Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Программное обеспечение ЭВМ и информационные технологии

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОМУ ПРОЕКТУ

НА ТЕМУ:

Создание базы данных демографической информации

Студент ИУ7-65Б _____ Андрей Владимирович Коротков
(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсового проекта _____ Кирилл Леонидович Тассов
(Подпись, дата) (И.О.Фамилия)

Москва, 2020 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой ИУ7
(Индекс)
_____ И.В. Рудаков
(И.О.Фамилия)
« » 20 г.

З А Д А Н И Е
на выполнение курсового проекта

по дисциплине Базы данных

Студент группы ИУ7-65Б

Коротков Андрей Владимирович
(Фамилия, имя, отчество)

Тема курсового проекта Создание базы данных демографической информации

Направленность КП (учебный, исследовательский, практический, производственный, др.)
учебный
Источник тематики (кафедра, предприятие, НИР) предприятие
График выполнения проекта: 25% к 4 нед., 50% к 7 нед., 75% к 11 нед., 100% к 14 нед.

Задание Разработать клиент-серверное приложение для 3-х видов пользователей:
1. администратор;
2. пользователь «Росстата»;
3. сотрудник любой организации, работающий с демографической информацией.

Спроектировать и реализовать базу данных. Реализовать приложение для взаимодействия с базой данных, которое включает в себя возможность добавление документов, просмотр статистики и администрирования базы данных.

Оформление курсового проекта:

Расчетно-пояснительная записка на 25-30 листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)
На защиту проекта должна быть представлена презентация. На слайдах должны быть:
постановка задачи, использованные методы и алгоритмы, структура комплекса программ, диаграмма классов, интерфейс, характеристики ПО.

Дата выдачи задания « 10 » марта 2020 г.

Руководитель курсового проекта _____ К.Л. Тассов
(Подпись, дата) (И.О.Фамилия)

Студент _____ А.В. Коротков
(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

Содержание

Введение

В современном мире базы данных используются повсеместно для хранения огромного количества информации, которую нельзя уместить на обычный компьютер. Вся информация, хранящаяся в базах данных, структурирована, что позволяет оптимизировать хранилище и беспрепятственно получать необходимую информацию.

В настоящее время у каждой государственной структуры существует своя база данных, минусом такого решения является сложность получения информации по определенному человеку, так как необходимо делать запрос в каждую государственную структуру. Очевидно, что времени, необходимого для получения информации из разных хранилищ, будет затрачено намного больше, чем времени запроса в одно хранилище. Поэтому логичным решением будет сформировать универсальное хранилище для всех государственных структур, которое легко масштабируется.

Целью данной курсовой работы является проектирование и реализация базы данных, а так же создание клиент-серверного приложения для взаимодействия с хранилищем.

Для достижения поставленной цели необходимо решить следующие задачи:

- формализовать задачу в виде необходимого функционала;
- проанализировать существующие модели данных;
- спроектировать базу данных, необходимую для хранения и структурирования данных;
- реализовать спроектированную базу данных с использованием выбранной СУБД;
- реализовать приложение для взаимодействия с реализованной БД.

1. Аналитический раздел

В этом разделе выполняется постановка задачи, рассматриваются общие сведения о БД, проводится анализ существующих СУБД и выбирается наиболее подходящее СУБД для решения поставленных задач.

1.1 Общие сведения о БД

База данных (БД) представляет собой совокупность определенным образом организованных данных, которые хранятся в памяти вычислительной системы и отображают состояние объектов и их взаимосвязей в рассматриваемой предметной области [1].

1.1.1 Типы БД

Модель данных определяет логическую структуру БД и то, каким образом данные будут храниться, организовываться и обрабатываться [2].

Основные типы моделей БД:

1. иерархическая;
2. сетевая;
3. реляционная.

Иерархическая модель БД

Иерархическая модель БД представляет собой древовидную (иерархическую) структуру, состоящую из объектов (данных) различных уровней. Каждый объект может включать в себя несколько объектов более низкого уровня. Такие объекты находятся в отношении предка (объект более близкий к корню) к потомку (объект более низкого уровня), при этом возможна ситуация, когда объект-предок имеет несколько потомков, тогда как у объекта-потомка обязателен только один предок.

Иерархической базой данных является файловая система, состоящая из корневого каталога, в котором имеется иерархия подкаталогов и файлов.

Связи записей реализуются в виде физических указателей с одной записи на другую. Основной недостаток иерархической структуры – невозможность реализовать отношения «многие-ко-многим» а также ситуации, в которых запись имеет несколько предков.

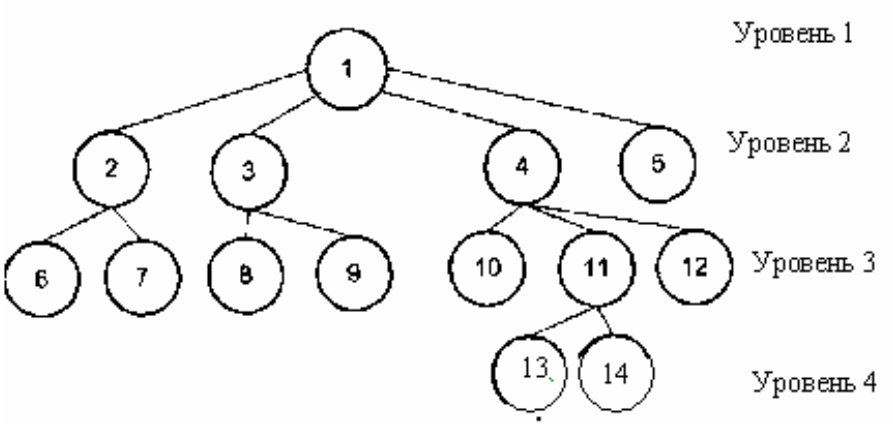

Рисунок 1.1. Иерархическая модель БД.

Сетевая модель БД

Сетевая модель данных является расширением иерархического подхода. Разница между иерархической моделью моделью данных и сетевой заключается в том, что в иерархических структурах запись-потомок должна иметь в точности одного предка, а в сетевой структуре у потомка может быть любое число предков. Записи в такой модели связаны списками с указателями.

Некоторые данные намного естественнее моделировать с несколькими предками для одного дочернего элемента. Сетевая модель позволяла моделировать отношения «многие ко многим». Так же из-за того, что у потомка может быть несколько предков, сетевая модель со временем становится слишком сложной и неудобной для управления.

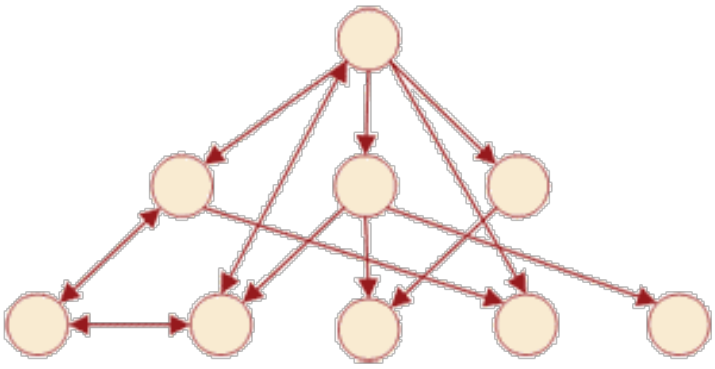

Рисунок 1.2. Сетевая модель БД.

Реляционная модель БД

В реляционной (термин «реляционный» означает, что теория основана на математическом понятии «отношение») модели не существует физических отношении. Вся информация хранится в виде таблиц (отношений), состоящих из рядов и столбцов. А данные двух таблиц связаны общими столбцами, а не физическими ссылками или указателями. Объекты и их отношения представлены таблицами.

В реляционных моделях нет необходимости просматривать все указатели, что облегчает выполнение запросов на выборку информации по сравнению с сетевыми и иерархическими БД.

Каждая реляционная таблица представляет собой двумерный массив и обладает следующими свойствами:

- каждый элемент таблицы — один элемент данных;
- все элементы в одном столбце имеют одинаковый тип;
- каждый столбец имеет уникальное имя;
- одинаковые строки (записи, кортежи) в таблице отсутствуют;
- порядок следования строк и столбцов может быть произвольным;
- каждое поле содержит одну характеристику объекта предметной области;
- в записи собраны сведения об одном экземпляре этого объекта.

Некоторые поля могут быть определены как ключевые. Это значит, что для ускорения поиска конкретных значении будет использоваться индексация. Когда поля двух различных таблиц получают данные из одного набора, можно использовать оператор JOIN для выбора связанных записей двух таблиц, сопоставив значения полей. Такие действия можно расширить до объединения нескольких полей в нескольких таблицах.

Сравнение моделей

Иерархическая модель данных поддерживает отношения типа «один-к-одному» или «один-ко-многим». Она позволяет быстро получать данные, но не отличается гибкостью. Иногда роль элемента (родителя или потомка) неясна и не подходит для иерархической модели.

Вторая, сетевая модель данных, имеет более гибкую структуру, чем иерархическая, и поддерживает отношение «многие ко многим». Но быстро становится слишком сложной и неудобной для управления.

Третья модель — реляционная — более гибкая, чем иерархическая, и проще в управлении, чем сетевая. Реляционная модель сегодня используется чаще всего, так как имеет множество преимуществ, таких как:

- простота использования;
- гибкость;
- независимость данных;
- безопасность;
- простота практического применения;
- слияние данных;
- целостность данных.

Вывод

В ходе сравнения моделей, становится очевидно, что использование реляционной модели данных является наиболее приоритетным.

1.2 Общие сведения о СУБД

Под системой управления базами данных (СУБД) понимается совокупность программных и языковых средств, предназначенных для создания и обработки БД [3].

Среди всех СУБД по управлению реляционными базами данных можно выделить:

1. MySQL;
2. PostgreSQL;
3. SQLite;
4. Oracle Database;
5. Microsoft SQL Server.

В данной работе будет использована СУБД PostgreSQL ввиду простоты масштабирования, мильтиплатформенности, поддержки параллельных процессов выполнения запросов и высокой безопасности аутентификации.

Вывод

В результате проведенного анализа в качестве модели данных была выбрана реляционная модель данных, в качестве СУБД — PostgreSQL.

2. Конструкторский раздел

В данном разделе будут рассмотрены диаграммы классов основных компонент клиент-серверного приложения, проектирование БД.

Программа должна обладать следующей функциональностью:

1. добавление документов в БД;
   1.1. так как у документов возможны различные типы, необходимо, чтобы в интерфейсе заполняемые поля соответствовали типу документа;
2. получение статистики на основе данных из БД;
   2.1. импортирование файла с готовыми SQL-запросами с возможностью их использования;
3. администрирование БД;
   3.1. ввод SQL-запроса;
   3.2. вывод результата в виде таблицы.

2.1 Проектирование БД

Сперва необходимо построить Use-case диаграмму, которая описывает взаимоотношения и зависимости между группами вариантов использования и действующих лиц, участвующими в процессе. Ниже на рис. 2.1 представлена Use-Case диаграмма:

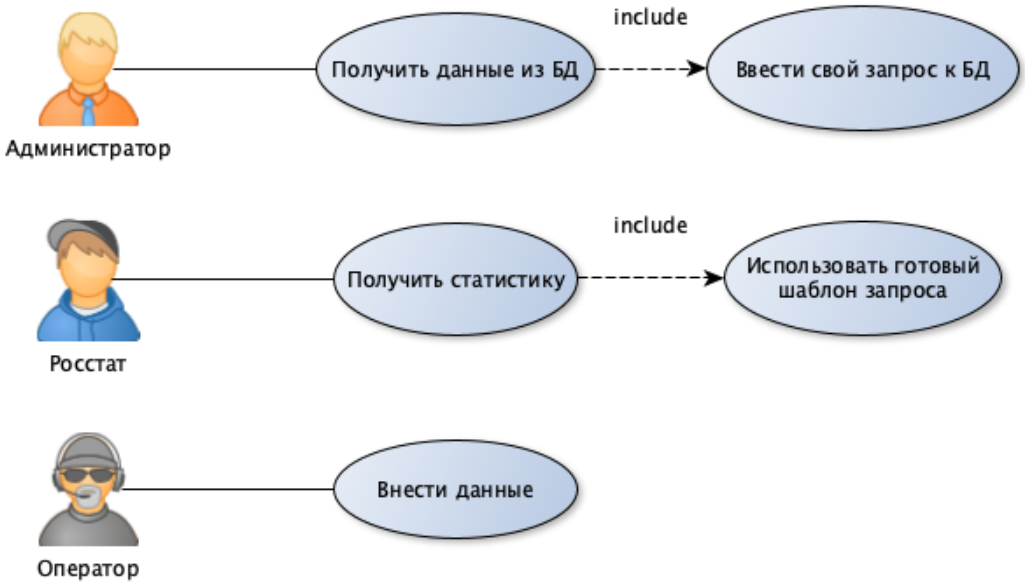

Рисунок 2.1. Use-Case диаграмма БД.

Далее необходимо построить ER-диаграмму (модель «сущность-связь») для описания концептуальной схемы предметной области проекта. С её помощью выделяются ключевые сущности и обозначаются связи, которые могут устанавливаться между этими сущностями. Ниже на рис. 2.2 представлена ER-диаграмма сущностей:

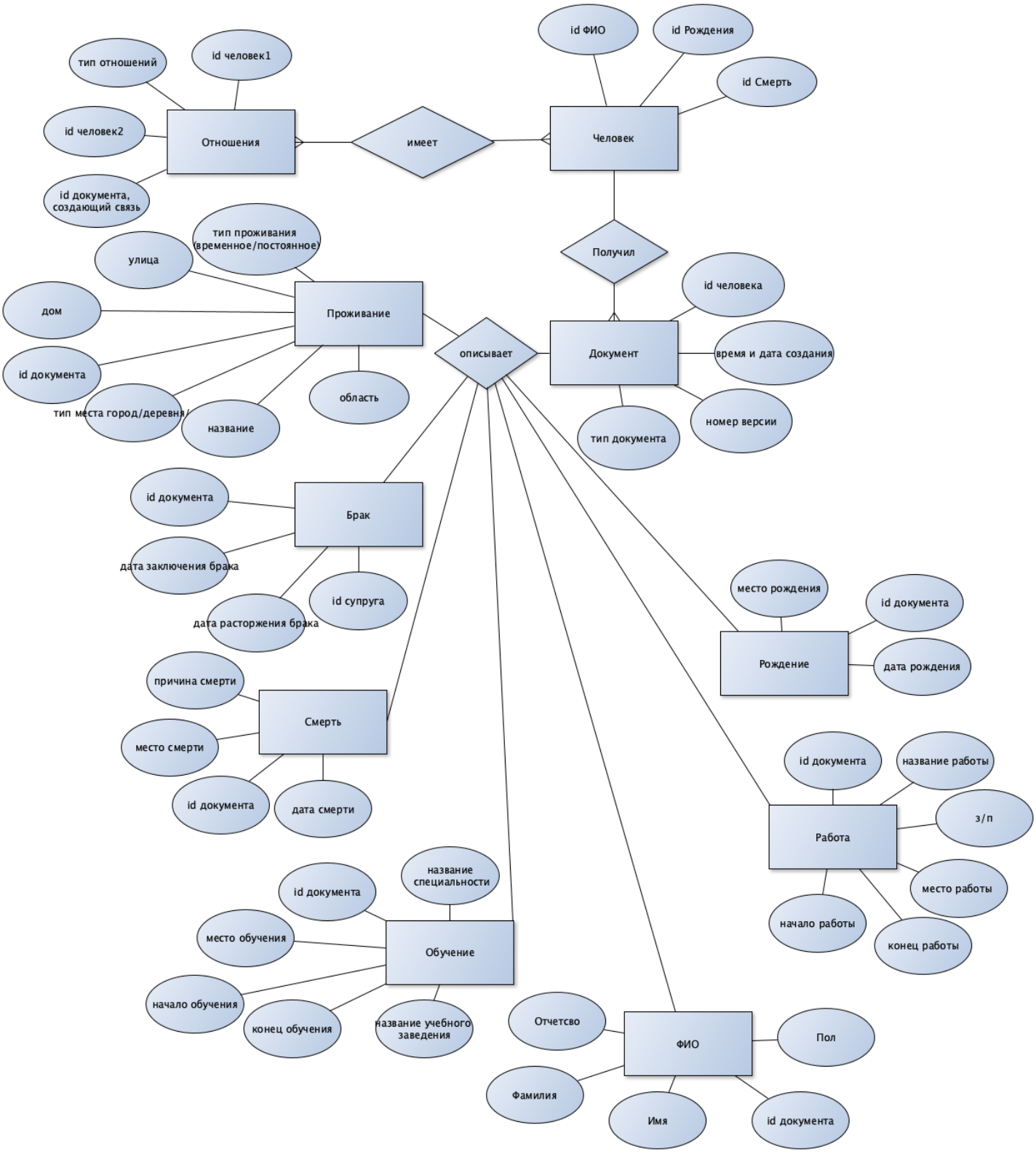

Рисунок 2.2. ER-диаграмма БД.

2.1.1 Проектирование таблиц БД

Исходя из ER-диаграммы (рис 2.2), база данных состоит из следующих таблиц:

1. таблица людей **Person**;
2. таблица отношений между людьми **Relation**;
3. таблица с именами людей **Fio**;
4. таблица, хранящая основную информацию о документах **Document**;
5. таблица с информацией о существующих типов документов в БД **DocType**;
6. таблица о работе **Work**;
7. таблица об обучении **Study**;
8. таблица о рождении **Birth**;
9. таблица о смерти **Death**;
10. таблица о месте жительства **Residence**;
11. таблица о браке **Marriage**;
12. таблица с дополнительными атрибутами для документа **Other-Atribute.**

Таблица Person

Содержит основную информацию о человеке (рождение, смерть, ФИО).

Таблица содержит следующие поля:

1. id — целочисленное поле, идентификатор человека;
2. id_fio — целочисленное поле, идентификатор ФИО человека;
3. id_birth — целочисленное поле, идентификатор рождения человека;
4. id_death — целочисленное поле, идентификатор смерти человека.

Таблица Relation

Содержит информацию об отношениях между людьми (работают вместе/ женаты/учатся вместе).

Таблица содержит следующие поля:

1. id — целочисленное поле, идентификатор отношений;

2. id_doc — целочисленное поле, идентификатор документа, из-за которого появилось отношение.

3. id_person1 — целочисленное поле, идентификатор одного из двух людей, которые связаны отношением;

4. id_person2 — целочисленное поле, идентификатор одного из двух людей, которые связаны отношением;

5. type_relations — целочисленное поле, тип отношений (работают вместе/женаты/учатся вместе).

Таблица Fio

Содержит информацию о фамилии, отчестве, имени человека, а так же о его поле.

Таблица содержит следующие поля:

1. id — целочисленное поле, идентификатор ФИО;

2. id_doc — целочисленное поле, идентификатор документа, к которому привязана запись в таблице Fio;

3. surname — символьное поле, фамилия человека;

4. name — символьное поле, имя человека;

5. patronymic — символьное поле, отчество человека;

6. sex — символьное поле.

Таблица Document

Содержит основную информацию о документе.

Таблица содержит следующие поля:

1. id — целочисленное поле, идентификатор документа;

2. id_person — целочисленное поле, идентификатор человека, к которому привязан документ;

3. type_doc — целочисленное поле, тип документа.

4. time_create — поле, хранящее дату и время создания документа;

5. time_issuing — поле, хранящее дату и время выдачи документа;
6. time_start_action_doc — поле, хранящее дату и время начала действия документа;
7. time_end_action_doc — поле, хранящее дату и время истечения срока действия документа;
8. version — целочисленное поле, версия документа;
9. num_doc — целочисленное поле, номер документа;
10. issuing_athority — символьное поле, название органа, выдавший документ.

Таблица DocType

Хранит типы документов.

Таблица содержит следующие поля:

1. id — целочисленное поле, идентификатор типа документа;
2. name — символьная строка, название типа документа.

Таблица Work

Содержит информацию о работе людей.

Таблица хранит следующие поля:

1. id — целочисленное поле, идентификатор работы;
2. id_doc — целочисленное поле, идентификатор документа, к которому привязана запись в таблице Work;
3. place — символьное поле, название места работы;
4. time_start — поле, хранящее дату и время начала работы;
5. time_end — поле, хранящее дату и время окончания работы;
6. organization — символьное поле, название организации;
7. position — символьное поле, занимаемая должность;
8. salary — целочисленное поле, зарплата.

Таблица Study

Содержит информацию об учебе людей.

Таблица хранит следующие поля:

1. id — целочисленное поле, идентификатор учебы;
2. id_doc — целочисленное поле, идентификатор документа, к которому привязана запись в таблице Study;
3. place — символьное поле, название места учебы;
4. time_start — поле, хранящее дату и время начала учебы;
5. time_end — поле, хранящее дату и время окончания учебы;
6. organization — символьное поле, название учебного заведения;
7. specialty — символьное поле, специальность.

Таблица Birth

Содержит информацию о рождении.

Таблица хранит следующие поля:

1. id — целочисленное поле, идентификатор рождения;
2. id_doc — целочисленное поле, идентификатор документа, к которому привязана запись в таблице Birth;
3. place — символьное поле, место рождения;
4. time — поле, хранящее дату и время рождения.

Таблица Death

Содержит информацию о смерти.

Таблица хранит следующие поля:

1. id — целочисленное поле, идентификатор смерти;
2. id_doc — целочисленное поле, идентификатор документа, к которому привязана запись в таблице Death;
3. place — символьное поле, место смерти;
4. time — поле, хранящее дату и время смерти;
5. type_death — целочисленное поле, тип смерти.

Таблица Residence

Содержит информацию о месте проживания.

Таблица хранит следующие поля:

1. id — целочисленное поле, идентификатор места проживания;
2. id_doc — целочисленное поле, идентификатор документа, к которому привязана запись в таблице Residence;
3. type_residence — целочисленное поле, тип проживания (временное/ постоянное);
4. region — символьное поле, область/край места жительства;
5. type_locality — целочисленное поле, тип места жительства (город/ поселок/деревня/хутор);
6. name_locality — символьное поле, название места жительства;
7. street — символьное поле, улица;
8. house — целочисленное поле, номер дома;
9. building — целочисленное поле, номер корпуса/строения;
10. porch — целочисленное поле, номер подъезда;
11. flat — целочисленное поле, номер квартиры.

Таблица Marriage

Содержит информацию о супружестве людей.

Таблица хранит следующие поля:

1. id — целочисленное поле, идентификатор супружества;
2. id_doc — целочисленное поле, идентификатор документа, к которому привязана запись в таблице Marriage;
3. time_start — поле, хранящее дату и время заключения брака;
4. time_end — поле, хранящее дату и время расторжения брака;
5. id_partner — целочисленное поле, хранящее идентификатор супруга.

Таблица OtherAtribute

Содержит информацию о дополнительных атрибутах.

Таблица хранит следующие поля:

1. id — целочисленное поле, идентификатор дополнительного поля;
2. id_doc — целочисленное поле, идентификатор документа, к которому привязана запись в таблице OtherAtribute;
3. name — символьное поле, название дополнительного атрибута;
4. value — символьное поле, значение дополнительного атрибута.

Ниже на рис. 2.3 приведена спроектированная база данных:

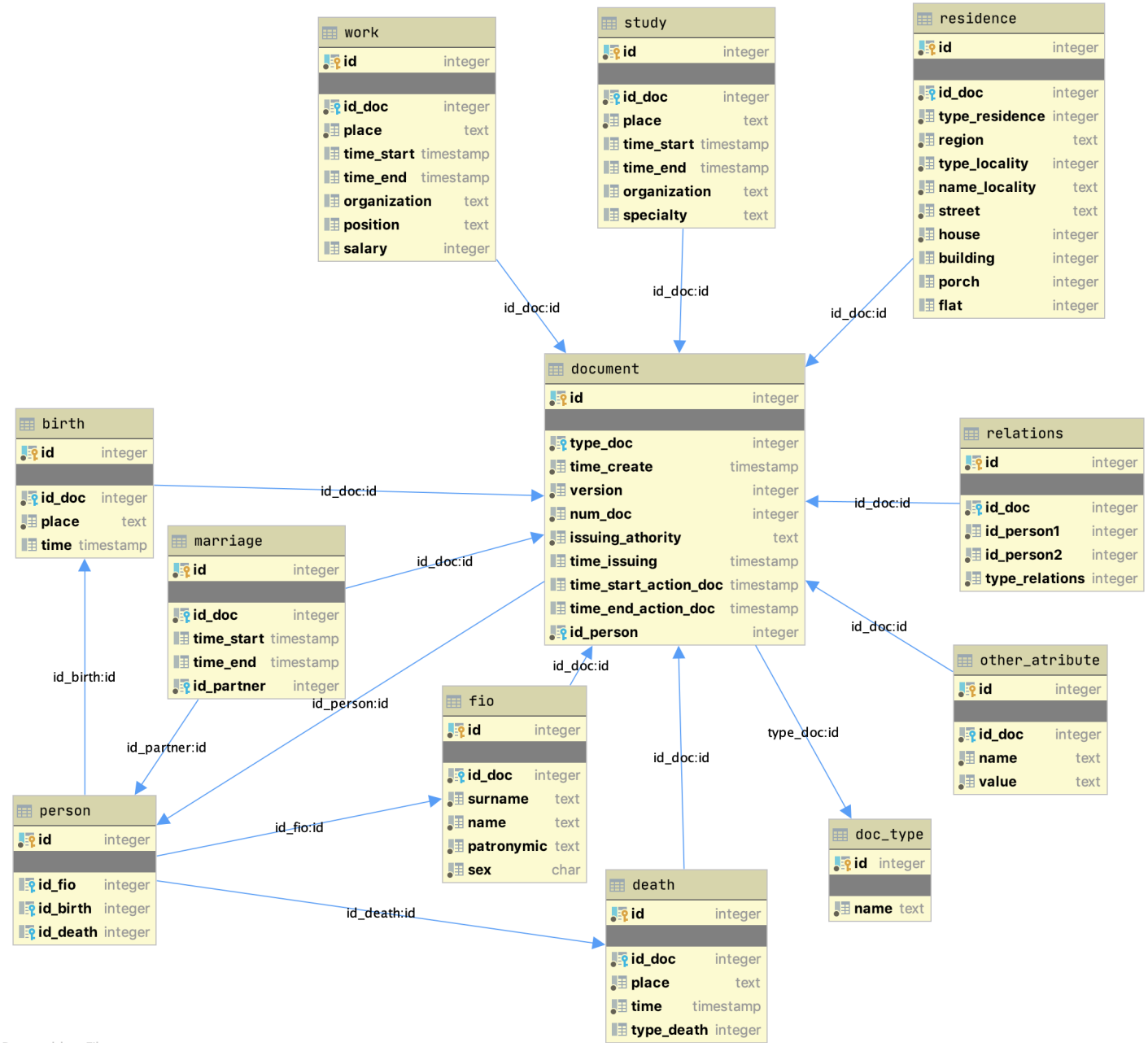

Рисунок 2.3. Диаграмма базы данных.

2.2 Диаграмма классов

Ниже на рисунке 2.4 представлена диаграмма компонентов всего приложения, а рисунках 2.5-2.8 представлены диаграммы классов для клиент-серверного приложения. Все приложение можно разбить на 3 компоненты:

1. компонента доступа к БД;
2. компонента бизнес-логики;
3. компонента графического пользовательского интерфейса (GUI).

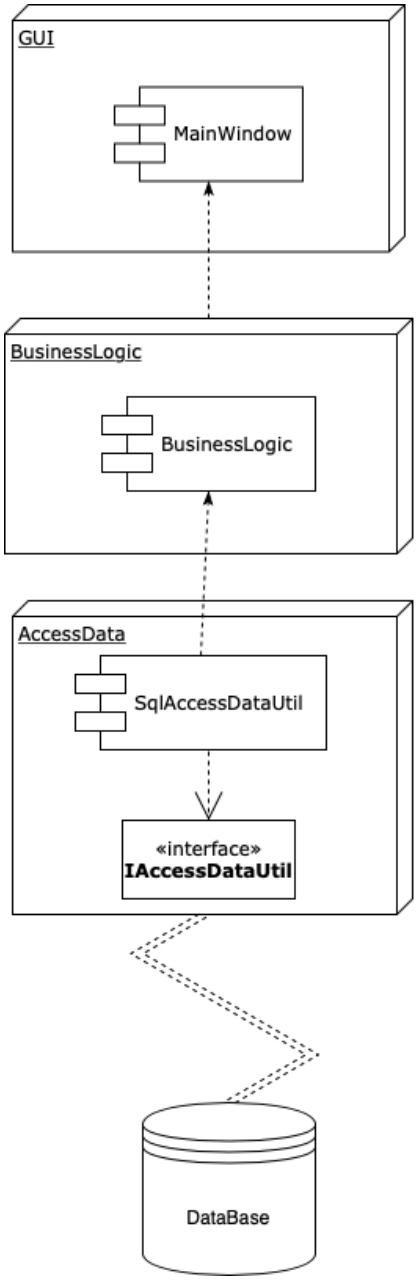

Рисунок 2.4. Диаграмма компонентов клиент-серверного приложения.

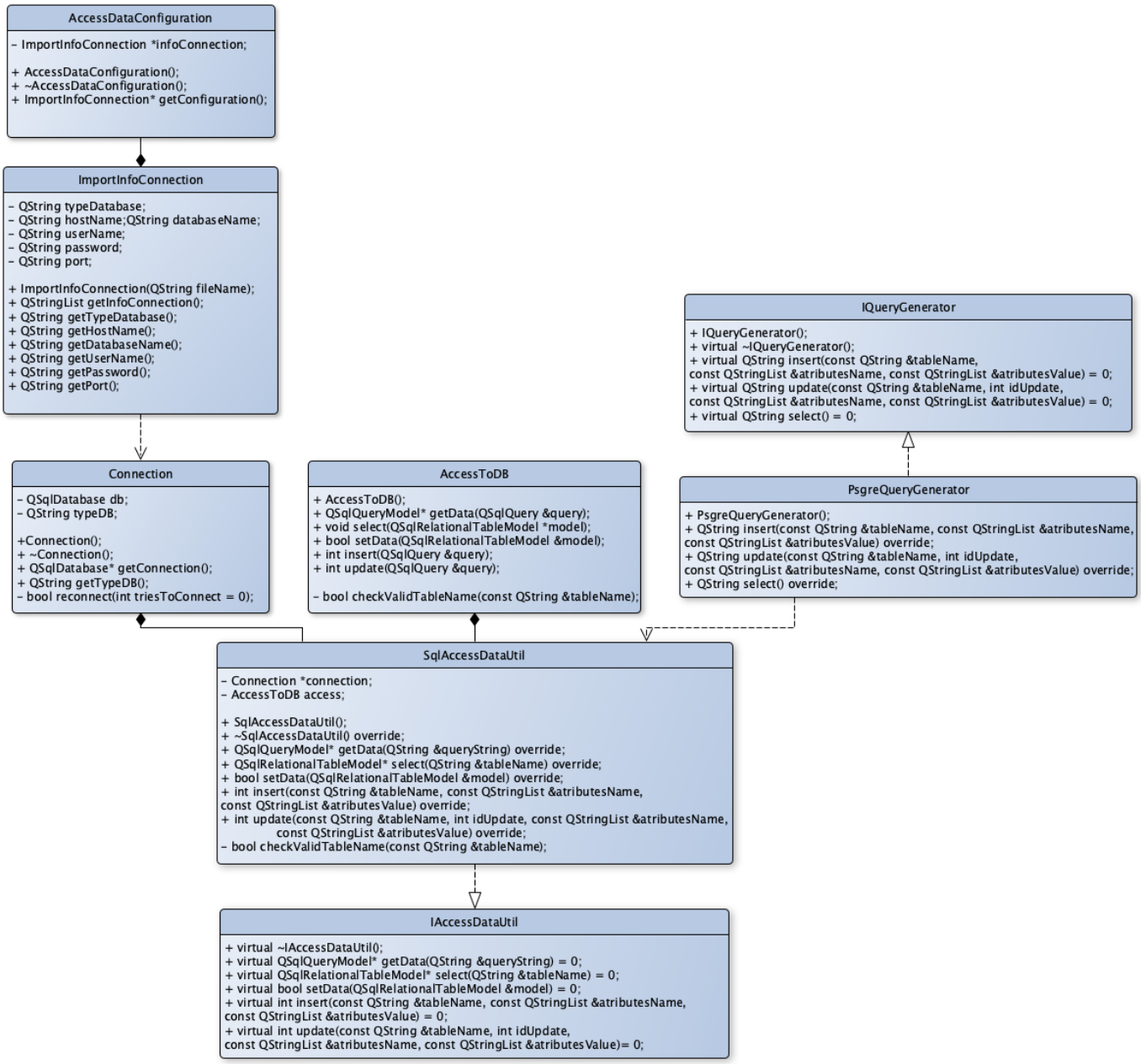

Рисунок 2.5. Диаграмма классов для компоненты доступа к БД.

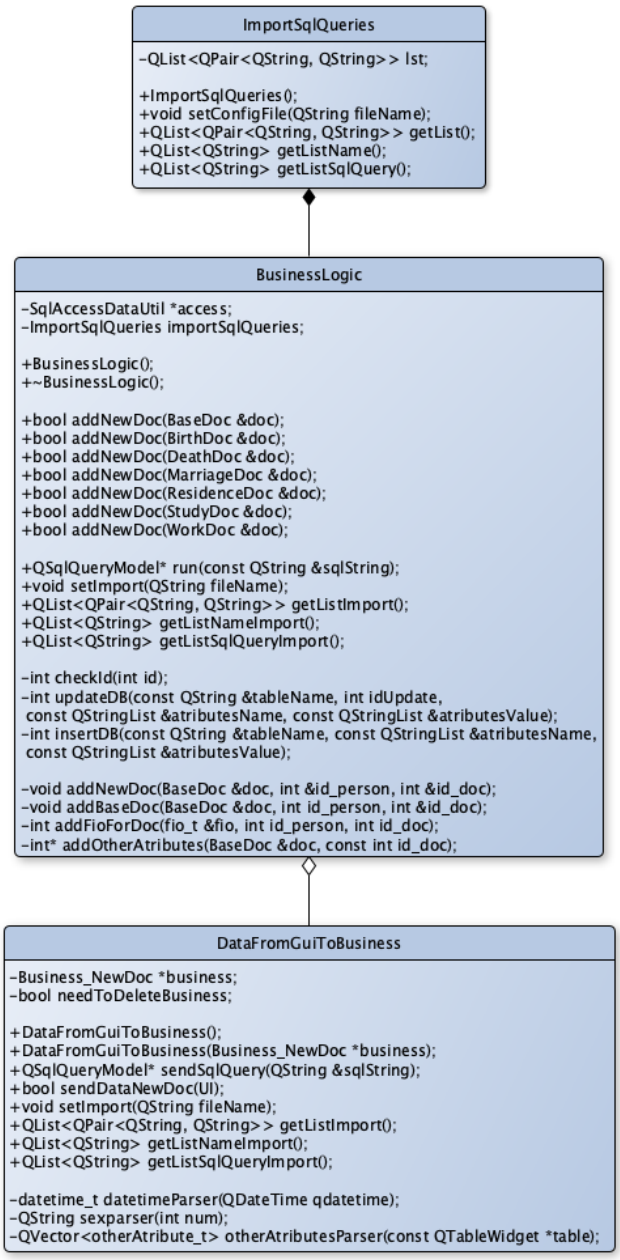

Рисунок 2.6. Диаграмма классов для компоненты бизнес-логики.

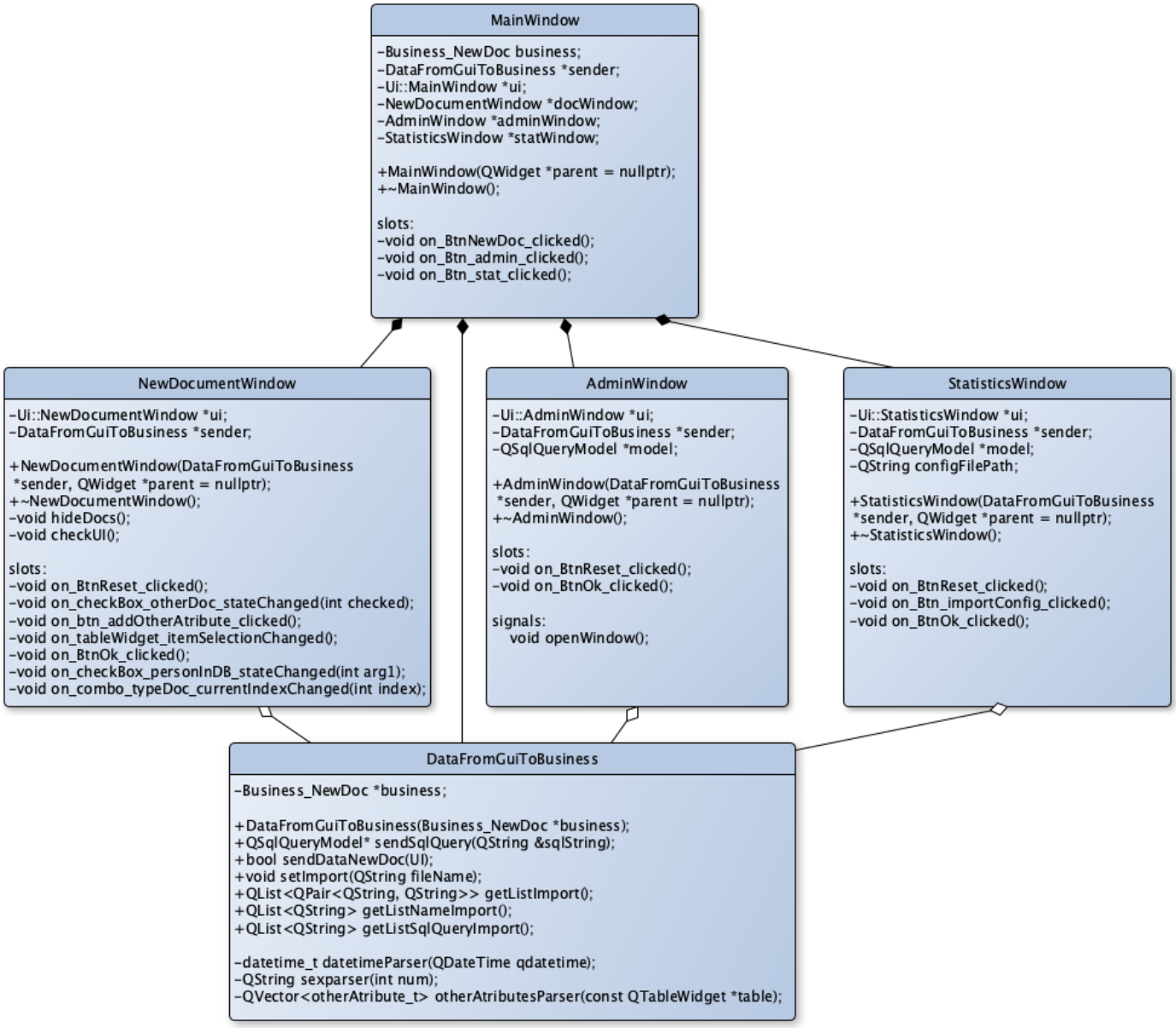

Рисунок 2.7.Диаграмма классов для компоненты GUI и модуля, связывающего GUI и бизнес-логику.

Вывод

В данном разделе рассмотрены диаграммы классов основных компонент клиент-серверного приложения, а также спроектирована БД.

3. Технологический раздел

В разделе рассмотрен выбор средств программной реализации, описаны основные моменты программной реализации.

3.1 Выбор средств программной реализации

В качестве языка программирования выбран язык C++. Этот язык поддерживает объектно-ориентированный подход программирования, что позволяет естественным образом декомпозировать задачу и легко модифицировать программу. Язык C++ содержит встроенные библиотеки для работы с многопоточностью, что позволяет использовать больше ресурсов компьютера.

Для реализации проекта выбрана среда программирования Qt Creator 4.8.2. Данная среда обладает удобным редактором кода и отладчиком, широким набором настроек, а также содержит встроенные библиотеки, облегчающие работу с данными и БД.

В качестве СУБД выбран PostgreSQL ввиду простоты масштабирования, мильтиплатформенности, поддержки параллельных процессов и высокой безопасности аутентификации.

3.2 Описание входных данных

В данной программе входными данными являются файлы с расширением .txt, в них содержится конфигурационная информация (имя пользователя, пароль, ip-адрес и т.д.) для подключения к БД, а так же файлы с SQL-запросами к БД для получения статистики.

Формат конфигурационного файла для подключения (построчно):

1. тип БД;
2. имя хоста;
3. имя БД;
4. имя пользователя;
5. пароль;
6. номер порта.

Формат файла с SQL-запросами:

- нечетные строки — название SQL-запроса, объяснение, что он делает;
- четные строки — сам SQL-запрос.

3.3 Инструкция по запуску программного обеспечения

Для корректного запуска ПО необходимо поместить конфигурационный файл соединения в одну директорию с приложением. Для получения статистики необходимо с помощью GUI выбрать файл с SQL-запросами.

3.4 Описание интерфейса программы

На рисунке 3.1-3.3 представлен интерфейс программы.

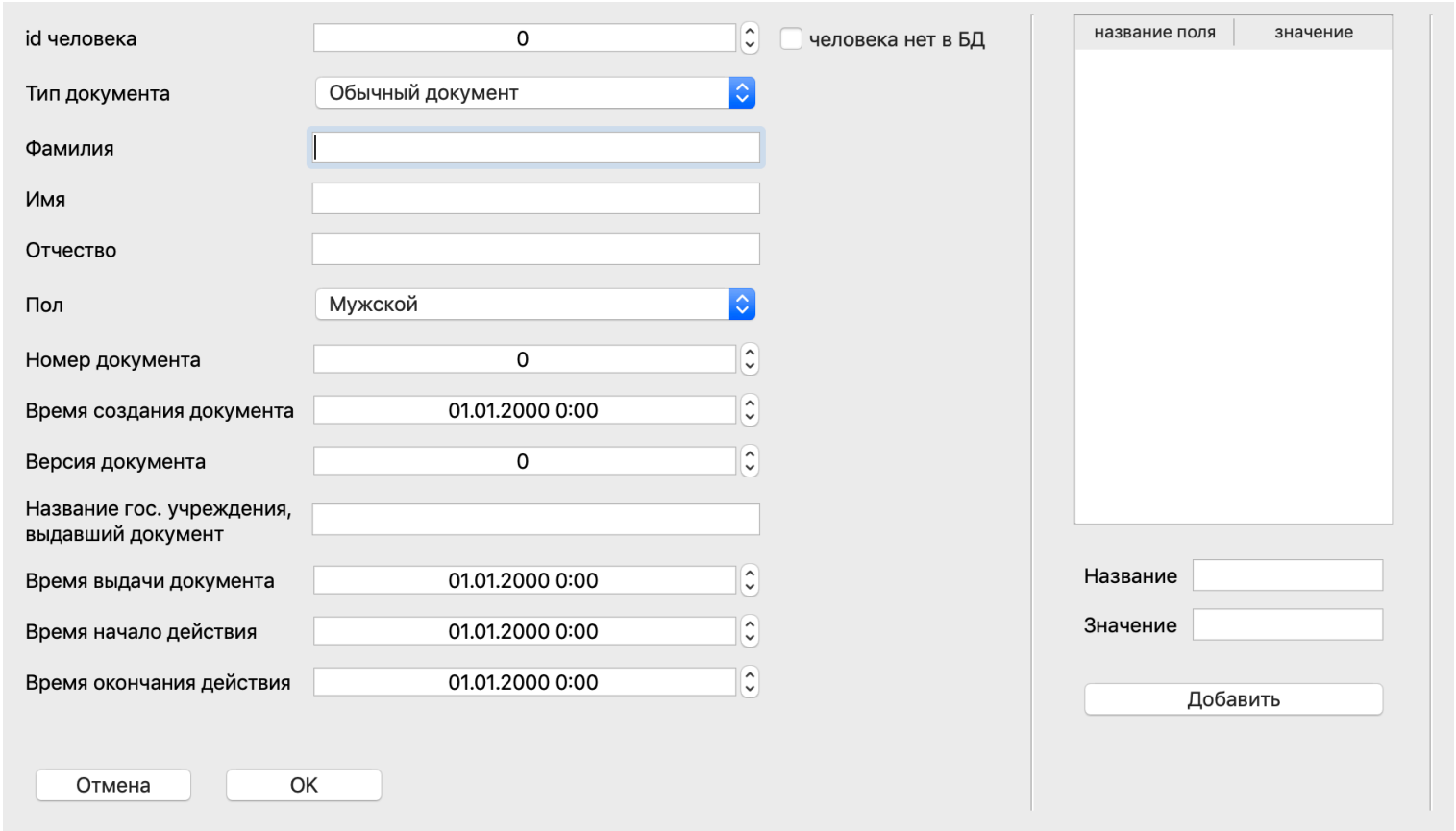

Рисунок 3.1. Интерфейс ПО для добавления документа.

Пользователю доступна возможность выбора типа документа, при необходимости ввода дополнительной информаци в интерфейсе появляются дополнительные поля ввода. При отсутствии необходимых полей ввода пользователь может ввести дополнительные названия и значения полей, которые будут помещены в таблицу OtherAtribute.

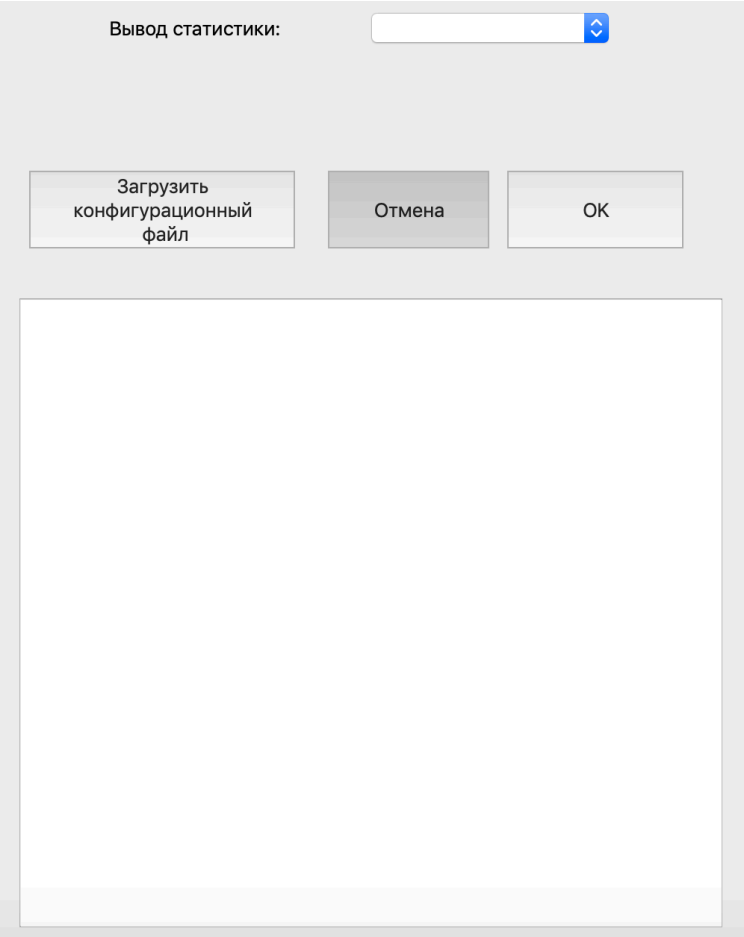

Рисунок 3.2. Интерфейс ПО для получения статистики.

Пользователю доступна возможность выбора текстового файла с SQL-запросами, после чего он может выбрать какую статистику необходимо получить. Результат отображается в виде таблицы.

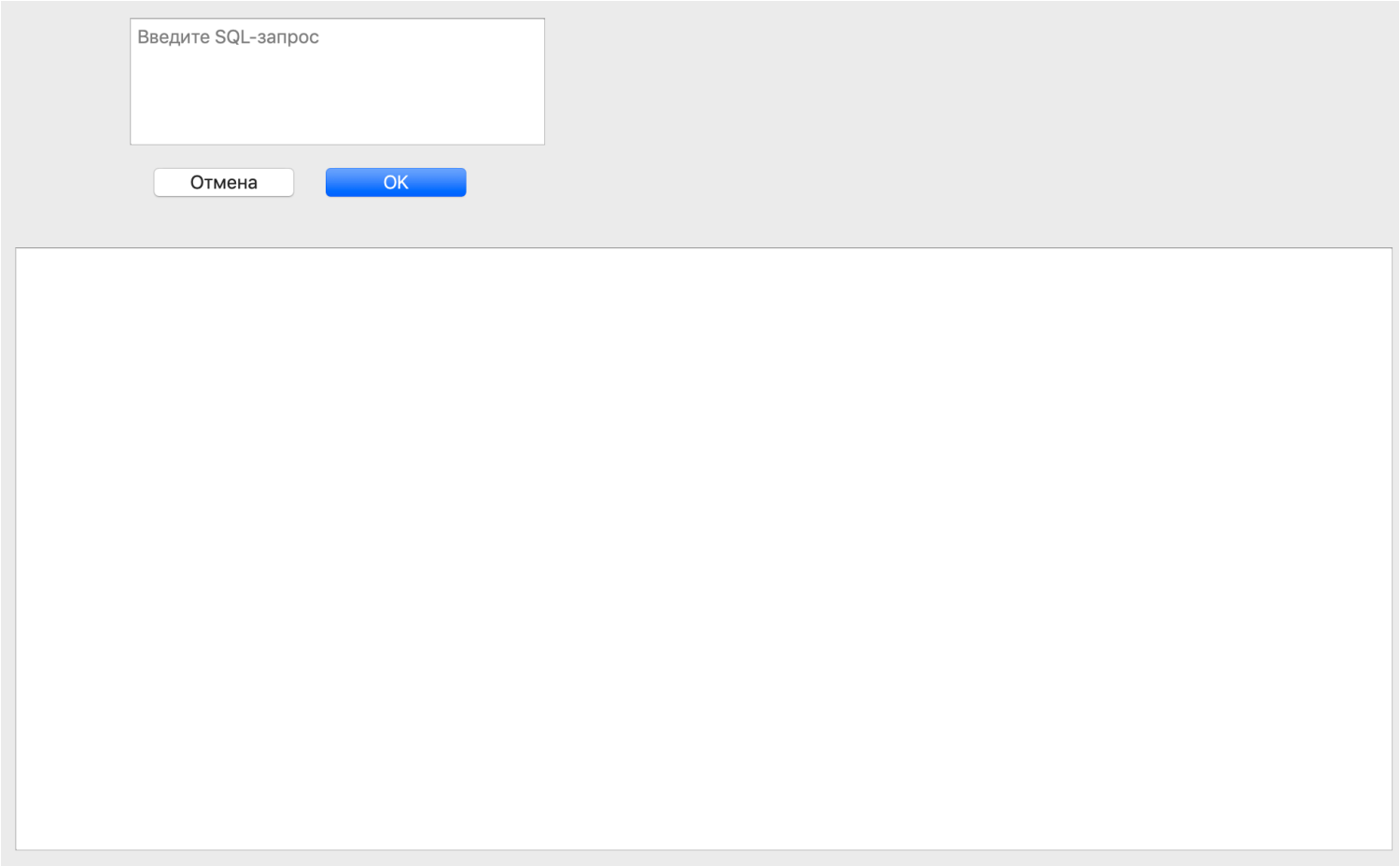

Рисунок 3.3. Интерфейс ПО для администрирования.

Пользователю доступна возможность ввода SQL-запроса и получение результата в виде таблицы.

3.5 Описание основных моментов реализации

Ниже в листингах 3.1-3.2 описаны основные моменты реализации.

Листинг 3.1. Методы компоненты бизнес-логики по обработке данных из документа и отправке данных в различные таблицы через компоненту доступа к БД.

```
1.   bool BusinessLogic::addNewDoc(BaseDoc &doc)
2.   {
3.       try
4.       {
5.           if (!access->transaction())
6.               throw "Unable to start transaction";
7.
8.           int id_person, id_doc;
9.           addNewDoc(doc, id_person, id_doc);
10.
11.          if (!access->commit())
12.              throw "Unable to commit transaction";
13.      }
14.      catch (...)
15.      {
```

```
16.           qDebug() << "Error in adding new BaseDoc";
17.           access->rollback();
18.       }
19.
20.       return true;
21.   }
22.   void BusinessLogic::addNewDoc(BaseDoc& doc, int &id_person, int &id_doc)
23.   {
24.       id_person = checkId(doc.id_person);
25.       if (id_person < 0)
26.           throw "Error in searching/creating human in table person";
27.
28.       addBaseDoc(doc, id_person, id_doc);
29.   }
30.   int BusinessLogic::checkId(int id)
31.   {
32.       if (id != -1)
33.           return id;
34.
35.       id = insertDB("person", QStringList({"id_fio", "id_birth", "id_death"}),
      QStringList({"NULL", "NULL", "NULL"}));
36.
37.       return id;
38.   }
39.   void BusinessLogic::addBaseDoc(BaseDoc &doc, int id_person, int &id_doc)
40.   {
41.       QStringList lst({timeparser(doc.time_create),
42.                        QString(QString::number(doc.version)),
43.                        QString(QString::number(doc.type_doc)),
44.                        QString(QString::number(doc.num_doc)),
45.                        QString(doc.issuing_athority),
46.                        timeparser(doc.time_issuing),
47.                        timeparser(doc.time_start_action_doc),
48.                        timeparser(doc.time_end_action_doc),
49.                        QString(QString::number(id_person))});
50.       listparser(lst);
51.
52.       id_doc = insertDB("document", QStringList({"time_create", "version",
      "type_doc", "num_doc", "issuing_athority", "time_issuing",
53.                                                  "time_start_action_doc",
      "time_end_action_doc", "id_person"}), lst);
54.       if (id_doc < 0)
55.           throw "Unable to create record in table document";
56.
57.       addFioForDoc(doc.fio, id_person, id_doc);
58.
59.       int *ids_otherAtributes = addOtherAtributes(doc, id_doc);
60.       delete[] ids_otherAtributes;
61.   }
```

Листинг 3.2. Методы компоненты доступа к БД по отправке значений в заданную таблицу.

```
1.    int SqlAccessDataUtil::insert(const QString &tableName, const QStringList
      &atributesName, const QStringList &atributesValue)
2.    {
3.        QSqlDatabase *db = nullptr;
4.        int res = 0;
5.        try
6.        {
7.            db = connection->getConnection();
8.            if (!db || !checkValidTableName(tableName))
9.                throw -1;
10.
11.           QSqlQuery query = QSqlQuery(*db);
12.           QString type = connection->getTypeDB();
13.           if (type == "QPSQL")
```

```
14.          {
15.              PsgreQueryGenerator querygen;
16.              query.prepare(querygen.insert(tableName, atributesName, atributes-
     Value));
17.          }
18.          else
19.              throw -2;
20.
21.          res = access.insert(query);
22.      }
23.      catch (int err)
24.      {
25.          switch (err)
26.          {
27.          case -1:
28.          {
29.              qDebug() << "Error in insert connecting to db or invalid tableName";
30.              res = -1;
31.              break;
32.          }
33.          case -2:
34.          {
35.              qDebug() << "Error in insert, unknown type DB";
36.              res = -2;
37.              break;
38.          }
39.          default:
40.          {
41.              qDebug() << "unknown error in insert";
42.              res = -3;
43.          }
44.          }
45.      }
46.      catch (...)
47.      {
48.           qDebug() << "unknown error in insert" << __FILE__ << "Line: " <<
     __LINE__ << endl;
49.           res = -4;
50.      }
51.
52.      return res;
53.  }
54.
55.  int AccessToDB::insert(QSqlQuery &query)
56.  {
57.      int res = 0;
58.      try
59.      {
60.          run(query);
61.
62.          query.next();
63.          res = query.record().value(0).toInt();
64.      }
65.      catch (...)
66.      {
67.          qDebug() << query.lastError() << "\nFile: " << __FILE__ << "\nLine: " <<
     __LINE__ << endl;
68.          res = -1;
69.      }
70.
71.      return res;
72.  }
```

Вывод

В данном разделе были выбраны средства реализации, рассмотрен интерфейс программы, а также был представлен листинг основных моментов реализации.

Заключение

В ходе работы был формализован необходимый функционал, проведен анализ существующих моделей данных, спроектирована и реализована база данных с использованием выбранной СУБД PostgreSQL, реализовано клиент-серверное приложение для взаимодействия с БД.

Список литературы

1. *Дейт К.Дж.* Введение в системы баз данных = Introduction to Database Systems. — 8-е изд. — М.: Вильямс, 2005. — 1328 с. — ISBN 5-8459-0788-8 (рус.) 0-321-19784-4 (англ.).
2. *Когаловский М. Р.* Энциклопедия технологий баз данных. — М.: Финансы и статистика, 2002. — 800 с. — ISBN 5-279-02276-4.Фень Юань. Программирование графики для Windows. -М.: 2004.
3. *Коннолли Т., Бегг К.* Базы данных. Проектирование, реализация и сопровождение. Теория и практика = Database Systems: A Practical Approach to Design, Implementation, and Management. — 3-е изд. — М.: Вильямс, 2003. — 1436 с. — ISBN 0-201-70857-4.
4. *Кузнецов С.Д.* Основы баз данных. — 2-е изд. — М.: Интернет-университет информационных технологий; БИНОМ. Лаборатория знаний, 2007. — 484 с. — ISBN 978-5-94774-736-2.
5. Документация по среде разработки Qt [Электронный ресурс]. – Режим доступа: https://doc.qt.io, свободный – (20.03.2020).