МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Вятский государственный университет»

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Допущено к защите

Руководитель проекта

\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_/

(подпись) ( Ф.И.О.)

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_2021г.

РАЗРАБОТКА ПРОГРАММЫ ДЛЯ ЭКСПОРТА ДАННЫХ ИЗ EXCEL-ФАЙЛА В БАЗУ ДАННЫХ

Пояснительная записка курсового проекта по дисциплине

«Технология разработки программного обеспечения»

ТПЖА.090301.869 ПЗ

Разработал студент группы ИВТм-11\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Баташев П.А./

Руководитель, доцент кафедры ЭВМ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Долженкова М.Л/

Работа защищена с оценкой «\_\_\_\_\_\_\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(оценка)(дата)

Члены комиссии \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

(подпись) (Ф.И.О)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

(подпись) (Ф.И.О)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

(подпись) (Ф.И.О)

Киров 2021

Реферат

Баташев П.А. Разработка программы для экспорта данных из Excel-файла в базу данных. ТПЖА.090301.869 ПЗ: Курс. проект / ВятГУ, каф. ЭВМ; рук. Долженкова М.Л. - Киров, 2021. – ПЗ 41с , 43 рис., 1 табл., 3 источников, 3 прил.

ЭКСПОРТ ДАННЫХ, EXCEL, EXCEL-ФАЙЛ, ПАРСИНГ, ПАРСИНГ EXCEL-ФАЙЛА, ДЕСКРИПТОР, БАЗА ДАННЫХ, SQL-ЗАПРОС, ГЕНЕРАТОР SQL-ЗАПРОСОВ, МАСТЕР ИМПОРТА И ЭКСПОРТА SQL SERVER, МАКРОСЫ

Цель курсовой работы – разработать программы для экспорта данных из Excel-файла в базу данных.

Ввод данных из файла в базу данных – это рутинный процесс, который при ручном выполнении может приводить к различным ошибкам. Автоматизация данного процесса позволит ускорить перенос данных и избежать различных ошибок.

Средства для экспорта данных из Excel-файла уже существуют, но все они имеют ограничения к структуре файла, например, Microsoft SQL Server Management не способен вычленять отдельные поля на странице, а также работать с несколькими страницами или с несколькими таблицами на одной странице.

Из-за отсутствия аналогов, позволяющих экспортировать из предоставленного Excel-файла с особой структурой, в базу данных, при этом производить не только вставку, но и различные проверки при экспорте, было принято решение разработать программу, имеющую перечисленные возможности.

Разработанная программа позволяет экспортировать данные из Excel-файла с любой структурой данных, при этом достаточно просто указать в дескрипторе файла свойства данных, необходимых для извлечения, а также создать SQL-запрос, в котором могут быть произведены различные действия с извлеченными данными перед вставкой в базу.

**Содержание**

[Введение 5](#_Toc84696209)

[1 Анализ существующих аналогов 6](#_Toc84696210)

[2 Техническое задание 7](#_Toc84696211)

[2.1 Введение 7](#_Toc84696212)

[2.2 Основание для разработки 7](#_Toc84696213)

[2.3 Назначение 7](#_Toc84696214)

[2.4 Требования к программе 7](#_Toc84696215)

[3 Разработка парсера Excel-файла 9](#_Toc84696216)

[3.1 Дескриптор Excel-файла общая информация 9](#_Toc84696217)

[3.2 Парсинг дескриптора Excel-файла 14](#_Toc84696218)

[4 Разработка исполнителя SQL-запросов 15](#_Toc84696219)

[5 Разработка пользовательского интерфейса 17](#_Toc84696220)

[ПРИЛОЖЕНИЕ А 23](#_Toc84696221)

[Шаблон Excel-файла 23](#_Toc84696222)

# Введение

Несмотря на то, что базы данных активно вошли в нашу жизнь и применяются в различных отраслях, есть случаи, когда данные из-за отсутствия построенной базы данных, либо из-за отсутствия доступа к базе данных, либо из-за отсутствия удобного интерфейса взаимодействия с базой данных, необходимо хранить в отдельных файлах. И в тот момент, когда база данных была создана, либо, когда доступ к базе данных появился напрямую или через посредника, то необходимо все накопленные данные, хранящиеся во множестве файлов, экспортировать из файлов в базу данных.

Процесс переноса данных из файла в базу данных рутинный, требующий много времени, внимания и концентрации, поэтому, если файлов много, или они приходят регулярно, то этот процесс необходимо автоматизировать.

При автоматизации процесса экспорта файлов одинаковой структуры в базу данных, может понадобиться возможность работать с различными страницами, с различными представлением данных, таких как одиночные поля в файле, или данные, представленные в виде таблиц, которых в свою очередь может быть несколько на одной странице. Все это требует определенного описания файла (дескриптора), по которому будет происходить поиск определенных полей или таблиц на определенных страницах.

Так же при автоматизации процесса экспорта файлов одинаковой структуры в базу данных, может понадобиться проверка извлеченных данных из файла перед вставкой в базу данных, и это может быть проверка не только ошибок в данных, но еще и сверка с данными из самой базы данных, распределение данных по разным таблицам и так далее. Все это требует определенного запроса, который одинаков для всех файлов и отличается только данными, с которыми работает данный запрос.

# Анализ существующих аналогов

# Техническое задание

## Введение

Требуется написать программу, которая предоставляет средства для экспорта данных из Excel-файла в базу данных, с возможностью изменения шаблона файла и шаблона SQL-запроса.

В качестве языка программирования выбран С#, в качестве среды программирования выбрана Visual Studio, так как она содержит все необходимые компоненты для создания программы с пользовательским интерфейсом.

## Основание для разработки

Основанием для разработки является необходимость переноса информации об оказанных социальных услугах, предоставляемых в Excel-файлах, шаблон которого изображен в приложении А, общественными некоммерческими организациями, у которых отсутствует доступ к базе данных ЭСРН.

## Назначение

Основным назначением программы является автоматизация переноса данных из Excel-файла в базу данных, с возможностью изменять шаблон Excel-файла и SQL-запрос, без изменения программного кода.

## Требования к программе

Требования к функциональным характеристикам:

* экспорт данных из Excel-файла в базу данных;
* наличие пакетной обработки файлов;
* возможность изменения шаблона Excel-файла без изменения программного кода;
* возможность изменения SQL-запроса без изменения программного кода;
* подробное описание ошибок на всех этапах экспорта данных.

Требования к минимальной конфигурации:

* минимальное разрешение экрана 800x600;
* наличие компьютерной мыши;
* наличие клавиатуры.

Требования к программной совместимости:

* операционная система: Windows 7,8,10.
  1. Требования к документации

Документация должна содержать:

* тексты программ со всеми необходимыми комментариями;
* пояснительную записку, содержащей описание разработки;
* руководство пользователя.

# Разработка парсера Excel-файла

Парсинг – это метод индексирования информации с последующей конвертацией ее в иной формат или даже иной тип данных. Парсинг позволяет взять файл в одном формате и преобразовать его данные в более удобоваримую форму, которую можно использовать в своих целях.

В нашем случае парсинг это процесс обработки данных из Excel-файла. Он подразумевает анализ текста, вычленение оттуда необходимых материалов и их преобразование в подходящий вид. Благодаря парсингу можно находить на страницах файла небольшие клочки полезной информации и в автоматическом режиме их оттуда извлекать, чтобы потом переиспользовать.

Вычленяемая информация в нашем случае это будет та информация, которая нужна для вставки в SQL-скрипт. Представлена она может быть как отдельными полями, так и табличными значениями.

## 3.1 Дескриптор Excel-файла общая информация

Под дескриптором Excel-файла будем понимать ту информацию, которая описывает извлекаемые объекты из файла. Дескриптор состоит из объектов и их атрибутов. На данный момент есть три вида объектов, которые заключаются в открывающий и закрывающий тег:

* <singleValue>…</singleValue> - информация об значении, хранившемся в одном поле;
* <table>…</table> - информация об таблице, в которой хранятся значения;
* <column>…</column> - является вложенным тегом в тег table и является информацией о столбце таблицы, в котором хранятся данные.

Атрибуты в свою очередь представляются в виде токенов. Под токеном подразумевается значение и короткого описания этого значения (имя токена). Токен представляется следующим образом:

Имя токена : значение токена;

Каждый токен объекта должен быть отделен точкой с запятой, при этом, если имя или значение имеют пробелы и табуляцию, то они должны быть заключены в кавычки. Таким образом дескриптор объекта, который имеет N атрибутов и один вложенный тег, выглядит следующим образом:

<tag>

nameToken\_1 : valueToken\_1;

nameToken\_2 : valueToken\_2;

…

nameToken\_N : valueToken\_N;

<nestedObject>

nameToken\_1 : valueToken\_1;

nameToken\_2 : valueToken\_2;

…

nameToken\_N : valueToken\_N;

</nestedObject>

</tag>

В таблице 1 представлены возможные имена атрибутов тега singleValue.

Таблица 1 – Возможные атрибуты тега singleValue

|  |  |  |  |
| --- | --- | --- | --- |
| Коды атрибутов | Допустимые значения | Описание | Если значение не указано |
| SHEET\_NUMBER | Целочисленные числа, больше 0 | Номер страницы, на которой производится поиск (отсчет страниц ведется с 1). | Поиск производится на всех страницах |
| SHEET\_NAME | Любая строка | Имя страницы, на которой производится поиск. | Поиск производится на всех страницах |
| SECTION\_NAME | Любая строка | Раздел на странице, относительно которого производится поиск (для полей с одинаковыми надписями). | Поиск производится на всей странице |
| SECTION\_BOTTOM\_LEFT | 1 или 0 | При условии, что атрибут SECTION\_NAME определен. Область снизу слева от раздела включать в поиск, рисунок 1. | Поиск не производится в левой нижней части от раздела (Аналогично значению 0). |
| SECTION\_BOTTOM\_RIGHT | 1 или 0 | При условии, что атрибут SECTION\_NAME определен. Область снизу справа от раздела включать в поиск, рисунок 1. | Поиск не производится в правой нижней части от раздела (Аналогично значению 0). |
| SECTION\_UP\_LEFT | 1 или 0 | При условии, что атрибут SECTION\_NAME определен. Область сверху слева от раздела включать в поиск, рисунок 1. | Поиск не производится в левой верхней части от раздела (Аналогично значению 0). |
| SECTION\_UP\_RIGHT | 1 или 0 | При условии, что атрибут SECTION\_NAME определен. Область сверху справа от раздела включать в поиск, рисунок 1. | Поиск не производится в правой верхней части от раздела (Аналогично значению 0). |
| FIELD | Любая строка | Текст ячейки, относительно которой ищется значение. | Аналогично пустой ячейке. |
| CODE | Любая строка | Код значения, по которому можно идентифицировать данное поле. | Значение данного объекта будет получено, но обратиться к нему будет нельзя в SQL-скрипте. |
| OFFEST\_ROW | Любое целочисленное число | Смещение по строке относительно FIELD. | Аналогично значению 0 |
| OFFEST\_COLUMN | Любое целочисленное число | Смещение по столбцу относительно FIELD. | Аналогично значению 0 |

В таблице 2 представлены возможные имена атрибутов тега table.

Таблица 2– Возможные атрибуты тега table

|  |  |  |  |
| --- | --- | --- | --- |
| Коды атрибутов | Допустимые значения | Описание | Если значение не указано |
| SHEET\_NUMBER | Целочисленные числа, больше 0 | Номер страницы, на которой производится поиск (отсчет страниц ведется с 1). | Поиск производится на всех страницах |
| SHEET\_NAME | Любая строка | Имя страницы, на которой производится поиск. | Поиск производится на всех страницах |
| SECTION\_NAME | Любая строка | Раздел на странице, относительно которого производится поиск (для полей с одинаковыми надписями). | Поиск производится на всей странице |
| SECTION\_BOTTOM\_LEFT | 1 или 0 | При условии, что атрибут SECTION\_NAME определен. Область снизу слева от раздела включать в поиск, рисунок 1. | Поиск не производится в левой нижней части от раздела (Аналогично значению 0). |
| SECTION\_BOTTOM\_RIGHT | 1 или 0 | При условии, что атрибут SECTION\_NAME определен. Область снизу справа от раздела включать в поиск, рисунок 1. | Поиск не производится в правой нижней части от раздела (Аналогично значению 0). |
| SECTION\_UP\_LEFT | 1 или 0 | При условии, что атрибут SECTION\_NAME определен. Область сверху слева от раздела включать в поиск, рисунок 1. | Поиск не производится в левой верхней части от раздела (Аналогично значению 0). |
| SECTION\_UP\_RIGHT | 1 или 0 | При условии, что атрибут SECTION\_NAME определен. Область сверху справа от раздела включать в поиск, рисунок 1. | Поиск не производится в правой верхней части от раздела (Аналогично значению 0). |
| CODE | Любая строка | Код таблицы, по которому можно идентифицировать данную таблицу. | Значения данной таблицы будут получены, но обратиться к ним будет нельзя в SQL-скрипте. |
| INCLUDE\_FINAL\_ROW | 1 или 0 | Записывать последнюю строку таблицы, которая определяется во вложенных объектах column. | Аналогично значению 0 |

В таблице 3 представлены возможные имена атрибутов тега column.

Таблица 3– Возможные атрибуты тега table

|  |  |  |  |
| --- | --- | --- | --- |
| Коды атрибутов | Допустимые значения | Описание | Если значение не указано |
| NAME | Любая строка | Заголовок столбца. | Поиск производится на всех страницах |
| CODE | Любая строка | Код столбца, по которому можно идентифицировать данный столбец. | Значения данного столбца будут получены, но обратиться к ним будет нельзя в SQL-скрипте. |
| FINAL\_CELL | Любая строка | Значение, говорящее, что данная ячейка последняя. Последней строкой в таблице считается та строка, когда у всех столбцов соответствующая ячейка равняется ее конечному значению. Если нужно указать, что пустая ячейка, то нужно указать двойные кавычки, идущие подряд без пробела. | Любой текст в ячейке является сигналом конечной строки. |

На рисунке 1 изображены области для поиска относительно раздела при соответственно установленных флагов.



Рисунок 1 – Области для поиска относительно раздела

## 3.2 Парсинг дескриптора Excel-файла

Парсинг дескриптора Excel-файла происходит следующим образом. Происходит проход по дескриптору, пока не будет найден значащий символ (любой символ кроме табуляции и пробелов). Далее определяется, является ли данный символ началом тега. Если нет, то выдается ошибка, иначе читается имя тега и происходит дальнейший проход. После тега так же ищется первый значащий символ. Далее определяется, является ли данный символ началом вложенного тега или окончанием текущего тега. Если ни тем, ни другим не является, то считается, что встретили атрибут объекта. Для атриубта берется подстрока от текущей позиции до символа точки с запятой, и из данной подстроки извлекается пара имени и значения токена, после чего прочитанный токен помещается в список токенов текущего объекта. Если же встретился вложенный тег, то обработка его происходит аналогично тому, как было описано выше, только после встречи закрывающего тега вложенного объекта происходит добавление прочитанного вложенного объекта в список вложенных объектов того объекта, в котором этот вложенный объект был встречен. Если же встречен закрывающий тег текущего объекта, то данный объект помещается в список дескриптора объекта и происходит дальнейший поиск тега, либо окончания дескриптора Excel-файла. Описанный алгоритм отображен на рисунке 2.



Рисунок 2 – Алгоритм получения дескриптора объекта

# Разработка исполнителя SQL-запросов

Задача исполнителя SQL-запросов заключается в том, чтобы занести прочитанные данные из Excel-файла в готовый SQL-запрос и отправить на сервер отредактированный запрос на выполнение.

Место для вставки одиночных данных в SQL-запрос должно представлять из себя код одиночного значения, заключенного в символы решетки. Например, была получена дата из Excel файла, которой был присвоен код dateReport. Место, в которое должно быть вставлено это значение, должно выглядеть следующим образом:

DECLARE @date = CONVERT(DATE, '#dateReport#')

И тогда, перед тем, как исполнить SQL-запрос, данный код будет заменен прочитанным значением, например 01-01-2021:

DECLARE @date DATE = CONVERT(DATE, '01-01-2021')

Для табличных значений необходимо на место, где они должны быть вставлены, вставить шаблон строки, которая будет копироваться и вставляться на место данного шаблона. При этом шаблон должен быть заключен в символы «<» и «>» и в начале иметь код таблицы, а далее в скобках указаны столбцы таблицы, заключенные в символы решетки. Допустим была прочитана таблица, код для которой был указан dataForInsert. Данная таблица имеет два столбца с кодами dateInfo и Info, тогда шаблон для вставки данных значений в SQL-запрос будет выглядеть следующим образом:

<dataForInsert (CONVERT(DATE, '#dateInfo#'), '#Info#')>

И тогда, перед тем, как исполнить SQL-запрос, данный шаблон будет заменен прочитанными значениями, например:

(CONVERT(DATE, '01-01-2021'), 'Новый год')

(CONVERT(DATE, '23-02-2021'), 'День защитника отечества')

При составлении SQL-запроса нужно учитывать, что при вставки значений не учитываются типы данных, поэтому, если данные должны быть вставлены как строка, то необходимо код, заключенный в символы решетки, заключить еще в одинарные кавычки. А если, к примеру, данные должны быть вставлены как дата, то нужно еще поместить шаблон в конструкцию по конвертации строки в дату, как это было показано выше.

# Разработка пользовательского интерфейса

Главное окно разработанной программы содержит в себе две основные секции – меню и панель обработки файлов, рисунок 3. При загрузке файлов отображается прогресс загрузки файлов, а также статус. При ошибке чтения фала в поле дополнительной информации отображается причина ошибки, рисунок 4.

После загрузки файла можно посмотреть данные, которые были извлечены парсером. Одна вкладка с одиночными данными, рисунок 6, и вкладки с табличными значениями, рисунок 7. Так же после загрузки файла можно посмотреть сгенерированный запрос со вставленными значениями, рисунок 8.

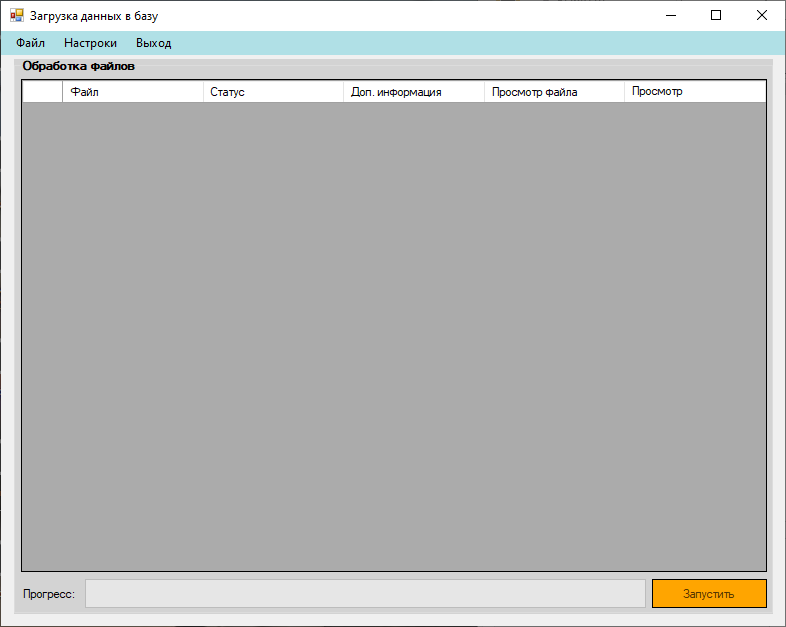


Рисунок 3 – Главное окно программы

На меню присутствуют следующие кнопки:

* Файл – при нажатии выпадает подпункты, где можно выбрать один файл или целую папку для обработки;
* Настройки – при нажатии на которую можно в открывшемся окне посмотреть результат парсинга дескриптора Excel-файла и шаблон SQL-запроса, а так же установить собственные файлы. По умолчанию файлы берутся из папки Source, которая располагается в папке с исполняющим файлом. По умолчанию файлы берутся с названием descriptor.txt и Query.sql;
* Выход – выход из программы.

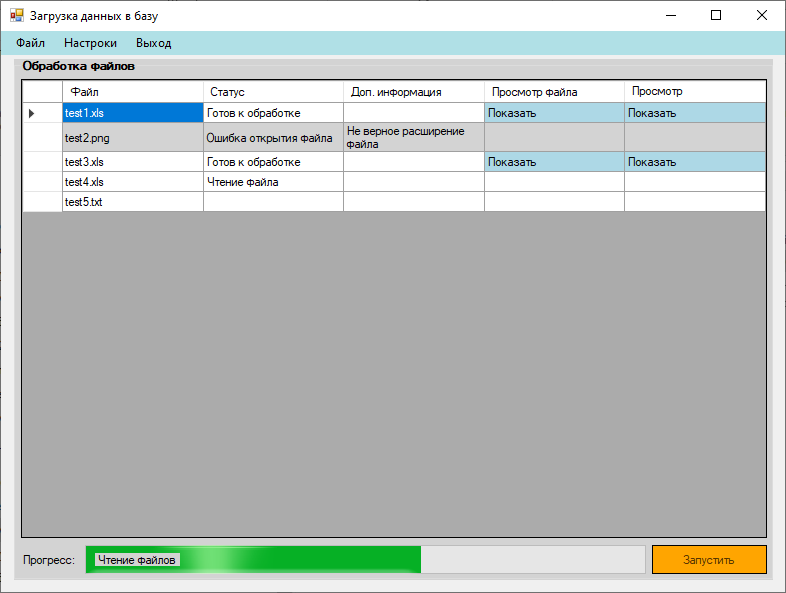


Рисунок 4 – Загрузка файлов

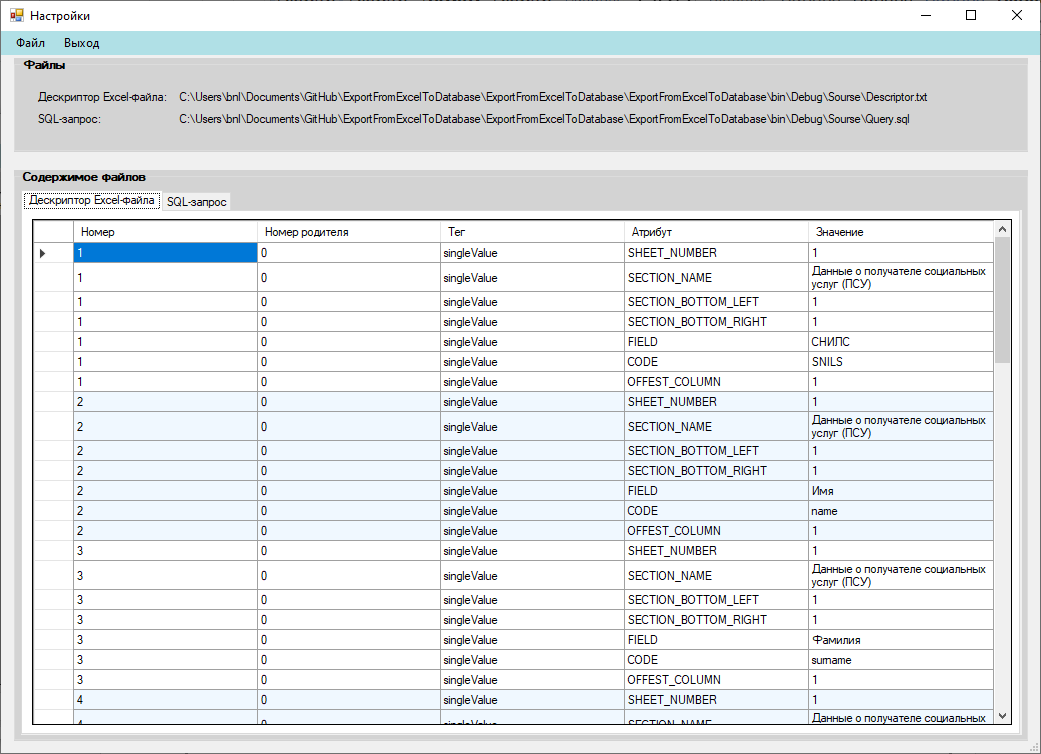


Рисунок 4 – Настройки (вкладка с отображением дескриптора)

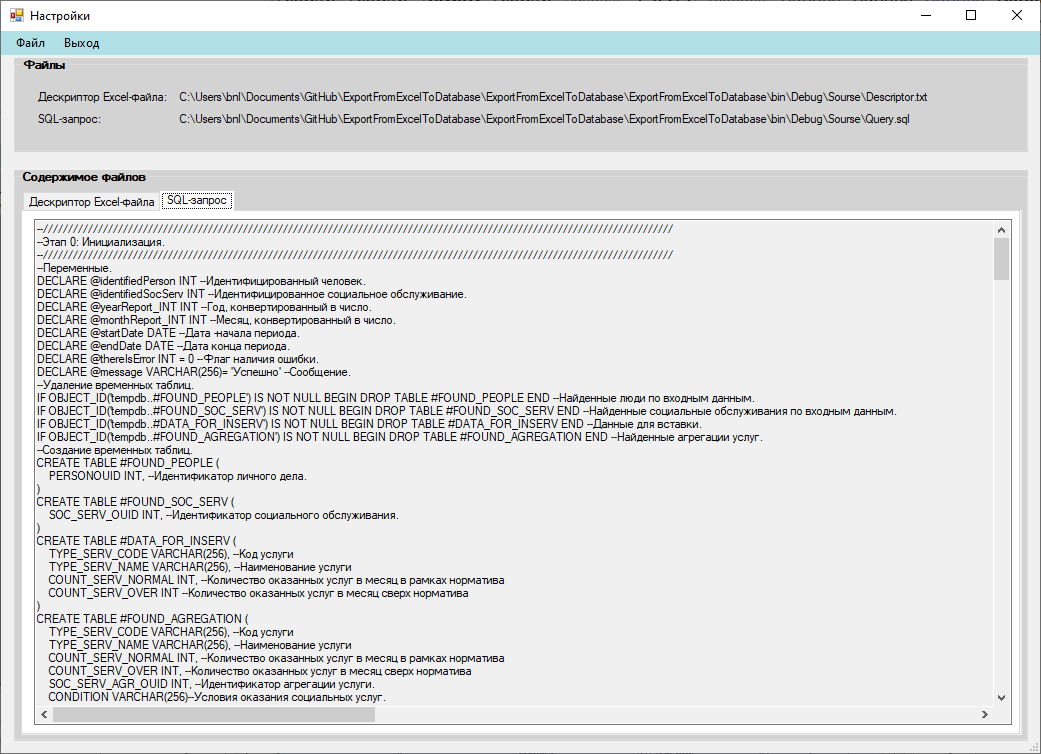


Рисунок 5 – Настройки (вкладка с отображением запроса)

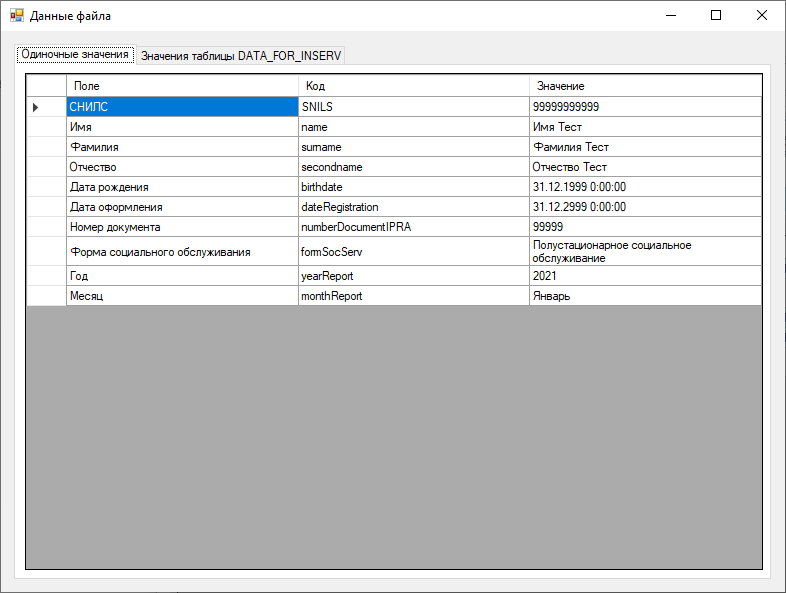


Рисунок 6 – Просмотр извлеченных данных (вкладка с одиночными значениями)

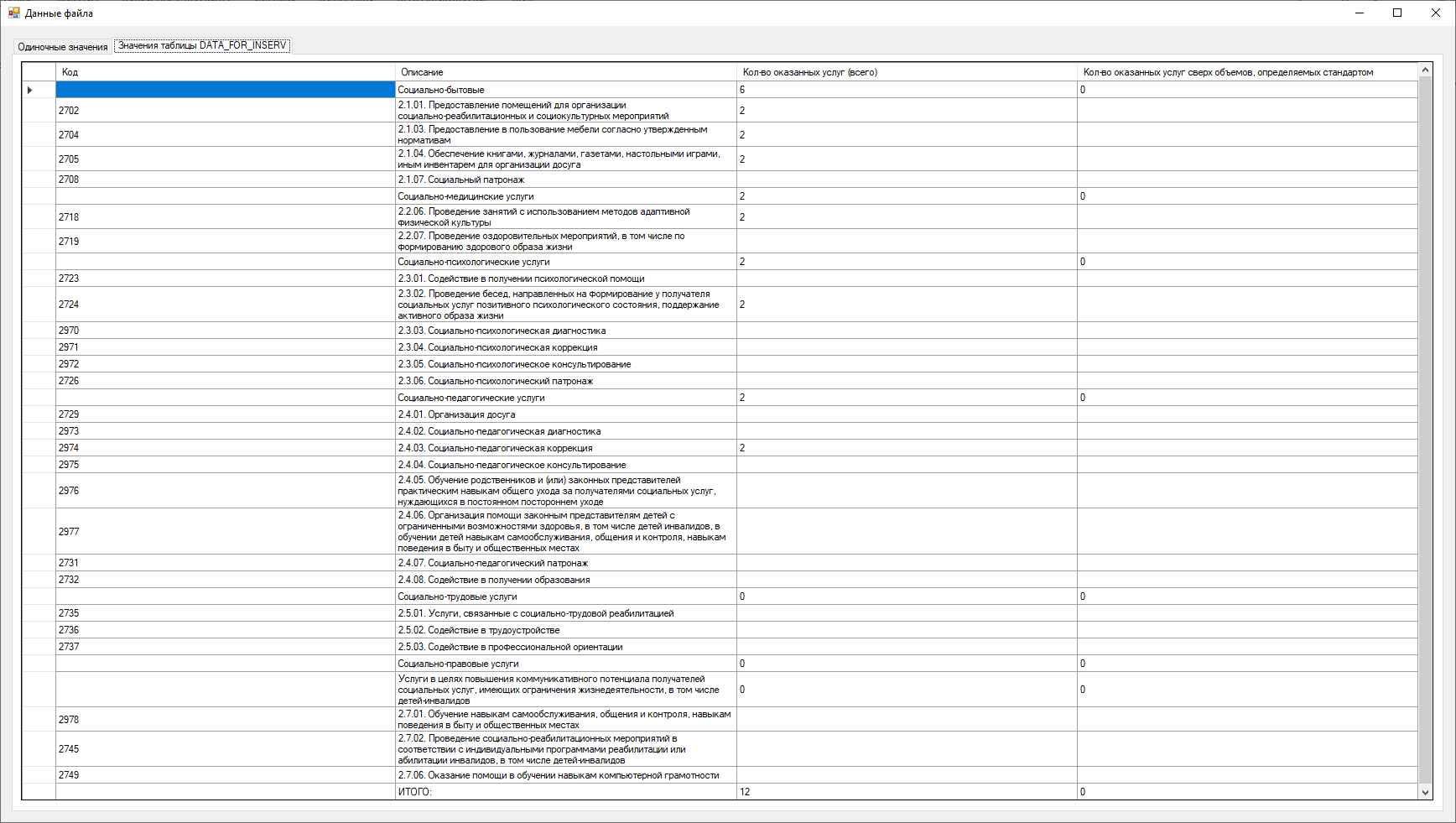


Рисунок 7 – Просмотр извлеченных данных (вкладка с табличными значениями)

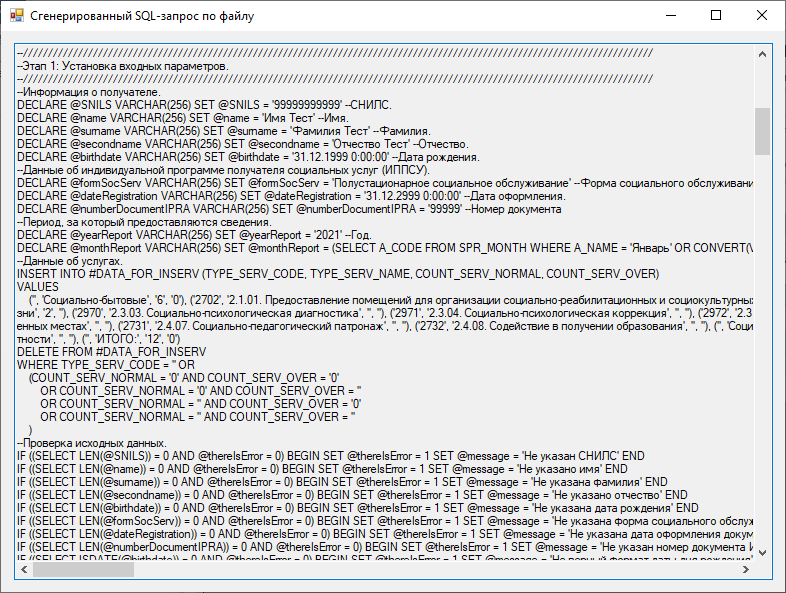


Рисунок 8 – Просмотр сгенерированного запроса

После прочтения всех файлов в папке или одиночного файла, откроется доступ к исполнению успешно сгенерированных SQL-запросов. В поле статуса будет информация об успешном выполнении запроса, и при наличии ошибки, в поле дополнительной информации будет информация об ошибке.

# ПРИЛОЖЕНИЕ А

# Шаблон Excel-файла

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 8 – Страница с информацией о получателе

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 8 – Страница с информацией об оказанных услугах