МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Рязанский государственный радиотехнический университет

им. В.Ф. Уткина»

Кафедра космических технологий

|  |  |  |
| --- | --- | --- |
|  |  | «К защите»  Заведующий кафедрой  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_»\_\_\_\_\_\_\_\_20\_\_\_ г. |

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ**

**РАБОТА БАКАЛАВРА**

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

на тему

**«Исследование метода Фелледжи-Хольта для автоматического редактирования статистических данных»**

Направление подготовки: 02.03.01 «Математика и компьютерные науки»

ОПОП «Математика и компьютерные науки»

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Левитова А.В.)

Научный руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Наумов Д.А.)

Руководитель ОПОП \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Таганов А.И.)

Рязань, 2021 г.

МИНИСТЕРСТВО НАУКИ И ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Рязанский государственный радиотехнический университет им. В.Ф. Уткина»

Утверждаю

Заведующий кафедрой

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_» \_\_\_\_\_\_20\_\_\_\_ г.

ЗАДАНИЕ

на дипломное проектирование

Студенту Левитовой А.В.

1.Тема проекта: Исследование метода Фелледжи-Хольта для автоматического редактирования статистических данных

2. Срок сдачи студентом законченного проекта:

3. Руководитель проекта Наумов Дмитрий Анатольевич, РГРТУ им. В.Ф. Уткина, доцент кафедры КТ

(фамилия, имя, отчество полностью, место работы, должность)

4.Исходные данные к проекту:

5. Содержание расчетно-пояснительной записки (технико-экономическое обоснование темы, расчетная, экспериментальная, экономическая часть и др. с расшифровкой задания по каждой части):

Введение.

1. Современные методы и технологии редактирования статистических данных

2. Разработка алгоритмов и структур данных для реализации метода Фелледжи-Хольта

3. Проектирование программной системы апробации метода Фелледжи-Хольта

4. Разработка программной документации

Заключение

Библиографический список

Приложения

Консультанты по проекту (с указанием относящихся к ним разделов проекта):

Консультант от кафедры\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания « \_\_\_» \_\_\_\_\_ 20\_\_\_ г.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Наумов Д.А.)

(подпись)

Задание принял к исполнению «\_\_\_\_» \_\_\_\_\_\_ 20\_\_\_ г.

Подпись студента \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Аннотация

Выпускная квалификационная работа изложена на 99 страницах, состоит из введения, четырех глав, заключения, включает 21 рисунок, 26 таблиц, библиографический список из 19 источников литературы, 1 приложение на 30 страницах.

Объектом исследования выпускной квалификационной работы является алгоритмы редактирования статистических данных.

Цель выпускной квалификационной работы – апробация алгоритмов восстановления некоректных и пропущеных данных на основе метода Филледжи-Хольта.

При разработке системы использовались следующие средства разработки:

* *с#*;
* формат базы данных: *Microsoft SQL Server 2012*;
* *Visual Studio Community*;
* *git*.

Выпускная квалификационная работа и ее результаты могут быть использованы в исследовательских и учебных целях.

СОДЕРЖАНИЕ

[Введение 6](#_Toc74139858)

[1. Современные методы и технологии редактирования статистических данных 11](#_Toc74139859)

[1.1. Спецификация правил редактирования 13](#_Toc74139860)

[1.2. Проверка состоятельности правил редактирования 15](#_Toc74139861)

[1.3. Обнаружение выделяющихся значений 17](#_Toc74139862)

[1.4. Метод Фелледжи-Хольта 17](#_Toc74139863)

[1.5. Генерация неявных правил редактирования 19](#_Toc74139864)

[1.6. Локализация ошибок 21](#_Toc74139865)

[1.7. Использование весов 22](#_Toc74139866)

[1.8. Методы импутации 23](#_Toc74139867)

[2. Разработка алгоритмов и структур данных для реализации метода Фелледжи-Хольта 31](#_Toc74139868)

[2.1 Алгоритм предварительной настройки, обработки и анализа данных 31](#_Toc74139869)

[2.2 Представление правил редактирования в нормальной форме 38](#_Toc74139870)

[2.3 Алгоритм получения полного множества правил редактирования 44](#_Toc74139871)

[2.4 Алгоритм определения минимального множества полей для импутации данных 46](#_Toc74139872)

[2.5 Алгоритм импутации данных 49](#_Toc74139873)

[3 Проектирование программной системы апробации метода Фелледжи-Хольта 55](#_Toc74139874)

[3.1 Формулировка требований к разрабатываемой системе 55](#_Toc74139875)

[3.2 Средства разработки программной системы 58](#_Toc74139876)

[3.3 Архитектура автоматизированной системы автокоррекции и импутации данных 59](#_Toc74139877)

[3.4 Информационная модель программной системы 63](#_Toc74139878)

[3.5 Тестирование программной системы 77](#_Toc74139879)

[4 Программная документация 89](#_Toc74139880)

[4.1. Описание применения 89](#_Toc74139881)

[4.2. Руководство системного программиста 90](#_Toc74139882)

[4.3. Руководство оператора 91](#_Toc74139883)

[БИБЛИОГРАФИЧЕСКИЙ СПИСОК 101](#_Toc74139884)

[ПРИЛОЖЕНИЕ А. ЛИСТИНГ ПРОГРАММНЫХ МОДУЛЕЙ 103](#_Toc74139885)

# Введение

Темой научно-исследовательской работы является "Исследование метода Фелледжи-Хольта для автоматического редактирования статистических данных".

Редактирование данных является одним из шагов подготовки статистических данных, и задача редактирования заключается в улучшении статистической информации. Накопленный опыт свидетельствует о том, что на редактирование может затрачиваться до 40% от общей стоимости обследования. При этом возникают следующие вопросы: оправдано ли расходование таких средств на редактирование, возможно ли применение более эффективных стратегий и можно ли повысить качество статистики за счет перераспределения части расходов, ранее приходившихся на редактирование данных, на другие методы предотвращения ошибок.

Главной задачей статистических организаций является такая организация своей деятельности, которая обеспечивает получение статистической продукции с наивысшим качеством, возможным при данных ресурсах. Качество статистической продукции можно определить как меру успешности производителя статистической продукции в обслуживании своих потребителей.

Качество статистической продукции определяется множеством факторов, включая соответствие (соответствие полученной и требуемой статистики), своевременность (интервал между моментом получения статистических данных и их использованием) и точность (различие между ценностью статистики, определяемой идеальным процессом сбора и обработки, и ценностью статистики, определяемой неидеальным процессом). В более широком смысле в определение качества статистики включаются также доступность, интерпретируемость и согласованность.

Влияние различных факторов на качество статистической продукции показано на рисунке В.1.



*Рис. В.1. Факторы, влияющие на качество статистической продукции*

Потребитель статистической продукции должен получить выгоду, чтобы оправдать процесс ее сбора и обработки, в связи с чем мы обязаны учитывать особенности рынка, на котором действуют производители и потребители статистической продукции. Рыночный механизм можно описать в виде двух функций: итоговой ценности продукции для потребителей как функции качества продукции и стоимости производства статистики также как функции качества.



*Рис. В.2. Распределение статистических процессов*

Подготовка статистики включает обычно множество процессов или действий. На рисунке В.2 представлена общая модель соотношения этих процессов с имеющимися ресурсами, методами и профессиональной компетентностью и вклад их в качество продукции. Редактирование направлено на повышение точности статистической продукции и, как правило, отбирает существенную часть бюджета обследования. Этим обусловливается интерес к анализу, настройке и совершенствованию процесса редактирования.

Процесс редактирования можно охарактеризовать с помощью четырех блоков характеристик:

1. *Архитектура* процесса редактирования, под которой понимаются принципы построения процесса редактирования. Это могут быть, например, алгоритмы обнаружения ошибок, алгоритмы автоматической коррекции ошибок одного вида и инструкции по ручной обработке ошибок других видов.
2. *Параметры* процесса редактирования, которые могут включать численные границы, используемые при редактировании, функции замены ошибочных значений и т. д. Эти данные подробно описывают процесс редактирования со всеми спецификациями.
3. *Показатели качества* статистической продукции. Эти данные включают также информацию о том, каким образом было достигнуто данное качество статистики.
4. *Стоимость* статистической продукции. Включает информацию о том, какие виды ресурсов были задействованы и как распределились затраты по различным видам деятельности.

Описание процесса редактирования посредством перечисленных четырех блоков данных, кроме обеспечения полезной информации для оценки и упорядочивания процесса, обеспечивает индикаторы для оценивания альтернативных подходов к редактированию с целью улучшения качества статистики.

Процесс редактирования подразделяется на два этапа:

1. классификация наблюдений на приемлемые и подозрительные;
2. коррекция наблюдений, признанных ошибочными (импутация).

До начала автоматизации обработки статистических данных редактирование выполнялось специалистами, зачастую на основе нескольких формальных правил редактирования. Позднее появились новые возможности за счет внедрения компьютерной обработки, редактирование стало более эффективным, но также требовало формализации процесса. В настоящее время имеется широкий спектр методов и инструментов редактирования.

Для наблюдения за процессом редактирования можно использовать различные индикаторы. Вот некоторые типичные примеры показателей, которые могут отслеживаться в процессе редактирования:

* общее число наблюдений;
* число отбракованных наблюдений;
* число замененных наблюдений;
* сумма значений по всем наблюдениям до редактирования (“сырых”);
* сумма значений по отбракованным наблюдениям до редактирования;
* сумма замененных значений по отбракованным наблюдениям;
* сумма отредактированных значений по всем наблюдениям;
* стоимость управления редактированием;
* стоимость замены.

Объектом научной исследовательской работы является исследование алгоритма Феллежди-Холья для автоматизированного редактирования статистических данных.

Пояснительная записка к дипломному проекту состоит из введения, четырех разделов, заключения и приложений. В заключении описана успешность и полнота решения поставленных задач выпускной квалификационной работы, дается оценка перспектив развития вопросов по теме работы.

# 1 Современные методы и технологии редактирования статистических данных

Под редактированием статистических данных понимается деятельность, направленная на обнаружение и обработку ошибок в данных. Процесс редактирования статистических данных в современном понимании, как правило, включает три этапа:

* оценку состоятельности правил редактирования;
* проверку данных на предмет удовлетворения всем правилам редактирования;
* импутацию (подстановку) данных, не удовлетворяющих правилам редактирования.

Часто под редактированием в более узком смысле понимают первые два этапа, направленные на обнаружение ошибок в данных. Импутация в этом случае рассматривается как самостоятельный процесс.

Импутация (вменение, подстановка) данных — подстановка значений вместо пропущенных или несостоятельных полей. Импутация имеет своей целью получение таких записей, которые удовлетворяют всем правилам редактирования. Методы импутации рассматриваются в последней части данного раздела.

Методы редактирования статистических данных можно разделить на две категории [1]:

1. Обычные методы.
2. Методы типа Фелледжи-Хольта.

Обычные методы редактирования в основном базируются на конструкциях типа IF‑THEN‑ELSE, с помощью которых определяются поля, подлежащие импутации. Проблема в применении подобных правил редактирования, помимо отсутствия оптимизации количества импутируемых значений, заключается в том, что даже при небольших изменениях в формах обследования и правилах редактирования исходные тексты программ (это могут быть тысячи строк исходного текста) должны изменяться и заново отлаживаться.

Кроме того, с ростом числа правил редактирования резко возрастает время, потребное для проверки состоятельности всей системы правил и отладки программы. На практике это приводит к тому, что разработчики сокращают количество правил до разумных, с точки зрения программирования и отладки, пределов, не используя всех потенциальных возможностей контроля.

В отличие от этого, методы Фелледжи-Хольта не требуют внесения изменений в код в подобных случаях и могут применяться специалистами по проведению обследований (статистиками, экономистами) без привлечения высококвалифици­рованных программистов и разработки программы для каждого обследования, ее отладки и проверки.

Общая схема процесса редактирования и импутации с применением технологии минимума изменений Фелледжи-Хольта приведена на рис. 2.1. Схема отражает общие современные подходы к реализации процесса редактирования и импутации статистических данных и не отражает, конечно, всех возможных вариантов. Методы, лежащие в основе элементов этой схемы, рассматриваются ниже.



*Рис. 1.1. Общая схема процесса редактирования и импутации*

## 1.1. Спецификация правил редактирования

Соотношения, связывающие между собой переменные, и используемые для контроля корректности данных обследования, называются правилами редактирования. Запись считается корректной, если она удовлетворяет всем правилам редактирования. Теоретически правила редактирования могут задаваться в самых различных формах и содержать различные линейные и нелинейные функции. Однако на практике в основном применяются следующие виды правил редактирования.

1. Линейные неравенства.
2. Неравенства с отношениями переменных.
3. Уравнения баланса (равенства).

Переменные (поля), входящие в правила редактирования, могут быть метрическими или категориальными. Обычно правило редактирования содержит переменные одного типа. В обследованиях социального характера большая часть переменных — категориальные, тогда как в обследованиях предприятий — количественные (метрические).

### 1.1.1. Линейные неравенства

Правила редактирования в виде линейных неравенств имеют следующий вид

,

,

где — задаваемые пользователем константы; — переменные (ответы на вопросы обследования).

### 1.1.2. Отношения

Правила редактирования в виде отношений определяют границы значений отношения пар переменных. Границы определяются на основе априорных данных или на основе разведочного анализа данных обследований.

Неравенства с отношениями задаются в следующем виде

,

где и — соответственно нижняя и верхняя границы; — переменные.

### 1.1.3. Уравнения баланса

С помощью правил редактирования в виде уравнений баланса проверяют равенства итоговых значений суммам составных частей. Такие правила называют также бухгалтерскими.

Уравнения баланса могут выделяться в отдельный класс правил редактирования, а могут рассматриваться как частный случай линейных неравенств. В последнем случае вводится, анализируется и используется для контроля вся система линейных неравенств и равенств без выделения уравнений баланса в отдельную группу.

### 1.1.4. Нелинейные правила

Наряду с правилами в виде линейных неравенств, отношений и уравнений баланса, между переменными могут существовать и быть априорно известными нелинейные соотношения, однако обработка нелинейных правил редактирования в автоматическом режиме крайне затруднительна и не предусматривается ни в одной из систем.

## 1.2. Проверка состоятельности правил редактирования

Современные технологии редактирования предусматривают автоматическую проверку состоятельности правил редактирования, целями которой являются [1, 2, 3, 4]:

* проверка состоятельности (непротиворечивости) правил;
* выявление избыточных правил;
* определение скрытых равенств в группе неравенств;
* выявление детерминированных переменных.

Система правил редактирования определяет некоторую область допустимых значений переменных в *n*-мерном пространстве. Под состоятельностью системы правил редактирования понимается то, что область допустимых значений является не пустой. Несостоятельной системе правил редактирования не может удовлетворять ни одна запись, потому что в нее входят противоречивые правила.

*Проверка состоятельности* правил редактирования осуществляется на основе методов линейного программирования, в частности, с использованием улучшенной версии симплекс-метода. В случае обнаружения несостоятельности группы правил редактирования может предусматриваться выделение подмножества правил редактирования, удаление которого из исходной группы правил приводит к состоятельной системе правил.

Проверка состоятельности системы правил редактирования и выделение подмножества правил, вступающих в противоречие с остальными, существенно облегчают работу специалистов в предметной области, разрабатывающих правила редактирования и не требует привлечения к этой работе программистов. Подобная проверка предусматривается в наиболее продвинутых системах редактирования и импутации, таких как GEIS [2, 4, 5], AGGIES [6, 5].

Проверка на наличие *избыточных* правил означает следующее. Избыточное правило редактирования не формирует какую-либо часть границы области допустимых значений переменных и, следовательно, в действительности не содержит ограничений на приемлемые значения переменных. Такое правило является избыточным и может быть без всякого ущерба исключено из набора правил редактирования. Исключение избыточных правил повышает вычислительную эффективность процесса редактирования.

Группа правил редактирования из линейных неравенств может содержать также *скрытые равенства*. Такие скрытые равенства могут быть неочевидными, если правила включают несколько переменных. Наличие скрытых равенств обычно говорит о том, что система правил редактирования содержит более жесткие ограничения, чем этого хотел пользователь при вводе неравенств. При выявлении скрытых равенств пользователь должен решить, оставлять ли эти скрытые равенства в системе правил редактирования, или же эти равенства являются следствием некорректностей в правилах и их следует исправить.

Выявление *детерминированных переменных* выполняется с целью выявления возможных некорректностей в правилах редактирования. Для каждой переменной определяются верхняя и нижняя границы допустимых значений в соответствии с областью, задаваемой правилами редактирования. Если оказывается, что для какой-либо переменной область допустимых значений представляет собой точку, это говорит о детерминированности такой переменной и может свидетельствовать об ошибках, допущенных при определении правил редактирования.

Все перечисленные проверки системы правил редактирования направлены на выявление возможных ошибок в правилах и существенно облегчают их отладку, в особенности при достаточно большом количестве правил и переменных.

## 1.3. Обнаружение выделяющихся значений

Обнаружение выделяющихся значений или выбросов является составной частью процесса редактирования. Эта функция реализована во многих системах редактирования и импутации.

Наибольшее распространение в качестве метода обнаружения выделяющихся значений получил метод Хидироглы-Бертелота, предложенный в 1986 году. Находит применение также графический метод обнаружения выбросов (модуль MacroView в системе SLICE) [5,7].

В системах с развитыми средствами редактирования, например GEIS, выделяющиеся значения могут подразделяться на два класса. В первую группу входят резко выделяющиеся значения, которые очевидно являются ошибочными. Эти значения подлежат импутации. Во вторую группу входят менее резко выделяющиеся значения, которые не импутируются, но исключаются из списка значений, используемых в качестве донорских.

Обнаружение выделяющихся значений может быть основано на анализе данных как текущего обследования, так и на анализе данных нескольких обследований. Во втором случае вместе со статистикой Хидироглы-Бертелота применяется метод исторических трендов.

## 1.4. Метод Фелледжи-Хольта

Общая характеристика подхода Фелледжи-Хольта к редактированию данных может быть сформулирована следующим образом [1, 8]:

1. Каждая запись данных должна удовлетворять всем правилам редактирования и достигаться это должно за счет изменения (импутации) наименьшего количества полей.
2. Правила импутации должны автоматически генерироваться из правил редактирования.
3. Процесс импутации должен поддерживать совместное распределение переменных.

Первый этап редактирования статистических данных методом Фелледжи-Хольта состоит в *локализации ошибок*. На этом этапе проверяется выполнение как явных, так и неявных правил редактирования. *Неявные правила* редактирования автоматически генерируются из явных правил. Если не выполняются какие-либо из неявных правил редактирования, это означает, что не выполняется, как минимум, одно из явных правил, использовавшихся при генерации неявных правил редактирования. Методы редактирования, предлагавшиеся многими авторами до появления в 1976 году работы Фелледжи-Хольта, не гарантировали того, что запись после импутации полей, указанных тем или иным методом, будет удовлетворять всем правилам редактирования.

Основополагающая теорема Фелледжи-Хольта утверждает, что всегда возможно найти такой набор полей в записи, изменение (импутация) которых обеспечит удовлетворение всех правил редактирования данной записью. Неявные правила редактирования обеспечивают информацию относительно явных правил редактирования, которым удовлетворяет исходная (неотредактированная) запись.

Хорошо разработанная система Фелледжи-Хольта очень значительно повышает эффективность редактирования данных и производительность, поскольку при использовании такой системы нет необходимости в том, чтобы команда высококвалифицированных программистов и специалистов в предметной области обследования разрабатывала для каждого обследования систему состоятельных правил редактирования. Метод Фелледжи-Хольта автоматизирует этот процесс. Правила редактирования для обследования могут разрабатываться одним специалистом в предметной области обследования.

С практической точки зрения система Фелледжи-Хольта характеризуется следующим образом:

* Правила редактирования представляются в виде легко модифицируемых таблиц.
* Логическая состоятельность всей системы правил редактирования может быть проверена до того, как будут получены данные обследования.
* Основная часть системы Фелледжи-Хольта реализуется в виде набора программ, который может использоваться без каких-либо изменений в других обследованиях.
* После единственного прохода через систему запись гарантированно удовлетворяет всем правилам редактирования.

## 1.5. Генерация неявных правил редактирования

Реализация системы Фелледжи-Хольта для категориальных данных, когда правила редактирования могут быть произвольными, и для непрерывных метрических данных, когда правила редактирования представляются в виде неравенств с отношениями или линейных неравенств, существенно различается. Поскольку большая часть переменных в обследованиях предприятий являются метрическими, рассмотрим принципы применения метода Фелледжи-Хольта на примере метрических переменных.

На практике алгоритмы генерации всех неявных правил редактирования разработаны только для правил редактирования в виде систем линейных неравенств и систем неравенств с отношениями переменных. Причем генерация правил редактирования для специального случая отношений переменных существенно легче. Это можно показать на следующем примере. Если в неравенствах с отношениями фигурирует *n* полей (переменных), то может существовать максимум *n(n­ 1)/2* явных и неявных правил редактирования.

В случае же правил редактирования в виде линейных неравенств количество неявных правил резко возрастает. Каждое неявное правило редактирования получается в результате линейной комбинации k явных правил редактирования. В результате в неявном правиле редактирования исключается как минимум (k-1) переменных. Количество неявных правил редактирования всегда больше исходного количества явных правил и это увеличение может быть очень существенным. Например, 30 явных правил редактирования и 10 полей (переменных) могут приводить к генерации 400 неявных правил. Количество правил редактирования возрастает при генерации неявных правил, но количество переменных сокращается, что и является основной целью подхода Фелледжи-Хольта.

Генерация неявных правил редактирования в связи с этим может занимать 24 часа на компьютере умеренной производительности для умеренного количества исходных линейных неравенств [1], если не применять каких-либо методов ускорения этого процесса.

Существенного ускорения генерации неявных правил редактирования удалось добиться за счет применения метода Черниковой Н.В. [9, 10] генерации вершин ограниченной области в *Rn,* границы которой задаются явными правилами редактирования в виде линейных неравенств. В 1979 году Г. Санд предложил изящный метод, основанный на алгоритме Черниковой, определения множества решений проблемы локализации ошибок без необходимости генерации всех неявных правил редактирования, поскольку минимум и максимум в области в *Rn* достигаются на вершинах.

Базовый алгоритм Черниковой был все же недостаточно быстрым, Санд модифицировал его в алгоритм генерации вершин с ограничениями, использовав результаты Д. Рубина. Дальнейшие теоретические и эвристические результаты позволили достичь еще большего повышения быстродействия. Так одна из эвристических модификаций базового алгоритма, реализованная в пакете GEIS статистического ведомства Канады, позволила достичь 60-кратного ускорения. Эта же версия алгоритма Черниковой реализована в пакете AGGIES, разработанном Департаментом сельскохозяйственной статистики США.

## 1.6. Локализация ошибок

Целью процесса локализации ошибок является идентификация минимального количества полей в каждой отдельной записи, изменение которых обеспечит выполнение всех правил редактирования для данной записи. Это — один из основных принципов метода Фелледжи-Хольта, целесообразность применения которого интуитивно понятна, поскольку максимально возможная часть данных остается неизменной в процессе редактирования, то есть в том виде, в котором эти данные получены от респондентов.

Принцип минимизации количества изменений означает, что всегда для импутации выбирается одно поле, даже если в значение этого поля вносится большое изменение, а не два поля с небольшими изменениями в их значениях.

В процессе локализации ошибок идентифицируются поля, изменение которых позволит переместить запись в область допустимых значений. При этом могут определяться также и сами изменения в переменных, внесение которых перемещает запись в область допустимых значений. Однако на практике такие решения не используются, поскольку полученное таким способом решение лежит на границе области допустимых значений. Импутируемые значения предпочтительнее определять отдельно, например, методом донора.

Для произвольной записи проблема локализации ошибок может иметь несколько решений. Например, если запись не удовлетворяет правилу редактирования

,

то не имеет значения какая из переменных будет импутирована, в предположении, что ни одна из этих переменных не нарушает других правил редактирования. Причем во всех решениях изменению подлежит одинаковое минимально возможное количество переменных, в данном случае — одна переменная.

В процессе локализации ошибок определяются все решения, содержащие минимальное количество подлежащих импутации полей. Выбор одного из этих решений может осуществляться в соответствии с каким-либо дополнительным критерием или случайно. Чаще всего применяется случайный выбор.

## 1.7. Использование весов

Модификация метода Фелледжи-Хольта, позволяющая пользователю устанавливать приоритеты в выборе переменных, подлежащих импутации, когда имеется несколько решений проблемы локализации ошибок, заключается в назначении весов переменным. В этом случае минимизируется не количество подлежащих импутации переменных, а сумма весов переменных. Например, если переменная имеет вес равный двум, это означает, что решение с изменением этой переменной равноценно решению с изменением двух переменных, имеющих единичные веса.

Переменные, которым назначены большие веса, импутируются реже, чем переменные, которым назначены меньшие веса. Таким образом пользователь получает возможность определять переменные, импутация которых нежелательна, но возможна в отсутствие других вариантов.

Основание для назначения весов переменным может быть большая надежность одних данных (переменных) и меньшая надежность других. Например, может наблюдаться такая тенденция, что респонденты указывают более точную информацию относительно совокупного дохода предприятия и несколько заниженные данные относительно фонда заработной платы. В этом случае более надежным переменным могут быть назначены большие веса с тем, чтобы при выборе между двумя переменными более надежная из них оставалась неизменной, а менее надежная подвергалась импутации.

Веса могут использоваться также с той целью, чтобы исключить импутацию некоторых переменных. Это бывает полезным, когда переменная входит в несколько групп данных. Импутация таких переменных может приводить к тому, что запись, удовлетворяющая правилам редактирования в одной группе (в исходном виде или после импутации) перестает удовлетворять этим правилам после редактирования и импутации в другой группе. Подобной проблемы можно избежать за счет назначения переменным, присутствующим в правилах редактирования в предыдущей группе, весов, превышающих сумму весов остальных переменных. Очевидно, что в такой ситуации очень важным оказывается порядок обработки групп данных и правил редактирования и выбору правильной последовательности обработки следует уделять большое внимание.

Веса часто применяются для правил редактирования в виде уравнений баланса для того, чтобы некоторые переменные относительно реже подвергались импутации. Пусть, например, имеется уравнение баланса, в котором сумма четырех переменных равна пятой переменной. Если более предпочтительно, чтобы в случае несоответствия корректировалась пятая переменная, то можно достичь этого, назначив единичные веса первым четырем переменным и вес между 2 и 3 пятой переменной. С другой стороны, если предпочтительнее корректировать составные части, а не итоговое значение, нужно назначить пятой переменной вес, несколько меньший весов остальных переменных.

## 1.8. Методы импутации

Под импутацией понимаются процедуры определения значений для элементов данных, по которым ответы респондентов отсутствуют или неприемлемы (ошибочны).

К настоящему времени предложено множество методов импутации. Наиболее широко применяемые методы можно свести в две группы: методы донора и методы оценок. В методах донора импутируется значение из другой записи без каких-либо изменений. Методы различаются способами выбора донорской записи. В методах оценок импутируются разнообразные функции переменных текущего или исторических обследований. Кроме того, технология Фелледжи-Хольта предполагает возможность применения еще одного метода — детерминистского, сущность которого поясняется ниже.

Следует также отметить, что редактирование и импутацию не всегда можно отделить друг от друга и оценить раздельно их влияние на качество результирующей статистики. В особенности это относится к технологии Фелледжи-Хольта. Так, например, в рамках Унифицированного обследования предприятий, объединившем около 200 отдельных обследований, статистическим ведомством Канады проводилось сравнение некоторых методов редактирования и импутации [11]. В частности, отмечалось, что если метод перевзвешивания (эквивалентный импутации средних значений) обнаруживает тенденцию к существенному завышению оценок, то метод минимума изменений Фелледжи-Хольта приводит к распределению оценок, более приближенному к действительному распределению наблюдений в генеральной совокупности.

### 1.8.1. Детерминистский метод

Под детерминистской импутацией понимается ситуация, когда для данной записи возможен только один определенный набор значений переменных, обеспечивающий удовлетворение всех правил редактирования. Например, если импутации подлежит одна переменная, а правило редактирования является уравнением баланса, то существует только одно определенное значение этой переменной, которое может быть импутировано.

В системах редактирования и импутации, реализующих метод Фелледжи-Хольта (AGGIES, GEIS и др.), возможность импутации детерминистским методом проверяется в первую очередь.

Детерминистский метод импутации предполагает выполнение следующих шагов:

* Исключение правил редактирования, которые удовлетворяются данной записью.
* В оставшиеся правила редактирования подставляются значения полей, не подлежащих импутации.
* Вычисляются максимальные и минимальные допустимые значения по каждой переменной.
* Если минимальное и максимальное значения совпадают, это и является импутируемым значением.

### 1.8.2. Методы донора

В соответствии с методом донора импутируемые значения берутся из записей, удовлетворяющих правилам редактирования. Это могут быть записи текущего обследования или исторические данные. Методы донора достаточно просты в реализации, но дают хорошие результаты. В ряде случаев они оказываются более предпочтительными, чем более сложные методы оценок или множественной импутации.

### 1.8.3. Случайный выбор

Случайный выбор записей, используемых в качестве доноров, обычно осуществляется в некоторых группах, называемых стратами, слоями или классами. Страты представляют собой достаточно однородные по заданным параметрам группы наблюдений.

Если в записи-реципиенте подлежат импутации несколько полей, то они все берутся из одной отобранной записи-донора. Это обеспечивает сохранение соотношений между переменными и гарантирует выполнение правил редактирования.

### 1.8.4. Метод ближайшего соседа

Импутация методом ближайшего соседа предполагает выбор такой записи из числа донорских, которая имеет наименьшее расстояние от импутируемой записи. В качестве метрики для определения расстояния могут выбираться различные функции, возможно, учитывающие некоторые априорные или исторические сведения о коррелированности некоторых параметров.

В методе ближайшего соседа необходимо определить не только метрику, но и поля, по которым будет оцениваться расстояние. Для каждой требующей импутации записи набор полей, по которым производится подбор ближайшего соседа, может быть различным. Кроме того, может оказаться, что все поля, присутствующие в метрике, требуют импутации в данной записи. В этом случае может быть подобрана по другому критерию, например, методом случайного выбора.

В конкретных реализациях метода ближайшего соседа могут применяться технологии, подобные технологиям добычи данных (Data Mining), сокращающие время выполнения операций подбора. Например, в системе GEIS для подбора ближайшего соседа применяется дерево решений.

### 1.8.5. Методы оценок

Существует множество методов оценок, применяемых для импутации пропущенных или несостоятельных значений в данных. Общая идея этих методов заключается в подстановке вместо отсутствующего значения некоторой оценки, которая может основываться на данных текущего обследования или на исторических данных предыдущих обследований. Следует отметить, что в случае применения любого из методов оценок всегда подразумевается в явном или неявном виде применение некоторой модели данных. Поэтому выбор той или иной оценки должен основываться на некоторых априорных или экспериментальных данных относительно модели.

После импутации методом оценок может потребоваться повторная проверка выполнения правил редактирования и, возможно, повторная импутация.

### 1.8.6. Предшествующие и текущие значения и средние

Это наиболее простые, но в ряде случаев достаточно эффективные оценки, применяемые для импутации. В этом классе имеются в виду три вида оценок:

* предшествующие (исторические) значения;
* текущие средние;
* предшествующие средние.

Средние значения могут вычисляться как по всем корректным записям текущего или предшествующего обследования, так и внутри некоторых классов (слоев), то есть с применением группирующих переменных. В качестве группирующей переменной может выступать, например, отрасль промышленности.

### 1.8.7. Тренды

Предшествующие значения и средние могут применяться в качестве оценок для импутации в совокупности с оценками трендов. Тренды могут вычисляться по некоторой вспомогательной переменной или переменным.

### 1.8.8. Регрессия

В качестве регрессионных оценок для импутации значений могут использоваться результаты оценки парной регрессии, множественной регрессии, авторегрессии (по лонгитюдных обследованиям). В подавляющем большинстве случаев применяется линейная (по оцениваемым коэффициентам) регрессия. Приведем некоторые примеры импутации на основе регрессионных оценок.

### 1.8.9. Байесовская оценка

Импутация значений на основе неявной байесовской модели можно разделить на три этапа [12, 13]:

* моделирование;
* оценивание;
* импутация.

Моделирование предполагает выбор подходящей модели данных. В результате оценивания определяется апостериорное распределение параметров выбранной модели, чтобы из него можно было случайным образом выбрать значение. На этапе импутации случайным образом выбирается одно значение для импутируемого значения *y* из апостериорного распределения.

### 1.8.10. Множественная импутация

Метод множественной импутации, разработанный Д. Рубином [14] и реализованный в коммерческом программном пакете SOLAS [15,16,5] ирландской компании Statistical Solutions, предполагает замену каждого пропущенного значения двумя или более импутируемыми значениями и получение результирующего значения путем их комбинации. Множественную импутацию можно рассматривать как усреднение нескольких байесовских оценок.

Множественная импутация на основе выявления тенденций предполагает следующие шаги [16]:

* на основе имеющихся данных создается логистическая регрессионная модель для пропущенных значений;
* на основе построенной модели логистической регрессии вычисляются тенденции и соответствующие оценки пропущенных значений на данный момент;
* наблюдения группируются на основе квантилей, внутри каждого квантиля формируется апостериорная плотность вероятности, на основе которой случайным образом выбираются по одному значению для каждого пропущенного;
* вся описанная процедура повторяется с тем, чтобы сформировать M полных наборов данных.

Он может применяться как для лонгитюдных, так и для одномоментных обследований. Пользователь может управлять при этом многими параметрами импутации.

### 1.8.11. Импутация с применением нейронных сетей

Нейросетевые технологии в применении к импутации пропущенных и некорректных данных рассматриваются как альтернатива, при определенных условиях, методам донора.

Нейронные сети в данном приложении представляют собой класс непараметрических регрессионных методов, применяемых для классификации и прогнозирования в тех случаях, когда классические статистические методы оказываются неэффективными. В отличие от классических статистических методов, как правило основывающихся на некоторых предположениях относительно закона распределения случайной величины, нейросетевые алгоритмы этого не требуют. Нейронная сеть автоматически подбирает модель данных, однако платой за это является необходимость очень большой выборки для настройки сети, если модель оказывается существенно нелинейной.

Имеются некоторые результаты применения нейронных сетей для импутации данных переписи населения, сельскохозяйственного обследования и др. [17,15,18,19].

В целом нейросетевой метод показал более низкие результаты, по сравнению с другими методами импутации, например, методами донора, в обработке результатов переписи населения 1991г. в Великобритании и не рекомендован к применению в переписи 2001г. [17]. В сельскохозяйственном обследовании результаты применения нейронной сети для импутации (в SAS) были неоднозначными. В частности, при импутации непрерывных переменных нейронная сеть показала худшие результаты по сравнению с линейной регрессией.

В целом можно отметить, что нейронные сети не нашли широкого применения для импутации статистических данных. Гораздо более простые в реализации методы пока дают лучшие результаты, чем нейронные сети.

# 2. Разработка алгоритмов и структур данных для реализации метода Фелледжи-Хольта

## 2.1 Алгоритм предварительной настройки, обработки и анализа данных

В соответствие с обобщенной схемой процесса автокоррекции и импутации первым этапом является этап инициализации и настройки системы на работу с конкретным типом переписных листов (анкет). Данный этап является обязательным, так как без этого этапа невозможно извлечение информации из базы данных. Задачи данного этапа выполняются компонентой предобработки данных. На данном этапе необходимо:

* для каждого типа анкет определить набор полей;
* определить множества допустимых значений (домен) для каждого поля.

### 2.1.1. Описание структуры анкетных данных

На этапе описания структуры анкеты требуется определение информации, задающей структуру данных, а также способ их получения и обработки, а именно:

* описание видов переписных листов;
* описание таблиц базы данных, хранящих информацию переписных листов;
* описание вопросов (показателей) по каждому переписному листу и допустимых ответов на каждый вопрос.

Для описания видов переписных листов необходимо лишь указать название и его расшифровку (табл. 2.1).

Таблица 2.1.

|  |  |  |
| --- | --- | --- |
| **№** | **Название переписного листа** | **Примечание** |
| **1** | 3ЛПХ | Переписной лист личных подсобных и других индивидуальных хозяйств населения |
| **2** | 3ЛПХ(КС) | Приложение к переписному листу личных подсобных и других индивидуальных хозяйств населения для районов Крайнего Севера и приравненных к ним местностей |
| **3** | 2КФХ | Переписной лист крестьянских (фермерских) хозяйств и индивидуальных предпринимателей |
| **4** | 2КФХ(КС) | Приложение к переписному листу крестьянских (фермерских) хозяйств и индивидуальных предпринимателей для районов Крайнего Севера и приравненных к ним местностей |

Для описания таблиц по каждому переписному листу необходимо указывать следующие данные (табл. 2.2):

* номер переписного листа, к которому относится таблица;
* имя таблицы, как оно задано в базе данных системы автоматизированной обработки данных сельскохозяйственной переписи;
* примечание к таблице;
* имя ключевого поля таблицы, как оно задано в базе данных системы автоматизированной обработки данных сельскохозяйственной переписи;
* имя поля – идентификатора хозяйства, как оно задано в базе данных системы автоматизированной обработки данных сельскохозяйственной переписи;
* тип таблицы – главная или вспомогательная; по каждому анкетному листу может быть только одна главная таблица и несколько вспомогательных.

Все показатели (вопросы) переписных листов разделяются на два вида:

* *качественные* показатели;
* *количественные* показатели.

К качественным показателям относятся показатели, которые могут иметь конечный набор значений или кодов. К качественным показателям относятся вопросы, на которые респондент может дать ответы «Да-Нет», ответы выбора одной из нескольких альтернатив.

Множество допустимых значений показателя будем называть доменом значений показателя, и, в соответствии с видами показателей переписных листов, также делятся на два вида: домены количественных показателей и домены качественных показателей (табл. 3.2).

Таблица 2.2.

|  |  |
| --- | --- |
| № | **Вид домена** |
| 1 | Домен количественного показателя |
| 2 | Домен качественного показателя |

При выполнении данного этапа требуется проанализировать каждый вопрос переписного листа и отнести его либо к существующему домену, либо создать новый домен. В табл. 2.3 представлены домены для Формы 3 (3ЛПХ).

Таблица 2.3.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Примечание** | **Вид домена** | **Мин.  значение** | **Макс.  значение** |
| **1** | Положительные вещественные числа | 1 | 0 | 1000000 |
| **2** | Положительные целые числа | 1 | 0 | 1000000 |
| **3** | Ответ Да-Нет | 2 | 1 | 2 |
| **4** | Пол | 2 |  |  |
| **5** | Тип сельскохозяйственного кооператива | 2 | 0 | 32 |
| **6** | Инфраструктура, техника и производственные помещения | 2 | 0 | 64 |
| **7** | Перерабатывающие мощности по производству | 2 | 0 | 256 |

Если вид домена - домен качественных показателей, то для него необходимо перечислить все возможные значения (и их коды). Данный шаг необходим для этапа построения правил редактирования качественных показателей, так как каждое правило редактирования качественных показателей задает *недопустимую комбинацию* *возможных значений* качественных показателей*.*

Кроме множества допустимых значений для каждого домена качественных показателей добавляется значение «Другое», которое как бы будет содержать все остальные (недопустимые значения). Добавление данного значения является *обязательным* шагом, так как на этапах проверки правил редактирования, локализации ошибки и импутации обязательно должны быть проверены *простые* правила.

Если в некотором вопросе переписного листа допустим неответ, то он также должен быть включен в список *допустимых* значений показателя домена и будет присутствовать в правилах редактирования наравне с другими допустимыми значениями показателей.

Таблица 2.4.

|  |  |  |  |
| --- | --- | --- | --- |
| **№** | **Номер домена** | **Код значения** | **Значение** |
| **1** | 2 | 1 | Да |
| **2** | 2 | 2 | Нет |
| **3** | 2 | - | Другое |
| **4** | 4 | 0 | Цель не указана |
| **5** | 4 | 1 | Самообеспечение продовольствием |
| **6** | 4 | 2 | Дополнительный источник денежных средств |
| **7** | 4 | 4 | Основной источник денежных средств |
| **8** | 4 | - | Другое |
| **9** | 5 | 0 | Не указан тип кооператива |
| **10** | 5 | 1 | Перерабатывающий |
| **11** | 5 | 2 | Сбытовой |
| **12** | 5 | 4 | Снабженческий |
| **13** | 5 | 8 | Обслуживающий |
| **14** | 5 | 16 | Кредитный |
| **15** | 5 | 32 | Страховой |
| **16** | 5 | - | Другое |

Завершающим шагом этапа предварительной настройки системы является задание информации о самих вопросах переписного листа и способах их хранения в таблицах базы данных. Для выполнения данного шага для каждого показателя следует задать:

* таблицу, к которой относится показатель;
* имя поля в таблице, которое содержит ответы на соответствующий вопрос переписного листа;
* код справочника (задается для вспомогательных таблиц);
* полное и сокращенное обозначение показателя;
* код ЭОД;
* номер домена, к которому принадлежит показатель.

Для построения правил редактирования количественных и качественных показателей на последующих этапах работы системы может потребоваться задание вспомогательных показателей (так называемых «смешанных» показателей). Для некоторых правил редактирования количественное поле может быть преобразовано к качественному представлению для использования в правилах редактирования качественных показателей.

### 2.1.2. Построение баз данных доноров и реципиентов

Информационной основой для построения баз доноров и реципиентов является общая база данных, прошедшая этап формально-логического контроля.

Присвоение хозяйству в общей базе данных признака донора или реципиента является вторым этапом построения базовой основы для восстановления отсутствующих сведений в переписных листах.

В базе данных каждому хозяйству в зависимости от наличия в переписных листах полных ответов или частичных неответов необходимо присвоить соответственно или признак донора, или признак реципиента.

Признак донора присваивается только тем хозяйствам, записи которых:

* удовлетворяют всем правилам редактирования, содержат ответы только респондентов;
* не содержат вмененных записей, за исключением ответов, вмененных на взаимосвязанные вопросы, например, если сезонные работники не привлекались, и их количество не указано, то количество сезонных работников считается равным нулю;
* значения каждого из количественных показателей должны принадлежать некоторому доверительному интервалу. Используется жесткий алгоритм разделения. То есть, если хотя одно поле не удовлетворяет условию однородности , то вся запись маркируется как не донорская.

Наследующем этапе осуществляется создание вспомогательной базы (базы исходных данных), которая содержит значения всех показателей каждого хозяйства-донора для проведения многомерной классификации и импутации методом ближайшего соседа. Данный шаг, который условно можно назвать нормализацией статистических данных, предназначен для ускорения работы алгоритмов многомерной классификации и импутации.

Таким образом, сформированная на этом этапе база хозяйств-доноров является основой для восстановления отсутствующих сведений в переписных листах выделенной базы домашних хозяйств-реципиентов (получателей).

## 2.2 Представление правил редактирования в нормальной форме

Пусть каждая запись, подлежащая редактированию, содержит *N* полей. Пусть *Ai* – множество допустимых значений поля *i* (множество может быть как конечным, так и бесконечным).

Недопустимая комбинация допустимых значений в различных полях записи представляет собой подмножество *F*  *A1*  *A2* ... *AN* кортежей   
<*a1*, *a2*, ..., *aN*>. Любое подмножество *F* может быть представлено (путем преобразований с использованием дистрибутивного закона и операций объединения и пересечения) в следующем виде:

,

где , *k = 1..M.*

Поле *j* *явно входит* в правило *k*, если .

*Нормальной формой* правила редактирования является правило следующего вида:

.

Любое правило редактирования (записанное в форме условия ЕСЛИ-ТО или еще каким-либо образом) может быть представлено в виде набора правил, находящихся в нормальной форме.

Множество правил, находящихся в нормальной форме, которые задаются экспертами предметной области, называются *явными правилами редактирования* (в отличие от неявных правил, которые будут рассмотрены позже). Все явные правила можно разделить на две группы:

* *простые* правила редактирования, задающие допустимые значения определенного поля записи. Данные правила могут быть приведены в нормальную форму автоматически: для любого поля множество *Ai* может быть расширено некоторым элементом i , который представляет *все* недопустимые значения поля с номером *i*. Таким образом, множество простых правил в нормальной форме могут быть представлены в следующем виде:   
  *ei* : {i}, *i = 1..N.*
* *сложные* правила редактирования вида «*Если в записи присутствует некоторая комбинация значений в одних полях, то из этого должна следовать некоторая комбинация значений в других полях*». Данные правила также должны быть преобразованы в нормальную форму правил редактирования.

Правила редактирования разделяются на два вида в зависимости от типов полей, явно входящих в правило:

* правила редактирования *качественных* показателей;
* правила редактирования *количественных* показателей.

К качественным показателям относятся показатели, которые могут иметь конечный набор значений или кодов. Для некоторых правил редактирования количественное поле может быть преобразовано к качественному представлению для упрощения процедур импутации.

### 2.2.1. Представление правил редактирования качественных показателей в нормальной форме

Правила редактирования качественных показателей в нормальной форме могут быть представлены в виде битовых строк, содержащих только символы 0 и 1.

Рассмотрим пример представления правил редактирования качественных показателей на примере анкеты, состоящей из двух полей:

* **Поле 1**: «Производили Вы в Вашем хозяйстве сельскохозяйственную продукцию», домен показателя – «Ответ Да-Нет»;
* **Поле 2:** «Укажите основную цель производства сельскохозяйственной продукции», домен показателя – «Цель производства продукции».

Для данной анкеты можно предложить четыре правила, которые полностью зададут все комбинации значений полей, при которых запись будет некорректна. Введем обозначения: Поле 1 – *a*1, Поле 2 – *a*2.

Правила имеют следующий вид:

* + *e*1 :  *a*1содержит значение, отличное от *1* или *2*;
  + *e*2 :  *a*1*=1* *и a*2*=0*;
  + *e*3 :  *a*1*=2* *и (a*2*=1 или a*2*=2 или a*2*=4)*;
  + *e*4 :  *a*2содержит значение, отличное от *0, 1, 2* или *4*;

Представим правила *e*1, *e*2, *e*3 и *e*4 в КНФ.

* *e*1 : *a*1=1и *a*2=*любое значение*;
* *e*2 : *a*1=*1* и *a*2=*0*;
* *e*3 : *a*1=*2* и (*a*2=1 или *a*2=2 или *a*2=4);
* *e*1 : *a*1= *любое значение* и *a*2=2

Пример представления правил редактирования приведен в виде булевой матрицы (табл. 3.9). Первый столбец таблицы содержит названия показателей, второй – возможные значения (коды) для всех показателей. Каждый из остальных столбцов содержит представление одного из правил редактирования.

В данном примере каждое правило полностью определяется двумя множествами кодов полей (множество кодов полей показателя 1 и показателя 2), которые задают недопустимые комбинации значений полей. Если в правиле присутствует код определенного поля, то в соответствующую ячейку таблицы ставится 1, в противном случае – 0.

Правила *e1* и *e4* представляют собой «простые» правила редактирования, т.е. правила редактирования для отдельных полей записи.

Сложные правила редактирования это правила вида «*Если в записи присутствует некоторая комбинация значений в одних полях, то из этого должна следовать некоторая комбинация значений в других полях*». Данные правила должны быть преобразованы в нормальную форму правил редактирования.

Таблица 2.5  
Представление правил редактирования качественных показателей

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Наименование культуры** | Коды полей | Правила | | | |
| **e1** | **e2** | **e3** | **e4** |
| Производили Вы в Вашем хозяйстве сельскохозяйственную продукцию | **1-да** | 0 | 1 | 0 | 1 |
| **2-нет** | 0 | 0 | 1 | 1 |
| **другое** | 1 | 0 | 0 | 1 |
| Укажите основную цель производства сельскохозяйственной продукции | **0** | 1 | 1 | 0 | 0 |
| **1** | 1 | 0 | 1 | 0 |
| **2** | 1 | 0 | 1 | 0 |
| **4** | 1 | 0 | 1 | 0 |
| **другое** | 1 | 0 | 0 | 1 |

### 2.2.2. Представление правил редактирования количественных показателей в нормальной форме

Правило редактирования количественных показателей может быть записано в виде *f(a1, ..., aN)* ≥ 0, где *f* – линейная функция от *a1, ..., aN*, а неравенство может быть строгим.

Любая запись, для которой *f* ≥ 0 не удовлетворяет данному правилу редактирования. При помощи данной формы записи равенства и интервальные ограничения могут быть представлены в виде двух неравенств – линейных правил редактирования.

Так как ограничения являются линейными, то любое правило полностью определяется *N + 1* коэффициентом и флагом типа неравенства (является ли неравенство строгим или нет), и может быть представлено в следующем виде.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Правило** | **Постоянный коэффициент** | **Поле 1** | **Поле 2** | **.........** | **Поле N** | **Флаг** |
| *er* | *ar0* | *ar1* | *ar2* |  | *arN* | * r* |

Если поле не входит в правило, то соответствующий коэффициент равен нулю. В целях нормализации первый ненулевой коэффициент должен быть приведен к значению ±1. * r =* 1в том случае, если неравенство строгое, и * r =* 0 – в противном случае.

Далее приведены некоторые правила редактирования количественных показателей для формы 3ЛПХ.

Таблица 2.6.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Формулировка вопроса** | **Код** | **e1** | **e2.1** | **e2.2** | **e3** | **e4** | **e5.1** | **e5.2** |
|  | **∂** | **0** | **1** | **1** | **0** | **0** | **1** | **1** |
| Укажите общее число лиц, совместно проживающих с Вами | 1\_6 | -1 |  |  |  |  |  |  |
| Укажите число членов семьи, занятых в Вашем хозяйстве | 1\_11 | 1 |  |  |  |  |  |  |
| Приусадебный участок, общая площадь, га | 2\_2 |  | 1 | -1 | -1 |  |  |  |
| Полевые земельные участки, общая площадь, га | 2\_5 |  | 1 | -1 | 1 |  |  |  |
| в собственности | 2\_7 |  | -1 | 1 |  |  |  |  |
| в пожизненном наследуемом владении | 2\_9 |  | -1 | 1 |  |  |  |  |
| в постоянном пользовании | 2\_11 |  | -1 | 1 |  |  |  |  |
| в безвозмездном срочном пользовании | 2\_13 |  | -1 | 1 |  |  |  |  |
| Общая площадь земли, находящаяся в собственности, га | 2\_17 |  |  |  | 1 | -1 | -1 | 1 |
| Сданная в аренду, га | 2\_19 |  |  |  |  | 1 | 1 | -1 |
| Переданная другим лицам, га | 2\_21 |  |  |  |  | 1 | -1 | 1 |
| Площадь земли, арендованная у других лиц, га | 3\_2 |  |  |  |  |  | -1 | 1 |
| Общая площадь земли, используемая хозяйством, га | 3\_4 |  |  |  |  |  | 1 | -1 |

## 2.3 Алгоритм получения полного множества правил редактирования

Запись, которая удовлетворяет всем правилам редактирования, называется «чистой» и не нуждается в какой-либо коррекции. Запись, которая не удовлетворяет хотя бы одному правилу редактирования, нуждается в некоторой коррекции. Несмотря на то, что нам известны те правила, которым не удовлетворяет запись, неизвестно, значения каких именно полей в данной записи приводят к тому, что запись некорректна.

Если явные правила являются необходимым и достаточным условием для определения того, является ли запись некорректной, то неявные правила, являющиеся логическим следствием явных правил, также должны быть рассмотрены для определения того, какие поля необходимо изменять для коррекции записи.

Если имеется *M* правил, то можно получить неявные правила как следствие из произвольных *двух, трех, ..., M* правил.

При получении неявных правил может возникнуть ситуация, когда новое правило либо уже известно, либо является следствие одного из известных правил. Такие правила не являются существенно-новыми правилами.

### 2.3.1 Полное множество правил редактирования качественных показателей

Процедура получения полного множества правил редактирования качественных показателей основывается на булевой матрицы редактирования. Рассмотрим получение неявного правила на основе двух явных.

В неявном правиле для каждого значения поля в ячейку таблицы записывается:

* логическое произведение значений полей исходных правил, для любого поля, которое не являющимся генерирующим;
* логическая сумма значений полей исходных правил для генерирующего поля.

Полученное правило не является существенно новым правилом, если:

* одно из полей содержит все нули;
* генерирующее поле не содержит все единицы;
* полученное правило является частью уже существующего (существует правило, содержащее единицы во всех строках, что и полученное).

Для получения всех неявных правил можно воспользоваться алгоритмом полного перебора, комбинируя по два, три и т.д. правила, используя каждое поле в качестве генерирующего.

Таблица 2.7. Получение неявных правил редактирования.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Формулировка вопроса** | **Коды полей** |  | | **Неявное правило** |
| **e6** | **e7** |
| Производили Вы в Вашем хозяйстве сельскохозяйственную продукцию | **1-да** | 1 | 1 | 1 |
| **2-нет** | 1 | 1 | 1 |
| Укажите основную цель производства сельскохозяйственной продукции | **0** | 1 | 1 | 1 |
| **1** | 1 | 1 | 1 |
| **2** | 1 | 1 | 1 |
| **4** | 1 | 1 | 1 |
| Получили Вы земельные доли при реорганизации | **1-да** | 1 | 1 | 1 |
| **2-нет** | 0 | 1 | 0 |
| Укажите суммарную площадь полученных земельных долей, га | **0** | 1 | 0 | 1 |
| **>0** | 0 | 1 | 1 |
| Земельные доли используются в хозяйстве | **0** | 1 | 1 | 1 |
| **>0** | 1 | 0 | 0 |
| Земельные доли сданы в аренду | **0** | 1 | 1 | 1 |
| **>0** | 1 | 0 | 0 |
| Земельные доли внесены в уставный капитал | **0** | 1 | 1 | 1 |
| **>0** | 1 | 0 | 0 |
| Земельные доли проданы | **0** | 1 | 1 | 1 |
| **>0** | 1 | 0 | 0 |

### 2.3.2 Полное множество правил редактирования количественных показателей

Процедура получения полного множества арифметических правил основывается на следующем утверждении: существенно-новое неявное правило *et*, получаемое из правил *er* и *es* используя поле *i* как генерирующее может быть рассчитано по формуле

*atk = ask ari - ark asi* , k=0,1, ... , N

при условии, что коэффициенты связаны соотношением *ari  asi < 0,* индексы *r* и *s* выбираются таким образом, что *ari >0* и *asi < 0,* а * t =  t  s.* Новое полученное правило также должно быть нормализовано приведением первого ненулевого коэффициента к значениям ±1.

## 2.4 Алгоритм определения минимального множества полей для импутации данных

Следующим этапом после получения полного множества правил редактирования является этап обработки записей. Если запись не удовлетворяет каким-либо правилам редактирования, то требуется определение минимального множества полей записи, которые будут подвержены процедуре импутации.

Рассмотрим матрицу размерности *R**N*,

где *R* – количество не удовлетворенных правил,

*N* – количество полей в записи.

Ячейка (*r, n*) данной матрицы содержит 1, если поле с номером *n* явно содержится в правиле *r*. В противном случае ячейка содержит 0. Данная матрица, называемая матрицей неудовлетворенных правил, генерируется автоматически.

Задача определения минимального множества полей для импутации для удовлетворения всех правил сводится к задаче поиска минимального набора полей, у которых в каждом столбце матрицы неудовлетворенных правил будет присутствовать по крайней мере одна единица.

Таблица 2.8 Пример матрицы неудовлетворенных правил

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Формулировка вопроса** | **Правила** | | | | | | |
| **e1** | **e2** | **e3** | **e4** | **e5** | **e6** | **e7** |
| Производили Вы в Вашем хозяйстве сельскохозяйственную продукцию | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| Укажите основную цель производства сельскохозяйственной продукции | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| Получили Вы земельные доли при реорганизации | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| Укажите суммарную площадь полученных земельных долей, га | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| Земельные доли используются в хозяйстве | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Земельные доли сданы в аренду | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Земельные доли внесены в уставный капитал | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Земельные доли проданы | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

В самом простом случае мы можем проверять каждое поле, каждую пару полей, тройку и т.д., входят ли они в каждое правило редактирования. При этом:

* все удовлетворенные правила могут не рассматриваться;
* в искомое множество полей обязательно включаются поля, которые не удовлетворяют правилам редактирования одного поля (проверки поля на допустимые значения, недопустимые неответы в поле и т.д.).

Для корректного и детерминированного процесса локализации ошибки каждому показателю присваивается весовой коэффициент, определяющий предпочтения при выборе данного показателя для проведения импутации, если выбор равнозначен. Данный весовой коэффициент должен определять «количество информации», обозначим *In*, которое содержится в значении показателя *n*, причем чем больше значение веса показателя, тем информации меньше.

Рассмотрим пример задания весовых коэффициентов для матрицы неудовлетворенных правил из табл. 3.16.

Таблица 3. 16.  
Весовые коэффициенты для матрицы неудовлетворенных правил

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№** | Формулировка вопроса | Правила | | | Вес |
| e3 | e5 | e7 |  |
| 1 | Производили Вы в Вашем хозяйстве сельскохозяйственную продукцию | **1** | **0** | **0** | 2 |
| 2 | Укажите основную цель производства сельскохозяйственной продукции | **1** | **0** | **0** | 1 |
| 3 | Получили Вы земельные доли при реорганизации | **0** | **1** | **1** | 2 |
| 4 | Укажите суммарную площадь полученных земельных долей, га | **0** | **1** | **0** | 0.5 |
| 5 | Земельные доли используются в хозяйстве | **0** | **0** | **1** | 1 |
| 6 | Земельные доли сданы в аренду | **0** | **0** | **1** | 1 |
| 7 | Земельные доли внесены в уставный капитал | **0** | **0** | **1** | 1 |
| 8 | Земельные доли проданы | **0** | **0** | **1** | 1 |

Пусть правила *e2, e5, e7* неудовлетворенны. Тогда (без учета весовых коэффициентов) мы должны выбирать между следующими альтернативами для импутации показателей:

* импутировать показатели №1, №3;
* импутировать показатели №2, №3.

Весовые коэффициенты выбирались из следующих соображений:

* *I1 < I2 ;*
* *I3 < I5 = I6 = I7 = I8 <= I4;*

Используя весовые коэффициенты предлагается следующий алгоритм выбора множества показателей для импутации.

* + 1. Во множество показателей, предназначенных для импутации добавляются показатели, для которых не удовлетворены «простые» правила редактирования. Данные показатели исключается из дальнейшего рассмотрения. Правила, в которые эти показатели входят явно, исключаются из рассмотрения.
    2. Для каждого показателя рассчитывается его характеристика как количество правил, в которые показатель входит явно, умноженное на вес показателя.
    3. Показатель с максимальной характеристикой добавляется во множество показателей, предназначенных для импутации. Если существует несколько показателей с максимальной характеристикой, то выбирается последний.
    4. Показатель, полученный в п. 2 исключается из дальнейшего рассмотрения.
    5. Правила, в которые показатель из п. 2 входит явно, исключается из рассмотрения.
    6. Если множество рассматриваемых правил не пустое, то возвращаемся к п. 1.

Для табл. 3.16. максимальную характеристику (4) будет иметь показатель №3, после чего он исключается из рассмотрения, правила e5, e7 исключаются из рассмотрения. На следующем шаге будет выбран показатель №1 с характеристикой равной 2, после чего алгоритм закончит работу, так как множество правил будет пустым.

Тогда с учетом весовых коэффициентов автоматически будет принято решение импутировать показатели №1, №3.

## 2.5 Алгоритм импутации данных

Этап импутация данных должен проходить в два шага:

* импутация количественных показателей;
* импутация качественных показателей.

### 2.5.1 Импутация количественных показателей

Рассмотрим алгоритм импутации количественных показателей на примере анкеты, содержащей пять показателей *a*1, *a2*, *a3*, *a4*, *a5*, связанных следующими соотношениями:

* *a*1 = *a*2 + *a*3;
* *a*3 = *a*4 + *a*5.

На основании этих соотношений могут быть получены следующие *явные* правила редактирования:

* *e*1 : *a*1 - *a*2 - *a*3 > 0;
* *e*2 : - *a*1 + *a*2 + *a*3 > 0;
* *e*3 : *a*3 - *a*4 - *a*5 > 0;
* *e*4 : - *a*3 + *a*4 + *a*5 >0.

Полное множество правил редактирования будет включать в себя еще два правила как следствие явных (обоих случаях в качестве генерирующего поле используется поле *a*3) и «простые правила» для показателей:

* *e*5 : *a*1 - *a*2 - *a*4 - *a*5 > 0;
* *e*6 : -*a*1 + *a*2 + *a*4 + *a*5 > 0;
* *e*7: *a*1 = 1 ;
* *e*8: *a*2 = 2 ;
* *e*9: *a*3 = 3 ;
* *e*10: *a*4 = 4;
* *e*11: *a*5 = 5 .

На основании анализа соотношений, связывающих показатели, можно задать следующие веса:

|  |  |  |
| --- | --- | --- |
| **№** | **Показатель** | **Вес** |
| **1** | *a*1 | 1 |
| **2** | *a*2 | 3 |
| **3** | *a*3 | 2 |
| **4** | *a*4 | 3 |
| **5** | *a*5 | 3 |

Пусть запись, требующая редактирования, имеет следующие значения показателей:

|  |  |  |
| --- | --- | --- |
| **№** | **Показатель** | **Значение** |
| **1** | *a*1 | 10 |
| **2** | *a*2 | 2 |
| **3** | *a*3 | недопустимое значение |
| **4** | *a*4 | 2 |
| **5** | *a*5 | 2 |

Построим матрицу неудовлетворенных правил (все пять правил из полного множества правил редактирования неудовлетворенны):

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№** | **Показатель** | **Правила** | | | | | | | **Вес** |
| **e1** | **e2** | **e3** | **e4** | **e5** | **e6** | **e7** | ***w(ei)*** |
| **1** | *a*1 | **1** | **1** | **0** | **0** | **1** | **1** | **0** | 1 |
| **2** | *a*2 | **1** | **1** | **0** | **0** | **1** | **1** | **0** | 3 |
| **3** | *a*3 | **1** | **1** | **1** | **1** | **0** | **0** | **1** | 2 |
| **4** | *a*4 | **0** | **0** | **1** | **1** | **1** | **1** | **0** | 3 |
| **5** | *a*5 | **0** | **0** | **1** | **1** | **1** | **1** | **0** | 3 |

На основании алгоритма локализации ошибки в множество столбцов для импутации последовательно будет добавлены показатели: *a*3 , *a*5.

Рассчитываем веса правил как сумму величин, обратных весам показателей, явно входящих в правило и требующих импутации:

* *w*(*p*1) = 1/*w*(*a*3) = 1/2;
* *w*(*p*2) = 1/*w*(*a*3) = 1/2;
* *w*(*p*3) = 1/*w*(*a*3) + 1/*w*(*e*5)= 1/2 + 1/3 = 5/6;
* *w*(*p*4) = 1/*w*(*a*3) + 1/*w*(*e*5)= 1/2 + 1/3 = 5/6;
* *w*(*p*5) = 1/*w*(*a*5) = 1/3;
* *w*(*p*6) = 1/*w*(*a*5) = 1/3;
* *w*(*p*7) = 1/*w*(*a*3) = 1/2.

Среди правил с минимальным количеством показателей, требующих импутации, ищется правило с минимальным весом.

В рассматриваемом случае правила *e*1, *e*5, *e*6, *e*7 содержат по одному показателю, требующему импутации. Минимальным весом обладают правила *e*5, *e*6. Выбираем правило *e*5.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№** |  | Правила | | | | | | |
| **e1** | **e2** | **e3** | **e4** | **e5** | **e6** | **e7** |
| 1 | *Вес правила, w(pi)* | **1/2** | **1/2** | **5/6** | **5/6** | **1/3** | **1/3** | **1/2** |

Далее проверяем, существует ли *симметричное* для *e*5 правило. Правило *e*i является *симметричным* для правила *e*j , если коэффициенты этих правил равны по модулю, но противоположны по знаку.

Для правила *e*5 симметричным является правило *e*6.

Тогда процесс импутации будет выглядеть следующим образом.

1. Если для выбранного правила не существует симметричного, то ищем ближайшего донора, используя метод ближайшего соседа;
2. Если для правила существует симметричное, то:
   * если правило содержит единственный показатель, требующий импутации, то он рассчитывается без использования метода ближайшего соседа;
   * если правило содержит несколько показателей, требующих импутации, то используются алгоритмы импутации методом уравнений баланса.
3. Если существуют неудовлетворенные правила, то переходим к пункту 1.

В соответствии с описанным алгоритмом на первом шаге показателю *a*5 будет импутировано значение *a*5 = *a*1 – *a*2 – *a*4 = 6.

|  |  |  |
| --- | --- | --- |
| **№** | **Показатель** | **Значение** |
| 1 | *a*1 | 10 |
| 2 | *a*2 | 2 |
| 3 | *a*3 | недопустимое значение |
| 4 | *a*4 | 2 |
| 5 | *a*5 | 6 |

Рассчитываем веса правил с учетом импутированого значения показателя *a*5 :

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№** |  | Правила | | | | | | |
| **e1** | **e2** | **e3** | **e4** | **e5** | **e6** | **e7** |
| 1 | *Вес правила, w(pi)* | **1/2** | **1/2** | **1/2** | **1/2** | **0** | **0** | **1/2** |

На данном шаге правила *e*5  и *e*6 удовлетворены, а оставшиеся правила содержат только один показатель, требующий импутации - *a*3. Правило *e*1 и *e*2 являются симметричными, правило *e*3 и *e*4 являются симметричными. Выбирая любую (например, первую) пару правил восстанавливаем значение показателя *a*3 = *a*1 - *a*2 = 8.

### 2.5.2 Импутация качественных показателей

Пусть поля записи с 1 по K должны быть импутированы. Рассмотрим множество правил редактирования *M*, которые потенциально могут быть не удовлетворены при импутации полей с 1 по K.

Множества  задают ограничение на значения полей тех записей, которые могут быть использованы для импутации данной записи методом ближайшего соседа. Данные множества задают «диапазон» изменения для каждого поля, которое не будет импутировано, позволяя отбирать те записи, которые могут использоваться как донорские. Следовательно, нет необходимости в вычислении всех возможных значений полей, которые могут быть импутированы в текущую запись.

Таким образом, процедура импутации будет состоять из следующих шагов.

1. Из множества правил редактирования выбирается множество правил *M*, которые могут оказаться не удовлетворенными при импутации выбранных полей.
2. Рассматриваются поля, которые не будут участвовать в импутации. Данные поля будут задавать ограничения на записи, которые будут участвовать в качестве донорских. Ограничения формируются по описанному выше правилу.
3. Осуществляется импутация значений в поля записи. Сначала производится поиск донорской записи, которая может быть использована для импутации значений. Затем осуществляется импутация по алгоритму ближайшего соседа.

# 3 Проектирование программной системы апробации метода Фелледжи-Хольта

## 3.1 Формулировка требований к разрабатываемой системе

### 3.1.1. Основные требования к системе

Разрабатываемая система автоматизированной импутации (САИ) статистических данных сельскохозяйственной на основе алгоритма Филледжи-Хольта должна отвечать следующим основным требованиям:

* обеспечивать автоматизацию технологического процесса автокоррекции и импутации;
* функционировать в составе локальной вычислительной сети (ЛВС).
* рабочая станция должна подключаться и функционировать в локальной сети через TCP/IP протокол;
* рабочее место аналитика должно функционировать под управлением операционной системы *Windows*.

### 3.1.2. Общие требования к программному обеспечению

Основные требования к программному обеспечению следующие:

* операционная система для всех рабочих мест – Microsoft Windows;
* система управления базами данных - Microsoft SQL 2012.

### 3.1.3. Требования к функциональным возможностям

САИ должна выполнять следующие функции:

* Ф1 – автокоррекция данных;
* Ф2 – импутация пропущеных и некорректных данных;
* Ф3 – получение отчетов о результатах автокоррекции и импутации;
* Ф4 – ведение протоколов автокоррекции и импутации;

#### 4.1.3.1. Автокоррекция данных

Автокоррекция данных предназначена для проверки записей на наличие некорректных значений показателей (или недопустимой комбинации корректных значений показателей) и выделения подмножества показателей, требующих импутации с целью удовлетворения всем правилам формального и логического контроля. Для каждой некорректной записи должна быть проведена локализации ошибки, которая определяет множество показателей, которые должны быть импутированы.

Результаты данного этапа должны быть внесены в протокол автокоррекции.

#### 4.1.3.2. Импутация пропущенных и некорректных данных

Этап импутации данных производит восстановление отсутствующих и некорректных данных таким образом, что:

* все правила редактирования удовлетворяются;
* восстановление данных происходит путем изменения минимального множества показателей;
* должны сохраняться статистические распределения значений показателей.

Этап импутации данных должен проходить в два шага: импутация количественных показателей и импутация качественных показателей. Результаты импутации помещаются в архив протоколов импутации.

#### 4.1.3.4. Получение отчетов о результатах автокоррекции и импутации

Получение отчетов должно обеспечиваться пунктами меню, предлагаемыми пользователю, при необходимости сопровождающегося заданием параметров построения отчета.

САИ должны быть способна формировать отчетную документацию в разных форматах, и в частности, в форматах MS Word и MS Excel. В отчеты может включаться:

1. представление исходных и донорских данных;
2. результаты работы алгоритмов многомерной классификации;
3. содержимое протоколов автокоррекции и импутации с подробными пояснениями

#### 4.1.3.3. Ведение протоколов автокоррекции и импутации

Протоколы автокоррекции и импутации предназначены для хранения результатов проверки данных, а также протоколов решения задач импутации. Должно быть предусмотрено два архива: архив протоколов автокоррекции и архив протоколов импутации. Протоколы должны содержать полную информацию о выполнении этапов автокоррекции и импутации:

* исходные данные, к которым применяются автокоррекция и импутация;
* принятие решения о выборе того или иного метода восстановления данных;
* выбранные модели и алгоритмы для проведения восстановления данных, их параметры;
* результаты выполнения алгоритмов автокррекции и импутации.

### 3.1.6. Требования к информационному обеспечению

Информационное обеспечение САИ составляют классификаторы и справочники, используемые для их обработки.

Состав классификаторов и справочников, используемых в САИ:

* ОКАТО - общероссийский классификатор административно-территориального деления;
* ОКЕИ - общероссийский классификатор единиц, измерения.

### 3.1.7. Требования к эргономике и технической эстетике

Интерфейс ПО пользователя должен быть прост в применении и ориентирован на неквалифицированный персонал. Интерфейс ПО пользователя должен обеспечивать выполнение всех функций системы.

## 3.2 Средства разработки программной системы

Средой разработки программного обеспечения является *Visual Studio Community*, позволяющая создавать приложения, функционирующие на платформе *Microsoft .NET Framework*.

Благодаря возможностям среды разработки Microsoft Visual Studio .NET и спецификации Microsoft .NET Framework разработчикам предоставляются самые эффективные средства для быстрого создания и развертывания современных приложений и веб-служб XML.

1. Платформа Microsoft .NET, благодаря средствам Visual Studio .NET и .NET Framework, позволит обеспечить более быструю разработку программных приложений.
2. С помощью средств .NET Framework и Visual Studio .NET обеспечивается более высокая надежность приложений.
3. Использование веб-служб XML позволит упростить и повысить эффективность интеграции приложений и служб, создаваемых на основе платформы .NET.

Платформа .NET включает также возможности доступа к базам данных, позволяя разработчикам включать в архитектуру создаваемых приложений ODBC-совместимые хранилища данных. Проектируемая система реализована на языке C#, который является новым объектно-ориентированным языком, разработанным компанией Microsoft для платформы .NET.

Современные языки программирования, такие как С#, предоставляют в распоряжение программиста обширный арсенал инструментальных средств, позволяющий проектировать мощные и гибкие приложения.

Все выбранные средства реализации произведены одной фирмой – Microsoft, и этим также можно обосновать данный выбор. В связи с тем, что фирма-производитель средств реализации одна и та же, взаимная интеграция отдельных частей системы будет проходить более устойчиво и надежно. Также необходимо помнить о том, что проектируемая система будет эксплуатироваться под управлением операционных систем MS Windows.

## 3.3 Архитектура автоматизированной системы автокоррекции и импутации данных

Исходя из целей преследуемых реализацией автоматизированной системы автокоррекции и импутации данных и требований, сформулированных в предыдущем разделе, предлагается архитектура, включающая следующие элементы.

1. Система управления исходными данными:
   * архив статистических данных;
   * архив исходных данных;
   * система выделения донорских данных;
   * система формирования исходных и донорских данных из статистических данных.
2. Система управления процессом импутации:

* система формально-логического контроля;
* система автокоррекции данных;
* система локализации ошибки;
* система импутации данных.

1. Система управления архивами протоколов автокоррекции и импутации:
   * архив протоколов автокоррекции;
   * архив протоколов импутации
   * система архивирования протоколов импутации.
2. Экспертная система анализа результатов алгоритмов автокоррекции и импутации:

* база знаний;
* машина извлечения знаний;
* система логического вывода;
* машина объяснений.

1. Система управления отчетной документацией:

* система формирования отчетов;
* отчеты, сформированные в различных форматах представления данных.

Графически структура разрабатываемой системы представлена на рис. 3.1.

**Система управления исходными данными** предназначена для обеспечения ввода, хранения статистических материалов, а также для формирования на их основе исходных данных, формирование донорских пулов, предназначенных непосредственно для использования в процессе работы алгоритмов импутации в методе ближайшего соседа.

**Система управления архивами протоколов автокоррекции и импутации** предназначена для хранения в архивах результатов проверки данных, а также протоколов решения задач импутации.

**Система управления процессом импутации** предназначена для решения задач определения некорректных записей, локализации ошибки (определения минимального набора показателей, требующих импутации), а также непосредственно для импутации. Данная система на входе получает исходные данные, требующие импутации и донорские данные, используемые в методах ближайшего соседа.

Система формально-логического контроля предназначена для проверки записей на наличие некорректных значений показателей (или недопустимой комбинации корректных значений показателей). Данная система использует банк алгоритмов линейной алгебры для количественных показателей и банк алгоритмов алгебры логики для качественных и смешанных показателей.

Система автокоррекции данных на основе правил редактирования формирует протокол автокоррекции и предает данные системе локализации ошибки.

Система локализации ошибки на основе данных системы автокоррекции определяет множество полей, которые должны быть импутированы. Выбор показателей для импутации может быть осуществлен как в автоматическом режиме (с использованием весовых коэффициентов показателей) так и в полуавтоматическом (при помощи решений экспертов).

Система импутации данных на основе *всех банков алгоритмов* производит восстановление отсутствующих и некорректных данных. Результаты в итоге помещаются в архив протоколов импутации.

В результате совместного функционирования данных систем данные передаются системе управления отчетной документацией.

**Система управления отчетной документацией** включает в себя систему формирования отчетов на основе данных, полученных на выходе, системы управления процессом импутации. Данная система способна формировать отчетную документацию в разных форматах, и в частности, в форматах MS Word и MS Excel. В отчеты может включаться:

1. представление исходных и донорских данных;
2. результаты работы алгоритмов многомерной классификации;
3. содержимое протоколов автокоррекции и импутации с подробными пояснениями.

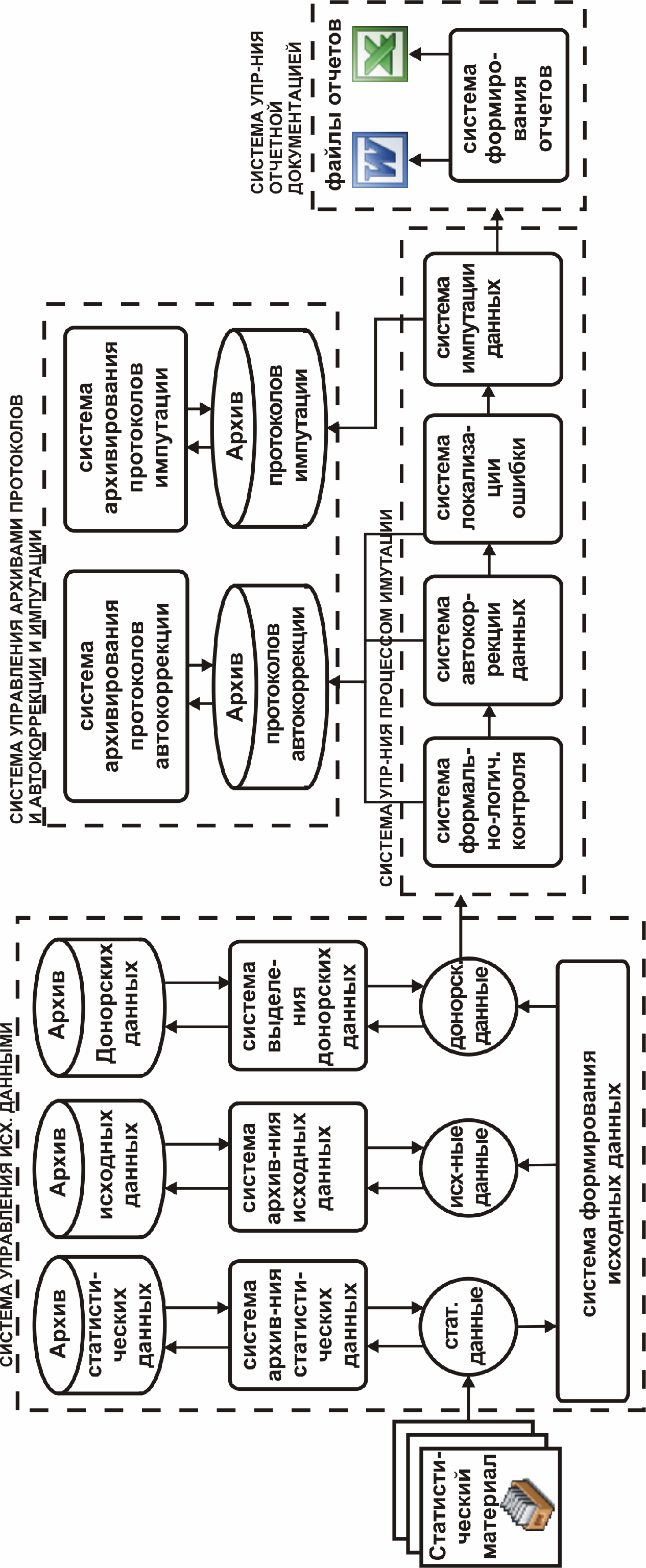


Рис. 3.1 – Архитектура системы

## 3.4 Информационная модель программной системы

Перед созданием базы данных необходимо располагать описанием выбранной предметной области, которое должно охватывать реальные объекты и процессы, определить все необходимые источники информации для удовлетворения предполагаемых запросов пользователя и определить потребности в обработке данных.

### 3.4.1. Спецификация сущностей

На основе проведённого анализа предметной области выделим её основные объекты и сформируем соответствующие им сущности.

Спецификация полученных сущностей и атрибутов представлена в таблице 3.1.

Таблица 3.1 – Описание сущностей

| **№ п/п** | **Название сущности** | **Определение сущности** |
| --- | --- | --- |
| 1 | Регионы | Справочник для хранения информации о регионах |
| 2 | Формы | Хранение информации об анкетных формах |
| 3 | Таблицы | Хранение информации о таблицах, которые используются для хранения данных в анкетных формах |
| 4 | Столбцы | Хранение информации о столбцах таблиц |
| 5 | Домены | Справочник доменов – определенного набора допустимых значений |
| 6 | ЗначенияДоменов | Справочник допустимых значений домена |
| 7 | Ответы | Хранение информации о заполненных анкетах |
| 8 | Разбиение на кластеры | Хранение информации о разбиеняих на кластеры данных какой-либо формы |
| 9 | Кластеры | Хранение информации о кластерах |
| 10 | Правила редактирования | Хранение информации о правилах редактирования |
| 11 | Параметры правила редактирования | Хранение информации о параметрах правилах импутации |
| 12 | Правила импутации | Хранение информации о правилах импутации |
| 13 | Тип правила редактирования | Перечисление типов правил редактирования |
| 14 | Тип правила импутации | Перечисление типов правил импутации |

Рассмотрим полученные сущности более подробно (таблица 3.2).

Таблица 3.2 – Спецификация сущностей

| **№ п/п** | **Название и обозначение сущности** | **Ключ сущности и его обозначение** | **Атрибуты сущности и их обозначение** |
| --- | --- | --- | --- |
| 1 | Регионы  (Рег) | Код региона  (КРег) | Название (Назв)  ОКАТО (ОКАТО) |
| 2 | Формы  (Формы) | Код формы  (КФормы) | Название (Назв)  Примечание (Прим) |
| 3 | Таблицы  (Табл) | Код таблицы  (КТабл) | Название (Назв)  Примечание (Прим)  Имя ключа (ИмяКл)  Имя кода хоз-ва (ИмяКХ)  Тип (Тип) |
| 4 | Столбцы  (Столбцы) | Код столбца  (КСтолбц) | Номер (Ном)  Имя (Имя)  Код (Код)  Название (Назв)  Признак обработки (ПрОбр)  Сокращенное название (Сокр)  КодЭОД (ЭОД) |
| 5 | Домены  (Дом) | Код домена  (КДомен) | Примечание (Прим)  Тип домена (ТДом)  Минимальное значение (МинЗн)  Максимальное значение (МаксЗн) |
| 6 | ЗначенияДоменов  (ЗначДом) | Код значения домена  (КЗначДом) | Значение (Знач)  Название (Назв) |
| 7 | Ответы  (Отв) | Код хозяйства  (КХоз) | Является донором (Дон) |
| 8 | Разбиение на кластеры  (РазбКласт) | -  (*данная сущность является зависимой*) | - |
| 9 | Кластеры  (Класт) | Код кластера  (ККласт) | Номер (Ном)  Название (Назв) |
| 10 | Правила редактирования  (ПрРед) | Код правила редактирования  (КПрРед) | Номер (Ном) |
| 11 | Параметры правила редактирования  (ПарПрРед) | Код параметра правила редактирования  (КПарПрРед) | Коэффициент (Коэфф) |
| 12 | Правила импутации  (ПрИмп) | Код правила импутации  (КПрИмп) | Номер (Ном)  Значение (Знач) |
| 13 | Тип правила редактирования  (ТПрРед) | Код типа правила редактирования  (КТПрРед) | Название (Назв) |
| 14 | Тип правила импутации  (ТПрИмп) | Код типа правила импутации  (КТПрИмп) | Название (Назв) |

Схемы сущностей представляются в традиционной нотации, ключи в схеме выделяются подчеркиванием (таблица 3.3).

Таблица 3.3 – Схемы сущностей

| **№ п/п** | **Название и обозначение сущности** | **Схема сущности** |
| --- | --- | --- |
| 1 | Регионы  (Рег) | Рег(КРег, Назв, ОКАТО) |
| 2 | Формы  (Формы) | Формы(КФормы, Назв, Прим) |
| 3 | Таблицы  (Табл) | Табл(КТабл, Назв, Прим, ИмяКл, ИмяКХ, Тип) |
| 4 | Столбцы  (Столбцы) | Столбцы(КСтолбц, Ном, Имя, Код, Назв, ПрОбр, Сокр, ЭОД) |
| 5 | Домены  (Дом) | Дом(КДомен, Прим, ТДом, МинЗн, МаксЗн) |
| 6 | ЗначенияДоменов  (ЗначДом) | ЗначДом(КЗначДом, Знач, Назв) |
| 7 | Ответы  (Отв) | Отв(КХоз, Дон) |
| 8 | Разбиение на кластеры  (РазбКласт) | РазбКласт() |
| 9 | Кластеры  (Класт) | Класт(ККласт, Ном, Назв) |
| 10 | Правила редактирования  (ПрРед) | ПрРед(КПрРед, Ном) |
| 11 | Параметры правила редактирования  (ПарПрРед) | ПарПрРед(КПарПрРед, Коэфф) |
| 12 | Правила импутации  (ПрИмп) | ПрИмп(КПрИмп, Ном, Знач) |
| 13 | Тип правила редактирования  (ТПрРед) | ТПрРед(КТПрРед, Назв) |
| 14 | Тип правила импутации  (ТПрИмп) | ТПрИмп(КТПрИмп, Назв) |

### 3.4.2. Описание связей

Спецификация связей представляет собой два перечня – список связей и список их идентификаторов. В первом перечисляются сущности и связи между ними. В каждой строке название каждой сущности записывается с новой буквы, а глагол, выражающий связь, полностью записывается большими буквами.

Спецификация получившихся при анализе объектов предметной области связей, их описания, степени связи и классы принадлежности представлены в таблице 3.4.

Таблица 3.4 – Описание связей

| № п/п | Название и обозначение сущностей | Идентификатор | Связь | Степень связи. Класс прнад-лежности |
| --- | --- | --- | --- | --- |
| 1 | Регион (Рег)  Ответы (Отв) | СОДЕРЖИТ | Рег\_СОДЕРЖИТ\_Отв | 1:М  О-НО |
| 2 | Регион (Рег)  Разбиение на кластеры  (РазбКласт) | СОДЕРЖИТ | Рег\_СОДЕРЖИТ\_РазбКласт | 1:М  О-НО(З) |
| 3 | Формы (Формы)  Разбиение на кластеры  (РазбКласт) | СОДЕРЖИТ | Формы\_СОДЕРЖИТ\_РазбКласт | 1:М  О-НО(З) |
| 4 | Формы (Формы)  Таблицы (Табл) | СОСТОИТ\_ИЗ | Формы\_СОСТОИТ\_ИЗ\_Табл | 1:М  О-НО |
| 5 | Формы (Формы)  Ответы (Отв) | СОДЕРЖИТ | Формы\_СОДЕРЖИТ\_Отв | 1:М  О-НО |
| 6 | Формы (Формы)  Правила редактирования  (ПрРед) | ИМЕЕТ | Формы\_ИМЕЕТ\_ПрРед | 1:М  О-НО |
| 7 | Разбиение на кластеры  (РазбКласт)  Кластеры (Класт) | СОДЕРЖИТ | РазбКласт \_СОДЕРЖИТ\_Класт | 1:М  О-НО |
| 8 | Разбиение на кластеры  (РазбКласт)  Столбцы (Столбцы) | СОДЕРЖИТ | РазбКласт \_СОДЕРЖИТ\_Столбцы | М:М  НО-НО |
| 9 | Таблицы (Табл)  Столбцы (Столбцы) | СОДЕРЖИТ | Табл \_СОДЕРЖИТ\_Столбцы | 1:М  О-НО |
| 10 | Ответ (Отв)  Кластер (Класт) | ОТНОСИТСЯ\_К | Отв\_ОТНОСИТСЯ\_К\_Класт | М:М  НО-НО |
| 11 | Кластер (Класт)  Столбцы (Столбцы) | ИМЕЕТ\_ЗНАЧЕНИЕ\_ПАРАМЕТРА | Класт\_ИМЕЕТ\_ЗНАЧЕНИЕ\_ПАРАМЕТРА\_Столбцы | М:М  НО-НО |
| 12 | Столбцы (Столбцы)  Правила импутации (ПрИмп) | ИМЕЕТ | Столбцы\_ИММЕТ\_ПрИмп | 1:М  О-НО |
| 13 | Правила импутации (ПрИмп)  Столбцы (Столбцы) | СВЯЗАНО\_С | ПрИмп\_СВЯЗАНО\_С\_Столбцы | М:М  НО-НО |
| 14 | Столбцы (Столбцы)  Домен (Дом) | ПРИНАДЛЕЖИТ | Столбцы\_ПРИНАДЛЕЖИТ\_Дом | М:1  НО-О |
| 15 | Параметр правила редактирования  (ПарПрРед)  Столбцы (Столбцы) | ОТНОСИТСЯ\_К | ПарПрРед\_ОТНОСИТСЯ\_К\_Столбцы | М:1  НО-О |
| 16 | Домен (Дом)  Значения домена (ЗначДом) | СОДЕРЖИТ | Дом\_СОДЕРЖИТ\_ЗначДом | 1:М  О-НО |
| 17 | Параметр правила редактирования  (ПарПрРед)  Значение домена (ЗначДом) | ОТНОСИТСЯ\_К | ПарПрРед\_ОТНОСИТСЯ\_К\_ЗначДом | М:1  НО-О |
| 18 | Правило редактирования (ПрРед)  Тип правила редактирования  (ТПрРед) | ИМЕЕТ | ПрРед\_ИМЕЕТ\_ТПрРед | М:1  НО-О |
| 19 | Правило импутации  (ПрИмп)  Тип правила импутации  (ТПрРед) | ИМЕЕТ | ПрИмп\_ИМЕЕТ\_ТПрИмп | М:1  НО-О |

### 3.4.3. Диаграмма сущность-связь

Семантическое моделирование представляет собой моделирование структуры данных, опираясь на смысл этих данных. В качестве инструмента семантического моделирования используются различные варианты ***диаграмм сущность-связь*** (***ER - Entity-Relationship***).

ER-диаграммы удобны тем, что процесс выделения сущностей, атрибутов и связей является итерационным. Разработав первый приближенный вариант диаграмм, мы уточняем их, опрашивая экспертов предметной области. При этом документацией, в которой фиксируются результаты бесед, являются сами ER-диаграммы. Диаграммы сущность-связь позволяют использовать наглядные графические обозначения для моделирования сущностей и их взаимосвязей.

При правильном определении сущностей, полученные таблицы будут сразу находиться в 3НФ (третья нормальная форма). Основное достоинство метода состоит в том, модель строится методом последовательных уточнений первоначальных диаграмм.

На основании проведённого анализа выбранной предметной области, полученных описаний сущностей, их атрибутов и связей между ними построим диаграмму для наглядного отображения взаимодействия между её объектами (рис. 3.2).



Рис. 3.2 – Диаграмма концептуальной модели базы данных

### 3.4.4. Формализация зависимостей

Определим первичные, внешние и потенциальные ключи отношений. Полученные результаты представим в виде таблицы (таблица 3.5).

Таблица 3.5 – Ключи отношений

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№ п/п** | **Отношение** | **Первичный**  **ключ** | **Внешний**  **ключ** | **Потенциаль-ный**  **ключ** |
| 1 | Рег | КРег | - | ОКАТО  Назв |
| 2 | Формы | КФорм | - | Назв |
| 3 | РазбКласт | КРег  КФорм | КРег  КФорм | - |
| 4 | Табл | КТабл | КФорм | - |
| 5 | Отв | КХоз | КФорм  КРег | - |
| 6 | Класт | ККласт | КФорм  КРег | - |
| 7 | Столбцы | КСтолбц | КТабл  КДом | - |
| 8 | ОтвКласт | КХоз  ККласт | КХоз  ККласт | - |
| 9 | СтолбцыРазб | КРег  КФорм  КСтолбц | КРег  КФорм  КСтолбц | - |
| 10 | ПарамКласт | ККласт  КСтолбц | ККласт  КСтолбц | - |
| 11 | Дом | КДом | - | - |
| 12 | ЗначДом | КЗначДом | КДом | - |
| 13 | ПрРед | КПрРед | КТПрРед | - |
| 14 | ПрИмп | КПрИмп | КТПрИмп | - |
| 15 | ПарПрРед | КПарПрРед | КПрРед  КСтолбц  КЗначДом | - |
| 16 | ТипПрРед | КТПрРед | - | Назв |
| 17 | ТипПрИмп | КТипПрИмп | - | Назв |

### 3.4.5. Определение таблиц базы данных

Далее каждое отношение трансформируется в таблицу (таблица 3.6). Имена отношений становятся именами таблиц, а имена атрибутов – именами колонок.

Таблица 3.6 – Отношения и таблицы

|  |  |  |
| --- | --- | --- |
| **№ п/п** | **Отношение** | **Таблица** |
| 1 | Рег | Regions |
| 2 | Формы | Forms |
| 3 | РазбКласт | ClusterSplitting |
| 4 | Табл | Tables |
| 5 | Отв | Values |
| 6 | Класт | Clusters |
| 7 | Столбцы | Columns |
| 8 | ОтвКласт | ClusterValues |
| 9 | СтолбцыРазб | SplittingColumns |
| 10 | ПарамКласт | ClusterParams |
| 11 | Дом | Domains |
| 12 | ЗначДом | DomainValues |
| 13 | ПрРед | EditRules |
| 14 | ПрИмп | ImputeRules |
| 15 | ПарПрРед | EditRuleParams |
| 16 | ТипПрРед | RuleTypes |
| 17 | ТипПрИмп | ImputeTypes |

Более подробная информация о свойствах получённых в таблице 3.7. таблиц представлена в таблице 3.7.

Таблица 3.7 – Описание свойств и полей таблиц

| **Имя таблицы** | **Имя атрибута** | **Тип** | **Ключевое поле** | **Обязатель- ное поле** |
| --- | --- | --- | --- | --- |
| Regions | *id\_region* | bigint | Да | Да |
| name | varchar(255) |  | Да |
| okato | varchar(50) |  | Да |
| Forms | id\_form | bigint | Да | Да |
| name | varchar(255) |  | Да |
| comment | varchar(1000) |  |  |
| ClusterSplitting | id\_region | bigint | Да | Да |
| id\_form | bigint | Да | Да |
| Tables | id\_table | bigint | Да | Да |
| id\_form | bigint |  | Да |
| name | varchar(255) |  | Да |
| comment | varchar(1000) |  |  |
| id\_object\_name | varchar(50) |  | Да |
| id\_form\_name | varchar(50) |  | Да |
| type | bigint |  | Да |
| Values | id\_value | bigint | Да | Да |
| id\_region | bigint |  | Да |
| id\_form | bigint |  | Да |
| is\_donor | bit |  |  |
| Clusters | id\_cluster | bigint | Да | Да |
| id\_region | bigint |  | Да |
| id\_form | bigint |  | Да |
| number | bigint |  | Да |
| name | varchar(255) |  |  |
| record\_count | bigint |  |  |
| Columns | id\_column | bigint | Да | Да |
| id\_domain | bigint |  | Да |
| id\_table | bigint |  | Да |
| number | bigint |  | Да |
| name | varchar(255) |  | Да |
| code | bigint |  |  |
| little | varchar(255) |  | Да |
| process | bigint |  | Да |
| shortname | varchar(255) |  | Да |
| uniquecode | bigint |  | Да |
| ClusterValues | id\_value | bigint | Да | Да |
| id\_cluster | bigint | Да | Да |
| SplittingColumns | id\_region | bigint | Да | Да |
| id\_form | bigint | Да | Да |
| id\_column | bigint | Да | Да |
| ClusterParams | id\_cluster | bigint | Да | Да |
| id\_column | bigint | Да | Да |
| mean | float |  |  |
| stddev | float |  |  |
| minvalue | float |  |  |
| maxvalue | float |  |  |
| Domains | id\_domain | bigint | Да | Да |
| comment | varchar(1000) |  |  |
| type | bigint |  | Да |
| minvalue | float |  | Да |
| maxvalue | float |  | Да |
| DomainValues | id\_domainvalue | bigint | Да | Да |
| id\_domain | bigint |  | Да |
| domain\_value | bigint |  | Да |
| title | varchar(255) |  |  |
| EditRules | id\_editrule | bigint | Да | Да |
| number | bigint |  | Да |
| ruletype | bigint |  | Да |
| EditRuleTypes | id\_editruletype | bigint | Да | Да |
| name | varchar(255) |  |  |
| ImputeRules | id\_imputrule | bigint | Да | Да |
| id\_editrule | bigint |  | Да |
| id\_column | bigint |  | Да |
| number | bigint |  | Да |
| value | float |  | Да |
| ImputeRuleTypes | id\_imputruletype | bigint | Да | Да |
| name | varchar(255) |  |  |

### 3.4.6. Схема данных

В СУБД процесс создания реляционной базы данных включает создание схемы данных. Схемой реляционной базы данных называется набор заголовков отношений, входящих в базу данных.

Схема призвана визуализировать (наглядно представить) состав отношений (таблиц) и взаимосвязей между ними. Каждое отношение изображается в виде прямоугольника, поделенного на части. Каждая часть символизирует атрибут и содержит его наименование. Стрелки показывают ссылки дочерних отношений на родительские. В схеме данных устанавливаются параметры обеспечения целостности связей в базе данных. Схема базы данных приведена на рис. 3.3.

### 3.4.7. Нормализация баз данных

Все итоговые отношения, полученные в результате тщательно проведённого анализа, находятся в 4НФ и обеспечивают адекватность предметной области, целостность информации, высокую скорость выполнения процедуры обновления данных, а следовательно и гибкую структуру хранимых данных.



Рис. 3.3 – Диаграмма физической модели базы данных

## 3.5 Тестирование программной системы

Тестирование  наиболее ответственный этап создания программного обеспечения и заключается он в выполнении программы с целью обнаружения ошибок.

Для тестирования корректности работы программной системы необходимо разработать план, в котором бы учитывалась специфика приложения.

Схема разработанного плана тестирования изображена на рис. 3.4.



Рис. 3.4. План тестирования системы

### 3.5.3 Содержание и результаты эксперимента по импутации пропущенных значений в рамках кластерного анализа

#### Описание способов оценки погрешности восстановления данных

X – множество хозяйств. Хозяйство *x* X. | X | – количество хозяйств в множестве X.

P – множество показателей опросного листа. Показатель *p*  P. | P | – количество показателей в опросном листе.

Значение показателя *p* для хозяйства *x* обозначается как *xp.*

*wp* – коэффициент нормирования показателя *p*. Нормированное значение показателя p для хозяйства x : . Нормирование может осуществляться как по всем хозяйствам, так и внутри кластера. В данном случае при оценке погрешности, *w*p = max(*xp*)-min(*xp*)

V  P – подмножество показателей, используемых для многомерной классификации.

Абсолютная погрешность импутации одного показателя для одного хозяйства:



Относительная погрешность импутации одного показателя для одного хозяйства:



Абсолютная погрешность импутации для одного хозяйства рассчитывается как сумма абсолютных погрешностей импутации по некоторому множеству показателей:

.

Множество V может содержать:

* все множество показателей,
* множество показателей.

Средняя абсолютная погрешность импутации для одного хозяйства:

.

Относительная погрешность импутации для одного хозяйства по некоторому множеству показателей:

Вводятся групповые (усредненные) оценки одного показателя и множества показателей (нормированных и ненормированных) для некоторого подмножества YX хозяйств.

#### Описание эксперимента

**На первом этапе** множество из 29010 хозяйств разбивалось на 16 классов по значениям 43 основных показателей.

**Затем** из каждого класса выбирались по 20 хозяйств. Если же число хозяйств, попавших в класс было меньше 20 (классы С2 – 14 хозяйств, С5 – 4 хозяйства, С11 – 3 хозяйства и С16 – 2 хозяйства), то в выборку включались все хозяйства такого класса. В таблице 4.1 приведены номера всех хозяйств, вошедших в выборку для проведения эксперимента по импутации данных. Содержание первых трех строк таблицы 4.1 расшифровано непосредственно в таблице. В следующих 20 строках указаны номера хозяйств, вошедших в каждый из 16 классов. Всего в выборку для проведения эксперимента вошли 263 хозяйства.

Таблица 3.7

Классы и номера хозяйств в классах, выбранных для импутации

| Имя кластера | CBF | EGF | HCD | HDC | CDB | CBE | HEC | OOC | DCB | FDC | FGH | OOA | OCG | OCF | EHF | ADE |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Номер | ***1*** | ***2*** | ***3*** | ***4*** | ***5*** | ***6*** | ***7*** | ***8*** | ***9*** | ***10*** | ***11*** | ***12*** | ***13*** | ***14*** | ***15*** | ***16*** |
| Кол-во эл-в | **2767** | **14** | **82** | **38** | **4** | **723** | **141** | **6658** | **1359** | **634** | **3** | **11373** | **2599** | **1422** | **1191** | **2** |
|  | 24469 | 27000 | 24881 | 24775 | 25084 | 24515 | 24491 | 24505 | 24521 | 24525 | 32324 | 24513 | 24478 | 24473 | 24476 | 26312 |
|  | 24470 | 27001 | 25276 | 24865 | 26432 | 24519 | 24520 | 24633 | 24748 | 24557 | 34675 | 24534 | 24483 | 24482 | 24502 | 26428 |
|  | 24471 | 28140 | 25454 | 24891 | 34396 | 24523 | 24575 | 24668 | 24812 | 24561 | 52411 | 24539 | 24493 | 24487 | 24531 |  |
|  | 24472 | 30007 | 26310 | 24892 | 36980 | 24524 | 24581 | 24704 | 24853 | 24588 |  | 24548 | 24506 | 24490 | 24589 |  |
|  | 24474 | 30008 | 26429 | 25023 |  | 24543 | 24638 | 24740 | 24882 | 24598 |  | 24554 | 24512 | 24504 | 24605 |  |
|  | 24475 | 30021 | 26512 | 25054 |  | 24545 | 24842 | 24741 | 24893 | 24641 |  | 24565 | 24516 | 24535 | 24615 |  |
|  | 24477 | 35261 | 26553 | 25376 |  | 24579 | 24884 | 24803 | 24913 | 24665 |  | 24574 | 24529 | 24544 | 24671 |  |
|  | 24479 | 36239 | 27157 | 26163 |  | 24587 | 24889 | 24828 | 24952 | 24746 |  | 24582 | 24538 | 24558 | 24713 |  |
|  | 24480 | 36775 | 27163 | 26474 |  | 24602 | 24907 | 24888 | 24972 | 24768 |  | 24608 | 24540 | 24571 | 24898 |  |
|  | 24481 | 37447 | 27192 | 26509 |  | 24604 | 25067 | 24920 | 24995 | 24771 |  | 24655 | 24578 | 24584 | 25082 |  |
|  | 24484 | 37662 | 27225 | 26741 |  | 24607 | 25104 | 24963 | 25087 | 24815 |  | 24656 | 24585 | 24594 | 25090 |  |
|  | 24485 | 37719 | 27407 | 26742 |  | 24618 | 25105 | 24975 | 25145 | 24835 |  | 24658 | 24590 | 24600 | 25092 |  |
|  | 24486 | 38139 | 27635 | 26822 |  | 24626 | 25238 | 24981 | 25166 | 24870 |  | 24666 | 24595 | 24619 | 25194 |  |
|  | 24488 | 38212 | 27974 | 27073 |  | 24627 | 25335 | 25038 | 25200 | 25069 |  | 24670 | 24596 | 24622 | 25196 |  |
|  | 24489 |  | 27981 | 27229 |  | 24647 | 25551 | 25048 | 25208 | 25073 |  | 24673 | 24603 | 24623 | 25229 |  |
|  | 24492 |  | 27992 | 27612 |  | 24714 | 25592 | 25055 | 25224 | 25081 |  | 24677 | 24614 | 24628 | 25235 |  |
|  | 24494 |  | 27995 | 27929 |  | 24722 | 25611 | 25068 | 25328 | 25086 |  | 24681 | 24616 | 24635 | 25242 |  |
|  | 24495 |  | 27996 | 28129 |  | 24745 | 25617 | 25106 | 25433 | 25091 |  | 24683 | 24636 | 24643 | 25245 |  |
|  | 24496 |  | 28392 | 28870 |  | 24747 | 25618 | 25134 | 25441 | 25193 |  | 24684 | 24650 | 24645 | 25247 |  |
|  | 24497 |  | 29203 | 28889 |  | 24766 | 25628 | 25164 | 25457 | 25233 |  | 24685 | 24705 | 24662 | 25250 |  |

**На третьем этапе** осуществлялся собственно эксперимент по восстановлению пропущенных (удаленных) данных. Моделировались два случая пропусков:

1. Пропущено значение только одного из 43 показателей, (моделирование проводилось для каждого показателя).
2. Пропущены значения всех показателей одной из 8 групп А, В, …, Н (моделирование проводилось для каждой группы).

**Восстановление (импутация)** пропущенного значения (значений) осуществлялось по следующему алгоритму:

- пропущенное значение Vkj, k = 1, 2, …, 43 – номер показателя из списка, j – номер хозяйства из выборки, последовательно заменялось соответствующим средним значением в кластере (в каждом кластере от первого до шестнадцатого). Вычислялись расстояния Из полученных шестнадцати чисел выбиралось наименьшее.

Таким образом, находится кластер, ближайший к j-му хозяйству. Затем внутри выбранного класса находится донор – ближайший сосед. Если показатель Vk не связан с другими показателями, то в j-е хозяйство импутировалось значение этого показателя от донора. Если показатель Vk связан с другими показателями, то импутировались значения всей группы.

*6.3.3 Анализ результатов эксперимента*

Относительная ошибка восстановления значений отдельного показателя вычислялась по формуле

где ∆ = Vkmax – Vkmin;

S = 20, если в кластере больше 20 элементов и S =14, 4, 3, 2 для кластеров С2, С5, С11 и С16 соответственно (см. таблицу 4.1).

Относительные ошибки восстановления δk, по всем 16 кластерам при удалении только одного показателя приведены в таблице 3.7.

Всего было выполнено 263 х 43 = 11309 восстановлений пропущенных (удаленных) значений.

В последней строке таблицы 4.2 приведены средние значения ошибки по каждому кластеру

а в последнем столбце – средние значения ошибки по каждому показателю

Таблица 3.8

Относительные ошибки восстановления при удалении одного элемента

|  |  | ***1*** | ***2*** | ***3*** | ***4*** | ***5*** | ***6*** | ***7*** | ***8*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **2767** | **14** | **82** | **38** | **4** | **723** | **141** | **6658** |
|  |  | 24469 | 27000 | 24881 | 24775 | 25084 | 24515 | 24491 | 24505 |
|  |  | 24470 | 27001 | 25276 | 24865 | 26432 | 24519 | 24520 | 24633 |
|  |  | 24471 | 28140 | 25454 | 24891 | 34396 | 24523 | 24575 | 24668 |
|  |  | 24472 | 30007 | 26310 | 24892 | 36980 | 24524 | 24581 | 24704 |
|  |  | 24474 | 30008 | 26429 | 25023 |  | 24543 | 24638 | 24740 |
|  |  | 24475 | 30021 | 26512 | 25054 |  | 24545 | 24842 | 24741 |
|  |  | 24477 | 35261 | 26553 | 25376 |  | 24579 | 24884 | 24803 |
|  |  | 24479 | 36239 | 27157 | 26163 |  | 24587 | 24889 | 24828 |
|  |  | 24480 | 36775 | 27163 | 26474 |  | 24602 | 24907 | 24888 |
|  |  | 24481 | 37447 | 27192 | 26509 |  | 24604 | 25067 | 24920 |
|  |  | 24484 | 37662 | 27225 | 26741 |  | 24607 | 25104 | 24963 |
|  |  | 24485 | 37719 | 27407 | 26742 |  | 24618 | 25105 | 24975 |
|  |  | 24486 | 38139 | 27635 | 26822 |  | 24626 | 25238 | 24981 |
|  |  | 24488 | 38212 | 27974 | 27073 |  | 24627 | 25335 | 25038 |
|  |  | 24489 |  | 27981 | 27229 |  | 24647 | 25551 | 25048 |
|  |  | 24492 |  | 27992 | 27612 |  | 24714 | 25592 | 25055 |
|  |  | 24494 |  | 27995 | 27929 |  | 24722 | 25611 | 25068 |
|  |  | 24495 |  | 27996 | 28129 |  | 24745 | 25617 | 25106 |
|  |  | 24496 |  | 28392 | 28870 |  | 24747 | 25618 | 25134 |
|  |  | 24497 |  | 29203 | 28889 |  | 24766 | 25628 | 25164 |
| v1 | ЧЧлСем | 27% | 29% | 38% | 24% | 38% | 24% | 8% | 7% |
| v2 | ЧислСезРаб | 0% | 14% | 5% | 0% | 0% | 0% | 0% | 0% |
| v3 | ПлЗемСобств | 2% | 14% | 9% | 13% | 34% | 2% | 2% | 0% |
| v4 | ПрЗемУч | 10% | 14% | 19% | 14% | 61% | 21% | 3% | 13% |
| v5 | ПоЗемУч | 0% | 0% | 0% | 0% | 50% | 0% | 0% | 0% |
| v6 | ОбщИспЗем | 2% | 14% | 9% | 13% | 67% | 2% | 2% | 4% |
| v7 | ОбщПосУр | 2% | 24% | 9% | 12% | 26% | 4% | 7% | 6% |
| v8 | КарПосУр | 29% | 23% | 25% | 23% | 67% | 9% | 47% | 49% |
| v9 | ОвОткрГр | 8% | 28% | 14% | 15% | 37% | 11% | 12% | 13% |
| v10 | ОвЗакрГр | 0% | 0% | 5% | 0% | 0% | 5% | 0% | 0% |
| v11 | КормКульт | 0% | 12% | 0% | 1% | 0% | 0% | 2% | 0% |
| v12 | ПлЗаПредУч | 0% | 0% | 0% | 5% | 0% | 0% | 0% | 0% |
| v13 | Ябл | 3% | 29% | 7% | 20% | 36% | 5% | 9% | 5% |
| v14 | Груша | 0% | 7% | 5% | 8% | 50% | 0% | 2% | 0% |
| v15 | Слива | 4% | 7% | 21% | 18% | 33% | 5% | 9% | 9% |
| v16 | Виншя | 2% | 6% | 4% | 6% | 67% | 8% | 3% | 3% |
| v17 | ЗемлКлубн | 1% | 27% | 10% | 23% | 0% | 9% | 2% | 2% |
| v18 | Смор | 2% | 0% | 6% | 32% | 0% | 6% | 3% | 6% |
| v19 | КРС | 7% | 16% | 13% | 20% | 25% | 7% | 5% | 3% |
| v20 | КРСмол | 7% | 16% | 13% | 20% | 25% | 7% | 5% | 3% |
| v21 | КРСмолКор | 35% | 35% | 13% | 17% | 25% | 15% | 20% | 0% |
| v22 | КРМмяс | 0% | 7% | 0% | 0% | 0% | 5% | 0% | 0% |
| v23 | КРСмясКор | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% |
| v24 | Свиньи | 5% | 7% | 14% | 6% | 17% | 6% | 16% | 0% |
| v25 | Овцы | 0% | 22% | 0% | 1% | 0% | 3% | 5% | 1% |
| v26 | Козы | 6% | 11% | 0% | 0% | 0% | 0% | 0% | 1% |
| v27 | Птица | 3% | 17% | 7% | 19% | 25% | 4% | 13% | 10% |
| v28 | КурЯич | 27% | 21% | 10% | 18% | 75% | 18% | 20% | 9% |
| v29 | КурМяс | 0% | 19% | 4% | 12% | 0% | 1% | 12% | 9% |
| v30 | Утки | 7% | 20% | 6% | 12% | 50% | 5% | 11% | 6% |
| v31 | Гуси | 0% | 13% | 10% | 0% | 0% | 0% | 7% | 1% |
| v32 | Лош | 3% | 17% | 5% | 6% | 0% | 0% | 0% | 0% |
| v33 | МСКРС | 11% | 19% | 13% | 18% | 31% | 9% | 12% | 7% |
| v34 | МССвин | 8% | 7% | 24% | 18% | 13% | 13% | 10% | 2% |
| v35 | МСОвКоз | 3% | 21% | 0% | 2% | 0% | 2% | 5% | 1% |
| v36 | МСЛош | 3% | 11% | 5% | 6% | 0% | 3% | 0% | 0% |
| v37 | МСПтиц | 12% | 22% | 7% | 16% | 50% | 15% | 17% | 4% |
| v38 | Тракт | 0% | 14% | 5% | 55% | 0% | 0% | 0% | 0% |
| v39 | ПлТракт | 0% | 0% | 5% | 50% | 0% | 0% | 7% | 0% |
| v40 | КосТракт | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% |
| v41 | Мблок | 0% | 0% | 0% | 15% | 0% | 0% | 0% | 0% |
| v42 | АвтГрПасс | 10% | 36% | 0% | 10% | 0% | 15% | 15% | 0% |
| v43 | УстДоил | 5% | 0% | 5% | 5% | 0% | 0% | 0% | 0% |
|  | Среднее | **6%** | **14%** | **8%** | **13%** | **21%** | **6%** | **7%** | **4%** |

Окончание табл. 3.8

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | DCB | FDC | FGH | OOA | OCG | OCF | EHF | ADE |  |
|  |  | ***9*** | ***10*** | ***11*** | ***12*** | ***13*** | ***14*** | ***15*** | ***16*** |  |
|  |  | **1359** | **634** | **3** | **11373** | **2599** | **1422** | **1191** | **2** |  |
| v1 | ЧЧлСем | 18% | 14% | 50% | 23% | 18% | 16% | 24% | 0% | **22%** |
| v2 | ЧислСезРаб | 0% | 3% | 0% | 0% | 0% | 0% | 0% | 0% | **1%** |
| v3 | ПлЗемСобств | 2% | 6% | 38% | 0% | 2% | 2% | 5% | 50% | **11%** |
| v4 | ПрЗемУч | 4% | 7% | 38% | 9% | 10% | 7% | 8% | 50% | **18%** |
| v5 | ПоЗемУч | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 50% | **6%** |
| v6 | ОбщИспЗем | 2% | 6% | 38% | 1% | 2% | 2% | 5% | 50% | **14%** |
| v7 | ОбщПосУр | 5% | 6% | 33% | 1% | 4% | 7% | 6% | 50% | **13%** |
| v8 | КарПосУр | 36% | 34% | 33% | 40% | 61% | 40% | 38% | 0% | **35%** |
| v9 | ОвОткрГр | 5% | 5% | 48% | 16% | 12% | 10% | 9% | 50% | **18%** |
| v10 | ОвЗакрГр | 0% | 0% | 0% | 0% | 0% | 4% | 0% | 0% | **1%** |
| v11 | КормКульт | 1% | 1% | 0% | 0% | 0% | 0% | 0% | 0% | **1%** |
| v12 | ПлЗаПредУч | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% | **0%** |
| v13 | Ябл | 4% | 3% | 53% | 0% | 1% | 1% | 17% | 50% | **15%** |
| v14 | Груша | 10% | 1% | 33% | 0% | 3% | 2% | 3% | 0% | **8%** |
| v15 | Слива | 10% | 2% | 33% | 1% | 2% | 1% | 7% | 50% | **13%** |
| v16 | Виншя | 4% | 5% | 0% | 1% | 1% | 4% | 4% | 50% | **10%** |
| v17 | ЗемлКлубн | 2% | 4% | 33% | 1% | 1% | 4% | 10% | 0% | **8%** |
| v18 | Смор | 10% | 5% | 0% | 7% | 5% | 2% | 6% | 50% | **9%** |
| v19 | КРС | 6% | 4% | 0% | 0% | 6% | 8% | 5% | 50% | **11%** |
| v20 | КРСмол | 6% | 4% | 0% | 0% | 6% | 8% | 5% | 50% | **11%** |
| v21 | КРСмолКор | 13% | 18% | 0% | 0% | 15% | 20% | 10% | 50% | **18%** |
| v22 | КРМмяс | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% | **1%** |
| v23 | КРСмясКор | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% | **0%** |
| v24 | Свиньи | 3% | 9% | 0% | 0% | 1% | 1% | 2% | 50% | **9%** |
| v25 | Овцы | 2% | 2% | 0% | 0% | 0% | 2% | 2% | 0% | **2%** |
| v26 | Козы | 1% | 3% | 0% | 0% | 0% | 0% | 0% | 0% | **1%** |
| v27 | Птица | 5% | 7% | 0% | 0% | 2% | 4% | 12% | 50% | **11%** |
| v28 | КурЯич | 9% | 11% | 33% | 1% | 36% | 10% | 12% | 0% | **19%** |
| v29 | КурМяс | 5% | 14% | 50% | 0% | 0% | 1% | 12% | 50% | **12%** |
| v30 | Утки | 3% | 14% | 33% | 1% | 5% | 5% | 2% | 50% | **14%** |
| v31 | Гуси | 0% | 3% | 33% | 0% | 1% | 7% | 4% | 50% | **8%** |
| v32 | Лош | 3% | 1% | 0% | 0% | 1% | 3% | 0% | 0% | **2%** |
| v33 | МСКРС | 7% | 9% | 0% | 4% | 7% | 12% | 7% | 50% | **14%** |
| v34 | МССвин | 8% | 6% | 0% | 3% | 2% | 4% | 4% | 50% | **11%** |
| v35 | МСОвКоз | 3% | 3% | 0% | 0% | 1% | 1% | 2% | 0% | **3%** |
| v36 | МСЛош | 0% | 4% | 0% | 0% | 2% | 3% | 1% | 0% | **2%** |
| v37 | МСПтиц | 11% | 11% | 3% | 4% | 6% | 18% | 18% | 50% | **17%** |
| v38 | Тракт | 0% | 5% | 0% | 0% | 0% | 0% | 0% | 50% | **8%** |
| v39 | ПлТракт | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% | **4%** |
| v40 | КосТракт | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 100% | **6%** |
| v41 | Мблок | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% | **1%** |
| v42 | АвтГрПасс | 3% | 0% | 33% | 0% | 0% | 5% | 0% | 0% | **8%** |
| v43 | УстДоил | 0% | 10% | 0% | 0% | 5% | 0% | 0% | 0% | **2%** |
|  | Среднее | **5%** | **6%** | **14%** | **3%** | **5%** | **5%** | **6%** | **28%** | **9%** |

Как и следовало ожидать, наибольшие значения ошибок восстановления наблюдаются в кластерах с небольшим числом элементов – в С5, С11 и С16. Это следует и из значений показателей δk и из средних значений ошибок по классу. Например, в классе С5: δ8 = 76%, δ16 =67%, δ28 = 75%, а = 21%. В классе С16 (2 хозяйства) = 28% является максимальными из всех 16 классов.

Минимальное значение средней ошибки восстановления наблюдается в классе С11: = 3%. Этот результат является логическим следствием выделения одного из наиболее однородных кластеров с относительно низкой, по сравнению с другими кластерами, вариацией показателей. Число показателей, восстановленных без ошибки в этом классе равно 28. Этот показатель также является наилучшим по сравнению с другими классами.

**Общий вывод:**

* наибольшие ошибки восстановления пропущенных значений наблюдается в классах с небольшим числом хозяйств, имеющих экстремальные значения отдельных показателей.
* для повышения точности восстановления необходимо учитывать функциональные и корреляционные взаимосвязи между показателями.

В исходных данных каждого хозяйства, вошедшего в выборку удалялись последовательно группы показателей и осуществлялось восстановление. Однако, в данном случае от найденного донора импутировались значение не одного показателя, а всей группы А (V1, V2), B (V3…V6) и т.д. Всего было выполнено 263 х 8 = 2104 восстановлений групп показателей.

Результаты этого эксперимента приведены в таблице 3.9.

Таблица 3.9

Ошибки восстановления при удалении группы показателей

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | CBF | EGF | HCD | HDC | CDB | CBE | HEC | OOC |
|  |  |  |  | ***1*** | ***2*** | ***3*** | ***4*** | ***5*** | ***6*** | ***7*** | ***8*** |
|  |  |  |  | **2767** | **14** | **82** | **38** | **4** | **723** | **141** | **6658** |
|  |  |  |  | 24469 | 27000 | 24881 | 24775 | 25084 | 24515 | 24491 | 24505 |
|  |  |  |  | 24470 | 27001 | 25276 | 24865 | 26432 | 24519 | 24520 | 24633 |
|  |  |  |  | 24471 | 28140 | 25454 | 24891 | 34396 | 24523 | 24575 | 24668 |
|  |  |  |  | 24472 | 30007 | 26310 | 24892 | 36980 | 24524 | 24581 | 24704 |
|  |  |  |  | 24474 | 30008 | 26429 | 25023 |  | 24543 | 24638 | 24740 |
|  |  |  |  | 24475 | 30021 | 26512 | 25054 |  | 24545 | 24842 | 24741 |
|  |  |  |  | 24477 | 35261 | 26553 | 25376 |  | 24579 | 24884 | 24803 |
|  |  |  |  | 24479 | 36239 | 27157 | 26163 |  | 24587 | 24889 | 24828 |
|  |  |  |  | 24480 | 36775 | 27163 | 26474 |  | 24602 | 24907 | 24888 |
|  |  |  |  | 24481 | 37447 | 27192 | 26509 |  | 24604 | 25067 | 24920 |
|  |  |  |  | 24484 | 37662 | 27225 | 26741 |  | 24607 | 25104 | 24963 |
|  |  |  |  | 24485 | 37719 | 27407 | 26742 |  | 24618 | 25105 | 24975 |
|  |  |  |  | 24486 | 38139 | 27635 | 26822 |  | 24626 | 25238 | 24981 |
|  |  |  |  | 24488 | 38212 | 27974 | 27073 |  | 24627 | 25335 | 25038 |
|  |  |  |  | 24489 |  | 27981 | 27229 |  | 24647 | 25551 | 25048 |
|  |  |  |  | 24492 |  | 27992 | 27612 |  | 24714 | 25592 | 25055 |
|  |  |  |  | 24494 |  | 27995 | 27929 |  | 24722 | 25611 | 25068 |
|  |  |  |  | 24495 |  | 27996 | 28129 |  | 24745 | 25617 | 25106 |
|  |  |  |  | 24496 |  | 28392 | 28870 |  | 24747 | 25618 | 25134 |
|  |  |  |  | 24497 |  | 29203 | 28889 |  | 24766 | 25628 | 25164 |
| **A** | **1** | v1 | ЧЧлСем | 21% | 29% | 24% | 38% | 38% | 24% | 8% | 7% |
| **2** | v2 | ЧислСезРаб | 0% | 14% | 0% | 5% | 0% | 0% | 0% | 0% |
| **B** | **1** | v3 | ПлЗемСобств | 1% | 14% | 19% | 14% | 34% | 3% | 3% | 0% |
| **2** | v4 | ПрЗемУч | 9% | 14% | 19% | 14% | 43% | 21% | 3% | 13% |
| **3** | v5 | ПоЗемУч | 55% | 0% | 0% | 0% | 75% | 61% | 0% | 0% |
| **4** | v6 | ОбщИспЗем | 1% | 14% | 19% | 14% | 33% | 3% | 3% | 11% |
| **C** | **1** | v7 | ОбщПосУр | 14% | 24% | 26% | 21% | 26% | 11% | 39% | 30% |
| **2** | v8 | КарПосУр | 27% | 23% | 26% | 22% | 76% | 9% | 44% | 50% |
| **3** | v9 | ОвОткрГр | 9% | 28% | 13% | 16% | 50% | 10% | 17% | 15% |
| **4** | v10 | ОвЗакрГр | 0% | 0% | 3% | 0% | 0% | 5% | 0% | 0% |
| **5** | v11 | КормКульт | 0% | 12% | 0% | 1% | 25% | 0% | 1% | 0% |
| **6** | v12 | ПлЗаПредУч | 0% | 0% | 0% | 5% | 0% | 0% | 0% | 0% |
| **D** | **1** | v13 | Ябл | 1% | 29% | 8% | 21% | 36% | 6% | 9% | 6% |
| **2** | v14 | Груша | 0% | 7% | 7% | 8% | 50% | 0% | 2% | 0% |
| **3** | v15 | Слива | 1% | 7% | 21% | 18% | 33% | 5% | 9% | 9% |
| **4** | v16 | Виншя | 0% | 6% | 7% | 6% | 67% | 7% | 0% | 3% |
| **5** | v17 | ЗемлКлубн | 0% | 27% | 8% | 21% | 0% | 9% | 2% | 2% |
| **6** | v18 | Смор | 0% | 0% | 3% | 32% | 0% | 6% | 2% | 6% |
| **E** | **1** | v19 | КРС | 15% | 21% | 23% | 32% | 25% | 13% | 12% | 3% |
| **2** | v20 | КРСмол | 15% | 20% | 23% | 32% | 25% | 13% | 12% | 3% |
| **3** | v21 | КРСмолКор | 33% | 38% | 20% | 20% | 25% | 20% | 25% | 0% |
| **4** | v22 | КРМмяс | 0% | 7% | 0% | 0% | 0% | 5% | 0% | 0% |
| **5** | v23 | КРСмясКор | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% |
| **6** | v24 | Свиньи | 6% | 7% | 14% | 6% | 17% | 6% | 16% | 0% |
| **7** | v25 | Овцы | 1% | 22% | 0% | 1% | 0% | 3% | 5% | 1% |
| **8** | v26 | Козы | 1% | 20% | 0% | 0% | 0% | 0% | 0% | 1% |
| **F** | **1** | v27 | Птица | 9% | 23% | 10% | 21% | 25% | 14% | 23% | 15% |
| **2** | v28 | КурЯич | 22% | 27% | 10% | 18% | 75% | 17% | 22% | 12% |
| **3** | v29 | КурМяс | 0% | 15% | 4% | 13% | 0% | 1% | 15% | 9% |
| **4** | v30 | Утки | 0% | 23% | 6% | 12% | 50% | 5% | 13% | 6% |
| **5** | v31 | Гуси | 0% | 17% | 7% | 2% | 0% | 0% | 2% | 0% |
| **6** | v32 | Лош | 3% | 19% | 5% | 6% | 0% | 0% | 0% | 0% |
| **G** | **1** | v33 | МСКРС | 13% | 19% | 11% | 18% | 31% | 11% | 13% | 8% |
| **2** | v34 | МССвин | 8% | 7% | 26% | 18% | 13% | 16% | 14% | 2% |
| **3** | v35 | МСОвКоз | 2% | 20% | 0% | 2% | 0% | 1% | 5% | 1% |
| **4** | v36 | МСЛош | 1% | 12% | 0% | 6% | 0% | 7% | 0% | 0% |
| **5** | v37 | МСПтиц | 11% | 24% | 7% | 17% | 50% | 20% | 21% | 5% |
| **H** | **1** | v38 | Тракт | 0% | 14% | 5% | 75% | 0% | 0% | 0% | 0% |
| **2** | v39 | ПлТракт | 0% | 0% | 5% | 65% | 0% | 0% | 7% | 0% |
| **3** | v40 | КосТракт | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% |
| **4** | v41 | Мблок | 0% | 0% | 0% | 15% | 0% | 0% | 0% | 0% |
| **5** | v42 | АвтГрПасс | 5% | 36% | 0% | 15% | 0% | 15% | 10% | 0% |
| **6** | v43 | УстДоил | 0% | 0% | 5% | 5% | 0% | 0% | 0% | 0% |
|  |  |  | Среднее | **7%** | **15%** | **9%** | **15%** | **21%** | **8%** | **8%** | **5%** |

Окончание табл. 3.10

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  | DCB | FDC | FGH | OOA | OCG | OCF | EHF | ADE |  |
| **A** | v1 | ЧЧлСем | 18% | 14% | 50% | 23% | 18% | 16% | 24% | 0% | **22%** |
|  | v2 | ЧислСезРаб | 0% | 3% | 0% | 0% | 0% | 0% | 0% | 0% | **1%** |
| **B** | v3 | ПлЗемСобств | 4% | 7% | 38% | 0% | 7% | 4% | 8% | 50% | **13%** |
|  | v4 | ПрЗемУч | 4% | 7% | 38% | 9% | 10% | 7% | 8% | 50% | **17%** |
|  | v5 | ПоЗемУч | 0% | 0% | 0% | 0% | 13% | 17% | 0% | 50% | **17%** |
|  | v6 | ОбщИспЗем | 4% | 7% | 38% | 6% | 7% | 4% | 8% | 50% | **14%** |
| **C** | v7 | ОбщПосУр | 16% | 22% | 33% | 21% | 30% | 40% | 33% | 50% | **27%** |
|  | v8 | КарПосУр | 36% | 35% | 33% | 41% | 65% | 40% | 38% | 0% | **35%** |
|  | v9 | ОвОткрГр | 6% | 7% | 48% | 15% | 12% | 10% | 10% | 50% | **20%** |
|  | v10 | ОвЗакрГр | 0% | 0% | 0% | 0% | 0% | 4% | 1% | 0% | **1%** |
|  | v11 | КормКульт | 1% | 1% | 0% | 0% | 0% | 0% | 0% | 0% | **3%** |
|  | v12 | ПлЗаПредУч | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% | **0%** |
| **D** | v13 | Ябл | 4% | 5% | 53% | 0% | 2% | 1% | 18% | 50% | **16%** |
|  | v14 | Груша | 10% | 2% | 33% | 0% | 3% | 2% | 3% | 0% | **8%** |
|  | v15 | Слива | 8% | 3% | 33% | 1% | 2% | 1% | 8% | 50% | **13%** |
|  | v16 | Виншя | 3% | 5% | 0% | 1% | 1% | 4% | 5% | 50% | **10%** |
|  | v17 | ЗемлКлубн | 2% | 4% | 33% | 1% | 1% | 4% | 5% | 0% | **7%** |
|  | v18 | Смор | 8% | 6% | 0% | 7% | 5% | 2% | 5% | 50% | **8%** |
| **E** | v19 | КРС | 9% | 6% | 0% | 0% | 7% | 17% | 6% | 50% | **15%** |
|  | v20 | КРСмол | 9% | 6% | 0% | 0% | 7% | 17% | 6% | 50% | **15%** |
|  | v21 | КРСмолКор | 15% | 19% | 0% | 0% | 13% | 25% | 6% | 50% | **19%** |
|  | v22 | КРМмяс | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% | **1%** |
|  | v23 | КРСмясКор | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% | **0%** |
|  | v24 | Свиньи | 3% | 9% | 0% | 0% | 1% | 1% | 2% | 50% | **9%** |
|  | v25 | Овцы | 2% | 2% | 0% | 0% | 0% | 2% | 2% | 0% | **2%** |
|  | v26 | Козы | 1% | 3% | 0% | 0% | 0% | 0% | 0% | 0% | **2%** |
| **F** | v27 | Птица | 9% | 15% | 33% | 1% | 14% | 11% | 18% | 50% | **18%** |
|  | v28 | КурЯич | 10% | 10% | 33% | 1% | 36% | 10% | 11% | 0% | **20%** |
|  | v29 | КурМяс | 5% | 15% | 58% | 0% | 1% | 2% | 14% | 50% | **13%** |
|  | v30 | Утки | 3% | 13% | 33% | 1% | 5% | 5% | 2% | 50% | **14%** |
|  | v31 | Гуси | 0% | 1% | 67% | 0% | 1% | 9% | 4% | 50% | **10%** |
|  | v32 | Лош | 0% | 3% | 0% | 0% | 1% | 3% | 0% | 0% | **2%** |
| **G** | v33 | МСКРС | 6% | 10% | 0% | 4% | 6% | 14% | 8% | 50% | **14%** |
|  | v34 | МССвин | 8% | 9% | 0% | 3% | 2% | 3% | 4% | 50% | **11%** |
|  | v35 | МСОвКоз | 3% | 4% | 0% | 0% | 1% | 1% | 2% | 0% | **3%** |
|  | v36 | МСЛош | 0% | 5% | 0% | 0% | 2% | 3% | 1% | 0% | **2%** |
|  | v37 | МСПтиц | 11% | 13% | 3% | 5% | 8% | 19% | 22% | 50% | **18%** |
| **H** | v38 | Тракт | 0% | 5% | 0% | 0% | 0% | 0% | 0% | 100% | **12%** |
|  | v39 | ПлТракт | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% | **5%** |
|  | v40 | КосТракт | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 100% | **6%** |
|  | v41 | Мблок | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% | **1%** |
|  | v42 | АвтГрПасс | 3% | 0% | 33% | 0% | 0% | 5% | 0% | 0% | **8%** |
|  | v43 | УстДоил | 0% | 10% | 0% | 0% | 5% | 0% | 0% | 0% | **2%** |
|  |  | Среднее | **5%** | **7%** | **16%** | **3%** | **7%** | **7%** | **7%** | **29%** | **11%** |

Как следует из таблицы 3.10, результаты восстановления при удалении групп показателей в целом близки к данным таблицы 3.11. То есть точность восстановления групп одноименных, а часто и взаимосвязанных показателей, незначительно отличается от точности восстановления одного из 43 показателей. Средняя ошибка по последнему столбцу в таблице 4.3 равна 9%, а в таблице 3.4 – 11%.

Как при восстановлении значения одного показателя, так и при восстановлении значений группы показателей наилучшие результаты получены в кластере С11(11373 хозяйства) - = 3%, а наиболее плохие в кластерах с небольшим числом хозяйств: С5 (4 хозяйства) - = 219; С11 (3 хозяйства – 16%), С16 (2 хозяйства – 29%).

Следующий этап проверки эффективности процедур импутации в условиях предварительного выделения однородных групп хозяйств методами кластерного анализа заключался в восстановлении значений показателей из различных разделов переписного листа 3 ЛПХ, не вошедших в группу 43 основных показателей. В первую группу таких показателей вошли все 5 показателей раздела I «Общая характеристика» переписного листа, два показателя из раздела II «Трудовые ресурсы» (таблица 3.11). Во вторую группу включены все показатели второй страницы переписного листа 3 ЛПХ (Раздел III «Земельные ресурсы и их использование»), кроме показателей «Приусадебный земельный участок, общая площадь, га» и «Полевые земельные участки, общая площадь, га». Последние два показателя входят в группу 43 основных (таблица 3.9), по значениям которых осуществлялось разбиение всей совокупности на кластеры. В третьей группе один показатель Group 3-17 «Укажите суммарную площадь полученных долей, га». В последней четвертой группе 6 показателей из различных разделов переписного листа 3 ЛПХ. Их наименования и условные обозначения приведены в таблице 3.11.

Таблица 3.11 –

Обозначение признаков, используемых при импутации показателей

|  |  |  |  |
| --- | --- | --- | --- |
| **Группа** | **Номер** | **Наименование** | **Сокращение** |
| **1** | **1** | Число совместно проживаемых лиц | Group1\_6 |
| **2** | Производилась ли сельскохозяйственная продукция | Group1\_7 |
| **3** | Основная цель производства | Group1\_8 |
| **4** | Являетесь ли Вы членом сельскохозяйственного потребительского кооператива | Group1\_9 |
| **5** | Членом какого сельскохозяйственного кооператива Вы являетесь? | Group1\_10 |
| **6** | Привлекали ли Вы сезонных работников? | Group1\_12 |
| **7** | Численность сезонных работников | Group1\_13 |
| **2** | **1** | Число приусадебных земельных участков, шт | Group2\_1 |
| **2** | Число полевых земельных участков, шт | Group2\_4 |
| **3** | Площадь земли в собственности, га | Group2\_7 |
| **4** | Площадь земли в пожизненном наслед. владении, га | Group2\_9 |
| **5** | Площадь земли в постоянном пользовании, га | Group2\_11 |
| **6** | Площадь земли в безвозмездном срочном пользовании, га | Group2\_13 |
| **7** | Площадь арендованной земли, га | Group2\_15 |
| **8** | Площадь земли, сданная в аренду другим лицам, га | Group2\_17 |
| **9** | Площадь земли, переданная другим лицам, га | Group2\_19 |
| **10** | Земляника и клубника | Group2\_21 |
| **3** | **1** | Укажите суммарную площадь полученных земельных долей, га | Group3\_17 |
| **4** | **1** | Продано картофеля, % от общего объема | Group13\_28 |
| **2** | Продано скота и птицы, % от общего объема | Group14\_5 |
| **3** | Продано молока, % от общего объема | Group14\_6 |
| **4** | Продано яйцо птицы, % от общего объема | Group14\_7 |
| **5** | Общая площадь теплиц, кв. м. | Group15\_2 |
| **6** | Мукомольное оборудование, шт | Group16\_15 |

# 4 Программная документация

## 4.1. Описание применения

### 4.1.1. Назначение программы

Разрабатываемая система автокоррекции и импутации предназначена для решения следующиз задач:

* автокоррекция. Под автокоррекцией (редактированием) понимается:
  + проверка значений каждого поля отдельной записи (т.е. зафиксированного ответа на данный вопрос анкеты) на допустимость этого значения;
  + проверка записи на допустимость комбинаций значений различных полей.
* импутация. Под импутацией для конкретной записи понимается изменение значений некоторых ее полей таким образом, чтобы все соотношения, связывающие между собой переменные, и используемые для контроля корректности данных обследования, были удовлетворены.

В соответствии со структурой процесса автокоррекции и импутации работа с системой будет состоять из следующих этапов:

* этап инициализации и настройки системы на работу с конкретным типом переписных листов (анкет).
* этап описания и анализа правил редактирования. На данном этапе необходимо:
  + задать правила редактирования;
  + задать категории анкет;
  + проанализировать правила на полноту, непротиворечивость и избыточность.
* этап автокоррекции и импутации. Данный этап состоит из следующих шагов:
  + проверка записей на корректность правилами редактирования;
  + локализация ошибки;
  + импутация различными методами;
  + ведение протокола импутации.

### 4.1.2. Условия применения

Системные требования, необходимые для установки и эффективного функционирования программного комплекса:

1. Microsoft Windows 7/8/10.
2. Свободной памяти 512MB RAM.
3. Пространство жесткого диска: 100 MB.

### 4.1.3. Входные и выходные данные

Входной информацией для системы являются:

* база данных, содержащая статистические данные;
* результаты работы подсистемы кластерного анализа, сохраненные в соответствующие таблицы базы данных системы.

Выходной информацией является:

* протоколы автокоррекции;
* протоколы импутации.

## 4.2. Руководство системного программиста

### 4.2.1. Общие сведения о программе

Представленная система разработана с помощью системы управления базой данных Microsoft SQL Server 2012, среды программирования Microsoft Visual Studio Community 2019 и технологии доступа к базе данных ADO.NET.

### 4.2.2. Сообщения программисту

Сообщения программисту используются для контроля за корректностью работы проекта, прежде всего для проверки подключения источника данных к программному приложению комплекса. Проверка подключения к базе данных осуществляется в модуле «RCEIS», в случае появления ошибки выдаётся специальное сообщение (рис. 4.1, 4.2)



Рис. 4.1. Сообщение программисту о приостановке работы сервера



Рис. 4.2. Сообщение программисту о невозможности подключения к серверу

## 4.3. Руководство оператора

Использование программной системы предполагает выполнение следующих действий:

1. подключение к базе данных;
2. описание структуры хранимой информации – форм, таблиц, показателей, доменов показателей, регионов;
3. построение баз доноров и реципиентов;
4. создание и анализ правил редактирования для количественных и качественных показателей;
5. определение правил импутации для каждого показателя;
6. проведение процесса импутации данных.

При подключении пользователь должен ввести строку подключения в соответствующем окне, нажать на кнопку подключения на панели инструментов главной формы или выбрать пункт меню «Подключиться к RCEIS» или «Подключиться к ...».

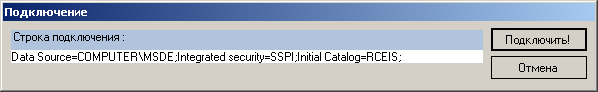


Рис. 4.3. Диалоговое окно соединения с базой данных

После подключения к базе данных становятся доступными элементы для просмотра информации и выполнения действий по импутации и автокоррекции.

На рис. 4.4. представлено главное окно системы RCIES, состоящее из:

* главного меню;
* панели инструментов;
* строки состояния;
* элемента управления с закладками «Данные», «Правила редактирования», «Правила импутации», «Автокоррекция и импутация», «Протокол».

На закладке «Данные» отображается информация о структуре исходных данных: переписных листах, таблицах, показателях. На данной закладке также отображается общая информация по выбранному типу переписного листа и по конкретной записи.

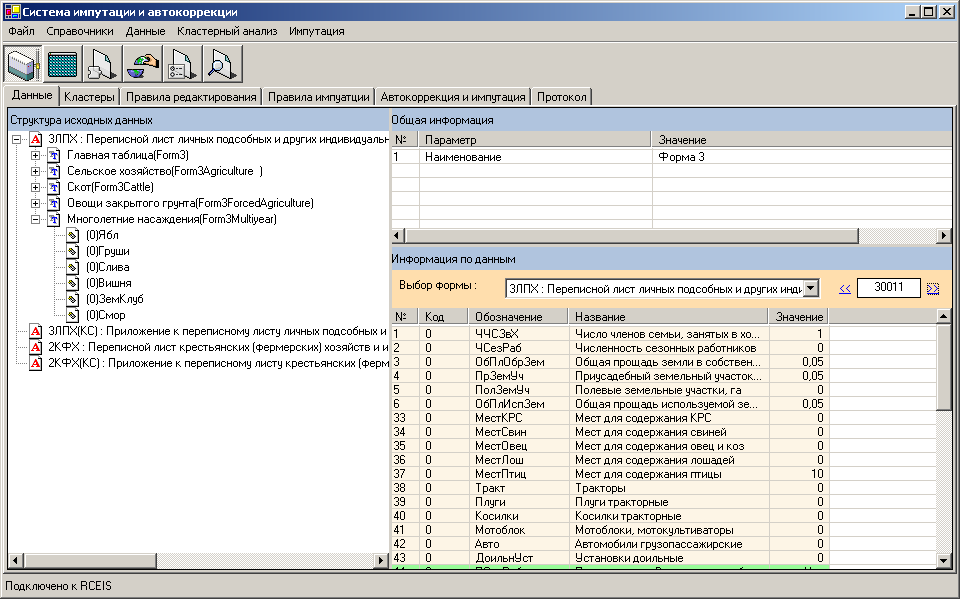


Рис. 4.4. Главное окно системы RCIES. Закладка «Данные»

В системе хранится рад справочников, к некоторым из них (справочник регионов и справочник доменов показателей) пользователь имеет доступ через главное меню системы.

На рис. 4.5 и рис. 4.6 представлены окна для просмотра информации о доменах показателей и регионах соответственно.

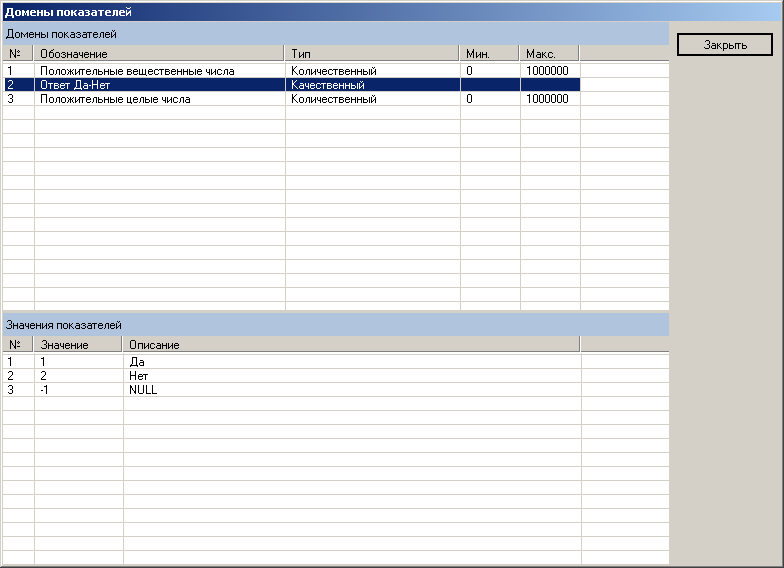


Рис. 4.5. Окно просмотра информации о доменах показателей и из значений

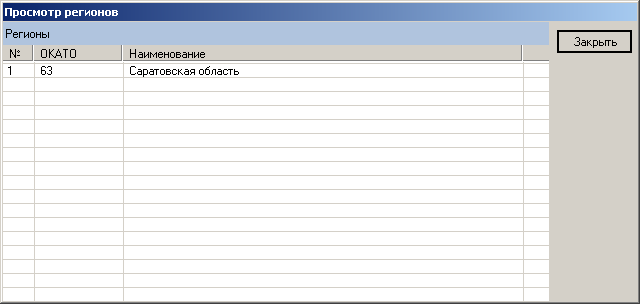


Рис. 4.6. Окно просмотра информации о регионах

На рис. 4.7 представлена закладка «Правила редактирования», позволяющая просмотреть информацию о правилах редактирования для конкретного типа переписного листа. Переключатель «Отображать правила редактирования» позволяет выбрать тип отображаемых правил редактирования: правила редактирования количественных или качественных показателей.

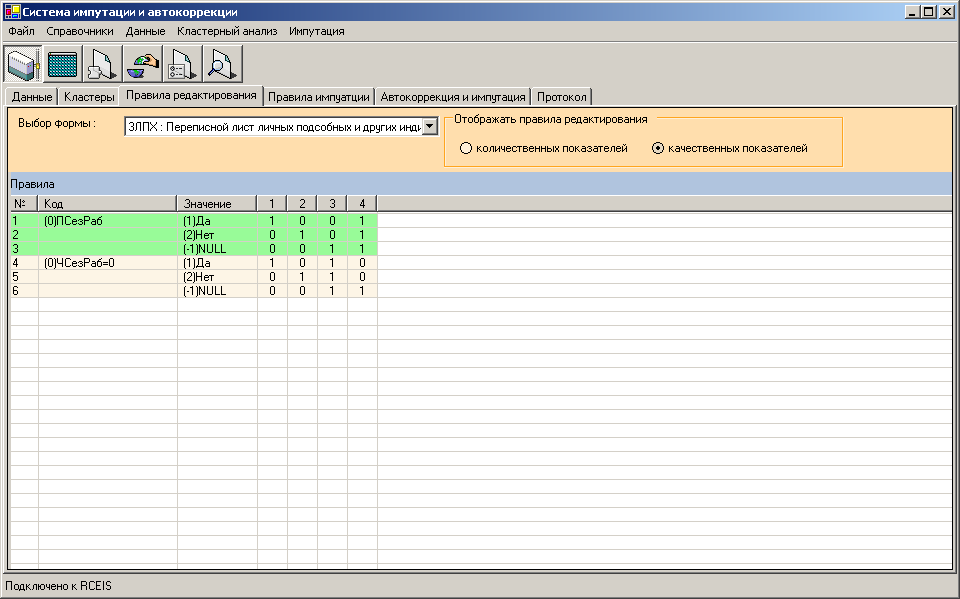


Рис. 4.7 – Главное окно системы. Закладка «Правила редактирования»

На рис. 4.8 представлена закладка «Правила импутации», позволяющая просмотреть информацию о правилах импутации для каждого показателя конкретного типа переписного листа.

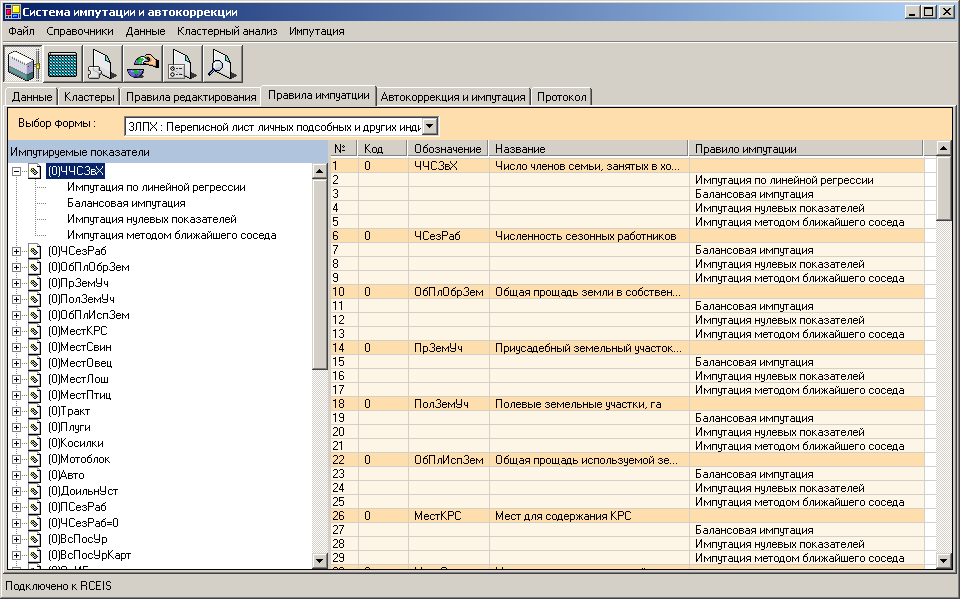


Рис. 4.8 – Главное окно системы RCIES. Закладка «Правила импутации»

На рис. 4.9 представлена закладка «Автокоррекция и импутация», обеспечивающая доступ к информации о процессе импутации и автокоррекции. Для выполнения данного этапа достаточно выбрать запись и нажать на кнопку импутации на панели инструментов. Система произведет следующие операции:

* будет выполнена проверка записи на соответствие правилам редактирования количественных и качественных показателей и выведена соответствующая информация (рис. 4.10);
* будет локализована ошибка, то есть определено минимальное множество показателей, значения которых должны быть импутированы (рис. 4.11);
* будет проведена идентификация кластера, ближайшего к данной записи и осуществлен поиск ближайшего донора в данном кластере (рис. 4.12);
* будет проведена импутация значений показателей, требующих импутации (рис. 4.13).

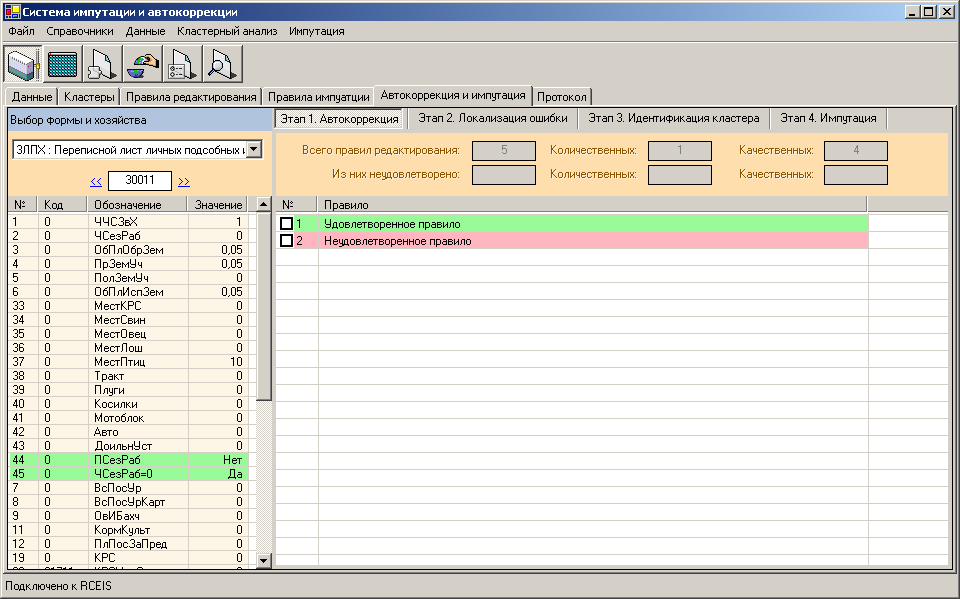


Рис. 5.10 – Автокоррекция и импутация/Этап1. Автокоррекция

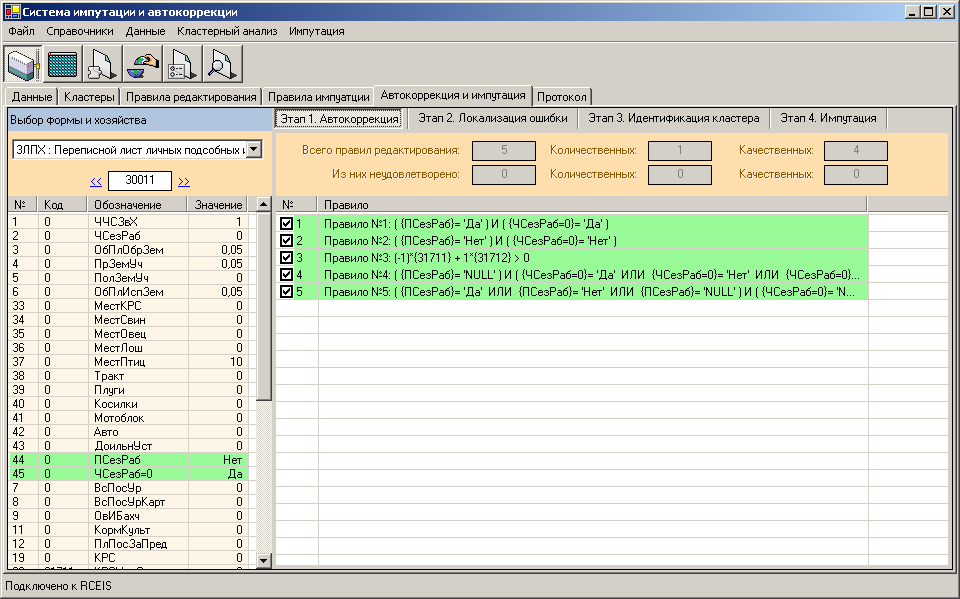


Рис. 4.11 – Этап автокоррекции завершен

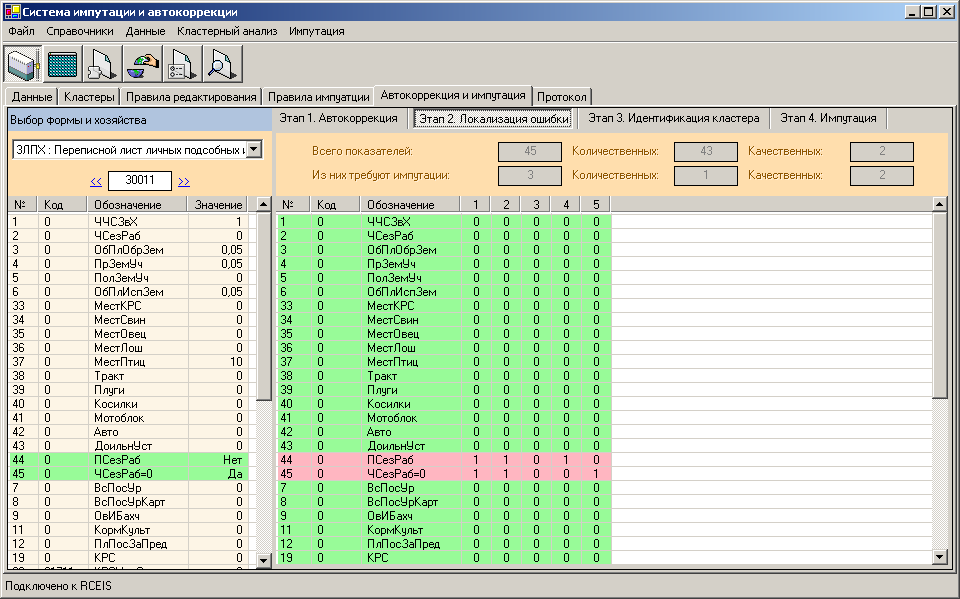


Рис. 4.12 – Этап локализации ошибки завершен

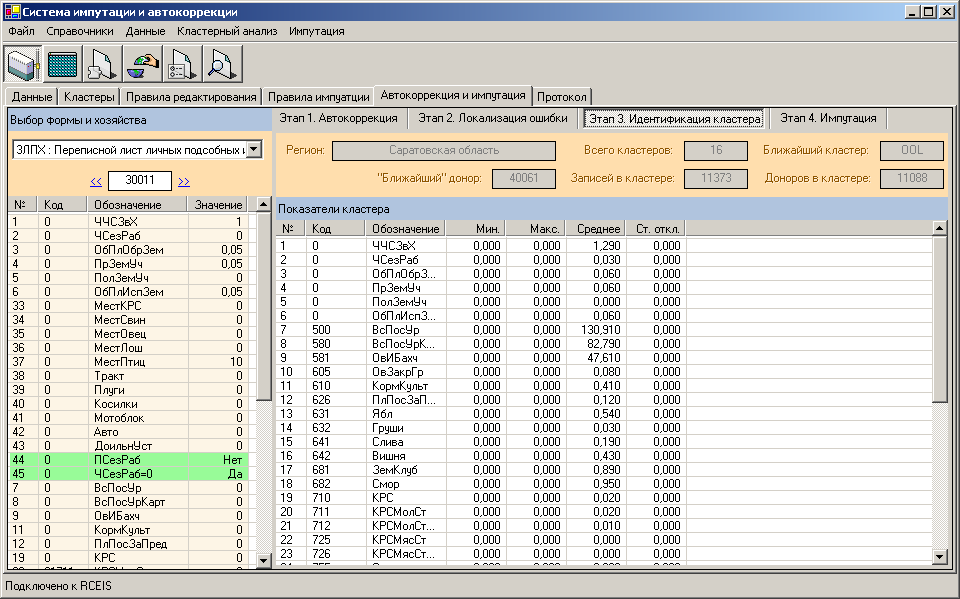


Рис. 4.13 – Этап идентификации кластера и поиск ближайшего соседа завершен

На закладке «Этап 4. Импутация» отображаются исходные и импутированные значения показателей выбранного хозяйства, а также примененные правила (рис. 4.14). Измененные показатели выделяются цветом.

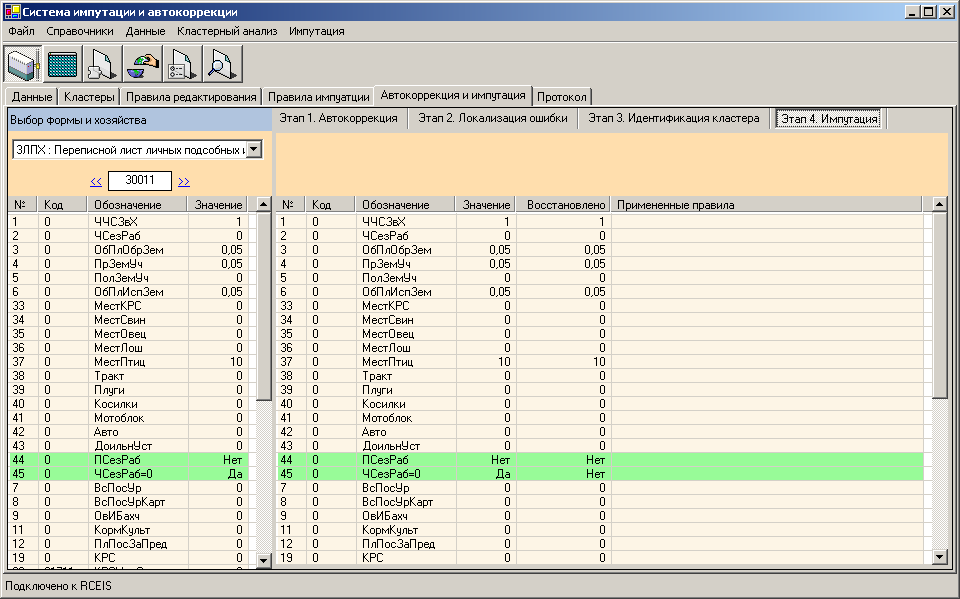


Рис. 4.14 – Этап импутации ошибки завершен

На рис. 4.15 представлена закладка «Протокол» для просмотра следующих сведений из протокола автокоррекции и импутации:

* дата выполнения импутации;
* форма (тип переписного листа);
* код хозяйства;
* импутируемый показатель;
* старое и новое значение показателя;
* примененные правила.

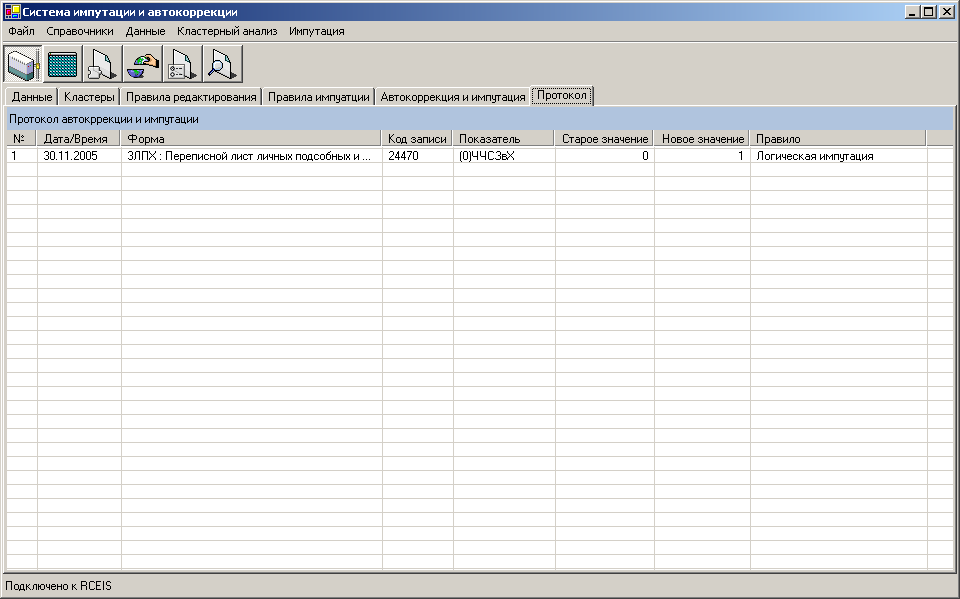


Рис. 4.15 – Окно просмотра протокола импутации

# ЗАКЛЮЧЕНИЕ

В рамках представленного дипломного проекта разработана система автоматизированной импутации данных на основе алгоритма Филледжи-Хольта.

Разработанная система тестирования комплекса позволила произвести полную проверку проекта с выявлением и исправлением всех скрытых дефектов, возникающих в процессе длительной эксплуатации.

В дальнейшем предлагаются следующие мероприятия по дополнению функционального назначения проекта:

* интерактивная коррекция данных;
* проверка правил редактирования на полноту и непротиворечивость;
* использование статистики Хидироглы-Бертелота для определения резко выделяющихся хозяйств;
* реализация случайного выбора для метода донора;
* реализация различных методов оценок (тренды, регресии, пропорциональные преобразования и т.д.);
* создание обучающего модуля.

# БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Winkler W. E. (1999): State of Statistical Data Editing and Current Research Problem. UN/ECE Work Session on Statistical Data Editing, Rome.
2. Cotton, C. (1993). “Functional Description of the Generalized Edit and Imputation System,” Statistics Canada Technical Report.
3. Draper, L. and Winkler, W. (1997). “Balancing and Ratio Editing With the New SPEER System,” Proceedings of the Section on Survey Research Methods, American Statistical Association, pp. 81-89.
4. "Functional Description of the Generalized Edit and Imputation System." Business Survey Methods Division, Statistics Canada. July 25, 1991. Revised May 31, 1999.
5. Poirier C. (1999): A Functional Evaluation of Edit and Imputation Tools. UN/ECE Work Session on Statistical Data Editing, Rome.
6. Todaro, T. (1999). “Evaluation of the AGGIES Automated Edit and Imputation System,” NASS Research Report RD 99-01, Washington, DC: National Agricultural Statistics Service, United States Department of Agriculture.
7. Waal, T. de and H. Wings, 1999. From CherryPi to SLICE. Report, Statistics Netherlands.
8. Fellegi, I. and Holt, D. (1976). “A Systematic Approach to Automatic Edit and Imputation,” Journal of the American Statistical Association, No. 71, pp. 17-35.
9. Черникова Н.В. (1964). Алгоритм поиска общей формулы неотрицательных решений системы линейных уравнений. Журнал вычислительной математики и математической физики № 4, 151-158.
10. Черникова Н.В. (1965). Алгоритм поиска общей формулы неотрицательного решения системы линейных неравенств. Журнал вычислительной математики и математической физики № 5, с. 228-233.
11. Whitridge P., Bernier J. (1999): The Impact of Editing on Data Quality. UN/ECE Work Session on Statistical Data Editing, Rome.
12. West S.A., Butani S., Witt M Alternative Imputation Methods for Wage Data. U.S. Bureau of Labor Statistics.
13. West S.A., Kratzke D., Robertson K. Alternative Imputation Procedures for Item Non-response New Establishments in the Universe. U.S. Bureau of Labor Statistics.
14. Rubin, D.B. (1978). "Multiple imputations in Sample Surveys - A Phenomenological Bayesian Approach to Nonresponse". Proceedings of the Section on Survey Research Methods, American Statistical Association.
15. Larsen B.S., Madsen B. (1999) Error Identification and Imputations with Neural Network. UN/ECE Work Session on Statistical Data Editing, Rome.
16. McDaniel, H. (1999). Multiple Imputation for Missing Data Using the "SOLAS for MissingData Analysis" Software Application. UN/ECE Work Session on Statistical Data Editing, Rome.
17. Cruddas M., Chambers R. (1997): Neural Network Imputation — Statistical Evaluation. UN/ECE Work Session on Statistical Data Editing, Prague.
18. Nordbotten, S. (1995): "Editing Statistical Records by Neural Networks". Journal of Official Statistics, Vol. 11, No. 4.
19. Nordbotten, S. (1996): "Neural Network Imputation Applied to the Norwegian 1990 Population Census Data". Journal of Official Statistics, Vol. 12, No. 4.

# ПРИЛОЖЕНИЕ А. ЛИСТИНГ ПРОГРАММНЫХ МОДУЛЕЙ

***Листинг модуля управления процессом импутации ControlImpute.cs***

using System;

using System.Data;

using System.Data.SqlClient;

namespace RCEIS

{

public class ControlDonorRecipient

{

public long DataTotal;

public long DataDonor;

public long DataRecipient;

private double [,] data;

private double [,] donors;

private bool [] donor;

private bool [] recipient;

static Questionarie qst;

static Region region;

public static double [,] LoadData(SqlConnection conn, Questionarie \_qst, Region \_region, bool is\_donor, bool is\_recipient, long id\_cluster)

{

qst = \_qst;

region = \_region;

string formname = qst.tableCollection[0].Name.Trim();

int d = is\_donor?1:0, r=is\_recipient?1:0;

string sqlText = "select IValues\_" + formname + ".\* FROM";

sqlText += " ClusterValues INNER JOIN [Values] ON ClusterValues.id\_value = [Values].id\_value INNER JOIN IValues\_"+formname+" ON [Values].id\_value = IValues\_"+formname+".FormID";

sqlText += " WHERE left(OKATO, 2)='"+region.OKATO.Trim()+"' AND is\_donor=" + d.ToString() + " AND is\_recipient = " + r.ToString() + " AND id\_cluster = " + id\_cluster.ToString();

SqlDataAdapter da = new SqlDataAdapter(sqlText, conn);

DataSet ds = new DataSet();

da.Fill(ds);

return KMeans.KMeans.ConvertDataTableToArray(ds.Tables[0]);

}

public void SplitData(SqlConnection conn, Questionarie \_qst, Region \_region, double alpha)

{

qst = \_qst;

region = \_region;

string sqlText = "select \* from IValues\_" + qst.tableCollection[0].Name.Trim()+" where left(OKATO, 2)='"+region.OKATO.Trim()+"'";

SqlDataAdapter da = new SqlDataAdapter(sqlText, conn);

DataSet ds = new DataSet();

da.Fill(ds);

data = KMeans.KMeans.ConvertDataTableToArray(ds.Tables[0]);

ColumnCollection cc = qst.GetColumnCollection();

long c\_count = cc.Count + 2; //первые два поля - formID и OKATO

double [] v\_min = new double[ c\_count ];

double [] v\_max = new double[ c\_count ];

for(int i=2; i<c\_count; i++)

{

double avg = Maths.Average(data, i);

double dev = Maths.StandardDeviation(data, i);

v\_min[ i ] = avg - alpha \* dev;

v\_max[ i ] = avg + alpha \* dev;

}

long v\_count = data.GetUpperBound(0) + 1;

donor = new bool[v\_count];

recipient = new bool[v\_count];

long nd\_count = 0;

long rc\_count = 0;

for(int i=0; i<v\_count; i++)

{

donor[ i ] = true;

for(int j=2; j<c\_count; j++)

{

if (cc[j-2].Domain.DomainType == DomainType.QualitativeDomain)

continue;

if ( data[ i, j ] == -1.0)

{

recipient[ i ] = true;

rc\_count++;

break;

}

if ( (data[ i, j ] < v\_min[ j ]) || (data[ i, j ] > v\_max[ j ]) )

{

donor[ i ] = false;

nd\_count++;

break;

}

}

}

DataTotal = v\_count;

DataDonor = v\_count - nd\_count - rc\_count;

DataRecipient = rc\_count;

}

private void SaveSplit(SqlConnection conn, long id\_value, long id\_form, long id\_region, bool is\_donor, bool is\_recipient)

{

SqlCommand cmd = new SqlCommand("sp\_updateValue", conn);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.Add("@id\_value", SqlDbType.BigInt);

cmd.Parameters.Add("@id\_form", SqlDbType.BigInt);

cmd.Parameters.Add("@id\_region", SqlDbType.BigInt);

cmd.Parameters.Add("@is\_donor", SqlDbType.Bit);

cmd.Parameters.Add("@is\_recipient", SqlDbType.Bit);

cmd.Parameters["@id\_value"].Value = id\_value;

cmd.Parameters["@id\_form"].Value = id\_form;

cmd.Parameters["@id\_region"].Value = id\_region;

cmd.Parameters["@is\_donor"].Value = is\_donor;

cmd.Parameters["@is\_recipient"].Value = is\_recipient;

try

{

cmd.ExecuteNonQuery();

}

catch(System.Data.SqlClient.SqlException ex)

{

}

catch(System.Exception ex)

{

}

}

public void SaveSplit(SqlConnection conn)

{

for(int i=0; i<DataTotal; i++)

{

long id\_value = (long)data[i, 0];

SaveSplit(conn, id\_value, qst.ID, region.ID, donor[i], recipient[i]);

}

}

}

public class ControlImpute

{

SqlConnection conn;

public Questionarie qst;

public Region region;

public Value val;

public Value ngb;

public Value corrected;

private bool [,] failEditMatrix;

public double [,] all\_data;

ColumnCollection fcColl;

public EditRuleCollection failedEdits;

public EditRuleCollection failedQntEdits;

public EditRuleCollection failedQltEdits;

public EditRuleCollection satisfiedEdits;

public ClusterSplitting clusterSplitting;

public Cluster nearestCluster;

public long totalValuesCount;

public long donorValuesCount;

public ColumnCollection FailedColumnCollection

{

get {return fcColl;}

}

public ControlImpute(SqlConnection \_conn, Questionarie \_qst)

{

conn = \_conn;

qst = \_qst;

failedEdits = new EditRuleCollection();

failedQntEdits = new EditRuleCollection();

failedQltEdits = new EditRuleCollection();

satisfiedEdits = new EditRuleCollection();

fcColl = new ColumnCollection ();

all\_data = null;

}

public void LoadValue(SqlConnection conn, long id, RegionCollection rc)

{

val = new Value(qst);

val.Load(conn, id, qst);

region = rc.FindByOKATO(val.OKATO);

}

public void CheckEdits()

{

bool [][] qlt = val.GetQLT(qst);

double [] qnt = val.GetQNT(qst);

bool check;

foreach(EditRule er in qst.editRuleCollection)

{

if (er.RuleType == RuleTypes.Qualitative)

{

Edits.QLT\_EditRule qlt\_er = er.GetQLT(qst);

check = qlt\_er.CheckValue(qlt);

if (!check)

{

failedQltEdits.Add(er);

failedEdits.Add(er);

}

}

else

{

Edits.QNT\_EditRule qnt\_er = er.GetQNT(qst);

check = qnt\_er.CheckValue(qnt);

if (!check)

{

failedQntEdits.Add(er);

failedEdits.Add(er);

}

}

if (check)

satisfiedEdits.Add(er);

}

}

public void LocaliseError()

{

EditRuleCollection erColl = failedEdits /\*satisfiedEdits\*/;

if (erColl.Count == 0) return;

ColumnCollection icColl = qst.GetColumnCollection();

long e\_count = erColl.Count;

long c\_count = icColl.Count;

failEditMatrix = new bool[ c\_count, e\_count ];

long i = 0;

foreach(EditRule rule in erColl)

{

long j = 0;

foreach(Column column in icColl)

{

failEditMatrix [ j , i ] = rule.IncludeExplicitly( column );

j ++;

}

i ++;

}

fcColl = new ColumnCollection();

do

{

//Ищем показатель с максимальной характеристикой

double max\_column = 0.0;

int idx\_column = 0;

int cur\_idx = 0;

foreach(Column column in icColl)

{

double cur\_column = 0;

foreach(EditRule rule in erColl)

{

if (rule.IncludeExplicitly( column ))

cur\_column += column.Weight;

}

if ( cur\_column >= max\_column )

{

max\_column = cur\_column;

idx\_column = cur\_idx;

}

cur\_idx ++;

}

//Удаляем правила, в которые показатель входит

int k = 0;

while (k < erColl.Count )

{

EditRule rule = erColl[ k ];

Column column = icColl[idx\_column];

if ( rule.IncludeExplicitly(column) )

erColl.RemoveAt( k );

else

k ++;

}

//Добавляем его в fcColl, удаляем из icColl

fcColl.Add(icColl[idx\_column]);

icColl.RemoveAt(idx\_column);

//Пока не будет пустой коллекция правил

} while(erColl.Count != 0);

}

public bool[] GetFailedQltColumnsIndex()

{

ColumnCollection cc = qst.GetColumnCollection(DomainType.QualitativeDomain);

bool [] beImputed = new bool[cc.Count];

int i = 0;

foreach(Column column in cc)

{

if (fcColl.FindByID(column.ID) == null)

beImputed[ i ] = false;

else

beImputed[ i ] = true;

i++;

}

return beImputed;

}

public void FindNearestCluster()

{

ColumnCollection cc = qst.GetColumnCollection(DomainType.QuantitiveDomain);

double min\_distance = Double.MaxValue;

int idx = -1;

int cur = 0;

double [] val\_data = val.GetQNT(qst);

foreach(Cluster cluster in clusterSplitting.clusterCollection)

{

//вычислим расстояние от центра кластера до точки

double cur\_distance = 0;

foreach(ClusterParam cp in cluster.clusterParamCollection)

{

double mean = cp.Mean;

double data = val\_data[ cc.GetIndexByID(cp.Column.ID) ];

double weight = clusterSplitting.GetWeight(cp.Column);

if ( weight == 0) weight = 1;

cur\_distance += Math.Pow( mean-data , 2.0 ) / weight;

}

if ( cur\_distance < min\_distance )

{

min\_distance = cur\_distance;

idx = cur;

}

cur++;

}

nearestCluster = clusterSplitting.clusterCollection[idx];

}

public void FindNearestValue(ref double [,] all\_data)

{

ColumnCollection cc = qst.GetColumnCollection();

ColumnCollection cc\_qnt = qst.GetColumnCollection(DomainType.QuantitiveDomain);

Edits.QLT\_EditRulesCollection qlt\_rc = failedQltEdits.GetQLT\_EditRuleCollection(qst);

bool [][] filter = qlt\_rc.GetFilter(qst, GetFailedQltColumnsIndex() ,val);

double min\_distance = Double.MaxValue;

int idx = -1;

double [] val\_data = val.GetQNT(qst);

if (all\_data == null)

this.all\_data = ControlDonorRecipient.LoadData(conn, qst, region, true, false, nearestCluster.ID);

else

this.all\_data = all\_data/\*ControlDonorRecipient.LoadData(conn, qst, region, true, false, nearestCluster.ID)\*/;

long v\_count = all\_data.GetUpperBound(0)+1;

long c\_count = val\_data.GetUpperBound(0)+1;

totalValuesCount = nearestCluster.RecordCount;

donorValuesCount = v\_count;

//Для тестирования и восстановления кач. показателей

if (v\_count == 0)

{

all\_data = ControlDonorRecipient.LoadData(conn, qst, region, false, false, nearestCluster.ID);

v\_count = all\_data.GetUpperBound(0)+1;

}

if (v\_count != 0)

{

double [] max\_values = new double[all\_data.GetUpperBound(1)+1];

for(int i=2; i<all\_data.GetUpperBound(1)+1; i++)

{

max\_values[ i ] = Maths.Max(all\_data, i);

if (max\_values[i] == 0)

max\_values[i] = 1;

}

for (int i=0; i<v\_count; i++)

{

Value newVal = new Value(qst);

newVal.Load(qst, (long)all\_data[i, 0], all\_data[i, 1].ToString("F0"), ref all\_data, i);

if (newVal.ID == val.ID) //для тестирования, чтоб не находил тоже хозяйство

continue;

bool f = qlt\_rc.ApllyFilter(qst, filter, newVal);

if (!f) continue;

double cur\_distance = 0;

for(int j=0; j<c\_count; j++)

{

if (cc[j].Domain.DomainType == DomainType.QualitativeDomain)

continue;

long idx\_col = cc[j].Number + 1;

cur\_distance += Math.Pow((val\_data[j] - all\_data[ i, idx\_col])/ max\_values[ idx\_col ] , 2);

}

if ( cur\_distance < min\_distance )

{

min\_distance = cur\_distance;

idx = i;

}

}

ngb = new Value(qst);

ngb.Load(qst, (long)all\_data[idx, 0], all\_data[idx, 1].ToString("F0"), ref all\_data, idx);

}

else

{

ngb = null;

}

corrected = new Value(qst);

/\*corrected.Load(conn, val.ID, qst);\*/

int n = qst.GetColumnCollection().Count;

for (int i=0; i<n; i++)

corrected[i] = (object)(double)val[i];

/\*corrected.Load(qst, val.ID, val.OKATO, ref all\_data);\*/

}

public void DoImpute(DomainType dt)

{

long idx = 0;

if (dt == DomainType.QualitativeDomain)

{

//проводим импутацию качественных показателей

foreach(Column column in fcColl.GetColumnCollection(DomainType.QualitativeDomain))

{

foreach(ImputeRule ir in column.imputeRuleCollection)

{

//проводим логическую импутацию, остальные - невозможны

if (ir.ImputeType == ImputeTypes.Logical)

{

idx = qst.GetColumnCollection().GetIndexByID(column.ID);

int num = (int)idx;

if (ngb != null)

corrected[num] = ngb[num];

else

corrected[num] = null;

break;

}

}

}

}

else

{

//проводим импутацию качественных показателей

foreach(Column column in fcColl.GetColumnCollection(DomainType.QuantitiveDomain))

{

idx = qst.GetColumnCollection().GetIndexByID(column.ID);

int num = (int)idx;

foreach(ImputeRule ir in column.imputeRuleCollection)

{

//Если параметр можно восстановить по регрессии

if (ir.ImputeType == ImputeTypes.Statistical)

{

long id\_col\_x = qst.GetColumnCollection().GetIndexByID(Int32.Parse(ir.Parameter));

//Если связанную колонку не требуется импутировать

if (fcColl.FindByID(Int32.Parse(ir.Parameter)) != null)

{

double a, b;

Maths.LinearRegression(this.all\_data, (int)id\_col\_x, (int)idx, out a, out b);

corrected[num] = a \* (double)val[(int)id\_col\_x] + b;

break;

}

}

if (ir.ImputeType == ImputeTypes.HotDeck)

{

//восстанавливаем по методу ближайшего соседа

if (ngb != null)

corrected[num] = ngb[num];

else

corrected[num] = null;

}

}

}

}

}

}

}

***Листинг модуля работы с регионами Regions.cs***

using System;

using System.Data;

using System.Data.SqlClient;

using System.Text;

using System.Configuration;

using System.Windows.Forms;

namespace RCEIS

{

/// <summary>

/// </summary>

[Serializable]

public class Region

{

private long id;

private string name;

private string okato;

public long ID

{

get {return id;}

set {id = value;}

}

public string Name

{

get {return name;}

set {name = value;}

}

public string OKATO

{

get {return okato;}

set {okato = value;}

}

public override string ToString()

{

return okato + " : " + name;

}

}

/// <summary>

/// A collection of Cluster objects or Clusters

/// </summary>

[Serializable]

public class RegionCollection : System.Collections.CollectionBase

{

/// <summary>

///

/// </summary>

/// <param name="region"></param>

public virtual void Add (Region region)

{

this.List.Add(region);

}

/// <summary>

///

/// </summary>

public virtual Region this[int Index]

{

get

{

return (Region)this.List[Index];

}

}

public Region FindByID(long id)

{

foreach(Region region in this)

{

if (region.ID == id)

return region;

}

return null;

}

public Region FindByOKATO(string OKATO)

{

foreach(Region region in this)

{

if (region.OKATO == OKATO)

return region;

}

return null;

}

public void Load(SqlConnection conn)

{

Clear();

SqlCommand cmd = new SqlCommand("sp\_getRegionList", conn);

cmd.CommandType = CommandType.StoredProcedure;

SqlDataReader dr = cmd.ExecuteReader();

while( dr.Read() )

{

Region region = new Region();

region.ID = dr.GetInt64(0);

region.Name = dr.GetString(1);

region.OKATO= dr.GetString(2);

Add( region );

}

dr.Close();

}

public void Insert(SqlConnection conn, Region region)

{

SqlCommand cmd = new SqlCommand("sp\_addRegion", conn);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.Add("@name", SqlDbType.VarChar);

cmd.Parameters.Add("@okato", SqlDbType.VarChar);

cmd.Parameters["@name"].Value = region.Name;

cmd.Parameters["@okato"].Value = region.OKATO;

try

{

cmd.ExecuteNonQuery();

}

catch(System.Data.SqlClient.SqlException ex)

{

MessageBox.Show(ex.Message);

}

catch(System.Exception ex)

{

MessageBox.Show(ex.Message);

}

}

public void Update(SqlConnection conn, Region region)

{

SqlCommand cmd = new SqlCommand("sp\_updateRegion", conn);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.Add("@name", SqlDbType.VarChar);

cmd.Parameters.Add("@okato", SqlDbType.VarChar);

cmd.Parameters["@name"].Value = region.Name;

cmd.Parameters["@okato"].Value = region.OKATO;

try

{

cmd.ExecuteNonQuery();

}

catch(System.Data.SqlClient.SqlException ex)

{

MessageBox.Show(ex.Message);

}

catch(System.Exception ex)

{

MessageBox.Show(ex.Message);

}

}

public void Delete(SqlConnection conn, Region region)

{

SqlCommand cmd = new SqlCommand("sp\_deleteRegion", conn);

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.Add("@okato", SqlDbType.VarChar);

cmd.Parameters["@okato"].Value = region.OKATO;

try

{

cmd.ExecuteNonQuery();

}

catch(System.Data.SqlClient.SqlException ex)

{

MessageBox.Show(ex.Message);

}

catch(System.Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

}

***Листинг главного модуля проекта RCEIS.cs***

using System;

using System.Text;

using System.Configuration;

using System.Windows.Forms;

using System.Data.SqlClient;

namespace RCEIS

{

public struct ColumnValue

{

public Column column;

public DomainValue value;

}

public class RCEIS

{

private bool is\_connected;

public SqlConnection conn;

public RegionCollection regionCollection;

public QuestionarieCollection questionarieCollection;

public ClusterSplittingCollection csCollection;

public DomainCollection domainCollection;

public ProtocolRecordCollection protocolCollection;

public long failedAllEditCount;

public long failedQntEditCount;

public long failedQltEditCount;

public long failedAllColumnCount;

public long failedQntColumnCount;

public long failedQltColumnCount;

public Region region;

public ControlImpute controlImpute;

public RCEIS()

{

is\_connected = false;

regionCollection = new RegionCollection();

questionarieCollection = new QuestionarieCollection();

csCollection = new ClusterSplittingCollection();

domainCollection = new DomainCollection();

protocolCollection = new ProtocolRecordCollection();

}

public bool Connected

{

get

{

return is\_connected;

}

set

{

is\_connected = value;

}

}

public void Connect(string cstr)

{

StringBuilder result = new StringBuilder();

conn = new SqlConnection(cstr);

try

{

conn.Open();

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

return;

}

is\_connected = true;

}

public void Disconnect()

{

if (is\_connected)

{

is\_connected = false;

conn.Close();

}

}

public void LoadRegionCollection()

{

if (is\_connected) regionCollection.Load(conn);

}

public void LoadQuestionarieCollection()

{

if (is\_connected) questionarieCollection.Load(conn, domainCollection);

}

public void LoadClusterSplittingCollection()

{

if (is\_connected) csCollection.Load(conn, questionarieCollection);

}

public void LoadDomainCollection()

{

if (is\_connected) domainCollection.Load(conn);

}

public void LoadProtocolCollection()

{

if (is\_connected) protocolCollection.Load(conn, questionarieCollection);

}

public void UpdateProtocol(ListView lv)

{

lv.Items.Clear();

int i = 0;

foreach(ProtocolRecord pr in protocolCollection)

{

i++;

ListViewItem li = lv.Items.Add(i.ToString());

li.SubItems.Add(pr.Writed.ToShortDateString());

li.SubItems.Add(pr.Questionarie.ToString());

li.SubItems.Add(pr.ID\_Value.ToString());

li.SubItems.Add(pr.Column.ToString());

li.SubItems.Add(pr.OldValue.ToString());

li.SubItems.Add(pr.NewValue.ToString());

li.SubItems.Add(ImputeRule.ToString((ImputeTypes)pr.ID\_RuleType));

li.Tag = pr;

}

}

public void UpdateRegion(ComboBox cb)

{

cb.Items.Clear();

foreach(Region region in regionCollection)

{

cb.Items.Add(region);

}

}

public void UpdateQuestionarie(ComboBox cb)

{

cb.Items.Clear();

foreach(Questionarie qst in questionarieCollection)

{

cb.Items.Add(qst);

}

}

public void UpdateQuestionarie(TreeView tv)

{

LoadQuestionarieCollection();

tv.Nodes.Clear();

foreach(Questionarie qst in questionarieCollection)

{

TreeNode tn = tv.Nodes.Add(qst.ToString());

tn.ImageIndex = 0;

tn.SelectedImageIndex = 0;

tn.Tag = qst;

foreach(Table table in qst.tableCollection)

{

TreeNode tn\_child = tn.Nodes.Add(table.ToString());

tn\_child.ImageIndex = 1;

tn\_child.SelectedImageIndex = 1;

tn\_child.Tag = table;

foreach(Column column in table.columnCollection)

{

TreeNode tn\_child\_2 = tn\_child.Nodes.Add(column.ToString());

tn\_child\_2.ImageIndex = 2;

tn\_child\_2.SelectedImageIndex = 2;

tn\_child\_2.Tag = column;

}

}

}

}

public void UpdateClustersSplitting(TreeView tv, long id\_region)

{

LoadClusterSplittingCollection();

tv.Nodes.Clear();

foreach(ClusterSplitting cs in csCollection)

{

if (cs.ID\_Region == id\_region)

{

Questionarie qst = questionarieCollection.FindByID(cs.ID\_Form);

TreeNode tn = tv.Nodes.Add(qst.ToString());

tn.ImageIndex = 0;

tn.SelectedImageIndex = 0;

tn.Tag = cs;

TreeNode tn\_child1 = tn.Nodes.Add("Показатели");

tn\_child1.ImageIndex = 1;

tn\_child1.SelectedImageIndex = 1;

tn\_child1.Tag = cs.columnCollection;

foreach(Column column in cs.columnCollection)

{

TreeNode tn\_child3 = tn\_child1.Nodes.Add(column.ToString());

tn\_child3.ImageIndex = 2;

tn\_child3.SelectedImageIndex = 2;

tn\_child3.Tag = column;

}

TreeNode tn\_child2 = tn.Nodes.Add("Кластеры");

tn\_child2.ImageIndex = 1;

tn\_child2.SelectedImageIndex = 1;

tn\_child2.Tag = cs.clusterCollection;

foreach(Cluster cluster in cs.clusterCollection)

{

TreeNode tn\_child3 = tn\_child2.Nodes.Add(cluster.ToString() + "[ " + cluster.RecordCount.ToString("D5") + " ]");

tn\_child3.ImageIndex = 2;

tn\_child3.SelectedImageIndex = 2;

tn\_child3.Tag = cluster;

}

}

}

}

public void UpdateSplittingColumns(TreeView tv, ListView lv)

{

TreeNode node = tv.SelectedNode;

if (node != null)

{

if (node.Parent != null)

return;

ClusterSplitting cs = (ClusterSplitting)(node.Tag);

lv.Items.Clear();

foreach(Column column in cs.columnCollection)

{

ListViewItem li = lv.Items.Add(column.Number.ToString());

li.SubItems.Add(column.UniqueCode.ToString());

li.SubItems.Add(column.ShortName.ToString());

li.SubItems.Add(column.Title.ToString());

li.Tag = column;

}

}

}

public void UpdateClusterParams(ListView lvCluster, ListView lv)

{

foreach(ListViewItem liCluster in lvCluster.SelectedItems)

{

Cluster cluster = (Cluster)(liCluster.Tag);

UpdateClusterParams(cluster, lv);

}

}

public void UpdateClusterParams(Cluster cluster, ListView lv)

{

lv.Items.Clear();

int i=0;

foreach(ClusterParam param in cluster.clusterParamCollection)

{

i++;

ListViewItem li = lv.Items.Add(i.ToString());

li.SubItems.Add(param.column.Code.ToString());

li.SubItems.Add(param.column.ShortName);

li.SubItems.Add(param.MinValue.ToString("F3"));

li.SubItems.Add(param.MaxValue.ToString("F3"));

li.SubItems.Add(param.Mean.ToString("F3"));

li.SubItems.Add(param.StdDev.ToString("F3"));

li.Tag = param;

}

}

public void UpdateSplittingClusters(TreeView tv, ListView lv)

{

TreeNode node = tv.SelectedNode;

if (node != null)

{

if (node.Parent != null)

return;

ClusterSplitting cs = (ClusterSplitting)(node.Tag);

lv.Items.Clear();

foreach(Cluster cluster in cs.clusterCollection)

{

ListViewItem li = lv.Items.Add(cluster.Number.ToString());

li.SubItems.Add(cluster.Name);

li.SubItems.Add(cluster.RecordCount.ToString());

li.SubItems.Add(cluster.DonorCount.ToString());

li.Tag = cluster;

}

}

}

public void UpdateDomains(ListView lv)

{

LoadDomainCollection();

lv.Items.Clear();

int i = 0;

foreach(Domain domain in domainCollection)

{

i++;

ListViewItem li = lv.Items.Add(i.ToString());

li.SubItems.Add(domain.ToString());

if (domain.DomainType == DomainType.QuantitiveDomain)

{

li.SubItems.Add("Количественный");

li.SubItems.Add(domain.MinValue.ToString());

li.SubItems.Add(domain.MaxValue.ToString());

}

else

li.SubItems.Add("Качественный");

li.Tag = domain;

}

}

public void UpdateDomainValues(ListView lvDomain, ListView lv)

{

lv.Items.Clear();

foreach(ListViewItem lidomain in lvDomain.SelectedItems)

{

int i = 0;

Domain domain = (Domain)lidomain.Tag;

foreach(DomainValue domain\_value in domain.valueCollection)

{

i++;

ListViewItem li = lv.Items.Add(i.ToString());

li.SubItems.Add(domain\_value.Value.ToString());

li.SubItems.Add(domain\_value.Name);

li.Tag = domain\_value;

}

}

}

public void UpdateColumnValues(ListView lv, long id\_form, DomainType domain\_type)

{

//обновление списка показателей

lv.Items.Clear();

int i = 0;

bool first;

System.Drawing.Color color = System.Drawing.Color.PaleGreen;

Questionarie qst = questionarieCollection.FindByID(id\_form);

foreach(Column column in qst.GetColumnCollection())

{

if (color == System.Drawing.Color.PaleGreen)

color = System.Drawing.Color.OldLace;

else

color = System.Drawing.Color.PaleGreen;

if (column.Domain.DomainType == domain\_type)

{

if (column.Domain.DomainType == DomainType.QuantitiveDomain)

{

i++;

ListViewItem li = lv.Items.Add(i.ToString());

li.SubItems.Add(column.ToString());

li.SubItems.Add("");

li.BackColor = System.Drawing.Color.OldLace;

ColumnValue cv = new ColumnValue();

cv.column = column;

cv.value = null;

li.Tag = cv;

}

else

{

first = true;

foreach(DomainValue domain\_value in column.Domain.valueCollection)

{

i++;

ListViewItem li = lv.Items.Add(i.ToString());

if (!first)

li.SubItems.Add(" ");

else

li.SubItems.Add(column.ToString());

li.SubItems.Add(domain\_value.ToString());

li.BackColor = color;

ColumnValue cv = new ColumnValue();

cv.column = column;

cv.value = domain\_value;

li.Tag = cv;

first = false;

}

}

}

}

//обновление столбцов

int k = lv.Columns.Count - 3;

for(int j=1; j<=k; j++)

{

lv.Columns.RemoveAt(3);

}

k = 0;

foreach(EditRule rule in qst.editRuleCollection)

{

if (

(rule.RuleType == RuleTypes.Qualitative)&&(domain\_type == DomainType.QualitativeDomain)||

(rule.RuleType != RuleTypes.Qualitative)&&(domain\_type != DomainType.QualitativeDomain)

)

{

k++;

ColumnHeader ch = lv.Columns.Add(k.ToString(), 30, HorizontalAlignment.Center);

//добавляем для всех строк SubItems

foreach(ListViewItem lvItem in lv.Items)

{

ColumnValue cv = (ColumnValue)(lvItem.Tag);

lvItem.SubItems.Add(rule.GetCoeeficient(cv.column, cv.value).ToString());

}

}

}

}

public void UpdateQuestionarieColumns(TreeView tv, long id\_questionarie)

{

Questionarie qst = questionarieCollection.FindByID(id\_questionarie);

tv.Nodes.Clear();

ColumnCollection cc = qst.GetColumnCollection();

foreach(Column column in cc)

{

TreeNode tn\_child = tv.Nodes.Add(column.ToString());

tn\_child.ImageIndex = 2;

tn\_child.SelectedImageIndex = 2;

tn\_child.Tag = column;

foreach (ImputeRule ir in column.imputeRuleCollection)

{

TreeNode tn\_child\_1 = tn\_child.Nodes.Add(ir.ToString());

tn\_child\_1.ImageIndex = 3;

tn\_child\_1.SelectedImageIndex = 3;

tn\_child\_1.Tag = ir;

}

}

}

public void UpdateQuestionarieColumns(ListView lv, long id\_questionarie)

{

Questionarie qst = questionarieCollection.FindByID(id\_questionarie);

lv.Items.Clear();

ColumnCollection cc = qst.GetColumnCollection();

int i = 0;

foreach(Column column in cc)

{

ListViewItem li = lv.Items.Add((++i).ToString());

li.SubItems.Add(column.UniqueCode.ToString());

li.SubItems.Add(column.ShortName);

li.SubItems.Add(column.Title);

li.BackColor = System.Drawing.Color.NavajoWhite;

li.Tag = column;

foreach (ImputeRule ir in column.imputeRuleCollection)

{

ListViewItem li\_r = lv.Items.Add((++i).ToString());

li\_r.SubItems.Add("");

li\_r.SubItems.Add("");

li\_r.SubItems.Add("");

li\_r.SubItems.Add(ir.ToString());

li\_r.BackColor = System.Drawing.Color.OldLace;

li\_r.Tag = ir;

}

}

}

public void UpdateRegions(ListView lv)

{

lv.Items.Clear();

int i=0;

foreach(Region region in regionCollection)

{

i++;

ListViewItem li = lv.Items.Add(i.ToString());

li.SubItems.Add(region.OKATO);

li.SubItems.Add(region.Name);

li.Tag = region;

}

}

public void UpdateQuestionarieColumns(ListView lv, long id\_form, bool applyformat)

{

Questionarie qst = this.questionarieCollection.FindByID(id\_form);

ColumnCollection cc = qst.GetColumnCollection();

lv.Items.Clear();

foreach(Column column in cc)

{

ListViewItem li = lv.Items.Add(column.Number.ToString());

li.SubItems.Add(column.UniqueCode.ToString());

li.SubItems.Add(column.ShortName.ToString());

li.SubItems.Add(column.Title.ToString());

li.SubItems.Add("");

if(applyformat)

switch(column.Domain.DomainType)

{

case DomainType.QualitativeDomain:

li.BackColor = System.Drawing.Color.PaleGreen;

break;

case DomainType.QuantitiveDomain:

li.BackColor = System.Drawing.Color.OldLace;

break;

}

li.Tag = column;

}

}

public void UpdateValueData(ListView lv, ComboBox cb, long id)

{

Questionarie qst = (Questionarie)cb.Items[cb.SelectedIndex];

Value v = new Value(qst);

v.Load(conn, id, qst);

int count = qst.GetColumnCollection().Count;

for(int i=0; i<count; i++)

{

lv.Items[i].SubItems[4].Text = v.ToString(qst, i);

}

region = regionCollection.FindByOKATO(v.OKATO);

}

public void DoOnlyImpute(ComboBox cb, long id, ListView lv, Value val, Region region, ref double[,] data)

{

Questionarie qst = (Questionarie)cb.Items[cb.SelectedIndex];

controlImpute = new ControlImpute(conn, qst);

controlImpute.val = val/\* LoadValue(conn, id, regionCollection)\*/;

controlImpute.region = region;

controlImpute.CheckEdits();

controlImpute.LocaliseError();

controlImpute.clusterSplitting = this.csCollection.Find(region.ID, controlImpute.qst.ID);

controlImpute.FindNearestCluster();

controlImpute.FindNearestValue(ref data);

controlImpute.DoImpute(DomainType.QualitativeDomain);

controlImpute.DoImpute(DomainType.QuantitiveDomain);

}

public void UpdateEditChecking(ComboBox cb, long id, ListView lv, Value val, Region region)

{

Questionarie qst = (Questionarie)cb.Items[cb.SelectedIndex];

controlImpute = new ControlImpute(conn, qst);

controlImpute.val = val/\* LoadValue(conn, id, regionCollection)\*/;

controlImpute.region = region;

controlImpute.CheckEdits();

failedQntEditCount = controlImpute.failedQntEdits.Count;

failedQltEditCount = controlImpute.failedQltEdits.Count;

failedAllEditCount = failedQntEditCount + failedQltEditCount;

lv.Items.Clear();

int i=0;

foreach(EditRule er in controlImpute.failedQntEdits)

{

ListViewItem li = lv.Items.Add((++i).ToString());

li.Checked = false;

li.SubItems.Add(er.ToDescriptionString(qst));

li.BackColor = System.Drawing.Color.LightPink;

}

foreach(EditRule er in controlImpute.failedQltEdits)

{

ListViewItem li = lv.Items.Add((++i).ToString());

li.Checked = false;

li.SubItems.Add(er.ToDescriptionString(qst));

li.BackColor = System.Drawing.Color.LightPink;

}

foreach(EditRule er in controlImpute.satisfiedEdits)

{

ListViewItem li = lv.Items.Add((++i).ToString());

li.Checked = true;

li.SubItems.Add(er.ToDescriptionString(qst));

li.BackColor = System.Drawing.Color.PaleGreen;

}

}

public void UpdateFailedEdits(ListView lv)

{

//обновление столбцов

int k = lv.Columns.Count - 4;

for(int j=1; j<=k; j++)

{

lv.Columns.RemoveAt(4);

}

foreach(ListViewItem lvItem in lv.Items)

{

lvItem.SubItems.RemoveAt(4);

}

k = 0;

foreach(EditRule rule in controlImpute.failedEdits)

{

k++;

ColumnHeader ch = lv.Columns.Add(k.ToString(), 30, HorizontalAlignment.Center);

//добавляем для всех строк SubItems

foreach(ListViewItem lvItem in lv.Items)

{

Column column = (Column)(lvItem.Tag);

if(rule.IncludeExplicitly(column))

{

lvItem.SubItems.Add("1");

}

else

lvItem.SubItems.Add("0");

}

}

}

public void UpdateLocalizedError(ListView lv)

{

//обновление столбцов

controlImpute.LocaliseError();

foreach(ListViewItem lvItem in lv.Items)

{

Column column = (Column)(lvItem.Tag);

if( controlImpute.FailedColumnCollection.FindByID(column.ID) != null )

{

if (column.Domain.DomainType == DomainType.QualitativeDomain)

failedQltColumnCount++;

else

failedQntColumnCount++;

lvItem.BackColor = System.Drawing.Color.LightPink;

}

else

{

lvItem.BackColor = System.Drawing.Color.PaleGreen;

}

}

failedAllColumnCount = failedQltColumnCount + failedQntColumnCount;

}

public void UpdateClusterIndentifiction(ListView lv, TextBox te1, TextBox te2, TextBox te3)

{

Region region = regionCollection.FindByOKATO(controlImpute.val.OKATO);

controlImpute.clusterSplitting = this.csCollection.Find(region.ID, controlImpute.qst.ID);

controlImpute.FindNearestCluster();

te1.Text = controlImpute.clusterSplitting.clusterCollection.Count.ToString();

te2.Text = controlImpute.nearestCluster.Name;

te3.Text = region.Name;

UpdateClusterParams(controlImpute.nearestCluster, lv);

}

public void GenerateAllImputeRules(long id\_form)

{

Questionarie qst = questionarieCollection.FindByID(id\_form);

foreach(Column column in qst.GetColumnCollection())

{

column.GenerateImputeRules(conn);

}

}

public void UpdateNearestValue(TextBox teTotal, TextBox teDonor, TextBox teNearestValue)

{

controlImpute.FindNearestValue(ref controlImpute.all\_data);

teTotal.Text = controlImpute.totalValuesCount.ToString();

teDonor.Text = controlImpute.donorValuesCount.ToString();

if (controlImpute.ngb != null)

teNearestValue.Text = controlImpute.ngb.ToString();

else

teNearestValue.Text = "-";

}

public void UpdateImputedValue(ListView lv)

{

controlImpute.DoImpute(DomainType.QualitativeDomain);

controlImpute.DoImpute(DomainType.QuantitiveDomain);

int i = 0;

foreach(ListViewItem li in lv.Items)

{

while(li.SubItems.Count > 5)

li.SubItems.RemoveAt(5);

li.SubItems[4].Text = (controlImpute.val.ToString(controlImpute.qst, i));

li.SubItems.Add(controlImpute.corrected.ToString(controlImpute.qst, i++));

}

}

public void UpdateQuestionarieInfo(TreeView tv, ListView lv)

{

TreeNode ti = tv.SelectedNode ;

while (ti.Parent != null)

ti = ti.Parent;

Questionarie qst = (Questionarie)ti.Tag;

lv.Items.Clear();

ListViewItem li = lv.Items.Add("1");

li.SubItems.Add("Наименование формы");

li.SubItems.Add(qst.Name);

li = lv.Items.Add("2");

li.SubItems.Add("Описание");

li.SubItems.Add(qst.Comment);

li = lv.Items.Add("3");

li.SubItems.Add("Всего записей");

li.SubItems.Add(qst.CountID.ToString());

li = lv.Items.Add("4");

li.SubItems.Add("Минимальный номер записи");

li.SubItems.Add(qst.MinID.ToString());

li = lv.Items.Add("5");

li.SubItems.Add("Максимальый номер записи");

li.SubItems.Add(qst.MaxID.ToString());

}

}

}