МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ Федеральное государственное бюджетное образовательное учреждение высшего образования

«Чувашского государственного университета имени И. Н. Ульянова»

Факультет информатики и вычислительной техники

Кафедра вычислительной техники

Курсовая работа

по дисциплине «Базы данных» на тему:

## «Транспортная компания СДЭК»

Выполнил: студент группы ИВТ-41-20 Галкин Дмитрий Сергеевич Проверил:

доцент Ржавин В. В.

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_\_\_

Чебоксары, 2022

Заголовок

**Введение3**

**Описание предметной области4**

**ER-диаграмма концептуальная модель9**

**ER-диаграмма логическая модель11**

**Список всех возможных атрибутов БД12**

**Структура всех таблиц с обозначением имен и типов полей и их характеристик, а также при необходимости маски ввода, условия на значения и значения по умолчанию13**

**Перечень запросов к базе данных16**

**Название и содержание созданных отчетов20**

**Название и содержание созданных макросов25**

**Описание процедур для интеграции Access с компонентами Office27**

**Описание схемы хранилища данных OLAP32**

**Описание XML-документов для импорта и экспорта данных33**

**Выводы34**

**Список использованной литературы35**

**Приложение с текстами программ36**

# Введение

Microsoft Office Access – мощный редактор, который представляет собой систему управления базами данных, основанными на реляционной модели. Также необходимо отметить, что он использует динамическую модель обмена между сетевыми ресурсами и приложениями. При этом редактор использует продвинутые инструменты для обработки любого типа информации представление их в виде четкой последовательной структуры.

Редактор применяется для полной автоматизации работы с БД в разных сферах деятельности, бизнесе, управлении кадрами и т.д. Стоит заметить, что благодаря тому, что программное обеспечение имеет универсальную структуру.

Список основных компонентов программного обеспечения, который присутствует в Access:

* ***Таблица*** – запись и хранения данных.
* ***Запрос*** – получение информации из таблиц.
* ***Форма*** – объект используется для представления введенной информации в более удобном для пользователя виде.
* ***Отчет*** – позволяет получить конечный результат в виде готового документа.
* ***Макрос*** – элемент, содержащий в себе последовательное описание для выполнения того или иного действия, можно создать команду, которая будет выполнять определенную задачу.
* ***Модуль*** – компонент, который содержит в себе ПО, написанное на языке программирования Visual Basic. Используется для расширения функционала приложения.

Базы данных – совокупность сведений, относящихся к определенной теме или задаче, организованная таким образом, чтобы обеспечить удобное представление общую совокупность определенной темы.

В данной курсовой работе рассматривается: проектирование БД, в которое входят такие аспекты, как:

* Логическое проектирование
* Физическое проектирование
* Формирование запросов SQL
* Реализация форм для работы с данными в БД
* Создание итоговых документов/отчетов
* Формирование макросов

# Анализ предметной области

Предметной областью данного курсового проекта является база данных для логистической компании СДЭК.

# Словарь терминов

|  |  |
| --- | --- |
| **Термин, сокращение** | **Описание** |
| **Лицевой счет** | Это специальный счет для ведения учета расчетов с физическими и юридическими и юридическими лицами, на котором отражаются все финансово-кредитные операции с определенным клиентом. |
| **ИНН** | Документ, содержащий цифровой шифр, присваивающийся всем плательщикам налогов в нашей стране. Используется всеми государственными структурами. |
| **Физическое лицо** | Гражданин РФ, иностранного государства либо лицо без гражданства, наделённое правами и обязанностями в силу самого факта существования. |
| **Юридическое лицо** | Организация, которая имеет в собственности, хозяйственном ведении или оперативном управлении обособленное имущество, отвечает по своим обязательствам этим имуществом. |
| **Франчайзинг** | Вид отношений между рыночными субъектами, когда одна сторона передаёт другой стороне за плату право на определенный вид предпринимательства, используя разработанную бизнес-модель его ведения. |
| **Страховой полис** | Именной документ, подтверждающий заключение договора страхования, выдаваемый страховой компанией страхователю. |
| **СДЭК** | Служба доставки экспресс курьер. |
| **Наложенный платеж** | является возможностью заплатить за товар при получении в пунктах выдачи или у курьеров. При отправлении посылки наложенным платежом принимаются только наличные. |
| **Контакт-центр** | Клиентская служба компании, осуществляющая обслуживание пользователей по телекоммуникационным каналам связи. Контакты клиентской службы указаны на Сайте в разделе «Контакты». |
| **Перевозчик** | Лицо, которое на основании договора с Продавцом оказывает услуги по доставке Товара. Агрегатор не несет ответственности за действия Перевозчика, в том числе за нарушение Перевозчиком сроков доставки Товара. |

*Таб. 1*

# Общее описание предметной области

База данных создается для транспортной компании СДЭК. Проектируемая система зависит от вопросов учета контактной информации о сотрудниках, ввод, редактирование, просмотр информации о клиентах, поиск информации используя разные условия сортировки.

Компания является исключительно логистическим решением, и сама по себе не является транспортной. Это посредник, который забирает посылку от заказчика, выбирает удобный способ транспортировки по сроком и стоимости и потом уже передает груз основным транспортным компаниям для перевозки к получателю.

Перед отправкой посылки через компанию СДЭК, заказчику стоит ознакомиться со списком запрещенных товаров для пересылки.

* Токсичные, ядовитые вещества
* Окисляющие составы (перекись водорода)
* Любые драгоценности
* Продукцию из федерального списка экстремистских материалов
* Музыкальные инструменты, изготовленные более 50-ти лет назад
* Любые средства и предметы, способные повредить или испачкать контейнер, упакованный материал, другие отправления.
* Спиртосодержащие продукты (водка, пиво и т.д)
* Самодельные предметы (инструменты, запчасти и т.д) без лицензии
* Продукты питания с коротким сроком хранения
* Жидкость для электронных сигарет и любые другие жидкие субстанции
* Биологические материалы
* Взрывоопасные грузы
* Огнестрельное оружие
* Живые существа
* Денежные средства и ценные бумаги
* Наркотические и психотропные вещества

Так же заказчиком может являться физическое лицо и юридическое лицо. В зависимости от типа лица, будут предлагаться разные тарифы отправки.

Заказчик может отправить посылку как по России, так и международным способом, оплата осуществляется 3-мя способами:

* Наличными
* Оплата картой
* Наложенным платежом (стоимость будет чуть дороже, чем вышеперечисленными способами)

# Описание бизнес правил

|  |  |  |  |
| --- | --- | --- | --- |
| **Идентификатор** | **Бизнес-правило** | **Тип** | **Статическое или динамическое** |
| BR-1 | Пользователь вправе предъявить требования о возврате денежных средств: 1) Если заказ не был доставлен в сроки, предусмотренные заказом.  2) Направил продавцу уведомление об отказе от исполнения договора купли-продажи. | Факт | Статическое |
| BR-2 | Для отправки бандероли через СДЭК отправитель обязан указать в заявке следующие данные: ФИО, паспортные данные, адрес официальной регистрации (прописки), адрес фактического места жительства, номер телефона. | Ограничение | Динамическое |

*Таб. 2*

# Состав возможных пользователей

|  |  |  |
| --- | --- | --- |
| **Идентификатор** | **Пользователь** | **Полномочия** |
| АСТ-1 | Системный администратор | Добавление, обновление реестра запросов |
| АСТ-2 | Оператор контакт-центра | Добавление, удаление, изменение клиента;  Добавление, удаление, изменение компании транспортировки;  Оформление нового заказа, внесение изменений, связанных с заказом; |
| АСТ-3 | Начальник франшизы | Выгрузка сформированных отчетов по работе персонала и статистики заказов |

*Таб. 3*

# Перечень задач

Необходимые функции:

* Получить список доступных транспортных компаний (Оператор)
* Получить информацию о заказе (Оператор)
* Получить список клиентов (Оператор)
* Получить список операторов (Администратор)
* Добавить/изменить запрос и обновить реестр запросов (Администратор)

Желательные функции:

* Получение заказа по номеру заказа (Оператор)
* Формирование накладной (Начальник франшизы)

Возможные функции: -

Отсутствующие функции: -

# Описание входных документов

Любой документ, подтверждающий личность: все документы, которые имеют информацию о ФИО, адрес проживания, личные данные клиента.

Договора о сотрудничестве с транспортными компаниями.

Договора франшиз, заключаются с юридическими лицами, которые покупают франшизу у компании.

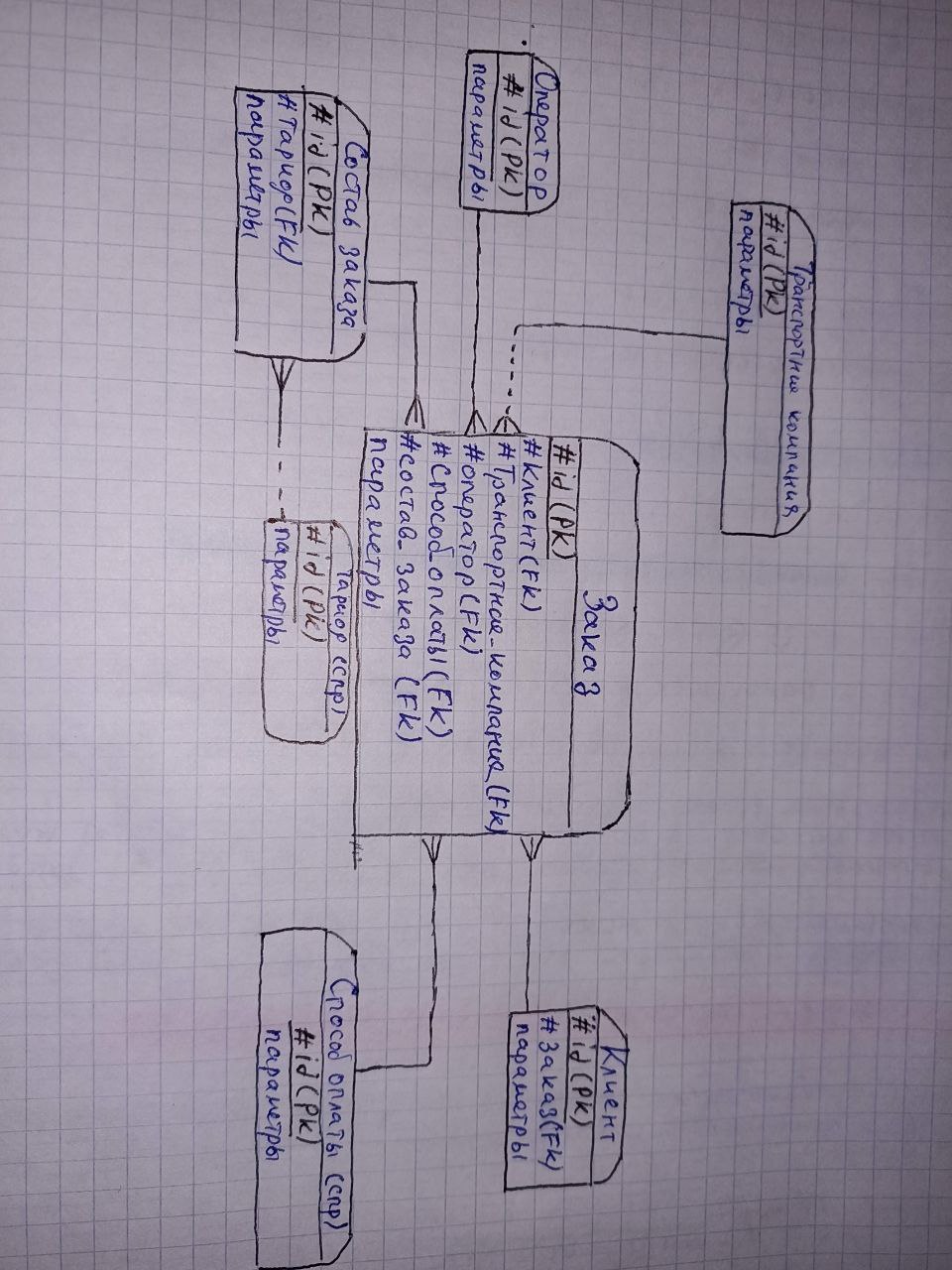
# Описание выходных документов

|  |  |  |
| --- | --- | --- |
| **№** | **Наименование отчета** | **Потребитель отчета** |
| **1** | Виды тарифов | Потребитель |
| **2** | Список всех заказов | Оператор |
| **3** | Список транспортных заказов | Оператор |
| **4** | Список операторов | Администратор |
| **5** | Список клиентов | Оператор |
| **6** | Накладная по заказам | Начальник франшизы |

*Таб. 4*

# Концептуальное проектирование

# Построение ER-Диаграммы



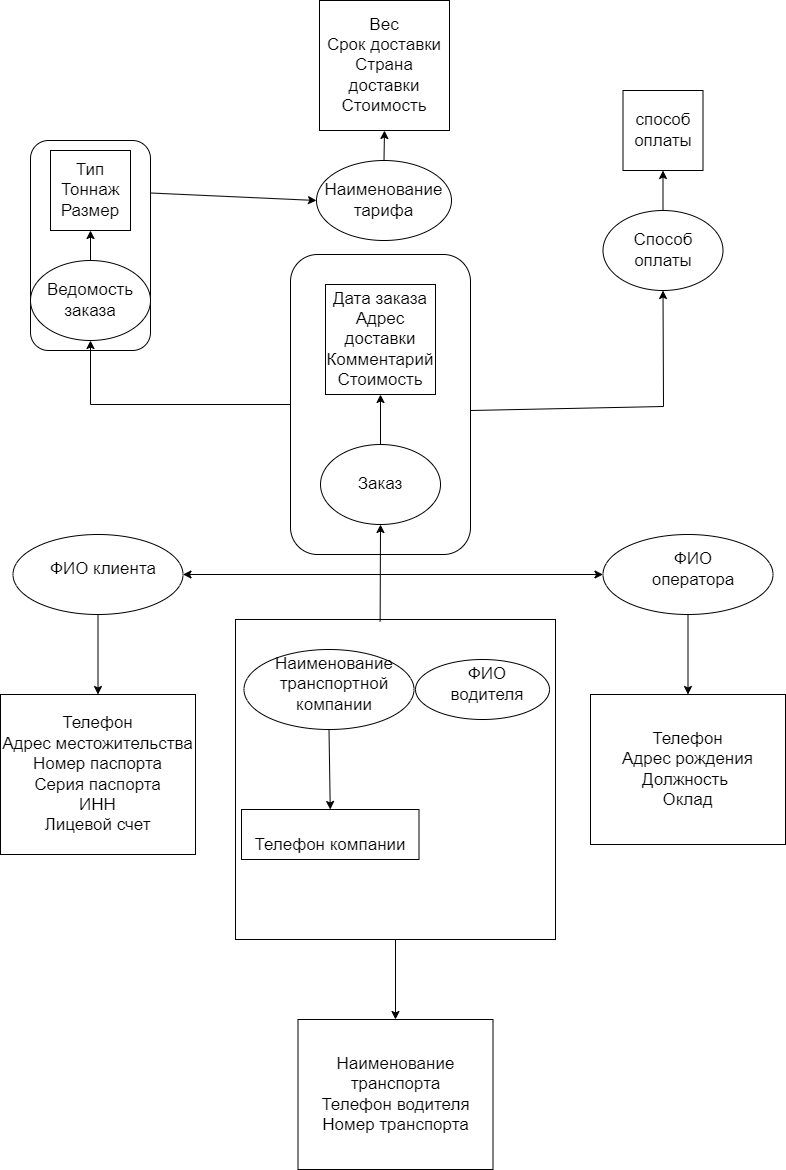
*Рис. 1*

# Получить набор – предварительных отношений, используя правила перехода к реляционной модели

|  |  |  |  |
| --- | --- | --- | --- |
| **Сущность 1** | **Описание связи** | **Кратность** | **Сущность 2** |
| Группа транспортных служб | Группа должно включать в себя много транспортных служб | N, M | Транспортная служба |
| Транспортная служба **входит** в состав **одной** группы | 0, 1 |
| Заказ | Заказ **должен** включать **одну** и **более** строк заказа | 1, М | Строка заказа |
| Строка заказа **должна** относиться ровно к **одному** заказу | 1,1 |
| Ведомость | Ведомость **может** содержать **много** строк | 0, М | Строка ведомости |
| Строка ведомости **должна** относиться ровно к **одной** ведомости | 1, 1 |

*Таб. 5*

**Диаграмма функциональных зависимостей**

****

**Пример ER-диаграммы, не приведенной ко 2-й нормальной форме (2НФ)**

Составным ключом будут являться – Адрес\_заказа, ФИО\_клиента. Т.к заказ принадлежит только одному клиенту.

Исходя из диаграммы функциональной зависимости видим, что:

Имя\_заказа Адрес\_дост

Имя\_заказа Стоимость

* Следовательно, атрибуты Адрес\_заказа, Стоимость зависят от части составного ключа
* В итоге 2 нормальная форма нарушена

**Пример ER-диаграммы, не приведенной к 3-й нормальной форме (3НФ)**

|  |
| --- |
| **Заказ** |
| #id\_заказа |
| Код\_клиента |
| ФИО\_клиента |

Первичным ключом будут являться – #id\_заказа

Исходя из диаграммы функциональной зависимости видим, что:

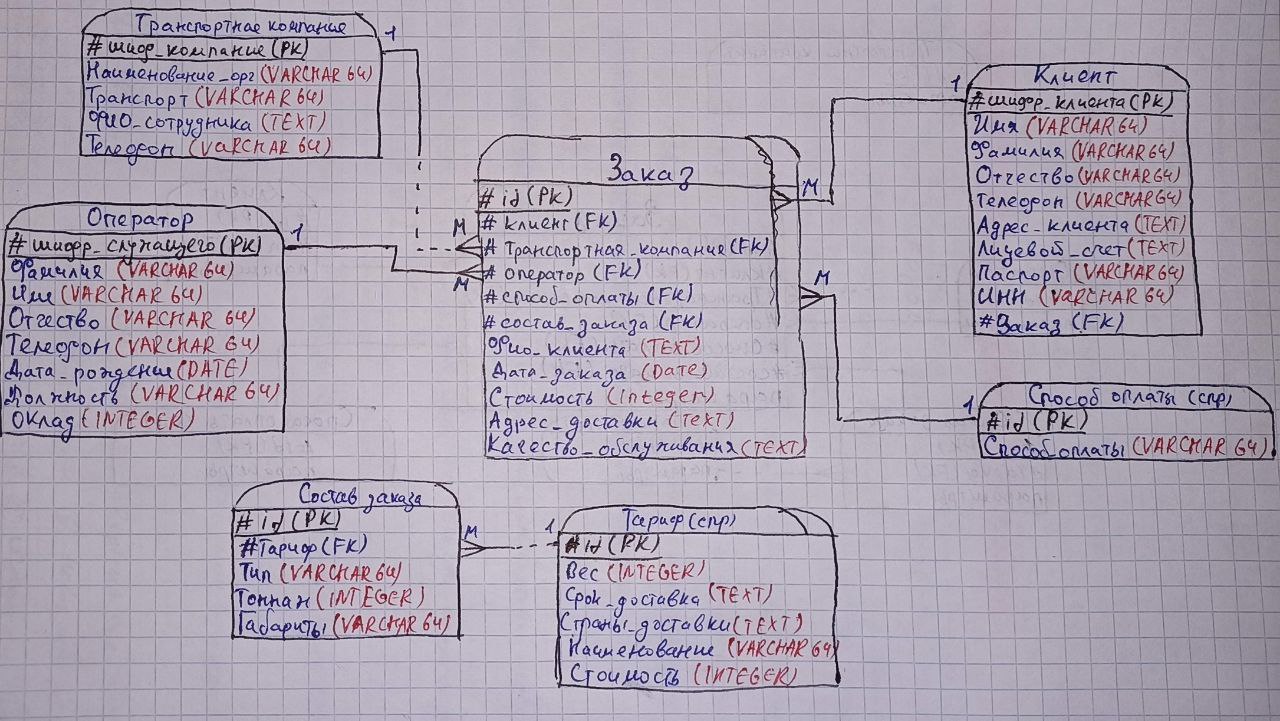
id\_заказа Адрес\_дост

Код\_клиента ФИО\_клиента

* Следовательно, атрибут ФИО\_клиента только транзитивно зависит от ключа
* В итоге 3 нормальная форма нарушена

# Логическое проектирование

# 3.1 Построение ER-Диаграммы



*Рис. 2*

# 3.2 Структура всех таблиц с обозначением имен и типов полей и их характеристик, а также при необходимости маски ввода, условия на значения и значения по умолчанию:

|  |  |  |
| --- | --- | --- |
| **№** | **Название таблицы** | **Название полей** |
| **1** | Клиент (Client) | *#Код\_клиента*, фамилия, имя, отчество, номер телефона, адрес местожительства, номер паспорта, серия паспорта, прописка, ИНН, лицевой счет |
| **2** | Оператор (Operator) | *#Код\_оператора*, фамилия, имя, отчество, номер телефона, дата рождения, должность, оклад, старший оператор |
| **3** | Транспортная компания (Transport Company) | #Код\_компании, наименование компании, наименование транспорта, ФИО водителя, телефон водителя, телефон компании, номер транспорта |
| **4** | Тарифы (Rate) - справочник | *#Код\_тарифа*, вес, срок доставки, страна доставки, стоимость, наименование тарифа |
| **5** | Способ оплаты (Payment Method) - справочник | *#Код\_оплаты*, способ оплаты |
| **6** | Ведомость заказа (Order list) | *#Код\_ведомости*, тип, тоннаж, размер, #id\_тариф |
| **7** | Заказ (Order) | *#Код\_заказа*, дата заказа, адрес доставки, комментарий, стоимость, #id\_способ\_оплаты, #id\_ведомость\_заказа, #id\_оператора, #id\_транпортной\_компании, заказ принят |

*Таб. 6 – Состав базы данных*

* **A1** – Тип данных
* **A2** – Размер поля
* **A3** – Формат вывода значений данного поля
* **A4** – Подпись поля в форме
* **A5** – Значение по умолчанию
* **A6** – Обязательное поля
* **A7** – Индексированное поле
* имяА – такое же описание, как и имя атрибута

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **Имя атрибута** | **A1** | **A2** | **A3** | **A4** | **A5** | **A6** | **A7** |
| 1 | *#Код\_клиента* | Счетчик | lg | “КЛ-0000” |  |  | Да |  |
| 2 | фамилия | Строка | 64 |  | имяА |  | Да |  |
| 3 | имя | Строка | 64 |  | имяА |  | Да |  |
| 4 | отчество | строка | 64 |  | имяА |  |  |  |
| 5 | номер телефона | строка | 64 | +7... | имяА |  | Да |  |
| 6 | адрес местожительства | Число | 255 |  | имяА |  | Да |  |
| 7 | номер паспорта | строка | 64 | “00000” | имяА |  | Да |  |
| 8 | серия паспорта | строка | 64 | 0000 | имяА |  | Да |  |
| 9 | прописка | строка | 255 |  | имяА |  | Да |  |
| 10 | ИНН | строка | 64 | “000000000000” | имяА |  |  |  |
| 11 | лицевой счет | строка | 64 | “000000000000” | имяА |  | Да |  |

*Таб. 7 – Описание характеристик атрибутов клиента*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **Имя атрибута** | **A1** | **A2** | **A3** | **A4** | **A5** | **A6** | **A7** |
| 1 | #Код\_оператора | счетчик | lg | “СР-0000” |  |  | Да |  |
| 2 | фамилия | строка | 64 |  | имяА | “Иванов” | Да |  |
| 3 | имя | строка | 64 |  | имяА | “Иван” | Да |  |
| 4 | отчество | строка | 64 |  | имяА | “Иванович” |  |  |
| 5 | номер телефона | строка | 64 | +7... | имяА |  | Да |  |
| 6 | дата рождения | дата/время |  | “00/00/0000;0;” | имяА |  | Да |  |
| 7 | должность | строка | 64 |  | имяА |  | Да |  |
| 8 | оклад | число |  |  | имяА | 0 | Да |  |
| 9 | старший оператор | число |  |  | имяА |  |  |  |

*Таб. 8 – Описание характеристик атрибутов оператора*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **Имя атрибута** | **A1** | **A2** | **A3** | **A4** | **A5** | **A6** | **A7** |
| 1 | *#Код\_компании* | счетчик | lg | “ТР-0000” |  |  | Да |  |
| 2 | Наименование компании | строка | 64 |  | имяА |  | Да |  |
| 3 | Наименование транспорта | строка |  |  | имяА |  | Да |  |
| 4 | ФИО водителя | строка | 255 |  | имяА |  | Да |  |
| 5 | телефон водителя | строка | 255 | +7... | имяА |  | Да |  |
| 6 | телефон компании | строка | 64 | “00\-00\-00” | имяА |  | Да |  |
| 7 | номер транспорта | строка | 64 | “\A000\BC\-00” | имяА |  | Да |  |

*Таб. 9 – Описание характеристик атрибутов транспортной компании*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **Имя атрибута** | **A1** | **A2** | **A3** | **A4** | **A5** | **A6** | **A7** |
| 1 | *#Код\_тарифа* | счетчик | lg | “Т-0000” |  |  | Да |  |
| 2 | вес | число |  | 00" кг" | имяА | 0 | Да |  |
| 3 | срок доставки | строка | 64 | "от "0" д" | имяА |  | Да |  |
| 4 | страна доставки | строка | 50 |  | имяА |  | Да |  |
| 5 | стоимость | число |  |  | имяА |  | Да |  |
| 6 | наименование тарифа | строка | 64 |  | имяА |  | Да |  |

*Таб. 10 – Описание характеристик атрибутов тарифов*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **Имя атрибута** | **A1** | **A2** | **A3** | **A4** | **A5** | **A6** | **A7** |
| 1 | *#Код\_оплаты* | счетчик | lg | “Р-0000” |  |  | Да |  |
| 2 | способ оплаты | строка | 64 |  | имяА |  |  |  |

*Таб. 11 – Описание характеристик атрибутов способа оплаты*

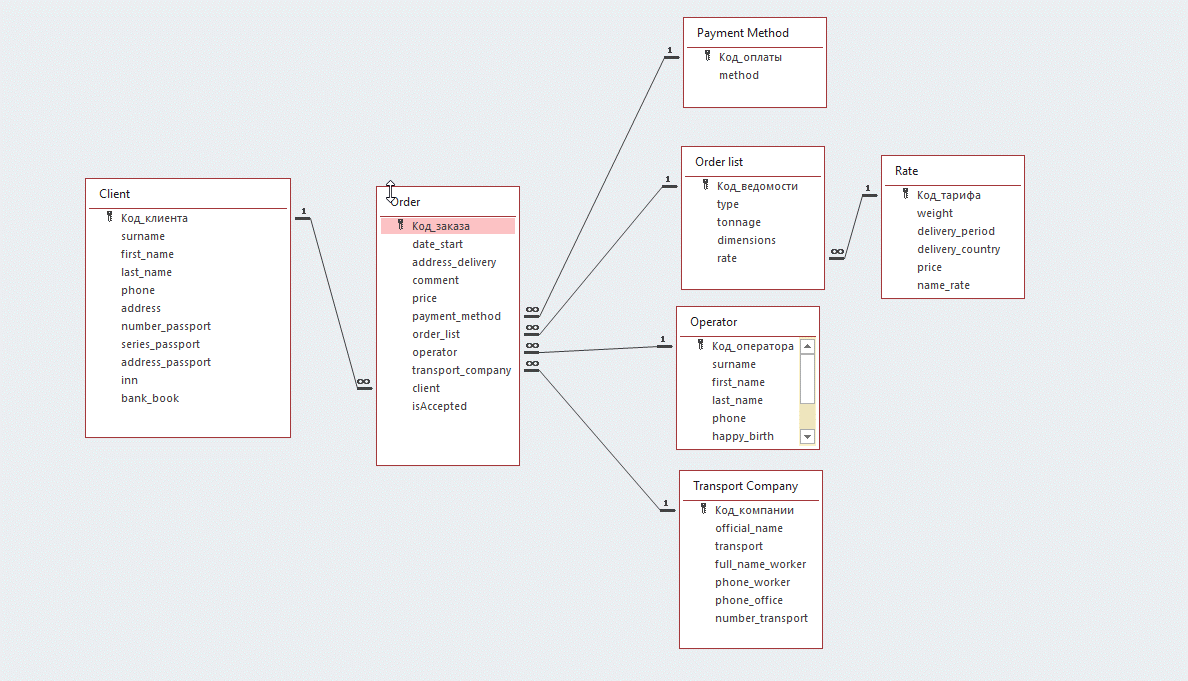
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **Имя атрибута** | **A1** | **A2** | **A3** | **A4** | **A5** | **A6** | **A7** |
| 1 | *#Код\_ведомости* | счетчик | lg | “К-0000” |  |  | Да |  |
| 2 | тип | строка | 64 |  | имяА |  | Да |  |
| 3 | тоннаж | число |  |  | имяА |  | Да |  |
| 4 | размер | строка | 64 | 00\-00\-00" см" | имяА |  | Да |  |
| 5 | #id\_тарифа | число |  |  | имяА |  | Да |  |

*Таб. 12 – Описание характеристик атрибутов ведомости заказа*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **Имя атрибута** | **A1** | **A2** | **A3** | **A4** | **A5** | **A6** | **A7** |
| 1 | *#Код\_заказа* | счетчик | lg |  |  |  | Да |  |
| 2 | дата заказа | Дата/время |  |  | имяА |  | Да |  |
| 3 | адрес доставки | Строка |  |  | имяА |  | Да |  |
| 4 | комментарий | Строка |  |  | имяА |  |  |  |
| 5 | стоимость | число |  |  | имяА | 0 | Да |  |
| 6 | #id\_способ\_оплаты | число |  |  |  |  | Да |  |
| 7 | #id\_ведомость\_заказа | число |  |  |  |  | Да |  |
| 8 | #id\_оператора | число |  |  |  |  | Да |  |
| 9 | #id\_клиента | число |  |  |  |  | Да |  |
| 10 | #id\_транспортной\_компании | число |  |  |  |  | Да |  |
| 11 | заказ принят | bool |  |  | имяА |  |  |  |

*Таб. 13 – Описание характеристик атрибутов заказа*

# Схема данных в Access



*Рис. 3*

# 4. Разработка транзакций (создание запросов)

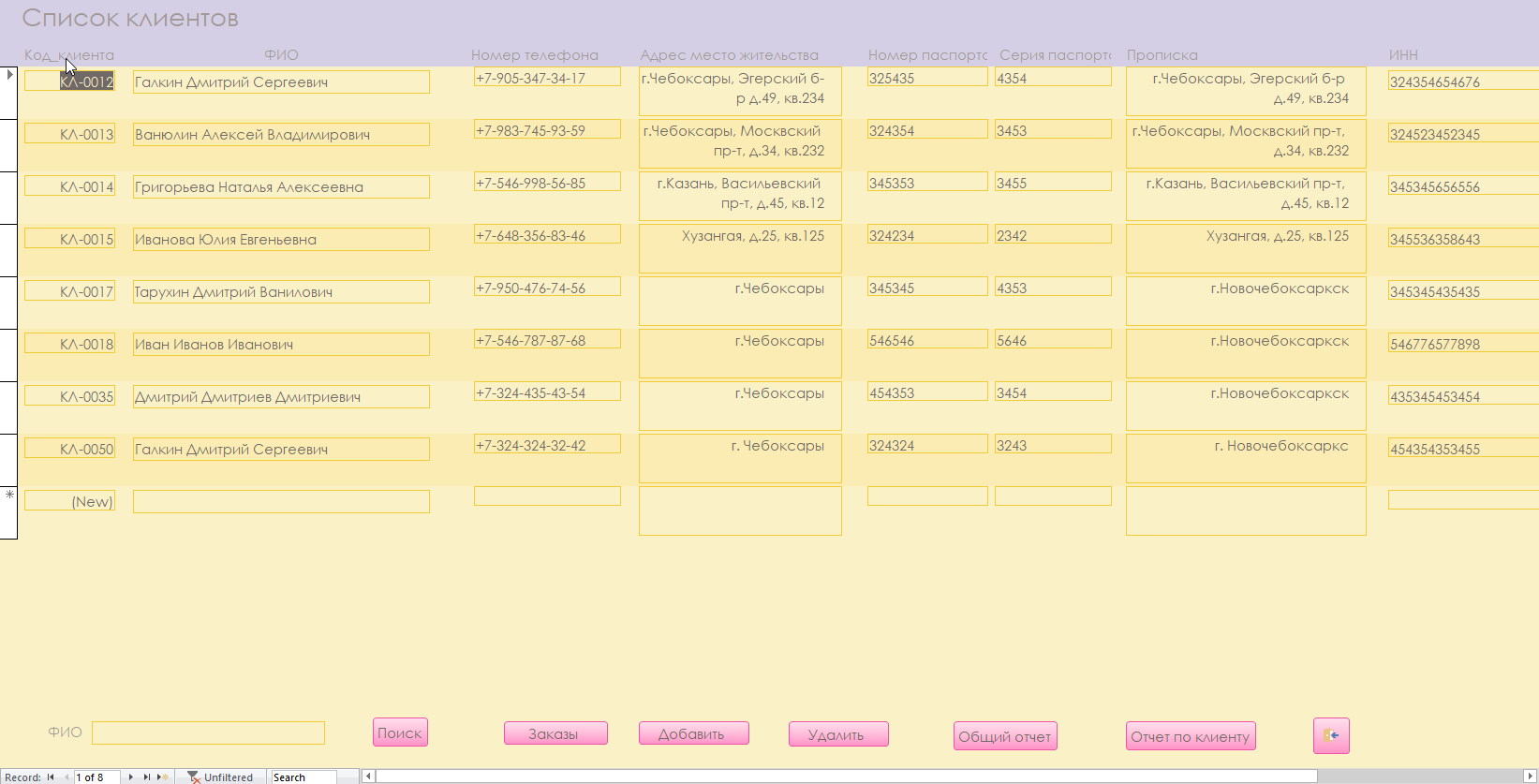
|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Имя запроса** | **Текст запроса** | **Запрос на SQL** |
| 1 | т\_ОбновРеестрЗапр | Обновляет реестр запросов | INSERT INTO т\_реестр\_запросов ( ID, QueryName, QueryTableDate )  SELECT Id, Name, DateCreate  FROM т\_ВсеЗапросы  WHERE (NOT EXISTS (SELECT ID FROM  т\_реестр\_запросов WHERE  т\_реестр\_запросов.ID = т\_ВсеЗапросы.Id)); |
| 2 | FindName | Поиск ФИО по списку клиентов | SELECT \*  FROM [Order]  WHERE [Формы]![Список клиентов]![FullName]=first\_name + ' ' + surname + ' ' + last\_name; |
| 3 | Заказы, которые были приняты после вводимой даты |  | SELECT \*  FROM [Order]  WHERE [Текущая дата] >= date\_start; |
| 4 | Заказы, которые еще не были доставлены |  | SELECT \*  FROM [Order]  WHERE isAccepted = false; |
| 6 | ЗапросыПользователей | Выводит все запросы, которые доступны для пользователей | SELECT QueryName  FROM т\_реестр\_запросов  WHERE QueryExpose=True; |
| 7 | Общая информация о клиентах | Выводит всю информацию о клиентах | SELECT Код\_клиента, surname, first\_name, last\_name, phone, address  FROM Client; |
| 8 | Общая информация о транспортных компаниях | Выводит всю информацию о транспортных компаниях | SELECT Код\_компании, official\_name, transport, full\_name\_worker, phone\_worker, phone\_office  FROM [Transport Company]; |
| 9 | Общая информация об операторах | Выводит всю информацию об операторах | SELECT Код\_оператора, surname, first\_name, last\_name, phone, happy\_birth, job\_title  FROM Operator; |
| 10 | Общая стоимость всех заказов |  | SELECT SUM(price) AS ["Общая стоимость"]  FROM [Order]; |
| 11 | Список главных операторов и их подчиненные | Выводит список главных операторов и их подчиненных | SELECT o.[surname] + ' ' + o.[first\_name] + ' ' + o.[last\_name] AS operator, e.[surname] + ' ' + e.[first\_name] + ' ' + e.[last\_name] AS senjor\_operator\_name  FROM Operator AS o INNER JOIN Operator AS e ON o.[Код\_оператора] = e.[senjor\_operator]; |
| 12 | т\_ВсеЗапросы | Выводит все запросы | SELECT Id, Name, DateCreate  FROM MSysObjects  WHERE (((Left([Name],4))<>"~sq\_") AND ((MSysObjects.Type)=5)); |

*Таб. 14*

# Разработка форм

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Название формы** | **Тип формы** | **Назначение** | **Примечание** |
| 1 | Заказы | *Форма для ввода данных/использование ЭУ календарь* | Показывают всю информацию о заказах, с возможностью добавление новой записи заказа | - Кнопка предыдущей записи  - Кнопка следующей записи  - Кнопка добавления записи  - Кнопка выхода из формы |
| 2 | Заказы клиента | *Иерархическая* | Вывод всех заказов, которые относятся к выбранному клиенту | -Дополнительная форма с информацией по заказу |
| 3 | Заказы транспортной компании | *Простая* | Вывод всех заказов для транспортной компании | - Информация выводится в табличной форме |
| 4 | Информация о заказе для клиента | *Простая* | Вывод всех заказов для клиента | - Информация выводится в табличной форме |
| 6 | Клиенты | *Простая/вкладки* | Удобный просмотр информации о клиентах, где информация выводится по вкладкам | - Кнопка предыдущей записи  - Кнопка следующей записи |
| 7 | Транспортная компания | *Иерархическая* | Вывод всех заказов, которые относятся к выбранной транспортной компании | -Дополнительная форма с информацией по заказу |
| 8 | Поиск по номеру заказа | *Форма на основе запроса с параметром/связанная* | Выводит информацию по номеру заказа | - Кнопка поиска  - Текстовое поле с номером заказа |
| 9 | Общая выплата за заказы | *Нахождение итоговой суммы* | Выводит весь список заказов, а также общую стоимость всех заказов | - В футоре прописано итоговое поле |

*Таб. 15*

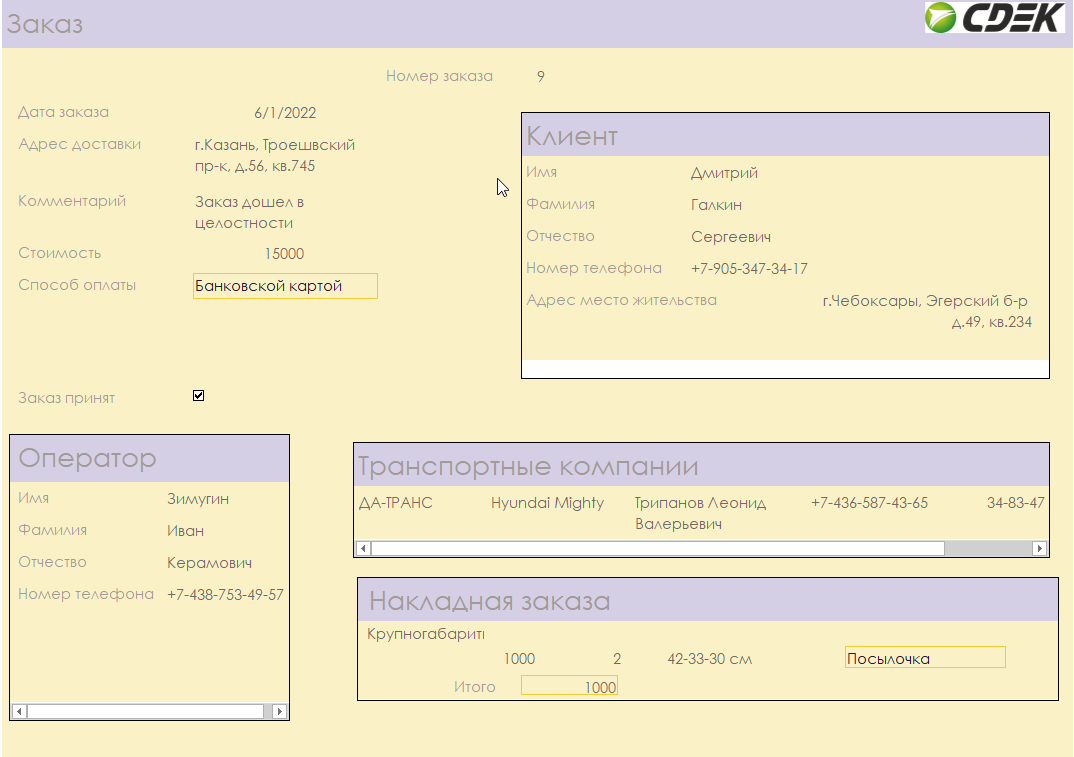


*Рис. 4*

# Разработка отчетов

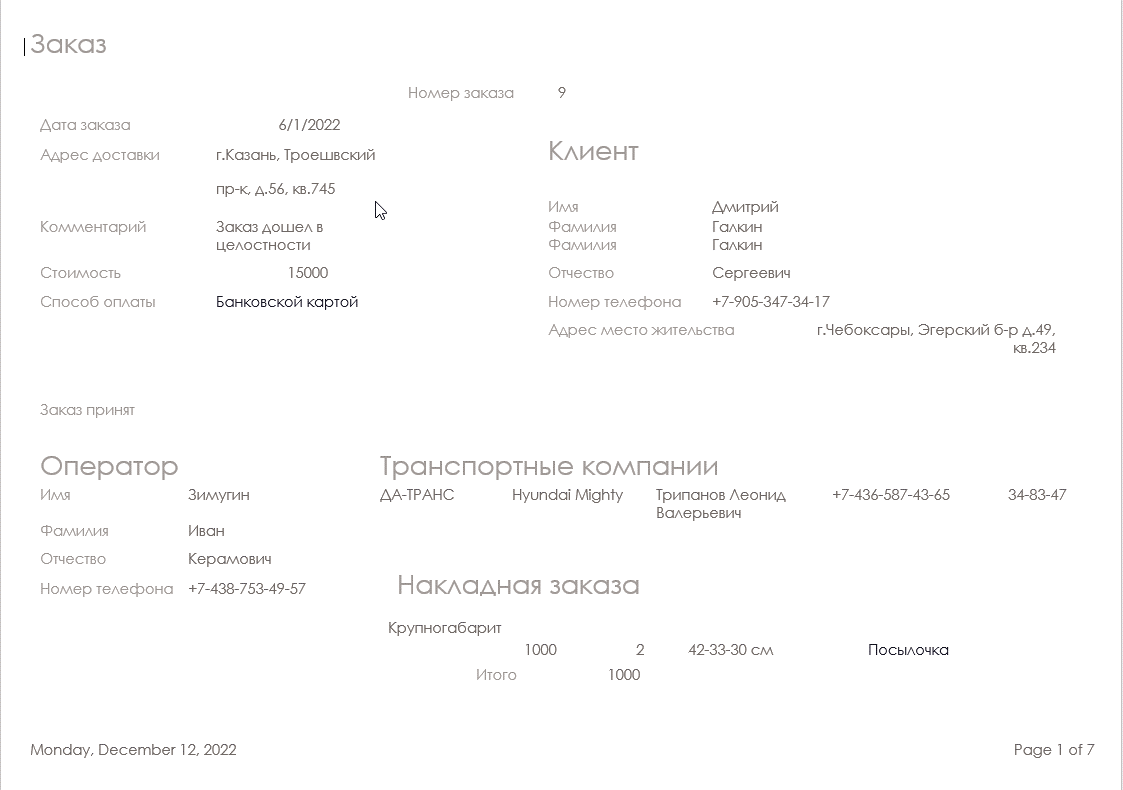
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Название отчета** | **Тип отчета** | **Назначение** | **Примечание** |
| 1 | Клиенты | *Простой* | Вывод информации о клиентах |  |
| 2 | Операторы | *Простой* | Вывод информации об операторах |  |
| 3 | Транспортные компании | *Простой* | Вывод информации о транспортных компаниях |  |
| 4 | Накладная заказа | *Простой*  с вычисляемым полем | Накладная | Суммируется сумма всех заказов |
| 6 | Информация о заказе по номеру заказа | Связь | Вывод информации о заказе, по номеру заказа | Для отчета заказ, связь по номеру заказа |
| 7 | Заказ | Составной отчет с подчинёнными | Вывод всей главной информации по заказу |  |
| 8 | Виды тарифов | Группировка/сортировка | Вывод значений с группированных по коду заказа сортировка по типу |  |

*Таб. 16*



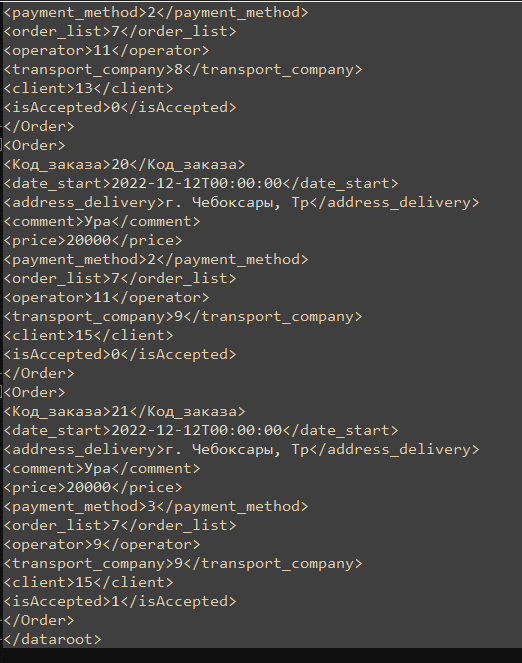
*Рис. 5*

**Отчет в формате Word**



*Рис. 6*

**Отчет в формате XML**



*Рис. 7*

# Разработка интерфейса приложения

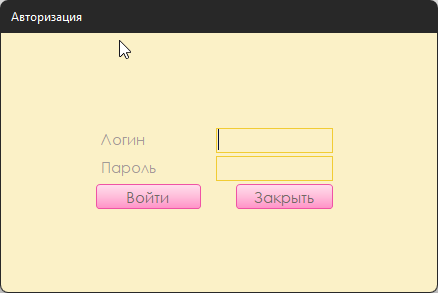
Разработан интерфейс приложения Access, который состоит из формы аутентификации и форм пользователя и администрации.  
Для того, чтобы войти под администратором нужно ввести:

* Login – admin
* Password – admin

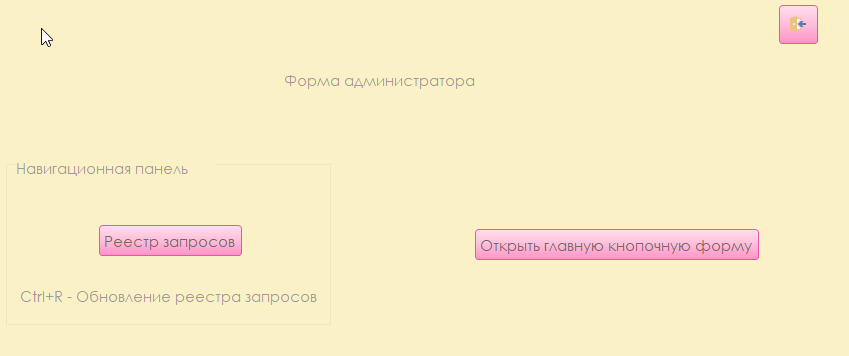
Для того, чтобы войти под пользователем нужно ввести:

* Login – user
* Password – user

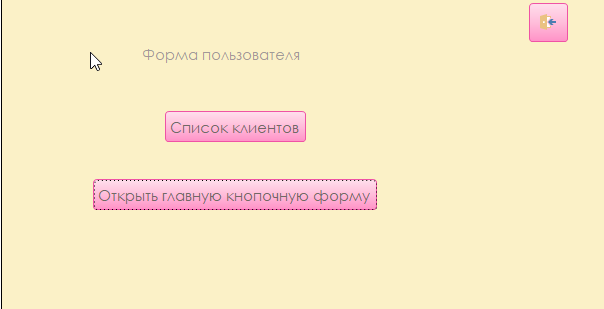
**Форма аутентификации**



**Форма администратора**



**Форма пользователя**



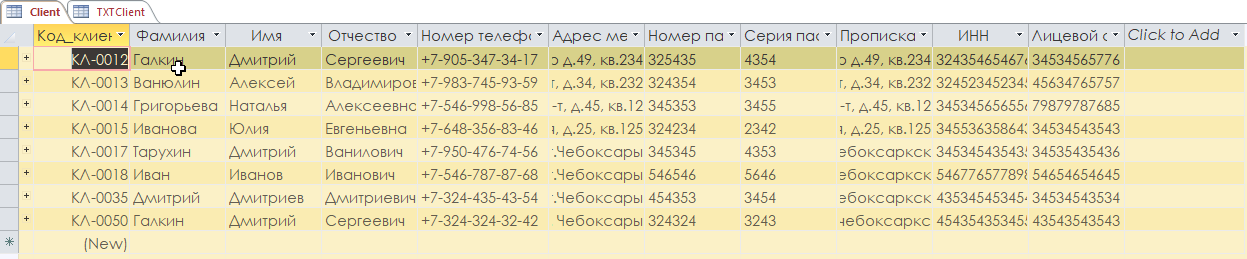
# Разработка макросов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Название макроса** | **Тип макроса** | **Назначение** | **Примечание** |
| 1 | Все запросы только для админа | *Изолированный макрос для формы* | Открывает форму со всеми запросами только для админа |  |
| 2 | Список главных операторов и их подчиненных | *Изолированный макрос для запроса* | Запускает запрос для вывода главных операторов и их подчиненных |  |
| 3 | Заказы | *Изолированный макрос для отчета* | Формирует отчет по всем заказам |  |
| 4 | Открытие всех таблиц | *Изолированный макрос для таблиц* | Открывает все таблицы |  |
| 5 | AutoKeys | *Макрос горячих клавиш / вложенный макрос* | Ctrl + R – открывает форму “все запросы, только для администратора  Ctrl + O – Открывает форму Реестр запросов  Ctrl + U – обновляет таблицу реестра запросов | При определенной комбинации клавиш выполняет определенное действие |
| 6 | Autoexec | *Автозапуск формы при открытии Access* | Запускает форму сразу как откроется Access |  |
| 7 | Главное меню | *Контекстное меню для главной формы* | Вывод своего контекстного меню на главной форме | При нажатии на правую кнопку мыши, реализует список команд, которые были добавлены. |
| 8 | Контекстное | *Макрос с вложенными запросами, которые будут в контекстном меню* | Реализует списки контекстного меню | При нажатии на правую кнопку мыши, реализует список команд, которые были добавлены. |
| 9 | FindFullName | *Макросы “Если”, “Применить фильтр”, “Открытие информационного сообщения”, “Поиск записи”* | Поиск ФИО по форме | - Если поле ввода пустое, то поиска по ФИО не будет  - Если ФИО нет в форме, то выведет, что такой записи нет  - Если ФИО есть, то выведет все записи с таким значением |
| 10 | ОткрытьОтчет | *Внедренные макросы* | При выборе определенного клиента и нажатию на кнопку выводит отчет по заказу клиента |  |
| 11 | Изменение поля во время изменения таблицы | *Макросы данных* | При добавления нового заказа автоматически подставляется дата |  |

*Таб. 17*

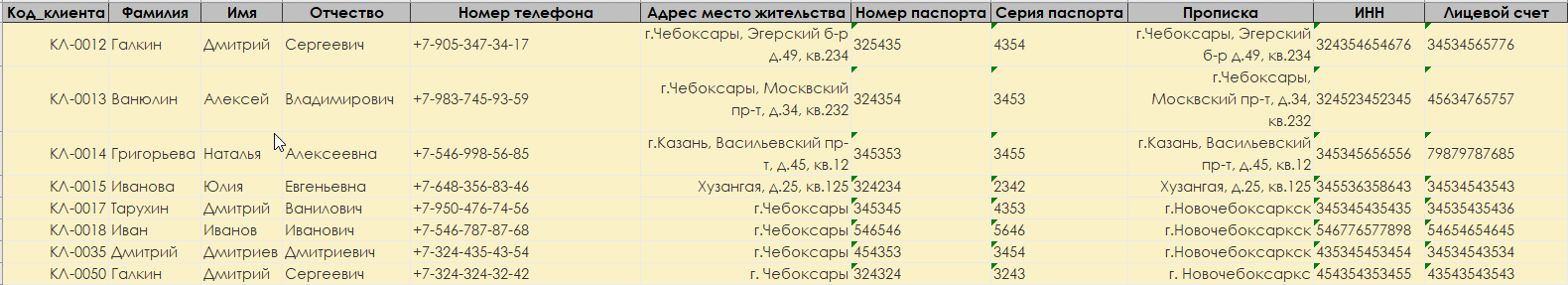
# Использование внешних данных

Исходная таблица клиентов:

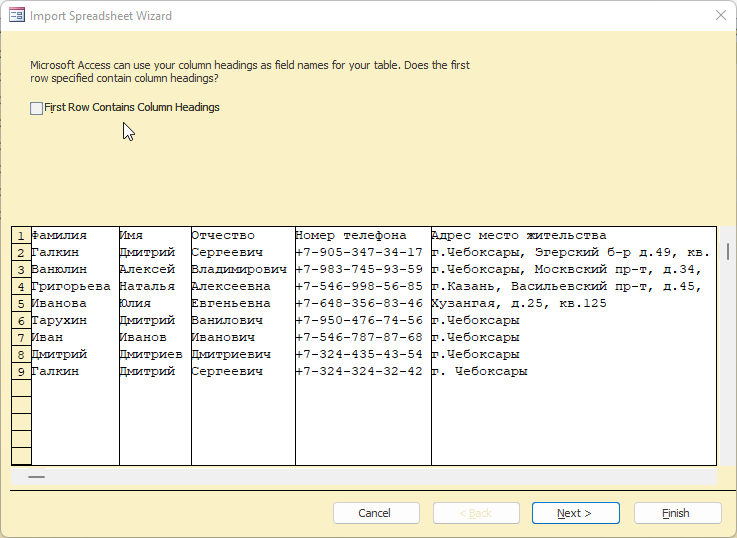


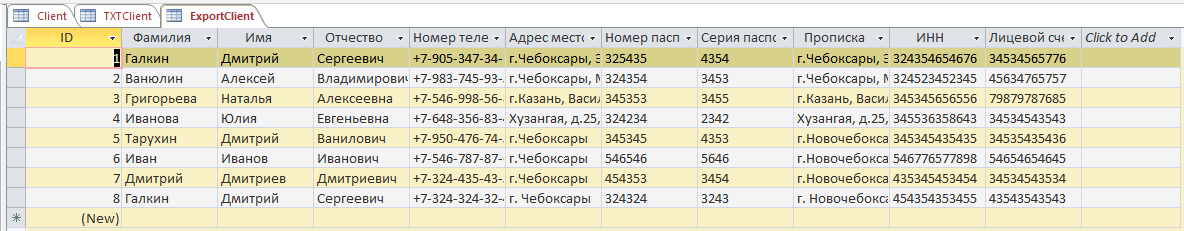
# Импорт и присоединение электронных таблиц

Экспорт данных в Excel:



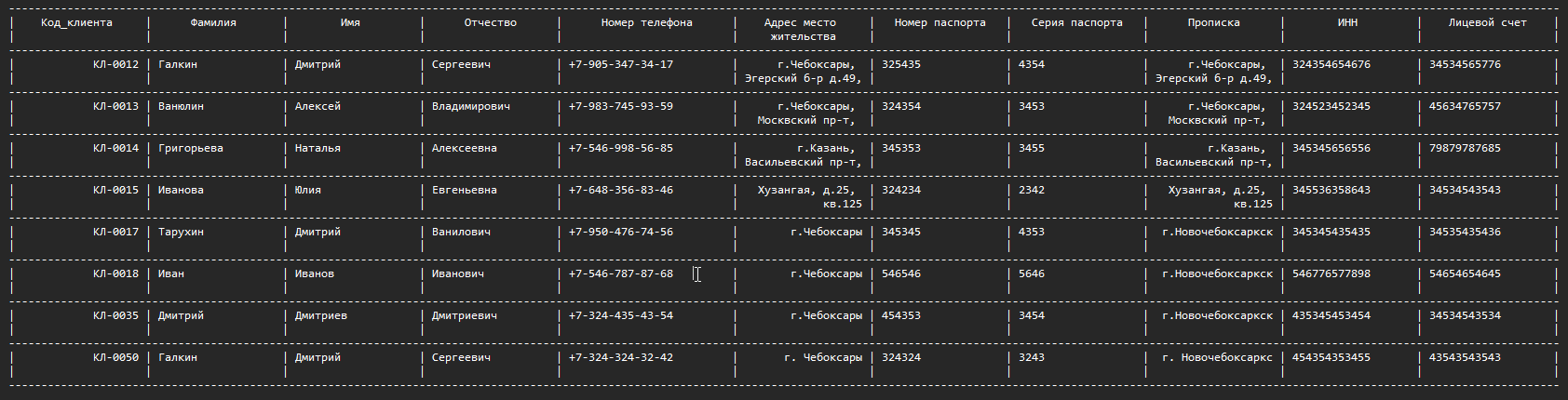
Импорт данных из Excel (добавим новую таблицу):



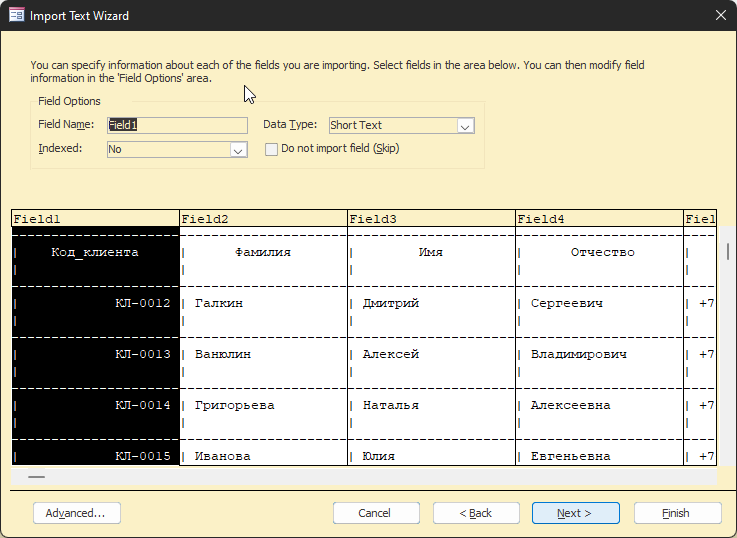


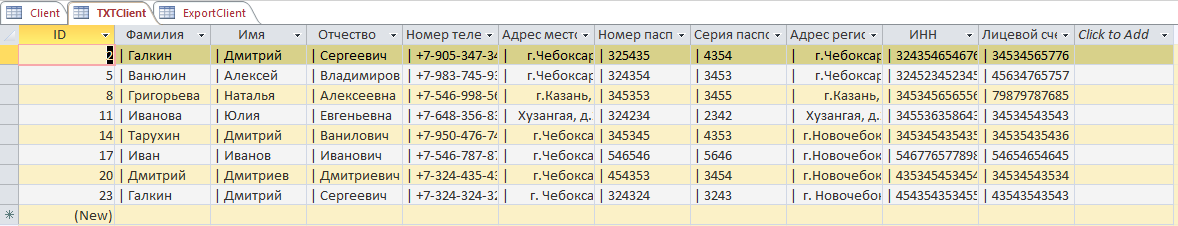
# Импорт и присоединение таблиц из текстовых файлов

Экспорт данных в текстовый файл:



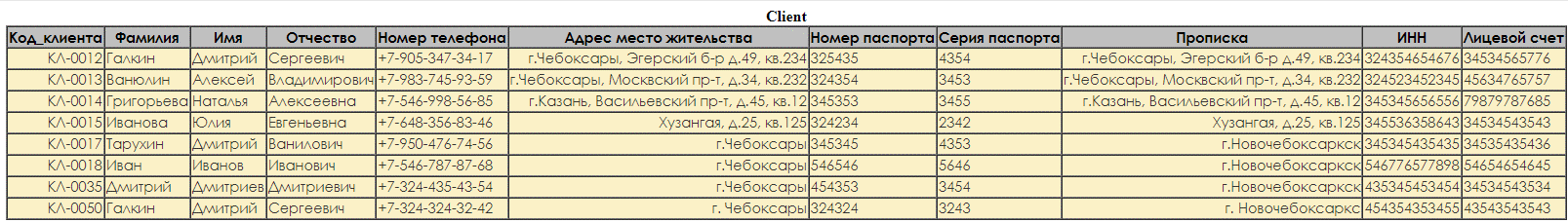
Экспорт данных из текстового файла (создадим новую таблицу):



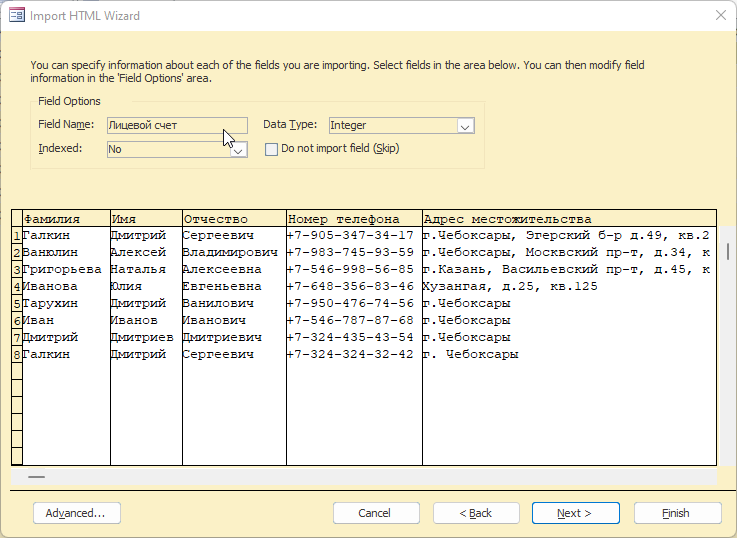


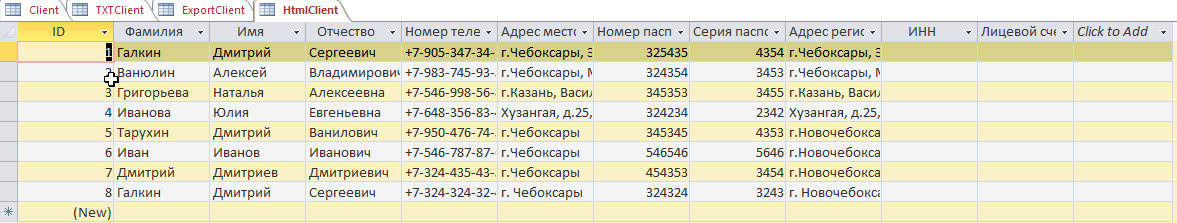
# Импорт и присоединение документов и схем HTML и данных с Web-узлов

Экспорт данных в html документ:



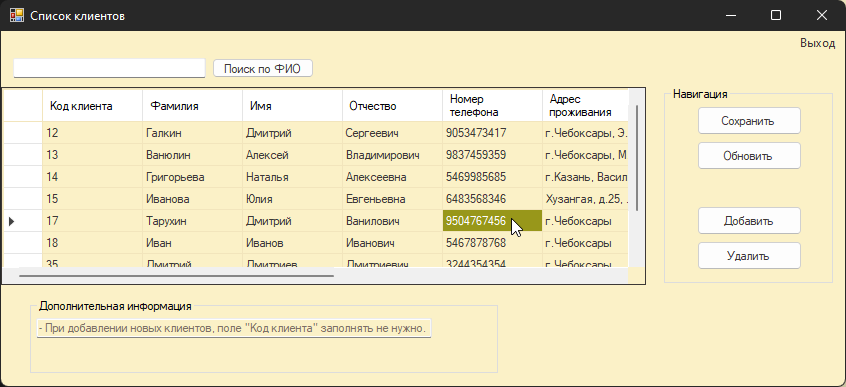
Импорт данных из html документа (создадим новую таблицу):





# Доступ к данным

Разработал Windows Form приложение, используя ADO.NET для возможности получения доступа к данным посредствам высокоуровневого языка программирования.



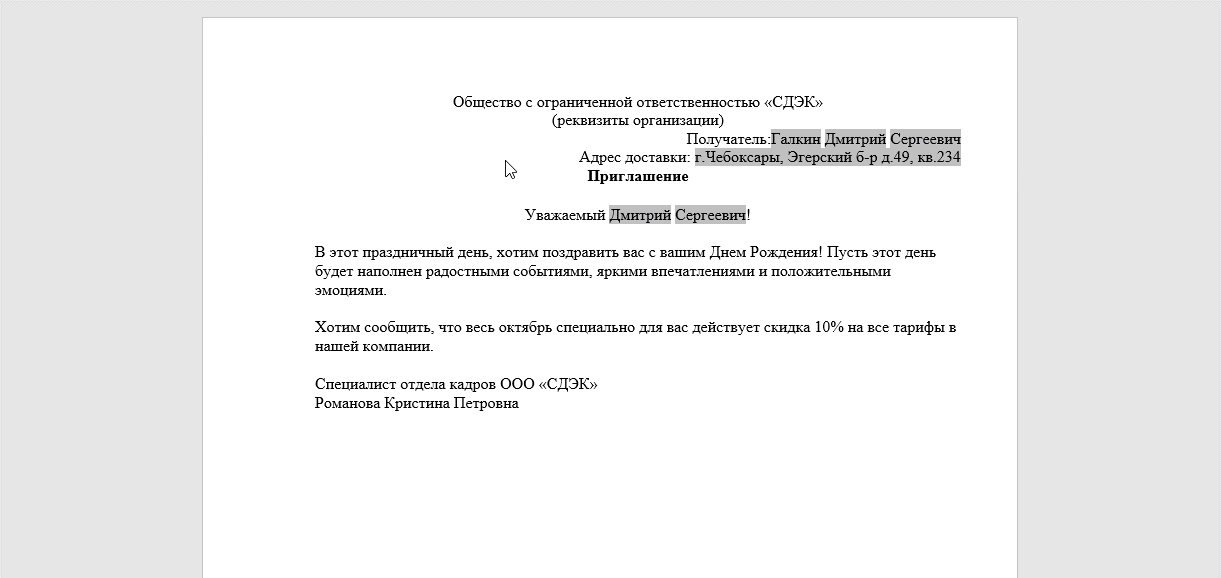
Реализация программной логики:

* Кнопка «Обновить» - обращается к базе данных Access и обновляет все записи формы клиентов.
* Кнопка «Сохранить» - при редактировании данных в форме клиентов, пользователь может нажать упомянутую кнопку для изменения данных в базе данных Access.
* Кнопка «Добавить» - при добавлении нового клиента(ов) в форму клиентов, пользователь может нажать упомянутую кнопку и добавить новые записи в базу данных Access.
* Кнопка «Удалить» - для удаления клиенты из базы данных Access, пользователь может выбрать нужные строки и нажать на упомянутую кнопку.
* Присутствует алгоритм фильтрации/выборки данных из формы клиентов. Пользователь может написать ФИО клиента(ов), которые ему нужные и нажать кнопку «Поиск по ФИО». Результатом будут все строки, с подходящими значениями.

# Интеграция Access с компонентами Office

Свяжем Word документ с поздравлением клиента со скидкой и таблицу Клиентов Access.



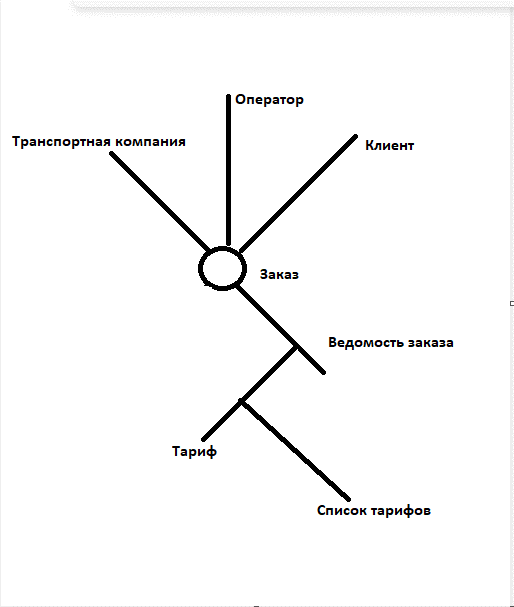


# Анализ многомерных данных OLAP

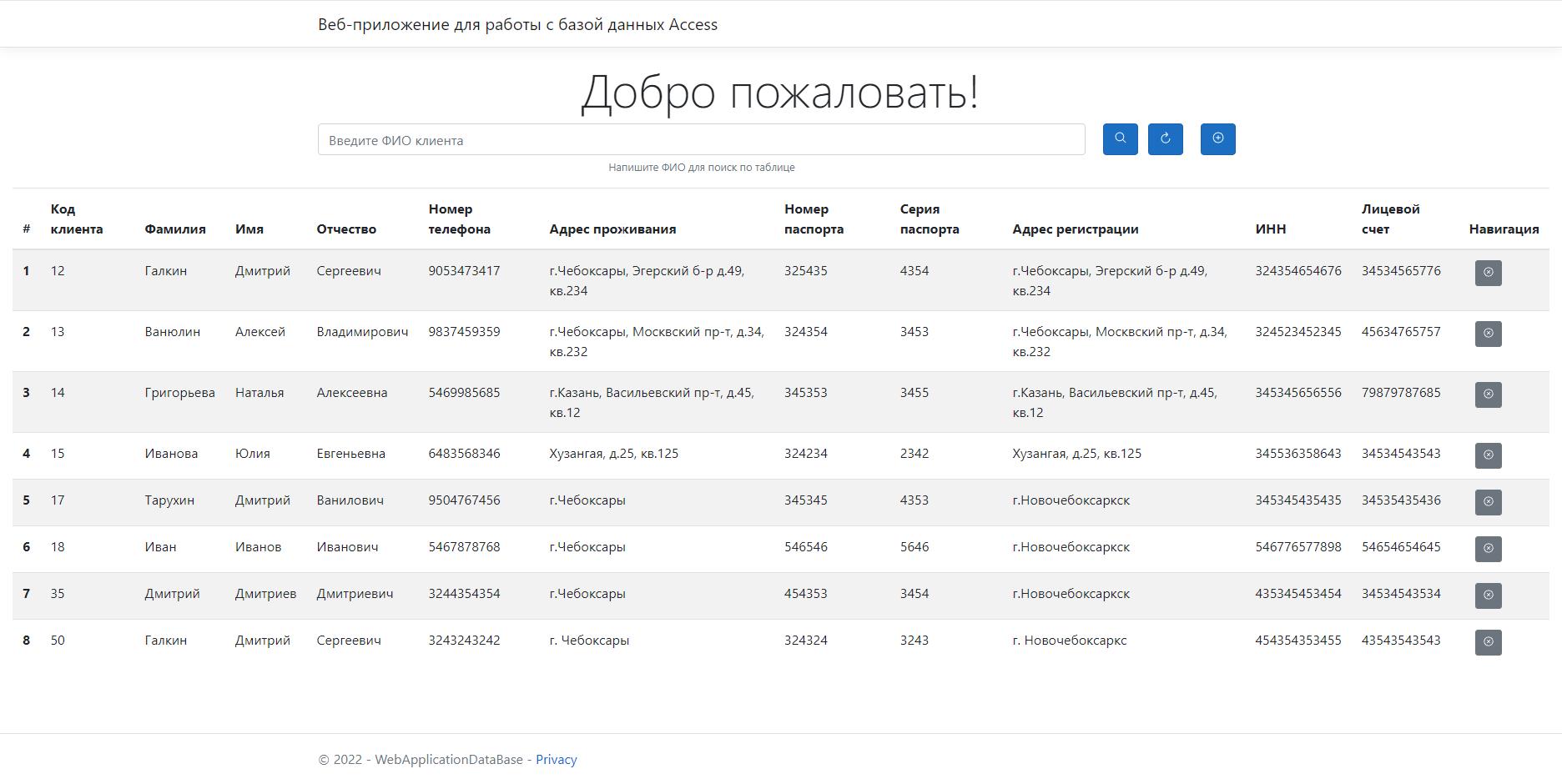
Информационные системы серьезного предприятия, как правило, содержат приложения, предназначенные для комплексного анализа данных, их динамики, тенденций.

Системы поддержки принятия решений обычно обладают средствами предоставления пользователю агрегатных данных для различных выборок из исходного набора в удобном для восприятия и анализа данных виде (таблицы, диаграммы). Традиционный подход сегментирования исходных данных использует выделение из исходных данных одного или нескольких многомерных наборов данных (нередко называемый гиперкубом или метокубом), оси которых содержат атрибуты, а ячейки – агрегируемые количественные данные.

Для предоставления многомерных данных я выберу схему снежинка:



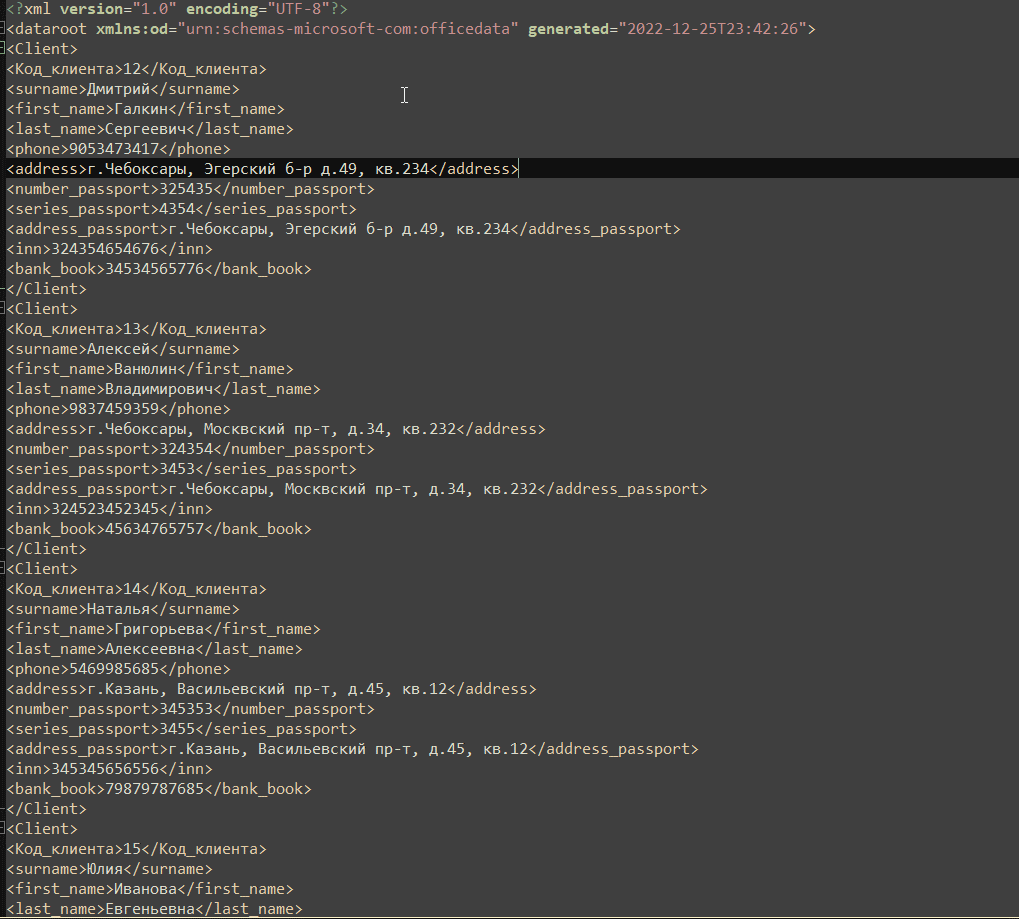
# Приложения для WEB-страниц



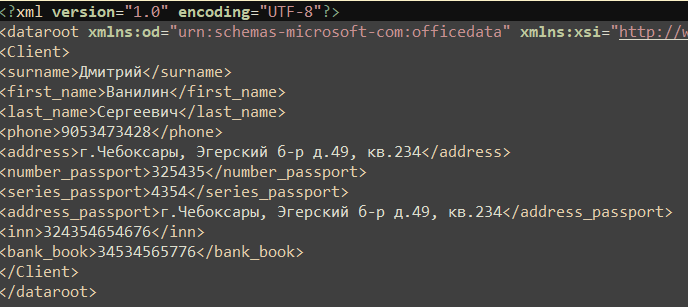
* Кнопка «Обновить» - обращается к базе данных Access и обновляет все записи формы клиентов.
* Кнопка «Добавить» - при добавлении нового клиента(ов) в форму клиентов, пользователь может нажать упомянутую кнопку и добавить новые записи в базу данных Access.
* Кнопка «Удалить» - для удаления клиенты из базы данных Access, пользователь может выбрать нужные строки и нажать на упомянутую кнопку.
* Присутствует алгоритм фильтрации/выборки данных из формы клиентов. Пользователь может написать ФИО клиента(ов), которые ему нужные и нажать кнопку «Поиск по ФИО». Результатом будут все строки, с подходящими значениями.

# Разработка XML-документа

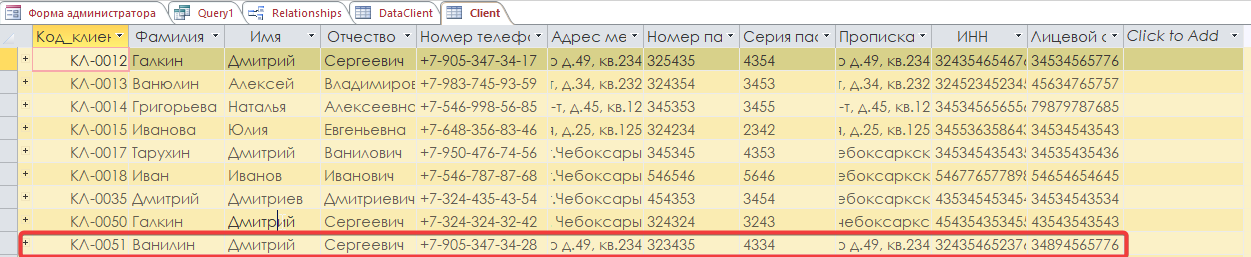
Экспортируемый XML из Access:



Изменения в XML документе:



Добавление новой записи в бд:



**Вывод**

В ходе выполнения курсового проекта «Проектирование базы данных» на тему «СДЭК» были получены навыки проектирования. Также были изучены возможности отслеживания поломок. Удалось освоить создание базы данных, таблиц, запросов, форм, отчетов, макросов, также научился работать с компонентами Office.

# Список литературы

**1. Кузнецов С.Д. Основы баз данных - 2-е изд. - М.: Интернет-университет информационых технологий; БИНОМ. Лаборатория знаний, 2007. - 484с.**

**2. Дубнов, П.Ю. Access 2000. Проектирование баз данных / П.Ю. Дуюновю - .:ДМК, 2014.**

**Приложение**

**Код программы на C# (Windows Form)**

*<?xml version*="1.0" *encoding*="utf-8"*?>*<configuration>  
 <startup>   
 <supportedRuntime *version*="v4.0" *sku*=".NETFramework,Version=v4.8" />  
 </startup>  
 <connectionStrings>  
 <add *name*="AccessConnection" *connectionString*="provider=Microsoft.ACE.OLEDB.16.0;Data Source=D:\Projects\Chuvsu\Chuvsu\_2Kurs\_Database\СДЭК.accdb" />  
 </connectionStrings>  
 <runtime>  
 <assemblyBinding *xmlns*="urn:schemas-microsoft-com:asm.v1">  
 <dependentAssembly>  
 <assemblyIdentity *name*="System.Threading.Tasks.Dataflow" *publicKeyToken*="b03f5f7f11d50a3a" *culture*="neutral" />  
 <bindingRedirect *oldVersion*="0.0.0.0-4.5.24.0" *newVersion*="4.5.24.0" />  
 </dependentAssembly>  
 </assemblyBinding>  
 </runtime>  
</configuration>

*public class* Client  
**{** *public int* IdClient { get; set; }  
 *public string* FirstName { get; set; }  
 *public string* Surname { get; set; }  
 *public string* LastName { get; set; }  
 *public string* Phone { get; set; }  
 *public string* Address { get; set; }  
 *public string* NumberPassport { get; set; }  
 *public string* SeriesPassport { get; set; }  
 *public string* AddressPassport { get; set; }  
 *public string* Inn { get; set; }  
 *public string* BankBook { get; set; }  
**}**

*private* OleDbConnection \_dbConnection;  
*private* OleDbCommand \_dbCommand;  
*private int* \_indexId = 90;  
*public* Form1()  
**{** InitializeComponent();  
   
 *string* connectionString = ConfigurationManager.ConnectionStrings["AccessConnection"].ConnectionString;  
 \_dbConnection = *new* OleDbConnection(connectionString); *// Creating connection* updateDataBase();  
**}**

*/// <summary>  
/// Check data in table  
/// </summary>  
/// <param name="dataGridViewRow"></param>  
/// <returns></returns>  
private bool* checkDataInTable(DataGridViewRow dataGridViewRow)  
{  
 *var* cellsCount = dataGridViewRow.Cells.Count;  
 *var* textMessage = $"Не заполнены поля: ";  
 *bool* isError = *false*;  
 dataGridViewRow.Selected = *true*;  
 *int*[] notIndex = *new int*[] { 0, 3, 9 };  
 *for* (*int* i = 0; i < cellsCount; i++)  
 {  
 *if* (notIndex.Contains(i)) *continue*;  
   
 *if* (dataGridViewRow.Cells[i].Value == *null*)  
 {  
 textMessage += $"{client\_list.Columns[i].HeaderText} ";  
 isError = *true*;  
 client\_list.CurrentCell = dataGridViewRow.Cells[i];  
 }  
 }  
  
 *if* (isError) MessageBox.Show(textMessage, "Внимание!");  
  
 *return* isError;  
}  
  
  
*private void* updateDataBase()  
**{** client\_list.Rows.Clear();  
 client\_list.Refresh();  
   
 *// Running Query in Db* \_dbConnection.Open(); *// Open connecting  
 string* query = "SELECT *\** FROM Client;"; *// string query* \_dbCommand = *new* OleDbCommand(query, \_dbConnection); *// command* OleDbDataReader dbDataReader = \_dbCommand.ExecuteReader(); *// reading data  
   
   
 // Check data  
 if* (!dbDataReader.HasRows)  
 {  
 MessageBox.Show("Данные были не найдены.", "Ошибка чтения!");  
 }  
 *else* {  
 *// Writing data in table forms  
 while* (dbDataReader.Read())  
 {  
 client\_list.Rows.Add(  
 dbDataReader["Код\_клиента"],  
 dbDataReader["first\_name"], dbDataReader["surname"], dbDataReader["last\_name"],  
 dbDataReader["phone"], dbDataReader["address"],  
 dbDataReader["number\_passport"], dbDataReader["series\_passport"],  
 dbDataReader["address\_passport"],  
 dbDataReader["inn"], dbDataReader["bank\_book"]  
 );  
 }  
  
 *var* indexEnd = client\_list.Rows.Count;  
 *for* (*int* i = 0; i < indexEnd - 1; i++)  
 {  
 \_indexId = \_indexId < (*int*)(client\_list.Rows[i].Cells[0].Value) ? (*int*)(client\_list.Rows[i].Cells[0].Value) : \_indexId;  
 }  
 }  
   
 *// Exiting connection* dbDataReader.Close();  
 \_dbConnection.Close();  
**}**

*/// <summary>  
/// Deleting data in Access button  
/// </summary>  
/// <param name="sender"></param>  
/// <param name="e"></param>  
private void* button\_del\_Click(*object* sender, EventArgs e)  
**{** *// Check count selected rows  
 if* (client\_list.SelectedRows.Count == 0)  
 {  
 MessageBox.Show("Выберите строки, которые нужно занести в базу данных", "Внимание!");  
 *return*;  
 }  
   
 *var* clientRows = client\_list.SelectedRows;  
 List<*int*> indexes = *new* List<*int*>();  
 *int* endIndex = clientRows.Count;  
  
 *for* (*int* i = 0; i < endIndex; i++)  
 {  
 *var* index = clientRows[i].Index;  
 indexes.Add(i);  
 }  
   
 \_dbConnection.Open();  
  
 *foreach* (*var* index *in* indexes)  
 {  
 Client client = *new* Client();  
 client.IdClient = (*int*)clientRows[index].Cells[0].Value;  
 client.FirstName = clientRows[index].Cells[1].Value.ToString();  
 client.Surname = clientRows[index].Cells[2].Value.ToString();  
 client.LastName = clientRows[index].Cells[3].Value.ToString();  
 client.Phone = clientRows[index].Cells[4].Value.ToString();  
 client.Address = clientRows[index].Cells[5].Value.ToString();  
 client.NumberPassport = clientRows[index].Cells[6].Value.ToString();  
 client.SeriesPassport = clientRows[index].Cells[7].Value.ToString();  
 client.AddressPassport = clientRows[index].Cells[8].Value.ToString();  
 client.Inn = clientRows[index].Cells[9].Value.ToString();  
 client.BankBook = clientRows[index].Cells[10].Value.ToString();  
  
 *string* query = $"DELETE FROM Client WHERE Код\_клиента = {client.IdClient.ToString()}";  
  
 \_dbCommand = *new* OleDbCommand(query, \_dbConnection);  
  
 *if* (\_dbCommand.ExecuteNonQuery() != 1)  
 MessageBox.Show("Ошибка выполнения запроса на удаления клиентов(а).", "Ошибка!");  
 *else* MessageBox.Show("Данные удалены.", "Внимание!");  
 }  
   
 \_dbConnection.Close();  
**}**

*/// <summary>  
/// Update data button  
/// </summary>  
/// <param name="sender"></param>  
/// <param name="e"></param>  
private void* button\_update\_Click(*object* sender, EventArgs e)  
**{** updateDataBase();  
**}**

*/// <summary>  
/// Adding data in Access button  
/// </summary>  
private void* button\_add\_Click(*object* sender, EventArgs e)  
{  
 *// Check count selected rows  
 if* (client\_list.SelectedRows.Count == 0)  
 {  
 MessageBox.Show("Выберите строки, которые нужно занести в базу данных", "Внимание!");  
 *return*;  
 }  
   
 *var* clientRows = client\_list.SelectedRows;  
 List<*int*> indexes = *new* List<*int*>();  
 *int* endIndex = clientRows.Count;  
  
 *for* (*int* i = 0; i < endIndex; i++)  
 {  
 *var* index = clientRows[i].Index;  
 *if* (checkDataInTable(clientRows[i])) *return*;  
 indexes.Add(i);  
 }  
   
 \_dbConnection.Open();  
  
 *foreach* (*var* index *in* indexes)  
 {  
 Client client = *new* Client();  
 client.IdClient = ++\_indexId;  
 client.FirstName = clientRows[index].Cells[1].Value.ToString();  
 client.Surname = clientRows[index].Cells[2].Value.ToString();  
 client.LastName = clientRows[index].Cells[3].Value.ToString();  
 client.Phone = clientRows[index].Cells[4].Value.ToString();  
 client.Address = clientRows[index].Cells[5].Value.ToString();  
 client.NumberPassport = clientRows[index].Cells[6].Value.ToString();  
 client.SeriesPassport = clientRows[index].Cells[7].Value.ToString();  
 client.AddressPassport = clientRows[index].Cells[8].Value.ToString();  
 client.Inn = clientRows[index].Cells[9].Value.ToString();  
 client.BankBook = clientRows[index].Cells[10].Value.ToString();  
  
 *string* query = $"INSERT INTO Client (first\_name, surname, last\_name, phone, address, number\_passport, series\_passport, address\_passport, inn, bank\_book) VALUES (" +  
 $"'{client.FirstName}', " +  
 $"'{client.Surname}', " +  
 $"'{client.LastName}', " +  
 $"'{client.Phone}', " +  
 $"'{client.Address}', " +  
 $"'{client.NumberPassport}', " +  
 $"'{client.SeriesPassport}', " +  
 $"'{client.AddressPassport}', " +  
 $"'{client.Inn}', " +  
 $"'{client.BankBook}')";  
  
 \_dbCommand = *new* OleDbCommand(query, \_dbConnection);  
   
 *if* (\_dbCommand.ExecuteNonQuery() != 1)  
 {  
 MessageBox.Show("Ошибка выполнения запроса на изменения записей.", "Ошибка!");  
 \_dbConnection.Close();  
 *return*;  
 }  
 }  
   
 MessageBox.Show("Данные добавлены.", "Внимание!");  
   
 \_dbConnection.Close();  
}

*/// <summary>  
/// Save data button  
/// </summary>  
/// <param name="sender"></param>  
/// <param name="e"></param>  
private void* button\_save\_Click(*object* sender, EventArgs e)  
**{** *// Check count selected rows  
 // if (client\_list.SelectedRows.Count == 0)  
 // {  
 // MessageBox.Show("Выберите строки, которые нужно занести в базу данных", "Внимание!");  
 // return;  
 // }  
   
 var* clientRows = client\_list.SelectedRows;  
 List<*int*> indexes = *new* List<*int*>();  
 *int* endIndex = clientRows.Count;  
  
 *for* (*int* i = 0; i < endIndex; i++)  
 {  
 *var* index = clientRows[i].Index;  
 *if* (checkDataInTable(clientRows[i])) *return*;  
 indexes.Add(i);  
 }  
   
 \_dbConnection.Open();  
  
 *foreach* (*var* index *in* indexes)  
 {  
 Client client = *new* Client();  
 client.IdClient = (*int*)clientRows[index].Cells[0].Value;  
 client.FirstName = clientRows[index].Cells[1].Value.ToString();  
 client.Surname = clientRows[index].Cells[2].Value.ToString();  
 client.LastName = clientRows[index].Cells[3].Value.ToString();  
 client.Phone = clientRows[index].Cells[4].Value.ToString();  
 client.Address = clientRows[index].Cells[5].Value.ToString();  
 client.NumberPassport = clientRows[index].Cells[6].Value.ToString();  
 client.SeriesPassport = clientRows[index].Cells[7].Value.ToString();  
 client.AddressPassport = clientRows[index].Cells[8].Value.ToString();  
 client.Inn = clientRows[index].Cells[9].Value.ToString();  
 client.BankBook = clientRows[index].Cells[10].Value.ToString();  
  
 *string* query = $"UPDATE Client SET " +  
 $"first\_name = '{client.FirstName}'," +  
 $"surname = '{client.Surname}'," +  
 $"last\_name = '{client.LastName}'," +  
 $"phone = '{client.Phone}'," +  
 $"address = '{client.Address}'," +  
 $"number\_passport = '{client.NumberPassport}'," +  
 $"series\_passport = '{client.SeriesPassport}'," +  
 $"address\_passport = '{client.AddressPassport}'," +  
 $"inn = '{client.Inn}'," +  
 $"bank\_book = '{client.BankBook}'" +  
 $"WHERE Код\_клиента = {client.IdClient.ToString()}";  
  
 \_dbCommand = *new* OleDbCommand(query, \_dbConnection);  
  
 *if* (\_dbCommand.ExecuteNonQuery() != 1)  
 {  
 MessageBox.Show("Ошибка выполнения запроса на изменения записей.", "Ошибка!");  
 \_dbConnection.Close();  
 *return*;  
 }  
 }  
   
 MessageBox.Show("Данные изменены.", "Внимание!");  
   
 \_dbConnection.Close();  
}

*/// <summary>  
/// Exiting the programm  
/// </summary>  
/// <param name="sender"></param>  
/// <param name="e"></param>  
private void* menuExit\_Click(*object* sender, EventArgs e)  
{  
 Application.Exit();  
}  
  
*/// <summary>  
/// Full Name Search button  
/// </summary>  
/// <param name="sender"></param>  
/// <param name="e"></param>  
private void* button\_find\_Click(*object* sender, EventArgs e)  
**{** *if* (client\_list.Rows.Count == 0)  
 {  
 MessageBox.Show("В таблице нет данных для поиска", "Внимание!");  
 *return*;  
 }  
   
 *var* fullname = FullName.Text;  
  
 *if* (fullname == String.*Empty*)  
 {  
 MessageBox.Show("Введите ФИО в строку поиска", "Внимание!");  
 *return*;  
 }  
  
 *var* endIndex = client\_list.Rows.Count;  
 *bool* isFullNameClient = *false*;  
  
 *for* (*int* i = 0; i < endIndex - 1; i++)  
 {  
 *var* client = client\_list.Rows[i];  
 *var* fullnameClient = $"{client.Cells[1].Value} {client.Cells[2].Value} {client.Cells[3].Value}";  
  
 *if* (!fullnameClient.Contains(fullname))  
 {  
 client.Visible = *false*;  
 }  
 *else* {  
 isFullNameClient = *true*;  
 }  
 }  
  
 *if* (!isFullNameClient)  
 {  
 MessageBox.Show("Клиента с таким ФИО в базе данных не обнаружжено.", "Внимание!");  
 updateDataBase();  
 }  
**}**