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 , 17 рис., 4 табл., 7 источников, 6 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](#_Toc86605259)

[1 Анализ существующих аналогов 6](#_Toc86605260)

[1.1 Экспорт данных с помощью средств в SQL Server 6](#_Toc86605261)

[1.2 Экспорт данных с помощью средств в Excel 7](#_Toc86605262)

[1.3 Сравнение аналогов 8](#_Toc86605263)

[2 Техническое задание 9](#_Toc86605264)

[2.1 Введение 9](#_Toc86605265)

[2.2 Основание для разработки 9](#_Toc86605266)

[2.3 Назначение 9](#_Toc86605267)

[2.4 Требования к программе 9](#_Toc86605268)

[3 Разработка парсера Excel-файла 11](#_Toc86605269)

[3.1 Дескриптор Excel-файла общая информация 11](#_Toc86605270)

[3.2 Парсинг дескриптора Excel-файла 17](#_Toc86605271)

[3.3 Парсинг Excel-файла 17](#_Toc86605272)

[4 Разработка исполнителя SQL-запросов 19](#_Toc86605273)

[5 Разработка пользовательского интерфейса 20](#_Toc86605274)

[ЗАКЛЮЧЕНИЕ 26](#_Toc86605275)

[СПИСОК ЛИТЕРАТУРЫ 27](#_Toc86605276)

[ПРИЛОЖЕНИЕ А.](#_Toc86605277) [Шаблон Excel-файла 28](#_Toc86605278)

[ПРИЛОЖЕНИЕ Б.](#_Toc86605279) [Дескриптор шаблона Excel-файла 30](#_Toc86605280)

[ПРИЛОЖЕНИЕ В.](#_Toc86605281) [SQL-запрос для вставки данных 32](#_Toc86605282)

[ПРИЛОЖЕНИЕ Г](#_Toc86605283). [Диаграмма классов .38](#_Toc86605284)

[ПРИЛОЖЕНИЕ Д.](#_Toc86605285) [Алгоритм извлечения одиночного значения из файла 39](#_Toc86605286)

[ПРИЛОЖЕНИЕ Е.](#_Toc86605287) [Алгоритм извлечения таблицы из файла 40](#_Toc86605288)

[ПРИЛОЖЕНИЕ Ж.](#_Toc86605289) [Пользовательский интерфейс 41](#_Toc86605290)

# 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

## Экспорт данных с помощью средств в SQL Server

В СУБД **Microsoft SQL Server** существует отличный функционал по импорту и экспорту данных, причем в разные форматы и разные базы данных. Его можно также использовать для простого переноса данных из одной базы в другую или с одного сервера на другой, рисунок 1.1

Данный способ отлично подходит, если данные представлены в виде таблицы, но данный способ не подходит в том случае, если данные располагаются в различных частях файла. Так же данный способ требует наличие установленной среды разработки Microsoft SQL Server Management Studio, поэтому данный способ подойдет системным администраторам, нежели обычным пользователям.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1.1 – Стартовое окно мастера импорта и экспорта SQL-Server

## Экспорт данных с помощью средств в Excel

Экспорт данных можно производить из самого Excel, для этого необходимо написать соответствующий макрос для обработки на языке Visual Basic. С помощью макроса можно производить поиск данных в файле, подключение к базе данных и выполнение написанных SQL-запросов.

Данный способ подойдет как к данным, представленным в виде таблицы, так и данным, располагающимся в разных частях файла. Но данный способ имеет ряд недостатков. Во-первых, необходимо производить действия для подключения макросов. Во-вторых, требуется установка дополнительных библиотек для Visual Basic, если они были использованы для написания макросов. В-третьих, отсутствует пользовательский интерфейс.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 1.2 – Окно для написания макросов в Visual Basic

## Сравнение аналогов

В таблице 1.1 представлено сравнение рассмотренных аналогов с разрабатываемой программой. Из таблицы видно, что разрабатываемая программа имеет ряд преимуществ, относительно рассмотренных аналогов.

Недостатками мастера импорта и экспорта SQL-сервер является то, что отсутствует пакетная обработка файлов, из-за чего требуется вручную проделывать ряд шагов (выбор файла, выбор базы данных и т.д.) перед экспортом для каждого файла. Так же что файлы должны иметь определенный шаблон: на странице должна располагаться одна таблица; заголовки столбцов должны располагаться в первой строке файла.

Недостатками макросов из Excel является отсутствие пользовательского интерфейса и необходимость установки дополнительных средств, чтобы макросы можно было запускать.

Таблица 1.1 – Сравнение аналогов

|  |  |  |  |
| --- | --- | --- | --- |
| Параметр | Мастер импорта и экспорта SQL-сервер | Исполнение макросов из Excel | Разрабатываемая программа |
| экспорт данных из Excel-файла в базу данных | + | + | + |
| возможность обработки Excel-файла любого шаблона | - | + | + |
| наличие пакетной обработки файлов | - | + | + |
| возможность подключения собственного SQL-запроса при экспорте | + | + | + |
| импорт сразу в несколько различных таблиц | - | + | + |
| запуск не требует установки дополнительных средств | - | - | + |
| удобный пользовательский интерфейс | + | - | + |

# 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 на рисунке 3.1 представлен дескриптор объекта, который имеет N атрибутов и один вложенный тег. А в приложении Б представлен дескриптор шаблона Excel-файла, представленного в приложении А.

<tag>

nameToken\_1 : valueToken\_1;

…

nameToken\_N : valueToken\_N;

<nestedObject>

nameToken\_1 : valueToken\_1;

…

nameToken\_N : valueToken\_N;

<nestedObject>

nameToken\_1 : valueToken\_1;

…

nameToken\_N : valueToken\_N;

</nestedObject>

</nestedObject>

</tag>

Рисунок 3.1 – Пример оформления дескриптора объекта

В таблице 3.1 представлены возможные имена атрибутов тега singleValue.

Таблица 3.1 – Возможные атрибуты тега singleValue

|  |  |  |  |
| --- | --- | --- | --- |
| Коды атрибутов | Допустимые значения | Описание | Если значение не указано |
| 1 | 2 | 3 | 4 |
| 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 3.2. | 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 3.2. | 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 3.2. | Поиск не производится в левой верхней части от раздела (Аналогично значению 0). |

Продолжение таблицы 3.1

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| SECTION\_UP\_RIGHT | 1 или 0 | При условии, что атрибут SECTION\_NAME определен. Область сверху справа от раздела включать в поиск, рисунок 3.2. | 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 3.2 представлены возможные имена атрибутов тега table.

Таблица 3.2– Возможные атрибуты тега table

|  |  |  |  |
| --- | --- | --- | --- |
| Коды атрибутов | Допустимые значения | Описание | Если значение не указано |
| 1 | 2 | 3 | 4 |
| 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 |

Продолжение таблицы 3.2

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| 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 3.2. | 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 3.2. | 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 3.2. | 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 3.2. | 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 3.3 представлены возможные имена атрибутов тега column.

Таблица 3.3– 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 3.2 изображены области для поиска относительно раздела при соответственно установленных флагов.



Рисунок 3.2 – 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 3.3.

## 3.3 Парсинг Excel-файла

Парсинг Excel-файла осуществляется по дескрипторам объектов, извлеченным из дескриптора Excel-файла.

Сначала определяется, на каких страницах файла будет произведен поиск. Если в дескрипторе объекта указан номер или название страницы, то поиск будет произведен только на странице с данным номером или данным названием, иначе поиск будет производиться на всех страницах файла, и результат будет выдан с той страницы, на которой впервые был найден объект для поиска.

Далее определяется область на странице для поиска. Если указано имя раздела, которое подразумевает собой текст ячейки, относительно которой будет производиться поиск, то относительно этой ячейки и на основе флагов, которые указывают, производить ли поиск в секциях справа снизу, слева снизу, справа сверху, слева сверху относительно ячейки, отвечающей за секцию. Сделано это для того, чтобы различать одиночные значения с одинаковыми полями, либо различать таблицы с одинаковыми столбцами.

После того, как область для поиска была определена, происходит поиск поля, относительно которого будет прочитано значение с указанным смещением в дескрипторе объекта, либо набор полей в одной строке, отвечающих за наименования столбцов таблицы. В случае, если производится поиск таблицы, то не важно, в какой последовательности будут располагаться столбцы, и какие еще столбцы будут располагаться между данными столбцами, главное, чтобы все указанные столбцы находились в строке, иначе таблица не будет считаться совпадающей, если хотя бы один столбец не будет найден. Так же для таблицы производится поиск последней строки на основе данных, указанных в дескрипторе.

После того, как данные были прочитаны, им присваивается код, указанный в дескрипторе данного объекта, при этом, если это таблица, то каждому столбцу так же присваивается код. И далее по данным кодам можно обращаться в SQL-запросе.

Алгоритмы извлечения одиночного значения и таблицы, указаны в приложении Д и Е.

# 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.



Рисунок 3.3 – Алгоритм получения дескриптора объекта

# Разработка пользовательского интерфейса

Главное окно разработанной программы содержит в себе две основные секции – меню и панель обработки файлов, рисунок 5.1. При загрузке файлов отображается прогресс загрузки файлов, а также статус. При ошибке чтения фала в поле дополнительной информации отображается причина ошибки, рисунок 5.2.

После загрузки файла можно посмотреть данные, которые были извлечены парсером. Одна вкладка с одиночными данными, рисунок 5.5, и вкладки с табличными значениями, рисунок 5.6. Так же после загрузки файла можно посмотреть сгенерированный запрос со вставленными значениями, рисунок 5.7.

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.

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.

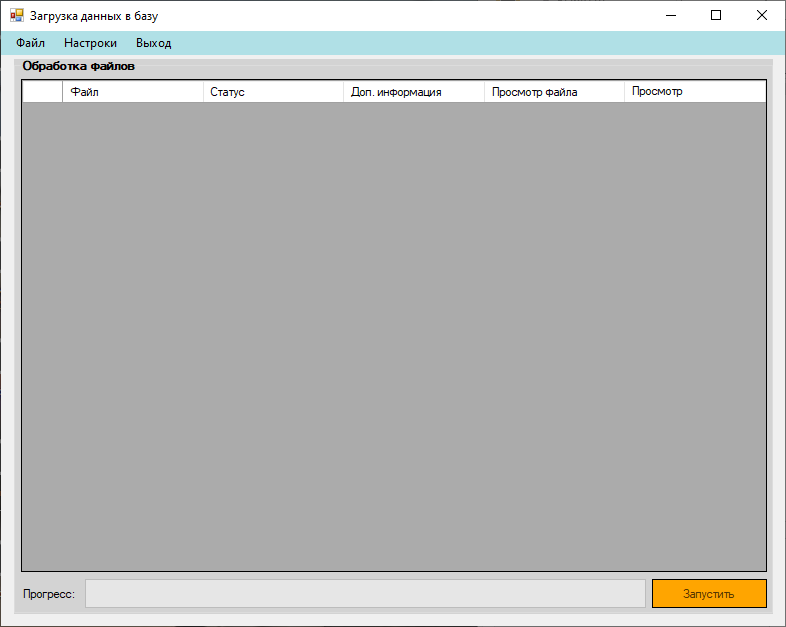


Рисунок 5.1 – Главное окно программы

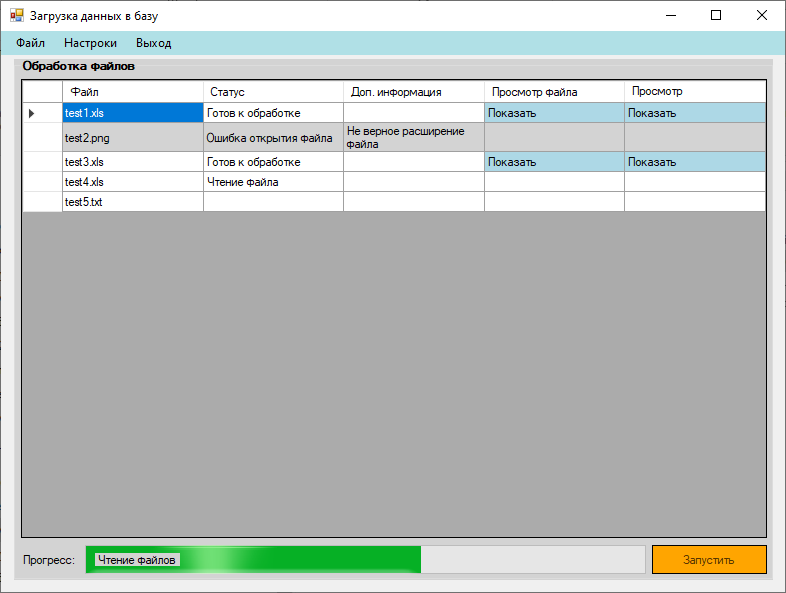


Рисунок 5.2 – Загрузка файлов

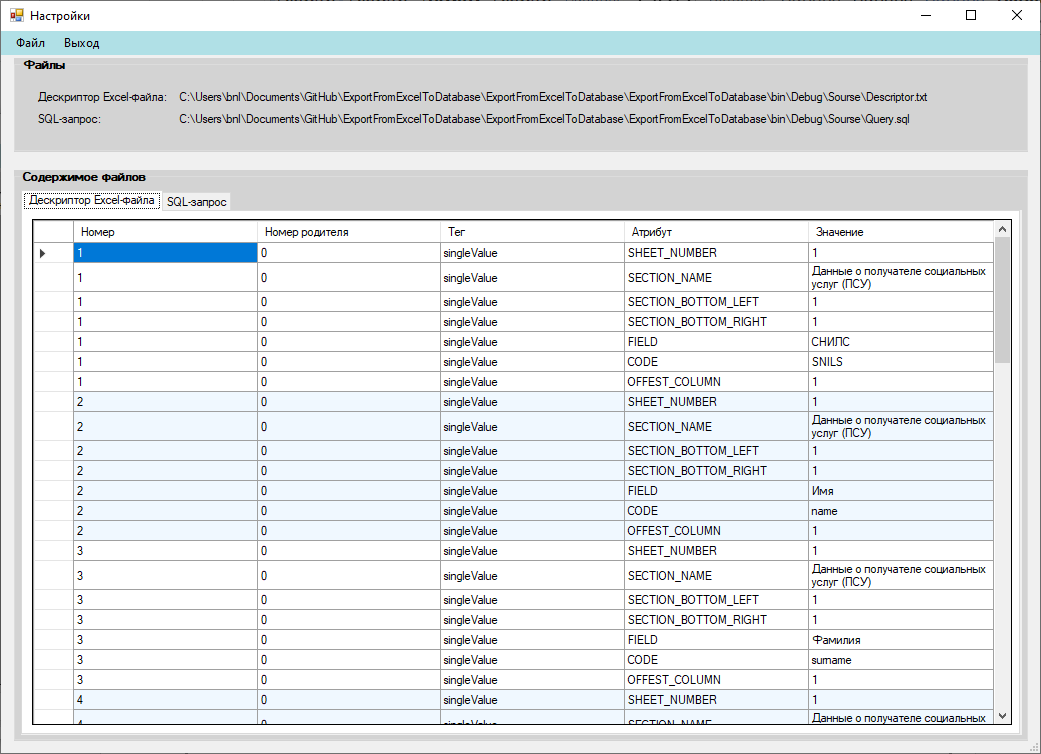


Рисунок 5.3 – Настройки (вкладка с отображением дескриптора)

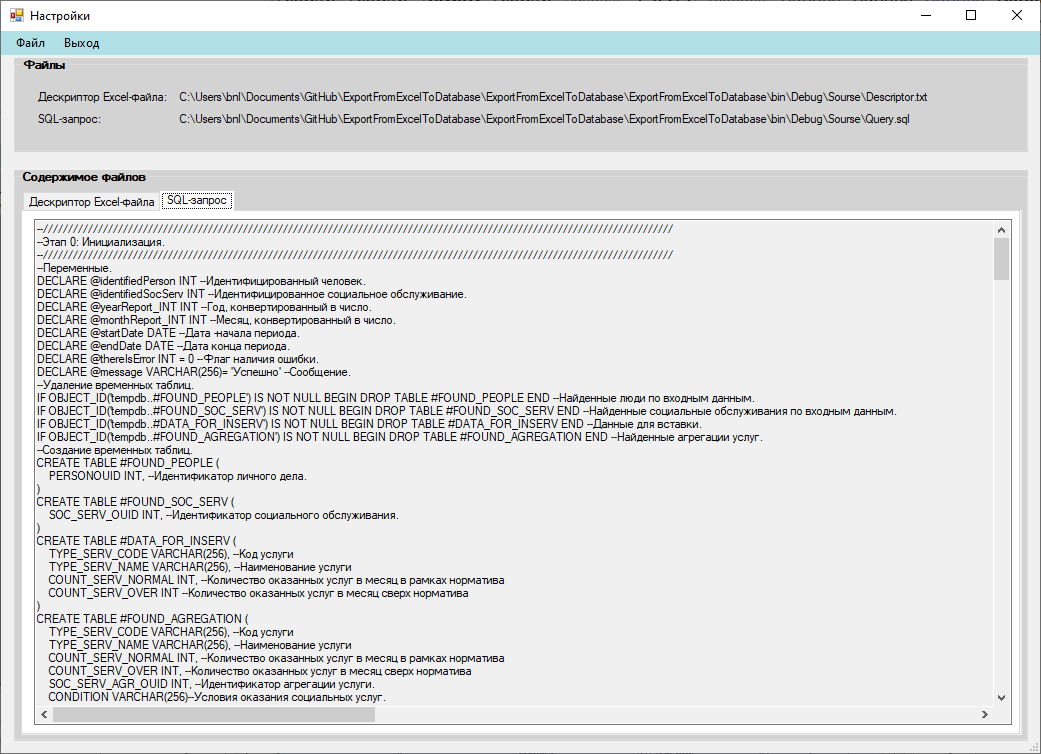


Рисунок 5.4 – Настройки (вкладка с отображением запроса)

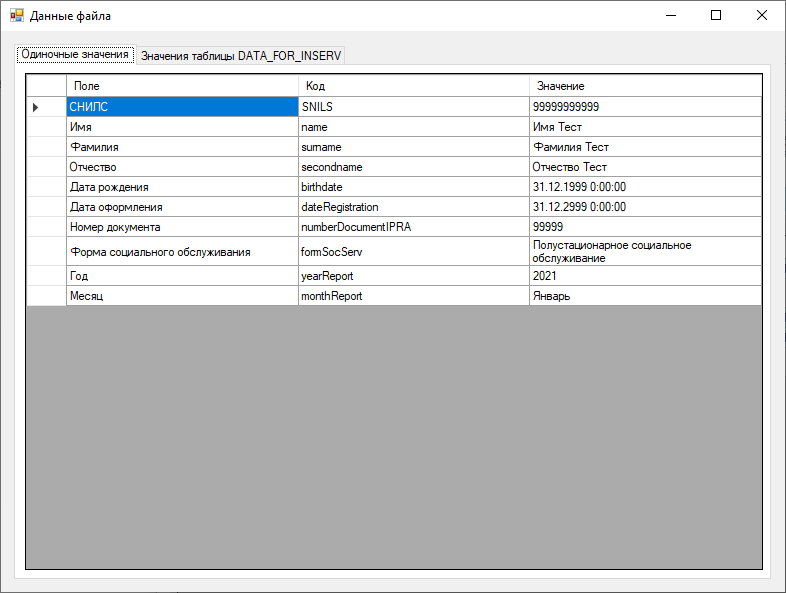


Рисунок 5.5 – Просмотр извлеченных данных (вкладка с одиночными значениями)

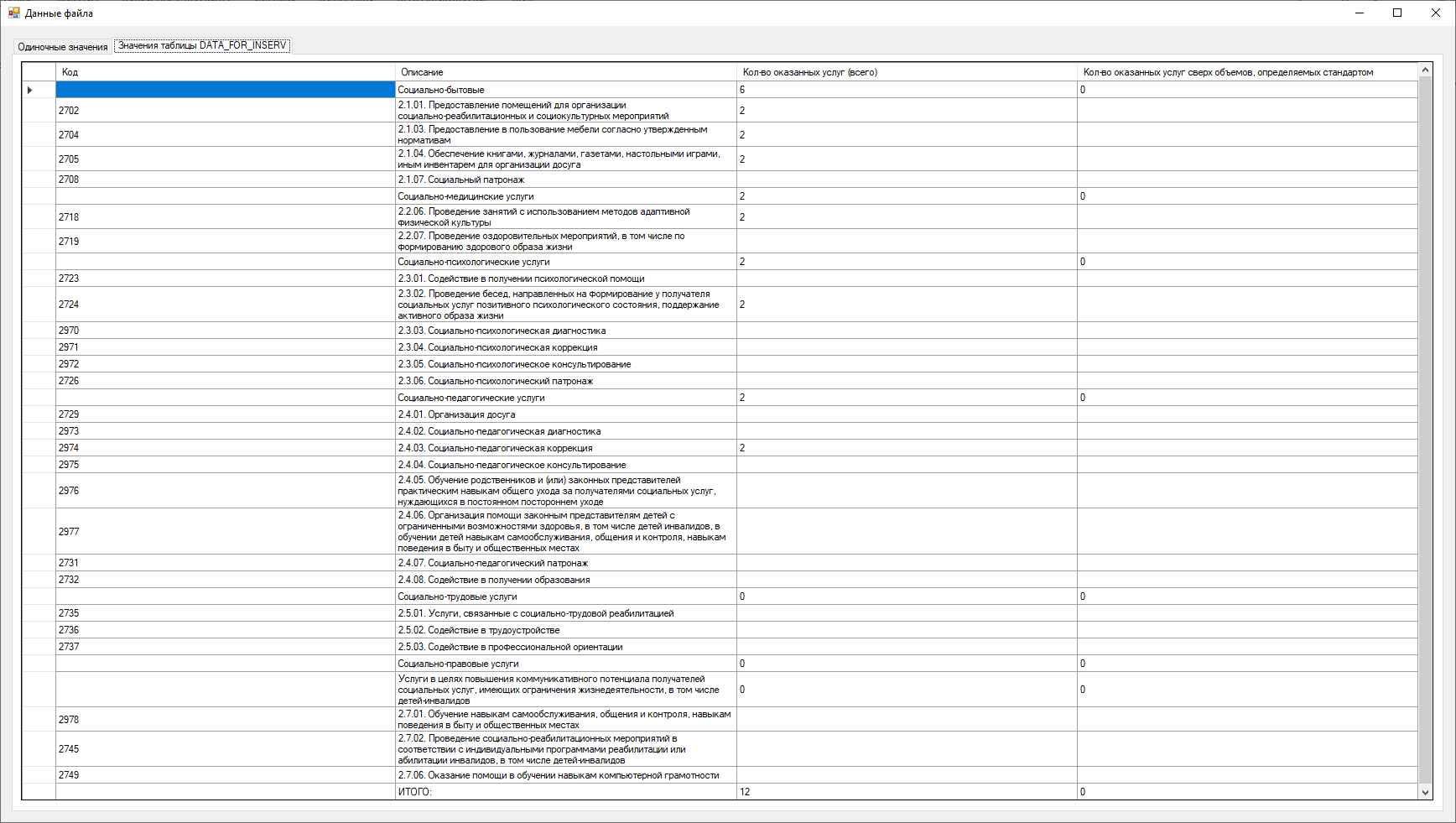


Рисунок 5.6 – Просмотр извлеченных данных (вкладка с табличными значениями)

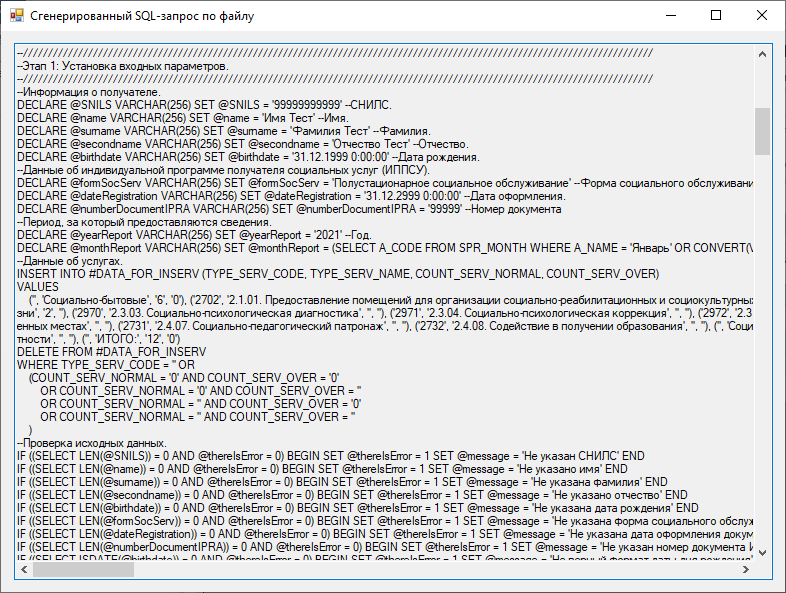


Рисунок 5.7 – Просмотр сгенерированного запроса

# ЗАКЛЮЧЕНИЕ

В рамках данной курсовой работы была разработана программа для экспорта данных из Excel-файла в базу данных.

Обстоятельством для разработки послужило то, что у имеющихся аналогов отсутствует нужный функционал для решения поставленной задачи. Например, у мастера импорта и экспорта SQL-сервер отсутствует пакетная обработка файлов, а также ограничение к шаблону файла: на странице должна располагаться одна таблица; заголовки столбцов должны располагаться в первой строке файла; экспорт может осуществляться только в отдельную таблицу. Решение проблемы с помощью макросов из Excel имеет так же ряд недостатков: отсутствие пользовательского интерфейса и необходимость установки дополнительных средств, чтобы макросы можно было запускать.

В ходе выполнения курсового проекта был разработан класс-парсер дескриптора Excel-файла, который собирает информацию об объектах, располагаемых в Excel-файле. По данной информации, которая помогает найти область считывания данных, класс-парсер Excel-файла извлекает данные из файла. Далее извлеченные данные вставляются классом-генератором запросов в специальные места SQL-запроса, которые отмечены специальными символами и кодами. Сгенерированный SQL-запрос отправляется на исполнение классом-исполнителем SQL-запросов.

Результатом работы является программа, которая имеет пакетную обработку файлов, шаблон которых можно задать с помощью дескриптора Excel-файла, и SQL-запрос для которых можно задать.

# СПИСОК ЛИТЕРАТУРЫ

1. Herbert S. C#: The Complete Reference [Текст] / Herbert S. – McGraw-Hill Osborne Media, 2010. – 949 с.
2. Weisfeld M. The Object-Oriented Thought Process [Текст] / Weisfeld M. – Addison-Wesley, 2009. – 304 с.
3. Автоматизация рутины в Microsoft Excel при помощи VBA [Электронный ресурс]. URL: https://habr.com/ru/post/112080/
4. Средство импорта и экспорта данных в Microsoft SQL Server [Электронный ресурс]. URL: https://info-comp.ru/sisadminst/357-import-export-data-mssql.html
5. Паттерны/шаблоны проектирования [Электронный ресурс]. URL: https://refactoring.guru/ru/design-patterns
6. 90 рекомендаций по стилю написания программ С++ [Электронный ресурс]. URL: https://habr.com/ru/post/172091/
7. 10 приемов, разрушающих хрупкую красоту кода [Электронный ресурс]. URL: https://habr.com/ru/post/59570/

# ПРИЛОЖЕНИЕ А

# Шаблон Excel-файла

Изображение выглядит как стол

Автоматически созданное описание

Страница с информацией о получателе

Изображение выглядит как стол

Автоматически созданное описание

Страница с информацией об оказанных услугах

# ПРИЛОЖЕНИЕ Б

# Дескриптор шаблона Excel-файла

<singleValue>

SHEET\_NUMBER: 1;

SECTION\_NAME: "Данные о получателе социальных услуг (ПСУ)";

SECTION\_BOTTOM\_LEFT: 1;

SECTION\_BOTTOM\_RIGHT: 1;

FIELD: СНИЛС;

CODE: SNILS;

OFFEST\_COLUMN: 1;

</singleValue>

<singleValue>

SHEET\_NUMBER: 1;

SECTION\_NAME: "Данные о получателе социальных услуг (ПСУ)";

SECTION\_BOTTOM\_LEFT: 1;

SECTION\_BOTTOM\_RIGHT: 1;

FIELD: Имя;

CODE: name;

OFFEST\_COLUMN: 1;

</singleValue>

<singleValue>

SHEET\_NUMBER: 1;

SECTION\_NAME: "Данные о получателе социальных услуг (ПСУ)";

SECTION\_BOTTOM\_LEFT: 1;

SECTION\_BOTTOM\_RIGHT: 1;

FIELD: Фамилия;

CODE: surname;

OFFEST\_COLUMN: 1;

</singleValue>

<singleValue>

SHEET\_NUMBER: 1;

SECTION\_NAME: "Данные о получателе социальных услуг (ПСУ)";

SECTION\_BOTTOM\_LEFT: 1;

SECTION\_BOTTOM\_RIGHT: 1;

FIELD: Отчество;

CODE: secondname;

OFFEST\_COLUMN: 1;

</singleValue>

<singleValue>

SHEET\_NUMBER: 1;

SECTION\_NAME: "Данные о получателе социальных услуг (ПСУ)";

SECTION\_BOTTOM\_LEFT: 1;

SECTION\_BOTTOM\_RIGHT: 1;

FIELD: "Дата рождения";

CODE: birthdate;

OFFEST\_COLUMN: 1;

</singleValue>

<singleValue>

SHEET\_NUMBER: 1;

SECTION\_NAME: "Данные об индивидуальной программе получателя социальных услуг (ИППСУ)";

SECTION\_BOTTOM\_LEFT: 1;

SECTION\_BOTTOM\_RIGHT: 1;

FIELD: "Дата оформления";

CODE: dateRegistration;

OFFEST\_COLUMN: 1;

</singleValue>

<singleValue>

SHEET\_NUMBER: 1;

SECTION\_NAME: "Данные об индивидуальной программе получателя социальных услуг (ИППСУ)";

SECTION\_BOTTOM\_LEFT: 1;

SECTION\_BOTTOM\_RIGHT: 1;

FIELD: "Номер документа ";

CODE: numberDocumentIPRA;

OFFEST\_COLUMN: 1;

</singleValue>

<singleValue>

SHEET\_NUMBER: 1;

SECTION\_NAME: "Данные о договоре (доп.соглашении)";

SECTION\_BOTTOM\_LEFT: 1;

SECTION\_BOTTOM\_RIGHT: 1;

FIELD: "Форма социального обслуживания";

CODE: formSocServ;

OFFEST\_COLUMN: 1;

</singleValue>

<singleValue>

SHEET\_NUMBER: 1;

SECTION\_NAME: "Период за который предоставляются сведения";

SECTION\_BOTTOM\_LEFT: 1;

SECTION\_BOTTOM\_RIGHT: 1;

FIELD: "Год";

CODE: yearReport;

OFFEST\_COLUMN: 1;

</singleValue>

<singleValue>

SHEET\_NUMBER: 1;

SECTION\_NAME: "Период за который предоставляются сведения";

SECTION\_BOTTOM\_LEFT: 1;

SECTION\_BOTTOM\_RIGHT: 1;

FIELD: "Месяц";

CODE: monthReport;

OFFEST\_COLUMN: 1;

</singleValue>

<table>

SHEET\_NUMBER: 2;

CODE: "DATA\_FOR\_INSERV";

INCLUDE\_FINAL\_ROW: "1";

<column>

NAME: "Код";

CODE: "TYPE\_SERV\_CODE";

</column>

<column>

NAME: "Описание";

CODE: "TYPE\_SERV\_NAME";

FINAL\_CELL: "ИТОГО:";

</column>

<column>

NAME: "Кол-во оказанных услуг (всего)";

CODE: "COUNT\_SERV\_NORMAL";

</column>

<column>

NAME: "Кол-во оказанных услуг сверх объемов, определяемых стандартом";

CODE: "COUNT\_SERV\_OVER";

</column>

</table>

# ПРИЛОЖЕНИЕ В

# SQL-запрос для вставки данных

--//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

--Этап 0: Инициализация.

--//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

--Переменные.

DECLARE @identifiedPerson INT --Идентифицированный человек.

DECLARE @identifiedSocServ INT --Идентифицированное социальное обслуживание.

DECLARE @yearReport\_INT INT --Год, конвертированный в число.

DECLARE @monthReport\_INT INT --Месяц, конвертированный в число.

DECLARE @startDate DATE --Дата -начала периода.

DECLARE @endDate DATE --Дата конца периода.

DECLARE @thereIsError INT = 0 --Флаг наличия ошибки.

DECLARE @message VARCHAR(256)= 'Успешно' --Сообщение.

--Удаление временных таблиц.

IF OBJECT\_ID('tempdb..#FOUND\_PEOPLE') IS NOT NULL BEGIN DROP TABLE #FOUND\_PEOPLE END --Найденные люди по входным данным.

IF OBJECT\_ID('tempdb..#FOUND\_SOC\_SERV') IS NOT NULL BEGIN DROP TABLE #FOUND\_SOC\_SERV END --Найденные социальные обслуживания по входным данным.

IF OBJECT\_ID('tempdb..#DATA\_FOR\_INSERV') IS NOT NULL BEGIN DROP TABLE #DATA\_FOR\_INSERV END --Данные для вставки.

IF OBJECT\_ID('tempdb..#FOUND\_AGREGATION') IS NOT NULL BEGIN DROP TABLE #FOUND\_AGREGATION END --Найденные агрегации услуг.

--Создание временных таблиц.

CREATE TABLE #FOUND\_PEOPLE (

PERSONOUID INT, --Идентификатор личного дела.

)

CREATE TABLE #FOUND\_SOC\_SERV (

SOC\_SERV\_OUID INT, --Идентификатор социального обслуживания.

)

CREATE TABLE #DATA\_FOR\_INSERV (

TYPE\_SERV\_CODE VARCHAR(256), --Код услуги

TYPE\_SERV\_NAME VARCHAR(256), --Наименование услуги

COUNT\_SERV\_NORMAL INT, --Количество оказанных услуг в месяц в рамках норматива

COUNT\_SERV\_OVER INT --Количество оказанных услуг в месяц сверх норматива

)

CREATE TABLE #FOUND\_AGREGATION (

TYPE\_SERV\_CODE VARCHAR(256), --Код услуги

TYPE\_SERV\_NAME VARCHAR(256), --Наименование услуги

COUNT\_SERV\_NORMAL INT, --Количество оказанных услуг в месяц в рамках норматива

COUNT\_SERV\_OVER INT, --Количество оказанных услуг в месяц сверх норматива

SOC\_SERV\_AGR\_OUID INT, --Идентификатор агрегации услуги.

CONDITION VARCHAR(256)--Условия оказания социальных услуг.

)

--//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

--Этап 1: Установка входных параметров.

--//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

--Информация о получателе.

DECLARE @SNILS VARCHAR(256) SET @SNILS = '#SNILS#' --СНИЛС.

DECLARE @name VARCHAR(256) SET @name = '#name#' --Имя.

DECLARE @surname VARCHAR(256) SET @surname = '#surname#' --Фамилия.

DECLARE @secondname VARCHAR(256) SET @secondname = '#secondname#' --Отчество.

DECLARE @birthdate VARCHAR(256) SET @birthdate = '#birthdate#' --Дата рождения.

--Данные об индивидуальной программе получателя социальных услуг (ИППСУ).

DECLARE @formSocServ VARCHAR(256) SET @formSocServ = '#formSocServ#' --Форма социального обслуживания.

DECLARE @dateRegistration VARCHAR(256) SET @dateRegistration = '#dateRegistration#' --Дата оформления.

DECLARE @numberDocumentIPRA VARCHAR(256) SET @numberDocumentIPRA = '#numberDocumentIPRA#' --Номер документа

--Период, за который предоставляются сведения.

DECLARE @yearReport VARCHAR(256) SET @yearReport = '#yearReport#' --Год.

DECLARE @monthReport VARCHAR(256) SET @monthReport = (SELECT A\_CODE FROM SPR\_MONTH WHERE A\_NAME = '#monthReport#' OR CONVERT(VARCHAR, A\_CODE) = '#monthReport#') --Месяц.

--Данные об услугах.

INSERT INTO #DATA\_FOR\_INSERV (TYPE\_SERV\_CODE, TYPE\_SERV\_NAME, COUNT\_SERV\_NORMAL, COUNT\_SERV\_OVER)

VALUES

<DATA\_FOR\_INSERV ('#TYPE\_SERV\_CODE#', '#TYPE\_SERV\_NAME#', '#COUNT\_SERV\_NORMAL#', '#COUNT\_SERV\_OVER#')>

DELETE FROM #DATA\_FOR\_INSERV

WHERE TYPE\_SERV\_CODE = '' OR

(COUNT\_SERV\_NORMAL = '0' AND COUNT\_SERV\_OVER = '0'

OR COUNT\_SERV\_NORMAL = '0' AND COUNT\_SERV\_OVER = ''

OR COUNT\_SERV\_NORMAL = '' AND COUNT\_SERV\_OVER = '0'

OR COUNT\_SERV\_NORMAL = '' AND COUNT\_SERV\_OVER = ''

OR COUNT\_SERV\_NORMAL = '0' AND COUNT\_SERV\_OVER = '0'

)

--Проверка исходных данных.

IF ((SELECT LEN(@SNILS)) = 0 AND @thereIsError = 0) BEGIN SET @thereIsError = 1 SET @message = 'Не указан СНИЛС' END

IF ((SELECT LEN(@name)) = 0 AND @thereIsError = 0) BEGIN SET @thereIsError = 1 SET @message = 'Не указано имя' END

IF ((SELECT LEN(@surname)) = 0 AND @thereIsError = 0) BEGIN SET @thereIsError = 1 SET @message = 'Не указана фамилия' END

IF ((SELECT LEN(@secondname)) = 0 AND @thereIsError = 0) BEGIN SET @thereIsError = 1 SET @message = 'Не указано отчество' END

IF ((SELECT LEN(@birthdate)) = 0 AND @thereIsError = 0) BEGIN SET @thereIsError = 1 SET @message = 'Не указана дата рождения' END

IF ((SELECT LEN(@formSocServ)) = 0 AND @thereIsError = 0) BEGIN SET @thereIsError = 1 SET @message = 'Не указана форма социального обслуживания' END

IF ((SELECT LEN(@dateRegistration)) = 0 AND @thereIsError = 0) BEGIN SET @thereIsError = 1 SET @message = 'Не указана дата оформления документа ИППСУ' END

IF ((SELECT LEN(@numberDocumentIPRA)) = 0 AND @thereIsError = 0) BEGIN SET @thereIsError = 1 SET @message = 'Не указан номер документа ИППСУ' END

IF ((SELECT ISDATE(@birthdate)) = 0 AND @thereIsError = 0) BEGIN SET @thereIsError = 1 SET @message = 'Не верный формат даты дня рождения' END

IF ((SELECT ISDATE(@dateRegistration)) = 0 AND @thereIsError = 0) BEGIN SET @thereIsError = 1 SET @message = 'Не верный формат даты оформления ИППСУ' END

IF ((SELECT ISNUMERIC(@yearReport)) = 0 AND @thereIsError = 0) BEGIN SET @thereIsError = 1 SET @message = 'Не верный формат года' END

IF ((SELECT ISNUMERIC(@monthReport)) = 0 AND @thereIsError = 0) BEGIN SET @thereIsError = 1 SET @message = 'Не верный формат месяца' END

IF ((SELECT COUNT(\*) FROM #DATA\_FOR\_INSERV) = 0) BEGIN SET @thereIsError = 1 SET @message = 'Нет не нулевых услуг' END

--Конвертация исходных данных.

IF (@thereIsError = 0) BEGIN

SET @yearReport\_INT = CONVERT(INT, @yearReport)

SET @monthReport\_INT = CONVERT(INT, @monthReport)

SET @startDate = CAST(@yearReport + '-' + @monthReport + '-01' AS DATE)

SET @endDate = DATEADD(MONTH, ((YEAR(@startDate) - 1900) \* 12) + MONTH(@startDate), -1)

END

--//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

--Этап 2: Идентификация человека.

--//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

IF (@thereIsError = 0) BEGIN

--Выбор людей, удовлетворяющих условиям.

INSERT #FOUND\_PEOPLE (PERSONOUID)

SELECT DISTINCT

personalCard.OUID AS PERSONOUID

FROM WM\_PERSONAL\_CARD personalCard --Личное дело гражданина.

----Фамилия.

LEFT JOIN SPR\_FIO\_SURNAME fioSurname

ON fioSurname.OUID = personalCard.SURNAME

----Имя.

LEFT JOIN SPR\_FIO\_NAME fioName

ON fioName.OUID = personalCard.A\_NAME

----Отчество.

LEFT JOIN SPR\_FIO\_SECONDNAME fioSecondname

ON fioSecondname.OUID = personalCard.A\_SECONDNAME

WHERE personalCard.A\_STATUS = 10 --Статус личного дела в БД "Действует".

AND personalCard.A\_PCSTATUS = 1 --Статус личного дела "Действует".

AND personalCard.A\_SNILS = @SNILS --СНИЛС совпадает.

AND ISNULL(personalCard.A\_NAME\_STR, fioName.A\_NAME) = @name --Имя совпадает.

AND ISNULL(personalCard.A\_SURNAME\_STR, fioSurname.A\_NAME) = @surname --Фамилия совпадает.

AND CONVERT(DATE, personalCard.BIRTHDATE) = CONVERT(DATE, @birthdate) --Дата рожденяи совпадает.

AND ISNULL(personalCard.A\_SECONDNAME\_STR, fioSecondname.A\_NAME) = @secondname --Отчество совпадает.

--Подсчет людей.

DECLARE @countFoundPeople INT

SET @countFoundPeople = (SELECT COUNT(\*) FROM #FOUND\_PEOPLE)

--Результат 2 этапа.

IF (@countFoundPeople > 1) BEGIN

SET @thereIsError = 1

SET @message = 'Найдено более одного человека, удовлетворяющего условиям'

END

ELSE IF (@countFoundPeople = 0) BEGIN

SET @thereIsError = 1

SET @message = 'Не найден человек, удовлетворяющий условиям'

END

ELSE BEGIN

SET @identifiedPerson = (SELECT TOP 1 PERSONOUID FROM #FOUND\_PEOPLE)

END

END

--//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

--Этап 3: Идентификация социального обслуживания.

--//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

IF (@thereIsError = 0) BEGIN

--Выбор назначений на социальной обслуживание.

INSERT #FOUND\_SOC\_SERV(SOC\_SERV\_OUID)

SELECT DISTINCT

socServ.OUID AS SOC\_SERV\_OUID

FROM ESRN\_SOC\_SERV socServ --Назначение социального обслуживания.

----Период предоставления МСП.

INNER JOIN SPR\_SOCSERV\_PERIOD period

ON period.A\_STATUS = 10 --Статус в БД "Действует".

AND period.A\_SERV = socServ.OUID --Связка с назначением.

AND @yearReport\_INT BETWEEN YEAR(period.STARTDATE) AND ISNULL(YEAR(period.A\_LASTDATE), 3000) --Год отчета входит в период действия назначения.

AND (@yearReport\_INT <> YEAR(period.STARTDATE) AND @yearReport\_INT <> ISNULL(YEAR(period.A\_LASTDATE), 3000) --Год отчета не равен крайнему.

OR @yearReport\_INT = YEAR(period.STARTDATE) AND @monthReport\_INT >= MONTH(period.STARTDATE) --Или равен начальному, но месяц позже начала.

OR @yearReport\_INT = YEAR(period.A\_LASTDATE) AND @monthReport\_INT <= ISNULL(MONTH(period.A\_LASTDATE), 12) --Или равен конечному, но месяц раньше конца.

)

----Индивидуальная программа.

INNER JOIN INDIVID\_PROGRAM individProgram

ON individProgram.A\_OUID = socServ.A\_IPPSU

AND individProgram.A\_STATUS = 10 --Статус индивидуальной программы в БД "Действует".

----Действующие документы.

INNER JOIN WM\_ACTDOCUMENTS actDocuments

ON actDocuments.OUID = individProgram.A\_DOC

AND actDocuments.A\_STATUS = 10 --Статус документа в БД "Действует".

AND actDocuments.DOCUMENTSNUMBER = @numberDocumentIPRA --Номер документа совпадает с требуемым.

----Форма социального обслуживания.

INNER JOIN SPR\_FORM\_SOCSERV formSocServ

ON formSocServ.A\_OUID = individProgram.A\_FORM\_SOCSERV

AND formSocServ.A\_NAME = @formSocServ --Совпадает с формой отчета.

----Органы социальной защиты.

INNER JOIN SPR\_ORG\_BASE organization

ON organization.OUID = socServ.A\_ORGNAME

WHERE socServ.A\_STATUS = 10 --Статус назначения в БД "Действует".

AND socServ.A\_PERSONOUID = @identifiedPerson --Льготодержатель - идентифицированный человек.

--Подсчет назначений.

DECLARE @countFoundSocServ INT

SET @countFoundSocServ = (SELECT COUNT(\*) FROM #FOUND\_SOC\_SERV)

--Результат 3 этапа.

IF (@countFoundSocServ > 1) BEGIN

SET @thereIsError = 1

SET @message = 'Найдено более одного социального обслуживания, удовлетворяющего условиям'

END

ELSE IF (@countFoundSocServ = 0) BEGIN

SET @thereIsError = 1

SET @message = 'Не найдено социальное обслуживание, удовлетворяющего условиям'

END

ELSE BEGIN

SET @identifiedSocServ = (SELECT TOP 1 SOC\_SERV\_OUID FROM #FOUND\_SOC\_SERV)

END

END

--//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

--Этап 4: Идентификация агрегаций по услугам.

--//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

IF (@thereIsError = 0) BEGIN

--Выбор агрегации по услуге.

INSERT INTO #FOUND\_AGREGATION(TYPE\_SERV\_CODE, TYPE\_SERV\_NAME, COUNT\_SERV\_NORMAL, COUNT\_SERV\_OVER, SOC\_SERV\_AGR\_OUID, CONDITION)

SELECT DISTINCT

forInsert.TYPE\_SERV\_CODE AS TYPE\_SERV\_CODE,

forInsert.TYPE\_SERV\_NAME AS TYPE\_SERV\_NAME,

forInsert.COUNT\_SERV\_NORMAL AS COUNT\_SERV\_NORMAL,

forInsert.COUNT\_SERV\_OVER AS COUNT\_SERV\_OVER,

socServAGR.A\_ID AS SOC\_SERV\_AGR\_OUID,

condition.A\_COND\_SOC\_SERV

FROM ESRN\_SOC\_SERV socServ --Назначение социального обслуживания.

----Агрегация по социальной услуге.

INNER JOIN WM\_SOC\_SERV\_AGR socServAGR

ON socServAGR.ESRN\_SOC\_SERV = socServ.OUID

AND socServAGR.A\_STATUS = 10 --Статус агрегации в БД "Действует".

----Тарифы на социальные услуги.

INNER JOIN SPR\_TARIF\_SOC\_SERV socServTarif

ON socServTarif.A\_ID = socServAGR.A\_SOC\_SERV

AND socServTarif.A\_STATUS = 10 --Статус тарифа в БД "Действует".

----Социальные услуги.

INNER JOIN SPR\_SOC\_SERV typeSocServ

ON typeSocServ.OUID = socServTarif.A\_SOC\_SERV

AND typeSocServ.A\_STATUS = 10 --Статус социальной услуги в БД "Действует".

----Данные для вставки.

INNER JOIN #DATA\_FOR\_INSERV forInsert

ON CHARINDEX(forInsert.TYPE\_SERV\_CODE, typeSocServ.A\_CODE) > 0

--AND forInsert.TYPE\_SERV\_NAME = typeSocServ.A\_NAME

INNER JOIN WM\_COND\_SOC\_SERV\_ONE condition

ON condition.A\_SOC\_SERV\_AGR = socServAGR.A\_ID --Агрегация отчета.

AND condition.A\_STATUS = 10

AND (@yearReport\_INT BETWEEN YEAR(A\_STARTDATE) AND ISNULL(YEAR(A\_LASTDATE), 9999) --Год отчета входит в период действия назначения.

AND (@yearReport\_INT <> YEAR(A\_STARTDATE) AND @yearReport\_INT <> ISNULL(YEAR(A\_LASTDATE), 9999) --Год отчета не равен крайнему.

OR @yearReport\_INT = YEAR(A\_STARTDATE) AND @monthReport\_INT >= MONTH(A\_STARTDATE) --Или равен начальному, но месяц позже начала.

OR @yearReport\_INT = YEAR(A\_LASTDATE) AND @monthReport\_INT <= MONTH(A\_LASTDATE) --Или равен конечному, но месяц раньше конца.

)

)

WHERE socServ.A\_STATUS = 10 --Статус назначения в БД "Действует".

AND socServ.OUID = @identifiedSocServ --Идентифицированное назначение.

--Подсчет услуг.

DECLARE @countTypeServForInsert INT

SET @countTypeServForInsert = (SELECT COUNT(\*) FROM #DATA\_FOR\_INSERV)

DECLARE @countFoundTypeServ INT

SET @countFoundTypeServ = (SELECT COUNT(\*) FROM #FOUND\_AGREGATION)

--Результат 4 этапа.

IF (@countTypeServForInsert <> @countFoundTypeServ) BEGIN

SET @thereIsError = 1

SET @message = 'Не все услуги были найдены в социальном обслуживании'

END

END

--//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

--Этап 4: Проверка введенных данных.

--//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

IF (@thereIsError = 0) BEGIN

--Проверка отсутствия уже введенных данных.

DECLARE @alreadyThereCount INT

SET @alreadyThereCount = (

SELECT

COUNT(DISTINCT cosSocServ.A\_ID)

FROM #FOUND\_AGREGATION foundAgregation

INNER JOIN WM\_COST\_SOC\_SERV cosSocServ

ON cosSocServ.A\_AGR\_SOC\_SERV = foundAgregation.SOC\_SERV\_AGR\_OUID

AND cosSocServ.A\_STATUS = 10

AND cosSocServ.A\_DATE\_START = @startDate

AND cosSocServ.A\_DATE\_LAST = @endDate

)

IF (@alreadyThereCount = @countTypeServForInsert) BEGIN

SET @thereIsError = 1

SET @message = 'Данные уже есть за данный период'

END

IF (@alreadyThereCount <> @countTypeServForInsert AND @alreadyThereCount <> 0) BEGIN

SET @thereIsError = 1

SET @message = 'Частично данные уже есть за данный период'

END

END

--//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

--Этап 5: Вставка стоимости и количество оказанных социальных услуг.

--//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

IF (@thereIsError = 0) BEGIN

INSERT INTO WM\_COST\_SOC\_SERV(A\_EMPLOYEE, A\_EDITOWNER, A\_STATUS, GUID, TS, SYSTEMCLASS, A\_CREATEDATE, A\_CROWNER, A\_AGR\_SOC\_SERV, A\_SUM\_SOC\_SERV\_PERIOD, A\_DATE\_START, A\_DATE\_LAST, A\_ACT\_VOLUME, A\_COMMENT, A\_ACT\_EXCESS\_QUANT, A\_COST\_DOP\_SOC\_SERV, A\_ACT\_QUANT\_NORM)

SELECT

CAST(NULL AS INT) AS A\_EMPLOYEE, --Сотрудники.

CAST(NULL AS INT) AS A\_EDITOWNER, --Изменил.

10 AS A\_STATUS, --Статус.

NEWID() AS GUID, --Глобальный идентификатор.

GETDATE() AS TS, --Дата модификации.

10284209 AS SYSTEMCLASS, --Класс объекта.

GETDATE() AS A\_CREATEDATE, --Дата создания.

10314303 AS A\_CROWNER, --Автор.

SOC\_SERV\_AGR\_OUID AS A\_AGR\_SOC\_SERV, --Социальная услуга (Агрегация по социальной услуге).

CAST(NULL AS FLOAT) AS A\_SUM\_SOC\_SERV\_PERIOD, --Сумма по услуге за период, руб.

@startDate AS A\_DATE\_START, --Дата начала периода.

@endDate AS A\_DATE\_LAST, --Дата окончания периода.

CAST(NULL AS FLOAT) AS A\_ACT\_VOLUME, --Фактический объем работ на 1 услугу.

CAST(NULL AS VARCHAR) AS A\_COMMENT, --Примечание.

COUNT\_SERV\_OVER AS A\_ACT\_EXCESS\_QUANT, --Количество оказанных услуг в месяц сверх норматива.

CAST(NULL AS FLOAT) AS A\_COST\_DOP\_SOC\_SERV, --Стоимость оказанных услуг, руб.

COUNT\_SERV\_NORMAL AS A\_ACT\_QUANT\_NORM --Количество оказанных услуг в месяц в рамках норматива.

FROM #FOUND\_AGREGATION foundAgregation

END

--//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

--Этап 6: Вставка суммы по услуге за календарный месяц.

--//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

IF (@thereIsError = 0) BEGIN

INSERT INTO WM\_COST\_SOC\_SERV\_MONTH(A\_YEAR, A\_MONTH, A\_SOC\_SERV\_MONTH, A\_SUM\_SOC\_SERV\_MONTH, A\_NORM\_EX, A\_EDITOWNER, A\_TS, A\_GUID, A\_STATUS, A\_CREATEDATE, A\_CROWNER, A\_AGR\_SOC\_SERV, A\_SYSTEMCLASS, A\_FULL\_COST, A\_PERCENT\_PART\_PAY, A\_SUM\_NORM\_EX, A\_IS\_PART\_PAY, A\_COND\_SOC\_SERV, A\_NORMSOCSERV, A\_ACT\_EXCESS\_QUANT)

SELECT

@yearReport\_INT AS A\_YEAR, --Год.

@monthReport\_INT AS A\_MONTH, --Месяц.

COUNT\_SERV\_NORMAL + COUNT\_SERV\_OVER AS A\_SOC\_SERV\_MONTH, --Количество оказанных услуг за месяц (Сумма норматива и превышения норматива).

0 AS A\_SUM\_SOC\_SERV\_MONTH,

CASE COUNT\_SERV\_OVER

WHEN 0 THEN 0

ELSE 1

END AS A\_NORM\_EX, --Норматив превышен (Если есть сумма превышения норматива).

CAST(NULL AS INT) AS A\_EDITOWNER, --Изменил.

GETDATE() AS A\_TS, --Дата модификации.

NEWID() AS A\_GUID, --Глобальный идентификатор.

10 AS A\_STATUS, --Статус.

GETDATE() AS A\_CREATEDATE, --Дата создания.

10314303 AS A\_CROWNER, --Автор.

SOC\_SERV\_AGR\_OUID AS A\_AGR\_SOC\_SERV, --Социальная услуга (Агрегация социальной услуги)

10284209 AS A\_SYSTEMCLASS, --Класс объекта.

--CASE @condition

-- WHEN 'free' THEN @costServOver

-- WHEN 'part' THEN @costServNormal / 50 + @costServOver

-- WHEN 'full' THEN @costServNormal + @costServOver

--END

0 AS A\_FULL\_COST, --Полная стоимость оказанных услуг, руб.

CASE CONDITION

WHEN 'part' THEN 50

ELSE 100

END AS A\_PERCENT\_PART\_PAY, --Размер частичной оплаты услуг, %

0 AS A\_SUM\_NORM\_EX, --Сумма превышения норматива, руб.

0 AS A\_IS\_PART\_PAY, --Без учета частичной оплаты

CONDITION AS A\_COND\_SOC\_SERV, --Условие оказания социальных услуг.

COUNT\_SERV\_NORMAL AS A\_NORMSOCSERV, --Количество оказанных услуг в рамках норматива.

COUNT\_SERV\_OVER AS A\_ACT\_EXCESS\_QUANT --Количество оказанных услуг в месяц сверх норматива.

FROM #FOUND\_AGREGATION

END

--//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

--Этап 7: Вставка стоимости всех оказанных услуг за календарный месяц

--//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

IF (@thereIsError = 0) BEGIN

--Условия оказания социальных услуг.

DECLARE @condition VARCHAR(256) SET @condition = (

SELECT

A\_COND\_SOC\_SERV

FROM WM\_COND\_SOC\_SERV WHERE A\_SOC\_SERV = @identifiedSocServ --Назначение

AND (@yearReport\_INT BETWEEN YEAR(A\_STARTDATE) AND ISNULL(YEAR(A\_LASTDATE), 9999) --Год отчета входит в период действия назначения.

AND (@yearReport\_INT <> YEAR(A\_STARTDATE) AND @yearReport\_INT <> ISNULL(YEAR(A\_LASTDATE), 9999) --Год отчета не равен крайнему.

OR @yearReport\_INT = YEAR(A\_STARTDATE) AND @monthReport\_INT >= MONTH(A\_STARTDATE) --Или равен начальному, но месяц позже начала.

OR @yearReport\_INT = YEAR(A\_LASTDATE) AND @monthReport\_INT <= MONTH(A\_LASTDATE) --Или равен конечному, но месяц раньше конца.

)

)

)

INSERT INTO WM\_FACT\_COST\_SOC\_SERV(TS,SYSTEMCLASS,GUID,A\_CREATEDATE,A\_CROWNER,A\_SERV,A\_SUM\_PERIOD,A\_FACT\_PAY,A\_EDITOWNER,A\_STATUS,A\_PAY\_DATE,A\_YEAR,A\_MONTH,A\_FULL\_COST\_MONTH,A\_SUM\_PAY,A\_PART\_PAY,A\_SUMM\_EXT\_SERV,A\_SUM\_NORM\_EX,A\_SUMM\_DOG,ESRN\_SOC\_SERV,A\_COND\_SOC\_SERV,A\_COMMENT,A\_SUM\_PERIOD\_BUDGET,A\_FULL\_COUNT,A\_SOC\_COUNT,A\_SOC\_COST\_MONTH,A\_DOP\_COUNT,A\_DOP\_COST\_MONTH,A\_OTHER\_COUNT,A\_OTHER\_COST\_MONTH)

SELECT

GETDATE() AS TS, --Дата модификации

10284266 AS SYSTEMCLASS, --Класс объекта

NEWID() AS GUID, --Глобальный идентификатор

GETDATE() AS A\_CREATEDATE, --Дата создания

10314303 AS A\_CROWNER, --Автор

CAST(NULL AS INT) AS A\_SERV, --Назначение

0 AS A\_SUM\_PERIOD, --Всего к оплате гражданином, руб.

'withoutPay' AS A\_FACT\_PAY, --Статус оплаты

CAST(NULL AS INT) AS A\_EDITOWNER, --Изменил

10 AS A\_STATUS, --Статус

CAST(NULL AS DATE) AS A\_PAY\_DATE, --Дата оплаты

@yearReport\_INT AS A\_YEAR, --Год

@monthReport\_INT AS A\_MONTH, --Месяц

0 AS A\_FULL\_COST\_MONTH, --Полная стоимость оказанных услуг, руб.

0 AS A\_SUM\_PAY, --Оплачено гражданином, руб.

0 AS A\_PART\_PAY, --Остаток к оплате, руб.

0 AS A\_SUMM\_EXT\_SERV, --Сумма по дополнительным услугам, руб.

0 AS A\_SUM\_NORM\_EX, --Сумма превышения норматива, руб.

CAST(NULL AS FLOAT) AS A\_SUMM\_DOG, --Размер частичной оплаты по договору, руб.

@identifiedSocServ AS ESRN\_SOC\_SERV, --Назначение на социальное обслуживание

@condition AS A\_COND\_SOC\_SERV, --Условие оказания социальных услуг

CAST(NULL AS VARCHAR) AS A\_COMMENT, --Комментарий

0 AS A\_SUM\_PERIOD\_BUDGET, --Всего к оплате из средств бюджета

SUM(t.TOTAL\_COUNT\_NORMAL) AS A\_FULL\_COUNT, --Количество оказанных услуг

SUM(t.TOTAL\_COUNT\_NORMAL) AS A\_SOC\_COUNT, --Количество оказанных соц. услуг

0 AS A\_SOC\_COST\_MONTH, --Полная стоимость оказанных соц. услуг, руб.

CAST(NULL AS INT) AS A\_DOP\_COUNT, --Количество оказанных доп. услуг

CAST(NULL AS FLOAT) AS A\_DOP\_COST\_MONTH, --Полная стоимость оказанных доп. услуг, руб.

CAST(NULL AS INT) AS A\_OTHER\_COUNT, --Количество оказанных иных услуг

CAST(NULL AS FLOAT) AS A\_OTHER\_COST\_MONTH --Полная стоимость оказанных иных услуг, руб.

FROM (

SELECT

socServAGR.A\_ID AS SOC\_SERV\_AGR\_OUID,

SUM(costSocServMonth.A\_NORMSOCSERV) AS TOTAL\_COUNT\_NORMAL

FROM ESRN\_SOC\_SERV socServ --Назначение социального обслуживания.

----Агрегация по социальной услуге.

INNER JOIN WM\_SOC\_SERV\_AGR socServAGR

ON socServAGR.ESRN\_SOC\_SERV = socServ.OUID

AND socServAGR.A\_STATUS = 10 --Статус агрегации в БД "Действует".

----Cуммы по услуге за календарный месяц.

INNER JOIN WM\_COST\_SOC\_SERV\_MONTH costSocServMonth

ON costSocServMonth.A\_AGR\_SOC\_SERV = socServAGR.A\_ID

AND costSocServMonth.A\_STATUS = 10

AND costSocServMonth.A\_YEAR = @yearReport\_INT

AND costSocServMonth.A\_MONTH = @monthReport\_INT

WHERE socServ.A\_STATUS = 10 --Статус назначения в БД "Действует".

AND socServ.OUID = @identifiedSocServ --Требуемое назначение.

GROUP BY socServAGR.A\_ID

) t

END

--//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

--Этап 8: Завершение

--//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

--Вывод результата.

SELECT @thereIsError AS thereIsError, @message AS message

--Удаление временных таблиц.

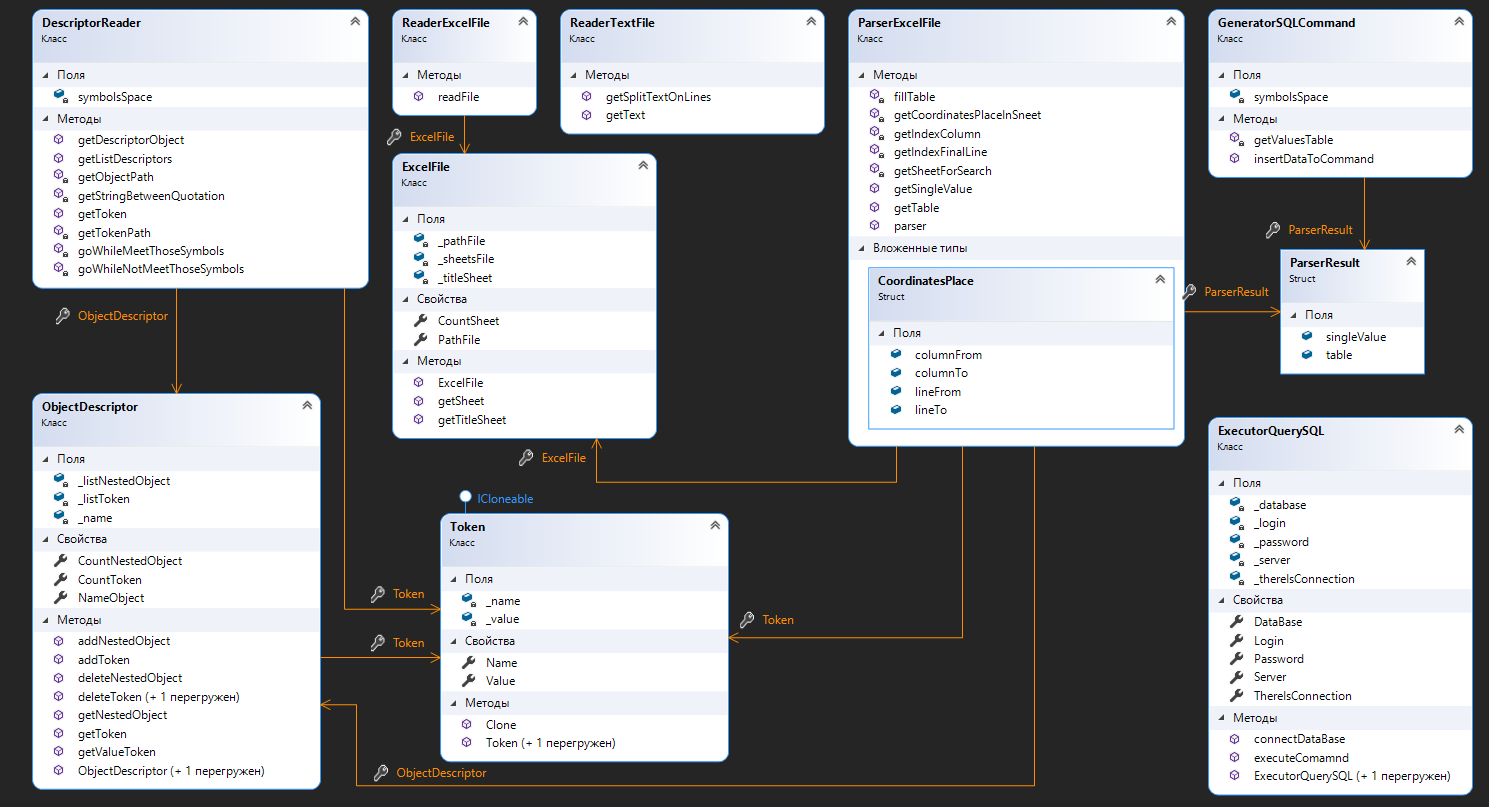
IF OBJECT\_ID('tempdb..#FOUND\_PEOPLE') IS NOT NULL BEGIN DROP TABLE #FOUND\_PEOPLE END --Найденные люди по входным данным.

IF OBJECT\_ID('tempdb..#FOUND\_SOC\_SERV') IS NOT NULL BEGIN DROP TABLE #FOUND\_SOC\_SERV END --Найденные социальные обслуживания по входным данным.

IF OBJECT\_ID('tempdb..#DATA\_FOR\_INSERV') IS NOT NULL BEGIN DROP TABLE #DATA\_FOR\_INSERV END --Данные для вставки.

# ПРИЛОЖЕНИЕ Г

# Диаграмма классов



# ПРИЛОЖЕНИЕ Д

# Алгоритм извлечения одиночного значения из файла



# ПРИЛОЖЕНИЕ Е

# Алгоритм извлечения таблицы из файла



# ПРИЛОЖЕНИЕ Ж

# Пользовательский интерфейс

