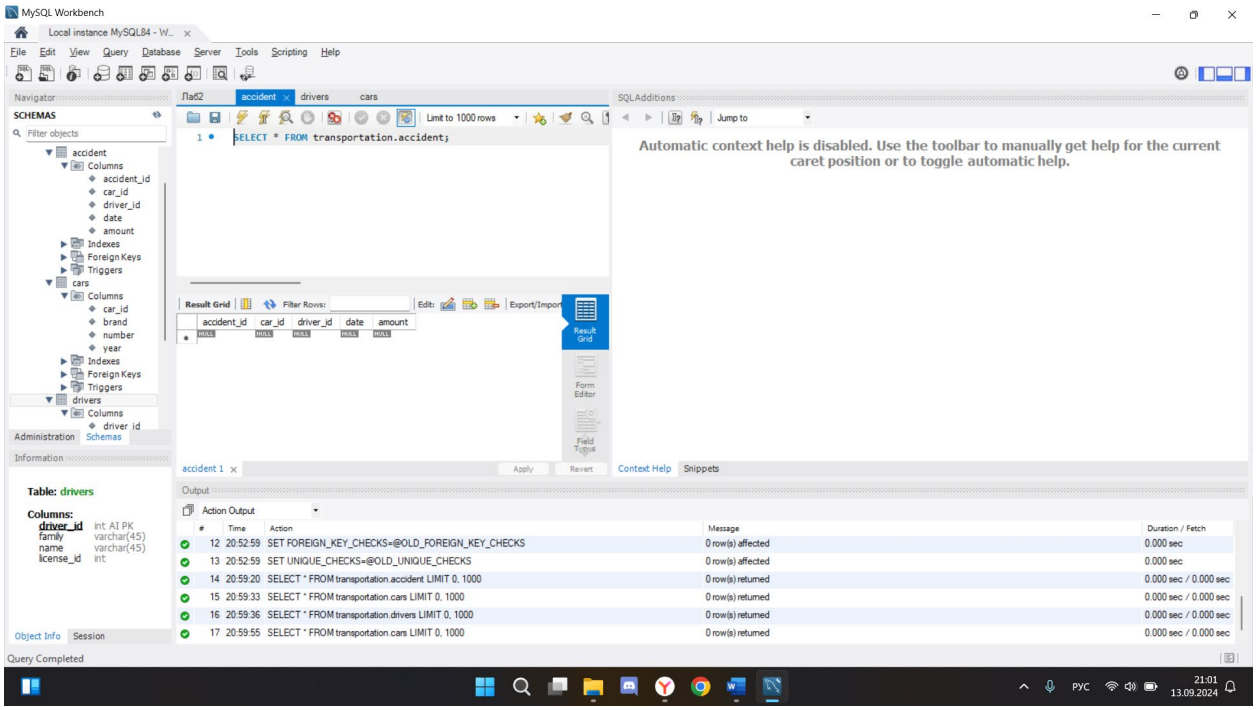

Рисунок 10 – Просмотр таблицы в табличном виде

4.3. **Заполнение данными БД:**

Первым делом заполним таблицу drivers с помощью скрипта.

```
USE transportation;
INSERT INTO drivers (family, name, license_id)
 VALUES ("Никитин", "Иван", 228),
```

```
 ("Поляков", "Влад", 123),
 ("Лебедев", "Влад", 339),
 ("Ярошенко", "Михайил", 555),
 ("Череховский", "Ярослав", 444);
SELECT * FROM drivers;
```

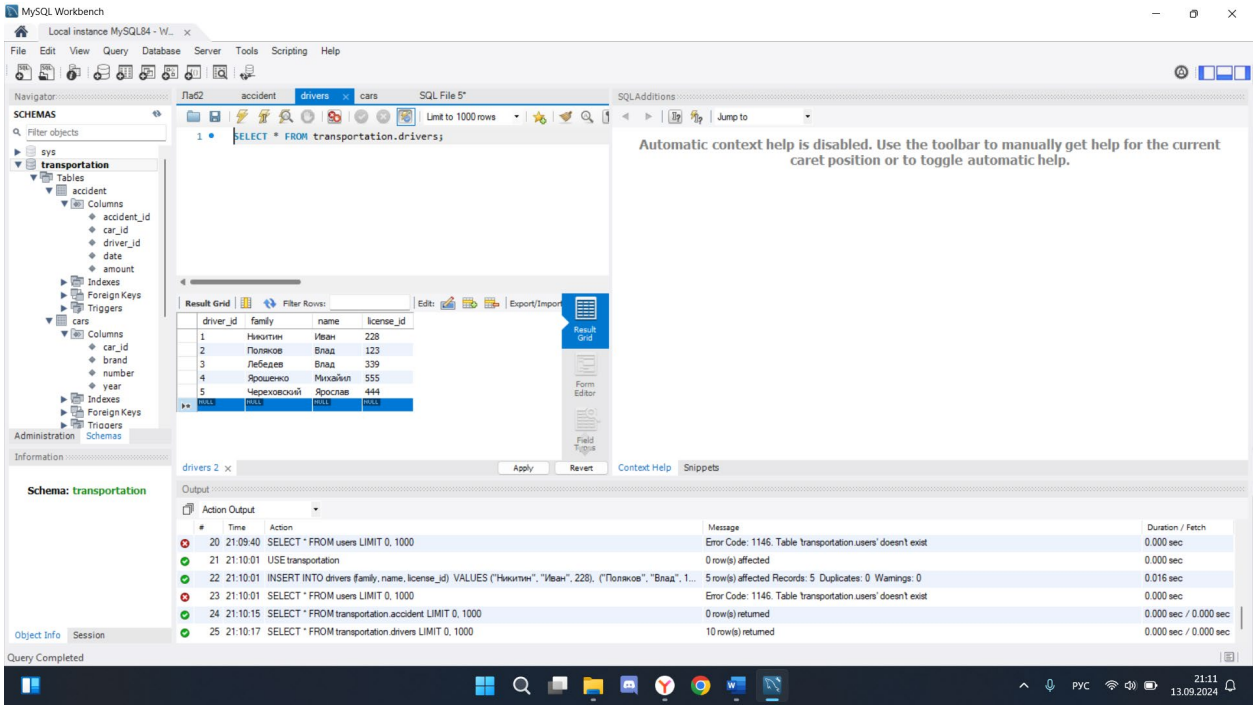

Рисунок 11 – Заполненная таблица drivers

Далее заполним таблицу cars с помощью скрипта.

```
USE transportation;
INSERT INTO cars (brand, number, year)
 VALUES ("Toyota Camry", "A123AA142", 2004),
 ("Lexus LX350", "C111CC142", 2008),
 ("Lada Granta", "O001OO142", 2024);
SELECT * FROM cars;
```

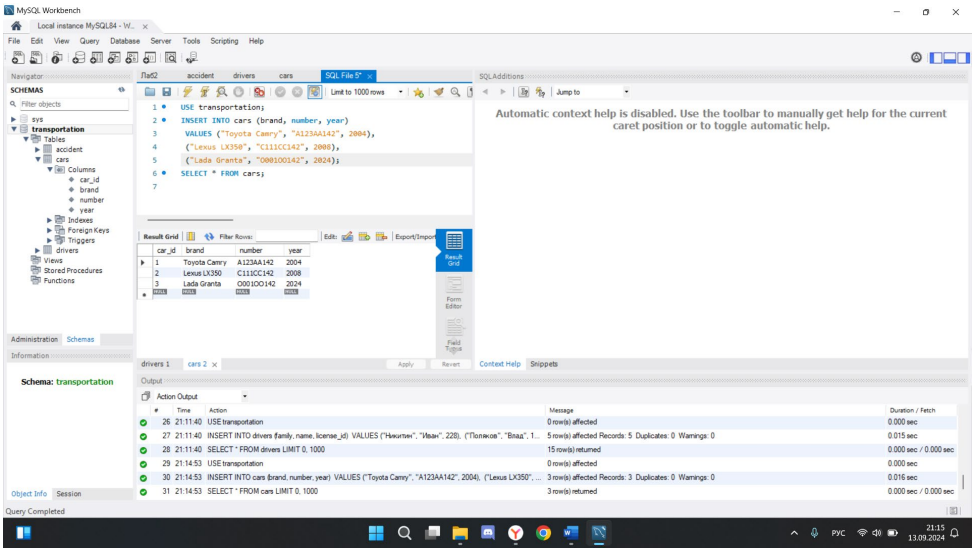

Рисунок 12 – Заполненная таблица cars

Далее заполним третью таблицу.

```
USE transportation;
INSERT INTO accident (car_id, driver_id, date, amount)
 VALUES (1, 1, "2008-10-10", 5000),
  (2, 4, "2007-11-12", 100000),
  (3, 4, "2024-10-11",  50000);
SELECT * FROM accident;
```

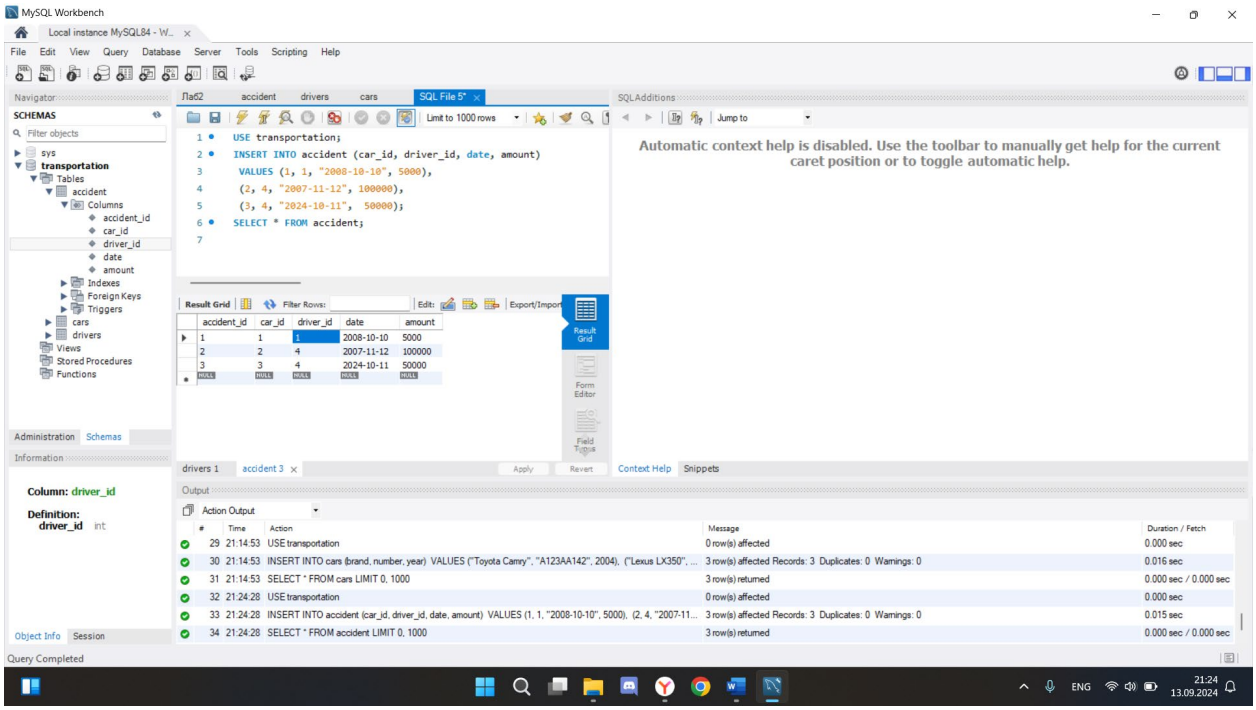

Рисунок 13 – Заполненная таблица accident

5. **Вывод:** в ходе лабораторной работы по созданию базы данных на MySQL-сервере была разработана база данных с тремя таблицами: "Водители" (drivers), "Машины" (cars) и "Происшествия" (accident). Определены атрибуты, ключи и связи между таблицами, создана ER-диаграмма. Сгенерированный SQL-скрипт был успешно выполнен на сервере, после чего таблицы были заполнены данными и проверена их корректность. Работа позволила освоить основные принципы проектирования и реализации реляционных баз данных, а также работу с MySQL Workbench.