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 популярных алгоритмов поиска решений ТЗ.

Объектом исследования является транспортная задача с постановкой в сетевой форме.

Предметом исследования является применение алгоритма для решения

1. Симплекс-метод
2. Метод потенциалов

Линейное программирование – математическая дисциплина и набор методов, используемых в математическом программировании. Эти методы созданы и используются для решения задач планирования, решение которых классическими методами невозможно.

Для нахождения решения транспортной задачи среди методов линейного программирования существует несколько алгоритмов.

Для принятия обоснованных и эффективных решений в производственной деятельности, в управлении экономикой и в политике современный специалист должен уметь с помощью компьютеров и средств связи получать, накапливать, хранить и обрабатывать данные, представляя результат в виде наглядных документов. В современном обществе информационные технологии развиваются очень стремительно, они проникают во все сферы человеческой деятельности.

В разных областях экономики зачастую приходится работать с данными из разных источников, каждый из которых связан с определенным видом деятельности. Для координации всех этих данных необходимы определенные знания и организационные навыки.

Продукт корпорации Microsoft Access объединяет сведения из разных источников в одной реляционной базе данных. Создаваемые в нем формы, запросы и отчеты позволяют быстро и эффективно обновлять данные, получать ответы на вопросы, осуществлять поиск нужных данных, анализировать данные, печатать отчеты, диаграммы и почтовые наклейки.

Целью данной курсовой работы является рассмотрение проектирования в теории и создания на практике базы данных в продукте корпорации Microsoft для управления базами данных Microsoft Access.

Система Access — это набор инструментов конечного пользователя для управления базами данных [1]. В ее состав входят конструкторы таблиц, форм, запросов и отчетов. Эту систему можно рассматривать и как среду разработки приложений. Используя макросы или модули для автоматизации решения задач, можно создавать ориентированные на пользователя приложения такими же мощными, как и приложения, написанные непосредственно на языках программирования. При этом они будут включать кнопки, меню и диалоговые окна.

В Access в полной мере реализовано управление реляционными базами данных. Система поддерживает первичные и внешние ключи и обеспечивает целостность данных на уровне ядра (что предотвращает несовместимые операции обновления или удаления данных). Кроме того, таблицы в Access снабжены средствами проверки допустимости данных, предотвращающими некорректный ввод вне зависимости от того, как он осуществляется, а каждое поле таблицы имеет свой формат и стандартные описания, что существенно облегчает ввод данных. Access поддерживает все необходимые типы полей, в том числе текстовый, числовой, счетчик, денежный, дата/время, MEMO, логический, гиперссылка и поля объектов OLE. Если в процессе специальной обработки в полях не оказывается никаких значений, система обеспечивает полную поддержку пустых значений.

# 1 Требования к реляционной базе данных

Базой данных (БД) называется организованная в соответствии с определенными правилами и поддерживаемая в памяти компьютера совокупность сведений об объектах, процессах, событиях или явлениях, относящихся к некоторой предметной области, теме или задаче [2]. Она организована таким образом, чтобы обеспечить информационные потребности пользователей, а также удобное хранение этой совокупности данных, как в целом, так и любой ее части.

Реляционные базы данных состоят из таблиц. Каждая таблица состоит из столбцов (их называют полями или атрибутами) и строк (их называют записями или кортежами). Таблицы в реляционных базах данных обладают рядом свойств.

* В таблице не может быть двух одинаковых строк. В математике таблицы, обладающие таким свойством, называют отношениями-по-английски relation, отсюда и название – реляционные.
* Столбцы располагаются в определенном порядке, который создается при создании таблицы. В таблице может не быть ни одной строки, но обязательно должен быть хотя бы один столбец.
* У каждого столбца есть уникальное имя (в пределах таблицы), и все значения в одном столбце имеют один тип (число, текст, дата...).
* На пересечении каждого столбца и строки может находиться только атомарное значение (одно значение, не состоящее из группы значений). Таблицы, удовлетворяющие этому условию, называют нормализованными.

Любая система управления базами данных (СУБД) позволяет выполнять следующие операции с данными:

* добавление записей в таблицы;
* удаление записей из таблицы;
* обновление значений некоторых полей в одной или нескольких записях в таблицах БД;
* поиск одной или нескольких записей, удовлетворяющих заданному условию.

Данная работа посвящена созданию реляционной базы данных «Регистрация больных в поликлинике».

База данных должна быть удобной для просмотра, получения общей и конкретной информации любому пользователю: как опытному, так и новичку. Вся информация не должна быть раскидана по БД, поэтому была создана главная страница с кнопками, ссылками на таблицы, формы.

Чтобы получить определенные данные, пользователю не нужно будет искать информацию во всех таблицах. Для удобства должны быть созданы запросы. Например, если пользователь хочет найти информацию определенного номера талона регистрации и узнать, кто является носителем данного талона, ему потребуется всего лишь нажать на кнопку запроса «Талон» и далее ввести в диалоговом окне интересующий номер.

Если же управляющему понадобится найти информацию о своих врачах, ему не потребуется собирать всю информацию, расположенную в разных таблицах. Это действие можно выполнить, нажав на запрос «Врачи», и далее ввести фамилию интересующего врача.

# 2 Проектирование системы управления базами данных-независимого концептуального представления данных

На этом этапе проектирования используется модель сущность-связь.

Модель сущность-связь (ER-модель) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) entity-relationship model, ERM) – [модель данных](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85), позволяющая описывать [концептуальные схемы](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%86%D0%B5%D0%BF%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D1%85%D0%B5%D0%BC%D0%B0) [предметной области](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D0%B4%D0%BC%D0%B5%D1%82%D0%BD%D0%B0%D1%8F_%D0%BE%D0%B1%D0%BB%D0%B0%D1%81%D1%82%D1%8C) [3]. С ее помощью можно выделить ключевые сущности и обозначить связи, которые могут устанавливаться между этими сущностями.

В данной предметной области могут быть выделены следующие информационные объекты-сущности: Врачи (фамилия, имя, отчество, адрес), Специализация (специальность), Пациенты (фамилия, имя, отчество, мед карта, страховой полис, паспорт), Кабинет (номер кабинета), График работы (график). Между этими информационными объектами существуют связи.

В ER-модели связи делятся на три типа по множественности:

* один-к-одному;
* один-ко-многим;
* многие-ко-многим.

Связь «один-к-одному» означает, что экземпляр одной сущности связан только с одним экземпляром другой сущности.

Связь «один-ко-многим» означает, что один экземпляр сущности, расположенный слева по связи, может быть связан с несколькими экземплярами сущности, расположенными справа по связи.

Связь «многие-ко-многим» означает, что одни экземпляр первой сущности может быть связан с несколькими экземплярами второй сущности, и наоборот, один экземпляр второй сущности может быть связан с несколькими экземплярами первой сущности.

ER-модель предметной области «Регистрация больных в поликлинике» представлена на рисунке 1.

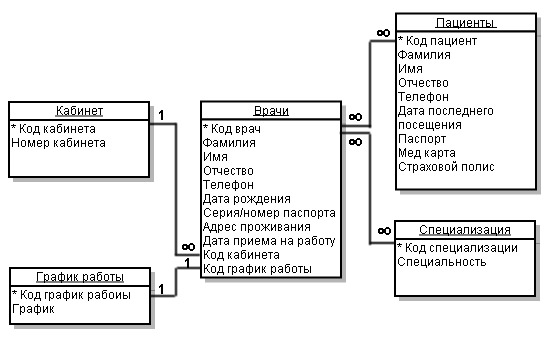


Рисунок 1 – ER-модель рассматриваемой предметной области

Здесь между сущностями Врачи и Пациенты, Врачи и Специализация существуют связи «многие-ко-многим». Между сущностями Кабинет и Врачи существует связь «один-ко-многим» со стороны сущности Кабинет. Между сущностями Врачи и график работы существует связь «один-к-одному».

# 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.

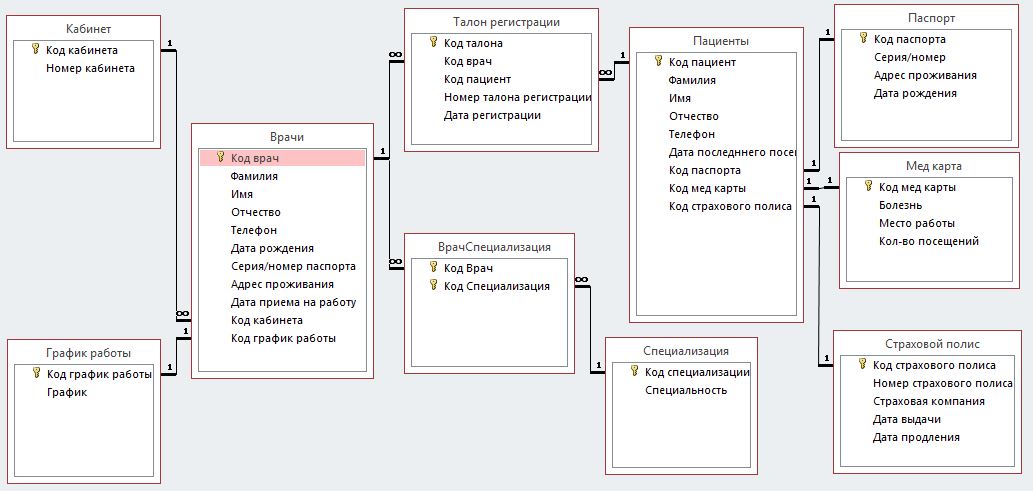


Рисунок 2 – Схема данных

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.

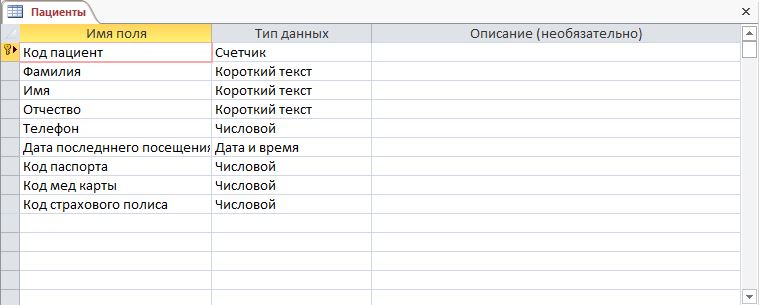


Рисунок3 *–* Вид таблицы «Пациенты» в режиме конструктора

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

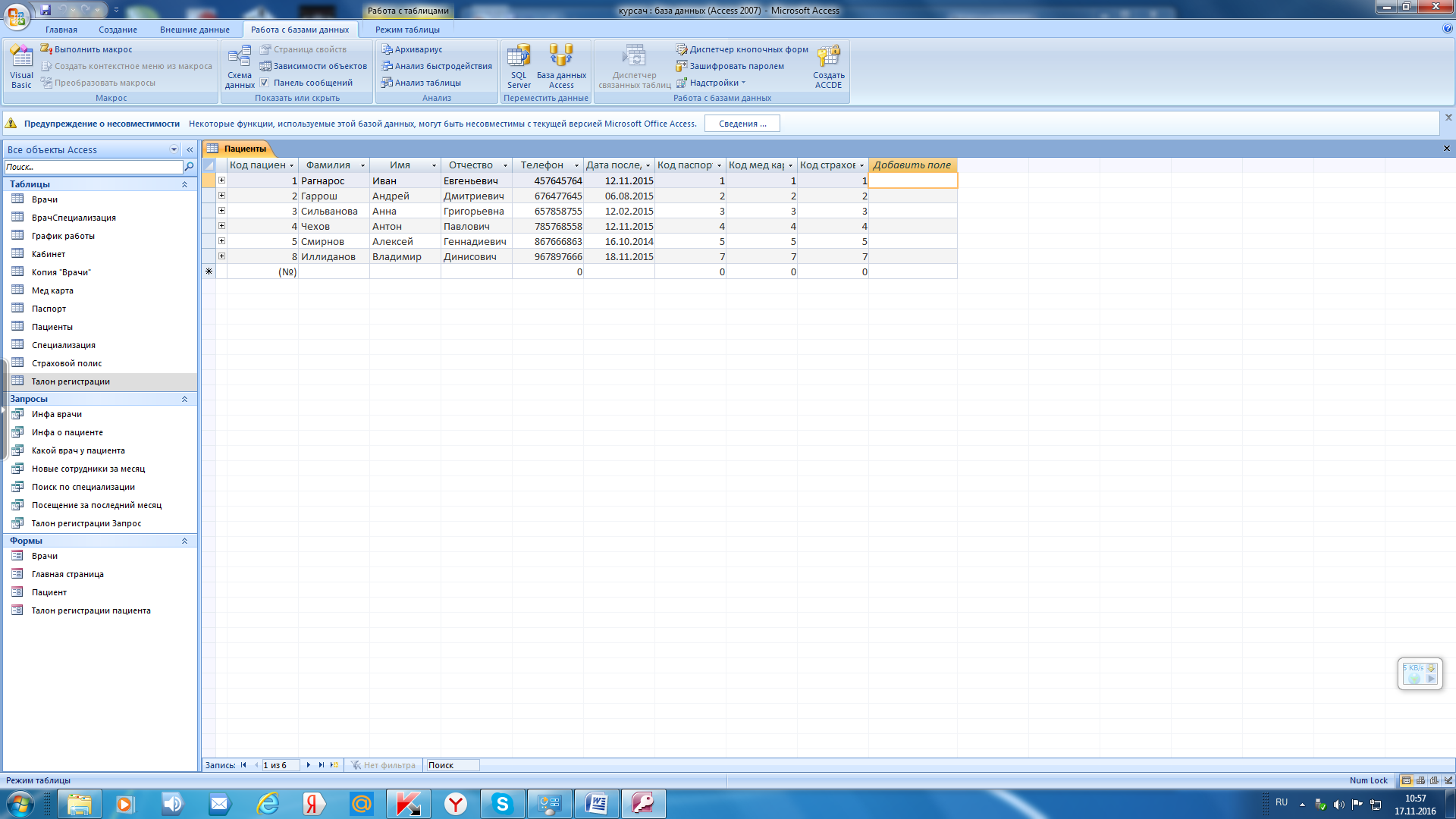


Рисунок 4 – Внешний вид таблицы «Пациенты» в табличном режиме

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

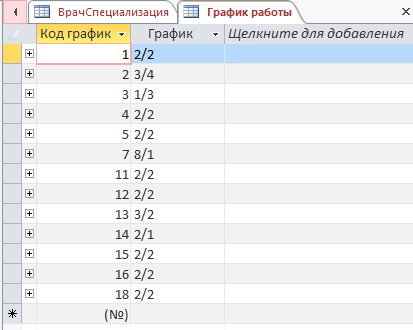


Рисунок 5 – Заполненная таблица «График работы»

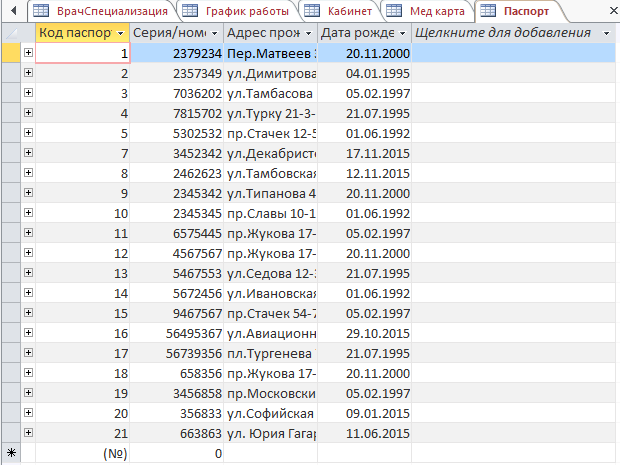


Рисунок 6 – Заполненная таблица «Паспорт»

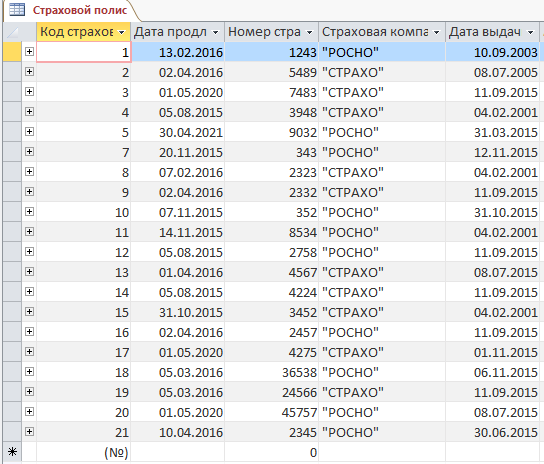


Рисунок 7 – Заполненная таблица «Страховой полис»

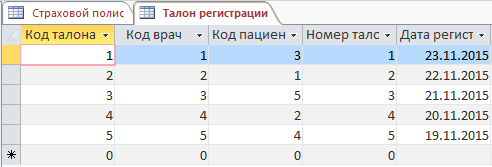


Рисунок 8 – Заполненная таблица «Талон регистрации»

## 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.

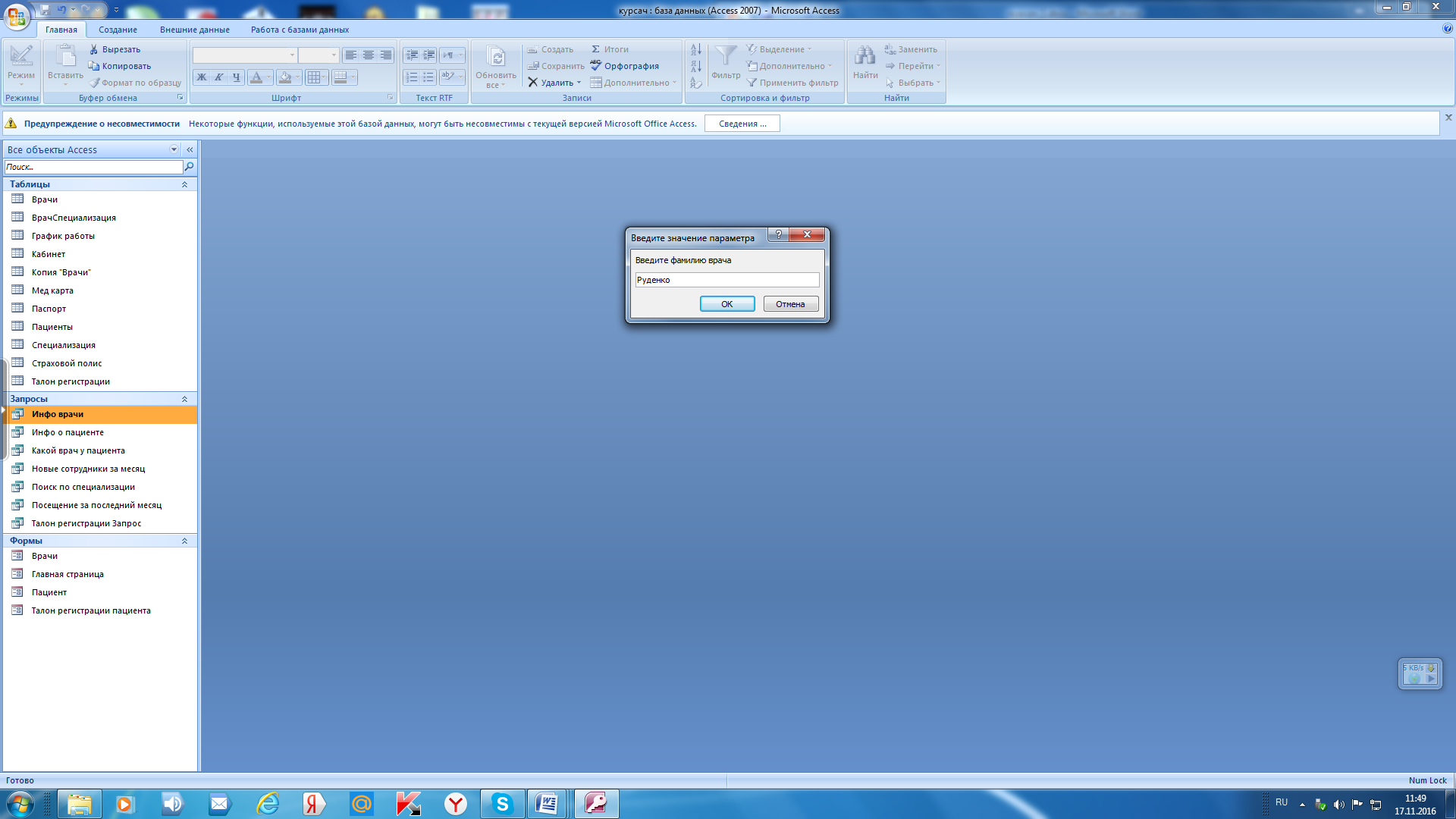


Рисунок 9 – Работа запроса на выборку информации об определенном враче

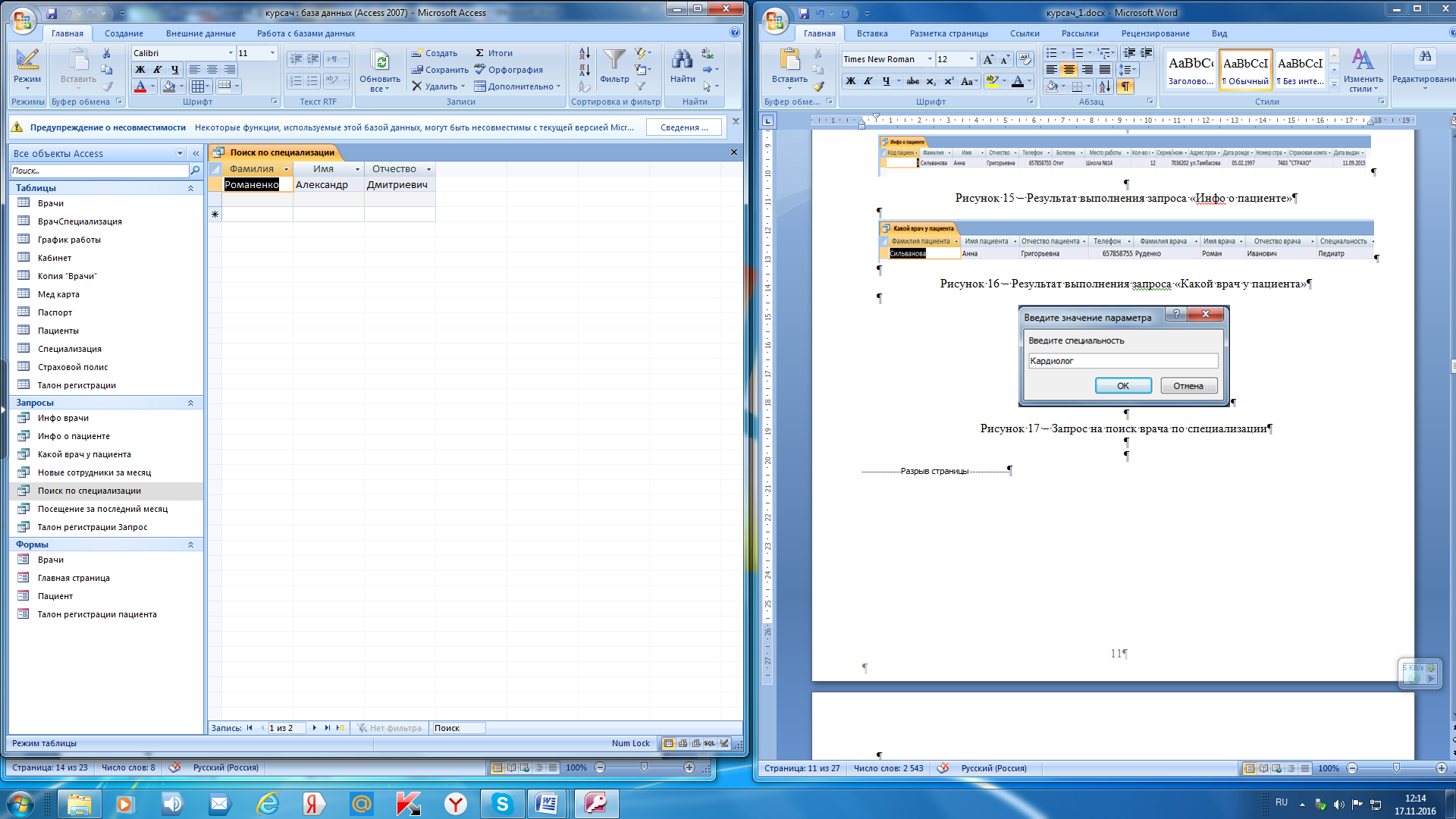


Рисунок 10 – Результат выполнения запроса «Поиск по специализации»

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.

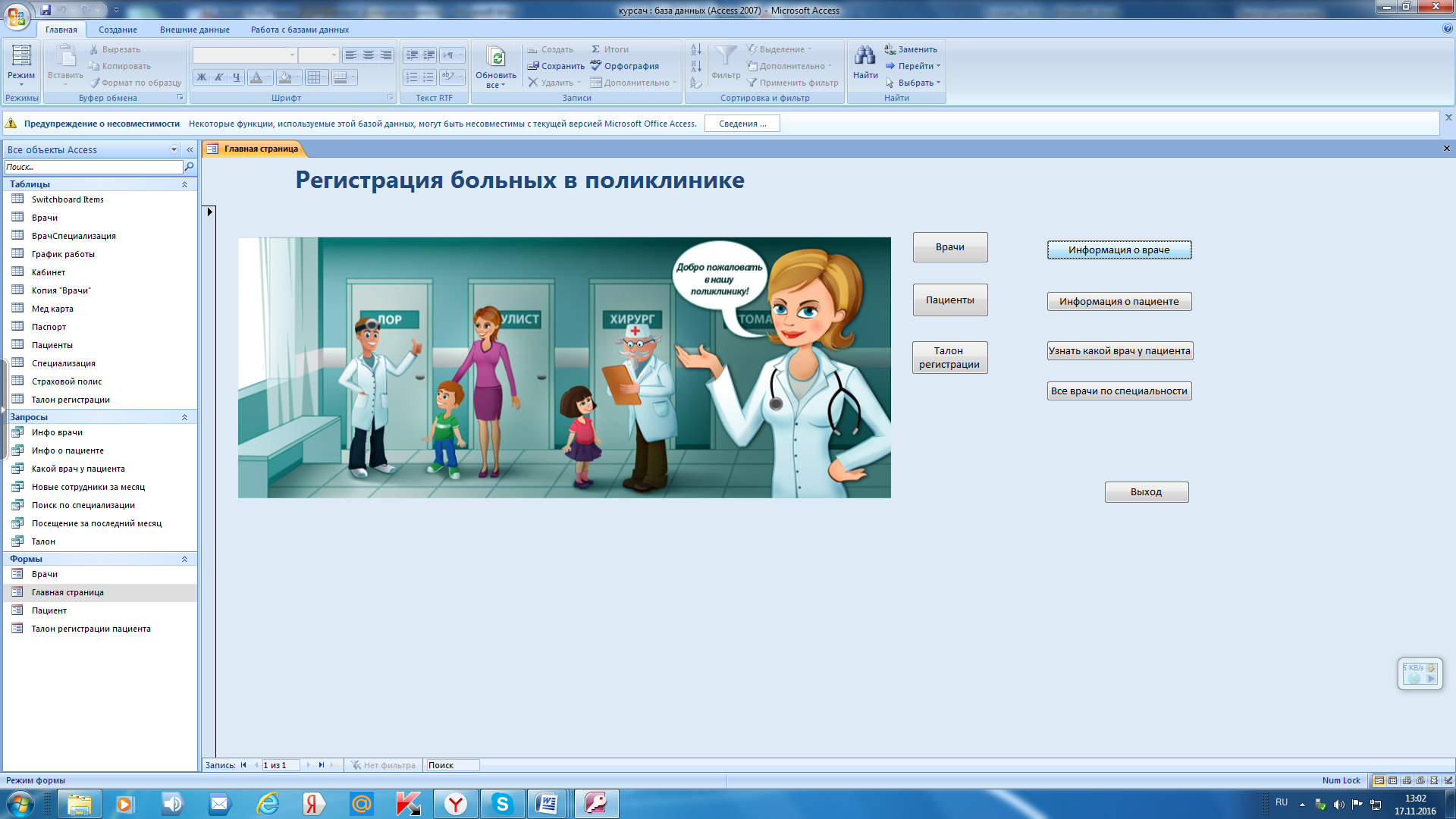


Рисунок 11 – Форма «Главная страница»

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.

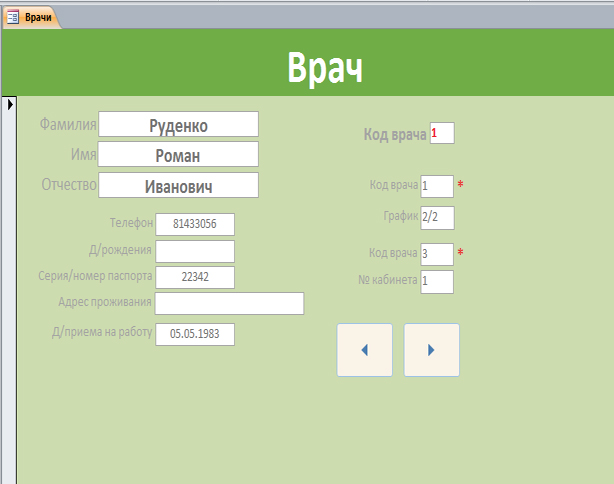


Рисунок 12– Форма «Врач»

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.

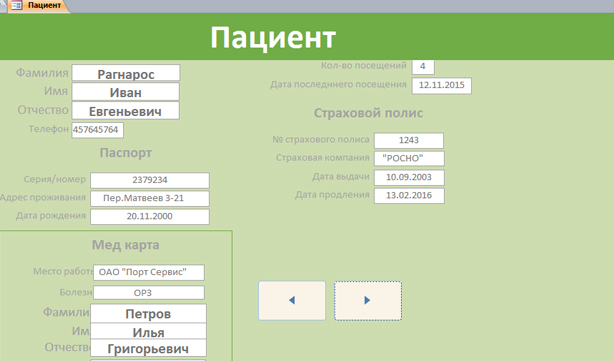


Рисунок 13 – Форма «Пациент»

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.

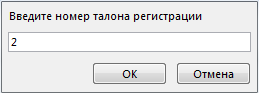


Рисунок 14 – Запрос на ввод номера талона регистрации

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

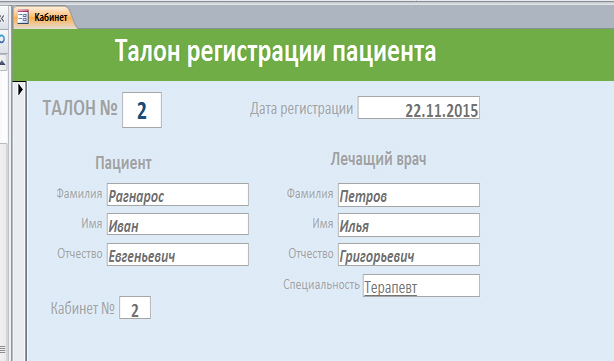


Рисунок 15 – Форма «Талон регистрации»

## 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.

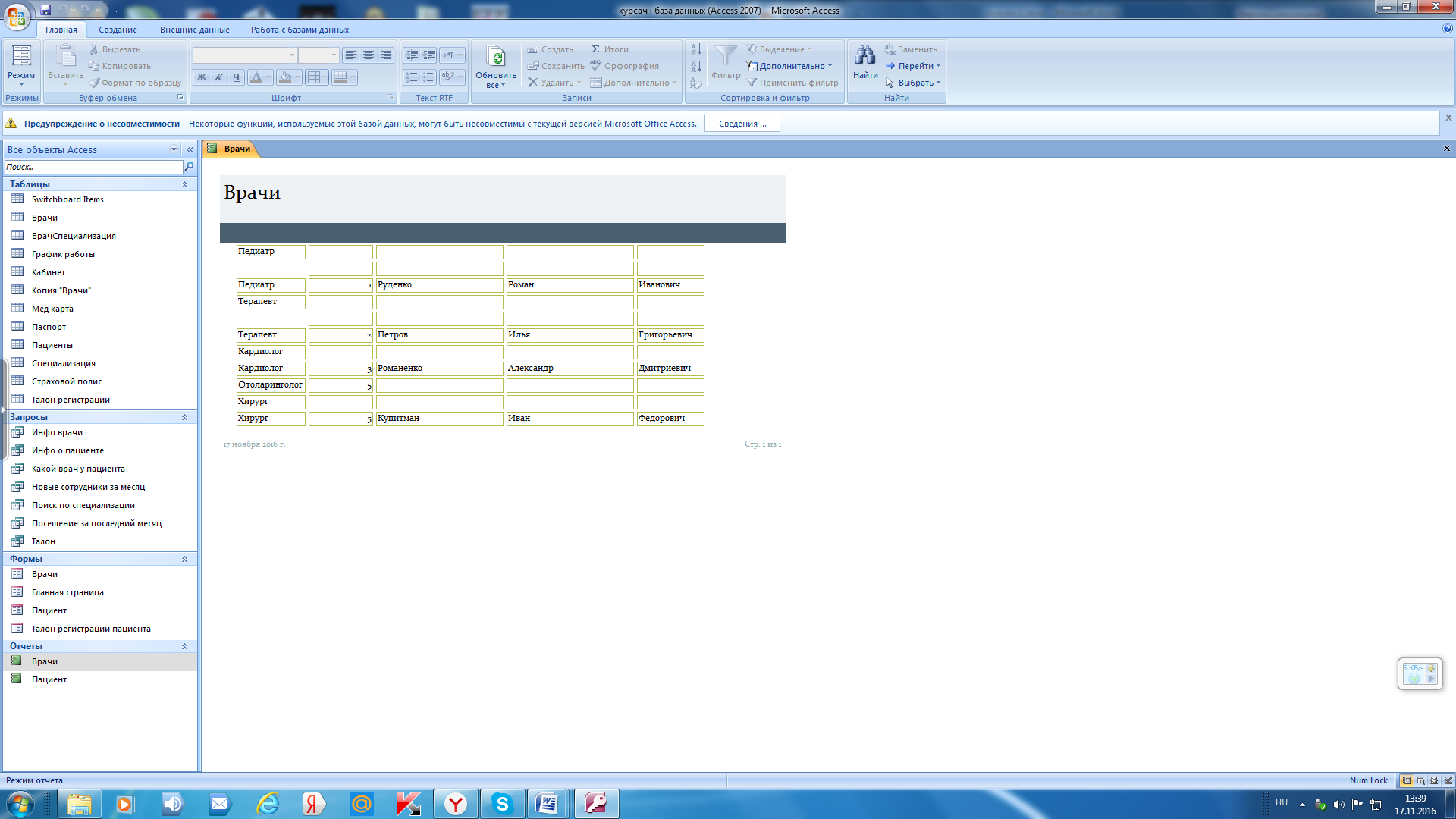


Рисунок 16 – Отчет «Врачи»

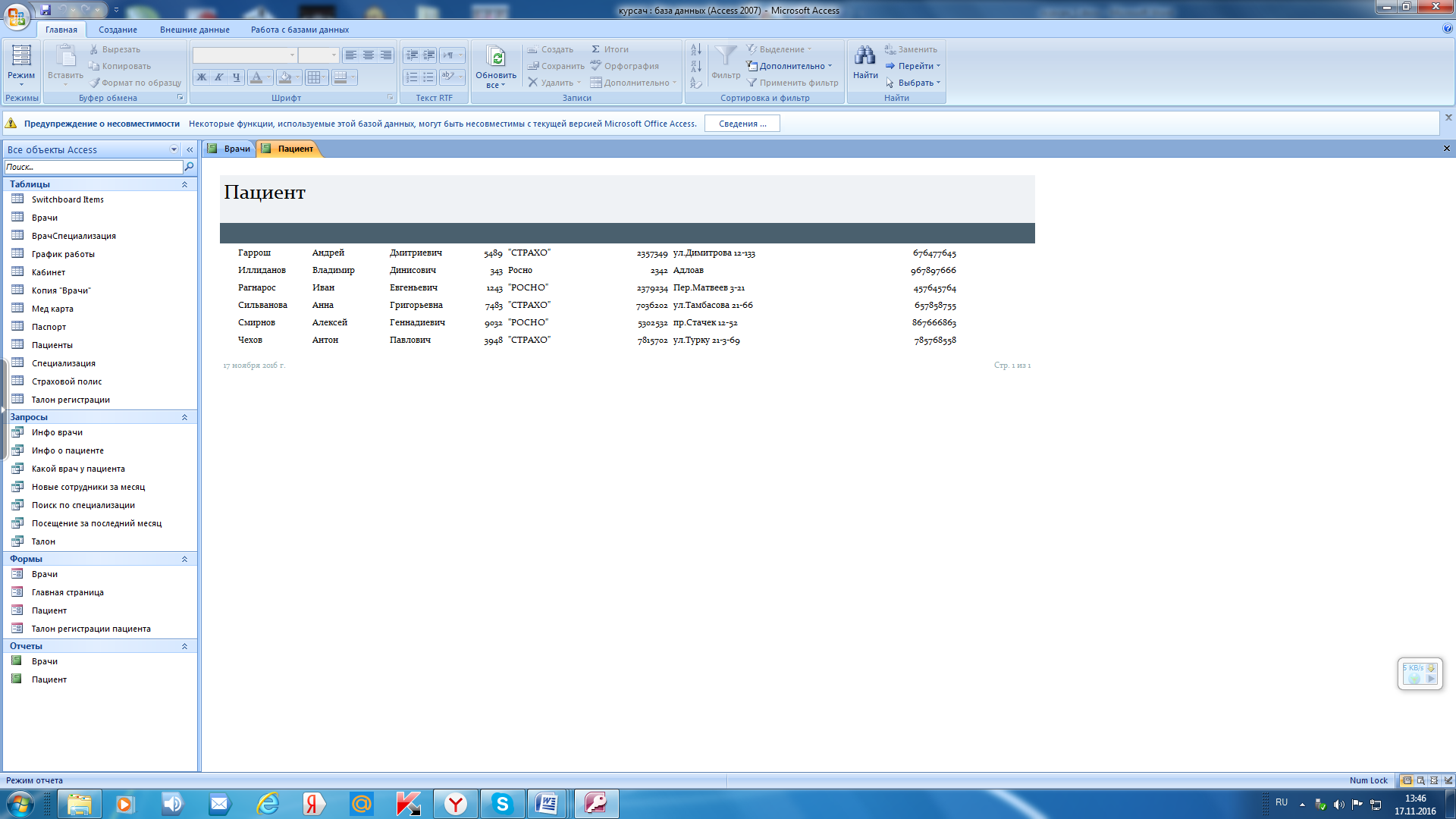


Рисунок 17 – Отчет «Пациенты»

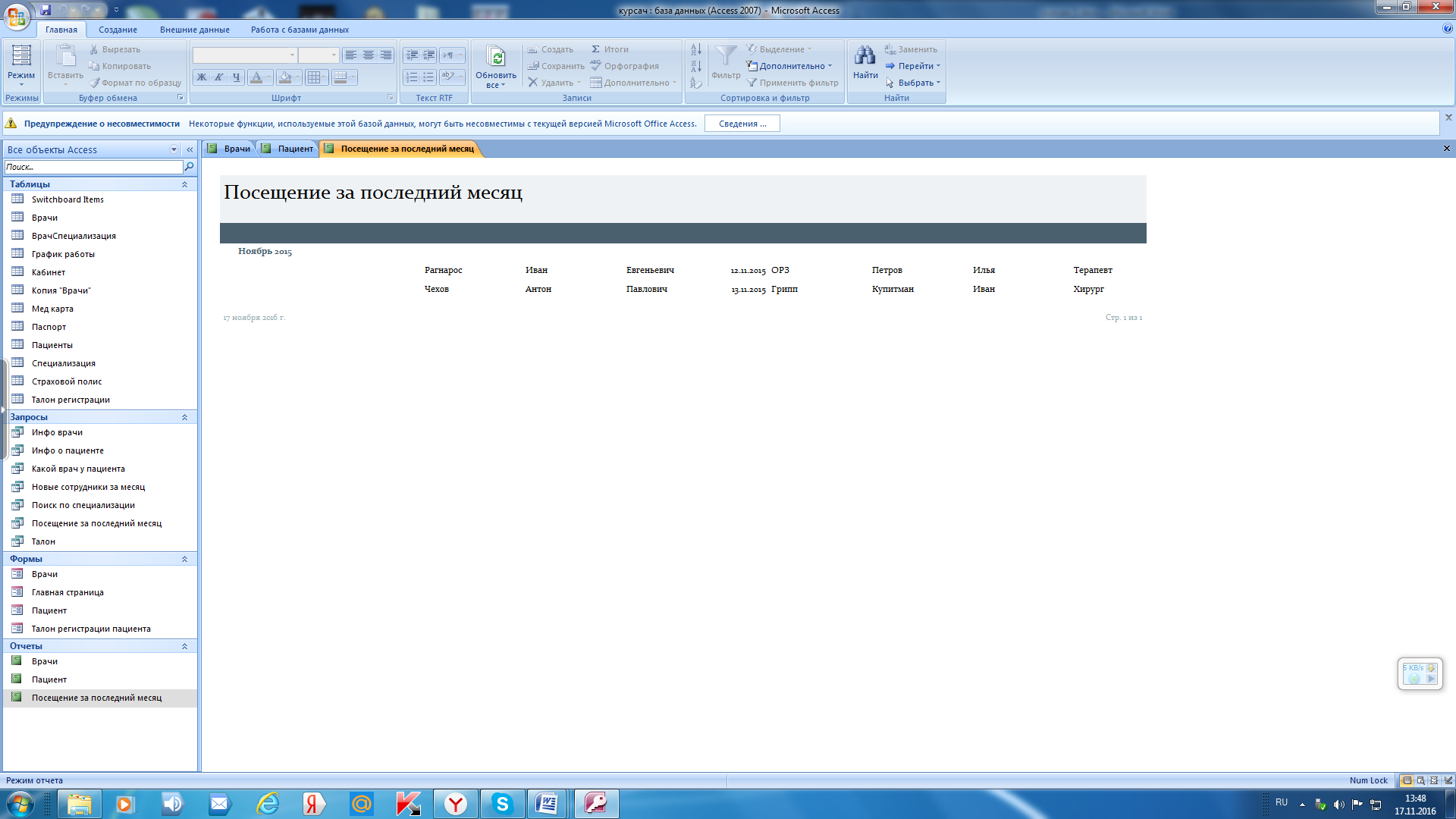


Рисунок 18 – Отчет «Посещение за последний месяц»

## 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.

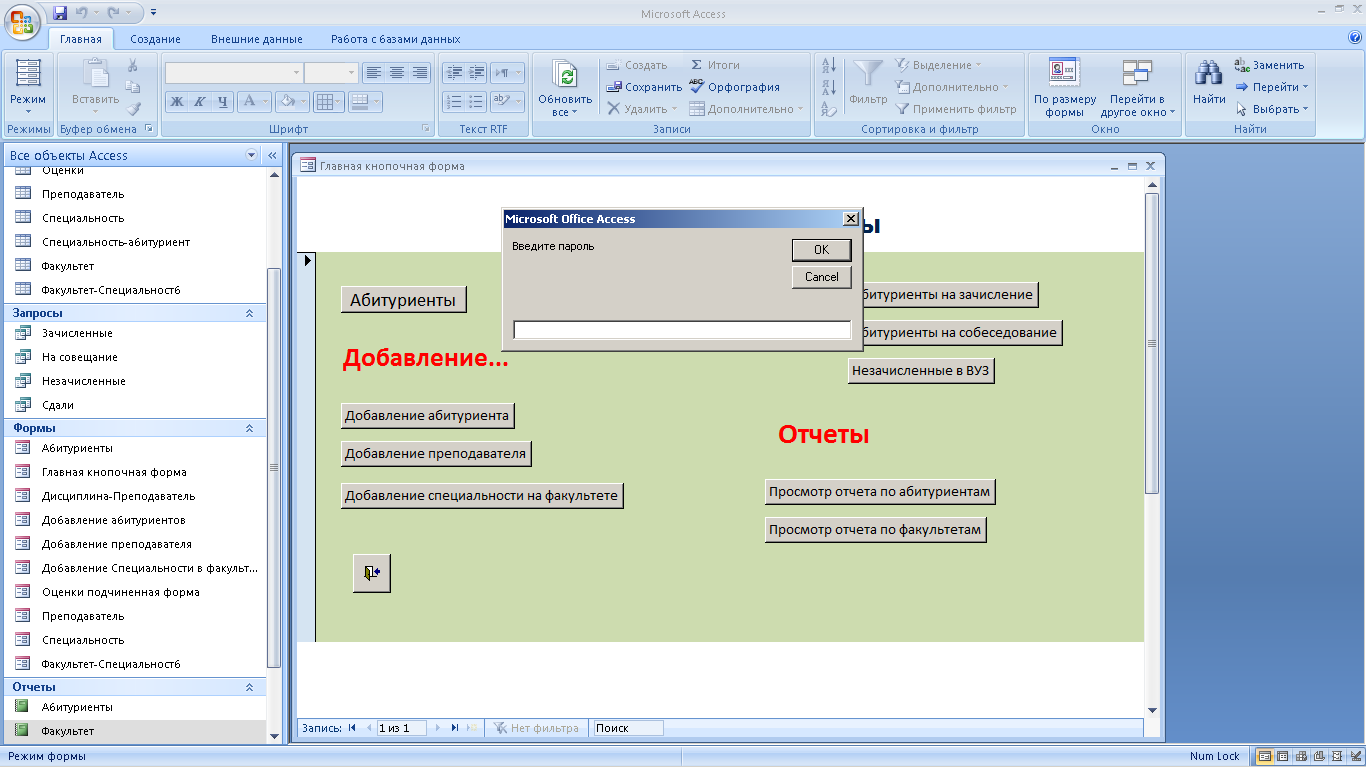


Рисунок 19 – Окно ввода пароля

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