МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное

учреждение высшего образования

«Московский государственный технологический университет «СТАНКИН»

(ФГБОУ ВО МГТУ «СТАНКИН»)

Институт

информационных систем и технологий

Кафедра

информационных систем

Отчёт по самостоятельной работе

по дисциплине **«Управление данными»**

на тему: **Проектирование базы данных поликлиники**

Студент _____ Махмудов Б.Н.

группа ИДБ-16-07 подпись

Руководитель _____ Быстрикова В.А.

Старший преподователь подпись

Москва 2018

Оглавление

Глава 1 Анализ предметной области

1.1 Определение анализа предметной области

Предметная область — часть реального мира, рассматриваемая в пределах данного контекста. Под контекстом здесь может пониматься, например, область исследования или область, которая является объектом некоторой деятельности.[1]

Деятельность, направленная на выявление реальных потребностей заказчика, а также на выяснения смысла высказанных требований, называется анализом предметной области. Одна из первых задач, с решением которых сталкивается разработчик программной системы - это изучение, осмысление и анализ предметной области. Дело в том, что предметная область сильно влияет на все аспекты проекта: требования к системе, взаимодействие с пользователем, модель хранения данных, реализацию и т.д. Анализ предметной области, позволяет выделить ее сущности, определить первоначальные требования к функциональности и определить границы проекта.

Предметной областью данной работы является деятельность поликлиники. Далее следует анализ деятельности поликлиники, выявление требований к разрабатываемой системе, а также определение функций, которые данная система должна будет предоставлять пользователям.

1.2 Поликлиника: описание предметной области

Поликлиника — многопрофильное или специализированное лечебно-профилактическое учреждение для оказания амбулаторной медицинской помощи больным на приёме и на дому. На территории России распределены по территориальному признаку, и являются базовым уровнем медицинского обслуживания населения. По мощности городские поликлиники делятся на

5 групп. В структуре городской поликлиники предусматриваются различные подразделения:

1. регистратура,
2. лечебно-профилактические подразделения,
3. терапевтические отделения,
4. отделение восстановительного лечения,
5. отделения по оказанию специализированных видов медицинской помощи (хирургическое, гинекологическое) с кабинетами соответствующих специалистов (кардиологический, ревматологический, неврологический, урологический, офтальмологический и т.п.

Число отделений и кабинетов, их потенциальные возможности определяются мощностью поликлиники и количеством штатных должностей, которые зависят от численности закрепленного за поликлиникой населения. Структура поликлиники (открытие тех или иных отделений, кабинетов и т. п.) зависит от обращаемости населения в это учреждение, от способности поликлиники предоставить больным необходимую медицинскую помощь.[2]

На сегодняшний день автоматизации подвержено подавляющее большинство сфер деятельности человека, включая здравохранение. Автоматизация в области здравохранения особенно актуальна ввиду роста человеческого населения и бюрократизации в сфере оказания медицинских услуг, что приводит к неудобствам и затруднениям для больных в получении вышеупомянутых услуг. Разработка информационной системы с централизованной базой данных, предоставляющая пользователям возможность удалённо получать справки и записываться на приём к врачам, позволит уменьшить нагруженность самого учреждения и улучшить качество услуг для пациентов. Таким образом автоматизация функционирования поликлиники, в частности разработка базы данных для неё позволит пациентам сэкономить время на очередях и бюрократических формальностях, а сотрудникам сосредоточить-

ся непосредственно на оказании медицинских услуг.

1.3 Существующие продукты, решающие проблему автоматизации

1.3.1 1С : Медицина. Поликлиника

Прикладное решение «1С:Медицина. Поликлиника» предназначено для автоматизации основных процессов медицинских организаций различных организационно-правовых форм, оказывающих медицинскую помощь в амбулаторно-поликлинических условиях.

Прикладное решение «1С:Медицина. Поликлиника» позволяет создать единое информационное пространство медицинской организации с разделением доступа к данным по ролевому принципу. Имеется возможность вести учет по нескольким медицинским организациям в одной информационной базе.

Программа позволяет вести несколько медицинских карт для одного пациента - амбулаторную карту, стоматологическую карту и т.д., пример карты пациента приведён на (рис. 1.1). Для каждого медицинского работника указывается, к какому типу карт он имеет доступ. В программе имеются гибкие механизмы квотирования, которые позволяют устанавливать ограничения на объемы оказываемой медицинской помощи. Учет деятельности медицинского персонала ведется по медицинским услугам. Пример пользвательского интерфейса программы показан на рис. 1.2

Предварительную запись пациентов может осуществлять как регистратура, так и врачи при выполнении назначений повторных приемов, консультаций, исследований, манипуляций. Для осуществления оперативного планирования врачебному медицинскому персоналу и кабинетам задаются графики работы, нормы загрузки, перечень выполняемых услуг. Оперативное плани-

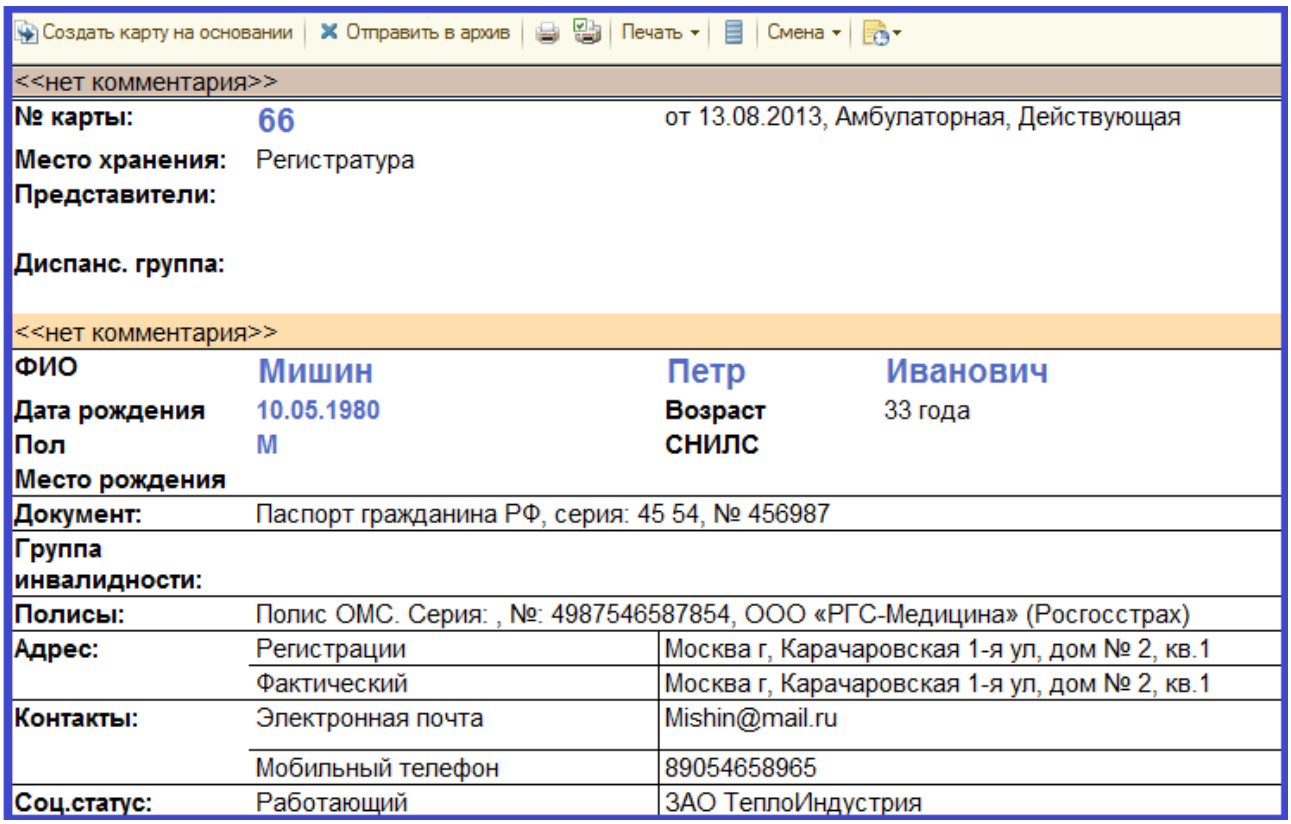

Рис. 1.1 Пример карты пациента

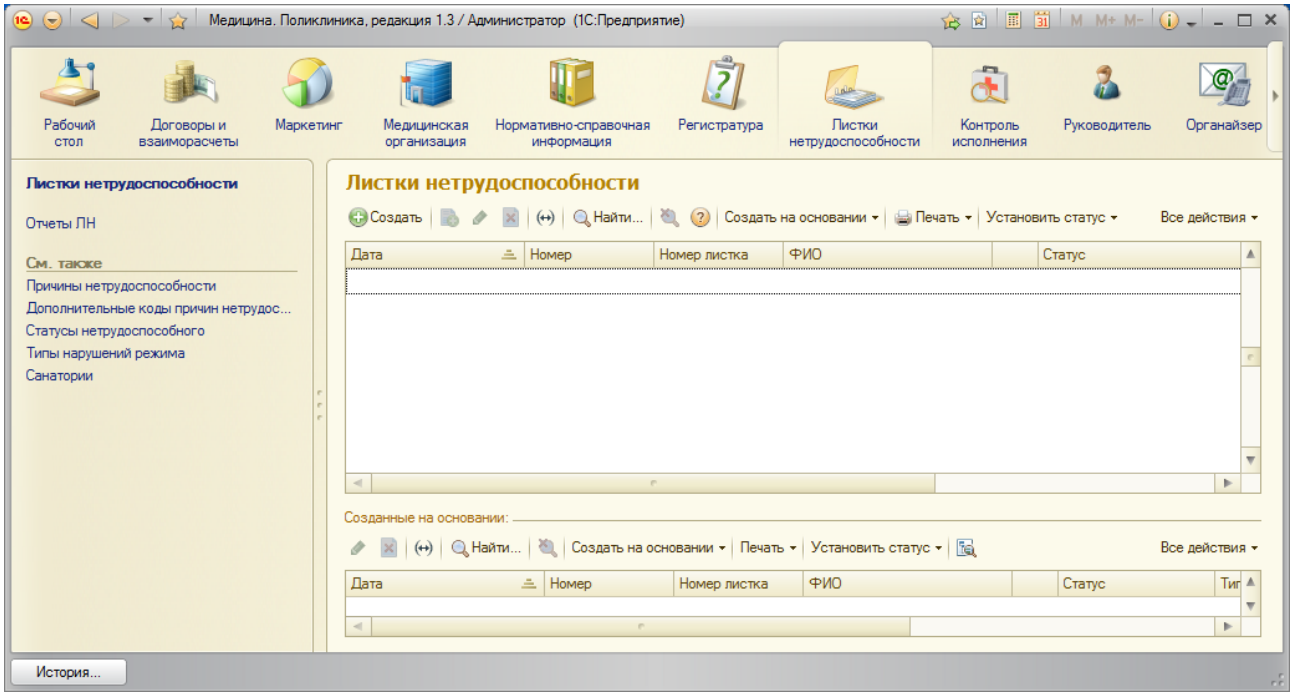

Рис. 1.2 Пользовательский интерфейс «1С:Медицина. Поликлиника»

рование деятельности кабинетов осуществляется по данным предварительной записи пациентов.[3] Можно выделить основные функциональные возможности «1С:Медицина. Поликлиника»:

1. Регистратура
2. Листки нетрудоспособности (больничные)
3. Договорной отдел
4. Контроль исполнения медицинских услуг персоналом
5. Руководитель и аналитическая (статистическая) служба
6. Электронные медицинские карты
7. Профосмотры
8. Интернет запись на прием и обмен данными с сайтами

1.3.2 Сайт частных поликлиник «СМ-Клиника»

Многопрофильный медицинский холдинг «СМ-Клиника» - это сеть многопрофильных медицинских центров для взрослых и детей, основанной в 2002 году. Услуги поликлиник предоставляются на коммерческой основе. Сайт компании доступен по адресу «http://www.smclinic.ru/». Скриншот главной страницы сайта приведён на рис. 1.3. На главной странице сайта можно выделить следующие функции:

- записаться на приём,
- личный кабинет,
- услуги,
- анализы и диагностика.

Функция «Записаться на приём» позволяет предварительно записаться на приём к лечащему врачу посредством заполнения со стороны пользователя соответствующей формы (рис. 1.4).

Функция «Личный кабинет» предоставляет широкий перечень возможностей для зарегистрированных пользователей:

– Получить доступ к своей медицинской карте - увидеть детальную историю всех посещений клиники, фамилии лечащих докторов, их специализации, точные даты посещений и другую полезную информацию;

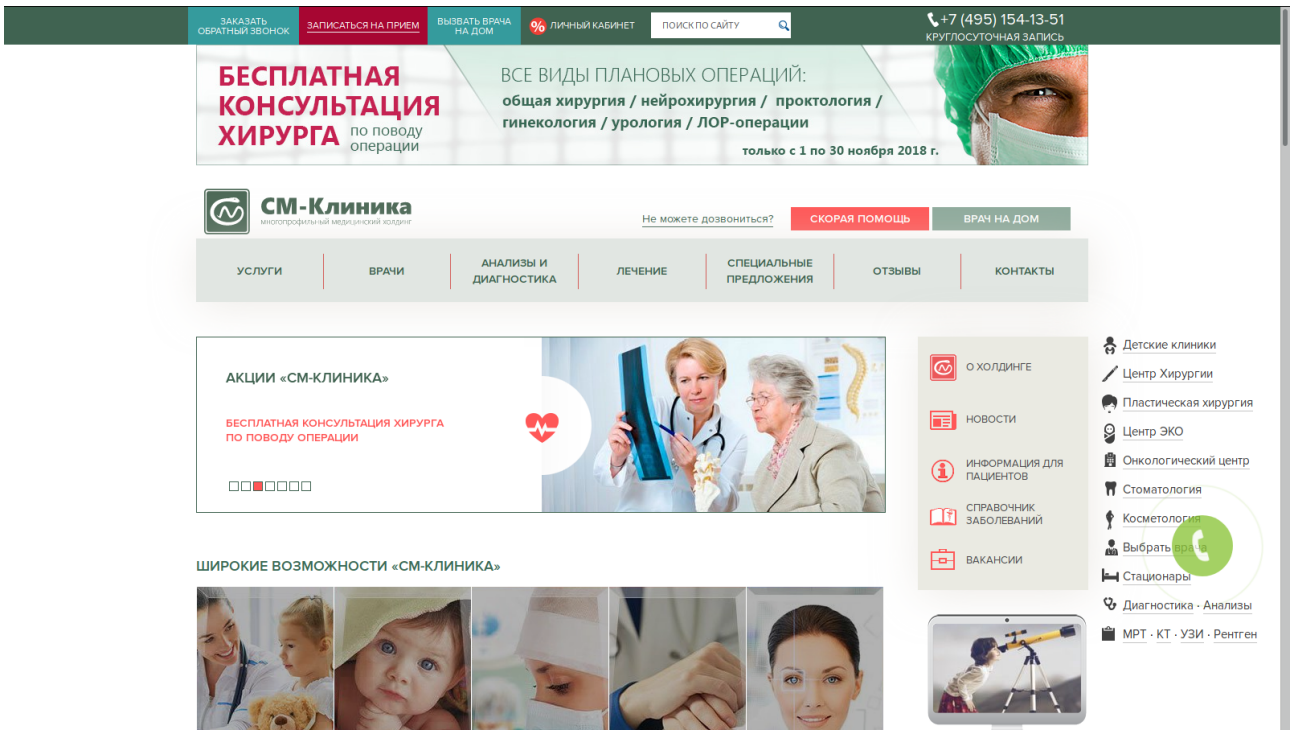

Рис. 1.3 Главная страница сайта «СМ-Клиника»

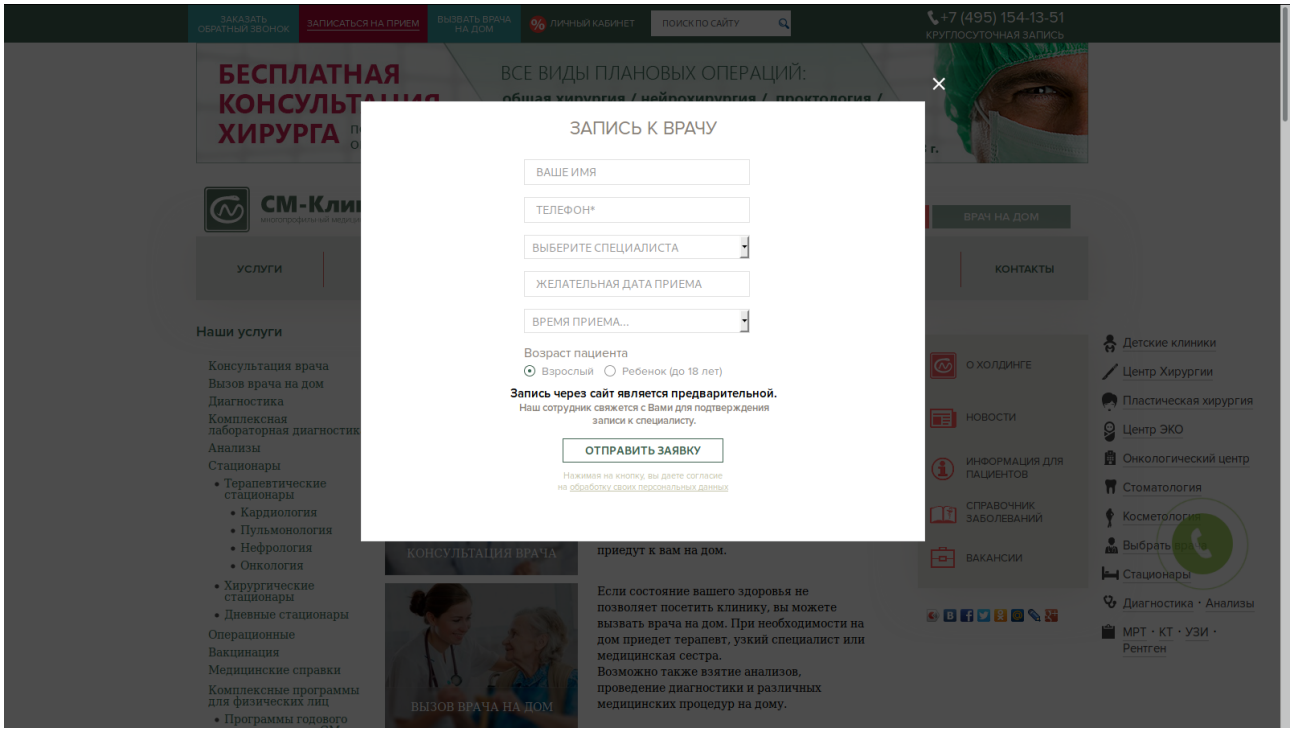

Рис. 1.4 Форма записи на приём к врачу в «СМ-Клиника»

– посмотреть назначенную схему лечения, рекомендации лечащих врачей, назначенные обследования и др.;
– ознакомиться с результатами анализов и обследований, сохранить их на локальный компьютер или сразу распечатать;

Рис. 1.5 Страница с перечнем предоставляемых услуг

– увидеть, когда лечащий врач пациента работает и какое время для приема на текущий момент у него свободно;
– самостоятельно записаться к врачу в удобное для пользователя время;
– посмотреть текущую скидку пользователя, актуальные акции и предложения клиник;
– оставить отзыв о враче или клинике.[4]

Функция «Услуги» позволяет подробно ознакомится с перечнем услуг предоставляемых поликлиникой (рис. 1.5).

1.3.3 Сравнение «1С:Медицина. Поликлиника» и сайта поликлиники «СМ-Клиника»

Оба рассмотренных продукта, хотя и ориентированы на разные категории пользователей, имеют перечень общих функций:

– доступ к электронной медицинской карте,
– запись на прием к врачу,
– ознакомиться с графиком лечащего врача,

– доступ к перечню предоставляемых поликлиникой услуг.

В рамках данной работы планируется создание единой платформы для взамодействия пациентов с сотрудниками поликлиники. Проанализировав программные продукты решающие схожие задачи в данной предметной области, можно выделить ряд функций, которые необходимо реализовать в процессе разработки:

– Просмотр и/или редактирование медицинских карт,
– просмотр и/или редактирование результатов анализов,
– запись на приём к врачу,
– просмотр и/или редактирование записей к врачам,
– создание листков нетрудоспособности,
– просмотр и/или редактирование результатов профосмотров,
– поиск всей информации касательно пациента,
– доступ к графику работы врачей;

Глава 2 Концептуальное проектирование

2.1 Определение концептуального проектирования

Концептуальное проектирование — построение семантической модели предметной области, то есть информационной модели наиболее высокого уровня абстракции. Такая модель создаётся без ориентации на какую-либо конкретную СУБД и модель данных. Чаще всего концептуальная модель базы данных включает в себя описание информационных объектов или понятий предметной области и связей между ними. Для визуализации концептуальной модели часто используется диаграмма вариантов использования.

Диаграмма вариантов использования (ДВИ) — диаграмма, отражающая отношения между действующими лицами и вариантами использования разрабатываемой системы, и позволяющей описать систему на концептуальном уровне. Основное назначение диаграммы — описание функциональности и поведения, позволяющее заказчику, конечному пользователю и разработчику совместно обсуждать проектируемую или существующую систему.

Действующее лицо - внешняя по отношению к ИС сущность, которая может взаимодействовать с системой. Действующим лицом могут быть как люди, так и внешние системы или устройства.

Вариант использования — возможность моделируемой системы (часть её функциональности), благодаря которой пользователь может получить конкретный, измеримый и нужный ему результат. [5]

2.2 Концептуальная модель базы данных поликлиники

В завершении анализа предметной области был выделен перечень функций, которые необходимо реализовать в рамках данной работы, и рассматривая данную ИС с точки зрения возможных вариантов использования,

можно разделить эти функции между действующими лицами.

Действующими лицами разрабатываемой системы являются:

1. сотрудник регистратуры,
2. врач,
3. пациент.

Далее приведены варианты использования для каждого из действующих лиц. Сама же диаграмма показана на рис. 2.1.

- Сотрудник регистратуры может выполнять в системе следующие действия:
    1. поиск по медицинским картам пациентов,
    2. выдача информации по графику работы врачей,
    3. создание медицинской карты пациента,
    4. запись пациента к врачу.
- Врач может выполнять в системе следующие действия:
    1. промотр истории болезней пациента.
    2. назначение лечения,
    3. назначение анализов,
    4. выписка рецептов пациентам,
    5. выписка листков нетрудоспособности.
- Пациент может выполнять в системе следующие действия:
    1. ознакомление с графиком работы врачей,
    2. запись на приём к врачу,
    3. просмотр личной медицинской карты,

Приняв во внимание всё вышесказанное, можно выделить данные, которые необходимо хранить в проектируемой базе данных:

– информация о пациентах,
– информация о сотрудниках поликлиники,
– информация о медицинских картах, т.е. их цифровые копии,
– информация о графике приема пациентов врачами,

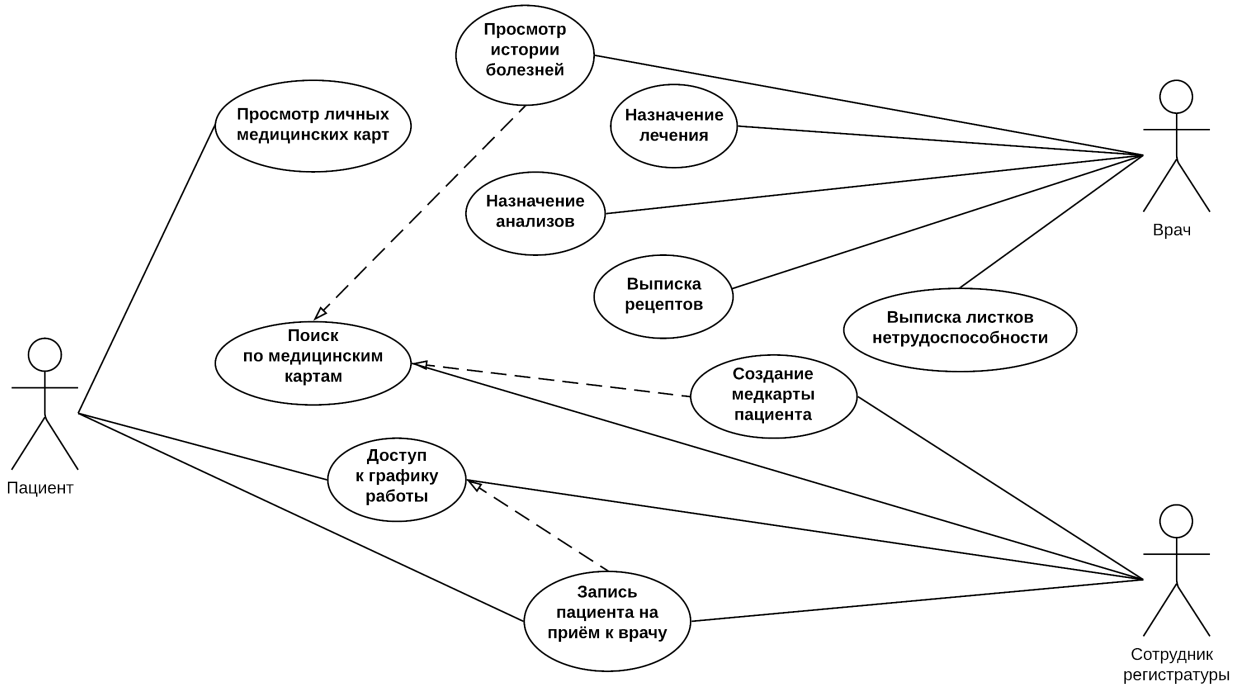

Рис. 2.1 Диаграмма вариантов использования

– информация о записях пациентов,
– информация об анализах,
– информация о выписанных рецептах,
– информация о выданных листках нетрудоспособности,
– информация о назначенных лечениях.

Перечисленные данные можно подразделить на две группы:

- условно-постоянные(данные пациентов, сотрудников),
- оперативно-обновляемые(медицинские карты, график приема пациентов врачами, информация об анализах и т.д.).

Глава 3 Логическое проектирование

3.1 Определение логического проектирования

Логическое проектирование — создание схемы базы данных на основе конкретной модели данных, например, реляционной модели данных. Для реляционной модели данных, логическая модель это набор схем отношений, обычно с указанием первичных ключей, а также «связей» между отношениями, представляющих собой внешние ключи. Логическая модель описывает понятия предметной области, их взаимосвязь, а также ограничения на данные, налагаемые предметной областью.

Опираясь на концептуальную модель, проектируется логическая модель. Часто для выполнения этой задачи используются различные модели данных, например ER-модель. На этапе логического проектирования учитывается специфика конкретной модели данных, но может не учитываться специфика конкретной СУБД.

ER-модель (от англ. entity-relationship model, модель «сущность — связь») — модель данных, позволяющая описывать концептуальные схемы предметной области. ER-модель используется при высокоуровневом (концептуальном) проектировании баз данных. С её помощью можно выделить ключевые сущности и обозначить связи, которые могут устанавливаться между этими сущностями.[5]

3.2 ER-модель базы данных поликлиники

На основе концептуальной модели поликлиники, описанной в конце предыдущей главы, можно спроектировать ER-модель данной предметной области, и представить полученую модель с помощью стандартной графической нотации ER-диаграммы (диаграммы сущность-связь).

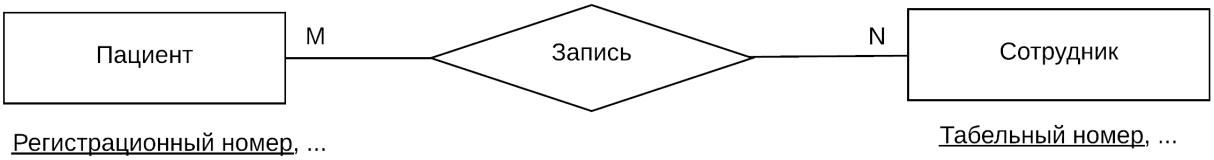

Рис. 3.1 Запись пациента к сотруднику (врачу)

Тип связи “Запись” (рис. 3.1) - M:N. Класс принадлежности необязательный для обоих экземпляров сущностей. Пациент может записаться к нулю или более сотрудникам (врачам), сотрудник может принять ноль или более пациентов.

Рис. 3.2 Посещения врача пациентом по медкарте

Тип связи “Посещения” (рис. 3.2) - M:N. Класс принадлежности необязательный для обоих экземпляров сущностей. Сотрудник может принимать нуль или более пациентов по медкарте, медкарта может содержать информацию о нуль или более посещениях пациента.

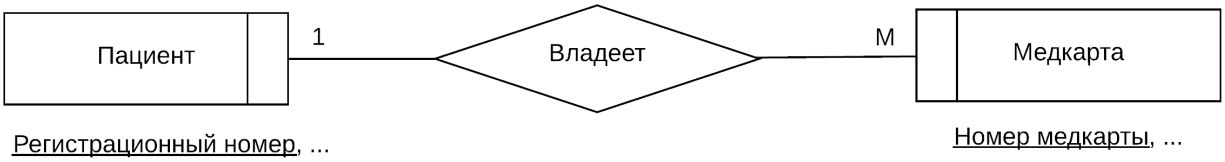

Рис. 3.3 Принадлежность медицинской карты пациенту

Тип связи “Владеет” (рис. 3.3) - 1:M. Класс принадлежности обязательный для обоих экземпляров сущностей. Пациент может владеть одной или более медкартами, медкарта должна принадлежать только одному пациенту.

Рис. 3.4 Назначение анализов пациенту по медкарте

Тип связи “Назначает” (рис. 3.4) - 1:M. Класс принадлежности обязательный только для экземпляра сущности Анализ. Сотрудник может назначить нуль или более анализов, анализ может быть назначен только одним сотрудником.

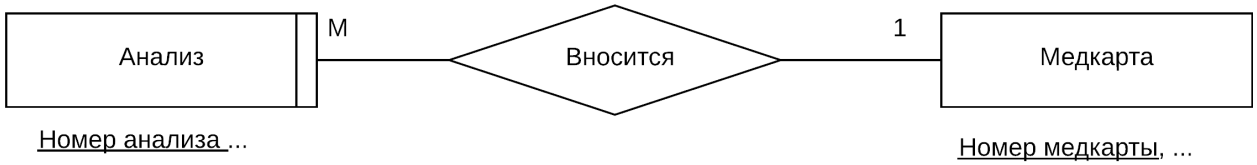

Рис. 3.5 Назначение анализов пациенту по медкарте

Тип связи “Вноситься” (рис. 3.5) - 1:M. Класс принадлежности обязательный только для экземпляра сущности Анализ. Анализ может быть сдан ровно одним пациентом по медкарте, по медкарте пациент может сдать нуль или более анализов

Рис. 3.6 Назначение лечения врачом

Тип связи “Предписывает” (рис. 3.6) - 1:M. Класс принадлежности обязательный только для экземпляра сущности Лечение. Лечение может быть предписано ровно в одним сотрудником, сотрудник может предписать нуль или более лечений.

Тип связи “Вносится в” (рис. 3.7) - 1:M. Класс принадлежности обязательный только для экземпляра сущности Лечение. Лечение может быть

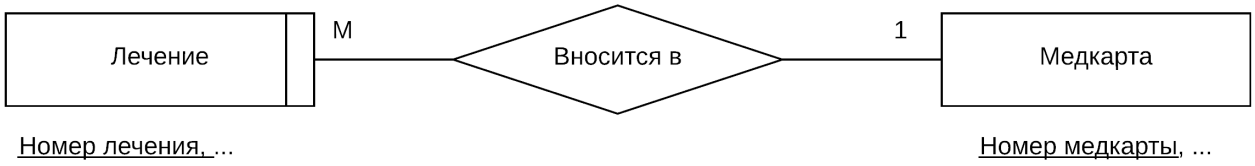

Рис. 3.7 Внесение записи о лечении в медицинскую карту

внесено ровно в одну медкарту, в медкарту можно внеси нуль или более лечений.

Рис. 3.8 Выписывание врачом рецепта пациенту

Тип связи “Выписывает” (рис. 3.8) - 1:M. Класс принадлежности обязательный только для экземпляра сущности Рецепт. Врач может выписать нуль или более рецептов, рецепт может быть выписан ровно одним врачом.

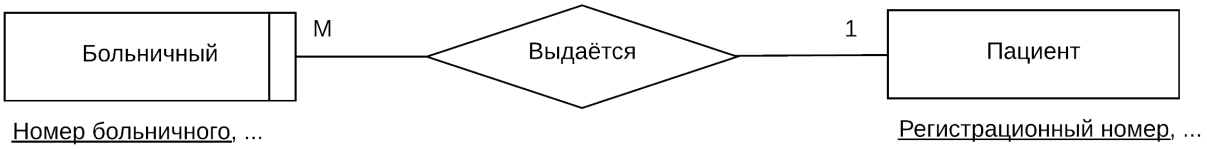

Рис. 3.9 Выдача больничного пациенту

Тип связи “Выдаётся” (рис. 3.9) - 1:M. Класс принадлежности обязательный только для экземпляра сущности Больничный. Больничный (как документ) может быть выдан ровно одному пациенту, пациент может получить нуль или более больничных.

Рис. 3.10 График работы сотрудников

Тип связи “Работает по” (рис. 3.10) - M:N. Класс принадлежности обязательный для экземпляра сущности Сотрудник. Сотрудник может работать более одного дня в неделю, в один день могут работать нуль или более сотрудников.

3.3 Преобразование ER-модели в реляционную

1. Связь “Запись” удовлетворяет условиям правила 6, откуда получаются следующие таблицы
   - Пациент(Регистрационный номер, ...);
   - Сотрудник(Табельный номер, ...);
   - Запись(Табельный номер, Регистрационный номер, ...).
2. Связь “Посещения” удовлетворяет условиям правила 6, откуда получаются следующие таблицы
   - Медкарта(Номер медкарты, Регистрационный номер, ...);
   - Сотрудник(Табельный номер, ...);
   - Посещения(Номер посещения, Табельный номер, Номер медкарты, ...).
3. Связь “Владеет” удовлетворяет условиям правила 4, откуда получаются следующие таблицы
   - Пациент(Регистрационный номер, ...);
   - Медкарта(Номер медкарты, Регистрационный номер, ...).
4. Связь “Назначает” удовлетворяет условиям правила 4, откуда получаются следующие таблицы
   - Сотрудник(Табельный номер, ...);
   - Анализ(Номер анализа, Табельный номер, ...).
5. Связь “Вносится” удовлетворяет условиям правила 4, откуда получаются следующие таблицы
   - Медкарта(Номер медкарты, ...);

- Анализ(Номер анализа, номер медкарты, ...).

6. Связь “Предписывает” удовлетворяет условиям правила 4, откуда получаются следующие таблицы
   - Сотрудник(Табельный номер, ...);
   - Лечение(Номер лечения, Табельный номер, ...).
7. Связь “Вносится в” удовлетворяет условиям правила 4, откуда получаются следующие таблицы
   - Медкарта(Номер медкарты, ...);
   - Лечение(Номер лечения, Номер медкарты, ...).
8. Связь “Рецепт” удовлетворяет условиям правила 4, откуда получаются следующие таблицы
   - Сотрудник(Табельный номер, ...);
   - Рецепт(Номер рецепта, Табельный номер, ...).
9. Связь “Выдаётся” удовлетворяет условиям правила 4, откуда получаются следующие таблицы
   - Пациент(Регистрационный номер, ...);
   - Больничный(Номер больничного, Регистрационный номер, ...).
10. Связь “Работает по” удовлетворяет условиям правила 6, но так как сущность «День недели» имеет только один атрибут, то в результате получаются две таблицы, с миграцией первичного ключа сущности «День недели» в дочернее отношение.
    - Сотрудник(Табельный номер, ...);
    - График(День недели, Табельный номер, ...).

В итоге получается десять таблиц, которые приведены ниже с перечнем всех относящихся к ним атрибутам.

1. Пациент(Регистрационный номер, ФИО, страховая компания, договор страхования, дата рождения, пол, адрес проживания, контактный телефон, пароль от ЛК);
2. Сотрудник(Табельный номер, ФИО, должность, стаж, дата рождения,

пол, адрес проживания, контактный телефон, пароль от ЛК);

3. Запись(Табельный номер, Регистрационный номер, дата и время записи);
4. Посещения(Номер посещения, Табельный номер, Номер медкарты, дата посещения, цель визита);
5. Медкарта(Номер медкарты, Регистрационный номер, дата заведения, тип);
6. Анализ(Номер анализа, Табельный номер, Номер медкарты, дата сдачи, вид анализа, результат);
7. Лечение(Номер лечения, Табельный номер, номер медкарты, дата назначения, заболевание, назначенное лечение);
8. Рецепт(Номер рецепта, Табельный номер, дата выписки, медикаменты);
9. Больничный(Номер больничного, Регистрационный номер, дата выдачи, начало больничного,окончание больничного, место работы/обучения);
10. График(День недели, Табельный номер, начало смены, конец смены, перерыв).

Глава 4 Физическое проектирование

Физическое проектирование — создание схемы базы данных для конкретной СУБД. Специфика конкретной СУБД может включать в себя ограничения на именование объектов базы данных, ограничения на поддерживаемые типы данных и т. п. Кроме того, специфика конкретной СУБД при физическом проектировании включает выбор решений, связанных с физической средой хранения данных (выбор методов управления дисковой памятью, разделение БД по файлам и устройствам, методов доступа к данным), создание индексов и т. д.[5]

4.1 Выбор СУБД

В данной работе для реализации базы данных поликлиники была выбрана реляционная СУБД SQLite. Такой выбор обоснован спецификой предметной области, а именно относительно небольшим объёмом хранимых данных и средней частотой обращения к базе данных. SQLite является встраиваемой, т.е. вместо использования привычной парадигмы клиент-сервер, SQLite представляет из себя библиотеку с интерфейсами для многих языков программирования. Этот факт делает SQLite чрезвычайно компактным и быстрым, т.к. СУБД встраиваится напрямую в разрабатываемое приложение. Несмотря на свою простоту данная СУБД может обслуживать базы данных размером до 140 террабайт и поддерживает параллельный доступ к БД несколькими процессами. [6]

4.2 Схема БД поликлиники

На основании ER-модели, полученной в конце предыдущей главы, где были описаны таблицы и их атрибуты, далее приведены окончательные структуры таблиц (табл. 4.1 - 4.12), созданных в СУБД SQLite. Поля пе-

речислены в том же порядке, который был описан ранее. SQL операторы использованные для создания таблиц приведены в приложении А, в приложении Б приведены примеры их заполнения.

Таблица 4.1

Структура таблицы «Сотрудник»

Столбец	**Тип данных**	**Нуль?**	**Ключ**	**По умолч.**	**Огранич.**	**Ссылка**
tabid	integer	not null	первичный			
fio	text	not null				
position	text	not null				
experien	integer	not null		0	< 100	
birthdate	date	not null			< date('now')	
gender	char(1)	not null			'М' или 'Ж'	
address	text					
pnumber	text					
password	text	not null		abs(random())		

Таблица 4.2

Структура таблицы «Пациент»

Столбец	**Тип данных**	**Нуль?**	**Ключ**	**По умолч.**	**Огранич.**	**Ссылка**
regid	integer	not null	первичный			
fio	text	not null				
insurcomp	text					
icontract	text					
birthdate	date	not null			< date('now')	
gender	char(1)	not null			'М' или 'Ж'	
address	text					
pnumber	text					
password	text	not null		abs(random())		

Таблица 4.3

Структура таблицы «Запись»

Столбец	**Тип данных**	**Нуль?**	**Ключ**	**По умолч.**	**Огранич.**	**Ссылка**
tabid	integer	not null	первичный, внешний			Сотрудник (tabid)
regid	integer	not null	первичный, внешний			Пациент (regid)
recdatetime	datetime	not null				

Таблица 4.4

Структура таблицы «Медкарта»

Столбец	**Тип данных**	**Нуль?**	**Ключ**	**По умолч.**	**Огранич.**	**Ссылка**
cardid	integer	not null	первичный			
regid	integer	not null	внешний			Пациент (regid)
crdate	date	not null		date('now')		
type	text					

Таблица 4.5

Структура таблицы «Посещения»

Столбец	**Тип данных**	**Нуль?**	**Ключ**	**По умолч.**	**Огранич.**	**Ссылка**
visid	integer	not null	первичный			
tabid	integer	not null	внешний			Сотрудник (tabid)
cardid	integer	not null	внешний			Медкарта (cardid)
visdate	date	not null		date('now')		
visgoal	text	not null				

Таблица 4.6

Структура таблицы «Рецепт»

Столбец	**Тип данных**	**Нуль?**	**Ключ**	**По умолч.**	**Огранич.**	**Ссылка**
receiptid	integer	not null	первичный			
tabid	integer	not null	внешний			Сотрудник (tabid)
issuedate	date	not null		date('now')		
medicine	text	not null				

Таблица 4.7

Структура таблицы «График»

Столбец	**Тип данных**	**Нуль?**	**Ключ**	**По умолч.**	**Огранич.**	**Ссылка**
weekday	text	not null	первичный			
tabid	integer	not null	первичный, внешний			Сотрудник (tabid)
shiftst	integer	not null				
shiftend	integer	not null				
break	integer	not null				

Таблица 4.8

Структура таблицы «Лечение»

Столбец	**Тип данных**	**Нуль?**	**Ключ**	**По умолч.**	**Огранич.**	**Ссылка**
treatid	integer	not null	первичный			
tabid	integer	not null	внешний			Сотрудник (tabid)
cardid	integer	not null	внешний			Медкарта (cardid)
trdate	date	not null	первичный	date('now')		
illness	text					
treatment	text					

Таблица 4.9

Структура таблицы «Больничный»

Столбец	**Тип данных**	**Нуль?**	**Ключ**	**По умолч.**	**Огранич.**	**Ссылка**
sickid	integer	not null	первичный			
regid	integer	not null	внешний			Пациент (regid)
issuedate	date	not null		date('now')		
stdate	date	not null				
endate	date	not null			endate > stdate	
destn	text	not null				

Таблица 4.10

Структура таблицы «Анализ»

Столбец	**Тип данных**	**Нуль?**	**Ключ**	**По умолч.**	**Огранич.**	**Ссылка**
anid	integer	not null	первичный			
tabid	integer	not null	внешний			Сотрудник (tabid)
cardid	integer	not null	внешний			Медкарта (cardid)
passdate	date	not null		date('now')		
type	text	not null				
result	text					

Таблица «Сотрудник» содержит информацию о сотрудниках поликлиники (см. табл. 4.1).

Таблица «Пациент» содержит персональные данные и контактную информацию пациентов поликлиники (см. табл. 4.2)

Таблица «Запись» содержит информацию о записях пациентов к врачам поликлиники, дополнительные ограничения на таблицу это требования уникальности табельного номера врача и времени записи unique(tabid, recdatetime), а также уникальность регистрационного номера пациента и времени записи unique(regid, recdatetime) (см. табл. 4.3).

Таблица «Медкарта» содержит информацию о медкартах пациентов, также на таблицу наложены дополнительные ограничения, такие как требования уникальности регистрационного номера пациента и типа медкарты unique(regid, type) (см. табл. 4.4).

Таблица «Посещения» содержит информацию о факте посещения врача пациентом (см. табл. 4.5).

Таблица «Рецепт» содержит информацию о рецептах выписанных врачами пациентам (см. табл. 4.6).

Таблица «График» содержит информацию о графиках сотрудников. (см. табл. 4.7).

Таблица «Лечение» содержит информацию о лечениях назначаемых врачами пациентам (см. табл. 4.8).

Таблица «Больничный» содержит информацию о листках нетрудоспособности выданных врачами пациентам (см. табл. 4.9).

Таблица «Анализ» содержит информацию об анализах назначенных врачамм пациентам поликлиники (см. табл. 4.10).

На рис. clinic4.1 можно ознакомиться с общей схемой базы данных созданной в СУБД SQLite. Визуализаация схемы была произведена средствами программного продукта DataGrip версия 2017.2.

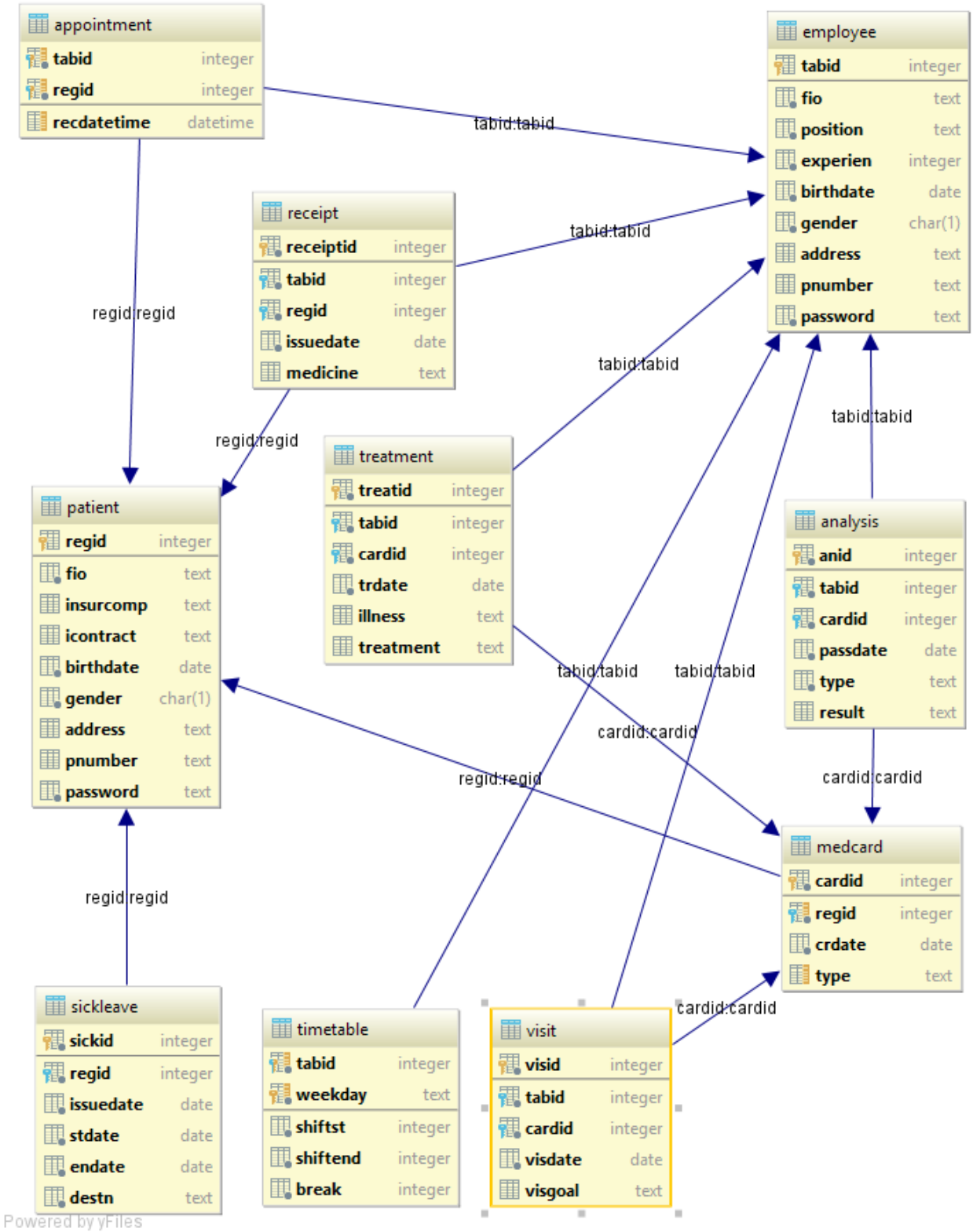

Рис. 4.1 Схема базы данных поликлиники

4.3 SQL операторы создания таблиц

1. Создание таблицы «Пациент»

```
CREATE TABLE patient (
    regid integer PRIMARY KEY,
    fio text NOT NULL,
    insurcomp text,
    icontract text,
    birthdate date NOT NULL,
    gender char(1) NOT NULL CHECK(gender = 'M' OR gender = 'Ж'),
    address text,
    pnumber text,
    password text NOT NULL DEFAULT (abs(random()))
);
```

2. Создание таблицы «Сотрудник»

```
CREATE TABLE employee (
    tabid integer PRIMARY KEY,
    fio text NOT NULL,
    position text NOT NULL,
    experien integer NOT NULL DEFAULT 0,
    birthdate date NOT NULL,
    gender char(1) NOT NULL CHECK(gender = 'M' OR gender = 'Ж'),
    address text,
    pnumber text,
    password text NOT NULL DEFAULT (abs(random()))
);
```

3. Создание таблицы «Запись»

```
CREATE TABLE appointment (
    tabid integer NOT NULL REFERENCES employee(tabid),
    regid integer NOT NULL REFERENCES patient(regid),
    recdatetime datetime,
    PRIMARY KEY(tabid, regid),
);
```

4. Создание таблицы «Больничный»

```
CREATE TABLE sickleave (
    sickid integer NOT NULL PRIMARY KEY,
    regid integer NOT NULL REFERENCES patient(regid),
    issuedate date NOT NULL DEFAULT (date('now')),
    stdate date NOT NULL,
    endate date NOT NULL,
    destn text NOT NULL
);
```

5. Создание таблицы «График»

```
CREATE TABLE timetable (
    tabid integer NOT NULL REFERENCES employee(tabid),
    weekday text NOT NULL,
    shiftst integer NOT NULL,
    shiftend integer NOT NULL,
    break integer NOT NULL,
    PRIMARY KEY(tabid, weekday)
);
```

6. Создание таблицы «Анализ»

```
CREATE TABLE analysis (
    anid integer NOT NULL PRIMARY KEY,
    tabid integer NOT NULL REFERENCES employee(tabid),
    cardid  integer NOT NULL REFERENCES medcard(cardid),
    passdate date NOT NULL DEFAULT (date('now')),
    type text NOT NULL,
    result text
);
```

7. Создание таблицы «Лечение»

```
CREATE TABLE treatment (
    treatid integer NOT NULL PRIMARY KEY,
    tabid integer NOT NULL REFERENCES employee(tabid),
    cardid integer NOT NULL REFERENCES medcard(cardid),
    trdate date NOT NULL DEFAULT (date('now')),
    illness text,
    treatment text
);
```

8. Создание таблицы «Медкарта»

```
CREATE TABLE medcard (
    cardid integer NOT NULL PRIMARY KEY,
    regid integer NOT NULL REFERENCES patient(regid),
    crdate date NOT NULL DEFAULT (date('now')),
    type text,
);
```

9. Создание таблицы «Посещения»

```
CREATE TABLE visit (
    visid integer NOT NULL PRIMARY KEY,
    tabid integer NOT NULL REFERENCES employee(tabid),
    cardid integer NOT NULL REFERENCES medcard(cardid),
    visdate date NOT NULL DEFAULT (date('now')),
    visgoal text
);
```

10. Создание таблицы «Рецепт»

```
CREATE TABLE receipt (
    receiptid integer NOT NULL PRIMARY KEY,
    tabid integer NOT NULL REFERENCES employee(tabid),
    regid integer NOT NULL REFERENCES patient(regid),
    issuedate date NOT NULL DEFAULT (date('now')),
    medicine text
);
```

4.4 Индексы, представления, триггеры необходимые для решеня задач БД

11. Индексы таблицы «Запись»

```
CREATE UNIQUE INDEX idx_appointment_tr ON appointment(tabid,
   recdatetime);
CREATE UNIQUE INDEX idx_appointment_rr ON appointment(regid,
   recdatetime);
```

12. Индексы таблицы «Медкарта»

```
CREATE UNIQUE INDEX idx_medcard_rt ON medcard(regid, type);
```

13. Представление «medicalcard» собирающее данные из таблиц «Анализ» «Посещения» «Лечение» «Сотрудник» по фильтру - идентификатор медкарты

```
CREATE VIEW medicalcard AS
SELECT
        "AN00000" || an.anid as type,
        "D000000" || an.tabid AS doctor_num, emp.fio,
        emp.position,
        "C000000" || an.cardid as card,
        an.passdate AS occurence,
        an.type AS antype,
        an.result
        FROM analysis an INNER JOIN employee emp ON emp.tabid = an.
           tabid
UNION ALL
SELECT
        "TR00000" || tr.treatid AS type,
        "D000000" || tr.tabid AS doctor_num,
        emp.fio,
        emp.position,
        "C000000" || tr.cardid AS card,
        tr.trdate AS occurence,
        tr.illness,
        tr.treatment
        FROM treatment tr INNER JOIN employee emp ON emp.tabid = tr.
           tabid
UNION ALL
SELECT
        "VS00000" || vis.visid AS type,
        "D000000" || vis.tabid AS doctor_num,
        emp.fio,
        emp.position,
        "C000000" || vis.cardid AS card,
        vis.visdate AS occurence,
        vis.visgoal,
        ' '
        FROM visit vis INNER JOIN employee emp ON emp.tabid = vis.tabid
ORDER BY occurence;
```

14. Представление «patient_info» собирающее данные из таблицы «Пациент» по фильтру - идентификатор пациента с переименованием полей

```
CREATE VIEW patient_info AS
SELECT
```

```
        fio AS ФИО,
        insurcomp AS 'Страховая компания',
        icontract AS 'Номер полисастрахования ',
        birthdate AS 'Дата рождения',
        CASE gender
        WHEN 'М' THEN 'мужчина'
        WHEN 'Ж' THEN 'женщина'
        END AS 'Пол',
        address AS 'Адрес проживания',
        pnumber AS 'Контактный телефон',
        regid
FROM patient;
```

15. Представление «patient_medcard» собирающее данные из таблиц «Пациент» «Медкарта» и из представления «medicalcard» по фильтру идентификатор пациента

```
CREATE VIEW patient_medcard AS
SELECT
        pat.regid AS regid,
        mc.cardid,
        pat.fio,
        mc.crdate AS occurence,
        mc.type,
        (SELECT count(*) FROM medicalcard WHERE card LIKE "%00" || mc.
            cardid) AS rec_count
        FROM medcard mc INNER JOIN patient pat ON pat.regid = mc.regid
ORDER BY occurence;
```

16. Триггер «insert_visit» срабатывающий после вставки в таблицу «Посещения» и удаляющий запись пациента к врачу

```
CREATE TRIGGER insert_visit AFTER INSERT ON visit
BEGIN
        INSERT INTO appointment_history SELECT * FROM appointment
        WHERE regid = (SELECT  regid FROM medcard WHERE cardid = new.
            cardid) AND date(recdatetime) <= new.visdate;
        DELETE FROM appointment
        WHERE regid = (SELECT  regid FROM medcard WHERE cardid = new.
            cardid) AND date(recdatetime) <= new.visdate
        AND tabid = new.tabid;
END;
```

17. Триггер «insert_treatment» срабатывающий после вставки в таблицу «Лечение» и удаляющий запись пациента к врачу

```
CREATE TRIGGER insert_treatment AFTER INSERT ON treatment
BEGIN
        INSERT INTO appointment_history SELECT * FROM appointment
        WHERE regid = (SELECT  regid FROM medcard WHERE cardid = new.
           cardid)
        AND date(recdatetime) <= new.trdate;
        DELETE FROM appointment
        WHERE regid = (SELECT  regid FROM medcard WHERE cardid = new.
           cardid) AND date(recdatetime) <= new.trdate
        AND tabid = new.tabid;
END;
```

18. Триггер «insert_analysis» срабатывающий после вставки в таблицу «Анализ» и удаляющий запись пациента к врачу

```
CREATE TRIGGER insert_analysis AFTER INSERT ON analysis
BEGIN
        INSERT INTO appointment_history SELECT * FROM appointment
        WHERE regid =
        (SELECT  regid FROM medcard WHERE cardid = new.cardid)
                AND date(recdatetime) <= new.passdate;
        DELETE FROM appointment
        WHERE regid = (SELECT  regid FROM medcard WHERE cardid = new.
           cardid)
        AND date(recdatetime) <= new.passdate
        AND tabid = new.tabid;
END;
```

19. Триггер «archive_card» срабатывающий при попытке удаления записи из таблицы «Медкарта» и архивирующий карту

```
CREATE TRIGGER archive_card BEFORE DELETE ON medcard
BEGIN
        INSERT INTO analysis_history SELECT * FROM analysis
        WHERE cardid = old.cardid;
        DELETE FROM analysis WHERE cardid = old.cardid;
        INSERT INTO treatment_history SELECT * FROM treatment
        WHERE cardid = old.cardid;
        DELETE FROM treatment WHERE cardid = old.cardid;
        INSERT INTO visit_history SELECT * FROM visit
        WHERE cardid = old.cardid;
        DELETE FROM visit WHERE cardid = old.cardid;
        DELETE FROM medcard WHERE cardid = old.cardid;
END;
```

20. Триггер «archive_patient» срабатывающий при попытке удаления записи из таблицы «Пациент» и архивирующий пациента

```
CREATE TRIGGER archive_patient BEFORE DELETE ON patient
BEGIN
        INSERT INTO medcard_history SELECT * FROM medcard
        WHERE regid = old.regid;
        DELETE FROM medcard WHERE regid = old.regid;
        INSERT INTO patient_history SELECT * FROM patient
        WHERE regid = old.regid;
        DELETE FROM patient WHERE regid = old.regid;
END;
```

Глава 5 Описание функционирования БД

5.1 Назначение и перечень функций базы данных

Далее приведены функции, которые предоставляет, реализованный в данной работе программный продукт, пользователям для работы с базой данных.

1. Поиск по пациентам, используя часть имени пациента в качестве критерия поиска;
2. Получение доступа к списку врачей с целью;
   - ознакомления с расписанием выбранного врача,
   - запись на прием к выбранному врачу.
3. Просмотр перечня карт выбранного пациента, и дальнейшее ознакомление с содержимым отдельно взятой карты;
4. Просмотр рецептов выписанных конкретному пациенту;
5. Просмотр листов нетрудоспособности выданных конкретному пациенту;
6. Назначение лечения пациенту с занесением в медицинскую карту;
7. Назначение сдачи анализов пациенту с занесением факта назначения и результа анализа в медицинскую карту;
8. Выписка рецепта пациенту;
9. Выдача листа нетрудоспособности пациенту;
10. Создание нового пациента в базе данных;
11. Создание новой медицинской карты выбранному пациента;
12. Удаление пациента из базы активных пациентов, с последующим перемещение всех его данных в архив;
13. Получение доступа к архиву пациентов.

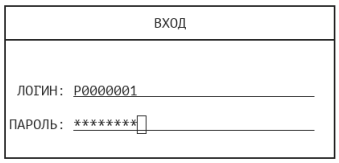

Рис. 5.1 Окно входа в систему

Разграничение доступа к какой-либо отдельно взятой функции, осуществляется на стороне клиентского приложения, в зависимости от роли пользователя (пациент, врач, сотрудник регистратуры).

5.2 Описание работы с базой данных

5.2.1 Вход в систему

При попытке входа в систему, программа определяет тип пользоваетеля и делает запрос в базу данных к соответсвующей таблице. В зависимости от ответа базы данных, программа либо запускает необходимый интерфейс пользоваетеля, либо выдает сообщение о неправильности введённых данных. Запросы которые отправляются в базу данных следующие

```
1. SELECT * FROM patient where regid = ? AND password = ?
2. SELECT * FROM employee where tabid = ? AND password = ?
```

Здесь и далее символ «?» используется для обозначения мест в sql запросах, куда будут подставлены параметры. Выбор первого или второго запроса совершает клиентское приложение, в зависимости от введённых пользоваетелем

данных. Если запрос вернёт кортеж, то такой пользоваетель сущностуют в базе, и программа открывает для него соответсвующий интерфейс.

5.2.2 Интерфейс пациента

Интерфейс пациента приведен на рис. 5.2, перечень доступных функций виден на рисунке в правом окне, в левом окне отображена личная информация пользователя. Функция записи к врачу (рис. **??**) состоит из трёх этапов:

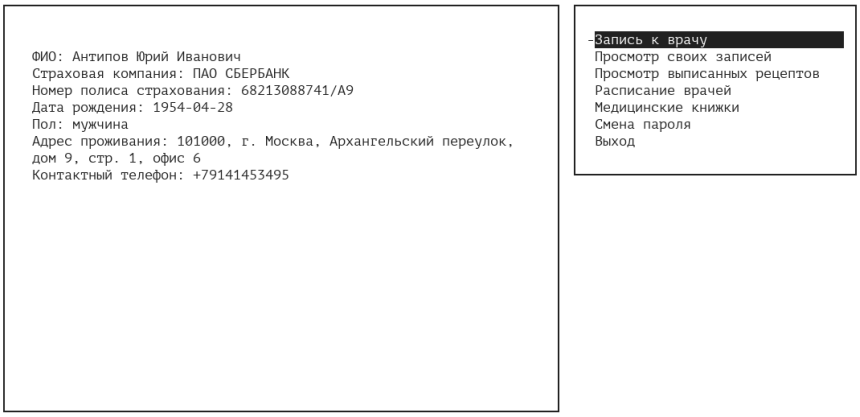

Рис. 5.2 Окно интерфейса пользоваетеля

выбор врача, выбор даты (доступны ближайшие семь дней), выбор времени приёма. Если врач имеет выходные в ближайшие семь дней, то эти даты не доступны для выбора (рис. **??**). Также присутствует проверка доступности времени приёма, таким образом из списка (рис. **??**) удаляются время перерыва врача, и то время на которое уже существуют записи. Запросы которые отправляются в базу данных следующие

```
1. SELECT vacant_time(?, ?)"
2. INSERT INTO appointment values(?, ?, ?)"
3. SELECT e.fio, a.recdatetime, a.tabid FROM appointment a
   INNER JOIN employee e ON a.tabid = e.tabid
   WHERE (a.regid = ? AND a.tabid = ?) OR (a.regid = ? AND a.
      recdatetime = ?)
```

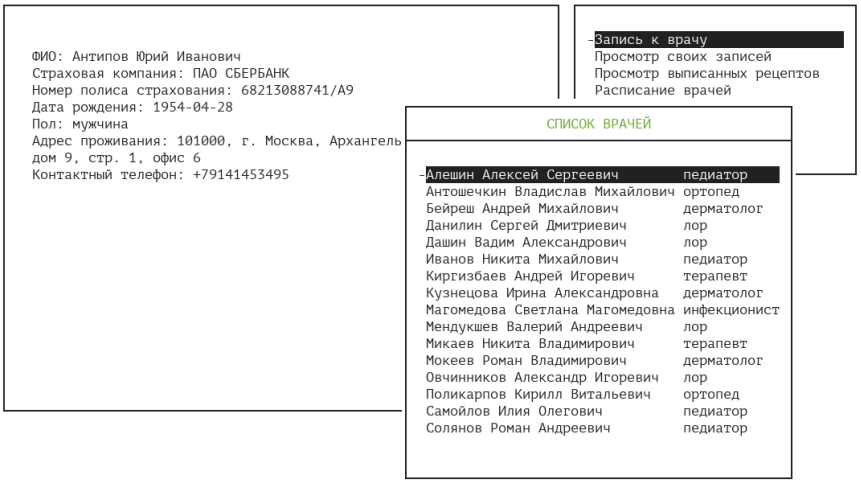

Рис. 5.3 Запись. Этап выбора врача

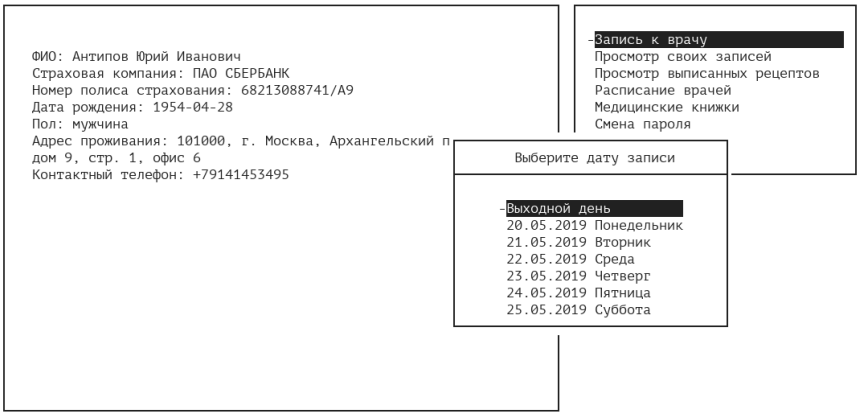

Рис. 5.4 Запись. Этап выбора даты

Первый запрос использует пользовательскую функцию для получения всего свободного времени которое доступно у выбранного врача. Второй запрос используется для вставки записи в таблицу «Запись». Третий запрос срабатывает в том случае если пациент уже имеет запись к какому либо врачу и выбирает то же время к другому врачу, тем самым вызывая нарушение

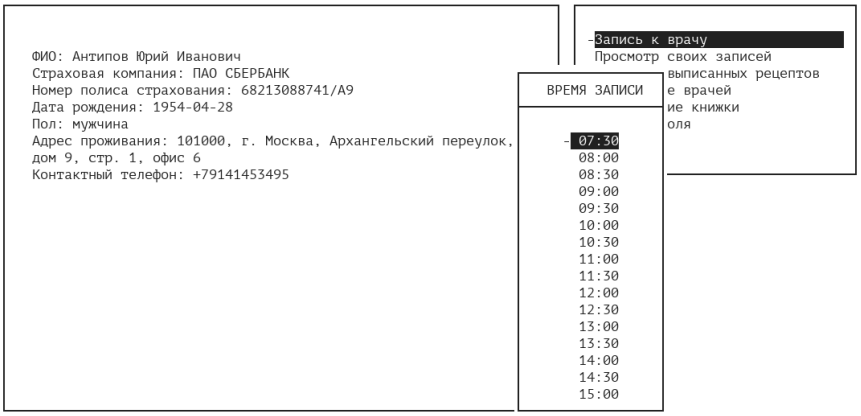

Рис. 5.5 Запись. Этап выбора времени приёма

целостности. Запрос уведомляет пользователя что пациент уже имеет запись на данное время.

Заключение

Целью данной работы является проектирование базы данных для поликлиники. В связи с ростом человеческого населения, и высокой степени бюрократизации поликлиник, проблема автоматизации работы подобного рода учреждений особенно актуальна, так как это непосредственно сказывается на качестве оказываемых услуг. В ходе выполнения работы были достигнуты следующие цели:

1. Была проанализирована предметная область поликлиника и её особенности, также были приведены аргументы в пользу необходимости в автоматизации. Были рассмотрены две существующих продукта, которые в той или иной мере решают проблему автоматизации, предоставляемые ими функции и их специфика. На основании анализа предметной области был выделен перечень функций для будущей реализации.
2. Следующим этапом стало концептуальное проектирование, в ходе которого были выделены три действующих лица (врач, сотрудник регистратуры, пациент), для которых была составлена диаграмма вариантов использования, с целью формализации функциональных требований к разрабатываемой системе.
3. Далее в ходе логического проектирования была создана ER-модель базы данных, где были определены основные сущности и их взаимодействия. Для визуализации ER-модели были использованы ER-диаграммы. В завершении данного этапа были построена схема БД в виде набора схем отношений. А именно следующие десять отношений: Пациент, Сотрудник, Запись, Посещения, Больничный, Лечение, Анализ, График, Медкарта, Рецепт.
4. Завершающим этапом стало физическое проектирование, в ходе которого в СУБД SQLite была создана база данных поликлиники со всеми таблицами, определенных на предыдущем этапе, после чего все таблицы

были заполнены данными.

Литература

[1] Предметная область: Wikipedia - свободная энциклопедия. — https://ru.wikipedia.org/wiki/Domain_knowledge (дата обращения: 09.11.2018).

[2] *Ольга Жидкова*. Медицинская статистика: конспект лекций. — Eksmo education, 2009. — С. 180.

[3] 1С:Медицина. Поликлиника. — https://solutions.1c.ru/catalog/clinic/features (дата обращения: 11.11.2018).

[4] Авторизация СМ-Клиника. — https://lk.smclinic.ru/ (дата обращения: 11.11.2018).

[5] Проектирование баз данных: Wikipedia - свободная энциклопедия. — https://ru.wikipedia.org/wiki/Database_design (дата обращения: 11.11.2018).

[6] *Owens Mike*. The Definitive Guide to SQLite. — 1-ое изд. — Apress, 2006. — С. 440.

Примеры заполнение таблиц данными

r...	fio	insurco...	icontract	birthd...		address	pnumb...	pas...
1	Махаров Кузьма Виктор...	ООО Полис...	1099131941...	1983-11-28	М	г. Москва, Ти...	+79314149...	moscow...
2	Никифорова Валерия Гр...	ОАО Док-П...	5039431949...	1995-10-01	Ж	г. Москва, ул...	+79652132...	tututo...
3	Алёшин Всеволод Григо...	ООО Ниарм...	1239451949...	1989-01-01	М	г. Москва, ул...	+79652132...	791849...

Рис. 1 Заполнение таблицы «Пациент»

t...	fio	position	e... 1	birthda...	g...	address	pnumber	passwo...
1	Варшавин А...	Главный в...	23	1978-04-20	М	г. Москва, Варл...	+79962429...	alex23
3	Волошин Ро...	Невропато...	18	1971-01-31	М	г. Москва, ул. ...	+79664582...	901319402...
2	Прокофьева...	Врач тера...	12	1981-01-31	Ж	г. Москва, ул. ...	+79664568...	03mazina03

Рис. 2 Заполнение таблицы «Сотрудник»

cardid	regid	crdate	type
1	1	2018-12-04	Амбулаторная
2	2	2017-11-05	Стационарная
3	1	2018-12-04	Стационарная
4	3	2018-04-25	Амбулаторная

Рис. 3 Заполнение таблицы «Медкарта»

an...	ta...	ca...	passd...	type 1	result
1	1	1	2018-08-09	Гемоглоб...	Уровень гемоглобин...
2	2	3	2018-08-09	Гепатит	Положительный
4	2	3	2018-09-01	Грибок	Отрицательный
3	3	2	2018-08-09	Псориаз	Отрицательный

Рис. 4 Заполнение таблицы «Анализ»

visid	tabid	regid	visdate	visgoal
1	2	1	2018-12-18	Консультация
2	3	2	2017-05-23	Осмотр
3	1	4	2018-10-10	Консультация

Рис. 5 Заполнение таблицы «Посещения»

receiptid	tabid	re...	issueda...	medicine
1	1	1	2018-11-29	Ибупрофен, 5-Фторурацил-Збеве, Глутамин-
2	2	3	2018-01-09	Абактал 0,08/мл 5мл, Эуфиллин 0,024/м...
3	3	2	2017-03-19	Живокост-бальзам, Иммунал, Кальцид
4	2	3	2018-09-01	Вазотон, Ревалгин

Рис. 6 Заполнение таблицы «Рецепт»

tabid	weekday	shiftst	shiftend	break
1	Понедельник	600	1080	780
2	Вторник	540	900	720
3	Среда	720	1140	840
2	Четверг	780	1200	900

Рис. 7 Заполнение таблицы «График»

	sickid	tab...	regid	issuedate	stdate	endate	destn
1	1	1	1	2018-11-29	2018-05-24	2018-10-19	ООО Химфармпром
2	2	2	3	2018-01-09	2017-11-04	2017-12-29	Школа № 3 по ...
3	3	3	2	2017-03-19	2017-01-23	2017-02-09	ИП Афанасьев И...

Рис. 8 Заполнение таблицы «Больничный»

tabid	regid	recdatetime
1	2	2018-01-01 14:30
2	1	2018-10-11 15:00
3	3	2016-12-11 10:30
1	3	2017-10-08 11:00

Рис. 9 Заполнение таблицы «Запись»

t...	t...	...	trdate	illness	treatment
1	1	2	2017-11-28	Острая амебная дизентерия	Трихоброл 2 недели по 1 таблетке, Эритро...
2	2	3	2018-01-28	Центральноевропейский кл...	Коктейль витаминов группы В внутримышечн...
3	3	1	2017-12-07	Грипп с пневмонией	Арбидол, постельный режим, Ринза

Рис. 10 Заполнение таблицы «Лечение»

Приложение Б

В связи с тем что полный код программы составляет более двух тысяч пятисот строк, в данном приложении приведены лишь выдержки из общей кодовой базы.

Код программы

21. Код интерфейса пациента

```
#include <patient.h>

void patient_interface(int regid) // the boss function to manage the
    view of interface
{
        sqlite3_stmt *stmt;
        char temp[1000];

        int i = 0, rows, cols;

        WINDOW *win_main_menu;

        PANEL *menu;
        MENU *main_menu;
        char *choices[] = {
                "Запись кврачу                    ",
                "Просмотр своихзаписей           ",
                "Просмотр выписанныхрецептов    ",
                "Расписание врачей                ",
                "Медицинские книжки               ",
                "Смена пароля                     ",
                "Выход                             ",
        };

        ITEM **menu_items;

        init_menu(&menu_items,
                        choices,
```

```
                sizeof(choices) / sizeof(char *));
main_menu = new_menu((ITEM **)menu_items);
scale_menu(main_menu, &rows, &cols);

win_main_menu = newwin(rows + 4, cols + 4, 4, 75);
set_menu_win(main_menu, win_main_menu);
set_menu_sub(main_menu,
                derwin(win_main_menu, rows, cols, 2, 2));
menu = new_panel(win_main_menu);
struct win_pan *pat = patient_info(regid);


keypad(win_main_menu, TRUE);
box(win_main_menu, 0, 0);

post_menu(main_menu);
update_panels();

doupdate();
while((i = wgetch(win_main_menu)) != KEY_F(2)){
        switch(i) {
case KEY_DOWN:
        menu_driver(main_menu, REQ_DOWN_ITEM);
        break;
case KEY_UP:
        menu_driver(main_menu, REQ_UP_ITEM);
        break;
case 10:
        if (current_item(main_menu) == menu_items[0])
                appointment(regid);
        else if (current_item(main_menu) == menu_items[1])
                show_appointments(regid);
        else if (current_item(main_menu) == menu_items[2])
                show_receipts(regid);
        else if (current_item(main_menu) == menu_items[3])
                timetable(0);
        else if (current_item(main_menu) == menu_items[4])
                medical_cards(regid, 0);
        else if (current_item(main_menu) == menu_items[5])
                password_change(regid);
```

```
                else
                        goto EXIT;
                }
        }

EXIT:
        i = 0;
        while(*(menu_items + i)) {
                free_item(*(menu_items + i++));
        }
        i = 0;
        del_panel(menu);
        del_panel(pat->pan);
        delwin(win_main_menu);
        delwin(pat->sub);
        delwin(pat->win);
        free(menu_items);
        free_menu(main_menu);
        free(pat);
}
```

22. Код интерфейса врача

```
void doctor_interface(int tabid)
{
        int regid, repeat = 0;
SEARCH:
        regid = patient_search();
        struct win_pan *pat = patient_info(regid);
        char *options[] = {
                "Просмотр картпациента ",
                "Назначение анализов",
                "Внести результатанализов ",
                "Назначение лечения",
                "Выписка рецепта",
                "Выдача листканетрудоспособности ",
                "Просмотр листовнетрудоспособности ",
                "Повторить поиск",
                "Выход",
                NULL
```

```
        };
        while(TRUE) {
                switch (show_menu(
                                        options,
                                        10,
                                        "Выберите действие",
                                        4,
                                        COLS - 70)) {
                        case 0:
                                medical_cards(regid, 0);
                                break;
                        case 1:
                                annalysis_create(tabid, regid);
                                break;
                        case 2:
                                annalysis_update(regid);
                                break;
                        case 3:
                                cure_create(regid, tabid);
                                break;
                        case 4:
                                receipt_issue(tabid, regid);
                                break;
                        case 5:
                                sick_leave_issue(regid, tabid);
                                break;
                        case 6:
                                sickleave_list(tabid, regid);
                                break;
                        case 7:
                                repeat = 1;
                                goto EXIT;
                                break;
                        default:
                                repeat = 0;
                                goto EXIT;
                }
        }

EXIT:
```

```
        del_panel(pat->pan);
        delwin(pat->sub);
        delwin(pat->win);
        update_panels();
        doupdate();
        if (repeat) {
                repeat = 0;
                goto SEARCH;
        }
        return;
}
```

23. Код интерфейса сотрудника регистратуры

```
#include <registry.h>

void registry_interface(void)
{
        char *options[] = {
                "Просмотр картпациентов ",
                "Добавить новогопациента ",
                "Изменить данныепациента ",
                "Удалить пациента",
                "Просмотр расписанияврачей ",
                "Создать записькврачу  ",
                "Архив пациентов",
                "Выход",
                NULL
        };
        int regid = 0;
        while (TRUE) {
                switch (show_menu(
                                        options,
                                        9,
                                        "Доступные действия",
                                        4,
                                        -1))
                {
                        case 0:
                                medcard_search();
```

```
                        break;
                case 1:
                        create_patient();
                        break;
                case 2:
                        alter_patient();
                        break;
                case 3:
                        delete_patient();
                        break;
                case 4:
                        timetable(0);
                        break;
                case 5:
                        regid = patient_search();
                        appointment(regid);
                        break;
                case 6:
                        patient_archive();
                        break;
                default:
                        goto EXIT;
                }
        }
EXIT:
        return;
}
```