*Федеральное государственное бюджетное образовательное учреждение высшего образования*

*«Владимирский государственный университет*

*имени Александра Григорьевича и Николая Григорьевича Столетовых»*

*Кафедра информационных систем и программной инженерии*

***КУРСОВОЙ ПРОЕКТ***

*Разработка структуры базы данных для*

*информационной системы «**Производство»*

*Выполнил: студент гр. ПРИ-120*

*Грачев Д. А.*

*Принял: доц. Вершинин В.В.*

*Владимир, 2022*

АНОТАЦИЯ

В данном курсовом проекте производилась разработка реляционной базы данных для информационной системы «Производство». Проект состоит из 4 этапов, включающих в себя описание предметной области, описание функционала, разработанный код и приложений, содержащих схемы данных. Для реализации ИС использовалась СУБД MySQL.

Реализованная система может применяться для хранения данных о автомобилях, доставках и компонентах. Реализованная система состоит из семи сущностей, представлений, функций, процедур и триггеров, что позволяет автоматизировать процесс работы с данной ИС. В рамках выполнения курсового проекта были освоены навыки работы с таблицами в MySQL и обработки запросов при помощи процедур и функций.

In this course project, a relational database was developed for the information system "Production". The project consists of 4 stages, including a description of the subject area, a description of the functionality, developed code and annex containing data schemas. To implement the IS, the MySQL DBMS was used.

The implemented system can be used to store data about cars, deliveries and components. The implemented system consists of seven entities, views, functions, procedures and triggers, which allows you to automate the process of working with this IS. As part of the course project, the skills of working with tables in MySQL and processing queries using procedures and functions were mastered.

СОДЕРЖАНИЕ

[1 ВВЕДЕНИЕ 4](#_Toc103159687)

[2 ПРОЕКТИРОВАНИЕ БД 5](#_Toc103159688)

[2.1. Цель и задачи, решаемые подсистемой хранения данных 5](#_Toc103159689)

[2.2. Описание предметной области 5](#_Toc103159690)

[2.3. Пользователи разрабатываемой подсистемы 5](#_Toc103159691)

[2.4. Начальная оценка и выделение сущностей 6](#_Toc103159692)

[2.5. Словарь предметной области 6](#_Toc103159693)

[2.6. Нормализация 6](#_Toc103159694)

[2.7. Описание связей между сущностями БД 7](#_Toc103159695)

[3 ФУНКЦИОНАЛЬНАЯ ДЕКОМПОЗИЦИЯ СИСТЕМЫ 10](#_Toc103159696)

[3.1. Функция DeliveryTime 10](#_Toc103159697)

[3.2. Функция ReleaseTime 10](#_Toc103159698)

[3.3. Функция FixDelivery 10](#_Toc103159699)

[3.4. Процедура CreateAuto 10](#_Toc103159700)

[3.5. Процедура SwitchDeliveryStatus 10](#_Toc103159701)

[3.6. Процедура AutoArchiving 10](#_Toc103159702)

[3.7. Представление AutoWithComponent 11](#_Toc103159703)

[3.8. Триггер OnDelivered 11](#_Toc103159704)

[3.9. Триггер PlusModelCount 11](#_Toc103159705)

[4 РЕАЛИЗАЦИЯ КОМПОНЕНТОВ БАЗЫ ДАННЫХ 12](#_Toc103159706)

[4.1. Физическая структура базы данных 12](#_Toc103159707)

[4.2. Спецификация функций 14](#_Toc103159708)

[4.3. Спецификация процедур 15](#_Toc103159709)

[4.4. Спецификация триггеров 17](#_Toc103159710)

[4.5. Спецификация представления 17](#_Toc103159711)

[5 ТЕСТИРОВАНИЕ СИСТЕМЫ 18](#_Toc103159712)

[5.1. Функция DeliveryTime() 18](#_Toc103159713)

[5.2. Функция ReleaseTime() 18](#_Toc103159714)

[5.3. Функция FixDelivery() 19](#_Toc103159715)

[5.4. Процедура CreateAuto() 20](#_Toc103159716)

[5.5. Процедура SwitchDeliveryStatus() 21](#_Toc103159717)

[5.6. Процедура AutoArchiving(2019) 22](#_Toc103159718)

[5.7. Триггер OnDelivered 23](#_Toc103159719)

[5.8. Триггер PlusModelCount 24](#_Toc103159720)

[5.9. Представление AutoWithComponent 25](#_Toc103159721)

[6 ЗАКЛЮЧЕНИЕ 26](#_Toc103159722)

[7 СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 27](#_Toc103159723)

[ПРИЛОЖЕНИЕ А 28](#_Toc103159724)

[ПРИЛОЖЕНИЕ Б 30](#_Toc103159725)

# ВВЕДЕНИЕ

В ходе выполнения курсового проекта требовалось разработать подсистему хранения данных «Производство» для автоматизации работы со сведениями о производстве грузовых автомобилей.

Цель работы: разработать подсистему хранения данных, которая будет позволять сохранять фактические данные о производстве грузовых автомобилей, их поставки заказчикам и наличия требуемых деталей у поставщиков.

Для достижения поставленной цели необходимо решить следующие задачи:

* хранение списка автомобилей;
* хранение списка производителей компонентов;
* отметка о доставке техники;

# ПРОЕКТИРОВАНИЕ БД

## Цель и задачи, решаемые подсистемой хранения данных

Цель работы: разработать подсистему хранения данных, которая будет позволять сохранять фактические данные о производстве грузовых автомобилей, их поставки заказчикам и наличия требуемых деталей у поставщиков.

Для достижения поставленной цели необходимо решить следующие задачи:

* хранение списка автомобилей;
* хранение списка производителей компонентов;
* отметка о доставке техники;

## Описание предметной области

На заводе, производящем грузовые автомобили, важно отслеживать выполнение заказа, его доставку и сборку. В БД хранится информация о различных моделях техники и их составных компонентов, так же хранится информация о доставках и о каждом произведенном грузовике отдельно

## Пользователи разрабатываемой подсистемы

Менеджер по заказам – сотрудник завода, может просматривать доставки, добавлять их, изменять статус доставки, редактировать таблицу с информацией об автомобилях.

Доставщик – сотрудник завода, может просматривать заказы, изменять статус заказа, делать выборки по адресам доставки, по компаниям заказчикам

Инженер – сотрудник завода, может просматривать таблицу моделей и создавать новые модели

Рабочий – сотрудник завода, может просматривать таблицы с моделями и заказами, делать выборки по моделям

Заказчик – компания, заказывающая технику, может просматривать состояние заказа

## Начальная оценка и выделение сущностей

Для предметной области «Производство» были выделены следующие сущности: автомобиль, компания, модель, корпус, двигатель, доставка.

## Словарь предметной области

*Автомобиль* – единица производимого товара.

Атрибуты: номер, номер модели, номер заказа, год выпуска

*Модель* – вид единицы техники.

Атрибуты: номер модели, название, номер корпуса, номер мотора, объем выпуска

*Доставка* – информация о доставке.

Атрибуты: номер доставки, заказчик, адрес доставки, телефон, статус доставки

*Состояние доставки* – информация о статусе доставки, может принимать 3 состояния: Изготавливается, В пути, Доставлен.

*Корпус* – информация о типе корпуса.

Атрибуты: номер корпуса, название, размер, компания изготовитель

*Двигатель* – информация о типе двигателе.

Атрибуты: номер мотора, название, мощность, компания изготовитель.

## Нормализация

Нормализация — это процесс организации данных в базе данных, включающий создание таблиц и установление отношений между ними в соответствии с правилами, которые обеспечивают защиту данных и делают базу данных более гибкой, устраняя избыточность и несогласованные зависимости.

**Первая нормальная форма**

* Отсутствуют повторяющиеся группы в отдельных таблицах
* Отдельная таблица для каждого набора связанных данных
* Каждый набор данных идентифицируется с помощью первичного ключа

**Вторая нормальная форма**

* Наличие отдельных таблиц для наборов значений, относящимся к нескольким записям
* Таблицы связаны с помощью внешнего ключа

**Третья нормальная форма**

* Устранены поля, не зависящие от ключа

**Результат применения аппарата нормализации**

В результате применения аппарата нормализации была достигнута 3 нормальная форма базы данных: в таблицах отсутствуют повторяющиеся строки; каждая таблица идентифицируется с помощью первичного ключа; таблица связаны с помощью внешнего ключа; отсутствуют поля, не зависящие от ключа.

## Описание связей между сущностями БД

**Логическая и физическая схема базы данных представлены на Рис. 1 и Рис. 2 соответственно**

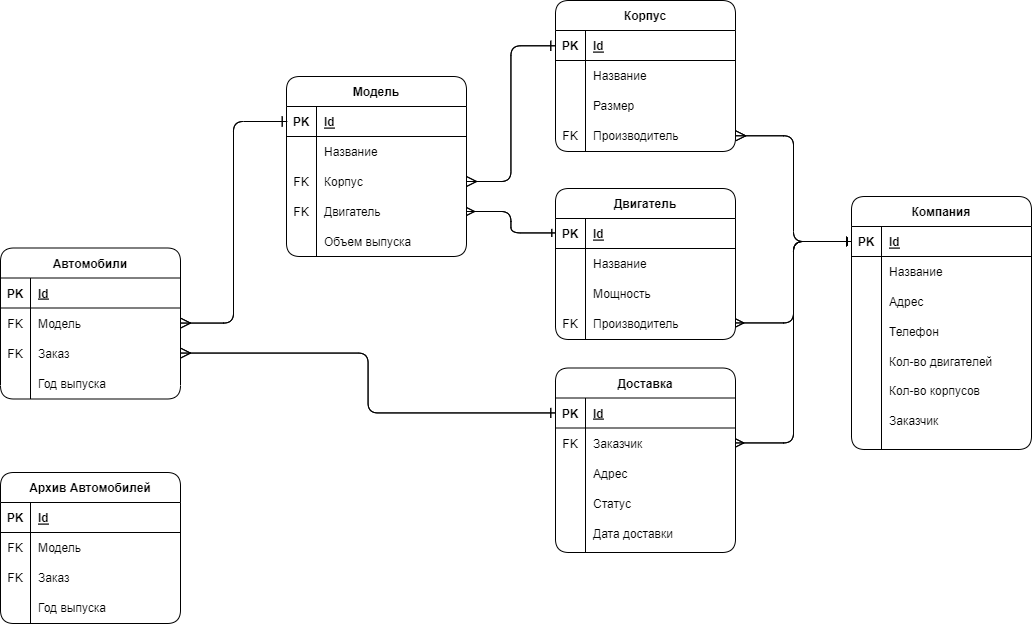
****

Рисунок 1.Логическая схема базы данных

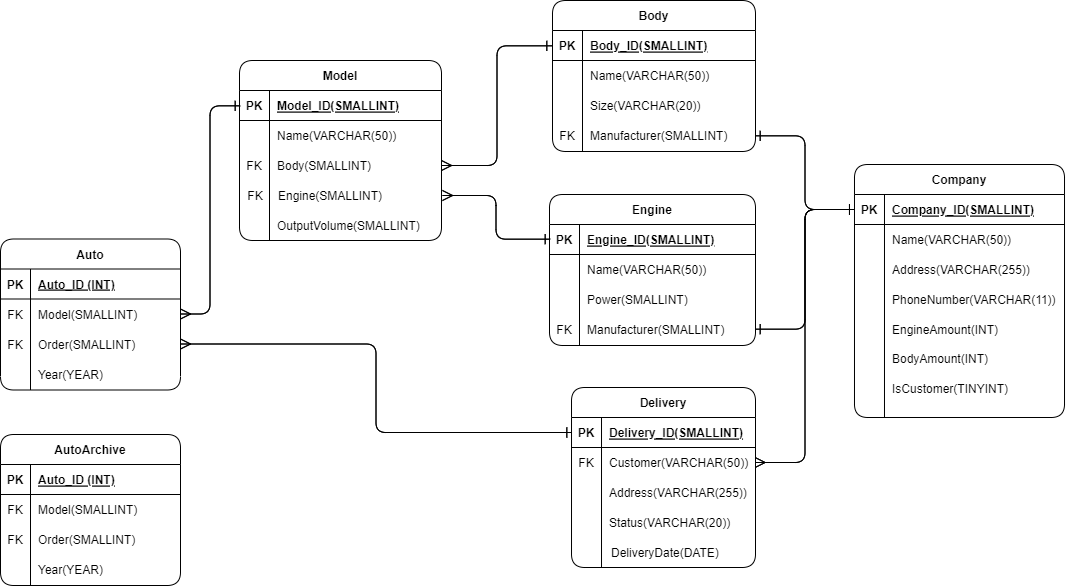


Рисунок 2. Физическая схема базы данных

Таблица 1. Связи между сущностями

|  |  |  |  |
| --- | --- | --- | --- |
| Связь | Обязательность | Кратность | Идентифицируемость |
| Таблица «Автомобили» и таблица «Модель» | Связь обязательна, так как у автомобиля есть модель | Один-ко-многим  Автомобиль имеет одну модель  Много автомобилей могут быть одной модели | Связь не идентифицируемая, так как первичный ключ в таблице «Автомобили» не является частью первичного ключа в таблице «Модель» |
| Таблица «Автомобили» и таблица «Доставка» | Связь обязательна, так как автомобиль производится для заказа | Один-ко-многим  Автомобиль может быть в одном заказе  В заказе может быть несколько автомобилей | Связь не идентифицируемая, так как первичный ключ в таблице «Автомобили» не является частью первичного ключа в таблице «Заказ» |
| Таблица «Модель» и таблица «Корпус» | Связь обязательна, так как у модель присутствует корпус | Один-ко-многим  У модели может быть один корпус  С одним корпусом может быть несколько моделей | Связь не идентифицируемая, так как первичный ключ в таблице «Модель» не является частью первичного ключа в таблице «Корпус» |
| Таблица «Модель» и таблица «Двигатель» | Связь обязательна, так как у модели присутствует двигатель | Один-ко-многим  У модели может быть один двигатель  С одним двигателем могут быть разные модели | Связь не идентифицируемая, так как первичный ключ в таблице «Модель» не является частью первичного ключа в таблице «Двигатель» |
| Таблица «Доставка» и таблица «Компания» | Связь обязательна, так как доставку заказывает определенная компания | Один-ко-многим  Компания может сделать много заказов  Один заказ может ассоциироваться с одной кампанией | Связь не идентифицируемая, так как первичный ключ в таблице «Компания» не является частью первичного ключа в таблице «Доставка» |
| Таблица «Корпус» и таблица «Компания» | Связь обязательна, так как у корпуса есть компания-производитель | Один-ко-многим  Компания может производить несколько видов корпусов  Один корпус может быть произведен одной компанией | Связь не идентифицируемая, так как первичный ключ в таблице «Корпус» не является частью первичного ключа в таблице «Компания» |
| Таблица «Двигатель» и таблица «Компания» | Связь обязательна, так как у двигателя есть компания-производитель | Один-ко-многим  Компания может производить несколько видов двигателей  Один двигатель может быть произведен одной компанией | Связь не идентифицируемая, так как первичный ключ в таблице «Двигатель» не является частью первичного ключа в таблице «Компания» |

# ФУНКЦИОНАЛЬНАЯ ДЕКОМПОЗИЦИЯ СИСТЕМЫ

## Функция DeliveryTime

Принимает в аргументы *ID автомобиля* и возвращает срок его доставки, в случае если автомобиль еще не доставлен – возвращает -1

## Функция ReleaseTime

Принимает в аргументы *ID модели* и возвращает время её производства путем высчитывания разницы между последним годом выпуска данной модели и первым

## Функция FixDelivery

Не принимает аргументов, с помощью **курсора deliveryCursor** проходит по таблице Delivery и проставляет статус «Доставлено» у строк, где есть дата доставки, но неправильно стоит статус, возвращает строку с ID исправленных строк в виде «ID-ID-ID»

## Процедура CreateAuto

Принимает на вход *ID модели, Покупателя, Адрес и ID доставки,* добавляет запись в таблицу Auto, если *ID доставки* был указан «-1» или не указан создает новую доставку для этого автомобиля, оформленную на *Покупателя*, если *ID доставки* передается, то привязывает новый автомобиль к этой доставке

## Процедура SwitchDeliveryStatus

Принимает на вход *ID заказа* и меняет его статус:

* Если *Статус* был «Изготавливается», то меняет его на «В пути», а также уменьшает *Количество двигателей и корпусов*, которые были использованы в производстве автомобилей из этого заказа
* Если *Статус* был «В пути», то меняет его на «Доставлен»

## Процедура AutoArchiving

Принимает на вход *Год* и архивирует автомобили, произведенные до заданного года, удаляя их из таблицы Auto и перенося в таблицу AutoArchive

## Представление AutoWithComponent

Предоставляет таблицу с информацией о автомобиле: *ID автомобиля,* *Названием модели, Название двигателя, Мощность двигателя, Название корпуса, Размеры корпуса* из таблиц Auto, Model**,** EngineиBody

## Триггер OnDelivered

При изменении *статуса* заказа в таблице Orderна «Доставлен», у этого заказа проставляется дата доставки, равная сегодняшнему дню

## Триггер PlusModelCount

При добавлении записи в таблицу Autoувеличивается *Объем выпуска* модели в таблице Model

# РЕАЛИЗАЦИЯ КОМПОНЕНТОВ БАЗЫ ДАННЫХ

## Физическая структура базы данных

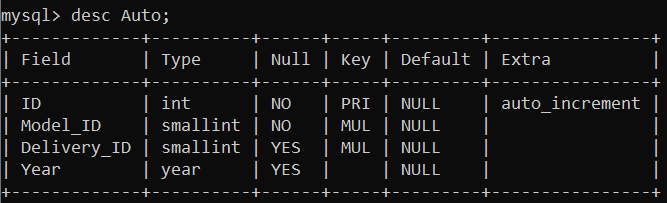


Рисунок 3. Структура таблицы Auto

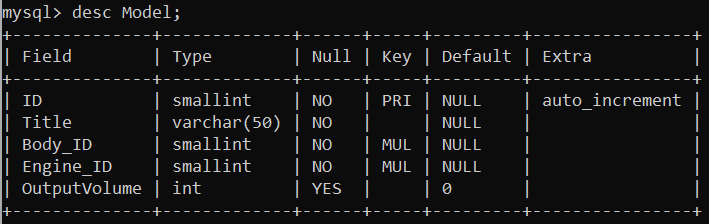


Рисунок 4. Структура таблицы Model

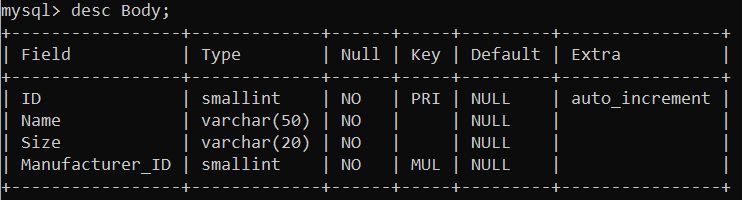


Рисунок 5. Структура таблицы Body

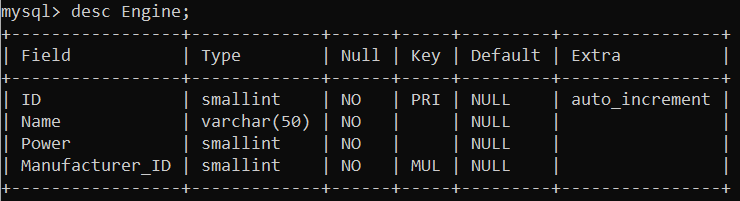


Рисунок 6. Структура таблицы Engine

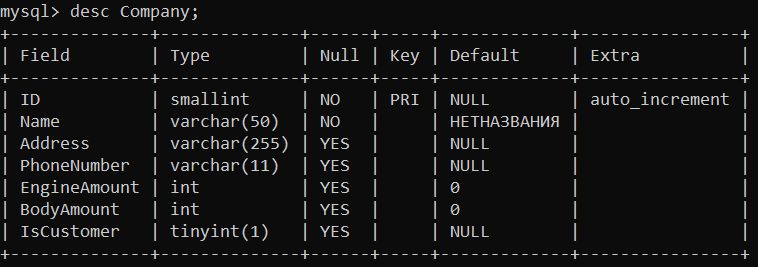


Рисунок 7. Структура таблицы Company

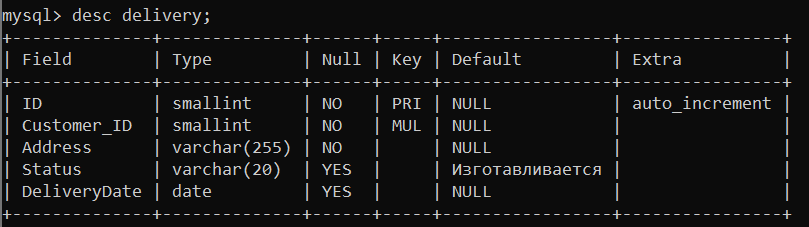


Рисунок 8. Структура таблицы Delivery

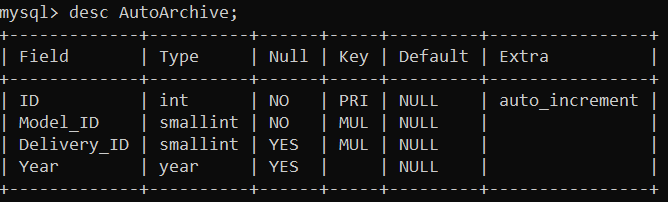


Рисунок 9. Структура таблицы AutoArchive

## Спецификация функций

* + 1. **Функция DeliveryTime()**

CREATE FUNCTION DeliveryTime(AutoID SMALLINT) RETURNS SMALLINT DETERMINISTIC

BEGIN

DECLARE ReleaseYear SMALLINT;

DECLARE DeliveryID SMALLINT;

DECLARE DeliveryYear SMALLINT;

SELECT Year,Delivery\_ID from

(SELECT ID,Delivery\_ID,Year FROM Auto

UNION

SELECT ID,Delivery\_ID,Year FROM AutoArchive)

AS AllAuto

WHERE ID = AutoID

INTO ReleaseYear,DeliveryID;

SELECT YEAR(DeliveryDate) from Delivery

WHERE ID = DeliveryID

INTO DeliveryYear;

IF DeliveryYear IS NULL THEN RETURN -1;

ELSE RETURN (DeliveryYear - ReleaseYear + 1);

END IF;

END$$

* + 1. **Функция ReleaseTime()**

CREATE FUNCTION ReleaseTime(ModelID SMALLINT) RETURNS SMALLINT DETERMINISTIC

BEGIN

DECLARE MinYear SMALLINT;

DECLARE MaxYear SMALLINT;

SELECT MIN(Year) FROM

(SELECT Model\_ID,Year FROM Auto

UNION

SELECT Model\_ID,Year FROM AutoArchive)

AS AllAuto

WHERE Model\_ID = ModelID

INTO MinYear;

SELECT MAX(Year) FROM

(SELECT Model\_ID,Year FROM Auto

UNION

SELECT Model\_ID,Year FROM AutoArchive)

AS AllAuto

WHERE Model\_ID = ModelID

INTO MaxYear;

RETURN (MaxYear - MinYear + 1);

END$$

* + 1. **Функция FixDelivery()**

CREATE FUNCTION FixDelivery() RETURNS VARCHAR(255) DETERMINISTIC

BEGIN

DECLARE done INT DEFAULT 0;

DECLARE DelID SMALLINT;

DECLARE DelStatus VARCHAR(20);

DECLARE DelDate DATE;

DECLARE ErrCounter VARCHAR(255);

DECLARE deliveryCursor CURSOR

FOR SELECT ID,Status,DeliveryDate

FROM Delivery;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;

OPEN deliveryCursor;

featchLoop: WHILE done = 0 DO

FETCH deliveryCursor INTO DelID,DelStatus,DelDate;

IF DelStatus != "Доставлен" AND DelDate IS NOT NULL THEN

UPDATE Delivery

SET Status = "Доставлен"

WHERE ID = DelID;

SET ErrCounter = CONCAT\_WS('-',ErrCounter,DelID);

END IF;

END WHILE featchLoop;

CLOSE deliveryCursor;

RETURN LEFT(ErrCounter,LENGTH(ErrCounter)-3);

END$$

## Спецификация процедур

* + 1. **Процедура CreateAuto()**

CREATE PROCEDURE CreateAuto(

IN ModelID SMALLINT,

IN Customer VARCHAR(50),

IN Address VARCHAR(255),

IN DeliveryID SMALLINT )

BEGIN

IF DeliveryID = -1 OR DeliveryID IS NULL THEN

SELECT ID FROM Company

WHERE Name = Customer

INTO @delivery\_company\_ID;

INSERT Delivery(Customer\_ID,Address)

VALUES (@delivery\_company\_ID, Address);

SELECT max(ID) FROM Delivery

INTO DeliveryID;

END IF;

INSERT Auto(Model\_ID,Delivery\_ID,Year)

VALUES (ModelID,DeliveryID,YEAR(NOW()));

END$$

* + 1. **Процедура SwitchDeliveryStatus()**

CREATE PROCEDURE SwitchDeliveryStatus(IN DeliveryID SMALLINT)

BEGIN

SELECT Status FROM Delivery

WHERE ID = DeliveryID

INTO @usedStatus;

IF @usedStatus = "Изготавливается" THEN

UPDATE Delivery

SET Status = "В пути"

WHERE Delivery.ID = DeliveryID;

UPDATE Company

JOIN Engine ON Company.ID = Engine.Manufacturer\_ID

JOIN Model ON Engine.ID = Model.Engine\_ID

JOIN Auto ON Model.ID = Auto.Model\_ID

SET EngineAmount = EngineAmount - 1

WHERE Auto.Delivery\_ID = DeliveryID;

UPDATE Company

JOIN Body ON Company.ID = Body.Manufacturer\_ID

JOIN Model ON Body.ID = Model.Body\_ID

JOIN Auto ON Model.ID = Auto.Model\_ID

SET BodyAmount = BodyAmount - 1

WHERE Auto.Delivery\_ID = DeliveryID;

ELSEIF @usedStatus = "В пути" THEN

UPDATE Delivery

SET Status = "Доставлен"

WHERE Delivery.ID = DeliveryID;

END IF;

END$$

* + 1. **Процедура AutoArchiving()**

CREATE PROCEDURE AutoArchiving(IN YearOfIssue SMALLINT)

BEGIN

INSERT INTO AutoArchive

SELECT \* from Auto

WHERE Auto.Year <= YearOfIssue;

DELETE FROM Auto

WHERE Auto.Year <= YearOfIssue;

END$$

## Спецификация триггеров

* + 1. **Триггер OnDelivered**

CREATE TRIGGER OnDelivered BEFORE UPDATE ON Delivery

FOR EACH ROW

IF NEW.Status="Доставлен" AND NEW.DeliveryDate IS NULL THEN

SET NEW.DeliveryDate = CURDATE();

END IF$$

* + 1. **Триггер PlusModelCount**

CREATE TRIGGER PlusModelCount BEFORE INSERT ON Auto

FOR EACH ROW

UPDATE Model

SET OutputVolume = OutputVolume + 1

WHERE Model.ID = NEW.Model\_ID;

## Спецификация представления

* + 1. **Представление** **AutoWithComponent**

CREATE VIEW AutoWithComponent AS

SELECT Auto.id AS AutoID,

Model.Title AS Model,

Engine.Name AS Engine,

Engine.Power AS Power,

Body.Name AS Body,

Body.Size AS Size

FROM Auto JOIN Model ON Auto.Model\_ID=Model.ID

JOIN Engine ON Model.Engine\_ID=Engine.ID

JOIN Body ON Model.Body\_ID=Body.ID;

# ТЕСТИРОВАНИЕ СИСТЕМЫ

## Функция DeliveryTime()

На Рис. 8 представлен скриншот теста данной функции

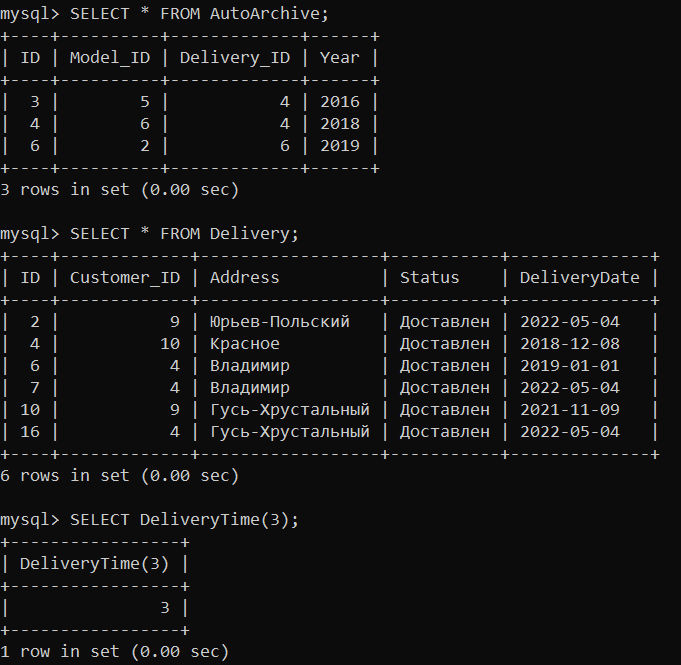


Рисунок 10

## Функция ReleaseTime()

На Рис. 9 представлен скриншот теста данной функции

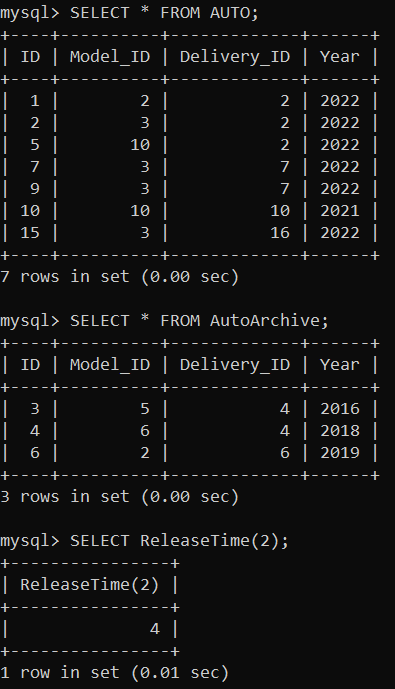


Рисунок 11

## Функция FixDelivery()

На Рис. 10 представлен скриншот теста данной функции

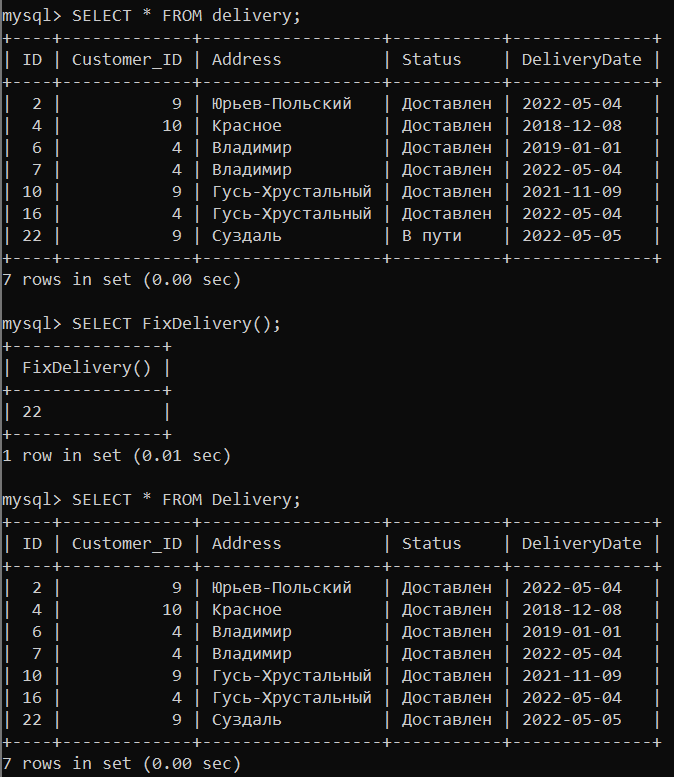


Рисунок 12

## Процедура CreateAuto()

На Рис. 11 представлен скриншот теста данной функции

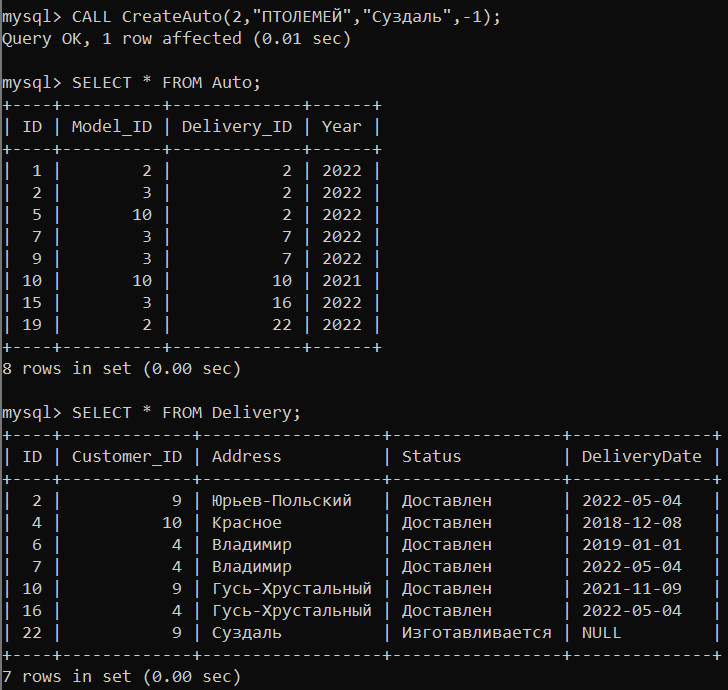


Рисунок 13

## Процедура SwitchDeliveryStatus()

На Рис. 12 представлен скриншот теста данной функции

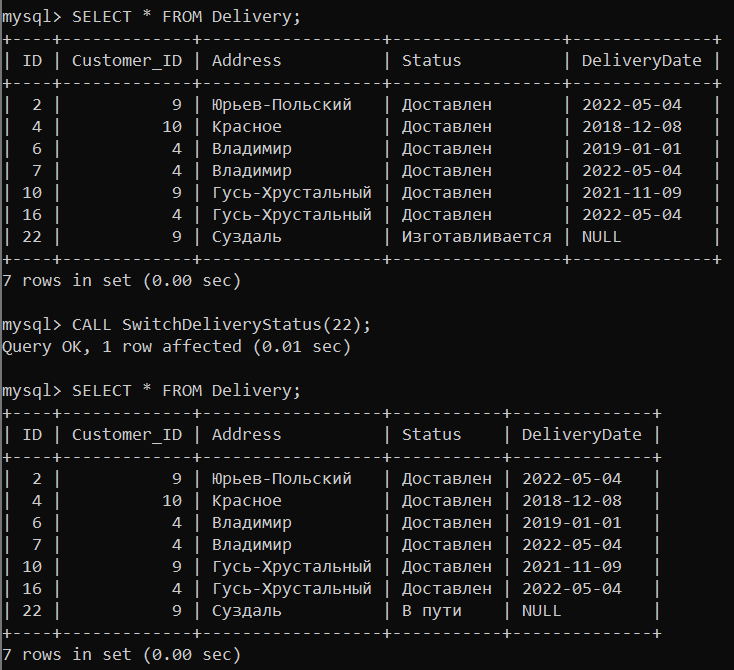


Рисунок 14

## Процедура AutoArchiving(2019)

На Рис. 13 и Рис. 14 представлен скриншот теста данной функции

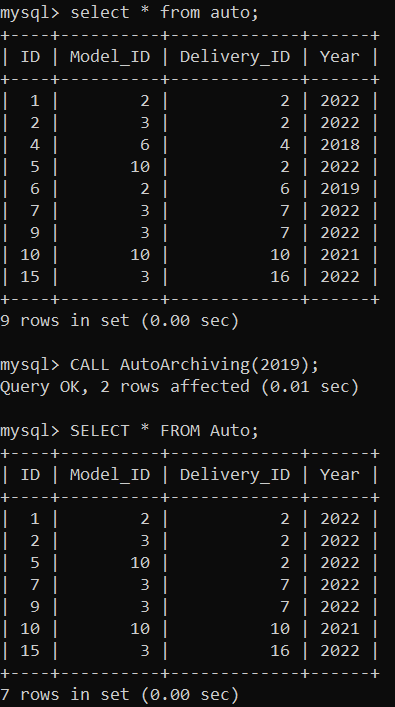


Рисунок 15

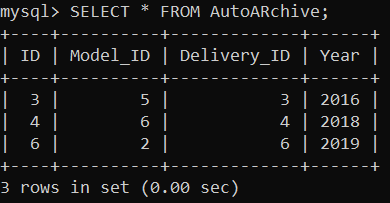


Рисунок 16

## Триггер OnDelivered

На Рис. 15 представлен скриншот теста данного триггера

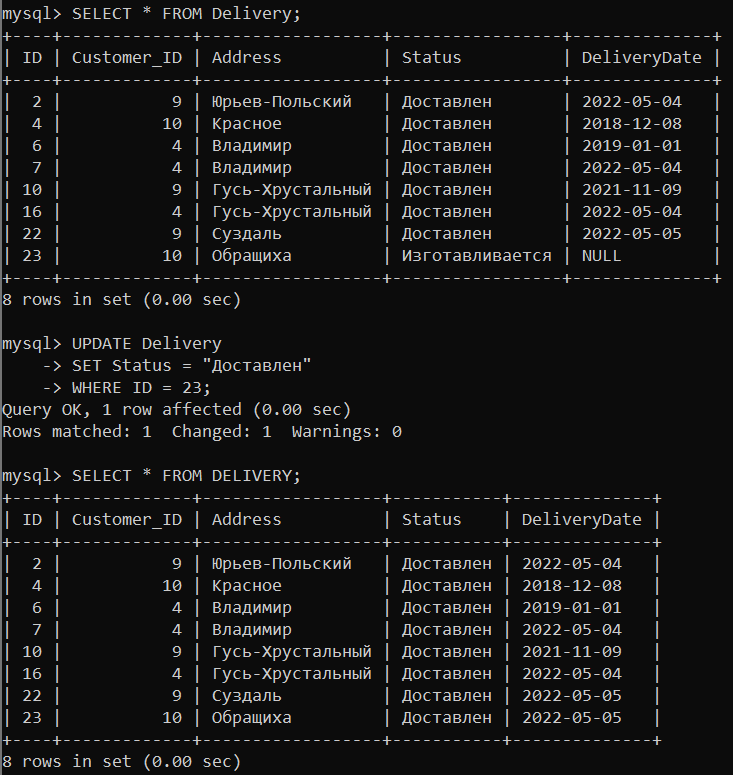


Рисунок 17

## Триггер PlusModelCount

На Рис. 16 представлен скриншот теста данного триггера

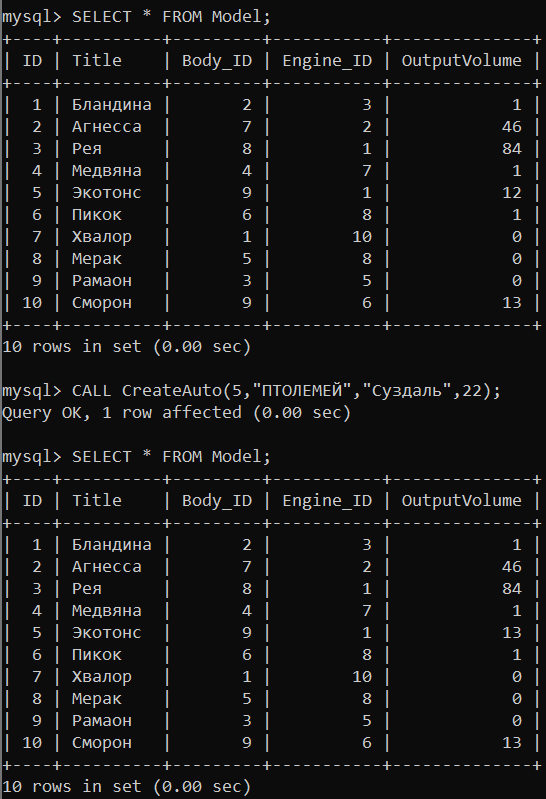


Рисунок 18

## Представление AutoWithComponent

На Рис. 17 представлен скриншот теста данного представления

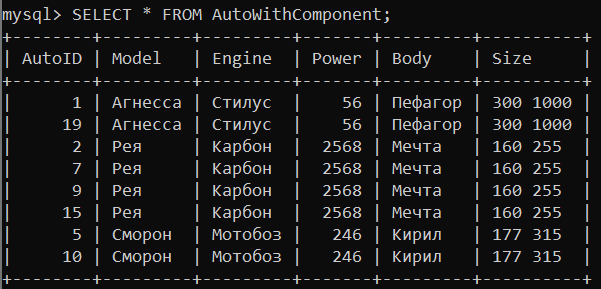


Рисунок 19

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта была разработана база данных для информационной системы «Производство», позволяющая автоматизировать работу с данными о автомобилях, доставках и компонентах. Было произведено исследование предметной области и получены навыки работы с СУБД MySQL.

# СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Кузнецов С.Д. Основы баз данных. – 2-е изд. – М.: Интернет-университет информационных технологий; БИНОМ. Лаборатория знаний, 2007. – 484 с. – ISBN 978-5-94774-736
2. Дейт К. Дж. Введение в системы баз данных – 8-е изд. – М.: Вильямс, 2005. – 1328 с – ISBN 5-8459-0788-8
3. Кириллов В.В. Основы проектирования реляционных баз данных. Учебное пособие. – СПб.: ИТМО, 1994. – 90 с.
4. Максим Кузнецов, Игорь Симдянов. Самоучитель MySQL: -- СПб.: БХВ-Петербург, 2006. – 546 с. – ISBN 978-5-94157-754-5, 5-94157-754-0
5. MySQL 8.0 Reference Manual Copyright 1997-2022 MySQL AB, 2022 Sun Microsystems, Inc.

## ПРИЛОЖЕНИЕ А

**СКРИПТ СОЗДАНИЯ БАЗЫ ДАННЫХ**

CREATE DATABASE Manufacture;

USE Manufacture;

CREATE TABLE `auto` (

`ID` int NOT NULL AUTO\_INCREMENT,

`Model\_ID` smallint NOT NULL,

`Delivery\_ID` smallint NOT NULL,

`Year` year DEFAULT NULL,

PRIMARY KEY (`ID`),

FOREIGN KEY (`Model\_ID`) REFERENCES `model` (`ID`),

FOREIGN KEY (`Delivery\_ID`) REFERENCES `delivery` (`ID`)

);

CREATE TABLE `autoarchive` (

`ID` int NOT NULL AUTO\_INCREMENT,

`Model\_ID` smallint NOT NULL,

`Delivery\_ID` smallint DEFAULT NULL,

`Year` year DEFAULT NULL,

PRIMARY KEY (`ID`)

);

CREATE TABLE `body` (

`ID` smallint NOT NULL AUTO\_INCREMENT,

`Name` varchar(50) NOT NULL,

`Size` varchar(20) NOT NULL,

`Manufacturer\_ID` smallint NOT NULL,

PRIMARY KEY (`ID`),

FOREIGN KEY (`Manufacturer\_ID`) REFERENCES `company` (`ID`)

);

CREATE TABLE `company` (

`ID` smallint NOT NULL AUTO\_INCREMENT,

`Name` varchar(50) NOT NULL DEFAULT 'НЕТНАЗВАНИЯ',

`Address` varchar(255) DEFAULT NULL,

`PhoneNumber` varchar(11) DEFAULT NULL,

`EngineAmount` int DEFAULT '0',

`BodyAmount` int DEFAULT '0',

`IsCustomer` tinyint(1) DEFAULT NULL,

PRIMARY KEY (`ID`)

);

CREATE TABLE `delivery` (

`ID` smallint NOT NULL AUTO\_INCREMENT,

`Customer\_ID` smallint NOT NULL,

`Address` varchar(255) NOT NULL,

`Status` varchar(20) DEFAULT 'Изготавливается',

`DeliveryDate` date DEFAULT NULL,

PRIMARY KEY (`ID`),

FOREIGN KEY (`Customer\_ID`) REFERENCES `company` (`ID`)

);

REATE TABLE `engine` (

`ID` smallint NOT NULL AUTO\_INCREMENT,

`Name` varchar(50) NOT NULL,

`Power` smallint NOT NULL,

`Manufacturer\_ID` smallint NOT NULL,

PRIMARY KEY (`ID`),

FOREIGN KEY (`Manufacturer\_ID`) REFERENCES `company` (`ID`)

);

CREATE TABLE `model` (

`ID` smallint NOT NULL AUTO\_INCREMENT,

`Title` varchar(50) NOT NULL,

`Body\_ID` smallint NOT NULL,

`Engine\_ID` smallint NOT NULL,

`OutputVolume` int DEFAULT '0',

PRIMARY KEY (`ID`),

FOREIGN KEY (`Body\_ID`) REFERENCES `body` (`ID`),

FOREIGN KEY (`Engine\_ID`) REFERENCES `engine` (`ID`)

);

## ПРИЛОЖЕНИЕ Б

**ЗАПОЛНЕНИЕ БАЗЫ ДАННЫХ**

INSERT Company (Name, Address, PhoneNumber, EngineAmount, BodyAmount, IsCustomer)

VALUES

(‘ТОМФАН’,’Киржач’,NULL,13,2,0),

(‘ВИГНАТ’,’Владимир’,’89005814406’,3,3,0),

(‘АРОМАС’,’Гусь-Хрустальный’,’89611421993’,0,0,1),

(‘ПРИОН’,’Уфа’,’89367197755’,1,7,0),

(‘ОРИФИЯ’,’Москва’,NULL,10,9,0),

(‘ВАСАТ’,’Рязань’,NULL,4,21,0),

(‘МИШНО’,’Сочи’,NULL,1,1,0),

(‘ПИЛОМЕЙ’,’Красноярск’,’89578738788’,0,0,1),

(‘ЕЛИЗАР’,’Киржач’,’\*9959872873’,0,0,1);

INSERT Engine(Name,Power,Manufacturer\_ID)

VALUES

(‘Карбон’,2568,5),

(‘Стилус’,56,1),

(‘Колбаса’,732,3),

(‘Кипарус’,54,7),

(‘ДжикДжик’,234,7),

(‘Мотобоз’,246,6),

(‘Каболдибуг’,243,1),

(‘Карина’,9999,7),

(‘Борис’,1,2),

(‘Странник’,90,3);

INSERT Body(Name,Size,Manufacturer\_ID)

VALUES

(‘Арбуз’,’180 500’,7),

(‘Арбузятина’,’250 800’,7),

(‘Абьюзер’,’150 300’,2),

(‘Антон’,’174 460’,1),

(‘ЛучШий’,’165 680’,5),

(‘Брегантина’,’186 400’,6),

(‘Пефагор’,’300 1000’,7),

(‘Мечта’,’160 255’,3),

(‘Кирил’,’177 315’,7),

(‘Аркаша’,’188 456’,1);

INSERT Model(Name,Body\_ID,Engine\_ID,OutputVolume)

VALUES

(‘Бландина’,2,3,0),

(‘Агнесса’,7,2,45),

(‘Рея’,8,1,83),

(‘Медвяна’,4,7,0),

(‘Экотос’,9,1,12),

(‘Пикок’,10,8,1),

(‘Хвалор’,1,10,0),

(‘Мерак’,5,8,0),

(‘Рамаон’,3,5,0),

(‘Сморон’,9,4,13);

INSERT Delivery(Customer\_ID,Address,Status,DeliveryDate)

VALUES

(4,’Приволжск’,’Изготавливается’,NULL),

(9,’Юрьев-Польский’,’В пути’,NULL),

(10,’Киржач’,’Доставлен’,’2016-04-18’),

(10,’Красное’,’Доставлен’,’2018-12-08’),

(10,’Рязань’,’Изготавливается’,NULL),

(4,’Владимир’,’Доставлен’,’2019-01-01’),

(4,’Владимир’,’В пути’,NULL),

(9,’Уфа’,’Доставлен’,’2019-04-29’),

(9,’Красноярск’,’Изготавливается’,NULL),

(9,’Гусь-Хрустальный’,’Доставлен’,’2021-11-09’);

INSERT Auto(Model\_ID,Delivery\_ID,Year)

VALUES

(2,2,’2022’),

(3,2,’2022’),

(5,3,’2016’),

(6,4,’2018’),

(10,2,’2022’),

(2,6,’2019’),

(3,7,’2022’),

(3,8,’2019’),

(3,7,’2022’),

(10,10,’2021’);