МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА ИНФОРМАЦИОННО-СЕТЕВЫХ ТЕХНОЛОГИЙ

КУРСОВАЯ РАБОТА (ПРОЕКТ)   
ЗАЩИЩЕНА С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| доцент |  |  |  | Н.А. Волкова |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ |
| ПОНЯТИЕ ТРАНСПОРТНОЙ ЗАДАЧИ (ТЗ). СЕТЕВАЯ ПОСТАНОВКА ТЗ |
| по дисциплине: ПРИКЛАДНЫЕ МОДЕЛИ ОПТИМИЗАЦИИ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | Z1431 |  |  |  | М.Д. Быстров |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2025

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc516226212)

[1 Требования к реляционной базе данных 5](#_Toc516226213)

[2 Проектирование системы управления базами данных-независимого концептуального представления данных 6](#_Toc516226214)

[3 Проектирование системы управления базами данных-ориентированного реляционного концептуального представления данных 8](#_Toc516226215)

[4 Реализация базы данных 12](#_Toc516226216)

[4.1 Таблицы, построенные при создании базы данных 12](#_Toc516226217)

[4.2 Запросы, на которые ориентирована база данных 14](#_Toc516226218)

[4.3 Формы, представленные в базе данных 15](#_Toc516226219)

[4.4 Отчеты, представленные в базе данных 18](#_Toc516226220)

[4.5 Защита данных средствами системы управления базами данных 19](#_Toc516226221)

[ЗАКЛЮЧЕНИЕ 21](#_Toc516226222)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 22](#_Toc516226223)

[ПРИЛОЖЕНИЕ А. SQL-коды запросов 23](#_Toc516226224)

[ПРИЛОЖЕНИЕ Б. Код процесса ввода пароля 26](#_Toc516226225)

# ВВЕДЕНИЕ

Для обеспечения наибольшей эффективности в производственной деятельности в современной производственной среде активно применяются алгоритмы оптимизации, расчеты по которым производятся с использованием вычислительных систем.

Одной из важнейших задач планирования производства является планирование доставки ресурсов в соответствии с потребностями потребителей и наличием ресурсов у поставщиков, а также транспортных расходов (расстояний от каждого поставщика к каждому потребителю). Эта задача может быть формализована как транспортная задача, которая является одной из задач линейного программирования.

Условия транспортной задачи могут быть заданы в различной форме. На практике оценка параметров транспортной работы производится в том числе с использованием картографических данных, которые позволяют определить расходы по транспортировке груза от каждого поставщика к каждому потребителю. Сетевая постановка транспортной задачи позволяет в более наглядной форме представить начальные условия, т.к. является сравнительно более удобной для восприятия формой определения условий транспортной задачи.

Целью данной курсовой работы является рассмотрение понятия транспортной задачи и постановки транспортной задачи в сетевой форме.

Рассмотрение понятия транспортной задачи будет произведено с помощью общих содержательной и формализованной постановок транспортной задачи.

Рассмотрение сетевой постановки транспортной задачи будет дано как описание особенностей постановки задачи в сетевой форме, а также обусловленных этими особенностями модификаций метода потенциалов для решения ТЗ.

Объектом исследования является транспортная задача с постановкой в сетевой форме.

Предметом исследования является применение метода потенциалов для решения транспортной задачи в сетевой постановке.

Транспортная задача является задачей линейного программирования. Основы линейного программирования заложил советский математик Л.В. Канторович в работе 1939 г. «Математические методы организации и планирования производства». Американский математик Джордж Бернард Данциг в 1949 году разработал эффективный метод решения задач линейного программирования – симплекс-метод. Применительно к транспортной задаче существует модификация симплекс-метода – метод потенциалов, который позволяет за конечное число итераций получить оптимальный план, основываясь на опорном решении. В свою очередь, для нахождения опорного решения существуют такие методы, как метод северо-западного угла, метод наименьшей стоимости.

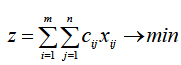
Для сетевой постановки транспортной задачи также применим метод потенциалов. Будут рассмотрены различия алгоритма при классической и сетевой постановке задачи.

# 1 Транспортная задача

Пусть имеются *m* пунктов отправления и *n* пунктов назначения груза. Обозначим через *сij,* стоимость перевозки груза из пункта отправления с номером *i* к пункту назначения с номером *j*, а через *xij* обозначим объём перевозки груза в вышеуказанных пунктах. Запасы груза в пунктах отправлений обозначим через *a1, a2, …am* , потребности к грузам в пунктах назначений обозначим через *b1, b2, …bn*. Общий стоимость перевозки груза обозначим через формулы:



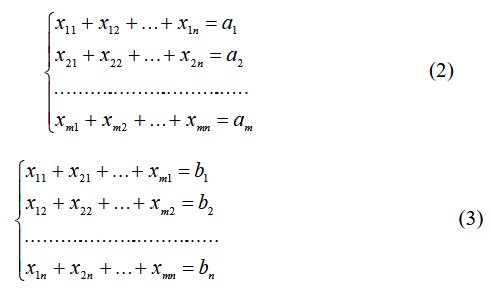
Общую стоимость перевозки груза необходимо уменьшить. Поэтому функцию *z* следует минимизировать:



Задачу представляем с помощью следующей таблицы:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | *1* | *2* | *…* | *n* | Запасы |
| *1* | *c*11 *x*11 | *c*12 *x*12 | … | *c*1*n*  *x*1*n* | *a*1 |
| *2* | *c*21 *x*21 | *c*22 *x*22 | … | *c*2*n*  *x*2*n* | *a*2 |
| *…* | … | … | … | … | … |
| *m* | *cm*1  *xm*1 | *cm*2 *xm*2 | … | *cmn*  *xmn* | *am* |
| Потребности | *b*1 | *b*2 | … | *bn* |  |

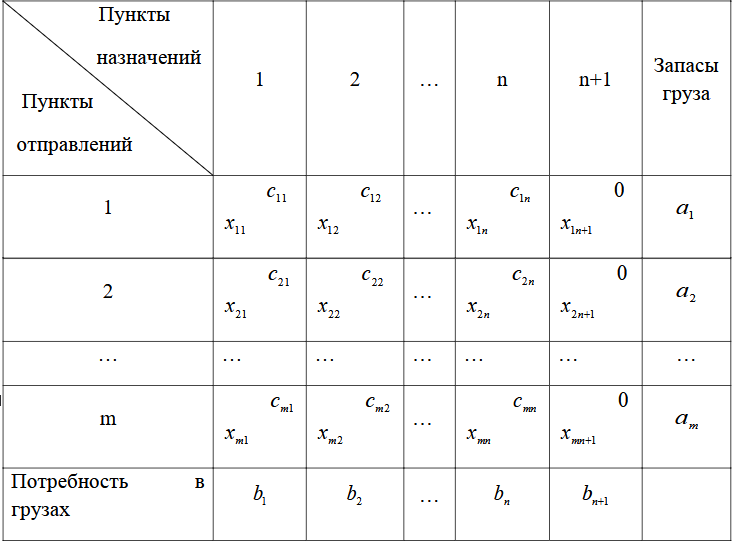
Перевозимый груз нужно распределить так, что при этом все грузы из пункта отправления должны быть вывезены, а потребности к грузу в пунктах назначений должны быть удовлетворены:



Если выполняется условие , то транспортная задача называется закрытой, и можно приступить к решению задачи.

Если выполняется условие , то задача называется открытой. В этом случае с помощью ввода дополнительных пунктов задача приводится к закрытой.

Например, если , то вводится дополнительный пункт отправления с номером m+1, с запасами  и со стоимостью перевозки груза, равной 0: . В результате задача принимает вид:



Если , составляются дополнительные пункты отправлений с запасами груза, и задача приводится к закрытой.

Клетки с грузами *xij* != 0 называются отмеченными, а клетки с грузами *xij* = 0 называются не отмеченными. Для отмеченных клеток с помощью формулы *vj* -*ui* = *cij* определяем значения потенциалов *v j* , *j* =1,2,...*n* и *ui* , *i* =1,2,...*m*.

Транспортная задача решается в два этапа:

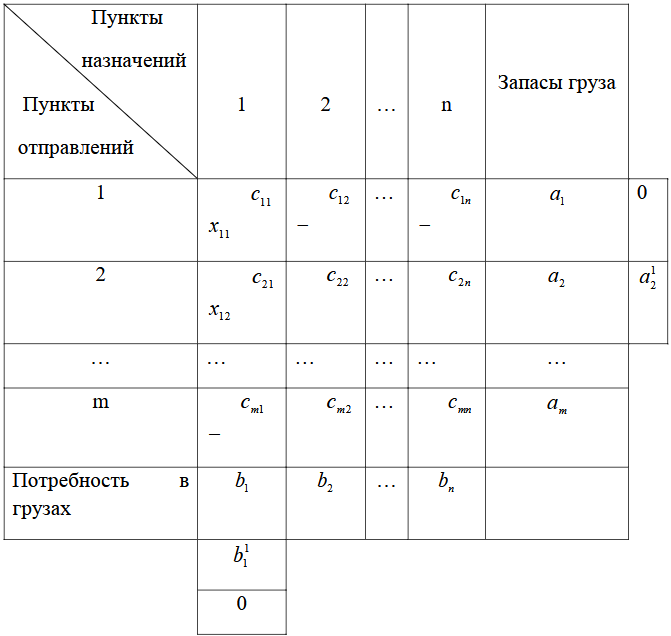
В первом этапе находится первоначальное решение   
*xij, i*=1*,*2*,*…*,m; j* =1*,*2*,*…*,n* , удовлетворяющее условиям (2)-(3). Имеются несколько способов для нахождения первоначального решения, например, метод северо-западного угла, метод минимального элемента и другие. В методе северо-западного угла выбирается клетка (1,1) и отметим, что *x*11 = *min(a*1*,b*1*)* . Если *min(a*1*,b*1*)* = *a*1, то это означает, что все грузы из 1-го пункта отправления направлены к 1-пункту назначений, другим пунктам назначений из 1- пункта отправления груз не отправляется. Поэтому, к остальным клеткам в строке, где находится *a*1 вставляется знак «-». В 1- пункте назначения потребность в грузах будет *b*11 = *b*1 -*a*1.



Если *min(a*1*,b*1*)* =*b*1, то в 1- пункте назначения потребность в грузах будет удовлетворена, в 1-пункте отправления остаётся груз *a*11 = *a*1 -*b*1. К первому пункту назначения из остальных пунктов отправлений груз не привозится.

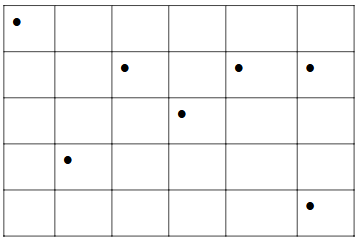


Продолжая вычисления по 1-таблице, переходим к клетке (2,1). Пусть будет *x*21 = *min*(*a*1*,b*11)=*b*11 . Заполняя клетку вышеуказанным способом, получаем следующее:

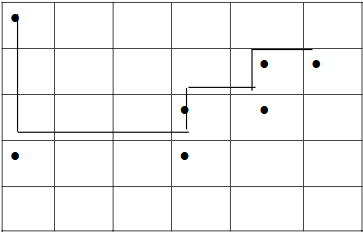


Продолжая вычисления таким образом до правого нижнего угла, определяем все значения *xij , i*=1*,*…*,m; j* =1*,*…*,n* . При этом должны выполняться условия (2)-(3).

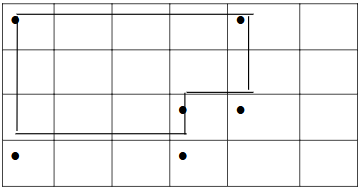
Во втором этапе находится оптимальное решение(план), удовлетворяющее условиям (1). Для нахождения оптимального плана имеется несколько способов, например метод потенциалов, метод распределений и т.д. Рассмотрим метод потенциалов. Для этого сначала ознакомимся с некоторыми понятиями. Произвольное множество точек в таблице называется набором. Например,



Если в наборе число точек в каждой строке не превышает двух, то такой набор называется цепью. Например,



Замкнутая цепь называется циклом. Например,



Если в таблице набор из *n* количество точек не образуют цикл, при добавлении определенной точки набор *n*+1 точек образуют цикл, то первоначальный набор *n* точек называется ациклическим планом.

Если в транспортной задаче *xij* > 0, то клетка *(i,j)* называется отмеченной.

Если в транспортной задаче для всех клеток находится план *xij , i*=1*,*…*,m; j* =1*,*..*,n* , для которой удовлетворяется условие *vj* -*ui* <=*cij* (4) , а для отмеченных клеток удовлетворяется условие *v j* -*ui* = *cij* , то полученный план называется оптимальным. Множество чисел *v j ,* *j* =1*,*2*,...,n;* *ui ,* *i* =1*,*2*,...,m* называются потенциалами.

Метод потенциалов в транспортной задаче выполняется в следующем порядке:

1. Составляется система уравнений для отмеченных клеток удовлетворяющая следующим условиям *v j* -*ui* = *cij* , *v j ,* *j* =1*,*2*,...,n;* *ui ,* *i* =1*,*2*,...,m* . При этом число уравнений на одно меньше, чем число неизвестных. Поэтому система имеет бесконечное число решений. Найдя одно частное решение системы (приняв одно из неизвестных равным нулю), определим значение потенциалов;
2. Для неотмеченных клеток проверим условие *vj* -*ui* =*cij* . Если это условие выполняется для всех клеток, то этот план будет оптимальным, и вычисляется значение функции ;
3. Если условие *vj* -*ui* <= *cij* не выполняется для некоторых клеток, то для этих клеток вычисляем   
     
   и находим   
   ;
4. клетка *(i*0*,j*0 *)* добавляется в набор отмеченных клеток, и для этого набора составляется цикл;
5. начиная с клетки *(i*0*,j*0 *)*, по очереди вставляем знаки «+» и «-» к клеткам цикла. Вставка происходит, начиная со знака «+»;
6. для клеток со знаком «-» определяем   
   ;
7. Из чисел *xij* в клетках со знаком «-» вычитываем  , к числам *xij* в клетках со знаком «+» прибавляем ;
8. Клетка с  удаляется из числа отмеченных клеток.

В результате получаем новый план. Для нового плана повторяем операции (1)-(7). Вышеуказанные операции повторяются до тех пор, пока не выполняется условие *vj* -*ui* <=*cij* для всех клеток.

# 2 Сетевая постановка транспортной задачи

Очень часто особенности исходной информации таковы, что возможно рассматривать их в сетевой постановке, и это позволяет иногда использовать более простые алгоритмы их решения. К числу таких задач относится и транспортная задача в сетевой постановке. Экономическая ситуация состоит в следующем.

Пусть имеется ***N*** пунктов (производства, потребления, транзитной транспортировки грузов), связанных между собой некоторой транспортной сетью. В каждом пункте сети заданы числа

*ai* . Если *ai* < 0, то в этом пункте продукция производится, если а, > 0, то продукция потребляется, а если *ai* = 0, то данный пункт является транзитным, т.е. все, что в него привезено, должно быть вывезено.

Пусть транспортная сеть содержит *s* участков пути. Под участком пути понимается часть сети, соединяющая любые два ее пункта. Рассмотрим *q*-ый участок пути, на котором осуществляется перевозка:

*xq*

*iq cq jq*

где:

*iq* - пункт, из которого груз вывозится;

*jq* - пункт, в который груз завозится;

*cq* - стоимость перевозки единицы груза на этом участке пути:

*xq* - объем перевозки из пункта *iq*  в пункт *jq*.

Стрелка → показывает направление перевозки груза.

Требуется найти такой план перевозки груза, при котором из пунктов производства вывозится вся продукция, в пунктах потребления удовлетворяются их потребности, а суммарные затраты на перевозку грузов были бы минимальных, т.е. необходимо найти такой вектор X = (*x1, x2, … xs*), для которого:

 (5)

при условии

 , ; *, ;*

где

- объем груза, ввозимого в пункт ***i***,

 - объем груза, вывозимого из ***i*** .

Необходимым и достаточным условием разрешимости данной задачи является

; (6)

т.е. все, что произведено, должно быть потреблено.

Решение задачи осуществляется методом потенциалов. Опорный невырожденный план транспортной задачи должен содержать ровно *(N - 1)* положительную перевозку, не иметь замкнутых маршрутов и «висячих» пунктов.

Первоначальный опорный план строится по методу наименьшей стоимости. Затем для каждого пункта сети находятся потенциалы по формуле

*Vjq - Vjq =Cq, Xq > 0*, (7)

где *Vjq* и *Vjq* - потенциалы пунктов, ограничивающих один и тот же *q*-ьм участок пути, a *Cq* - стоимость единицы перевозимого по этому участку груза.

Для оптимального плана транспортной задачи в сетевой постановке для всех участков пути должно выполняться условие:

 , . (8)

Если условие (8) не выполняется, то для тех участков пути, где оно не выполняется, рассчитывается невязка по формуле:

;

и на участке сети с наибольшей невязкой вводится *ε*-перевозка, определяется *ε*-маршрут, величина *ε*-перевозки, рассчитывается новый опорный план, который проверяется на оптимальность. Процедура повторяется до тех пор, пока не будет выполняться условие (8)..

# 3 Проектирование системы управления базами данных-ориентированного реляционного концептуального представления данных

На этом этапе проектирования необходимо построить корректную схему БД, ориентируясь на реляционную модель данных. Реляционную модель данных можно получить на основе разработанной ER-модели, используя алгоритм однозначного преобразования ее в реляционную модель данных [3].

Корректная схема БД – это схема, в которой отсутствуют нежелательные зависимости между атрибутами отношении. Проектирование корректной схемы данных может быть выполнено путем декомпозиции. При этом процесс проектирования представляет собой процесс последовательной нормализации схем отношений, при этом каждая следующая итерация соответствует нормальной форме более высокого уровня и обладает лучшими свойствами по сравнению с предыдущей [4]. Каждой нормальной форме соответствует некоторый определенный набор ограничений, и отношение находится в некоторой нормальной форме, если удовлетворяет свойственному ей набору ограничений.

В теории реляционных БД обычно выделяется следующая последовательность нормальных форм (НФ):

* первая нормальная форма (1НФ);
* вторая нормальная форма (2НФ);
* третья нормальная форма (3НФ);
* нормальная форма Бойса-Кодда (НФБК);
* четвертая нормальная форма (4НФ);
* пятая нормальная форма, или форма проекции-соединения (5НФ или НФПС).

В большинстве случаев достижение третьей нормальной формы или даже формы Бойса-Кодда считается достаточным для реальных проектов баз данных. Поэтому рассмотрим определения только для 1НФ, 2НФ, 3НФ, НФБК.

Первая и главная нормальная форма (1НФ) требует от таблицы (а точнее, от ее проектировщика) следования определенным правилам. Можно сказать, что таблица в 1НФ, так как:

* устранены повторяющиеся группы в отдельных таблицах;
* созданы отдельные таблицы для каждого набора связанных данных;
* каждый набор связанных данных идентифицирован с помощью первичного ключа.

В одной таблице не может использоваться несколько полей для хранения похожих данных.

Отношение находится во второй нормальной форме (2НФ) тогда и только тогда, когда оно находится в первой нормальной форме, и каждый его неключевой атрибут неприводимо зависим от первичного ключа. Можно сказать, что таблица находится в 2НФ, так как:

* созданы отдельные таблицы для наборов значений, относящихся к нескольким записям;
* эти таблицы были связаны с помощью внешнего ключа.

Третья нормальная форма (3НФ) данных расширяет две предыдущие, неся в себе два правила:

* таблица должна соответствовать второй нормальной форме;
* все столбцы, не входящие в полный первичный ключ, должны зависеть от него и не должны зависеть друг от друга.

Отношение находится в нормальной форме Бойса-Кодда (НФБК), если оно находится в 3НФ и в нем отсутствуют зависимости атрибутов первичного ключа от неключевых атрибутов.

Ситуация, когда отношение будет находиться в 3НФ, но не в НФБК, возникает при условии, что отношение имеет два (или более) возможных ключа, которые являются составными и имеют общий атрибут.

В результате преобразования ER-модели в реляционную и проведения нормализации отношений получаем корректную схему БД, представленную на рисунке 2.

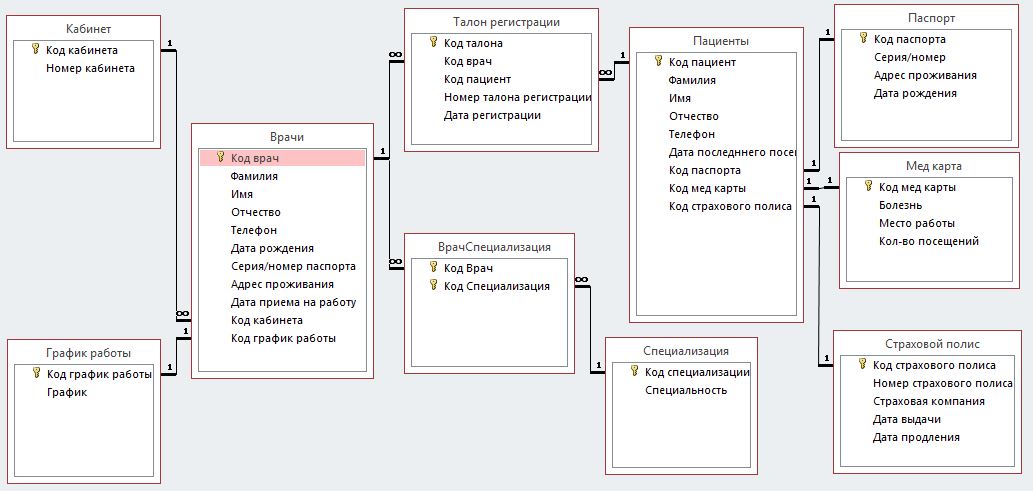


Рисунок 2 – Схема данных

На этой схеме представлены отношения и показано, как будут связаны элементы этих таблиц в Access.

Структуры основных отношений полученной схемы данных приведены в таблицах 1 – 2.

Таблица 1 – Структура отношения «Врач»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Параметры связи | | | Тип данных |
| Связь с таблицей | Внешний ключ таблицы | Тип отношения |
| 1 | 2 | 3 | 4 | 5 |
| Код врач | Талон регистрации  ВрачСпециализация | Код врач  Код врач | Один-ко-многим  Один-ко-многим | Счетчик |
| Фамилия |  |  |  | Текстовый |
| Имя |  |  |  | Текстовый |
| Отчество |  |  |  | Текстовый |

Продолжение таблицы 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| Телефон |  |  |  | Числовой |
| Дата рождения |  |  |  | Дата/Время |
| Серия/номер паспорта |  |  |  | Числовой |
| Адрес проживания |  |  |  | Текстовый |
| Дата приема на работу |  |  |  | Дата/Время |
| Код кабинета | Кабинет | Код кабинета | Многие-к-одному | Числовой |
| Код график работы | График работы | Код график работы | Многие-к-одному | Числовой |

Таблица 2 – Структура отношения «Пациент»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Параметры связи | | | Тип данных |
| Связь с таблицей | Внешний ключ таблицы | Тип отношения |
| Код пациент | Талон регистрации | Код пациент | Один-ко-многим | Счетчик |
| Фамилия |  |  |  | Текстовый |
| Имя |  |  |  | Текстовый |
| Отчество |  |  |  | Текстовый |
| Телефон |  |  |  | Числовой |
| Дата последнего посещения |  |  |  | Дата/Время |
| Код паспорта | Паспорт | Код паспорта | Один-к-одному | Числовой |
| Код мед карты | Мед карта | Код мед карты | Один-к-одному | Числовой |
| Код страхового полиса | Страховой полис | Код страхового полиса | Один-к-одному | Числовой |

# 4 Реализация базы данных

## 4.1 Таблицы, построенные при создании базы данных

На основе схемы данных (рисунок 2) были созданы десять таблиц.

В качестве примера рассмотрим процесс создания таблицы «Пациент». Ее вид в режиме конструктора представлен на рисунке 3.

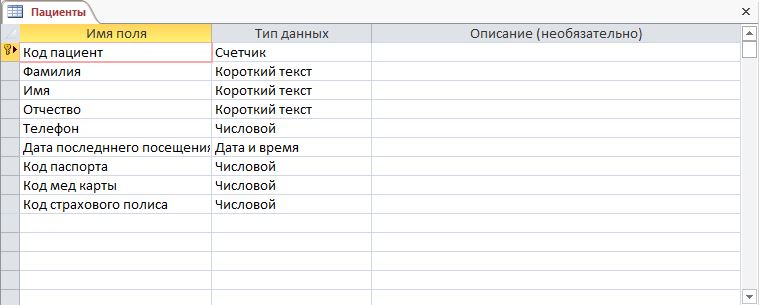


Рисунок3 *–* Вид таблицы «Пациенты» в режиме конструктора

Эта же таблица, но уже заполненная, как и все остальные, случайными и придуманными данными, представлена на рисунке 4.

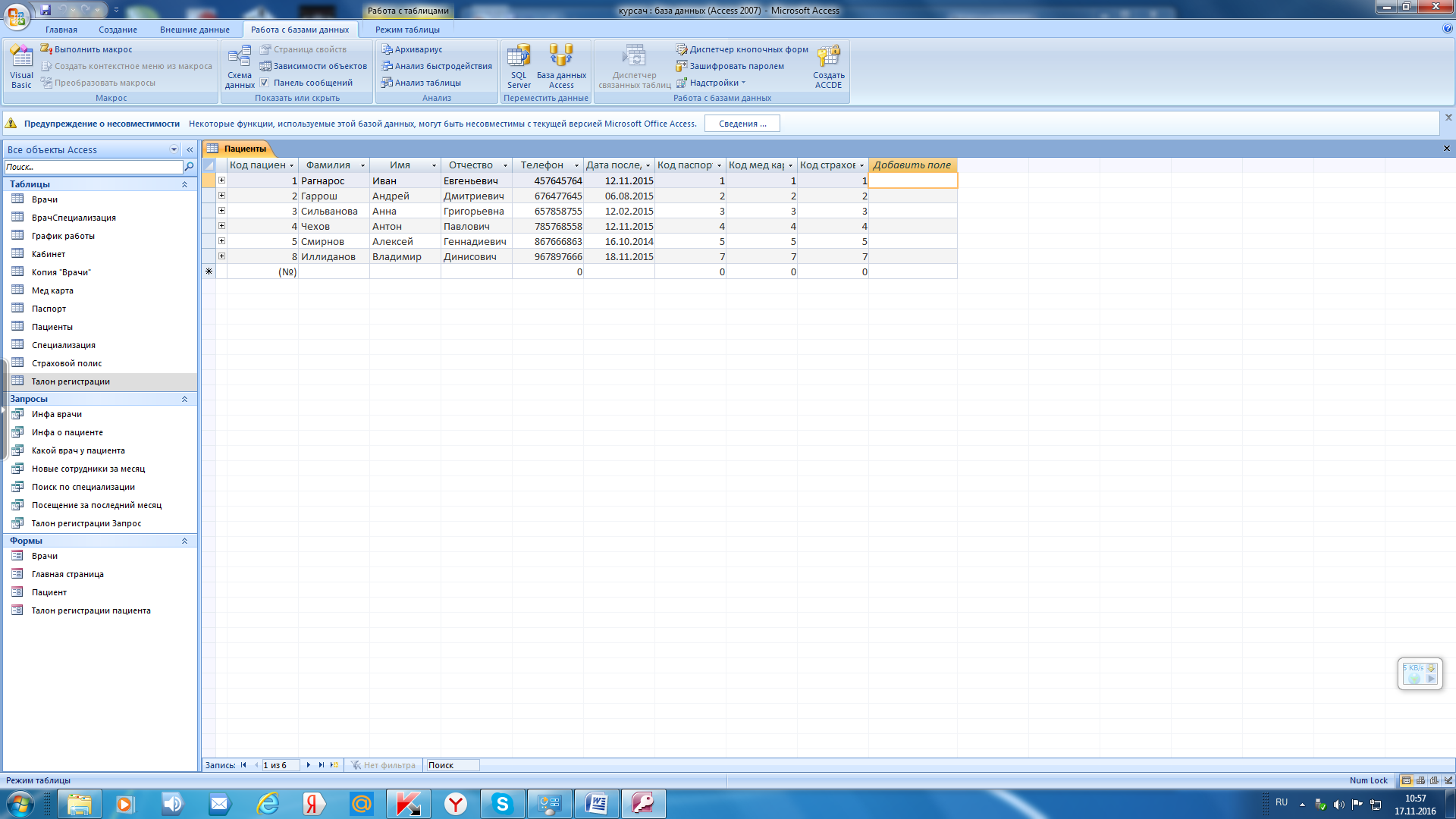


Рисунок 4 – Внешний вид таблицы «Пациенты» в табличном режиме

Внешний вид остальных таблиц, созданных в БД, представлен на рисунках 5 – 8.

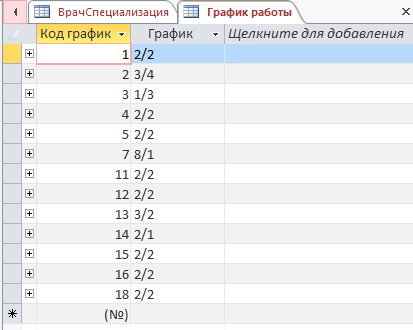


Рисунок 5 – Заполненная таблица «График работы»

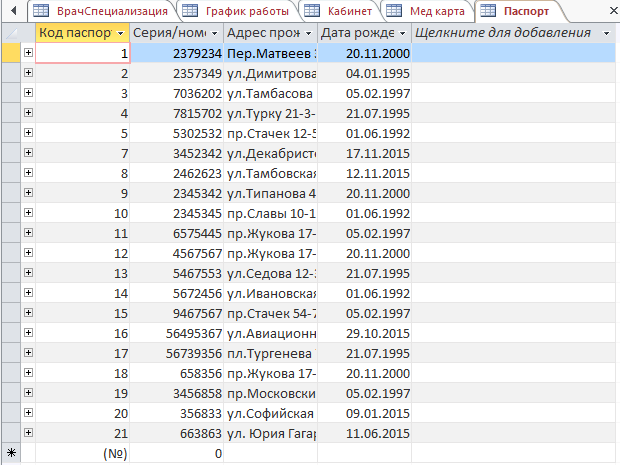


Рисунок 6 – Заполненная таблица «Паспорт»

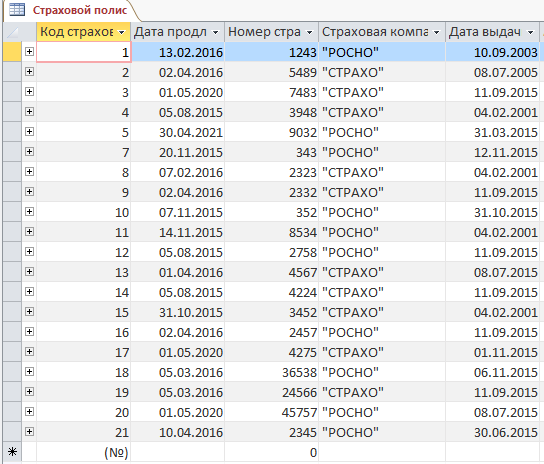


Рисунок 7 – Заполненная таблица «Страховой полис»

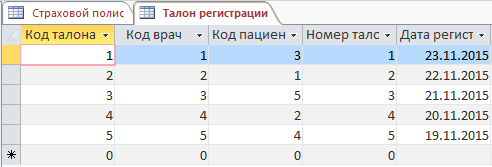


Рисунок 8 – Заполненная таблица «Талон регистрации»

## 4.2 Запросы, на которые ориентирована база данных

Запросы обеспечивают быстрый и эффективный доступ к данным, хранящимся в таблицах. Благодаря запросам можно выполнить сортировку или вычислить выражения. Например, свести вместе данные из связанных таблиц.

При разработке данной базы данных были реализованы запросы:

* «Инфо врачи» – выводит всю информацию, об интересующем враче;
* «Инфо о пациенте» – выводит всю информацию об интересующем пациенте;
* «Какой врач у пациента» – выводит ФИО и специальность врача у выбранного пациента;
* «Поиск по специализации» – запрос выводит ФИО врача или врачей с указанной специализацией;
* «Посещение за последний месяц» – выводит пациентов, посетивших поликлинику за последний месяц;
* «Талон» – выводит пациента и лечащего врача для указанного номера талона регистрации.

SQL-коды указанных запросов приведены в приложении А. Результаты работы запросов представлены на рисунках 9 – 10.

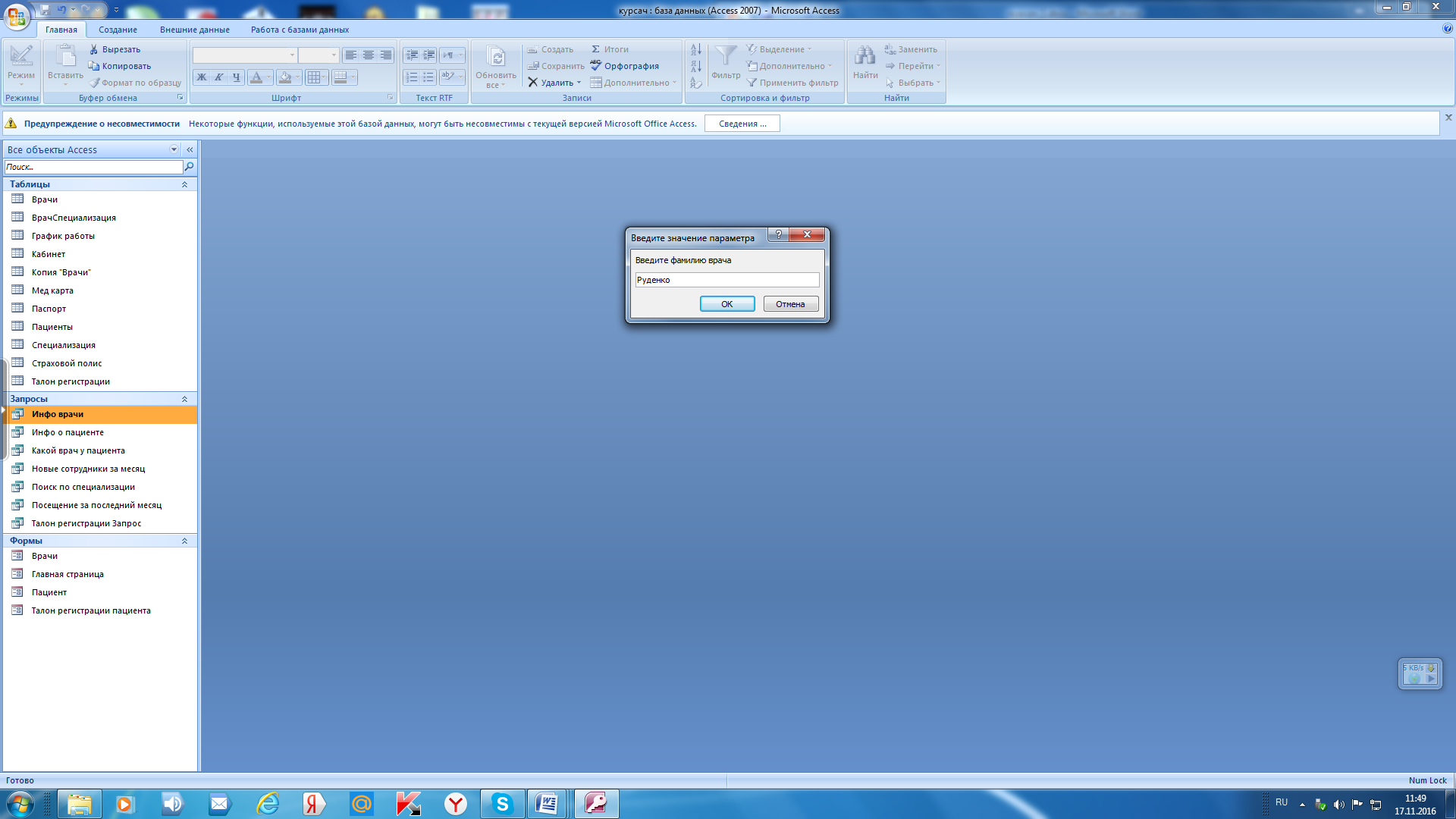


Рисунок 9 – Работа запроса на выборку информации об определенном враче

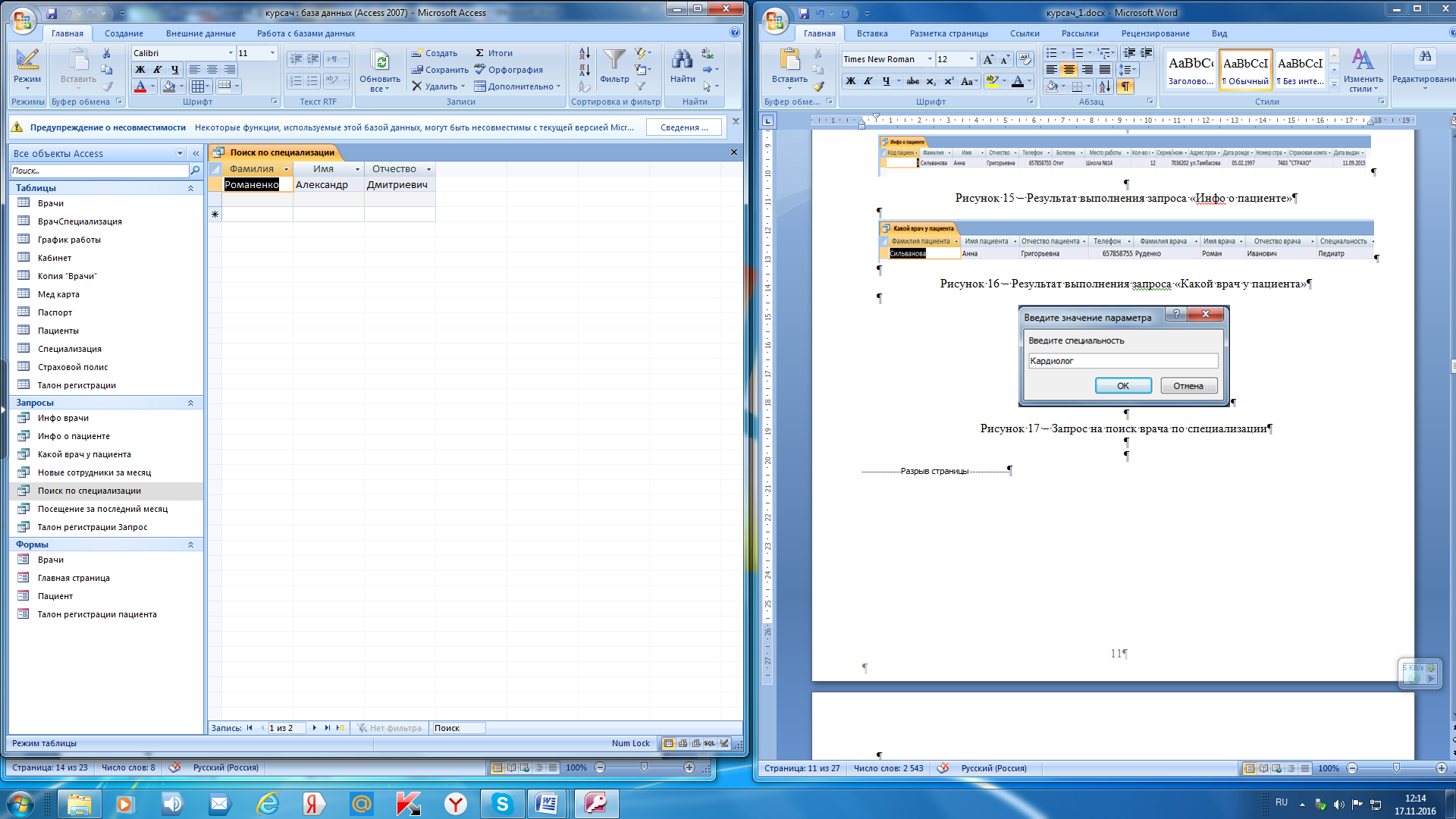


Рисунок 10 – Результат выполнения запроса «Поиск по специализации»

В дальнейшем запросы помогут нам создавать более подробные формы.

## 4.3 Формы, представленные в базе данных

Форма – это объект базы данных, который можно использовать для создания интерфейса пользователя для приложения базы данных. "Привязанная" форма напрямую соединена с источником данных, например, с таблицей или запросом, и может использоваться для ввода, изменения или отображения данных из источника данных. Как вариант, можно создать "свободную" форму, которая не связана напрямую с источником данных, но которая может содержать кнопки, надписи и другие элементы управления, необходимые для работы приложения [5].

Форма «Главная страница» (рисунок 11) открывается с запуском Access. С помощью главной страницы мы можем путешествовать по элементам БД, а также совершать выход из нее.

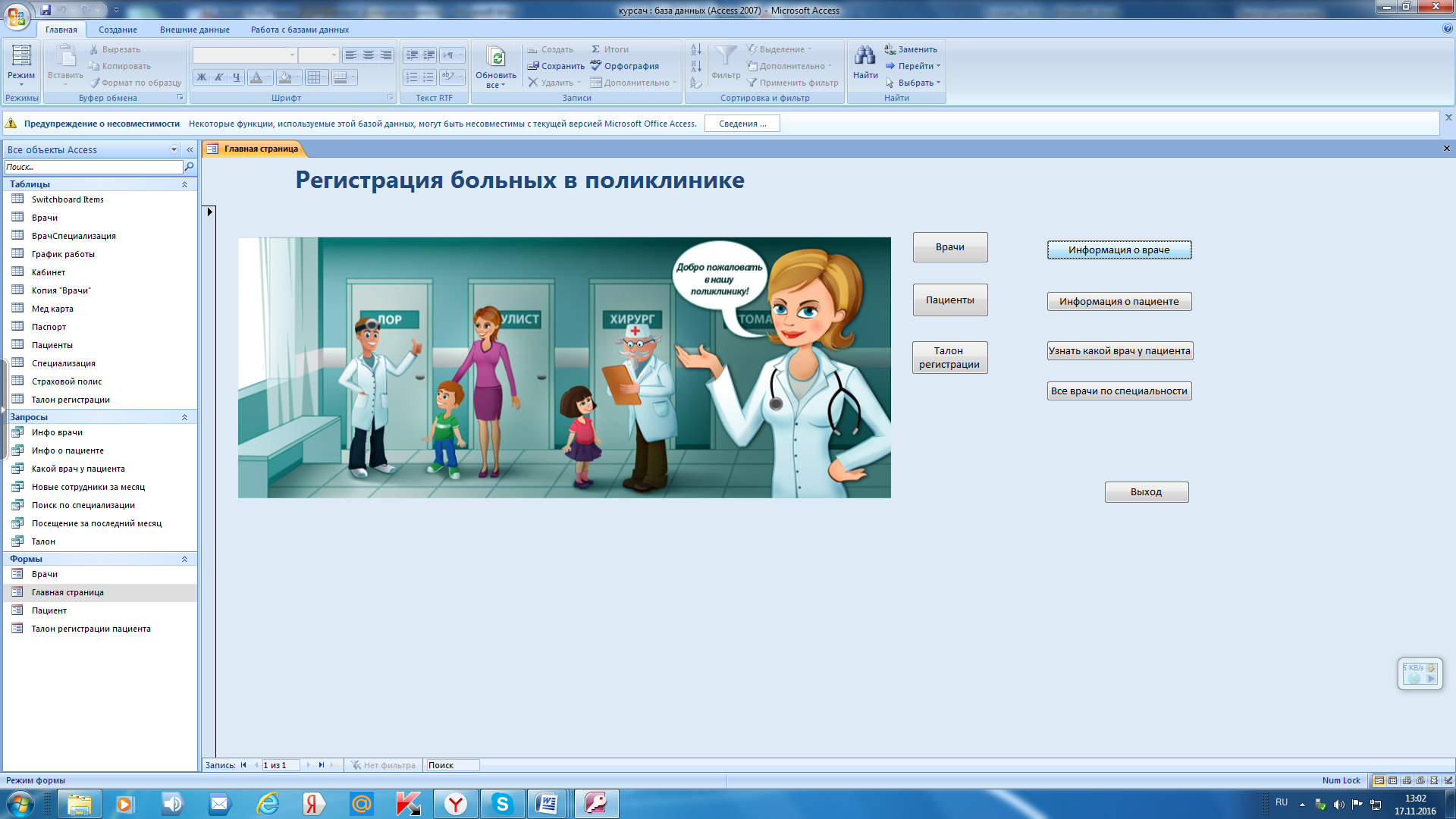


Рисунок 11 – Форма «Главная страница»

На этой форме представлены кнопки, позволяющие открывать формы и ранее созданные запросы.

При помощи кнопки «Врачи» можно открыть форму, содержащую и позволяющую вносить сведения о врачах. Данная форма представлена на рисунке 12.

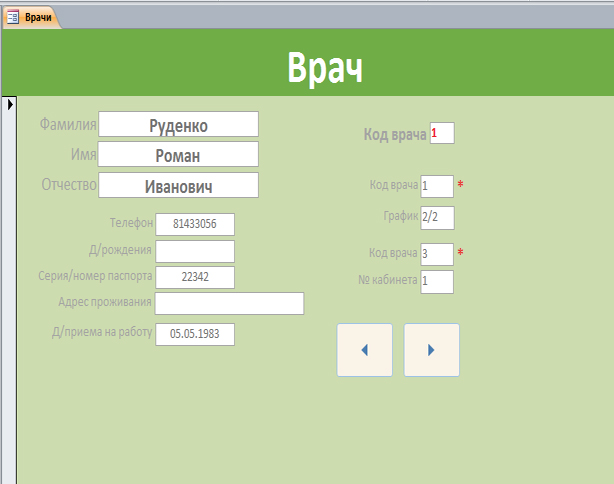


Рисунок 12– Форма «Врач»

Для удобства навигации на этой и на следующей формах созданы кнопки, помогающие передвигаться по записям: следующая / предыдущая.

Форма «Пациент» (рисунок 13) также открывается при помощи кнопки на главной форме.

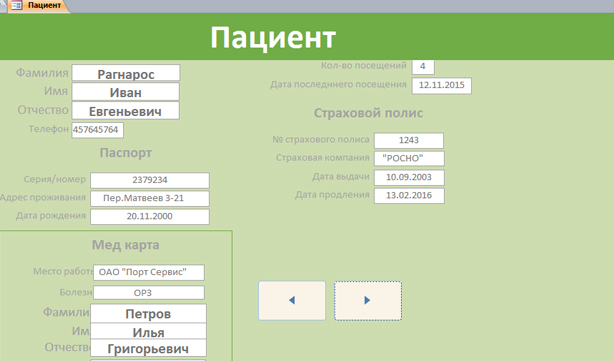


Рисунок 13 – Форма «Пациент»

Перед тем, как открыть форму «Талон регистрации», нам потребуется сначала ввести номер талона регистрации в диалоговом окне, например, номер 2, как показано на рисунке 14.

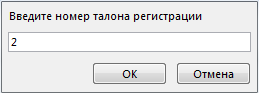


Рисунок 14 – Запрос на ввод номера талона регистрации

После этого откроется форма, содержащая сведения о пациенте и лечащем враче для указанного номера талона регистрации (рисунок 15).

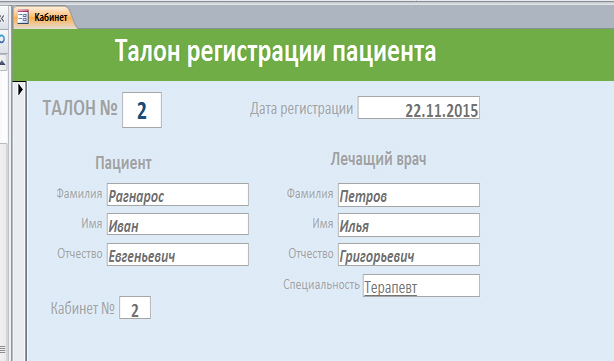


Рисунок 15 – Форма «Талон регистрации»

## 4.4 Отчеты, представленные в базе данных

При построении базы данных были реализованы отчеты:

* отчет «Врачи»;
* отчет «Пациенты»;
* отчет «Посещение за последний месяц».

Данные отчеты, в основном, дублируют запросы и выводят информацию в более удобном виде без возможности редактирования, что немаловажно, когда базой данных пользуется пациент.

Примеры некоторых из приведенных выше отчетов представлены на рисунках 16 – 18.

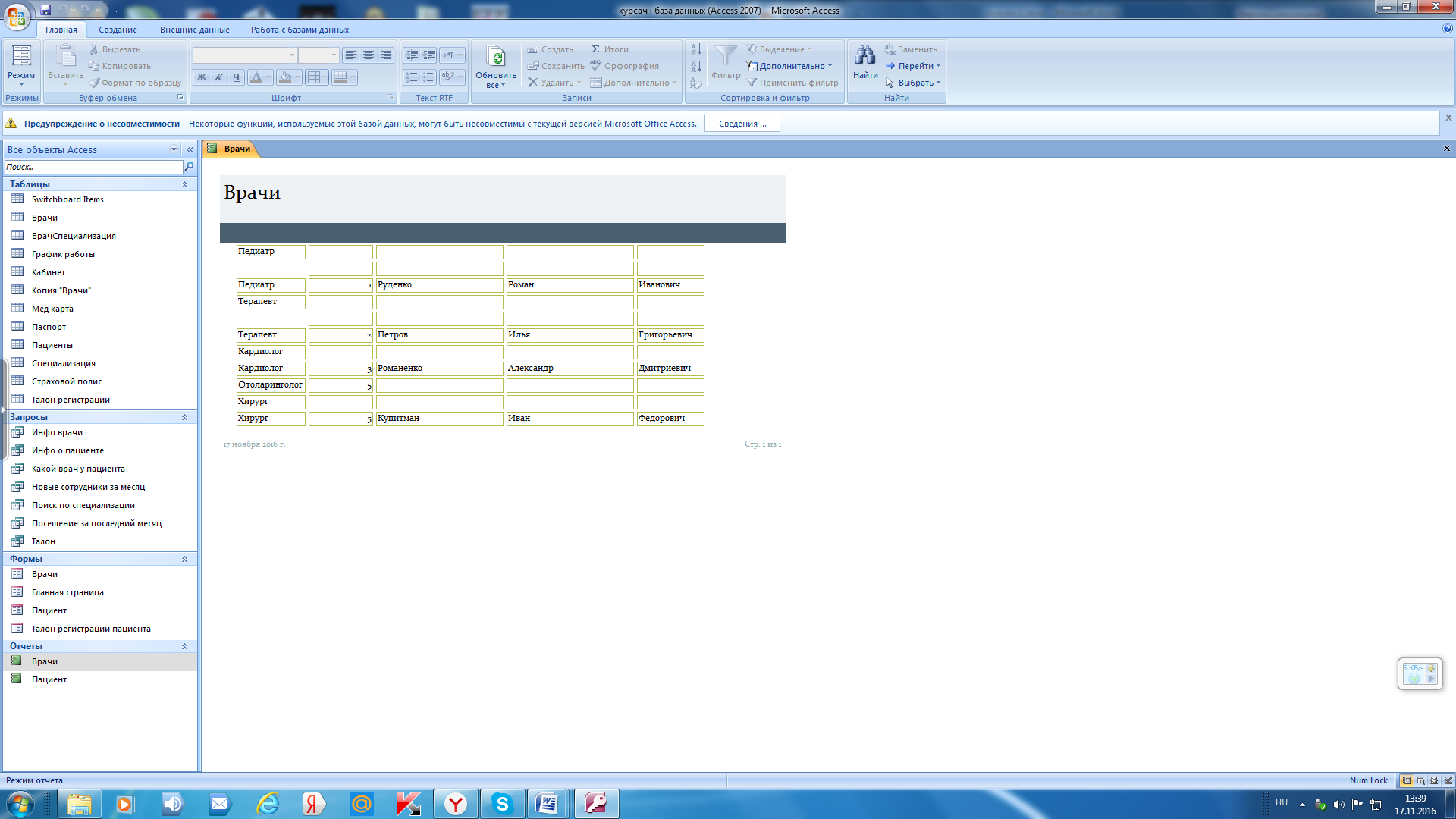


Рисунок 16 – Отчет «Врачи»

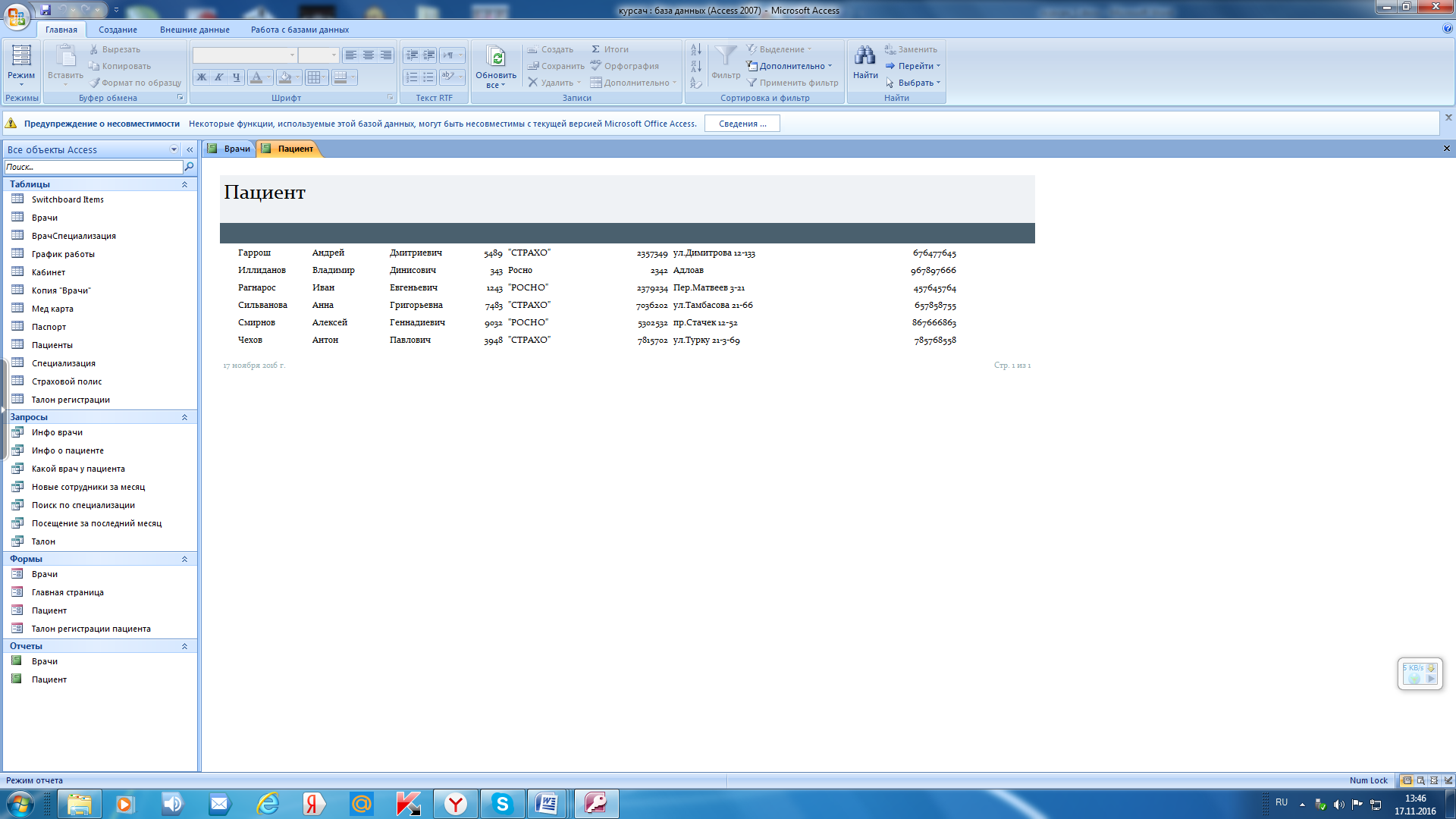


Рисунок 17 – Отчет «Пациенты»

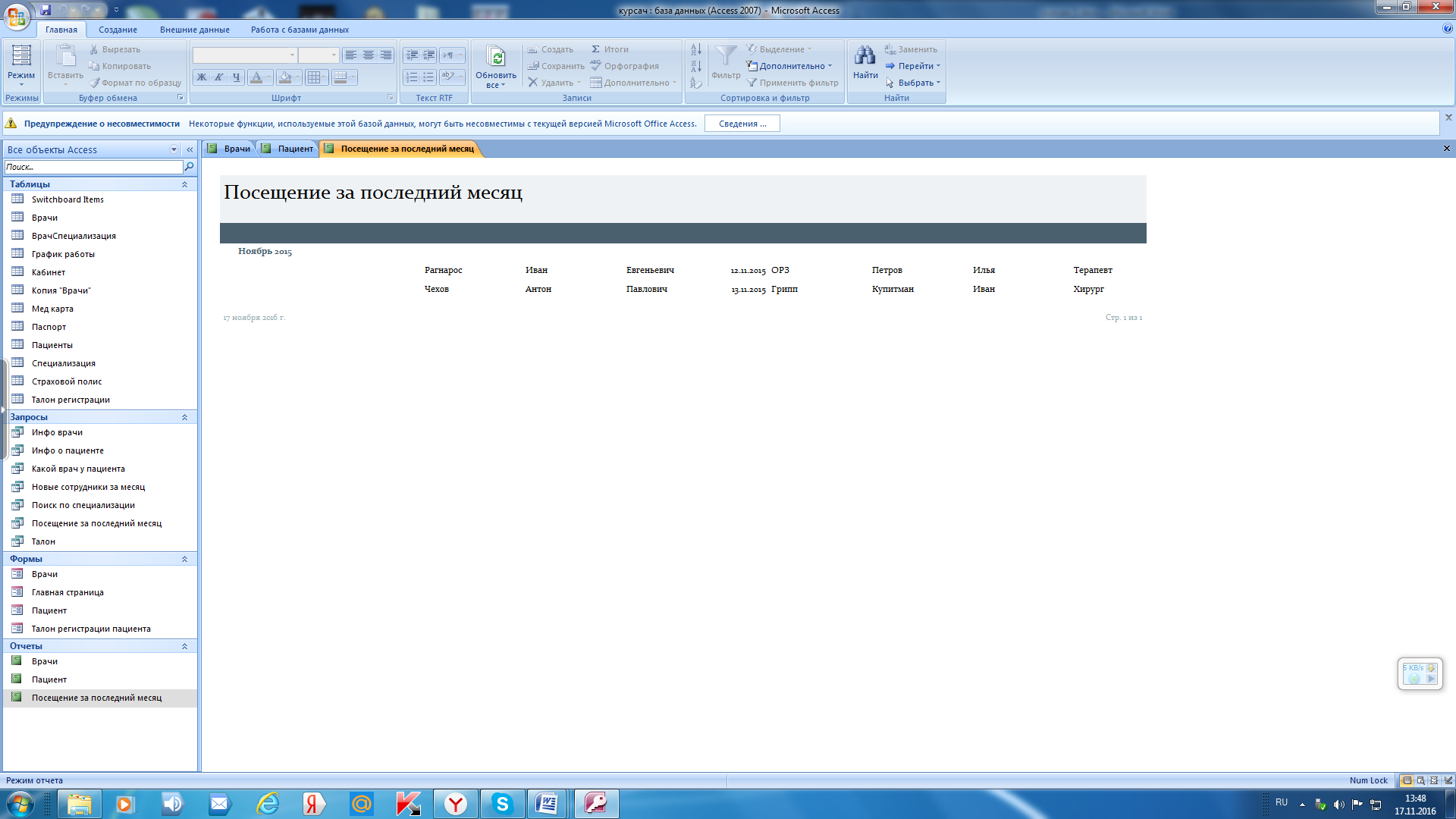


Рисунок 18 – Отчет «Посещение за последний месяц»

## 4.5 Защита данных средствами системы управления базами данных Для ограничения доступа нежелательных лиц к ресурсам базы данных реализован процесс верификации путем введения пароля для получения доступа к главной форме.

Пароль выводится при запуске главной кнопочной формы (рисунок 19). При введении корректного пароля открывается соответствующая форма. При введении некорректного пароля, либо при отказе от ввода пароля база данных будет немедленно закрыта.

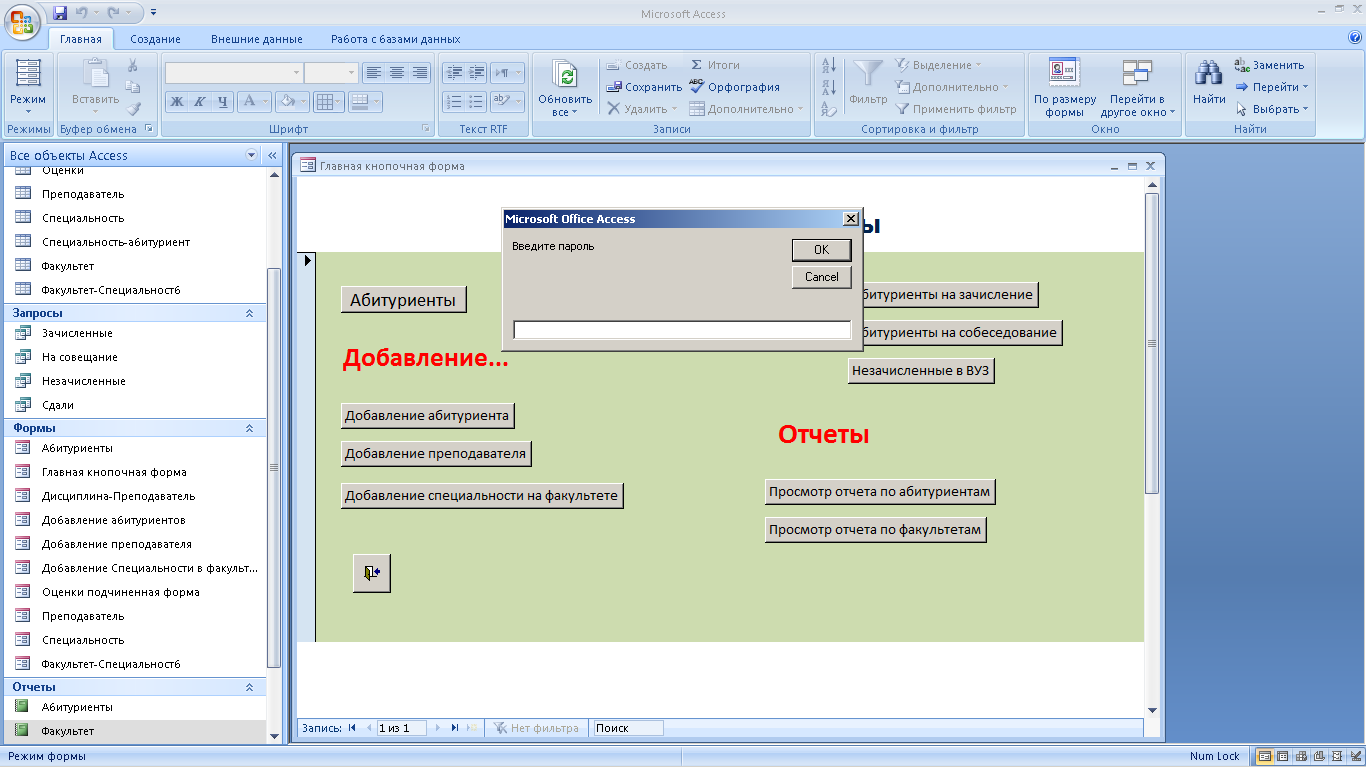


Рисунок 19 – Окно ввода пароля

Код процесса ввода пароля на языке VBA представлен в приложении Б.

# ЗАКЛЮЧЕНИЕ

Результатом курсовой работы является реализованная база данных «Регистрация больных в поликлинике». Она предназначена для автоматизации работы сотрудников регистрационного стола.

В базе данных «Регистрация больных в поликлинике» созданы следующие элементы: таблицы, запросы, формы, отчеты. Формы используются для просмотра данных, хранящихся в таблицах. Отчеты используются для удобства просмотра информации, которую полностью не видно в таблицах. Запросы используются для поиска сведений о конкретных врачах и пациентах, для вывода информации по талонам регистрации. Кнопочная форма объединяет основные возможности проекта для более удобной работы с ним.

Разработанная база данных способна без каких-либо проблем и затрат адаптироваться к постоянно меняющимся условиям работы и меняющемуся законодательству. В перспективе возможны доработка и развитие созданной БД путем внедрения новых возможностей.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Бекаревич, Ю. Б. Самоучитель Microsoft Access 2013 / Ю. Б. Бекаревич, Н. В. Пушкина. – СПб.: БХВ-Петербург, 2014. – 464 с.

2. Пирогов, В. Ю. Информационные системы и базы данных: организация и проектирование / В. Ю. Пирогов. – СПб.: БХВ-Петербург, 2009. – 528 с.

3. Карпова, Т. С. Базы данных: модели, разработка, реализация: учеб. пособие / Т. С. Карпова. – СПб.: ПИТЕР, 2002. – 304 с.

4. Туманов, В. Е. Основы проектирования реляционных баз данных: учеб. пособие / В. Е. Туманов. – М.: Бином, 2011. – 420 с.

5. Семененко, Т. В. Создание баз данных в среде MS Access: методические указания к выполнению лабораторных работ / Т. В. Семененко. – СПб.: изд-во ГУАП, 2014. – 94 с.

# ПРИЛОЖЕНИЕ А

# SQL-коды запросов

SQL-код запроса «Инфо врачи»

SELECT Врачи.Фамилия, Врачи.Имя, Врачи.Отчество, Специализация.Специальность, Врачи.Телефон, Врачи.[Дата рождения], Врачи.[Серия/номер паспорта], [График работы].График, Кабинет.[Номер кабинета]

FROM [График работы] INNER JOIN ((Кабинет INNER JOIN Врачи ON Кабинет.[Код кабинета] = Врачи.[Код кабинета]) INNER JOIN (Специализация INNER JOIN ВрачСпециализация ON Специализация.[Код специализации] = ВрачСпециализация.[Код Специализация]) ON Врачи.[Код врач] = ВрачСпециализация.[Код Врач]) ON [График работы].[Код график работы] = Врачи.[Код график работы]

WHERE (((Врачи.Фамилия)=[Введите фамилию врача]));

SQL-код запроса «Инфо о пациенте»

SELECT Пациенты.[Код пациент], Пациенты.Фамилия, Пациенты.Имя, Пациенты.Отчество, Пациенты.Телефон, [Мед карта].Болезнь, [Мед карта].[Место работы], [Мед карта].[Кол-во посещений], Паспорт.[Серия/номер], Паспорт.[Адрес проживания], Паспорт.[Дата рождения], [Страховой полис].[Номер страхового полиса], [Страховой полис].[Страховая компания], [Страховой полис].[Дата выдачи]

FROM [Страховой полис] INNER JOIN (Паспорт INNER JOIN ([Мед карта] INNER JOIN Пациенты ON [Мед карта].[Код мед карты] = Пациенты.[Код мед карты]) ON Паспорт.[Код паспорта] = Пациенты.[Код паспорта]) ON [Страховой полис].[Код страхового полиса] = Пациенты.[Код страхового полиса]

WHERE (((Пациенты.Фамилия)=[Введите фамилию пациента]));

SQL-код запроса «Какой врач у пациента»

SELECT Пациенты.Фамилия AS [Фамилия пациента], Пациенты.Имя AS [Имя пациента], Пациенты.Отчество AS [Отчество пациента], Пациенты.Телефон, Врачи.Фамилия AS [Фамилия врача], Врачи.Имя AS [Имя врача], Врачи.Отчество AS [Отчество врача], Специализация.Специальность

FROM Специализация INNER JOIN ((Врачи INNER JOIN (Пациенты INNER JOIN [Талон регистрации] ON Пациенты.[Код пациент]=[Талон регистрации].[Код пациент]) ON (Врачи.[Код врач]=[Талон регистрации].[Код врач]) AND (Врачи.[Код врач]=Пациенты.[Код врач])) INNER JOIN ВрачСпециализация ON Врачи.[Код врач]=ВрачСпециализация.[Код Врач]) ON Специализация.[Код специализации]=ВрачСпециализация.[Код Специализация]

WHERE (((Пациенты.Фамилия)=[Введите фамилию пациента]));

SQL-код запроса «Поиск по специализации»

SELECT Врачи.Фамилия, Врачи.Имя, Врачи.Отчество

FROM Врачи INNER JOIN (Специализация INNER JOIN ВрачСпециализация ON Специализация.[Код специализации] = ВрачСпециализация.[Код Специализация]) ON Врачи.[Код врач] = ВрачСпециализация.[Код Врач]

WHERE (((Специализация.Специальность)=[Введите специальность]));

SQL-код запроса «Посещение за последний месяц»

SELECT Пациенты.Фамилия, Пациенты.Имя, Пациенты.Отчество, Пациенты.[Дата последннего посещения], [Мед карта].Болезнь, Врачи.Фамилия AS [Фамилия врача], Врачи.Имя AS [Имя врача], Специализация.Специальность

FROM (Врачи INNER JOIN (Специализация INNER JOIN ВрачСпециализация ON Специализация.[Код специализации] = ВрачСпециализация.[Код Специализация]) ON Врачи.[Код врач] = ВрачСпециализация.[Код Врач]) INNER JOIN (([Мед карта] INNER JOIN Пациенты ON [Мед карта].[Код мед карты] = Пациенты.[Код мед карты]) INNER JOIN [Талон регистрации] ON Пациенты.[Код пациент] = [Талон регистрации].[Код пациент]) ON Врачи.[Код врач] = [Талон регистрации].[Код врач]

WHERE (((Пациенты.[Дата последннего посещения])>=Date()-30 And (Пациенты.[Дата последннего посещения])<=Date()));

SQL-код запроса «Талон»

SELECT [Талон регистрации].[Номер талона регистрации], Пациенты.Фамилия, Пациенты.Имя, Пациенты.Отчество, [Талон регистрации].[Дата регистрации], Кабинет.[Номер кабинета] AS [№ кабинета], Врачи.Фамилия AS [Фамилия врача], Врачи.Имя AS [Имя врача], Врачи.Отчество AS [Отчество врача], Специализация.Специальность

FROM Специализация INNER JOIN (Кабинет INNER JOIN ((Врачи INNER JOIN (Пациенты INNER JOIN [Талон регистрации] ON Пациенты.[Код пациент] = [Талон регистрации].[Код пациент]) ON (Врачи.[Код врач] = [Талон регистрации].[Код врач]) AND (Врачи.[Код врач] = Пациенты.[Код врач])) INNER JOIN ВрачСпециализация ON Врачи.[Код врач] = ВрачСпециализация.[Код Врач]) ON (Кабинет.[Код кабинета] = Врачи.[Код кабинета]) AND (Кабинет.[Код кабинета] = Пациенты.[Код кабинета])) ON Специализация.[Код специализации] = ВрачСпециализация.[Код Специализация]

WHERE ((([Талон регистрации].[Номер талона регистрации])=[Введите номер талона регистрации]));

# ПРИЛОЖЕНИЕ Б

# Код процесса ввода пароля

Private Sub ОбластьДанных\_Click()

Dim pass As String

pass = InputBox("Введите пароль")

If pass = "123" Then

MsgBox "Пароль верный"

End If

If pass <> "123" Then

MsgBox "Пароль неверный"

DoCmd.CloseDatabase

End If

End Sub