УДК 004.622

ЗАДАЧИ ИДЕНТИФИКАЦИИ ФИЗИЧЕСКИХ И ЮРИДИЧЕСКИХ ЛИЦ В ХРАНИЛИЩАХ ДАННЫХ

А.Ю. Гула, А.П. Игнатенко, А.В. Чадюк

ООО “ЭР-ДЖИ-ДЕЙТА”,
03056, Киев, ул. Политехническая, 33, к. 616.
Тел.: +380 44 241 9131; факс: +380 44 236 3188.
E-mail: alexg@rgdata.com.ua
Институт программных систем НАН Украины,
03187, Киев, проспект Академика Глушкова, 40.
Тел.: +380 44 526 6025.
E-mail: o.ignatenko@isofts.kiev.ua

Работа посвящена методам идентификации физических и юридических лиц в хранилищах данных. Предложен алгоритм подготовки, сравнения и идентификации записей. Проведена оценка сложности алгоритма. Предлагается схема программной реализации процесса обработки данных с применением методов нечёткой логики и системы правил.

The article describes methods of physical and legal persons’ records identification in data warehouses. Algorithm of data preparation and record identification is suggested. There is proposed structure of data processing application using fuzzy comparison methods and rule set system.

Введение

Хранилище данных как важнейший инструмент управления и развития бизнеса притягивает к себе все большее внимание. Развитие банковского и финансового секторов экономики требует создания и ведения больших интегрированных хранилищ, предназначенных для хранения жизненно важной для функционирования компании информации. Поэтому задачи сбора, унификации и согласования разнородных данных, корректного наполнения ими хранилища представляют значительный интерес.

Хранилище данных можно определить как предметно-ориентированный, интегрированный, зависимый от времени набор данных, предназначенный для поддержки принятия решений различными группами пользователей. Так как хранилище данных носит предметно-ориентированный характер, его организация нацелена на содержательный анализ информации, а не на автоматизацию бизнес-процессов. Это свойство определяет архитектуру построения хранилища данных и принципы проектирования модели данных, отличные от тех, что применяются в оперативных системах [1]. Интегрированность означает, что, например, данные о клиентах, подразделениях и банковских продуктах, полученные из различных источников, хранятся согласованно и централизованно.

Хранилище данных содержит исторические данные, или зависимый от времени набор данных. Иными словами, если в оперативных источниках представлены самые последние значения (например, текущее наименование клиента или его физический адрес), то хранилище данных будет содержать в себе всю их предысторию с указанием периода, когда те или иные данные были актуальны. Хранилище данных предназначено для поддержки принятия решений, и его пользователи — это высший и средний менеджмент банка, аналитики, представители подразделений финансового анализа и маркетинга.

Наиболее уязвимым местом использования хранилища данных, с точки зрения анализа и последующего принятия решения, является корректность загруженных в него данных. К сожалению, при создании хранилищ, как правило, очень мало внимания уделяется очистке поступающей в него информации. Такая практика приводит к засорению хранилища несогласованными, ошибочными данными. Очистка данных необходима, поскольку информация разнородна и собирается из различных источников. Именно наличие множества точек, используемых для сбора информации, делает процесс очистки особенно актуальным.

В любой сложной системе сбора информации возникают рассогласования, сбои или ошибки ввода. Возможно, иногда есть резон смириться с ними, чем тратить деньги и время на избавление от них. Но, в общем случае, нужно стараться любым способом снизить количество ошибок до приемлемого уровня. Типовые причины возникновения ошибок при наполнении хранилища можно разделить на следующие группы:

- противоречивая информация;
- пропуски в данных;
- аномальные значения;
- ошибки ввода данных.

Для решения каждой из этих проблем существуют отработанные методы [2, 3]. Конечно, ошибки можно править и вручную, но при больших объемах данных это становится просто невозможно. Поэтому приемлемым

решением этих задач является использование интеллектуальных алгоритмов обработки данных в автономном режиме при минимальном участии человека.

Следует отметить, что «грязные» данные представляют собой очень большую проблему. Фактически они могут свести на нет все усилия по созданию хранилища данных. Причем, речь идет не о разовой операции, а о постоянной работе в этом направлении. Идеальным вариантом было бы создание шлюза, через который проходят все данные, попадающие в хранилище. Механизмы фильтрации должны стать таким же атрибутом хранилищ данных как, например, OLAP [4].

Задача идентификации данных

В данной работе мы рассмотрим одну из характерных проблем, возникающих при ведении хранилища. Эта проблема возникает при обеспечении хранилища данных крупных предприятий, фирм и связана она с информацией, описывающей деятельность физических и юридических лиц. Первый вопрос, возникающий при заполнении хранилища новой информацией (что происходит регулярно) по некоторому субъекту деятельности – данное физическое или юридическое лицо уже присутствует в базе или является новой сущностью? Задача существенно усложняется наличием следующих факторов:

- независимость источников – у каждой информационной системы существуют собственные схемы хранения и правила ввода данных. Кроме того, список хранимых атрибутов и полнота описания может существенно отличаться. В результате чего одни и те же сущности описаны разным набором атрибутов с различным составом и полнотой;
- отсутствие или недостаточная проработанность процедур контроля при вводе и перегрузке данных. В итоге в хранилище происходит умножение экземпляров одного физического или юридического лица;
- ошибки ввода – как показывает практика, всегда сопутствуют вводу данных в информационную систему;
- наличие в хранилище данных, представление которых допустимо различным образом (формат, язык и т.п.). Например, фамилия физического лица может быть введена на русском, украинском или английском.

Решение этой задачи является достаточно нетривиальным и включает в себя как математические алгоритмы сравнения строк, так и создание вспомогательных структур хранилища, поддерживающих ведение эталонов и версий.

Предлагается решать задачу в рамках создания системы сопоставления записей, которая должна обеспечивать загрузку новых данных, их анализ, идентификацию и запись в хранилище. Различные аспекты создания такой системы описаны в [5]. Ее ядром является подсистема идентификации лиц, реализующая математические алгоритмы [6] сравнения записей рабочего набора – множества не идентифицированных записей с эталонным набором. Принципиальная схема работы такой системы показана на рис. 1. Основными ее этапами являются унификация, подготовка и идентификация данных хранилища.

Сравнение данных. Легко заметить, что основной вклад в эффективность ее работы вносит именно стадия интеллектуальной обработки данных.

Для оценки степени идентичности двух записей используется сравнение данных с использованием предварительной подготовки данных и применением алгоритмов нечёткой логики. Для оценки результатов сравнения также применяется система правил.

Нормализация данных. Для повышения точности сравнения данных необходима их предварительная подготовка, которая заключается в приведении разнородных данных из разных источников к единому стандарту. Информация, описывающая физическое или юридическое лицо, как правило, содержит поля, указанные в табл. 1.

Нормализация включает в себя унификацию структуры данных – загрузка данных из различных источников в таблицу единого формата.

Кроме того, нормализация также включает в себя унификацию данных – перевод строк в верхний регистр, удаление непечатных и повторяющихся символов, удаление пробелов в начале и конце строки.

Подготовка данных. В случае необходимости обработки больших информационных массивов (больше 1 млн. записей) после нормализации данных должна следовать их подготовка. Подготовка данных заключается в предварительной обработке записей в базе данных и сохранении промежуточной информации с целью уменьшения алгоритмической сложности при проведении идентификации.

Для подготовки данных предлагается такой алгоритм:

1. Объединение данных – данные полей, содержащих фамилию, имя, отчество для физических лиц и полное и короткое название для юридических лиц, объединяются в единую строку «название».

2. Кодирование данных:

2.1. Транслитерация в кириллицу символов латиницы: «ABCDEFGHIJKLMNOPQRSTUVWXYZ» -> «АБЦДЕФГХИЖКЛМНОПКРСТУВВХЙЗ».

2.2. Замена определённых небуквенных символов, похожих по написанию на соответствующие символы кириллицы, и находящихся в окружении букв: «1;:[]{}()/\|» → «И».

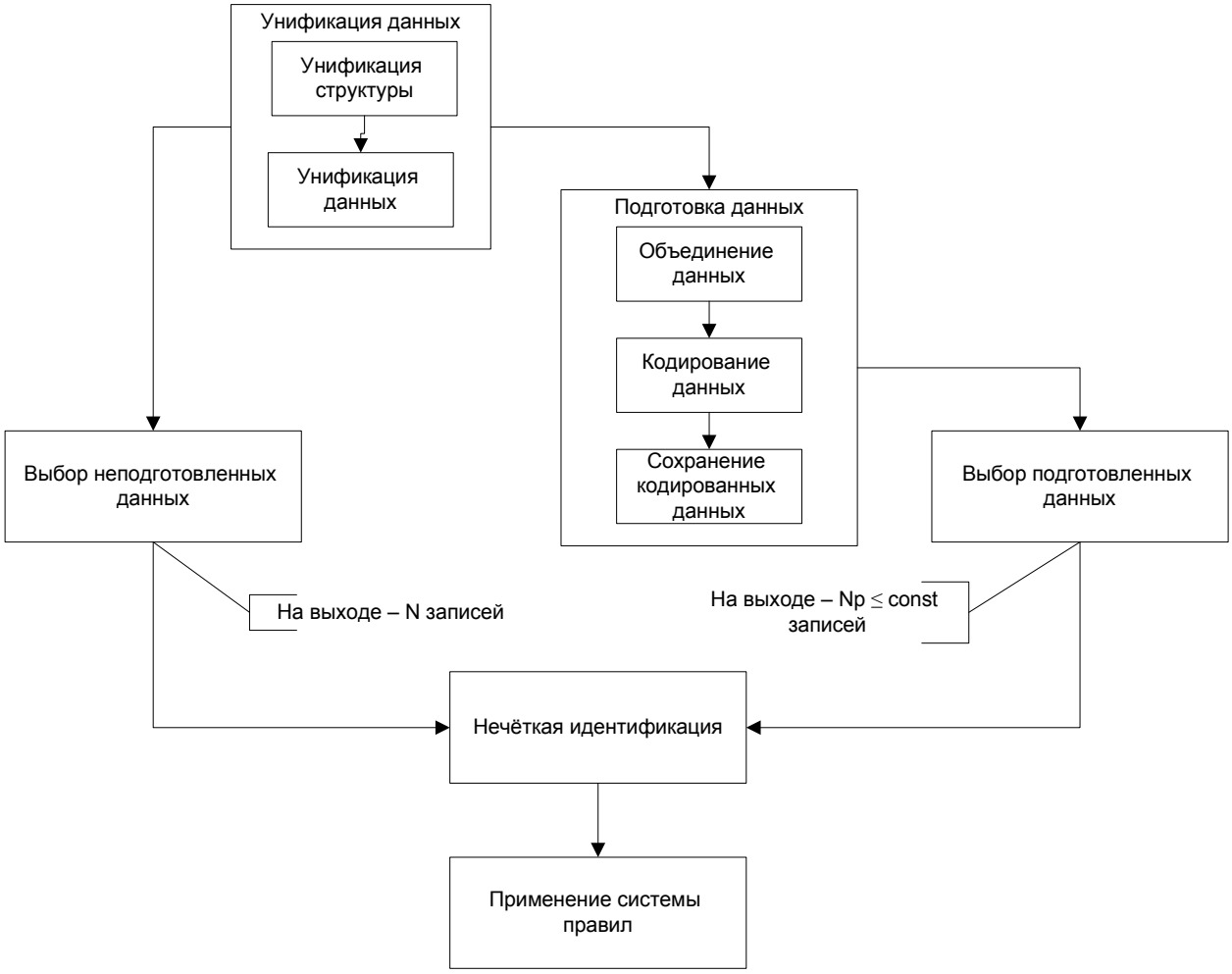

Рис. 1. Схема обработки данных

Таблица 1

№ п/п	Физические лица	Юридические лица
1	Идентификационный код	Код ЕДРПОУ
2	Фамилия	Полное название
3	Имя	Короткое название
4	Отчество	-
5	Дата рождения	Дата регистрации
6	Паспортные данные	Регистрационные данные

2.3. Замена гласных «ОАЯЮУЭЕЁИЙЫЇ» на гласную «А», выступающую признаком наличия гласной в данной позиции.

2.4. Замена согласных на глухие созвучные: «Б» → «П», «ВГҐКФ» → «Х», «Д» → «Т», «ЖЗЦЧШЩ» → «С».

2.5. Удаление повторяющихся символов, а также символов, входящих в набор «ЬЪ».

2.6. Название разбивается на слова, критерием разбиения являются все символы, не входящие в набор «АКЛМНПРСТХ». В результате разбиения получаем коды слов.

3. Сохранение кодированных данных – код слова привязывается к текущей записи в БД, причем, с целью устранения повторов слов, если этот код в записи уже встречался, он повторно не добавляется:

3.1. Если код слова уже существует в БД, существующий код привязывается к текущей записи, значение поля количества записей, к которым привязан данный код, инкрементируется.

3.2. Если код слова отсутствует в БД, он сохраняется в БД, а значение поля количества записей, к которым привязан данный код, устанавливается равным 1.

Результат подготовки используется для отбора данных на последующих этапах и выступает в роли фильтра данных для этих этапов. Поэтому задачей алгоритма является выявить минимально похожие записи в

БД, учитывая, что в первую очередь должна максимально снижаться вероятность ошибочного отсечения фильтром данных.

Алгоритм позволяет обрабатывать различные варианты транслитерации кириллицы и латиницы, перевода между русским и украинским языком и написания окончаний.

Выбор данных. Перед сравнением необходимо отобрать данные для обработки. В случае отсутствия этапа подготовки данных выбираются все существующие в базе данных записи, применяется прямой перебор каждой записи.

В случае проведения подготовки данных в соответствии с вышеприведенным алгоритмом, выбирается только ограниченный набор записей, что резко снижает алгоритмическую сложность. В таком случае применяется следующий алгоритм выбора данных:

1. Выбираются все коды, привязанные к записи, в убывающем порядке относительно количества записей, к которым привязан каждый код.
2. Все коды делятся на два класса относительно количества записей, к которым привязан каждый код – коды с количеством записей больше или равно заданного порога и меньше заданного порога.
3. На первом этапе для всех кодов с количеством записей меньше заданного порога производится объединение множеств записей, привязанных к этому коду. Так как количество кодов у одной записи ограничено, и количество записей в одном множестве ограничено условием порога, то полученное объединение также ограничено.
4. На втором этапе с множеством записей, полученных на первом этапе, производится пересечение множеств записей, привязанных к кодам с количеством больше или равно заданного порога.

Оценка сложности алгоритма. Рассмотрим алгоритмическую сложность проведения нормализации, подготовки данных, а также выбора и идентификации данных для вариантов полного перебора и выбора подготовленных данных.

Пусть задано N – количество записей в БД и пусть заданы значения, не зависящие от N:

t_s – среднее время выбора записи из БД;

t_f – среднее время выбора записи из БД и сравнения её заданной записью с использованием алгоритма нечёткого поиска;

t_u – среднее время проведения унификации заданной записи;

t_w – среднее время проведения кодирования слова;

N_p – количество записей, возвращаемых в результате выбора подготовленных данных для заданной записи, в соответствии с правилами выбора данных ограничено сверху значением $N_p^{\max}$;

N_w – количество слов в записи, исходя из ограниченности объёма данных одной записи БД, ограничено сверху значением $N_w^{\max}$.

Нормализация данных. Нормализация данных заключается в проведении унификации каждой записи, время подготовки i-й записи не зависит от наличия других записей и составляет:

$$p_i = t_u,$$

тогда время подготовки N записей составляет:

$$P_N = \sum_{i=1}^{N} p_i = \sum_{i=1}^{N} t_u = Nt_u,$$

обозначив константу $C_1 = t_u$, получаем:

$$T_N = C_1 N.$$

Таким образом, алгоритм нормализации данных имеет линейную сложность $O(N)$.

Подготовка данных. Подготовка данных заключается в проведении кодирования каждого слова каждой записи. Учитывая, что при проведении подготовки данных необходимо проверять наличие кода слова в БД, используя поиск с использованием B-tree индексов, время подготовки i-й записи составляет:

$$p_i = N_w t_w + N_w t_s \log_2 i,$$

тогда время подготовки N записей составляет:

$$P_N = \sum_{i=1}^{N} p_i = \sum_{i=1}^{N} (N_w t_w + N_w t_s \log_2 i) = NN_w t_w + N_w t_s \sum_{s\ i=1}^{N} \log_2 i,$$

обозначив константы $C_1 = N_w t_s$, $C_2 = N_w t_w$, получаем:

$$P_N = C_1 \sum_{i=1}^{N} \log_2 i + C_2 N \le C_1 N \log_2 N + C_2 N \le (C_1 + C_2) N \log_2 N .$$

Таким образом, алгоритм подготовки данных имеет логарифмическую сложность $O(N \log_2 N)$.

Идентификация данных с полным перебором. При использовании алгоритма прямого перебора, который заключается в нечёткой идентификации заданной записи с каждой записью в БД, время поиска для $i+1$-й записи составляет:

$$t_{i+1} = i t_f ,$$

отсюда получаем время, необходимое для нечёткой идентификации N записей:

$$T_N = \sum_{i=2}^{N} t_i = \sum_{i=2}^{N} (i-1) t_f = \frac{1}{2} N(N-1) t_f = \frac{1}{2} N^2 t_f - \frac{N}{2} t_f ,$$

обозначив константы $C_1 = \frac{t_f}{2}$, $C_2 = -\frac{t_f}{2}$, получаем:

$$T_N = C_1 N^2 + C_2 N \le (C_1 + C_2) N^2 .$$

Таким образом, алгоритм идентификации с прямым перебором имеет квадратичную сложность $O(N^2)$.

Идентификация данных с выбором подготовленных значений. Так как при идентификации данных с выбором подготовленных записей используется поиск записей с использованием B-tree индексов, то время поиска записей при использовании подготовленных данных для $i+1$-й записи составляет:

$$r_{i+1} = N_w t_s \log_2 i ,$$

тогда время нечёткого поиска с подготовкой записей для $i+1$-й записи:

$$t_{i+1} = r_{i+1} + N_p t_f = N_w t_s \log_2 i + N_p t_f ,$$

отсюда время, необходимое для нечёткой идентификации N записей:

$$T_N = \sum_{i=2}^{N} t_i = \sum_{i=2}^{N} (N_w t_s \log_2 (i-1) + N_p t_f) = N_w t_s \sum_{i=2}^{N} \log_2 (i-1) + (N-1) N_p t_f ,$$

обозначив константы $C_1 = N_w t_s$, $C_2 = N_p t_f$, $C_3 = -N_p t_f$, получаем:

$$T_N = C_1 \sum_{i=2}^{N} \log_2 (i-1) + C_2 N + C_3 \le C_1 N \log_2 N + C_2 N + C_3 \le (C_1 + C_2 + C_3) N \log_2 N .$$

Таким образом, алгоритм идентификации с предварительной подготовкой имеет логарифмическую сложность $O(N \log_2 N)$. Легко видеть, что в случае обработки больших массивов данных идентификация данных с предварительной подготовкой является предпочтительной и часто единственно возможной. Оценка количества операция для различных значений *N* показана в табл. 2.

Недостатком предварительной подготовки данных является увеличение вероятности возникновения ошибок второго рода (false negatives). Ошибка второго рода возникает, если система даёт заключение о том, что записи отличаются, хотя в реальности они похожи. Данная ошибка возможна вследствие отсеивания на этапе подготовки тех записей, которые при нечётком сравнении дали бы более высокую степень подобия. Однако выигрыш в производительности компенсирует этот недостаток.

Нечёткая идентификация данных. Для полученного в результате выбора данных набора записей, для каждой записи набора проводится подсчёт подобия записи набора и заданной записи. Для юридических лиц производится по алгоритму Q-грамм в варианте биграмм. Для физических лиц производится по модифицированному алгоритму расстояния Левенштейна. Из набора отбираются записи, степень подобия которых выше заданного порога для тестового набора записей был подобран порог 0,4 для юридических лиц и 0,65 для физических лиц.

Метод Q-грам. Это один из методов сравнения текстовых строк [7], который показывает хорошие результаты для таких видов данных, как фамилии или названия фирм. Суть метода заключается в том, что сравниваемые строки режутся на подстроки длины Q (Q-грамы), далее осуществляется сравнение наборов подстрок и, исходя из количества совпавших элементов, делается вывод об их степени схожести.

Таблица 2

N (количество записей)	$N^2 - N\log_2 N$ (количество операций)
100000	9998339035,95
200000	39996478071,91
300000	89994541619,11
400000	159992556143,81
500000	249990534215,72
600000	359988483238,22
700000	489986408103,22
800000	639984312287,62
900000	809982198391,07
1000000	999980068431,43

Критериями, в данном случае, могут служить следующие величины:

- количество совпавших подстрок с учетом длин сравниваемых строк;
- количество совпавших подстрок с учетом их позиций в сравниваемых строках;
- количество совпавших подстрок с учетом их позиций в сравниваемых строках и длин сравниваемых строк;
- количество совпавших подстрок с учетом допустимого интервала отклонения их позиций в сравниваемых строках и допустимого отклонения длин сравниваемых строк.

Рассмотрим применение подготовки данных на примере юридических лиц. Пусть задана целевая строка для сравнения – «А/Ф МАГАРАЧ ПО СИД». Также задано эталонное множество юридических и физических лиц (табл. 3). Разделять при сравнении юридические и физические лица по типу нецелесообразно, так как возможны случаи ошибочного указания типа лица. В табл. 3 приведены полное и короткое нормализованные названия, рассчитанные коды слов и степень подобия на основе нечёткого сравнения целевой строки с каждой записью эталонного набора с применением метода Q-грам.

При использовании полного перебора необходимо сравнить целевую запись с каждой записью эталонного набора. Выбор данных с использованием предварительно подготовленных кодов слов позволит уменьшить количество необходимых сравнений и исключить из нечёткого сравнения записи № 3-4. Дальнейшее применение нечёткого сравнения с пороговым значением 0,4 позволяет утверждать о высокой степени подобия целевой записи и записи № 1.

Метод расстояния редактирования. Расстояние Левенштейна (также дистанция Левенштейна или редактирования) в теории информации — это мера разницы двух последовательностей символов (строк) относительно минимального количества операций вставки, удаления и замены, необходимых для перевода одной строки в другую. Метод разработан в 1965 году советским математиком В. И. Левенштейном и назван его именем.

Расстояние Левенштейна активно применяется при поиске и обработке текстов:

- в поисковых системах для нахождения объектов или записей по имени;
- в базах данных при поиске с неполно-заданным или неточно-заданным именем;
- для коррекции ошибок при вводе текста;
- для коррекции ошибок в результате автоматического распознавания отсканированного текста или речи;
- в других приложениях, связанных с автоматической обработкой текстов.

Расстояние редактирования равно минимальному числу элементарных операций редактирования, необходимых для преобразования одной строки в другую. Таким образом, для любой цепочки преобразований одного слова в другое можно определить ее стоимость как количество элементарных операций редактирования данного преобразования.

Дерево решений. К множеству записей, обработанных на предыдущем этапе, применяется система правил в виде дерева решений.

Дерево решений – это структурированное множество правил, которое позволяет классифицировать данные по определенным критериям [6]. Дерево решений является плоским графом. Каждый узел дерева соответствует некоторому варианту сравнения пары полей – совпадает, не совпадает, отсутствие данных. Прохождение по дереву тем или иным путем соответствует степени отдаленности рабочей записи от эталонной. Таким образом, возникает задача сравнения «близости» двух строк. Проиллюстрируем некоторые наиболее характерные ошибки на примере данных ФИО (табл. 4). Подробное описание работы системы приведено в работе [8].

Таблица 3

№ п/п	Полное название (нормализованное)	Краткое название (нормализованное)	Коды слов	Степень подобия
Целевая запись				
1.	А/Ф МАГАРАЧ ПО СИД	–	Х А ПА САТ МАХАРАС	–
Эталонный набор				
1.	АГРОФИРМА МАГАРЧ ИВИВ МАГАРАЧ УКР АКАДЕМИИ АГРАРНЫХ НАУК	АГРОФИРМА МАГАРАЧ ИВИВ МАГАРАЧ	АХРАХАРМА НАХ АХР АХАХ АХАТАМА АХРАРНАХ МАХАРАС МАХАРС	0,47
2.	КФ ЗАТ СК МЕГАРУСС-Д	КФ ЗАТ СК МЕГАРУСС-Д	Х Т САТ СХ МАХАРАС	0,29
3.	ПРИВАТНЕ ПІДПРИЄМСТВО УКРПРОМТЕХСНАБ	ПП УКРПРОМТЕХСНАБ	П ПРАХАТНА ПАТПРАМСТХА АХРПРАМТАХСНАП	0,11
4.	ПП ФЕМІДА-С ПРИВАТ	П-ВО ФЕМІДА-С	ПРАХАТ С П ХА ХАМАТА	0,12
5.	МИГИРИЧ ТАМАРА	–	ТАМАРА МАХАРАС	0,24

Таблица 4

№ п/п	Фамилия	Имя	Отчество
1	Иванов	Александр	Петрович
2	Іванов	Олександр	Петровч
3	Ивнов	Аликсандр Петрович	–
4	!ВАНОВ	АЛЕКСАНРД	ПЕТРОВИЧ
5	ИВАНОВ А.П.	–	–

Реализация. Для реализации приложения интеллектуальной обработки данных была выбрана трёхуровневая архитектура.

Одним из недостатков архитектуры клиент-сервер является то, что любые изменения в реализации функциональности, вызывают изменения в клиентской части приложения, что вызывает необходимость повторного разворачивания приложения. Поэтому уровень пользователя представлен веб-клиентом, чем достигается необходимая гибкость переконфигурирования и развёртывания. Уровень хранения данных представлен реляционным хранилищем данных, реализованным в СУБД Oracle. В качестве технологии для разработки бизнес-логики приложения была выбрана переносимая объектно-ориентированная среда программирования Java.

В результате было создано приложение, которое включает систему идентификации, реализованную на базе программного обеспечения Blaze Advisor 5.5 компании Fair Isaac, настройку и редактирование системы правил которой осуществляет специально подготовленный пользователь – эксперт (рис. 2).

Рис. 2. Архитектура приложения

Пользователи-аналитики работают с программным комплексом с помощью интерфейса *АРМ ведения реестра лиц*. После загрузки данных из внешних источников проводится предварительная обработка данных, что включает в себя нормализацию и очистку данных. Далее проводится идентификация лиц с применением системы правил и методов нечёткого сравнения строк, выше описанных. После этого производится загрузка очищенных данных в автоматическом либо полуавтоматическом режиме.

Выводы

В работе рассматривалась актуальная проблема очистки и нормализации данных. При заполнении хранилища данными, которые приходят из различных систем возникает вопрос их согласованности и корректности. Информация из хранилища используется для проведения аналитической работы, анализа и прогноза функционирования всего предприятия, поэтому существование в нем противоречивых или загрязненных данных может существенно ухудшить эффективность аналитической работы. Одной из характерных проблем является идентификация физического или юридического лица, по которому пришла новая информация. Необходимо выяснить существует ли это лицо уже в хранилище и если да, то обнаружить его. Ситуация усложняется тем, что информация может быть неполна или ошибочна. Так паспортные данные, ФИО, идентификационный код могут отсутствовать или быть набраны с ошибкой. Особенно часто это случается с фамилиями, которые к тому же могут быть введены на русском, украинском или английском языке. Вторая сложность заключается в том, что хранилище может содержать информацию о нескольких миллионах лиц (отсортированных по внутреннему ключу), и при поиске прямым перебором лица, например, по фамилии необходимо провести миллион сравнений.

Таким образом, возникает необходимость разработки системы сравнения, которая могла бы надежно и эффективно искать лицо в хранилище с учетом возможно ошибочных или недостающих полей данных. При этом необходимо использование интеллектуальных методов обработки данных, позволяющих провести нечеткое сравнение без участия человека.

Предложен подход к построению системы очистки данных, поступающих в хранилище. К основным элементам относятся: система предварительной подготовки данных, система нечеткого сравнения, идентификация на основе деревьев правил. Описаны алгоритмы предварительной обработки и сравнения. Проведена оценка сложности алгоритма и продемонстрировано значительное улучшение производительности при применении предварительной обработки.

1. *Спирли Э.* Корпоративные хранилища данных. Планирование, разработка и реализация. Т. 1: Пер. с англ. – М.: Издательский дом «Вильямс», 2001. – 400 с.
2. *Rahm E, Do H.H.* Data Cleaning: Problems and Current Approaches // IEEE Techn. Bulletin on Data Engineering, Dec. – 2000.
3. *Kim W.* On Three Major Holes in Data Warehousing today // J. of Object Technology, 2002. – Vol. 1, N 4. – P. 39 – 47.
4. *Kimball R., Caserta J.* The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming and Delivering Data. Wiley 2004.
5. *Кадощук И.Т., Липчинский Е.А.* Обзор технологии хранилищ данных. http://www.olap.ru/basic/genstore.asp.
6. *Орлов Д.* Подсистема сопоставления записей в хранилище данных. http://www.olap.ru/basic/CompareLog_dw.asp.
7. *Graham A. Stephen* String Search. Technical Report TR-92-gas-01. School of Engineering Science, University College of North Wales, Gwynedd, UK, October 1992, русский перевод: http://www.3ka.mipt.ru/vlib/books/Programming/ComputerScience/StryngAnalysis/index.html.
8. *Гула А.Ю., Игнатенко А.П., Перечинский И.А.* Применение методов интеллектуальной обработки в задачах очистки хранилища данных // Сб. тр. конф. Системы поддержки принятия решений. Теория и практика (7 июня, 2007). – Киев; 2007. – С. 145 – 148.