**Министерство образования и науки Российской Федерации**

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«Владимирский государственный университет**

**имени Александра Григорьевича и Николая Григорьевича Столетовых»**

**«ВлГУ»**

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

Студент Горева Анна Дмитриевна

Институт ИТР

Направление 09.03.02 «Информационные системы и технологии»

**Тема выпускной квалификационной работы**

Информационная система «Децентрализованный

реестр документов об образовании»

Руководитель ВКР Вершинин В.В.

(подпись) (ФИО)

Студент Горева А.Д.

(подпись) (ФИО)

**Допустить выпускную квалификационную работу к защите**

**в государственной экзаменационной комиссии**

Заведующий кафедрой Жигалов И.Е.

(подпись) (ФИО)

« » 20 г.

Лист задания

АННОТАЦИЯ

Тема: ИС Децентрализованный реестр документов об образовании.

Ключевые слова: Blockchain, Ethereum, Solidity, реестр, документ об образовании.

В данной работе представлены этапы проектирования ИС Децентрализованный реестр документов об образовании, проведены расчеты надежности и достоверности системы, описаны особенности реализации и тестирования проектов на Ethereum, решены некоторые задачи информационного менеджмента в отношении ИС.

ВКР представлена на 30 страницах, рисунков – 8, таблиц – , использованных источников – 10.

ANNOTATION

Theme: Information system “Decentralized register of documents on education”

Keywords: Blockchain, Ethereum, Solidity, register, documents on education.

In this paper, the stages of designing the information system "Decentralized register of education documents" are presented, calculations of reliability and reliability of the system, features implementation and testing of projects on Ethereum are described, some issues of information management are resolved.

The final qualifying work is presented on 30 pages, drawings - 8, tables - 3, used sources - 20.

Оглавление

[Введение 4](#_Toc514406552)

[1. Постановка задачи 6](#_Toc514406553)

[1.1 Анализ объекта автоматизации 6](#_Toc514406554)

[1.2 Анализ существующих решений 11](#_Toc514406555)

[1.2.1 MDM-системы 12](#_Toc514406556)

[1.2.2 Talend Open Studio 15](#_Toc514406557)

[1.2.3 Pentaho Data Integration 16](#_Toc514406558)

[1.2.4 Сторонние сервисы 17](#_Toc514406559)

[1.2.5 Исследовательские системы 18](#_Toc514406560)

[1.2.6 Dedupe 19](#_Toc514406561)

[1.2.7 Duke 23](#_Toc514406562)

[1.2.8 Python Record Linkage Toolkit 25](#_Toc514406563)

[1.2.9 Сравнительный анализ библиотек 26](#_Toc514406564)

[2. Проектирование системы 29](#_Toc514406565)

[2.1 Функциональные требования 29](#_Toc514406566)

[2.2 Нефункциональные требования 31](#_Toc514406567)

[2.3 Структура базы данных 32](#_Toc514406568)

[2.4 Алгоритм работы приложения 32](#_Toc514406569)

[3. Реализация системы 33](#_Toc514406570)

[3.1 Загрузка и предобработка данных 33](#_Toc514406571)

[3.2 Поиск дубликатов 36](#_Toc514406572)

# Введение

Объём информации, который приходится хранить и обрабатывать современным информационным системам, растет экспоненциально. Информация – важный ресурс, позволяющий частным компаниям и государственным учреждениям прогнозировать ситуацию на рынке, повышать свою производительность, улучшать работу предприятия по многим экономическим показателям. Для того, чтобы получить корректный результат работы системы и пользоваться всеми преимуществами владения информацией, необходимо гарантировать её качество. Одна из проблем, с которой приходится сталкиваться при работе с данными, это повторяемость. К огромному сожалению, не существует стандарта, с помощью которого можно было бы однозначно отобразить любую часть окружающего мира в информационный объект. Широкая вариативность описания – одна из причин по которой каждая информационная система имеет свою структуру данных для хранения информации о материальных объектах реального мира. Разночтения могут заключаться, например, в количестве имеющихся атрибутов, наименованиях, формате и даже в типе данных для отображения одного и того же свойства идентичных объектов. Не стоит забывать и о том, что львиная доли информации поступает напрямую от человека и важно контролировать её достоверность и актуальность. В данные могут закрадываться грамматические и синтаксические ошибки, они могут нарочно или случайно дублироваться пользователем, что внесет значительную погрешность в конечный результат. Кроме того, на определенном этапе жизненного цикла может возникнуть необходимость интеграции нескольких информационных систем. Перечисленные выше разночтения значительно затруднят интеграцию данных и поставят под сомнение их качество. Задачей сопоставления данных между собой занимается такое направление как dublicate detection. Одна из задач, решаемых в этой области, это определение связи объекта с реальным миром. Такое сопоставление позволяет избавиться от дубликатов, первопричиной которых может являться как пользователь, так и неграмотно выстроенный процесс интеграции систем. Иронично, что для обозначения этого направления, существует ещё около десятка дублирующих названий: Recode linkage, Dublicate detection, Deduplication, Reference reconciliation, Reference matching, Object consolidation, Fuzzy matching, Entity clustering, Hardening soft databases и т.д.

# 1 Постановка задачи

## 1.1 Анализ объекта автоматизации

Россельхознадзор является федеральным органом исполнительной власти, осуществляющим функции по контролю и надзору в сфере ветеринарии и фитосанитарии. Автоматизацией деятельности Россельхознадзора занимается информационно-вычислительный центр при ФГБУ «Федеральный центр охраны здоровья животных (ВНИИЗЖ)». Разработанные ИВЦ системы отвечают за функционал, связанный с использованием, сбором и анализом информации об объектах надзора федеральной службы (субъекты, объекты, процессы) и являются единой Государственной информационной системой в области ветеринарии. [1]

В рамках данной выпускной квалификационной работы пойдет речь об АИС «Цербер», также для полноты понимания картины следует упомянуть АИС «Аргус» и АИС «Меркурий». «Цербер» предназначен для обеспечения работ государственных инспекторов в Российской Федерации в сфере надзора и иных юридически значимых действий. Основное назначение системы «Аргус» - это автоматизация процесса рассмотрения заявок на ввоз, вывоз или транзит животных, продуктов и сырья животного происхождения, процесса выдачи разрешений или отказов. «Меркурий» - это система оформления ветеринарных сопроводительных документов в электронном виде. [2]

Каждая из перечисленных информационных систем обладает транзакционной базой данных. Исходя из специфики назначения этих систем, частично они хранят аналогичную информацию, в том числе, сведения о производственных площадках. Производственная площадка – это место, предназначенное для выполнения всего цикла производства продукции животного происхождения или его определенной стадии.

Производственная площадка, в базе данных Цербер имеет наименование ActualObject. Основные поля этой сущности представлены в таблице 1.1. Многие их этих полей, включая наименование и адрес, могут пустовать. Кроме того, большинство адресов не имеют привязки к идентификатору и хранятся в виде строкового представления.

Таблица 1.1 – Структура ActualObject

|  |  |  |
| --- | --- | --- |
| Наименование | Тип | Описание |
| id | VARCHAR | Первичный ключ и идентификатор сущности |
| version | BIGINT | Версия сущности |
| class | VARCHAR | Тип производственной площадки |
| guid | VARCHAR | Идентификатор, использующийся для поддержания версионности |
| isActive | TINYINT | Является ли производственной площадкой на сегодняшний день |
| isLast | TINYINT | Последняя ли версия производственной площадки? |
| status | INT | Отражает состояние сущности. Например, для объединенных дубликатов проставляется статус 230 для мастер-объекта и 430 зависимой сущности |
| createDate | DATETIME | Дата регистрации производственной площадки в ИС «Цербер» |
| updateDate | DATETIME | Дата редактирования сведений о производственной площадке |
| name | VARCHAR | Наименование производственной площадки |
| address\_country\_name | VARCHAR | Наименование страны |
| address\_region\_name | VARCHAR | Наименование региона |
| address\_district\_name | VARCHAR | Наименование района |
| address\_locality\_name | VARCHAR | Наименование города |
| address\_street\_name | VARCHAR | Наименование улицы |
| address\_house | VARCHAR | Номер дома |
| address\_building | VARCHAR | Номер строения |
| address\_addressView | VARCHAR | Адрес полностью |
| address\_additionalInfo | VARCHAR | Дополнительная информация о местонахождении, если нет фиксированного адреса |
| ikarAddress | Boolean | Информирует о том, что адрес синхронизирован с ИС Икар |
| ownerBusinessEntity\_guid | VARCHAR | guid владельца производственной площадки |
| type | VARCHAR | Тип производственной площадки. Существует 4 типа: ENTERPRISE, MARKET, SHIP, UNKNOWN |
| class | VARCHAR | Класс производственной площадки, более широкое понятие, чем тип. Существует 4 класса: Enterprise, Transport |
| updateDate | Date | Время последнего редактирования данных |

До 17 октября 2017 года, сведения о производственных площадках не синхронизировались между системами. В системе «Цербер» существует специализированный модуль «Поднадзорные объекты», который создавался для осуществления государственного ветеринарного надзора. В данном модуле ведется реестр производственных площадок, который по большей своей части пересекался с реестром производственных площадок в системе «Меркурий» и «Аргус». Разобщенные данные по разным системам не давали цельной картины для аналитики и затрудняли работу систем в сфере электронной сертификации. В таких случаях ясно видна необходимость создания единого реестра данных. Было решено произвести интеграцию данных для создания подробного реестра. Однако, интеграция была произведена в ущерб качеству существующих данных. В результате появилась проблема дублирования производственных площадок в системе «Цербер». Изначально предполагалось, что дубли будут устранены путем предоставления уполномоченными лицам дополнительных возможностей по очистке данного реестра, однако этот способ оказался неэффективен в силу своей трудоёмкости и нежелании пользователей осуществлять ручной поиск. В настоящий момент единый реестр производственных площадок находится в автоматизированной системе «Цербер». Каждая производственная площадка в системе может находится в двух основных статусах: «на согласовании» (кандидат на включение в реестр) и «подтверждена» (включена в реестр). Для устранения дублирующих объектов реализован функционал по объединению двух или нескольких производственных площадок в одну. Данный функционал доступен только сотрудникам ветслужб субъектов и территориальных управлений Россельхознадзора с соответствующей ролью «Администратор по ведению реестра объектов». Такое решение проблемы довольно протяжено во времени, ручной поиск не удобен и замедляет процесс устранения дубликатов в реестре. [3]

С 1 июля 2018 года согласно Федеральному закону Российской Федерации от 28.12.2017 № 431-ФЗ «О внесении изменений в статью 4 Федерального закона «О внесении изменений в Закон Российской Федерации «О ветеринарии», процесс электронной сертификации в ИС Меркурий станет обязателен для всех, а использование бумажных сертификатов разрешается только в исключительных случаях. [4] Наличие дубликатов в системы затруднит оформление электронных сопроводительных документов, в котором указываются все производственные площадки, задействованные при производстве продукции животного происхождения.

Предмет исследования: дублирование данных.

Объект исследования: процесс сопоставления и дедупликации данных.

Цель исследования: ускорить процесс поиска дубликатов производственных площадок в реестре информационной системы «Цербер».

1.2 Анализ существующих решений

Данные, с которыми работают информационные системы, не являются однотипными. Для интеграции и синхронизации данных принято использовать ETL-процессы. ETL-процесс включает в себя следующие этапы:

* Extract – извлечение исходных данных;
* Transform – преобразование данных, а именно приведение к одному типу данных, агрегирование, вычисление новых значений, очистка данных;
* Load – загрузка данных в точку назначения: хранилище или базу данных.

Среди множества подвидов данных, таких как структурированные или неструктурированные данные, метаданные и т.д., можно встретить такое определение как мастер-данные. Мастер-данные хранят в себе информацию об основных бизнес-объектах, с которыми работает предприятие. В ГОСТ 8000-2 прописано следующее определение мастер-данных: «данные, находящиеся во владении организацией и описывающие основные объекты этой организации. На эти данные следует ссылаться при составлении транзакций». [5] Самым ярким примером могут послужить персональные данные клиента или сведения о производимой продукции. Хранящиеся в реестре «Цербера» производственные площадки тоже можно классифицировать как мастер-данные. Очевидно, что контроль качества мастер-данных важен для большинства предприятий и есть спрос на готовые решения, представленные на ИТ-рынке.

### 1.2.1 MDM-системы

MDM-системы (Master Data Management system) – это централизованные системы, которые не требуют доработки и без изменений могут быть внедрены в текущую инфраструктуру предприятия для централизованного управления данными. Основные преимущества этого ряда систем заключаются в создании единого информационного пространства и стандартизации правил хранения информации. Благодаря этому, повышается уровень контроля информации, изменения синхронизируются среди всех систем предприятия, снижается уровень ошибок. Важно, что в MDM-системах предусмотрен функционал по поиску дублей среди данных и их устранению.

Существует несколько подходов к внедрению MDM-систем [6]:

* Registry Model – система содержит только списки ссылок на другие источники данных предприятия и не аккумулирует данные непосредственно в себе.
* Repository Model – система хранит мастер-данные со всеми необходимыми другим системам атрибутами. Сторонние системы предприятия должны каждый раз обращаться к ней при работе с информацией.
* Hybrid Model – система совмещает в себе два первых подхода и хранит как данные, так и ссылки на них в других системах. Таким образом, при правильно выстроенном процессе синхронизации данных, сторонним системам не нужно будет постоянно обращаться к MDM-системе, но при этом она будет служить единым авторитетным источником данных на предприятии.

Консалтинговая компания Gartner ежегодно проводит анализ существующих на ИТ-рынке решений и публикует свои выводы в открытый доступ. В итоговый рейтинг за 2017 год вошли следующие MDM-системы:

* Ataccama
* EnterWorks
* IBM
* Informatica
* Oracle
* Orchestra Networks
* Profisee
* Riversand
* SAP
* Stibo Systems
* TIBCO Software

Gartner проанализировала сильные и слабые стороны каждой из перечисленных систем. (Рисунок 2.1) Лидером рынка была признана Informatica, эта система выбиралась пользователями для внедрения в свою компанию чаще остальных и получала больше положительных отзывов, чем все остальные представленные на IT-рынке системы. [7]



Рисунок 2.1 - Классификация MDM-систем Gartner 2017

К сожалению, все представленные MDM-системы имеют ограниченную сферу применения. Мастер-данные, с которыми они умеют работать, это прежде всего информация о клиентах – их фамилия, имя отчество, адрес, номера телефонов. Системы имеют закрытый исходный код, поэтому предприятие не может адаптировать их под себя, в случае, если этих полей недостаточно или не подходят правила обработки этих полей. Таким образом, системы хорошо оптимизированы для дедупликации имен, так как в них внедрены алгоритмы стемминга и лемматизации, но трудно адаптируются под предметную область. Кроме того, сведения о используемых алгоритмах и исходного кода данных систем нет в открытом доступе. Поэтому в рамках ветеринарного надзора и фитосанитарии эти преимущества не имеют силы.

### 1.2.2 Talend Open Studio

Talend Open Studio (TOS) — система с открытым исходным кодом на базе платформы Eclipse. Основное предназначение системы — построение ETL-процессов для интеграции данных из разных источников. Каждый ETL-процесс представляет из себя отдельную задачу, каждая из которых может состоять из подзадач или шагов и связей между ними. Таким образом, Talend Open Studio можно использоваться для решения следующих задач:

* интеграция данных;
* миграция данных;
* синхронизация данных;
* построение хранилищ данных.

Функциональных возможностей этой системы достаточно для того, чтобы подготовить данные и мигрировать из базы данных Цербера. Такая система хорошо подходит для построений ETL-процессов для очистки данных и их дальнейшей дедупликации.

### 1.2.3 Pentaho Data Integration

Система Pentaho Data Integration (PDI, также известная как Kettle) это компонент комплекса Pentaho отвечающий за ETL-процессы. Его основной функционал заключается в интеграции данных из разных источников с помощью ETL-процессов, которые можно выстраивать из компонентов в IDE Spoon. В Spoon можно формировать трансформации, состоящие из шагов. Некоторые из этих шагов отлично подходят для процесса очистки данных и перечислены в таблице 2.1.

Таблица 2.1 Шаги для трансформаций PDI

|  |  |
| --- | --- |
| Наименование | Описание |
| Table Input | Шаг загрузки данных из базы |
| Table Output | Шаг выгрузки данных в базу |
| String operations | В зависимости от настроек удаляет из входящей строки цифры, специальные символы, пробелы, приводит в нижний или верхний регистр, добавляет отступы. |
| Calculator | Может работать с одним полем или двумя одновременно вычисляя на их основе новое значение. Способен высчитывать расстояние Левенштейна, Metaphone, сходство Джаро — Винклера и т.д., удалять служебные символы, производить математические вычисления. |
| Modified Java Script Value | Позволяет выполнять собственный код на языке Java. |

Kettle является бесплатно распространяемым программным обеспечением, однако для него можно купить платные плагины для очистки данных Melissa Data Quality. Melissa способна работать со стандартным набором сведений о клиенте: фамилия, имя, отчество, номер телефона, e-mail, адрес. Также у Kettle есть интеграция с платформой DataCleaner. Бесплатная Community версия DataCleaner включает в себя базовые возможности, доступные и в Kettle, однако платная Professional версия предоставит компоненты для дедупликации и очистки данных. Данное ПО уже используется в ИВЦ Россельхознадзора для создания витрин данных, поэтому предпочтительнее, при разработке системы использовать именно его.

### 1.2.4 Сторонние сервисы

Дублирование данных – довольно популярная проблема, с которой сталкиваются не только крупные предприятия. Зачастую эта проблема может возникнуть на небольших или средних размерах предприятиях. Объём хранимой информации может быть небольшим, а поступление новое информации происходит довольно редко. В таком случае внедрять крупную MDM-систему не имеет смысла, дешевле и удобнее будет воспользоваться сторонним сервисом для обеспечения качества данных, в том числе, для дедупликации данных. DaData.ru это российский сервис, предоставляющий свой API для контроля за качеством данных. В сервисе реализован следующий функционал:

* Исправление ошибок в именах пользователей;
* Привязка адреса к ФИАС;
* Синхронизация кодов с КЛАДР, ФИАС, ОКАТО, ОКТМО и ИФНС;
* Исправление формата номеров телефонов;
* Проверка паспортных данных через ФМС;
* Поиск дубликатов среди аккаунтов пользователей.

Формат загрузки данных – xlsx. Цена пользования данным сервисом составляет 10 рублей за 1000 записей.

Из сервисов, ориентирующихся на англоговорящих пользователей, популярен Dedupe.io. API-интерфейс Dedupe.io позволяет искать дубликаты на любом языке, используя стандартную среду RESTful. С помощью веб-API можно дедуплицировать сведения о клиентах, а именно фамилию, имя, отчество, номер телефона, адрес, e-mail. С помощью данного API можно объединять дубли в одну сущность. Данный сервис не требует платы для объемов менее тысячи строк. Начиная с тысячи, минимальная цена составляет от 10 долларов и выше. Если количество строк превышает миллион, оформляется месячная подписка, цена которой обговаривается индивидуально.

### 1.2.5 Исследовательские системы

Существует большое количество свободно распространяемых систем сопоставления данных, разработанных исследователями как часть их научной работы по улучшению существующих, а также для создания новых алгоритмов и методов сравнения сущностей. Например, D-Dupe, FEBRL, OYSTER, MOMA и целый ряд схожих систем. Некоторые из этих них имеют графический пользовательский интерфейс и документацию, некоторые носят более фундаментальный характер [7]. В статье «Frameworks for entity matching: A comparison» Hanna Köpcke и Erhard Rahm приведена сравнительная характеристика одиннадцати таких систем. Критерии сравнения включают в себя как функциональные возможности, так и их эффективность в отношении сложности алгоритмов. [8] К сожалению, нацеленность этих систем на усовершенствование алгоритмов и методов, не позволяет применить их на практике. Основной целью создания этих систем был поиск лучшего пути для создания более приближенных к производству систем, а их реализация носит экспериментальный характер.

### 1.2.6 Dedupe

Для тех, кому не подошли варианты внедрения крупной MDM-системы и использования сторонних платных сервисов, существует третий вариант – реализация собственной информационной подсистемы обеспечения качества мастер-данных.

Dedupe – бесплатная библиотека на Python, которая при помощи машинного обучения способна выполнять нечеткое сравнение в структурированных данных. Платный сервис Dedupe.io, описанный в пункте 1.2.4, разработан с помощью этой библиотеки. Настройка системы для дедупликации может происходить в двух режимах: обучение с учителем и активное обучение. Активное обучение – частный случай обучения с учителем, когда система может в интерактивном режиме запрашивать у пользователя новую информацию для обучения. Dedupe выбирает несколько записей и спрашивает пользователя, являются ли они дубликатами друг друга. Пользователю доступно несколько вариантов ответа на вопрос: да, нет, не уверен, конец (что означает закончить настройку). По окончании интерактивного диалога формируется файл с настройками, который впоследствии будет использоваться системой для дедупликации данных. Минимальное количество данных, достаточных для формирования файла настроек – 10 подтвержденных дубликатов и 10 подтвержденных недубликатов. В Dedupe предусмотрен целый список типов данных, за каждым типом закреплен свой алгоритм для вычисления схожести, а также есть возможность расширить список алгоритмов, реализовав свою функцию сравнения. Типы данных и алгоритмы представлены в таблице 2.2.

Таблица 2.2 Типы данных в Dedupe

|  |  |
| --- | --- |
| Тип данных | Описание |
| String | Строковый тип данных, используется чаще всего. Алгоритм схожести: Affine gap string distance |
| ShortString | Эквивалентен String, однако поле не будет учитываться при кластеризации. Может использоваться, например, для описания штрих кода. Алгоритм схожести: Affine gap string distance |
| Text | Используется для описания больших объемов текстовых данных. Может использоваться, например, для поля описания продукции. Алгоритм схожести: Vector space model |
| LatLong | Используется для описания местонахождения через ширину и долготу. Алгоритм схожести: Haversine formula |
| Set | Подобен формату Text, используется если поле может хранить в себе сразу несколько значений. Например, у сущности Книга может быть указано несколько фамилий в поле Автор. Алгоритм схожести: Vector space model |
| Interaction | Используется для комбинации несколько. |
| Exact | Используется для точного сравнения. |
| Exists | Используется для проверки наличия поля |
| Categorical | Используется для описания свойства, значения которого строго определены, известны и имеют строковое представление. |
| Price | Используются для сравнения положительных, отличных от ноля величин. |
| DateTime | Используется для сравнения дат, кроме того, по умолчанию включен поиск даты в строке. |
| Address | Специализированный тип данных для описания адресов на английском языке. |
| Name | Специализированный тип данных для описания имен и наименований фирм на английском языке. Автоматически разбивает строку на такие компоненты, как имя, фамилия, форму собственности или аббревиатуру. |
| Fuzzy Category | Используется в случае, если к одному значению прикреплено несколько синонимов. |

Для описания одного и того же поля сущности допускается использование нескольких типов данных. Для каждого поля можно указать метку «has missing», в случае, если сведения имеют пропуски.

Для уменьшения количества пар сравнения и за счет этого снижения времени поиска дубликатов, в Dedupe используются предикатные функции и инвертированный индекс для блокировки. Основная идея блокировки заключается в том, чтобы объединить строки в группы по какому-либо признаку, гарантирующему, что дубликаты одной сущности находятся в одном блоке. Предикатным блоком называют группу записей, образованных с помощью специальной предикатной функции. В данной библиотеке присутствуют следующие предикатные функции:

* целое поле;
* поле токена;
* общее целое число;
* одинаковые три начальных символа;
* одинаковые пять начальных символов;
* одинаковые семь начальных символов;
* близко расположенные друг к другу целые числа;
* общая четырехграмма;
* общая шестиграмма.

Кроме блоков предикатов, для уменьшения количества пар сравнения в Dedupe используется инвертированный индекс. Инвертированный индекс — структура данных, в которой для каждого слова коллекции сущностей в соответствующем списке перечислены все сущности в коллекции, в которых оно встретилось. Инвертированный индекс используется для поиска по текстам. Важно, что, вычисление инвертированного индекса займет больше времени, чем вычисление предикатной функции.

### 1.2.7 Duke

Duke – бесплатная библиотека дедупликации данных на Java, написанная поверх библиотеки Apache Lucene. Apache Lucene - свободная библиотека на языке Java для высокопроизводительного полнотекстового поиска. В Duke присутствуют готовые шаблоны по предобработке данных, их список представлен в таблице 2.3. кроме того, можно реализовать собственный алгоритм очистки.

Таблица 2.3 - Шаблоны для предобработки данных в Duke

|  |  |
| --- | --- |
| Наименование | Описание |
| LowerCaseNormalizeCleaner | Самый используемый шаблон для строковых значений. Все буквы переводятся в нижний регистр, удаляются пробелы с начала и конца, нормализуется количество пробелов. Он также удаляет акуты, превращая é в e. |
| DigitsOnlyCleaner | Этот шаблон удаляет все символы, кроме цифр. Может быть использован, например, для очистки почтового индекса. |
| TrimCleaner | Просто обрезает пробелы в начале и конце строки. |
| RegexpCleaner | Этот шаблон использует регулярные выражения для очистки. Он сопоставляет ввод с заданным ему регулярным выражением. |
| MappingFileCleaner | Использует CSV файл для замены некоторых слов на сопоставленные им новые значения. |
| GenericValueCleaner | Этот шаблон используется после применения какого-либо шаблона ранее, для удаления бессмысленных значений. Например, если у многих людей номер телефона 999-999-9999, то информация вряд ли достоверна и ее можно удалить. |
| PhoneNumberCleaner | Этот очиститель понимает синтаксис телефонных номеров и нормализует номера телефонов в стандартном формате (+ номер CC, где CC = код страны). |
| PersonNameCleaner | Экспериментальный шаблон для английских имен, расшифровывает сокращенные имена. |
| NorwegianAddressCleaner | Шаблон для нормализации норвежских адресов. |
| NorwegianCompanyNameCleaner | Шаблон для нормализации наименований норвежских компаний. |

Для сравнения входных данных в Duke используются компараторы, некоторые из которых представлены в таблице 2.4.

Таблица 2.4 - Компараторы Duke

|  |  |
| --- | --- |
| Наименование | Описание |
| Levenshtein | Для определения меры схожести используется стандартный алгоритм Левенштейна. |
| WeightedLevenshtein | Конфигурируемая версия Levenshtein с весами. |
| JaroWinkler | Компаратор. использующий сходство Джаро — Винклера для сравнения строк. |
| SoundexComparator | SoundexComparators сравнивает термины с помощью Soundex алгоритма. Если две строки имеют одинаковый код Soundex, они считаются равными. |
| MetaphoneComparator | MetaphoneComparator похож на SoundexComparators, однако использует алгоритм Metaphone для сравнения. |

Duke используется норвежской энергетической компанией Hafslund в своей ERP-системе. Yoxel Portal - облачная CRM-система также задействовала эту библиотеку для синхронизации контактов клиентов в разных системах. Cityhotels - это норвежский портал бронирования гостиниц, который собирает информацию о гостиницах и ценах из разных источников. Для дедупликаии собранных данных и агрегировании информации об отелях и ценах из разных источников, портал использует Duke. Такая популярность библиотеки в Норвегии объясняется наличием специальных шаблонов для нормализации имен, адресов и наименований организаций на норвежском языке.

### 1.2.8 Python Record Linkage Toolkit

Python Record Linkage Toolkit – бесплатная библиотека для связывания записей в источниках данных и поиска дубликатов. Библиотека содержит методы индексирования, функции сравнения данных и классификаторы. Для сравнения данных предоставляются компараторы с реализацией таких алгоритмов как расстояние Дамерау — Левенштейна, алгоритм Левеншетейна, расстояние Джаро, Сходство Джаро — Винклера и т.д. Она была разработана для исследований и обработки небольших или средних размеров файлов.

### 1.2.9 Сравнительный анализ библиотек

В таблице 2.5 приведен сравнительный анализ трех библиотек: Dedupe 1.9.2, Duke 1.2, Python Record Linkage Toolkit 0.8.1. Основное внимание было сконцентрировано на первых двух библиотеках, а Python Record Linkage Toolkit был отвергнут на первоначальном этапе. Основным его недостатком является неумение работать с большими объемами данных. [9] Данная библиотека хорошо подходит для учебных или небольших исследовательских проектов. Duke и Dedupe способны работать с гораздо большими объёмами информации и используются в коммерческих продуктах. Несмотря на отсутствие в Dedupe 1.9.2 предобработки данных, нельзя сказать, что Duke сильно выигрывает в данном пункте, так как он не сохраняет результаты очистки. Таким образом, предобработка данных в Duke – процесс, повторяющийся для каждого процесса поиска дубликатов, даже для одного и того же набора данных. В целях экономии времени, целесообразнее делать предобработку данных сторонним средством, например, с помощью ETL-процессов в Pentaho Data Integration Kettle. Кроме того, в отличие от Duke, в Dedupe по умолчанию работает алгоритм поиска и игнорирования при анализе стоп-слов. Основной механизм индексации в Apache Lucene – инвертированный индекс. [10] Dedupe так же использует инвертированный индекс для блокировки данных. Таким образом, решающим фактором при сравнении двух библиотек стало количество доступных методов обучения системы. В Duke для каждой дублируемой сущности создается XML-конфигурация, где для каждого поля назначается компаратор и коэффициент – пороговое значение для принятия решения о том, является ли объект дубликатом или нет. Для оптимизации этих коэффициентов доступна возможность их оптимизации с помощью генетических алгоритмов. В Dedupe 1.9.2 систему можно обучить на ранее найденных дубликатах, или в интерактивном режиме с помощью активного обучения. Для производственных площадок уже накоплена информация о дубликатах, которые были ранее устранены пользователями информационной системы Цербер. Таким образом, лучший вариант библиотеки для разработки подсистемы является Dedupe 1.9.2, так как с её помощью проще обучить систему правильному поведению при поиске дубликатов производственных площадок.

Таблица 2.5 – Сравнительный анализ библиотек

|  |  |  |  |
| --- | --- | --- | --- |
|  | Dedupe 1.9.2 | Duke 1.2 | Python Record Linkage Toolkit |
| Язык программирования | Python | Java | Python |
| Количество компараторов | 14 | 14 | 5 |
| Предобработка данных | Нет | 10 шаблонов | Нет |
| Игнорирование стоп-слов | Да | Нет | Нет |
| Индексация | Предикатные блоки, Инвертированный индекс, | Apache Lucene | ElasticSearch |
| Обучение | Активное обучение, обучение с учителем | Отсутствует, только ручной подбор коэффициентов | Обучение с учителем, обучение без учителя |
| Объемы данных | Любые объемы данных | Любые объемы данных | Небольшие объемы данных |
| Форматы данных | CSV, СУБД | CSV, СУБД | CSV |
| Многопоточность | Да | Да | Нет |
| Использование в коммерческих продуктах | Dedupe.io | ERP Hafslund, CRM Yoxel Portal, Cityhotels | Нет |
| Интеграция данных из нескольких источников (Режим record linkage) | Да | Да | Да |
| Режим сопоставления со справочником | Да | Нет | Нет |

# 2 Проектирование системы

## 2.1 Функциональные требования

Прецедент «Загрузка данных».

Назначение: Получить новые данные и обновить существующие для поиска дублей.

Входные данные: Сведения для подключения к базе данных (логин и пароль пользователя с соответствующими правами доступа, хост и порт на которой запущена СУБД, наименование базы данных и таблицы).

Выходные данные: данные, среди которых необходимо найти потенциальные дубликаты.

Прецедент «Очистка данных».

Назначение: Привести данные в удобную для анализа форму.

Входные данные: массив необработанных грязных данных.

Выходные данные: очищенные данные.

Прецедент «Индексация».

Входные данные: очищенные данные.

Выходные данные: сформированные инвертированные индексы.

Прецедент «Кластеризация».

Входные данные: список индексов.

Выходные данные: принятое решение на вопрос, могут ли объекты быть дубликатами.

Прецедент «Создание заявки на объединение».

Входные данные: данные о результатах кластеризации.

Выходные данные: заявка на объединение объектов в один.

На рисунке 2.1 представлена диаграмма прецедентов.



Рисунок 2.1 Диаграмма прецедентов

## 2.2 Нефункциональные требования

В настоящий момент транзакционная база Цербера хранит около двух с половиной миллионов записей о производственных площадках. Разрабатываемая система должна без сбоев выдерживать нагрузку при индексировании и поиске дубликатов среди этого объема записей. При таком объеме данных становится невозможным сравнение каждой записи со всеми другими записями базы, так как это потребует высокое время обработки. Таким образом можно выделить следующие нефункциональные требования:

* Сложность алгоритма поиска дубликата должна быть проще, чем О(n2);
* Сохранение стабильности работы системы во время обработки двух миллионов записей за один сеанс работы;

## 2.3 Структура базы данных

Особенности структуры данных ActualObject учтены в структуре базы данных подсистемы, которая представлена на рисунке 2.2. Сущность actual\_object\_duple\_entity представляет из себя копию необходимых данных о производственной площадке, прошедших очистку перед загрузкой в базу данных подсистемы. Таблица join\_request – заявка на объединение площадок, которые система посчитала дубликатами. Таблица statistic\_info хранит информацию о том, когда производились запуски системы, сколько сущностей было обработано и сколько дубликатов найдено.

## 2.4 Алгоритм работы приложения

Вставить IDEF3 –

2.5

# 3 Реализация системы

## 3.1 Загрузка и предобработка данных

Прежде чем данные из базы данных Цербер пройдут через процесс поиска дубликатов, они должны пройти этап их предобработки и загрузки в БД подсистемы. Для первичной и последующих процессов загрузки и очистки были разботанные ETL-процессы в Pentaho Data Integration Spoon. Процесс первичной загрузки данных представлен на рисунках 3.1 и 3.2.

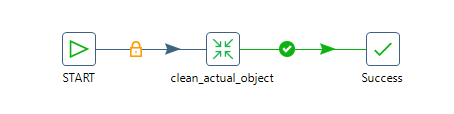


Рисунок 3.1 Первичная загрузка и предобработка данных. Работа.

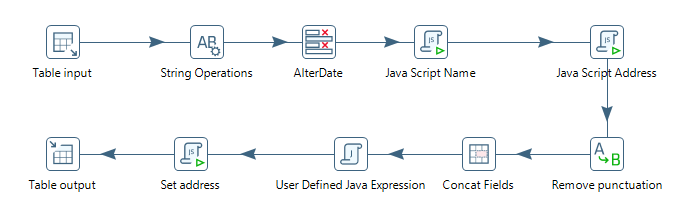


Рисунок 3.2 Первичная загрузка и предобработка данных. Трансформация.

С помощью шага Table input выполнятеся SQL-запрос выгрузки данных из базы данных Цербера. Далее, шаг String Operations приводит все строковые представления к нижнему регистру, очищает поле индекса от любых символов кроме цифр. Шаг Alter Date приводит дату к необходимому для загрузки в базу формату. Шаги Java Script Name и Java Script Address предназначены для поиска и удаления стоп-слов. Несмотря на то, что Dedupe умеет искать стоп-слова, мы не можем контролировать его словарь, поэтому здесь присутствуют эти шаги.

Список стоп-слов с учётом регистра для адреса: российская федерация, улица, квартира, область, район, город, корпус, обл, село, дом, кв, стр, пгт, ст-ца, пр-кт.

Список стоп-слов с учётом регистра для наименования: закрытое акционерное общество, акционерное общество, публичное акционерное общество, личное подсобное хозяйство, муниципальное бюджетное общеобразовательное учреждение, производственный кооператив, сельскохозяйственный производственный кооператив, кафе, кфх, крестьянское фермерское хозяйство, пищеблок, муниципальное, казенное, казённое, бюджетное, общеобразовательное, дошкольное, учреждение, общество с ограниченной ответственностью, муниципальное, бюджетное, образовательное, учреждение, теплоход, магазин, рыболовное судно, рыботранспортное судно, транспортное судно, индивидуальный предприниматель, федеральное, государственное, предприятие, ип, лпх, ооо, оао, зао, клс, пао, гуп, гкфх, гбпоу, свх, сош, гбу, гпоу, мбоу, мкдоу, мдоу, мадоу, мбдоу, фгуп, моу, мау, рш, ао, сртм-к, стр, тр, тсм, батм, бмрт, ртм.

Шаг Remove Punctuation предназначен для очистки полей от символов пунктуации. Так исходная таблица производственных площадок ActualObject содержит большое количество полей, предназначенных для хранения различных атрибутов адреса, было принято решение с помощью Concat Fields вычислить новое поле complex\_address как сумму полей этих атрибутов. Это необходимо, так как у некоторых производственных площадок пустует поле address\_addressView, но заполнены другие атрибуты, такие как address\_region\_name или address\_district\_locality. User Defined Java Expression – это шаг, с помощью которого можно встраивать использование регулярных выражений на Java в ETL-процесс. На этом этапе используется регулярное выражение поиска избыточного количества пробелов и удаления из строки. Как было сказано ранее, поле адреса может пустовать. В этом случае, с помощью шага Set Address в поле адреса устанавливается поле complex\_address, вычисленное на предыдущих этапах. Шаг Table Output является завершающим в этом ETL-процессе, он выгружает обработанные данные в таблицу actual\_object\_duple\_entity базы данных подсистемы qualitydb.

Процесс повторной загрузки данных представлен на рисунке 3.3. Единственным отличием от первичной загрузки, является замена шага Table Output на Insert/Update, который сначала проверяет наличие загружаемой производственной площадки в базе. Если такая площадка была загружена ранее, данный шаг сверяет поле update\_date, которое хранит время последнего изменения записи, и обновляет запись, только в том случае, если она была изменена.

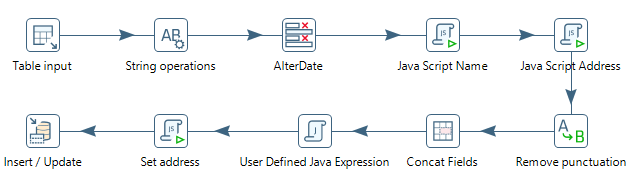


Рисунок 3.3 Повторная загрузка и предобработка данных. Трансформация.

## 3.2 Поиск дубликатов

### 3.2.1 Обучение

Для обучения Dedupe необходимо сформировать выборку из реальных данных. В базе данных ИС Цербер хранится информация о всех, ранее найденных дубликатах площадок. Для закачки сведений в базу данных подсистемы был выстроен ETL-процесс, аналогичный процессу первичной загрузки данных, который был описан в пункте 3.2. Кроме этого, из таблицы ActualObject было выгружено небольшое количество производственных площадок, все они были проверены на предмет отсутствия в них дубликатов вручную.

### 3.2.2 Тестирование