

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра компьютерных систем в управлении и проектировании
(КСУП)

СОЗДАНИЕ АВТОМАТИЗИРОВАННОЙ ИНФОРМАЦИОННОЙ
СИСТЕМЫ С УЧЕБНО-ИССЛЕДОВАТЕЛЬСКОЙ БАЗОЙ ДАННЫХ:
СИСТЕМА УПРАВЛЕНИЯ СЕРВИСОМ ТАКСИ

Курсовая работа по дисциплине «Базы данных»

Студент гр. 580-1

_____ А.П. Белошицкий

“ _____ ” _____ 2022 г

Руководитель

доцент каф. КСУП

_____ К.С. Сарин

оценка

“ _____ ” _____ 2022 г

Томск 2022

РЕФЕРАТ

Курсовая работа, 70 с., 66 рис., 1 табл., 0 источников, 6 прилож.

**БАЗЫ ДАННЫХ, РЕЛЯЦИОННЫЕ МОДЕЛИ, НОРМАЛИЗАЦИЯ
РЕЛЯЦИОННЫХ МОДЕЛЕЙ, АВТОМАТИЗИРОВАННЫЕ СИСТЕМЫ.**

Цель нашей работы - создание программы, позволяющей автоматизировать работу с отсортированными данными сервиса такси.

Для этого созданы: модель бизнес процессов в методологии IDEF0, на её основе концептуальная модель. На основе концептуальной модели создана реляционная модель в методологии IDEFX. На основе модели в методологии IDEFX созданы таблицы базы данных. Работа с созданной базой данных осуществляется через интерфейс программы, разработанной на языке C# и с использованием компонентов Windows Forms. Также в программе была реализована система запросов, с использованием языка запросов SQL.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра компьютерных систем в управлении и проектировании
(КСУП)

УТВЕРЖДАЮ

Заведующий кафедрой КСУП,

д-р техн. наук, профессор

_____ Ю.А. Шурыгин

“ ____ ” _____ 2022 г

Задание

На курсовую работу по дисциплине «Базы данных» студенту Белошицкому Артёму Павловичу группы 580-1 факультета вычислительных систем.

1 Проектирование инфологической модели данных: Тема работы: Создание автоматизированной информационной системы с учебно-исследовательской базой данных: _Система_управления_сервисом_такси

2 Исходные данные к работе:

2.1 Реляционная СУБД;

2.2 Данные по предметной области;

2.3 Название предметной области;

3 Срок сдачи студентом законченной работы: до 25 декабря 2022 г.

4 Содержание курсовой работы:

4.1 Проектирование инфологической модели данных:

- описание структуризации предметной области (описание бизнес-процессов, диаграммы IDEF0);
- представление модели «Сущность-связь» (ER-модель);
- сценарий пользовательского интерфейса.

4.2 Проектирование даталогической (логической) модели данных:

- проектирование реляционной базы данных на основе принципов нормализации;
- проектирование концептуальной модели данных (использование методологии IDEF1X);
- составление глоссария модели.

4.3 Физическое проектирование БД:

- создание базы данных и необходимых элементов;
- описание ограничений на базу данных;
- сопоставление логических и физических имен.

4.4 Написание программы обработки и работы с данными:

- генерация программы меню, реализующей пользовательский интерфейс;
- режим просмотра данных с использованием экранных форм; использование режимов редактирования данных;
- процедуры поиска и манипулирования данными (сортировки, фильтры и пр.);
- использование SQL операторов (SQL запросы, операторы определения данных, операторы манипулирования данными);
- возможность получить отчетную форму на принтер и экран;
- обеспечение безопасности данных.

5 Содержание пояснительной записки:

- титульный лист;
- реферат на русском языке;

- задание;
- содержание;
- введение;
- вопросы проектирования БД;
- обоснование выбора программных средств;
- руководство пользователю;
- описание прикладной программы;
- заключение; список использованных источников;
- приложения (структуры БД, экранные формы, отчетная форма, листинг программы).

Пояснительная записка должна быть оформлена в соответствии со стандартом ТУСУР.

В конверте на обложке приложить диск с БД, исходными текстами программы с соответствующими файлами, исполнительными файлами, пояснительной запиской.

6 Дата выдачи задания:

« 12 » сентября 2022 г.

Задание согласовано:

Руководитель работы К.С. Сарин

« 12 » сентября 2022 г.

Задание принято к исполнению

« 12 » сентября 2022 г.

СОДЕРЖАНИЕ

ЗАДАНИЕ	3
СОДЕРЖАНИЕ	6
ВВЕДЕНИЕ.....	9
1 КОНЦЕПТУАЛЬНОЕ (ИНФОЛОГИЧЕСКОЕ) ПРОЕКТИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ (ПО).....	10
1.1 Неформальное описание предметной области.....	10
1.2 Описание бизнес-процессов ПО в методологии функционального моделирования IDEF0.....	11
1.3 Цели автоматизации ПО.....	12
1.4 Бизнес-процессы, подлежащие автоматизации.....	12
1.5 Описание бизнес-процессов ПО в IDEF0 после внедрения автоматизированной информационной системы.....	13
1.6 Основные объекты ПО, информация о которых будет накапливаться в БД. Их характеристики и свойства (атрибуты)....	14
1.7 Определение связей между объектами ПО.....	14
1.8 Графическое представление концептуальной информационной модели данных.....	15
2 ПРОЕКТИРОВАНИЕ РЕЛЯЦИОННОЙ МОДЕЛИ БАЗЫ ДАННЫХ...	17
2.1 Определение отношений и связей между отношениями на основе концептуальной информационной модели. Первичные и внешние ключи	17
2.2 Нормализация логической модели данных.....	19
2.3 Графическое представление логической модели данных в методологии IDEF1х.....	20
2.4 Физическое проектирование с учетом выбранной СУБД. Определение типов данных атрибутов отношений.....	20
2.5 Графическое представление физической модели данных в методологии IDEF1х.....	28

2.6 Представление физической модели данных на языке SQL.....	30
2.7 Представление политики безопасности пользователей на языке SQL.....	33
3 ПРОГРАММНЫЙ КОМПЛЕКС ДЛЯ РАБОТЫ С СУБД.....	35
3.1 Создание приложения для работы с данными сервиса такси...	35
3.2 Подготовка главной формы.....	36
3.3 Подключение к базе данных.....	37
3.4 Отображение данных в приложении.....	38
3.5 Реляционная модель данных.....	42
3.6 Обработка исключений и ошибок.....	43
3.7 Поиск фильтрация данных.....	44
3.8 Реализация выбора пользователя в заказе.....	46
3.9 Создание вычисляемых колонок.....	47
3.10 Создание подстановочный полей.....	48
3.11 Создании формы для запросов.....	49
3.12 Полная запись оператора SELECT.....	51
3.13 Подзапросы.....	53
3.14 Запрос на добавление данных.....	54
3.15 Запрос на изменение данных.....	54
3.16 Запрос на удаление данных.....	55
ЗАКЛЮЧЕНИЕ.....	56
ПРИЛОЖЕНИЕ А.....	57
ПРИЛОЖЕНИЕ Б.....	59
ПРИЛОЖЕНИЕ В.....	61
ПРИЛОЖЕНИЕ Г.....	65
ПРИЛОЖЕНИЕ Д.....	67
ПРИЛОЖЕНИЕ Е.....	69

ВВЕДЕНИЕ

Базы данных – это организованный набор данных, который хранится в доступном электронном виде. Большие базы данных размещаются в облачных хранилищах и компьютерных кластерах, а небольшие могут храниться в файловой системе. Разработка баз данных включает в себя практические соображения, моделирование данных, формальные методы, языки запросов, безопасность, включающую конфиденциальность данных, а также проблемы распределения вычислений. С базами данных тесно связаны системы управления базами данных – программное обеспечение, взаимодействующее с конечными пользователями, различными приложениями и самой базой данных для анализа и сбора данных, а также включает средства администрирования баз данных.

В нашей работе применялась СУБД Microsoft SQL Server. Данная СУБД обеспечивает хранение данных в виде набора таблиц с типизированными столбцами. Поддерживаются различные типы данных, такие как integer, char, float decimal, varchar, binary, text и другие. Также поддерживаются определяемые пользователем составные типы данных. Кроме самих таблиц, имеется возможность создавать и хранить представления, процедуры, индексы и ограничения.

СУБД позволяет создавать резервные копии. В том случае, если необходимо откатить базу данных к какому-либо состоянию, можно осуществить это, сделав восстановление к заранее сделанной резервной копии базы данных.

Внешнее взаимодействие с базой данных будет осуществляться через интерфейс прикладной программы. В нашем случае это программа, созданная в Microsoft Visual Studio с использованием Windows Forms на языке C#. Программа позволяет взаимодействовать с данными в базе данных и создавать запросы.

Целью нашей работы - создание программы, позволяющей автоматизировать работу с данными сервиса такси. Для этого созданы: модель бизнес процессов в методологии IDEF0, на её основе концептуальная модель. На основе концептуальной модели создана реляционная модель в методологии IDEFX. На основе модели в методологии IDEFX созданы таблицы базы данных. Работа с созданной базой данных осуществляется через интерфейс разработанной на языке C# программы.

1 КОНЦЕПТУАЛЬНОЕ (ИНФОЛОГИЧЕСКОЕ) ПРОЕКТИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ (ПО)

1.1 Неформальное описание предметной области

Сервис такси, позволяющий оперативно перевозить клиентов из одной точки населённого пункта в другую, извлекая из этого прибыль. Имеет штат сотрудников, каждому из которых, отведена конкретная роль в работе сервиса. Так же имеет свой автопарк, но, так же, водитель может использовать в своей работе личное транспортное средство, если оно прошло обязательный техосмотр, а также техосмотр, проводящийся специальной бригадой механиков, роль которых - поддержание транспортных средств (собственности компании) в надлежащем - исправном состоянии, также эта роль частично ложится и на водителя.

1.2 Описание бизнес-процессов ПО в методологии функционального моделирования IDEF0

Пусть требуется создать программную систему, предназначенную для сервиса перевозки. Такая система должна обеспечивать хранение сведений о пользователях, которые зарегистрированы в сервисе, заказах, сотрудниках(водителях), всех машинах и маршрутах, для автоматического заполнения полей заказа. Пользователь делает заказ, указав своё местоположение (адрес по которому он хочет вызвать такси), адрес куда он хочет, чтобы его доставили и выбирает класс машины. Сервис доставки рассматривает заказ и регистрирует его. Далее ищется свободный водитель с соответствующим классом машины, если такого не нашлось, сервис предложит альтернативные варианты. Далее сервис считает расстояние и соответственно считает цену поездки. Далее, если пользователя всё устраивает, он выбирает способ оплаты и нажимает кнопку «ВЫЗВАТЬ

ТАКСИ» или «ПОЕХАЛИ!». Водитель получает соответствующее сообщение и выдвигается за заказчиком, оплачивает клиент в зависимости от выбранного способа оплаты.

Федеральный закон 69-ФЗ "О внесении изменений в отдельные законодательные акты Российской Федерации" вносит поправки к предыдущему Федеральному закону 34-ФЗ позволяющему регулировать деятельность юридических лиц, оказывающих услуг по перевозке.

Основные ограничения накладываемые на предметную область:

- Автопарк;
- Расценки;
- 69-ФЗ;

Основным объектом предметной области является: персонал.

На рисунке 1.1 представлена предметная область такси сервиса, IDEF0 нулевого уровня, до применения автоматизации процесса.



Рисунок 1.1 — IDEF0 нулевого уровня

На рисунке 1.1 представлена предметная область такси сервиса, IDEF0 нулевого уровня, до применения автоматизации процессов.

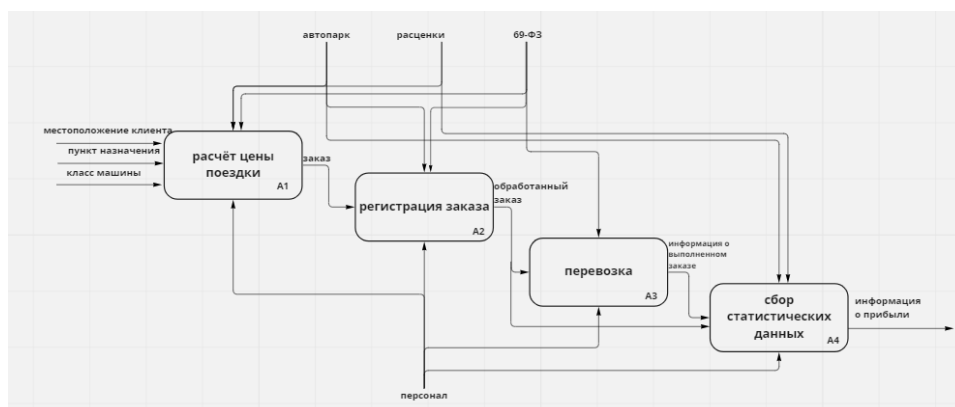


Рисунок 1.2 — IDEF0 первого уровня

1.3 Цели автоматизации ПО

Процессу автоматизации (рисунки 1.3 и 1.4) подлежит регистрация заказа, сбор и обработка статистических данных, расчёт цены поездки относительно расстояния и класса машины. Если какой-либо процесс IDEF0 N уровня подлежит автоматизации, то на схеме IDEF0 N+1 уровня также должна присутствовать автоматизированная система (рисунок 1.5), иначе процесс не автоматизируется.

1.4 Бизнес-процессы, подлежащие автоматизации

Список процессов, подлежащие автоматизации:

- Расчёт цены поездки;
- Регистрация заказа;
- Перевозка;
- Сбор статистических данных;

1.5 Описание бизнес-процессов ПО в IDEF0 после внедрения автоматизированной информационной системы

На рисунке 1.3 представлена предметная область такси сервиса, IDEF0 нулевого уровня, после автоматизации процесса.



Рисунок 1.3 — IDEF0 нулевого уровня после автоматизации

На рисунке 1.4 представлена предметная область такси сервиса, IDEF0 первого уровня, после автоматизации процессов.

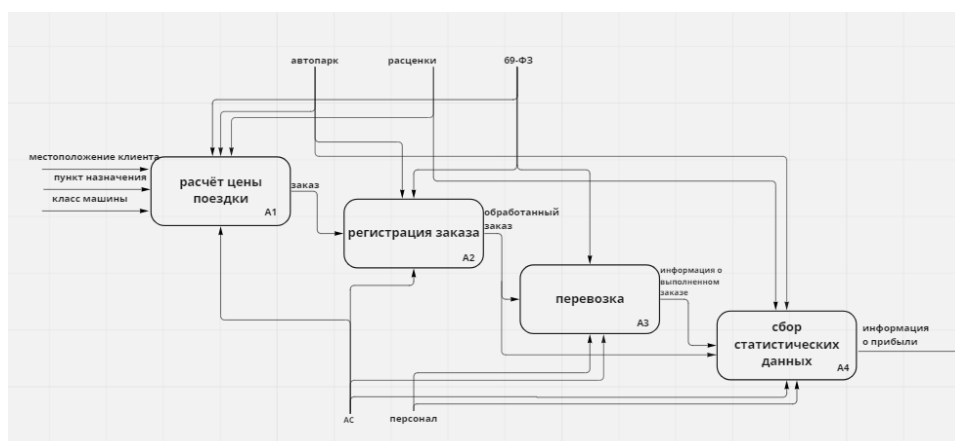


Рисунок 1.4 — IDEF0 первого уровня после автоматизации

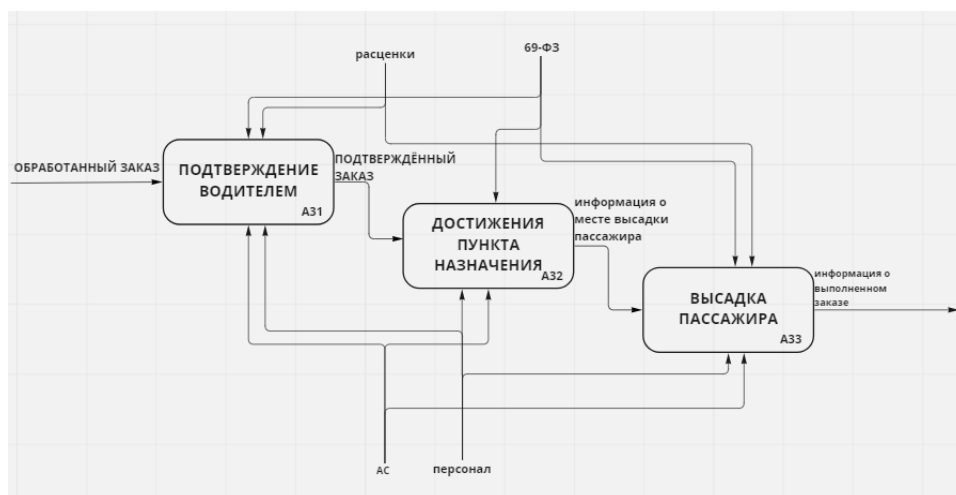


Рисунок 1.5 — IDEF0 второго уровня после автоматизации блока “перевозка”

1.6 Основные объекты ПО, информация о которых будет накапливаться в БД. Их характеристики и свойства (атрибуты)

На основе концептуальной информационной модели ПО, были выделены 4 сущности:

- 1) Клиент - ID-номер, Ф.И.О., Местоположение.
- 2) Заказ - Расстояние, статус, ID-номер, пункты отправления, пункт назначения, цена поездки.
- 3) Водитель - ID-номер, статус, Ф.И.О., Сотовый телефон.
- 4) Транспортное средство - статус, регистрационный номер, Класс автомобиля, ПТС, Страховой полис автомобиля.

1.7 Определение связей между объектами ПО

Также определены мощности связей между сущностями:

- 1) Клиент - Заказ(1:M): один клиент может оформить несколько заказов, но заказ прикрепляется только к одному клиенту, поэтому на концептуально-сущностной модели связь “1 ко многим”;

2) Заказ - Водитель(M:1): один водитель может выполнить несколько заказов, но каждый заказ выполняется одним конкретным водителем;

3) Водитель - Транспортное средство(1:M): за каждым водителем может быть закреплено несколько автомобилей, как служебных, так и собственных, но каждое транспортное средство закрепляется только за одним конкретным водителем.

Определенные сущности и связи между ними, можно наблюдать на рисунке 1.6



Рисунок 1.6 – Модель сущность-связь (ER-уровень)

1.8 Графическое представление концептуальной информационной модели данных

На основе данных об сущностях, их атрибутах и связях между ними была составлена концептуальная информационная модель. Она представлена на рисунке 1.7.

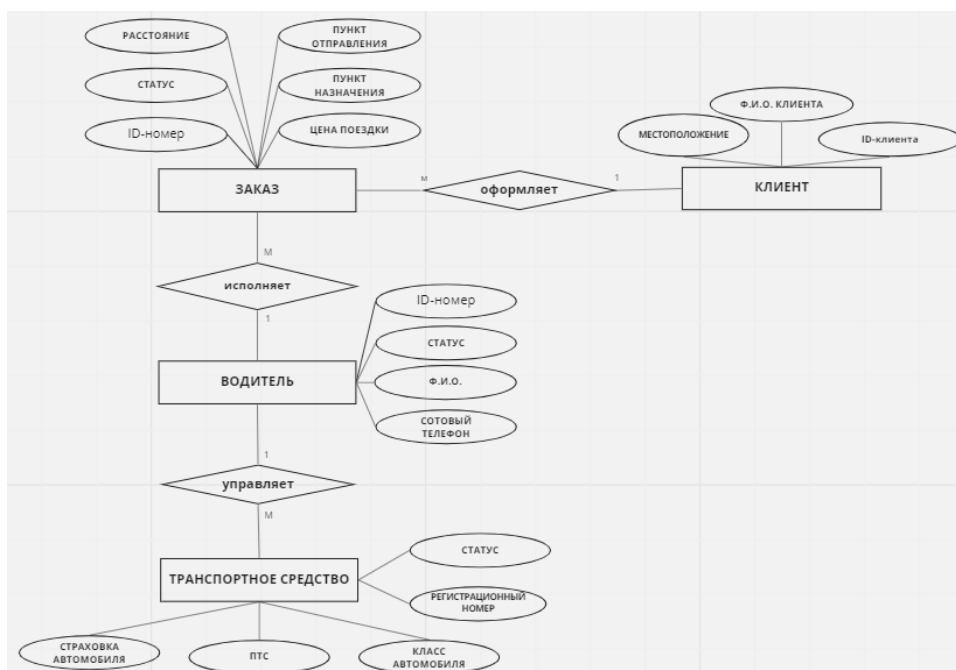


Рисунок 1.7 – Концептуальная информационная модель данных для ПО

2 ПРОЕКТИРОВАНИЕ РЕЛЯЦИОННОЙ МОДЕЛИ БАЗЫ ДАННЫХ

2.1 Определение отношений и связей между отношениями на основе концептуальной информационной модели. Первичные и внешние ключи

В соответствии с заданной предметной областью была определена соответствующая модель на уровне ключей, состоящая из таких объектов как: Клиент, Заказ, Водитель, Транспортное средство, а далее была представленная на рисунке 2.1.

Первичный ключ Клиента – “ID-номер клиента”(РК) - уникальный код в сервисе такси, создаваемый персонально для инициализации каждого клиента в базе данных.

Первичный ключ Водителя - “ID-номер водителя”(РК) - уникальный код в сервисе такси, создаваемый персонально для инициализации каждого водителя в базе данных.

Первичный ключ Заказа – “ID-номер заказа”(РК) - уникальный код в сервисе такси, создаваемый персонально для инициализации каждого заказа в базе данных. Внешними ключами заказа являются: “ID-номер клиента”(FK) и “ID-номер водителя”(FK), с их помощью реализуется связь между таблицами “Клиент” - “Заказ” и - “Заказ” - “Водитель”.

Первичный ключ Транспортного средства – “регистрационный номер”(РК) - Государственный регистрационный номер транспортного средства - индивидуальное буквенно-цифровое обозначение, присваиваемое транспортному средству регистрационным подразделением, так как он уникален для каждого автомобиля - мы можем использовать его в качестве первичного ключа для инициализации каждого

заказа в базе данных(так же автомобиль может быть инициализирован в базе данных по номеру ПТС или номеру страхового полиса, но в связи с ненадёжностью и относительно частой их сменой, логично выбрать регистрационный номер). Внешним ключом Транспортного средства является: “ID-номер водителя”(FK), с его помощью реализуется связь между таблицами “Транспортное средство” и “Водитель”.



Рисунок 2.1 – Модель на уровне ключей (КВ-уровень)

В соответствии с заданной предметной областью были определены не ключевые атрибуты таких объектов как: Клиент, Заказ, Водитель, Транспортное средство.

Не ключевые атрибуты Клиента:

- Ф.И.О;
- Местоположение.

Не ключевые атрибуты Водителя:

- Ф.И.О;
- Статус;
- Сотовый телефон.

Не ключевые атрибуты Заказа:

- Пункт отправления;
- Пункт назначения;
- Расстояние;
- Цена поездки;
- Статус.

Не ключевые атрибуты Транспортного средства:

- ПТС;
- Страховка автомобиля;
- Статус;
- Класс автомобиля.

Модель на уровне ключей (рисунок 2.1), была дополнена не ключевыми атрибутами объектов ПО (рисунок 2.2)



Рисунок 2.2 – Полная атрибутивная модель (FA-уровень)

2.2 Нормализация логической модели данных

После анализа модели на уровне ключей было выявлено следующее:

- 1) Модель удовлетворяет первой нормальной форме, так как все записи полей являются уникальными и все атрибуты атомарными.
- 2) Модель удовлетворяет второй нормальной форме, так как находится в первой нормальной форме и ее не ключевые поля полностью функционально зависят от всего первичного ключа, не являясь его частью.
- 3) Модель не удовлетворяет третьей нормальной форме, так как отсутствуют транзитивные функциональные зависимости не ключевых атрибутов от ключевых.

“Расстояние” - не на прямую зависит от “ID-номера заказа” - зависит от: “Пункт назначения” и “Пункт отправления”. Проведём нормализацию из-за несоответствия “заказа” - 3-ей нормальной форме, результат нормализации показан на рисунке 2.3.

2.3 Графическое представление логической модели данных в методологии IDEF1x

После проведения нормализации - реляционная модель данных представлена на рисунке 2.3.



Рисунок 2.3 – Нормализованная полная атрибутивная модель

2.4 Физическое проектирование с учетом выбранной СУБД. Определение типов данных атрибутов отношений

Построчное описание сущности пользователя в базе данных.

“ID_номер_клиента” – уникальный идентификатор пользователя:

- Является первичным ключом для таблицы “Клиент”.
- Имеет числовой тип данных.
- Ограничен значением: ≥ 0 .

Тип данных для этого поля установлен в счётчик (автоинкремент). Значения в данном поле будут указываться, начиная с 1 с шагом 1. Для каждого следующего первичного ключа будут заданы идентичные параметры.

Результат настройки параметров представлен на рисунке 2.4.

Сжатый тип данных	int
▶ Спецификация вычисляемого столбца	
▲ Спецификация идентификатора	Да
(Идентификатор)	Да
Начальное значение идентификатора	1
Шаг приращения идентификатора	1
▶ Спецификация полнотекстового столбца	Нет

Рисунок 2.4 — Настройка счётчика

“Ф.И.О.” – фамилия и имя пользователя соответственно:

- Представляет текстовый формат данных с размером поля 50;
- Является обязательным полем.

“Местоположение” - фактическое местонахождение клиента, делающего заказ (предполагается, что оно может отличаться от “точки отправления”):

- Представляет текстовый формат данных с размером поля 50.
- Является обязательным полем.

Описанная таблица клиента перенесена в конструктор таблиц и представлена на рисунке 2.5.

	Имя столбца	Тип данных	Разрешить ...
🔑	ID_номер_клиента	int	<input type="checkbox"/>
	[Ф.И.О.]	nvarchar(50)	<input type="checkbox"/>
	Местоположение	nvarchar(MAX)	<input type="checkbox"/>
			<input type="checkbox"/>

Рисунок 2.5 — таблица, описывающая сущность клиента в базе данных

Опишем таблицу “Водитель”, описывающую сущность водителя.

“ID_номер_водителя” – уникальный идентификатор водителя:

- Является первичным ключом для таблицы “Водитель”.
- Имеет числовой тип данных.
- Ограничен значением: ≥ 0 .

“Ф.И.О.” имеют то же описание, ограничение и тип данных, что и для пользователя.

“Статус_активности” – логическое поле, показывающее готовность/не готовность водителя принять заказ:

- Тип данных – bit;
- Может задать null значение равное, соответствующее состоянию “водитель не активен”;

- Обязательное поле.

“Телефон” – номер личного сотового телефона водителя:

- Является обязательным полем;
- Представляет текстовый формат данных с размером поля 50;
- Имеет форму записи, заданную ограничением: “(999) 999-99-99” .

Пример установки этого ограничения представлен на рисунке 2.6.

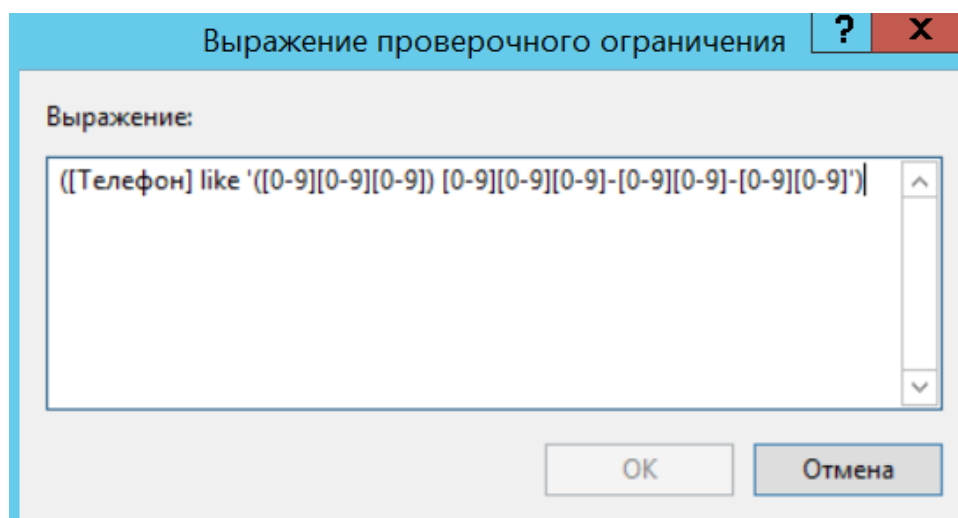


Рисунок 2.6 — Ограничение “СК_Водитель_Телефон” для поля “Телефон”

Описанная таблица водителя перенесена в конструктор таблиц и представлена на рисунке 2.7.

	Имя столбца	Тип данных	Разрешить значения NULL
🔑	ID_номер_водителя	int	<input type="checkbox"/>
	[Ф.И.О.]	nvarchar(50)	<input type="checkbox"/>
	Статус_активности	bit	<input checked="" type="checkbox"/>
	Телефон	varchar(15)	<input type="checkbox"/>

Рисунок 2.7 — таблица описывающая сущность водителя в базе данных

Построчное описание сущности транспортное средство в базе данных на основе рисунка 2.3.

“Регистрационный_номер” – уникальный государственный номерной знак идентифицирующий транспортное средство:

- Является первичным ключом для таблицы “Транспортное средство”;
- Представляет текстовый формат данных с размером поля 10;
- Ограничен значением: 1, 5, 6 символы - строчные буквы; 2, 3, 4 - цифры; 7 символ всегда - “|”; 8, 9, 10 символы - цифры.

Пример установки этого ограничения представлен на рисунке 2.8.

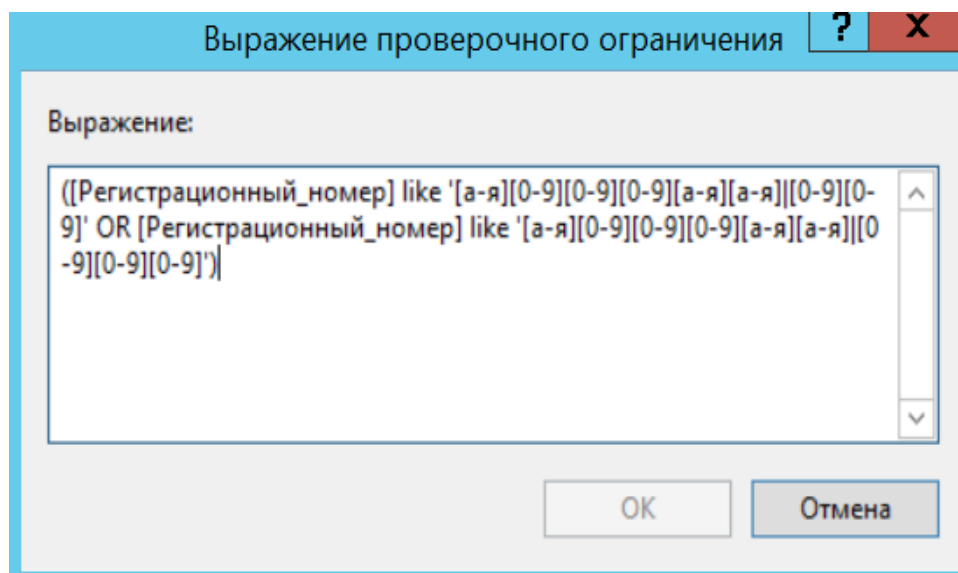


Рисунок 2.8 — Ограничение “СК_Транспортное_средство_Рег.номер” для поля “Транспортное средство”

“ID_Номер_водителя” – уникальный идентификатор курьера:

- Является внешним ключом для таблицы “Заказ”.
- Имеет числовой тип данных большого диапазона.
- Ограничен значением: ≥ 0 .

“Статус” – поле, идентифицирующее завершённость заказа:

- Представляет текстовый формат данных с размером поля 50.
- Принимает значение одной из заданной ограничением строки.
- Обязательное поле.

Далее, на рисунке 2.9 представлено ограничение “СК_Транспортное_средство_Статус”, задающее диапазон значений поля “Статус”.

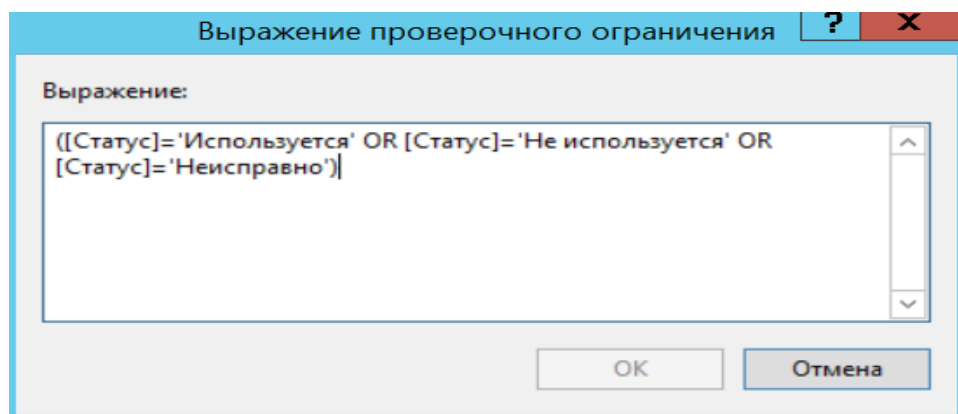


Рисунок 2.9 Ограничение “СК_Транспортное_средство_Статус” для поля “Статус”

“Класс_автомобиля” – Условный показатель, отражающий престиж автомобиля, на основе него рассчитывается цена всей поездки:

- Представляет текстовый формат данных с размером поля 8.
- Принимает значение одной из заданной ограничением строки.
- Обязательное поле.

Далее, на рисунке 2.10 представлено ограничение “СК_Транспортное_средство_Статус”, задающее диапазон значений поля “Статус”.

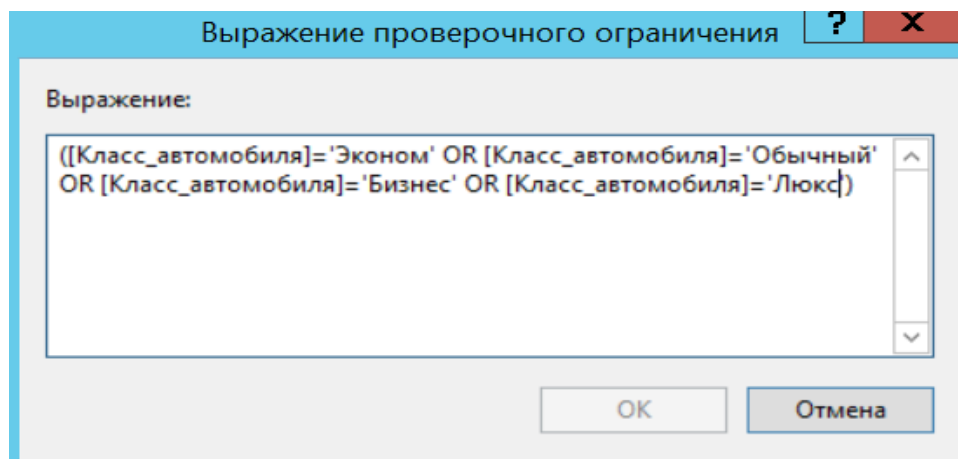


Рисунок 2.10 - Ограничение “СК_Транспортное_средство_Класс_авто”
для поля “Класс_автомобиля”

“ПТС” и “Страховка_автомобиля” – два поля - фотографии полиса страхования и паспорта транспортного средства:

- Тип данных - фотография.
- Не обязательное поле.

Описанная таблица товара перенесена в конструктор таблиц и представлена на рисунке 2.11.

	Имя столбца	Тип данных	Разрешить ...
🔑	Регистрационный_ном...	nvarchar(10)	<input type="checkbox"/>
	ID_номер_водителя	int	<input type="checkbox"/>
	Статус	nvarchar(20)	<input type="checkbox"/>
	Класс_автомобиля	nvarchar(8)	<input type="checkbox"/>
	ПТС	image	<input checked="" type="checkbox"/>
▶	Страховка_автомобиля	image	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Рисунок 2.12 — таблица, описывающая сущность транспортного средства
в базе данных

Построчное описание заказа в базе данных.

“ID_номер_заказа” – уникальный идентификатор заказа:

- Является первичным ключом для таблицы “Заказ”.
- Имеет числовой тип данных большого диапазона.
- Ограничен значением: ≥ 0 .

“Пункт_отправления” и “Пункт_назначения” – составной идентификатор маршрута:

- Являются внешними ключами для таблицы “Заказ”.
- Представляет текстовый формат данных с размером поля 50.

“ID_Номер_водителя” – уникальный идентификатор курьера:

- Является внешним ключом для таблицы “Заказ”.
- Имеет числовой тип данных большого диапазона.
- Ограничен значением: ≥ 0 .

“ID_номер_клиента” – уникальный идентификатор пользователя:

- Является внешним ключом для таблицы “Заказ”.
- Имеет числовой тип данных большого диапазона.
- Ограничен значением: ≥ 0 .

“Статус” – поле, идентифицирующее завершённость заказа:

- Представляет текстовый формат данных с размером поля 50.
- Принимает значение одной из заданной ограничением строки.
- Обязательное поле.

Далее на рисунке 2.13 представлено ограничение “СК_Заказ_Статус”, задающее диапазон значений поля “Статус”.

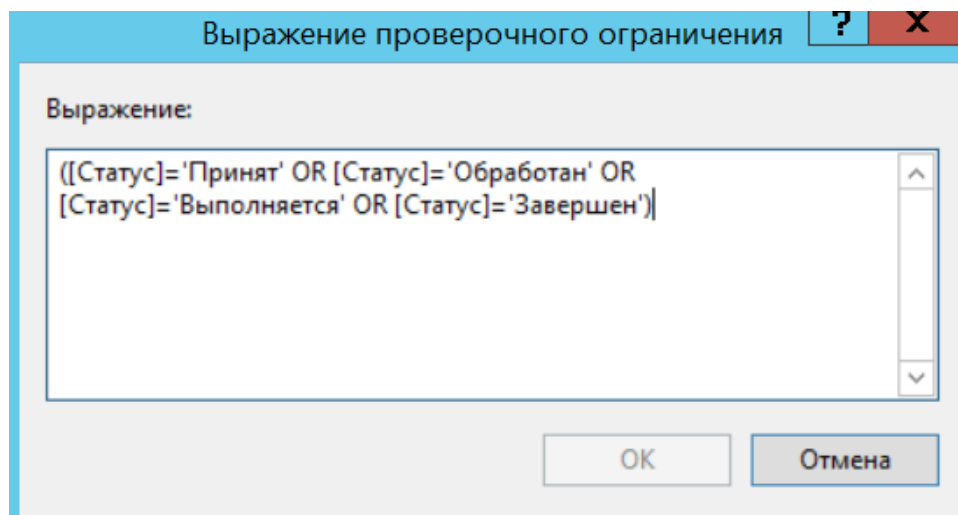


Рисунок 2.13 - Ограничение “СК_Заказ_Статус” для поля “Статус”

Описанная таблица заказа, перенесена в конструктор таблиц и представлена на рисунке 2.14.

	Имя столбца	Тип данных	Разрешить ...
🔑	ID_номер_заказа	int	<input type="checkbox"/>
	Пункт_отправления	nvarchar(50)	<input type="checkbox"/>
	Пункт_назначения	nvarchar(50)	<input type="checkbox"/>
	ID_Номер_водителя	int	<input type="checkbox"/>
	ID_номер_клиента	int	<input type="checkbox"/>
▶	Статус	nvarchar(13)	<input type="checkbox"/>
			<input type="checkbox"/>

Рисунок 2.14 - Таблица, описывающая сущность заказа в базе данных

Построчное описание таблицы маршрута на основе рисунка 3.1

“Пункт_отправления” и “Пункт_назначения” – составной идентификатор маршрута:

- Являются внешними ключами для таблицы “Заказ”.
- Представляет текстовый формат данных с размером поля 50.

“Расстояние, км” – длина маршрута от точки отправления до точки назначения:

- Имеет числовой тип данных.
- Ограничен значением: ≥ 0 .
- Является обязательным полем.

“Цена_поездки, руб” – количество денежных средств(в рублях), которое должен отдать(перевести) клиент, водителю(перевести на его счёт) за поездку:

- Ограничен значением: ≥ 0 .
- Является обязательным полем.
- Имеет числовой тип данных.

Описанная таблица маршрута перенесена в конструктор таблиц и представлена на рисунке 2.15.

	Имя столбца	Тип данных	Разрешить ...
►	Пункт_отправления	nvarchar(50)	<input type="checkbox"/>
►	Пункт_назначения	nvarchar(50)	<input type="checkbox"/>
	[Расстояние, км]	real	<input type="checkbox"/>
	[Цена_поездки, руб]	money	<input type="checkbox"/>
			<input type="checkbox"/>

Рисунок 2.15 — таблица, описывающая маршрута в базе данных

2.5 Графическое представление физической модели данных в методологии IDEF1x

Создадим диаграмму для нашей базы данных для системы управления сервисом такси, добавив на нее таблицы: “Маршрут”, “Транспортное_средство”, “Клиент”, “Водитель” и “Заказ” (Рисунок 2.16).

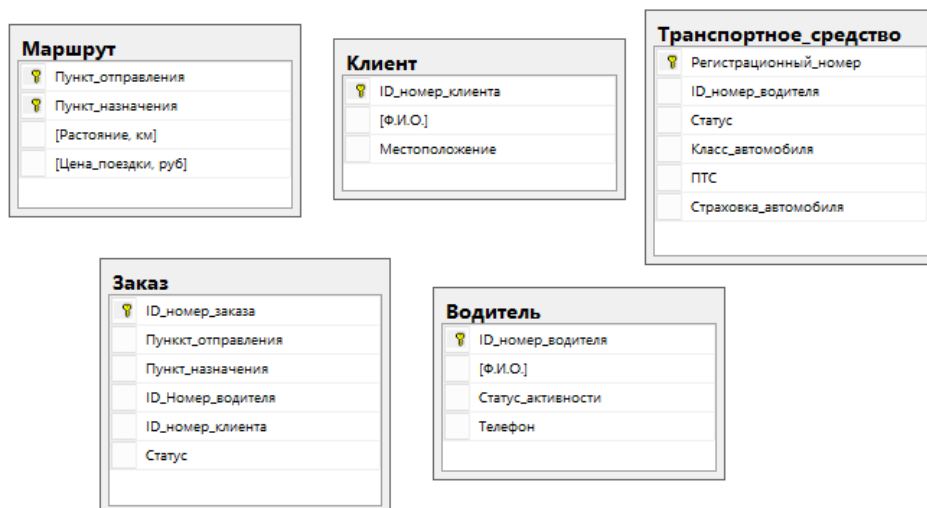


Рисунок 2.16 — диаграмма базы данных без межтабличной связи

Теперь установим связь между таблицами базы данных согласно структуре на рисунке 2.3. Ключевое поле “ID_номер_клиента” родительской таблицы “Клиент” связано с дочерней таблицей “Заказ” через поле “ID_номер_клиента”.

Устанавливаем механизм каскадного обновления связанных записей, так как “ID_номер_клиента” является типом данных строка и при этом является первичным ключом - отсутствие связанного обновления приведёт к ошибке.

Механизм каскадного удаления связанных записей не устанавливаем, чтобы при удалении данных о клиенте, связанные с ним заказы автоматически не удалялись.

Результат данных настроек представлен на рисунке 2.17.

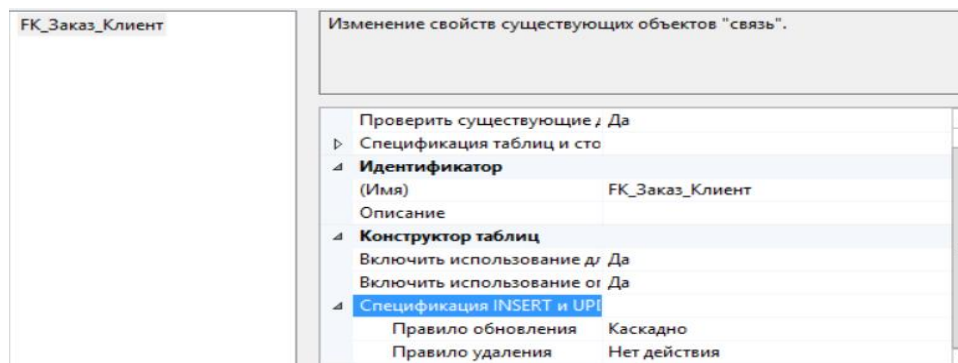


Рисунок 2.17 — Настройка табличной связи

Аналогично установлена связь для оставшихся таблиц.

От дочерней таблицы “Заказ” к родительской “Водитель” через поле “ID_номер_водителя”.

От дочерней таблицы “Заказ” к родительской “Маршрут” через поля “Пункт_отправления” и “Пункт_назначения”.

От дочерней таблицы “Транспортное_средство” к родительской “Водитель” через поле “ID_номер_водителя”.

Окончательная диаграмма базы данных представлен на рисунке 2.18.

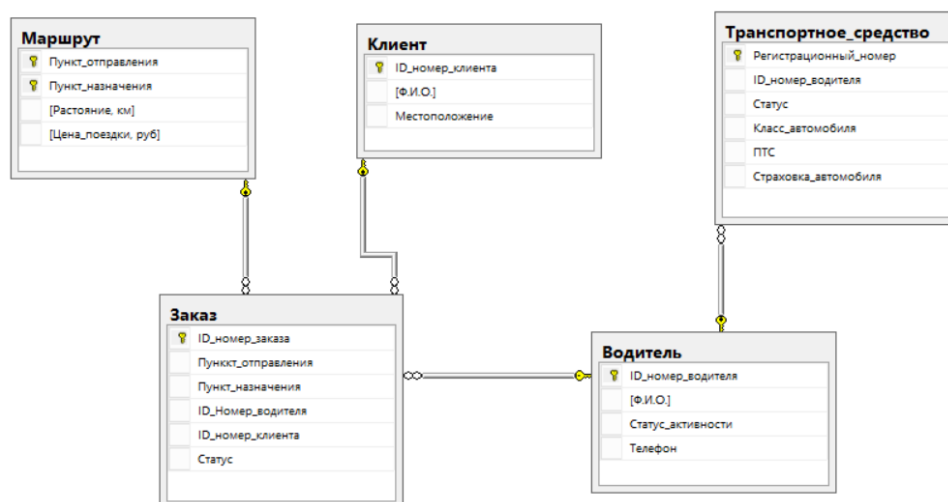


Рисунок 2.18 — диаграмма базы данных для сервиса такси

2.6 Представление физической модели данных на языке SQL

Для того чтобы заполнять данные в определенной таблице, выделим желаемую таблицу в узле Таблицы (Tables), нажмём правую кнопку мыши и выберем пункт изменить первые 200 строк и добавим Водителя со следующими данными:

- ID_номер_водителя: 4.

- [Ф.И.О.]: Федотов Константин Андреевич.
- Статус_активности: False.
- Телефон: (999) 999-99-94.

Результат представлен на рисунке 2.19.

ID_номер_вод...	Ф.И.О.	Статус_активн...	Телефон
1	Гусев Данила ...	True	(999) 999-99-91
2	Алиев Констан...	True	(999) 999-99-92
3	Мамедов Вале...	True	(999) 999-99-93
4	Федотов Конст...	False	(999) 999-99-94
8	rrrrrr pp pp	False	(999) 899-99-89
NULL	NULL	NULL	NULL

Рисунок 2.19 — Водителях в базе данных

При заполнении данных в данной таблицы были наложены ограничения целостности, нарушение которых будут приводить к соответствующим ошибкам, например:

- Если повторить значения в первичных ключах.
- Если оставить пустое или отрицательное значение в первичном ключе.
- В значениях внешних ключей указаны значения несуществующих полей.
- Статус_активности: ограничен значением: “True” и “False”.
- Не заполнены обязательные поля.
- Телефон введён некорректно.

Например, если при заполнении таблицы с водителями ввести некорректное значение в поле “Статус_активности”, то запрос будет отклонён со следующей ошибкой (Рисунок 2.20).

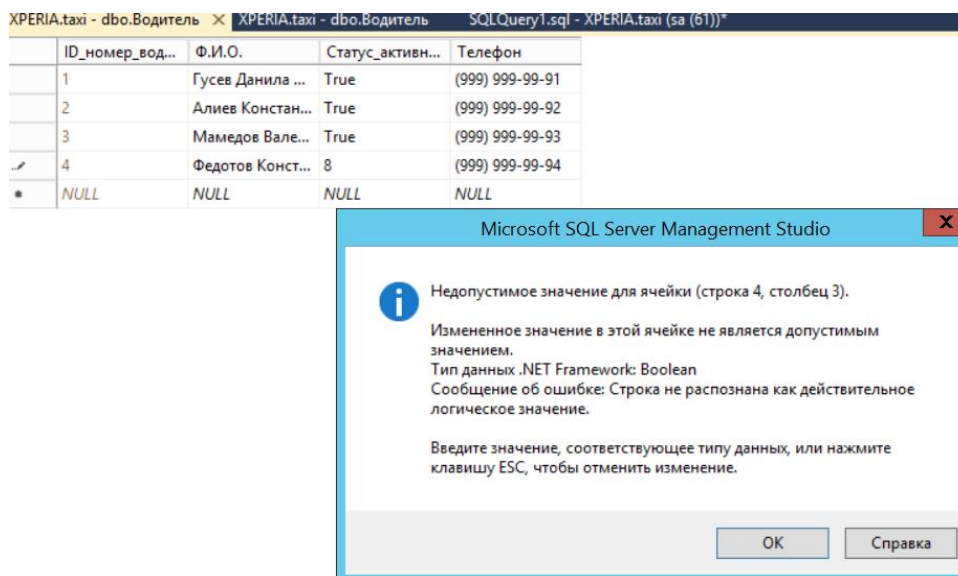


Рисунок 2.20 — Ошибка при заполнении информации о водителе в базе данных

Аналогично заполним оставшиеся таблицы. Результат представлен на рисунке 2.21, 2.22, 2.23, 2.24.

Пункт_отправ...	Пункт_назнач...	Расстояние, км	Цена_поездки,...
пер. Карпова 2	пр. Победы 44	11	275,0000
пер. Карпова 2	ул. Нанаева 102	5	125,0000
пер. Комсомо...	пр. Победы 44	7	175,0000
ул. Нанаева 102	пр. Горлова 2	4,5	112,5000
ул. Победы 56	пер. Совета 1	2	50,0000
NULL	NULL	NULL	NULL

Рисунок 2.21 — Информация о маршрутах в базе

ID_номер_кли...	Ф.И.О.	Местоположе...
1	Ахметов Вален...	пер. Карпова 2
2	Носова Людм...	ул. Победы 56
3	Пролов Никит...	пер. Комсомо...
NULL	NULL	NULL

Рисунок 2.22 — Информация о клиентах в базе

ID_номер_зака...	Пункт_отпра...	Пункт_назнач...	ID_Номер_вод...	ID_номер_кли...	Статус
5	пер. Карпова 2	ул. Нанаева 102	1	1	Завершен
8	пер. Комсомо...	пр. Победы 44	1	3	Обработан
14	ул. Победы 56	пер. Совета 1	2	2	Выполняется
20	ул. Нанаева 102	пр. Горлова 2	1	2	Завершен
23	пер. Карпова 2	пр. Победы 44	3	2	Выполняется
NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 2.23 — Информация о заказах в базе

д565тп 77	4	Не используется	Обычный	<Двоичные да...	<Двоичные да...
е613оо 56	2	Используется	Обычный	<Двоичные да...	<Двоичные да...
к294ос 777	1	Не используется	Эконом	NULL	NULL
к997шк 77	1	Неисправно	Люкс	NULL	NULL
н217рт 777	3	Используется	Люкс	NULL	NULL
о531шг 75	4	Неисправно	Люкс	<Двоичные да...	<Двоичные да...
NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 2.24 — Информация о Транспортных средствах в базе

2.7 Представление политики безопасности пользователей на языке SQL

Политика безопасности базы данных заключается в создании и распределении ролей между пользователями. Для базы данных необходимы три роли с разными уровнями доступа. Роль «Администратор» - предоставляет доступ ко всем таблицам базы данных. Предоставляется администратору сервиса. Роль «Водитель» - предоставляет доступ к большинству таблиц базы данных, необходима для работы с данными заказах, маршрутах и транспортных средствах, но исключает из доступа редактирование информации о заказах и маршрутах. Роль «Пользователь» - обладает наименьшим доступом к базе данных, необходима для создания записей и регистрации новых заказов, но исключает из доступа таблицы, содержащие информацию о водителях, маршрутах и транспортных средствах. Матрица доступа к объектам базы данных представлена в таблице 2.1

Таблица 2.1 — Матрица доступа к объектам базы данных

		Таблицы			
		Клиент	Маршрут	Заказ	Транспортное средство
Роли	Администратор	Чтение Запись Редактирование	Чтение Запись Редактирование	Чтение Запись Редактирование	Чтение Запись Редактирование
	Водитель	Чтение	Чтение Запись	Чтение Запись	Чтение Запись Редактирование
	Пользователь	Чтение	Чтение	Чтение Запись	Чтение

3 ПРОГРАММНЫЙ КОМПЛЕКС ДЛЯ РАБОТЫ С СУБД

3.1 Создание приложения для работы с данными сервиса такси

Разрабатываемое приложение предназначено для работы с базой данных для системы управления сервисом такси, её модель представлена на рисунке 3.1. В качестве СУБД используется Microsoft SQL Server.

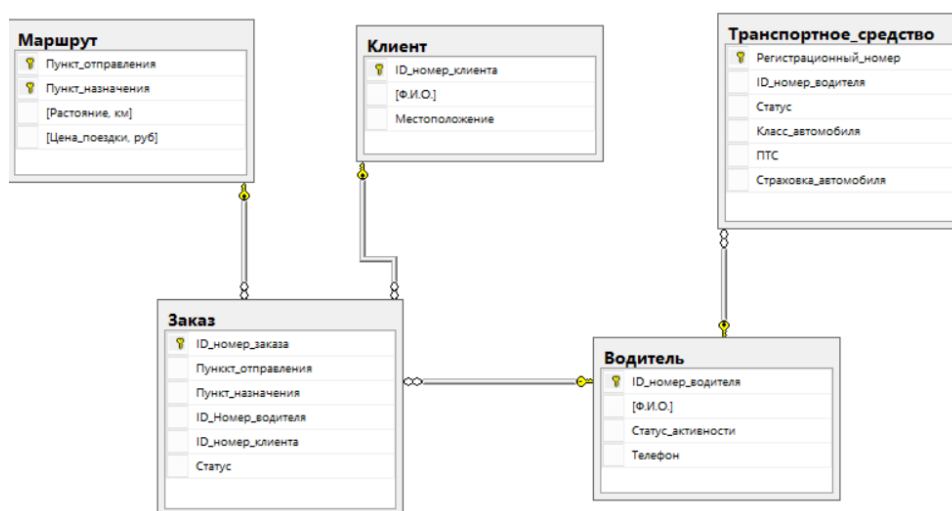


Рисунок 3.1 — Реляционная модель данных в методологии IDEF1x

В качестве среды разработки используется Visual Studio. Созданное оконное приложение на технологии Windows Forms Application представлено на рисунке 3.2.

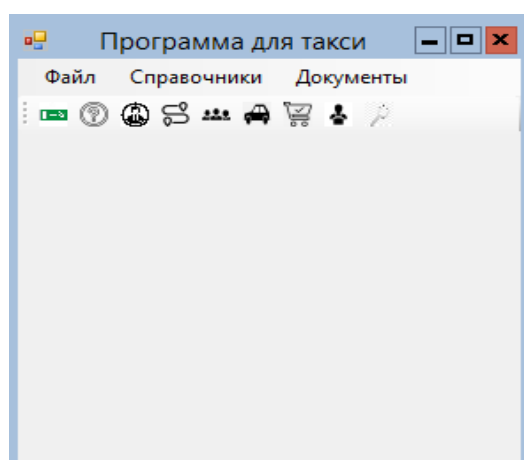


Рисунок 3.2 — Вид проекта в Visual Studio

3.2 Подготовка главной формы

Главное меню приложения создаётся с использованием компонента MenuStrip во вкладке ToolBox. Имеем пункты “Выход” и “О программе”, для меню “Файл”. Для данных пунктов устанавливаем пиктограммы через свойство Image. А также сочетания клавиш с помощью параметра ShortcutKeys. Результат созданного меню представлен на рисунке 3.3.

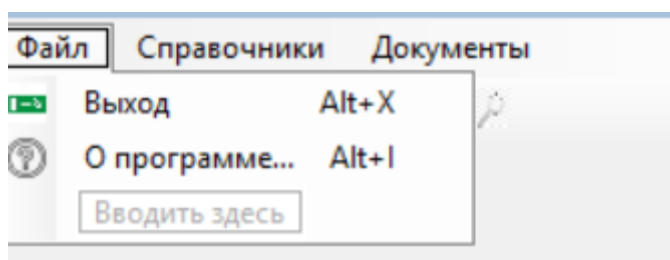


Рисунок 3.3 — Основное меню приложения

Запрограммированные обработчики для события Click (клик мышью), а также метод в событии (вкладка Events) FormClosing у MainForm вызов диалога, подтверждающего закрытие программы, представлен на рисунке 3.4.

```
private void ExitToolStripMenuItem_Click(object sender, EventArgs e)
{
    Close();
}

private void MainForm_FormClosing(object sender, FormClosingEventArgs e)
{
    DialogResult result = MessageBox.Show("Вы хотите закрыть программу?", "Внимание",
        MessageBoxButtons.YesNo, MessageBoxIcon.Question);
    e.Cancel = (result != DialogResult.Yes);
}

private void AboutToolStripMenuItem_Click(object sender, EventArgs e)
{
    MessageBox.Show("(С)ТУСУР,КСУП,Белошицкий Арём Павлович,580-1,2022", "О программе",
        MessageBoxButtons.OK, MessageBoxIcon.Information);
}
```

Рисунок 3.4 — Обработчики для событий

По аналогии создадим пункты меню для панели инструментов, а также для контекстного (локального, всплывающего) меню. И вызовем в них ранее написанные обработчики (рисунок 2.4), чтобы дважды данные действия не программировать.

3.3 Подключение к базе данных

Выполним подключение к базе данных через меню Project/Add New Data Source и выберем Microsoft SQL Server. Далее укажем имя сервера (Server name), режим необходимой аутентификации (Log on to the server), имя базы данных (Select or enter a database name). Результат проверки подключения к базе данных представлен на рисунке 3.5.

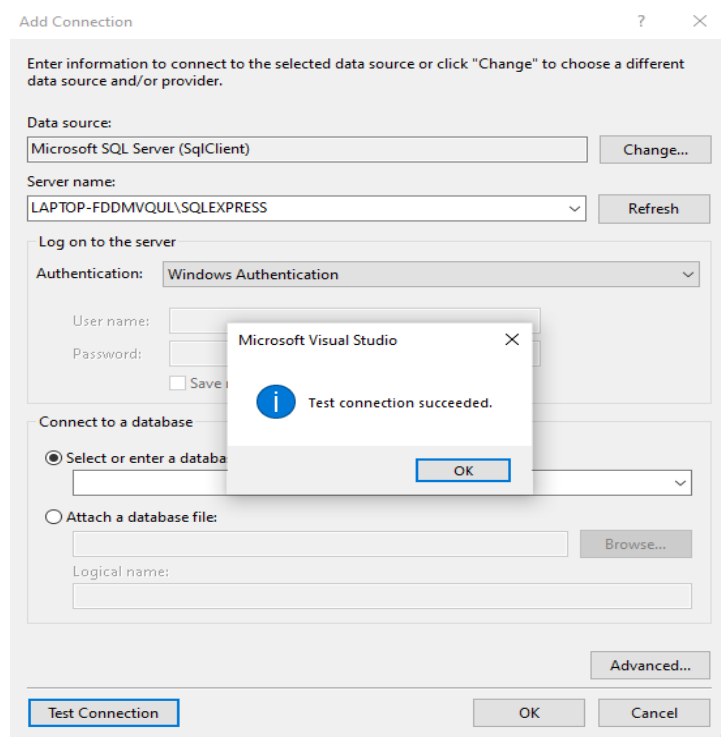


Рисунок 3.5 — Успешное подключение к базе данных

Таким образом, нами сформирован набор данных DataSet, являющийся источником данных (DataSource) в приложении. Объекты типа DataAdapter позволяют проводить основные операции в наборе данных

соответствующих объектов БД. TableAdapter позволяет работать с данными в наборе данных соответствующих таблиц БД через соответствующие свойства. Как видно на рисунке 3.6, сформированы классы таблиц, адаптеры и методы Fill и GetData.

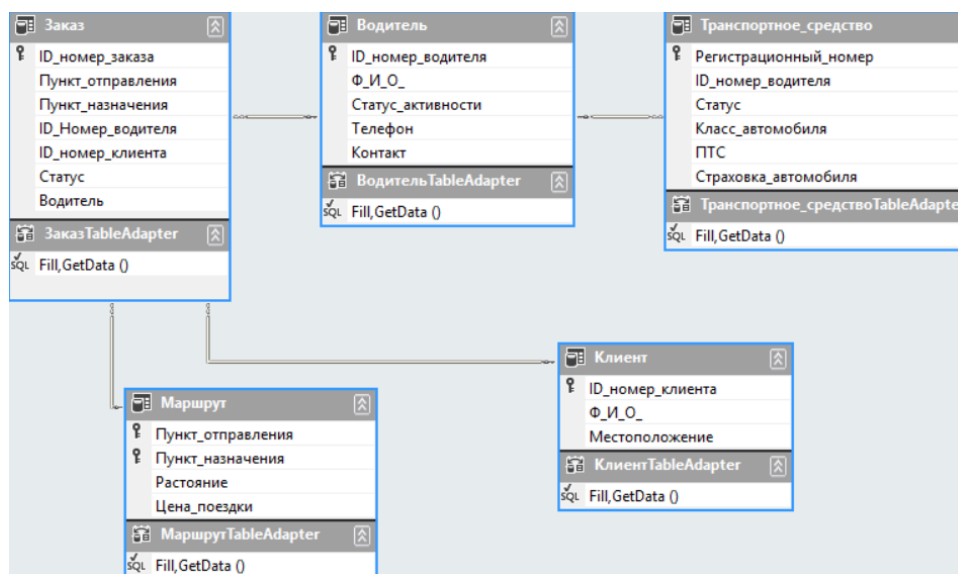


Рисунок 3.6 — Сформированная база данных в приложении

3.4 Отображение данных в приложении

Будем показывать информацию по транспортным средствам в отдельной форме. Для этого добавим в проект новую форму через пункт Project\Add\Windows Forms. Зададим для новой формы имя CourierForm. На данной форме будем отображать данные транспортного средства. Для этого через источник данных выберем таблицу транспортного средства из базы данных, и добавим её на созданную форму представления данных, благодаря чему каждое поле базы данных привяжется к отдельному компоненту форму. В результате получится форма представленная на рисунке 3.7.

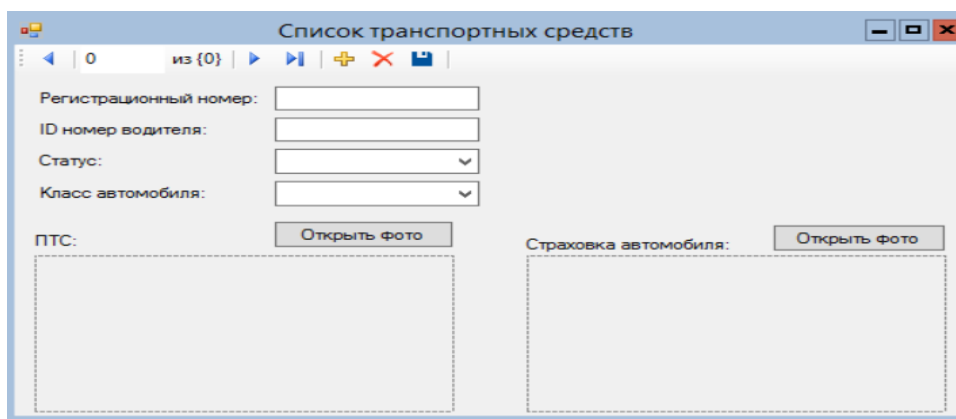


Рисунок 3.7 — Форма для отображения информации транспортного средства

В нашей базе данных есть поля фото для хранения фотографий ПТС и Страхового полиса автомобиля. Для них отображение используется `fotoPictureBox`. Для удобства загрузки фотографии из файла разместим рядом с `fotoPictureBox` компонент `button`. Для загрузки фото из файла диалог `openFileDialog1`. Запрограммируем нажатие кнопки `buttonOpenPhoto` для загрузки фото в `fotoPictureBox`, через диалог `openFileDialog1` (рисунок 3.6). Он позволяет загрузить файлы следующих форматов: BMP, GIF, EXIF, JPG, PNG, TIFF.

```
private void ButtonOpenPhoto_Click(object sender, EventArgs e)
{
    OpenFileDialogPhoto.Title = "Укажите файл для фото";
    if (OpenFileDialogPhoto.ShowDialog() == DialogResult.OK)
    {
        try
        {
            fotoPictureBox.Image = new Bitmap(OpenFileDialogPhoto.FileName);
        }
        catch
        {
            MessageBox.Show("Выбран не тот формат файла", "Ошибка",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}
```

Рисунок 3.6 — Обработчик для нажатия кнопки

Для отображения формы используется метод Show(), благодаря чему возможно переходить от открытой формы для транспортного средства к главной форме и наоборот. Из-за чего в запущенной программе пользователь может создавать неограниченное количество различных второстепенных форм. Для устранения этого недостатка было решено использовать шаблон программирования под названием «Одиночка» (англ. Singleton). Он гарантирует, наличие только одного экземпляра класса, и предоставляет ссылку на этот единственный экземпляр (объект). Реализация шаблон проектирования «Одиночка» (англ. Singleton) для класса CourierForm представлена на рисунке 3.7.

```
private static CourierForm _uniqueInstance;

private CourierForm()
{
    InitializeComponent();
}

public static CourierForm GetInstance()
{
    if (_uniqueInstance is null || _uniqueInstance.IsDisposed)
    {
        _uniqueInstance = new CourierForm();
    }
    return _uniqueInstance;
}
```

Рисунок 3.7 — Реализация шаблона «Одиночка» в классе CourierForm

Добавим еще форму в проект для отображения данных по заказам FormOrder. Будем использовать для этого DataGridView, так как количество заказов может исчисляться десятками тысяч. Вследствие чего их отображения примет вид таблицы, удобной для анализа и взаимодействия человека с таким объёмом данных. Установим тип колонки, отвечающей за статус заказа, в DataGridViewComboBoxColumn в котором можно будет выбрать четыре значения: “Принят”, “Обработан”, “Выполняется” или

“Завершен”. Также установим для формы свойство Dock в Fill (всю область). Теперь форма имеет вид представленный на рисунке 3.8.

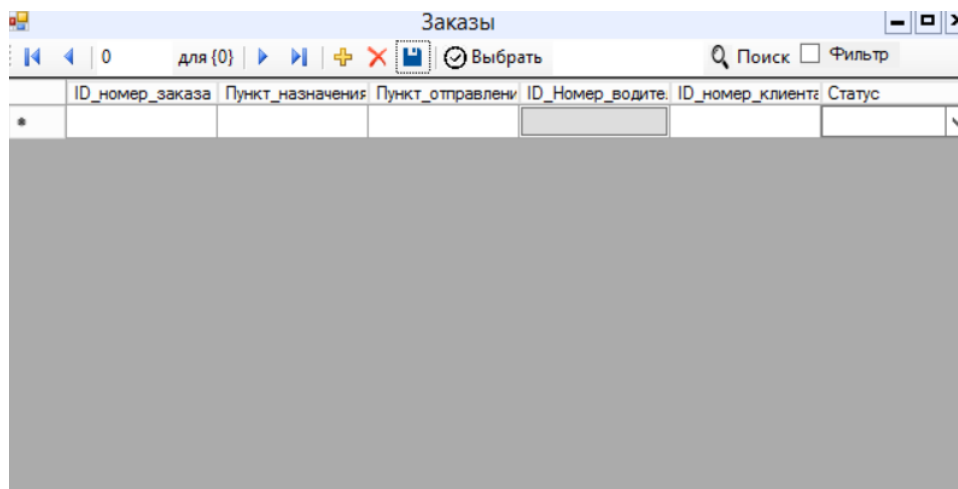


Рисунок 3.8 — Форма для отображения информации о заказах

Прделаем аналогичные действия для оставшихся форм: FormDishesList, FormRoute, FormClient. Они представлена на рисунках 3.9, 3.10, 3.11 соответственно.

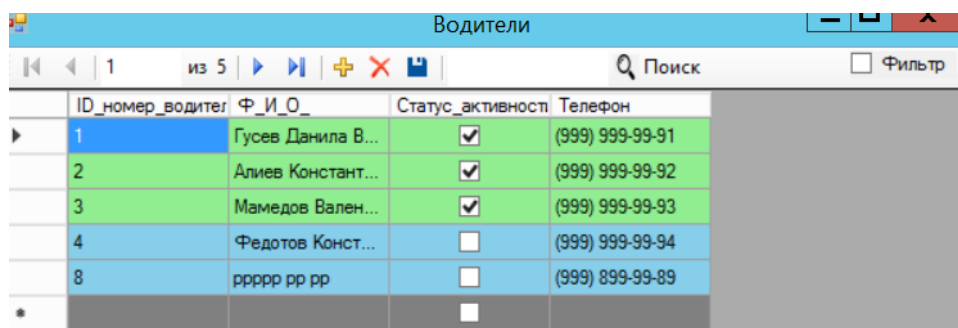


Рисунок 3.9 — Форма для отображения информации о водителях

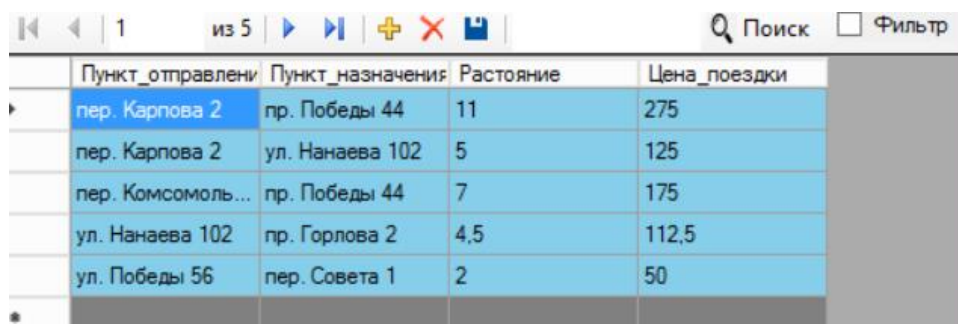


Рисунок 3.10 — Форма для отображения информации о маршрутах

ID_номер_клиента	Ф_И_О_	Местоположение
1	Ахметов Валентин Игоревич	пер. Карпова 2
2	Носова Людмила Викторов...	ул. Победы 56
3	Пролов Никита Константин...	пер. Комсомольски...

Рисунок 3.11 — Форма для отображения информации о клиентах

3.5 Реляционная модель данных

Добавим еще форму в проект для отображения данных по заказам OrderForm. Также добавим на форму информацию по Транспортным средствам в данном заказе. Добавим на форму информацию о заказах, для этого через источник данных, выберем таблицу водитель из базы данных, и добавим её на созданную форму представления данных, благодаря чему каждое поле базы данных привяжется к отдельному компоненту форму. Разместив на панели дополнительный компонент GroupBox, для него укажем Name: groupBox2 и свойство Text: Транспортные средства. Добавим на ранее созданный компонент таблицу по водителям в выбранном заказе. Форма для работы с данными по связанным таблицам представлена на рисунке 3.12.

Рисунок 3.12 — Форма для работы с данными по связанным таблицам

Чтобы при добавлении нового транспортного средства, Телефон и Статус активности были присвоены по умолчанию (DefaultValue) пропишем в обработчике события Form6_Load следующее (рисунок 3.13).

```
private void Form6_Load (object sender, EventArgs e)
{
    taxiDataSet.Водитель.Columns["Телефон"].DefaultValue = "(999) 999-99-99";
    taxiDataSet.Водитель.Columns["Статус_активности"].DefaultValue = false;
}
```

Рисунок 3.13 — Обработка события по загрузке формы

3.6 Обработка исключений и ошибок

Реализовать обработку исключений при оформлении данных в таблице Заказы, для случаев когда поле в “ID_номер_водителя” введён не существующих водитель. Реализация обработки исключений с конструкцией try...catch представлена на рисунке 3.14.

```
private void водительBindingNavigatorSaveItem_Click (object sender, EventArgs e)
{
    try
    {
        this.Validate();
        this.водительBindingSource.EndEdit();
        this.tableAdapterManager.UpdateAll(this.deliveryDataSet);
    }
    catch (Exception error)
    {
        MessageBox.Show(error.Message, "Ошибка",
            MessageBoxButtons.OK,
            MessageBoxIcon.Error);
    }
}
```

Рисунок 3.14 — Обработка исключений при сохранении данных

Теперь будет обработка следующего исключений: в таблице заказ указан не существующих код водитель(Рисунок 3.15).

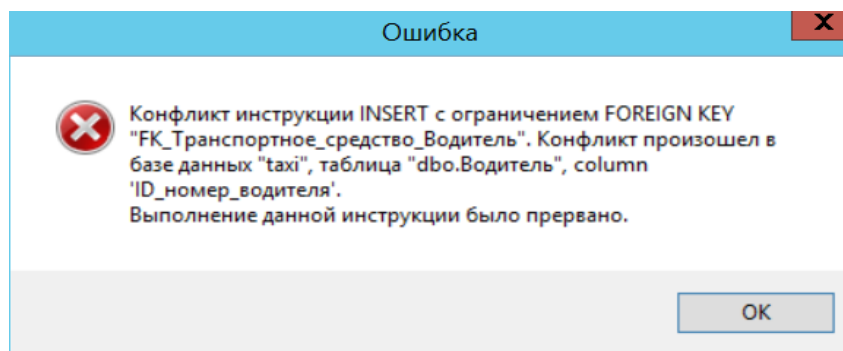


Рисунок 3.15 — Пример обработки исключения

Реализовать обработку исключений при оформлении данных в таблице Водитель, для случаев когда поле в “Ф_И_О_” не введено. Исключение вызванное этой ошибкой представлена на рисунке 3.16.

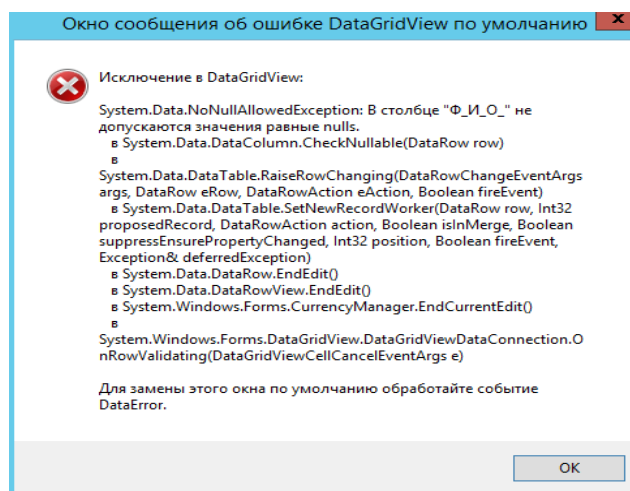


Рисунок 3.16 — Пример обработки исключения

По аналогии запрограммируем обработку исключений для оставшихся форм.

3.7 Поиск и фильтрация данных

Реализуем поиск и фильтрацию данных необходимого клиента. Данная возможность пригодится, так как количество клиентов может исчисляться десятками тысяч. Будем осуществлять поиск и фильтр

заданного в строке ввода значения по выбранной пользователем колонке. Для этого разместим на панели разделитель через Add ToolStripButton, а также TextBox, Button, CheckBox. Получившийся внешний вид формы и работа фильтрации представлен на рисунке 3.17.

ID_номер_водител	Ф_И_О_	Статус_активности	Телефон
1	Гусев Данила В...	<input checked="" type="checkbox"/>	(999) 999-99-91
2	Алиев Констант...	<input checked="" type="checkbox"/>	(999) 999-99-92
3	Мамедов Вален...	<input checked="" type="checkbox"/>	(999) 999-99-93
4	Федотов Констант...	<input type="checkbox"/>	(999) 999-99-94
8	rrrrr rr rr	<input type="checkbox"/>	(999) 899-99-89

Рисунок 3.17 — Форма для представления информации о водителях

Как было упомянуто ранее, наша задача осуществлять поиск заданного в строке ввода ToolStripTextBoxFind значения по выбранной пользователем в DataGridView колонке. Для этого пропишем обработчик для отбора выбранной пользователем колонки в DataGridView (Рисунок 3.18), результат работы покажем на Рисунке 3.19.

```
private void ToolStripButtonFind_Click(object sender, EventArgs e) {
    if (ToolStripTextBoxFind.Text == "") {
        MessageBox.Show("Вы ничего не задали", "Внимание", MessageBoxButtons.OK,
            MessageBoxIcon.Information);
        return;
    }
    int indexPos;
    Try {
        indexPos = водительBindingSource.Find(GetSelectedFieldName(), toolStripTextBoxFind.Text);
    }
    catch (Exception error) {
        MessageBox.Show("Ошибка поиска \n" + error.Message);
        return;
    }
    if (indexPos > -1) {
        водительBindingSource.Position = indexPos;
    }
    Else {
        MessageBox.Show("Таких сотрудников нет", "Внимание", MessageBoxButtons.OK,
            MessageBoxIcon.Information);
        водительBindingSource.Position = 0;
    }
}
```

Рисунок 3.18 — Обработчик фильтрации результатов

ID_номер_водител	Ф_И_О_	Статус_активности	Телефон
3	Мамедов Вален...	<input checked="" type="checkbox"/>	(999) 999-99-93

Рисунок 3.19 — Результат работы фильтрации

3.8 Реализация выбора пользователя в заказе

При работе с заказами не всегда удобно помнить код необходимого водителя. Кроме того, следует отметить, что нами была установлена ссылочная целостность в базе данных, т.е. должен быть указан код существующего в базе данных водителя, иначе будет ошибка при работе с данными. В нашем случае, чтобы посмотреть код необходимого водителя, нужно перейти на форму “Водитель” и посмотреть для него код. Данную ситуацию можно существенно упростить, будем организовывать вызов формы “Водитель” из формы “Заказы”.

Добавим кнопку на форму “Заказы”, по нажатию на которую будем показывать форму с данными по водителям. В данной форме пользователю будет необходимо выбрать водителя и его данные (ID номер водителя), которые заносятся в соответствующий заказ. Реализуем логику следующим образом: если пользователь нажал кнопку “Выбрать” (toolStripButtonOK), то запоминаем ID выбранного водителя. Если он закрыл форму, нажав на крестик, то ID водителя не запоминаем, считая, что пользователь передумал выбирать. Реализация данной логики представлена на рисунке 3.20 и 3.21, для формы “Заказ” и “Водитель” соответственно. Аналогичную форму создаём для клиента, и проводим над ней аналогичные манипуляции:

```

private void водителиButton_Click (object sender, EventArgs e)
{
    long id = -1;
    if (((DataRowView)заказBindingSource.Current)[" ID_номер_водителя "].ToString() != "")
    {
        id = (long)(((DataRowView)заказBindingSource.Current)[" ID_номер_водителя "]);
    }
    id = FormDishesList.fw.ShowSelectForm(id);
    if (id >= 0)
    {
        MessageBox.Show(id.ToString());
        ((DataRowView)заказBindingSource.Current)[" ID_номер_водителя "] = id;
        заказBindingSource.EndEdit();
    }
}

```

Рисунок 3.20 — Реализация открытия формы для выбора водителя

```

public long idCurrent = -1;
public long ShowSelectForm(long id)
{
    toolStripButtonOK.Visible = true;
    idCurrent = id;
    if (ShowDialog() == DialogResult.OK)
    {
        Return (int)((DataRowView) заказBindingSource.Current)[" ID_номер_водителя "];
    }
    else
    {
        return -1;
    }
}
private void toolStripButtonOK_Click(object sender, EventArgs e)
{
    DialogResult = DialogResult.OK;
}

```

Рисунок 3.21 — Реализация обработки выбранного водителя

3.9 Создание вычисляемых колонок

Создадим вычисляемую колонки для таблицы “Маршрут” - “Цена_поездки”. Она будет вычисляться по формуле: расстояние * 25. У добавленной колонки в сетке таблицы, в окне Properties, зададим свойство Expression. Введём значение вычисляемого выражения, в нашем случае это “Расстояние*25”. Аналогично создадим вычисляемые колонки: “Водитель”

- “Контакт”, для которой зададим свойство Expression: “ID_номер_водителя+' '+Ф_И_О_+' '+Телефон”. И “Заказ” - “Водитель”, для которой зададим свойство Expression: “Parent(FK_Заказ_Водитель).[Ф_И_О_]+' '+Parent(FK_Заказ_Водитель).[Телефон]”. Результат созданных полей представлен на рисунке 3.22.

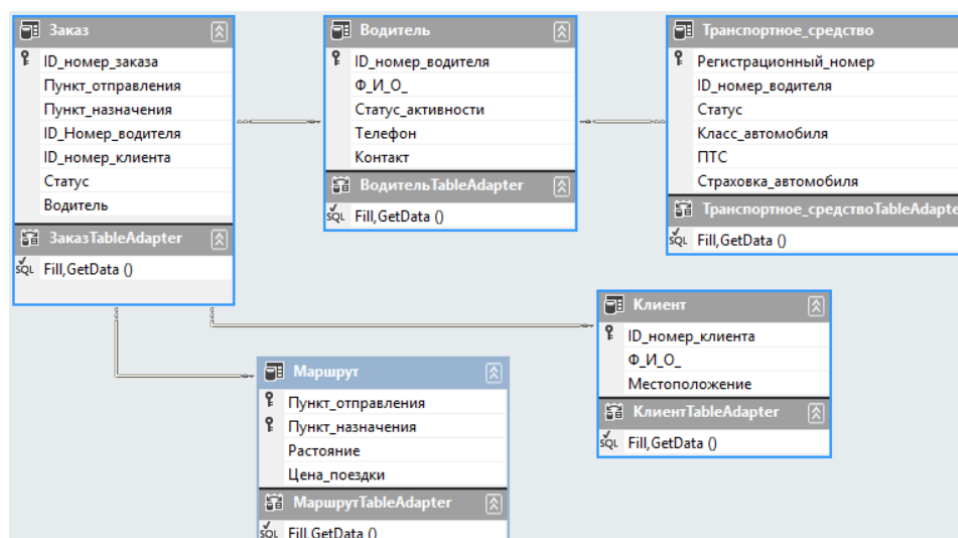


Рисунок 3.22 — Автоматически вычисляемые колонки

3.10 Создание подстановочный полей

Теперь у нас есть вычисляемое поле “Контакт” и “Водитель”. Можно для удобства пользователей сделать на форме о заказах показ не только кода сотрудника, но и его Контактных данных. Для этого добавим из источника данных DataSources на форму данное полевые из таблицы в виде метки (Label). Результат представлен на рисунке 3.23.

ID_номер_заказа	Пункт_назначения	Пункт_отправления	ID_Номер_водителя	ID_номер_клиента	Статус
6	ул. Нанаева 102	пер. Карпова 2	1	1	Завершен
8	пр. Победы 44	пер. Комсомоль...	1	3	Обработан
14	пер. Совета 1	ул. Победы 56	2	1	Выполняется
20	пр. Горлова 2	ул. Нанаева 102	1	2	Завершен
23	пр. Победы 44	пер. Карпова 2	3	3	Выполняется

Все водители Все клиенты Водитель: Алиев Константин Сергеевич (999) 999-99-92

Рисунок 3.23 — Пример подстановочных полей на форме

Аналогично поступим с контактными данными для формы транспортных средств(рисунок 3.24).

ID номер водителя: 2
 Ф И О : Алиев Константин И
 Статус активности: ☒ checkBox1
 Телефон: (999) 999-99-92 Контакт: Алиев Константин Сергеевич (999) 999-99-92

Транспортные средства

1 из 1

Регистрационный номер: e613ooJ56 Все водители

ID номер водителя: 2

Статус: Используется

Класс автомобиля: Обычный

ПТС: Страховка автомобиля:

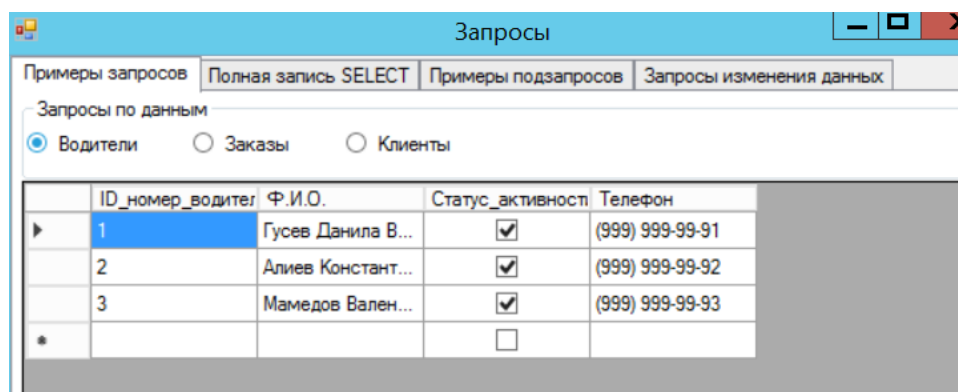
Рисунок 3.24 — Пример подстановочных полей на форме

3.11 Создании формы для запросов

Создадим для запросов отдельную форму с именем “Запросы”. С палитры компонент Toolbox, разместим на форму SqlForm компонент tabControl – многостраничный контейнер и зададим ему имя. Добавим на форму groupBoxSelect для отображения данных запросов, а так же кнопки

для выбора типа запроса и запрограммируем для выполнения следующих запросов:

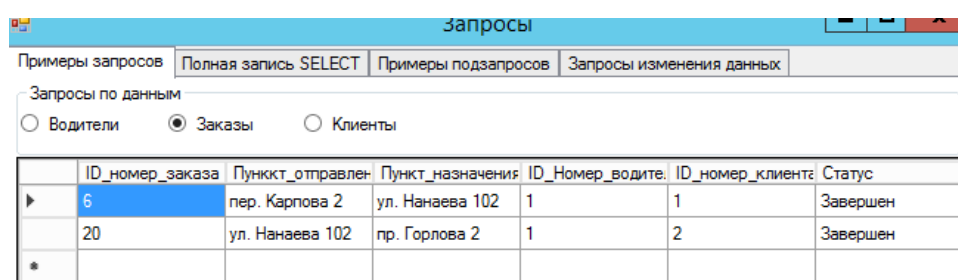
Для кнопки “Водители” - "SELECT * FROM Водитель WHERE Водитель.Статус_активности = 'True'"(все активные в данный момент водители), результат работы показан на рисунке 3.25.



	ID_номер_водител	Ф.И.О.	Статус_активности	Телефон
▶	1	Гусев Данила В...	<input checked="" type="checkbox"/>	(999) 999-99-91
	2	Алиев Констант...	<input checked="" type="checkbox"/>	(999) 999-99-92
	3	Мамедов Вален...	<input checked="" type="checkbox"/>	(999) 999-99-93
*			<input type="checkbox"/>	

Рисунок 3.25 — Результат работы запроса

Для кнопки “Заказы” - "SELECT ID_номер_заказа, Пункт_отправления, Пункт_назначения, ID_Номер_водителя, ID_номер_клиента, Статус FROM Заказ WHERE Заказ.Статус = 'Завершен'" (все завершённые на данный момент заказы), результат работы показан на рисунке 3.26.



	ID_номер_заказа	Пункт_отправлен	Пункт_назначения	ID_Номер_водите	ID_номер_клиент	Статус
▶	6	пер. Карпова 2	ул. Нанаева 102	1	1	Завершен
	20	ул. Нанаева 102	пр. Горлова 2	1	2	Завершен
*						

Рисунок 3.26— Результат работы запроса

Для кнопки “Клиенты” - "SELECT * FROM Клиент" (все клиенты), результат работы показан на рисунке 3.27.

Запросы			
Примеры запросов		Полная запись SELECT	Примеры подзапросов Запросы изменения данных
Запросы по данным			
<input type="radio"/> Водители <input type="radio"/> Заказы <input checked="" type="radio"/> Клиенты			
	ID_номер_клиента	Ф.И.О.	Местоположение
▶	1	Ахметов Валент...	пер. Карпова 2
	2	Носова Людмил...	ул. Победы 56
	3	Пролов Никита ...	пер. Комсомоль...
*			

Рисунок 3.27— Результат работы запроса

Листинг кода для запросов содержимого таблиц представлен приложении Е. Для остальных сущностей базы данных он аналогичен.

3.12 Полная запись оператора SELECT

Создадим запрос для детализированного вывода прибыли. Для этого будем суммировать стоимость всех заказов, в которых ID_номер_водителя совпадает с ID_номер_водителя водителя, чья фамилия вводится в строке ввода, а так же выше определённой суммы. Запрос будет выполняться по нажатию кнопки “Выполнить”.

Результаты выполнения запросов, отсортированных по убыванию прибыли, представлены на рисунках 3.28 и 3.29, детализированная прибыль всех водителей с сортировкой по убыванию показана на рисунке 3.30.

Запросы			
Примеры запросов		Полная запись SELECT	Примеры подзапросов Запросы изменения данных
Прибыль сотрудников			
Ф.И.О. сотрудника		Гусев Данила Владиславович	
<input checked="" type="checkbox"/> Выбрать прибыль более		100	
<input type="checkbox"/> Включить сортировку по убыванию прибыли			
<div>Прибыль сотрудников</div>			
	ID_номер_водител	Ф.И.О.	Прибыль
▶	1	Гусев Данила В...	237,5000
*			

Детализация прибыли сотрудников
☒ Прибыль водителя только по завершённым заказам
☐ Прибыль водителя по ВСЕМ заказам (по выполняющимся тоже)
☐ Показать прибыль всех водителей

Рисунок 3.28 — Детализированная прибыль только по завершённым заказам

Прибыль сотрудников

Ф.И.О. сотрудника:

☒ Выбрать прибыль более:

☐ Включить сортировку по убыванию прибыли

Детализация прибыли сотрудников

☐ Прибыль водителя только по завершённым заказам

☒ Прибыль водителя по ВСЕМ заказам (по выполняющимся тоже)

☐ Показать прибыль всех водителей

ID_номер_водител	Ф.И.О.	Прибыль
1	Гусев Данила В...	412,5000

Рисунок 3.29 — Детализированная прибыль по всем заказам

Прибыль сотрудников

Ф.И.О. сотрудника:

☐ Выбрать прибыль более:

☒ Включить сортировку по убыванию прибыли

Детализация прибыли сотрудников

☐ Прибыль водителя только по завершённым заказам

☐ Прибыль водителя по ВСЕМ заказам (по выполняющимся тоже)

☒ Показать прибыль всех водителей

ID_номер_водител	Ф.И.О.	Прибыль
1	Гусев Данила В...	412,5000
3	Мамедов Вален...	275,0000
2	Алиев Констант...	50,0000

Рисунок 3.30 — Детализированная прибыль всех водителей с сортировкой по убыванию

Листинг кода для обработки детализированного запроса по прибыли водителей представлен в приложении А.

3.13 Подзапросы

Создадим форму в которой будем просматривать подзапросы для заказа, в зависимости о того, Ф.И.О. какого водителя введены в строку ввода, будем просматривать все заказы, которые выполнил\выполняет этот водитель. Запрос будет выполняться по нажатию кнопки “Выполнить”. Результат работы коррелированного запроса представлен на рисунке 3.31, а для не коррелированного на рисунке 3.32.

	Фамилия	ID_номер_заказа	Пункт_отправлен	Пункт_назначения	ID_Номер_водите.	ID_но
▶	Гусев Данила В...	6	пер. Карпова 2	ул. Нанаева 102	1	1
	Гусев Данила В...	8	пер. Комсомоль...	пр. Победы 44	1	3
	Гусев Данила В...	20	ул. Нанаева 102	пр. Горлова 2	1	2
*						

Рисунок 3.31 — Результат выполнения коррелированного подзапроса

	ID_номер_заказа	Пункт_отправлен	Пункт_назначения	ID_Номер_водите.	ID_номер_клиента	Статус
▶	6	пер. Карпова 2	ул. Нанаева 102	1	1	Завер
	8	пер. Комсомоль...	пр. Победы 44	1	3	Обраб
	20	ул. Нанаева 102	пр. Горлова 2	1	2	Завер
*						

Рисунок 3.32 — Результат выполнения не коррелированного подзапроса

Листинг кода для обработки коррелированного и не коррелированного подзапроса представлен в Приложении Б.

3.14 Запрос на добавление данных

При добавлении данных в таблицу Транспортное средство в соответствующие поля будут добавлены соответствующие значения из соответствующих компонент на форме. Результат выполнения запроса на добавление представлен на рисунке 3.33.

Регистрационный	ID_номер_водител	Статус	Класс_автомобил	ПТС	Ст
д565от77	2	Используется	Бизнес		
д565тп77	2	Не используется	Люкс		
е613оо56	2	Используется	Обычный		
к294оо555	1	Используется	Люкс		
к294оо777	1	Не используется	Эконом		

Рисунок 3.33 — Результат выполнения запроса на добавление данных

Листинг кода для формирования параметрического запроса на добавление представлен в приложении В.

3.15 Запрос на изменение данных

По аналогии программируем запрос на изменение данных. Результат выполнения запроса изменение данных представлен на рисунке 3.34.

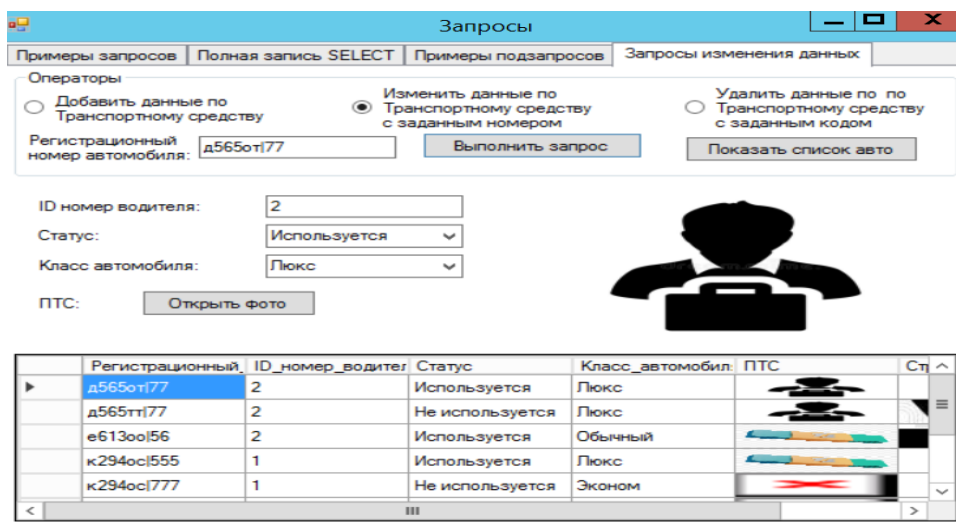


Рисунок 3.34 — Результат выполнения запроса на изменение данных

Листинг кода для формирования параметрического запроса на изменение представлен в приложении Г.

3.16 Запрос на удаление данных

По аналогии программируем запрос на удаление данных. Результат выполнения запроса на удаление данных представлен на рисунке 3.35.

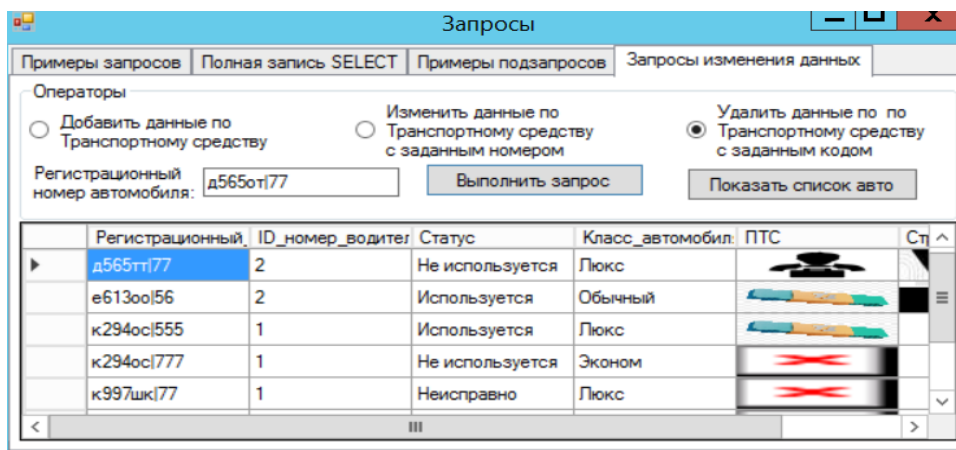


Рисунок 3.35 — Результат выполнения запроса на удаление данных

Листинг кода для формирования параметрического запроса на удаление представлен в приложении Д.

ЗАКЛЮЧЕНИЕ

В курсовой работе было проведено автоматизирование предметной области (обработка и систематизация заказа такси сервиса). Благодаря чему автоматически идет проверка активности водителя в данный момент времени, автоматически рассчитывается стоимость поездки, которая формируется в зависимости от расстояния между пунктами отправления и назначения, автоматически рассчитывается доход каждого водителя. Построена база данных для сервиса, проанализированы какие данные должны храниться в ней и как пользователь должен взаимодействовать с ней.

Были разработаны сущности и наполнены атрибутами. Разработана база данных в среде Microsoft SQL server express. Создано приложение, позволяющее проводить быстрый анализ данных за счет поиска по данным, их фильтрации и создания запросов, на Windows Form, через которое пользователь может осуществлять удобное взаимодействие с базой данных, а также выполнять запросы к ней.

Программа включает в себя базу данных, что позволяет хранить данные на цифровом носителе. В результате чего появляется возможность быстро создавать резервные копии данных.

В дальнейшем программа может быть расширена и улучшена. Например, вход через обращение к полям таблицы можно заменить на работу с ролями базы данных.

Приложение А

(обязательное)

Листинг кода для обработки детализированного запроса по прибыли
водителей

```
private void buttonF_select_Click(object sender, EventArgs e)
{
    string sqlSelect = "";
    if (radioButtonAllDriver.Checked)
    {
        sqlSelect = @"SELECT w.ID_номер_водителя, [Ф.И.О.],
sum(d.[Цена_поездки, руб]) AS Прибыль
FROM Водитель w, Заказ s, Маршрут d
WHERE (w.ID_номер_водителя = s.ID_Номер_водителя)
AND (d.Пункт_отправления = s.Пункт_отправления)
AND (d.Пункт_назначения = s.Пункт_назначения)
GROUP BY w.ID_номер_водителя, [Ф.И.О.]";
        textBoxWorker.Text = "";
    }
    else if (String.IsNullOrEmpty(textBoxWorker.Text))
    {
        MessageBox.Show("Обязательно укажите фамилию необходимого
сотрудника.\nДопустим ввод первых символов.", "Внимание",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }
    if (checkBoxMore.Checked && String.IsNullOrEmpty(textBoxMore.Text))
    {
        MessageBox.Show("Не указана прибыль в условии", "Внимание",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
        checkBoxMore.Checked = false;
        return;
    }

    if (radioButtonDet_OnlyEnd.Checked)
        sqlSelect = @"SELECT w.ID_номер_водителя, [Ф.И.О.],
sum(d.[Цена_поездки, руб]) AS Прибыль
FROM Водитель w, Заказ s, Маршрут d
WHERE (w.[Ф.И.О.] = '@last_name') AND
(w.ID_номер_водителя = s.ID_Номер_водителя)
AND (s.Статус = 'Завершен')
AND (d.Пункт_отправления = s.Пункт_отправления)
AND (d.Пункт_назначения = s.Пункт_назначения)
GROUP BY w.ID_номер_водителя, [Ф.И.О.]";
    else
```

```

        if (radioButtonDet_All.Checked)
            sqlSelect = @"SELECT w.ID_номер_водителя, [Ф.И.О.],
sum(d.[Цена поездки, руб]) AS Прибыль
FROM Водитель w, Заказ s, Маршрут d
WHERE (w.[Ф.И.О.] = '@last_name') AND
(w.ID_номер_водителя = s.ID_Номер_водителя)
AND (d.Пункт_отправления = s.Пункт_отправления)
AND (d.Пункт_назначения = s.Пункт_назначения)
GROUP BY w.ID_номер_водителя, [Ф.И.О.]";

if (checkBoxMore.Checked)
    sqlSelect += "HAVING Sum(d.[Цена поездки, руб]) >@amount";

if (checkBoxOrder.Checked)
    sqlSelect += " ORDER BY Sum(d.[Цена поездки, руб]) desc";
SqlConnection connection = new
SqlConnection(Properties.Settings.Default.taxiConnectionString);
SqlCommand command = connection.CreateCommand();
sqlSelect = sqlSelect.Replace("@last_name", textBoxWorker.Text);
command.CommandText = sqlSelect;
//command.Parameters.AddWithValue("@last_name", textBoxWorker.Text + "%");

if (checkBoxMore.Checked)
    try
    {
        command.Parameters.Add("@amount", SqlDbType.Money).Value =
        Double.Parse(textBoxMore.Text);
    }
    catch
    {
        MessageBox.Show("Прибыль в условии должна быть задана числом",
"ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        checkBoxMore.Checked = false;
        return;
    }
SqlDataAdapter adapter = new SqlDataAdapter();
adapter.SelectCommand = command;
DataTable table = new DataTable();
adapter.Fill(table);
dataGridViewFSelect.DataSource = table;
if (table.Rows.Count == 0) MessageBox.Show("Нет значений!",
"Информация", MessageBoxButtons.OK, MessageBoxIcon.Information);
}

```

Приложение Б

(обязательное)

Листинг кода для обработки коррелированного и не коррелированного подзапроса

```
private void buttonSubquery_Click(object sender, EventArgs e)
{
    if (String.IsNullOrEmpty(textBoxNumber.Text))
    {
        MessageBox.Show("Обязательно укажите номер необходимой продажи",
            "Внимание", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }

    string sqlSelect = "";
    if (radioButtonCorrelated.Checked)
        sqlSelect = @"select * from Заказ t1
        where @number in
        (SELECT t2.[Ф.И.О.]
        FROM Водитель t2
        WHERE t2.ID_номер_водителя = t1.ID_Номер_водителя)";

    else
        if (radioButtonNoCorrelated.Checked)
            sqlSelect = @"select
            (select t2.[Ф.И.О.] FROM Водитель t2
            WHERE t2.[Ф.И.О.] like (@number) as Фамилия, t1.*
            from Заказ t1
            where t1.ID_Номер_водителя in
            (select t2.ID_номер_водителя
            FROM Водитель t2
            WHERE t2.[Ф.И.О.] like (@number))";
    else
    {
        MessageBox.Show("Не выбрали вид подзапроса", "Ошибка",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }

    SqlConnection connection = new
    SqlConnection(Properties.Settings.Default.taxiConnectionString);
    SqlCommand command = connection.CreateCommand();
    command.CommandText = sqlSelect;

    try
    {
```

```

        if (radioButtonCorrelated.Checked)
        {
command.Parameters.Add("@number", SqlDbType.Char).Value= textBoxNumber.Text;
        }
else
        command.Parameters.Add("@number", SqlDbType.Char).Value = '%' +
textBoxNumber.Text + '%';
        }
        catch
        {
            MessageBox.Show("Номер продажи в условии должен быть задан числом",
"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }

        SqlDataAdapter adapter = new SqlDataAdapter();
        adapter.SelectCommand = command;
        DataTable table = new DataTable();
        adapter.Fill(table);
        dataGridViewSubquery.DataSource = table;

        if (table.Rows.Count == 0)
            MessageBox.Show("Нет значений!", "Информация", MessageBoxButtons.OK,
MessageBoxIcon.Information);
    }

```

Приложение В

(обязательное)

Листинг кода для формирования параметрического запроса на добавление

```
private void buttonSubquery_Click(object sender, EventArgs e)
{
    if (String.IsNullOrEmpty(textBoxNumber.Text))
    {
        MessageBox.Show("Обязательно укажите номер необходимой продажи",
            "Внимание", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }

    string sqlSelect = "";
    if (radioButtonCorrelated.Checked)
        sqlSelect = @"select * from Заказ t1
        where @number in
        (SELECT t2.[Ф.И.О.]
        FROM Водитель t2
        WHERE t2.ID_номер_водителя = t1.ID_Номер_водителя)";

    else
        if (radioButtonNoCorrelated.Checked)
            sqlSelect = @"select
            (select t2.[Ф.И.О.] FROM Водитель t2
            WHERE t2.[Ф.И.О.] like (@number) as Фамилия, t1.*
            from Заказ t1
            where t1.ID_Номер_водителя in
            (select t2.ID_номер_водителя
            FROM Водитель t2
            WHERE t2.[Ф.И.О.] like @number)";
        else
        {
            MessageBox.Show("Не выбрали вид подзапроса", "Ошибка",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }

    SqlConnection connection = new
    SqlConnection(Properties.Settings.Default.taxiConnectionString);
    SqlCommand command = connection.CreateCommand();
    command.CommandText = sqlSelect;

    try
    {
```

```

        if (radioButtonCorrelated.Checked)
        {
            command.Parameters.Add("@number", SqlDbType.Char).Value =
textBoxNumber.Text;
        } else
            command.Parameters.Add("@number", SqlDbType.Char).Value = '%' +
textBoxNumber.Text + '%';
    }
    catch
    {
        MessageBox.Show("Номер продажи в условии должен быть задан числом",
"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }

    SqlDataAdapter adapter = new SqlDataAdapter();
    adapter.SelectCommand = command;
    DataTable table = new DataTable();
    adapter.Fill(table);
    dataGridViewSubquery.DataSource = table;

    if (table.Rows.Count == 0)
        MessageBox.Show("Нет значений!", "Информация", MessageBoxButtons.OK,
MessageBoxIcon.Information);
    }

    string fileImage = "";
    private void buttonOpenPhoto_Click(object sender, EventArgs e)
    {
        openFileDialogDish.Title = "Укажите файл для фото";
        if (openFileDialogDish.ShowDialog() == DialogResult.OK)
        {
            fileImage = openFileDialogDish.FileName;
            try
            {
                pictureBoxPhoto_dish.Load(openFileDialogDish.FileName);
            }
            catch
            {
                MessageBox.Show("Выбран не тот формат файла", "Ошибка",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
        }
        else fileImage = "";
    }

    void InsertDish()
    {
        if ((String.IsNullOrEmpty(регистрационный_номерTextBox.Text) ||
(String.IsNullOrEmpty(iD номер водителяTextBox.Text))))

```

```

    {
        MessageBox.Show("Обязательно введите регистрационный номер и ID номер
водителя", "Внимание", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }

    if (String.IsNullOrEmpty(регистрационный_номерTextBox.Text))
    {
        MessageBox.Show("Некоректное значение номера!", "Внимание",
        MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }
    string reg_number = регистрационный_номерTextBox.Text;

    if (String.IsNullOrEmpty(iD_номер_водителяTextBox.Text))
    {
        MessageBox.Show("Некоректное значение id водителя!", "Внимание",
        MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }
    string id = iD_номер_водителяTextBox.Text;

    string sqlInsert = @"INSERT INTO
Транспортное_средство(Регистрационный_номер, ID_номер_водителя, Статус,
Класс_автомобиля, ПТС)
VALUES (@reg_number, @id, @statys, @class, @PhotoPTS)";

    SqlConnection connection = new
    SqlConnection(Properties.Settings.Default.taxiConnectionString);
    connection.Open();
    SqlCommand command = connection.CreateCommand();
    command.CommandText = sqlInsert;
    command.Parameters.AddWithValue("@id", id);
    command.Parameters.AddWithValue("@reg_number",
    регистрационный_номерTextBox.Text);
    //или другим способом, если необходимо явное указание типа данных
    //command.Parameters.Add("@Type", SqlDbType.NVarChar).Value =
textBoxType_dish.Text;
    //command.Parameters.AddWithValue("@Price", price);
    //command.Parameters.AddWithValue("@Weight", weight);

    if (!String.IsNullOrEmpty(статусComboBox.Text))
        command.Parameters.AddWithValue("@statys", статусComboBox.Text);
    else
        command.Parameters.AddWithValue("@statys", DBNull.Value);

    if (!String.IsNullOrEmpty(класс_автомобиляComboBox.Text))
        command.Parameters.AddWithValue("@class",
класс_автомобиляComboBox.Text);
    else

```

```

        command.Parameters.AddWithValue("@class", DBNull.Value);

        if (!String.IsNullOrEmpty(fileImage))
            command.Parameters.AddWithValue("@PhotoPTS",
File.ReadAllBytes(fileImage));
        else
        {
            command.Parameters.Add("@PhotoPTS", SqlDbType.VarBinary);
            command.Parameters["@PhotoPTS"].Value = DBNull.Value;
        }
        try
        {
            command.ExecuteNonQuery();
        }
        catch (Exception err)
        {
            MessageBox.Show("Ошибка выполнения запроса.\n" + err.Message,
"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }
        connection.Close();
        buttonSelectDishes_Click(this, EventArgs.Empty);
    }

```


Приложение Г

(обязательное)

Листинг кода для формирования параметрического запроса на изменение
представлен

```
void UpdateDish()
{
    if (String.IsNullOrEmpty(регистрационный_номерTextBox.Text))
    {
        MessageBox.Show("Обязательно введите регистрационный номер и ID номер  
водителя", "Внимание", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }

    if (String.IsNullOrEmpty(регистрационный_номерTextBox.Text))
    {
        MessageBox.Show("Некоректное значение номера!", "Внимание",  
MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }
    string reg_number = регистрационный_номерTextBox.Text;

    if (String.IsNullOrEmpty(iD_номер_водителяTextBox.Text))
    {
        MessageBox.Show("Некоректное значение id водителя!", "Внимание",  
MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }
    string id = iD_номер_водителяTextBox.Text;

    string sqlUpdate = "UPDATE Транспортное_средство SET {0} WHERE  
Регистрационный_номер=@reg_number";
    SqlConnection connection = new  
SqlConnection(Properties.Settings.Default.taxiConnectionString);
    connection.Open();
    SqlCommand command = connection.CreateCommand();

    string sqlValues = "";
    if (!String.IsNullOrEmpty(регистрационный_номерTextBox.Text))
        sqlValues += "Регистрационный_номер=@reg_number,";

    if (!String.IsNullOrEmpty(iD_номер_водителяTextBox.Text))
        sqlValues += "ID_номер_водителя=@id,";

    if (!String.IsNullOrEmpty(статусComboBox.Text))
        sqlValues += "Статус=@statys,";
```

```

        if (!String.IsNullOrEmpty(класс_автомобиляComboBox.Text))
            sqlValues += "Класс_автомобиля=@class,";

        if (!String.IsNullOrEmpty(fileImage))
            sqlValues += "ПТС=@PhotoPTS,";

        sqlValues = sqlValues.Substring(0, sqlValues.Length - 1);
        command.CommandText = String.Format(sqlUpdate, sqlValues);

        //или другим способом, если необходимо явное указание типа данных
        if (!String.IsNullOrEmpty(регистрационный_номерTextBox.Text))
            command.Parameters.Add("@reg_number", SqlDbType.NVarChar).Value =
регистрационный_номерTextBox.Text;

        if (!String.IsNullOrEmpty(iD_номер_водителяTextBox.Text))
            command.Parameters.AddWithValue("@id",
iD_номер_водителяTextBox.Text);

        if (!String.IsNullOrEmpty(статусComboBox.Text))
            command.Parameters.AddWithValue("@statys", статусComboBox.Text);

        if (!String.IsNullOrEmpty(класс_автомобиляComboBox.Text))
            command.Parameters.AddWithValue("@class",
класс_автомобиляComboBox.Text);

        if (!String.IsNullOrEmpty(fileImage))
            command.Parameters.AddWithValue("@PhotoPTS",
File.ReadAllBytes(fileImage));

        try
        {
            command.ExecuteNonQuery();
        }
        catch (Exception err)
        {
            MessageBox.Show("Ошибка выполнения запроса:\n" + err.Message,
"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        connection.Close();
        buttonSelectDishes_Click(this, EventArgs.Empty);
    }

```

Приложение Д

(обязательное)

Листинг кода для формирования параметрического запроса на удаление

```
void DeleteDish()
{
    if (String.IsNullOrEmpty(регистрационный_номерTextBox.Text))
    {
        MessageBox.Show("Обязательно введите регистрационный номер и ID номер водителя", "Внимание", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }

    if (String.IsNullOrEmpty(регистрационный_номерTextBox.Text))
    {
        MessageBox.Show("Некорректное значение номера!", "Внимание", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return;
    }
    string reg_number = регистрационный_номерTextBox.Text;

    string sqlDelete = @"DELETE FROM Транспортное_средство WHERE Регистрационный_номер=@reg_number";
    SqlConnection connection = new SqlConnection(Properties.Settings.Default.taxiConnectionString);
    connection.Open();
    SqlCommand command = connection.CreateCommand();
    command.CommandText = sqlDelete;
    command.Parameters.AddWithValue("@reg_number", reg_number);
    try
    {
        command.ExecuteNonQuery();
    }
    catch (Exception err)
    {
        MessageBox.Show(err.Message, "Ошибка удаления");
    }
    connection.Close();
    buttonSelectDishes_Click(this, EventArgs.Empty);
}

private void buttonExecutedML_Click(object sender, EventArgs e)
{
    if (radioButtonInsert_dish.Checked)
        InsertDish();
    else
        if (radioButtonUpdate_dish.Checked)
```

```
        UpdateDish();
    else
        if (radioButtonDelete_dish.Checked)
            DeleteDish();
        else
            MessageBox.Show("Вы не выбрали действие", "Внимание",
                MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }

    private void radioButtonDelete_dish_CheckedChanged(object sender, EventArgs e)
    {
        panelDish.Visible = !radioButtonDelete_dish.Checked;
    }
}
```

Приложение Е

(обязательное)

Код для формирования запросов

```
DataTable FillDataGridView(string sqlSelect)
{
    //Создаем объект connection класса SqlConnection для соединения с БД
    //CafeConnectionString – строка описания соединения с источником данных
    SqlConnection connection = new
        SqlConnection(Properties.Settings.Default.DeliveryConnectionString);

    //Создаем объект command для SQL команды
    SqlCommand command = connection.CreateCommand();

    //Заносим текст SQL запроса через параметр sqlSelect
    command.CommandText = sqlSelect;

    //Создаем объект adapter класса SqlDataAdapter
    SqlDataAdapter adapter = new SqlDataAdapter();

    //Задаем адаптеру нужную команду, в данном случае команду Select
    adapter.SelectCommand = command;

    //Создаем объект table для последующего отображения результата запроса
    DataTable table = new DataTable();

    //заполним набор данных результатом запроса
    adapter.Fill(table);

    return table;
}

private void OrderContentRadioButton_CheckedChanged(object sender, EventArgs e)
{
    dataGridViewSelect.DataSource = FillDataGridView("SELECT * FROM Водитель WHERE
    Водитель.Статус_активности = 'True'");
}

private void radioButtonOrder_CheckedChanged(object sender, EventArgs e)
{
    dataGridViewSelect.DataSource = FillDataGridView(@"SELECT ID_номер_заказа,
    Пункт_отправления, Пункт_назначения, ID_Номер_водителя, ID_номер_клиента, Статус FROM
    Заказ WHERE Заказ.Статус = 'Завершен'");
}

private void radioButtonClient_CheckedChanged(object sender, EventArgs e)
{
    dataGridViewSelect.DataSource = FillDataGridView(@"SELECT * FROM Клиент");
}
```

