Федеральное государственное учреждение
«ФЕДЕРАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ЦЕНТР
«ИНФОРМАТИКА И УПРАВЛЕНИЕ»
РОССИЙСКОЙ АКАДЕМИИ НАУК»

На правах рукописи

Тищенко Владимир Александрович

Методы построения многоуровневого классификатора по лексикографическому признаку применительно к ключевому уровню массива ООСУБД НИКА

Специальность 05.13.01 – Системный анализ, управление и обработка информации
(информационно-вычислительное обеспечение)

ДИССЕРТАЦИЯ
на соискание учёной степени
кандидата технических наук

Научный руководитель —
доктор технических наук
Соловьёв Александр Владимирович

Москва
2020

СОДЕРЖАНИЕ

Светлой памяти профессора Николая Евгеньевича Емельянова, руководителя и благодетеля моего посвящается.

“Как будто нельзя число четыре обозначить как дважды два, прямые линии как линии без изгибов, родину как отечество, и что-нибудь другое иначе, когда одно и то же может быть выражено различными словами. По правде говоря, подобает знать, что буквами, слогами, речью, знаками и словами мы пользуемся ради чувств.”

Священномученик Дионисий Ареопагит

Введение

Проблема быстрого интерактивного поиска данных в массиве в форме классификатора по лексикографическому признаку формулируется в следующем виде. Существует информационная потребность в оптимальном классификаторе по причине увеличения объёмов данных и повышение сложности структуры данных. Сформированные в виде баз данных они составляют основу информационных систем в интернет, поэтому при работе с ними, особенно с использованием сенсорных устройств без клавиатуры, для пользователя критично количество “кликов” при поиске ключа. С другой стороны, проблема построения классификатора не решена, несмотря на многочисленные попытки в виде однобуквенных, двухбуквенных алфавитных классификаторов или диапазонных классификаторов и т.п. Такая противоречивая ситуация и определяет необходимость решения важной научно-технической проблемы построения оптимального классификатора как альтернативы к запросной системе. При изменении БД необходимо динамически перестраивать классификаторы, являющиеся способом доступа к ключевым массивам. В более простых случаях, когда, например, строится классификатор для БД товаров потребления достаточно разделить товары на различные классы, но в случае фактографических БД со сложной структурой, в которых есть более сотни полей и по этим полям построены индексы с большим количеством ключей для объектов одного класса, например по фамилиям, то для организации доступа к таким ключевым массивам необходимо построение классификаторов по лексикографическому признаку.

Что касается проблемы организации доступа к базе данных (БД), то обычная практика такова, что БД хранения рассматривается отдельно от системы доступа. При этом обычно исследуются запросы, их оптимизация, среднее время, минимальное время, минимизация этих временных характеристик. Остаётся не изученной проблема интерактивного доступа и проблема оптимизации такого доступа. Распространённой формой

интерактивного доступа к базам данных служит поисковая форма, содержащая запросные поля. Существующий способ поиска в ключевом массиве в виде поля ввода необходимо дополнить **классификатором по лексикографическому признаку**, который предоставляет визуальный просмотр класса ключей на выбранный префикс и даёт возможность обнаружить наличие или отсутствие префиксов искомых ключей, что ускоряет поиск. Такая же проблема интерактивного доступа существует и для ключевых массивов больших объёмов структурно сложных БД, опубликованных в интернет. Широкое распространение мобильных устройств без клавиатуры ставит проблему удобного интерфейса с пользователем. Классификатор предоставляет такой интерфейс в виде иерархического списка существующих префиксов-подсказок к ключам массива.

Стандартным интерфейсом для просмотра реляционных баз данных, в том числе в интернет, является табличная форма представления данных с выбором столбцов (например, Oracle, MS Access и др.). В случае поисковых систем в интернет (google, yahoo, yandex и др. — проиндексированы миллиарды страниц, индексы содержат петабайты данных) стандартным способом отображения результатов поиска является список, разделённый на части. Навигация в списке с большим количеством текстовых ссылок на документы при поиске требуемой информации может быть упрощена посредством классификатора. Для более удобного способа навигации по данным, представленным в виде таблицы, списка текстовых ключей или записей с текстовыми ключами служит классификатор, предоставляющий способ обзора массива текстовых ключей и позволяющий найти по префиксам искомые ключи, упорядоченные в алфавитном порядке. В случае библиографических баз данных встречаются такие классификаторы, но чаще всего однобуквенные, что является недостаточным при больших объёмах массивов, разделённых на группы ключей. Если брать группы по 20 ключей, то для **оптимальной навигации** по массиву объёмом в несколько тысяч ключей

уже недостаточно однобуквенного указателя. Под оптимальной навигацией понимается минимальное число переходов пользователя в классификаторе по ссылкам при поиске по ключу. Примером многоуровневого классификатора может служить БД “Жертв политического террора в СССР” (3,1 млн. имён — 2019г.). Здесь применяется трёхуровневый алфавитный классификатор. Однако он не оптимален в том смысле, что на втором уровне здесь происходит деление на трёхбуквенные диапазоны ключей, которые не позволяют определить наличие или отсутствие префиксов внутри диапазонов, а для этого необходимо просматривать третий уровень классификатора. Третий уровень представляет собой список фамилий, на каждую из которых присутствует различное число ключей. Такой классификатор не оптимизирован по общему числу операций, а префиксы выбраны “по смыслу” и им можно пользоваться до тех пор, пока не будут добавлены новые ключи в массив, т.к. он становится не удобным в использовании из-за большого количества одинаковых фамилий. В результате увеличения числа записей в новой версии БД используется только форма ввода, т.к. построенный классификатор стал непригодным из-за большего объёма данных. Вышеизложенное описание проблемы и приведённые примеры показывают, что для БД больших объёмов является **актуальным** построение оптимального классификатора, который строится автоматически по исходному массиву при минимизации общего числа операций в классификаторе. **Актуальность темы** можно охарактеризовать следующими положениями:

- Электронная публикация структурно сложных БД больших объёмов.
- Изменение БД требует динамической перестройки классификатора.
- Широкое распространение мобильных устройств без клавиатуры.
- Список не даёт обзора ключей на предмет существования или отсутствия определённых классов ключей на заданные префиксы.
- Системы с многоуровневыми классификаторами по лексикографическому признаку существуют и востребованы, но не оптимизированы по числу переходов пользователя.

- Классификаторы по различным категориям в виде онтологий применяются в информационных системах и являются естественными элементами пользовательского интерфейса.

Также можно сделать вывод о том, что данная область **малоизучена**, т.к. не существует на данный момент стандартных средств построения такого классификатора, библиографические БД ограничиваются однобуквенным указателем, существующие системы не используют специально разработанных методов построения классификаторов. В диссертации предлагается решение проблемы в виде метода построения оптимального классификатора, который является **новаторским**. Метод имеет существенные достоинства по **сравнению** с другими методами интерактивного доступа.

- Список, разделённый на части, не даёт возможности обзора существующих ключей массива и быстрой навигации.
- Классификатор дополняет онтологии, используемые для формализации областей знаний.
- Приведённый пример системы, который является единичным, с использованием нескольких уровней классификатора не оптимизирован по числу нажатий пользователя. Возможная причина узкого распространения таких систем в отсутствии метода автоматического построения оптимального классификатора.
- Оптимизированный классификатор построен на основе методов формализации задачи интерактивного доступа в виде оптимального классификатора, построенного посредством минимизации функционала общего числа операций (на примере СУБД НИКА). Такой пользовательский интерфейс даёт наиболее быстрый способ перехода к искомым ключам в виде “префиксов”-подсказок.

Концептуально классификатор представляет собой “схему” ключевого уровня массива и является элементом базы знаний в данной предметной области, является средством визуализации ключевого уровня массива, а также

альтернативным способом быстрого перехода по ключу в массиве по отношению к полю ввода. Существует множество подходов для разделения объектов на классы. Основу проблемы классификации по лексикографическому признаку составляет префиксное дерево, представляющее собой пространство состояний классификатора. Это же дерево (или древовидная структура trie) лежит в основе лучевого поиска. Лучевой поиск — это технология быстрого многопутевого принятия решения по индексу, в котором содержатся буквы, имеющиеся в ключах исходного массива. Особенность классификатора состоит в том, что структура trie лучевого поиска, размещаемая в памяти компьютера, берётся за основу организации интерактивного интерфейса в форме многоуровневого иерархического классификатора.

В первой половине 60-х годов Эдвард Сассенгат составил комбинированную стратегию цифрового поиска, сочетающую бинарный и последовательный поиск. Лучевая память в виде древовидной структуры trie впервые была создана Бриандe. Моррисон предложил сжатую структуру trie в виде дерева PATRICIA без однопутевых ветвей. Такие структуры могут применяться для организации кэша. В настоящее время trie также активно применяется, например, в 2015 г. вышла статья[1] нидерландских специалистов о построении словаря RDF-данных с использованием упомянутых структур.

Проблема доступа к данным в виде некоторой разновидности структуры trie, размещённой в памяти, достаточно хорошо изучена. При этом остаётся не рассмотренной проблема, связанная с организацией интерактивного доступа к данным. Организация многоуровневого классификатора по лексикографическому признаку на основе префиксного дерева как пользовательского интерфейса ключевого массива позволит сделать некоторый шаг в этом направлении. Применение комбинированной стратегии лучевого

[1] Hamid R. Bazoobandi , Steven Rooij , Jacopo Urbani , Annette Teije , Frank Harmelen , Henri Bal, A Compact In-Memory Dictionary for RDF Data, Proceedings of the 12th European Semantic Web Conference on The Semantic Web. Latest Advances and New Domains, May 31-June 04, 2015

поиска Сассенгата и сжатого дерева PATRICIA Моррисона составляют основу для построения классификатора.

Относительно **степени разработанности** метода необходимо отметить, что метод лучевого поиска Сассенгата неоднократно использовался и модифицировался в различных исследованиях вплоть до настоящего времени. Применение префиксного дерева в качестве интерфейса с пользователем является новым способом использования этой структуры. Разработанные методы адаптируют метод Сассенгата для организации пользовательского интерфейса. Известно, что среднее время доступа к таким структурам для n строк составляет O(log$n$), а пространственная сложность порядка O(n). В 2005г. Резник на основании более поздних исследований выделил класс структур trie со значительно более быстрым временем доступа порядка O(log log n) и изучил его асимптотические свойства. К таким структурам относятся LC-trie (сжатые по уровню структуры trie). Оптимальный классификатор является сжатым по поддеревьям деревом trie и играет роль интерфейса с пользователем, поэтому в нём существенна временная, а не пространственная сложность. В разделе 2.1 диссертации получено, что временные показатели классификатора лучше обычного дерева trie, но хуже сжатого по уровням дерева LC-trie. Дерево LC-trie не применимо в виде интерфейса с пользователем, т.к. должно быть заполнено всеми буквами алфавита на заданное число уровней. Применение структуры, используемой в лучевом поиске, для организации интерактивного доступа к ключевому массиву имеет свои особенности. Для построения оптимального классификатора необходимо минимизировать общее число операций в алфавитном классификаторе при поиске по ключу. При этом необходимо получить оптимальное число переходов при поиске ключа в классификаторе и число ключей в списке на выбранный префикс.

Ещё одним важным аспектом построения оптимального классификатора является применение **алгоритмов минимаксного размещения букв** или их сочетаний по каналам обслуживания при параллельном подсчёте числа

операций. Самые быстрые алгоритмы дают удвоенное оптимальное решение за линейное время. Улучшить этот класс алгоритмов нельзя, т.к. было доказано, что при коэффициенте аппроксимации меньшим, чем 2 получается NP-трудная задача[2]. В виду небольшого числа однобуквенных и двухбуквенных префиксов и числа каналов обслуживания возможно применение аппроксимации 1+ε или даже алгоритмов точного решения задачи.

Основная **цель** диссертационной работы заключается в разработке методов построения многоуровневого классификатора по лексикографическому признаку, адаптирующего структуру лучевого поиска Сассенгата для организации интерактивного доступа к ключевому массиву для повышения эффективности, надёжности и качества гипертекстовой системы.

Для её достижения требуется решить следующие **задачи**:

1. На основе структуры лучевого поиска Сассенгата необходимо проанализировать неравномерность распределения ключей массива по n-граммным префиксам посредством средней длины префикса классификатора с использованием модельных неравномерных распределений на примере индексных текстовых полей в ООСУБД НИКА.

2. Для исследования вида случайных распределений величин длины префикса класса и числа ключей в классе выделить характерные из семейства распределений, зафиксировав максимальное число ключей в классе.

3. Построить регрессионную модель зависимости средней длины префикса класса от максимального числа ключей в классе, позволяющую сопоставить классификатору среднюю длину префикса.

4. Выбрать оптимальный классификатор посредством минимизации функционала общего числа операций в классификаторе на заданных диапазонах максимального числа ключей в классе, разделённого на равные группы, и числа

[2] *Gonzalez T.* Clustering to minimize the maximum intercluster distance // Theoretical Computer Science. — 1985. — Vol. 38. — P. 293–306.

ключей в группе. Оптимальному классификатору сопоставить среднюю длину префикса из регрессионной зависимости.

5. Разработать программу для расчёта оптимального значения функционала общего числа операций в классификаторе, соответствующего оптимальному классификатору, с использованием нескольких параллельных каналов обработки буквенных префиксов с близкими средними частотами на основе подходящих минимаксных алгоритмов размещения объектов.

Основные **положения, выносимые на защиту**:

1. Метод модельных распределений для анализа неравномерности распределения ключей массива по n-граммным префиксам на основе префиксного дерева сочетаний.

2. Функции плотности распределения для случайных величин длины префикса класса и числа ключей в классе при фиксированном максимальном значении числа ключей в классе в виде асимптотического разложения, основанного на нормальном распределении. Распределение длины префикса получилось мультимодальным, а распределение числа ключей в классе унимодально при небольшом максимальном значении числа ключей.

3. Метод построения классификатора на основе регрессионной зависимости средней длины префикса алфавитного классификатора от максимального числа ключей на любой префикс методом ортогональных полиномов Чебышева.

4. Метод построения оптимального классификатора на основе математической модели и алгоритма выбора оптимального классификатора с использованием префиксного дерева сочетаний посредством минимизации функционала общего числа операций в классификаторе.

5. Программные модули, реализующие подходы и алгоритмы, представленные в диссертации.

Научная новизна, выносимых на защиту результатов состоит в следующем:

– в рамках диссертационной работы впервые формулируется и решается задача адаптации структуры лучевого поиска Сассенгата для организации интерактивного доступа к ключевому массиву в глобальных гипертекстовых системах, требующая новых моделей и подходов;

– впервые представлены в аналитическом виде характерные случайные распределения длины префикса класса и числа ключей в классе, выбранные из семейства случайных распределений, для фиксированного максимального числа ключей в классе с использованием разложения в ряд Эджворта;

– впервые предложена модель регрессии на ортогональных полиномах для зависимости средней длины ключа от максимального числа ключей в классе для определения средней длины ключа оптимального классификатора;

– впервые предложен алгоритм построения оптимального классификатора по лексикографическому признаку на основе префиксного дерева сочетаний при минимизации функционала общего числа операций в дереве, в результате которого также определяется максимальное число ключей в классе оптимального классификатора.

Методы исследования, используемые в диссертационной работе, включают в себя системный анализ, методы математической статистики, регрессионный анализ, методы оптимизации, методы классификации и кластеризации.

Объектом исследования являются классификаторы по лексикографическому признаку на основе префиксных деревьев.

Предметом исследования являются методы построения классификаторов по лексикографическому признаку и их оптимизация для организации интерактивного доступа к ключевому массиву.

Теоретическая и практическая значимость работы. Диссертационная работа имеет как теоретическую, так и практическую значимость.

Теоретическая значимость работы заключается в первую очередь в постановке исследуемой задачи, предложенном оптимальном классификаторе

по лексикографическому признаку, а также в разработанных моделях и алгоритмах для описания свойств оптимального классификатора на основе префиксного дерева сочетаний в виде функционала общего числа операций и задачи построения регрессионной зависимости средней длины префикса классификатора от максимального числа ключей в классе. Полученные результаты могут быть использованы для дальнейшего развития науки в данной области.

Практическая значимость диссертационной работы подтверждается тем, что её результаты внедрены (см. справки о внедрении) в информационно-справочную систему на основе электронной публикации материалов ежегодника “Системные исследования” за более чем 25-летний период издания. Другим применением является автоматизированная система управления электронными публикациями баз данных «Философия и методология науки в журнале ”Вопросы философии”». Система включает в себя коллекцию из полутора тысяч статей за более чем 50-летний период издания. Наконец, третье применение — это информационно-поисковая система по репрессированным за годы советской власти, содержащая на данный момент более 36 тыс. биографических справок о пострадавших. Система содержит индексы по текстовым полям, представленные в виде многоуровневых классификаторов по лексикографическому признаку с использованием полученных результатов.

В **первой главе** диссертации проводится обзор основных методов классификации и делается постановка задачи построения оптимального классификатора для организации интерактивного доступа к ключевому массиву сложноструктурированной БД. **Вторая глава** содержит методы построения алфавитного классификатора и функции плотности распределения случайных характеристик классификатора: метод модельных распределений для анализа неравномерности распределения ключей массива по n-граммным префиксам на основе префиксного дерева сочетаний, функции плотности распределения для случайных величин длины префикса класса и числа ключей в классе, метод на

основе регрессионной зависимости средней длины префикса алфавитного классификатора от максимального числа ключей в классе. В **третьей главе** описывается основной результат — метод построения оптимального классификатора на основе математической модели и алгоритма выбора оптимального классификатора с использованием префиксного дерева сочетаний посредством минимизации функционала общего числа операций в классификаторе. В **четвёртой главе** излагаются теория, методы и средства построения гипертекстовой системы для ООСУБД НИКА, позволяющей осуществить практическую реализацию оптимального классификатора. **Пятая глава** состоит из примеров применения классификатора.

Результаты, связанные с оптимальным классификатором, получены на основе структуры префиксного дерева и её модификаций, которые достаточно подробно разработаны в **литературных источниках** (Fredkin,1960; Sussenguth,1963; Morrison,1968; Patt,1968; Stanfel,1970; Bayer,1972,1977; Andersson,1993; Nilsson,1998; Ferragina,1999; Reznik,2002,2005; Таранов,2011; Bazoobandi,2015; Wen,2016; Prokopec,2018).

Основные положения и результаты диссертационной работы **прошли апробацию** на следующих международных научных конференциях: XVIII Международной конференции “Advances in Science and Technology”, Москва, 2019; "XXII– XXV Ежегодных международных конференциях ПСТГУ", секции факультета ИПМ 2011-2014; конференции "Информационные системы в науке - 95", Москва, 1995; конференции "The 3-rd international workshop on "Advances in databases and information systems", ACM SIGMOD, 1996; III Международной конференции "Развитие и применение открытых систем", 1996. Помимо научных конференций результаты диссертационной работы были обсуждены на научно-исследовательском семинаре в ОИВТА РАН.

Основные результаты опубликованы в 20 публикациях, в том числе патент на изобретение (Воробьёв,Сомин,97-07-90055; Тищенко,2003,2013,2018,2019x2; Емельянов,Богданов,96-01-01840;

Емельянов,Тищенко,1995,1997,1999,2009,2010; Emelyanov,Tishchenko,1996; Емельянов,Муханов,1996х2; Соловьёв,Тищенко,2018; Чернозуб,Емельянов,2012; Емельянов,Садовский,1997; Арлазаров,Тищенко,2019), из них 13 в соавторстве.

Глава 1. Обзор методов классификации. Постановка задачи

1.1. Основные проблемы методов классификации

Классические способы деления объектов на классы изложены в монографии Кендалла и Стьюарта (Кендалл, 1976, с.438). Они определяют три вида такой процедуры — дискриминация, классификация и разбиение. В случае дискриминации существование классов дано, в последних двух случаях выявление классов составляет задачу. Последние два случая различаются только тем, что классификация представляет собой естественное разделение объектов на классы в соответствии с законами природы, а разбиение – это искусственное разделение объектов искусственной природы на классы в соответствии с некоторым логическим правилом (Микони, 2016, с.69). Микони С.В. определяет математическую модель класса, как некоторое множество X, которое состоит из всех элементов x, обладающих некоторым заданным свойством или свойствами. Эти свойства математически формулируются в виде логического правила. Микони С.В. на с.78 утверждает, что искусственная классификация также может быть научной, как и естественная. “В цитированных источниках только естественные классификации рассматриваются как научные. Однако помимо законов природы существуют законы человеческого мышления… Поэтому, на наш взгляд к научным можно относить и те искусственные классификации, которые согласуются с этими законами. Непременными условиями научности следует также считать соответствие частных целей классификации ее общей цели и подтверждение практикой.” Он отмечает на с.76, что “любая классификация, имеющая опытное или теоретическое начало, подвергается многократной доработке для того, чтобы отразить закономерности предметной области”. Орлов А.И. (Орлов, 2006, см.3.2.4) выделяет два критерия “естественности” или научности классификации. Во-первых, естественная классификация должна быть устойчивой при переходе к различным алгоритмам автоматической

классификации (например, алгоритмам ближайшего и дальнего соседа). Во-вторых, естественная классификация должна давать возможность прогноза, предсказания новых свойств, сжатия информации и т.д. Орлов А.И. (Орлов, 2006, см.3.2.4) разграничивает задачи построения естественной и искусственной классификации, отмечая, что они “принципиально различны, хотя для их решения могут применяться одни и те же алгоритмы.” Существенным является то, что методы группирования в обоих случаях могут быть одинаковыми. Здесь он выделяет проблему, связанную с возможностью разбиения на классы. Если “совокупность объектов достаточно однородна и не разбивается на резко разделяющиеся между собой” классы, то возможна только их группировка, т.е. построение искусственной классификации.

В современной терминологии вводится термин кластеризация (Орлов, 2006, см.3.2.4). Кластеризация отличается от классификации тем, что необходимо выявить кластеры, к которым относятся объекты, а в случае классификации, в современном понимании, существование классов, по которым распределяются объекты, дано изначально. Кластеризация связана с поиском структуры на множестве несгруппированных данных (Мандель,1988,с.10). В случае классификации множество классов задано, а необходимо распределить объекты по этим классам, т.е. по сути дела это есть задача дискриминационного анализа (Кендалл, 1976).

Для достижения основной цели данной работы необходимо построение многоуровневого классификатора и выявление иерархической структуры на основе ключевого массива, упорядоченного по лексикографическому признаку. Для этого необходимо решить задачу кластеризации. Неформальное определение кластеризации или "автоматической классификации": “процесс группирования некоторым образом подобных объектов” (Matteucci,2019). Кластер — множество объектов, “подобных” между собой и “неподобных” объектам других кластеров. Критерием подобия может быть расстояние: два или более объектов принадлежат одному кластеру, если они “близки” в

соответствии с данным расстоянием. Это называется кластеризацией, основанной на расстоянии (Matteucci,2019). Другой вид кластеризации — это кластеризация, основанная на признаковом описании объектов (Matteucci,2019): два или более объектов принадлежат одному кластеру, если для него определен общий признак, объединяющий эти объекты. При формальной постановке задачи кластеризации вводится расстояние между объектами. Пусть X — множество объектов, Y — множество наименований кластеров, причем Y изначально может быть неизвестным. Задана функция расстояния между объектами $\rho(x_1,x_2)$. Имеется конечная обучающая выборка объектов $\{x_1,...,x_m\}$ из X, причем любому x_i из обучающей выборки соответствует некоторое y_j из Y, если известно Y. Тогда кластеризация — это есть некоторая функция $f(x)$, ставящая в соответствие каждому объекту x из множества X некоторое наименование кластера y из множества Y, которое формируется в процессе группирования. Функция $f(x)$ разбивает выборку на непересекающиеся подмножества, называемые кластерами так, чтобы каждый кластер состоял из объектов, близких по метрике ρ, а объекты разных кластеров существенно отличались. Цель кластеризации — определение существенного группирования несгруппированных данных. Не существует абсолютно “наилучшего” критерия, который был бы независим от конечной цели кластеризации (Matteucci,2019). Таким образом, в задачу исследования входит выбор подходящего критерия. Например, можно искать “наиболее характерные” объекты для однородных групп (сжатие данных), “естественные кластеры” и описывать их неизвестные свойства (“естественные” типы данных), полезное и подходящее группирование (“полезные” классы данных), или необычные объекты данных (обнаружение исключений) (Matteucci,2019). Существует ряд требований, предъявляемых к методам кластеризации (Matteucci,2019): масштабируемость, возможность использования различных типов атрибутов, обнаружение кластеров произвольной формы, минимальные требования для ПО к определению входных параметров, возможность работы с исключениями,

нечувствительность к порядку входных записей, многомерность анализа, интерпретируемость и удобство в использовании.

Кластеризация подразделяется на исключающую, нечеткую, иерархическую и вероятностную (Matteucci,2019). В первом случае, если объект принадлежит одному кластеру, то он не может быть включён в другой кластер. Алгоритм *k*-средних (*k*-means) (MacQueen,1967) реализует эту процедуру. Он является наиболее простым методом кластеризации, но обладает рядом недостатков: инициализация средних случайным образом; результат кластеризации зависит от начальных значений средних, от выбора метрики, от заданного числа кластеров; некоторые кластеры могут содержать только центроиды. Не существует общего теоретического решения для поиска оптимального числа кластеров для любого заданного множества данных (Ким, 1989, с.184). При нечеткой кластеризации объект может принадлежать разным кластерам с различной степенью принадлежности. Метод нечеткой кластеризации С-средних (fuzzy c-means) (Bezdek,1981) часто используется при распознавании образов. В этом методе, также как и в случае алгоритма k-средних, на каждой итерации пересчитываются центроиды кластеров и, кроме того, матрица степеней принадлежности объектов кластерам. Преимуществом метода является то, что он учитывает возможную не нулевую степень принадлежности объекта разным кластерам, а не исключает их всех кроме одного кластера, которому принадлежит объект. К недостаткам метода можно отнести то, что нет способа определить число кластеров; при различной инициализации, может возникнуть разное число итерационных шагов, хотя и приводящих к одному результату; алгоритм сходится к локальному минимуму. Полученный минимум целевой функции может отличаться от глобального минимума. В виду большого объема вычислений поиск глобального минимума не может быть сделан с использованием только одного алгоритма нечетких С-средних. В этом случае необходимо привлечение дополнительных знаний из предметной области или других алгоритмов. Например, в статье Тараскиной

(Тараскина,2006) приводится пример совместного использования нечеткой кластеризации С-средних в комбинации с генетическим алгоритмом (Goldberg, 1989), что дает возможность “найти решение задачи кластеризации, близкое к оптимальному”, т.е. “к глобальному минимуму”. Иерархическая кластеризация может быть проведена с использованием объединительных (агломеративные) и разделяющих (дивизивные) алгоритмов. Предполагается наличие вложенных групп (кластеров различного порядка). Агломеративный алгоритм основан на объединении между двумя соседними кластерами, причем в начале процедуры каждый объект рассматривается как отдельный кластер и за несколько итераций все объекты объединяются в один кластер. *Алгоритм Джонсона* выполняет эту процедуру (Matteucci,2019). Разделяющая, или *дивизивная*, кластеризация (divisive clustering) начинается со всех объектов, сгруппированных в единственном кластере. Кластеры делят (расщепляют) до тех пор, пока каждый объект не окажется в отдельном кластере. *Иерархическая дивизивная кластеризация* используется редко (Matteucci,2019). Вероятностная кластеризация оптимизирует приближение данных с помощью модели. Каждый кластер математически представляется некоторым вероятностным распределением непрерывным или дискретным. Все множество данных моделируется смесью этих распределений. *EM-алгоритм (Expectation-Maximization) для нахождения оценок параметров вероятностной модели* реализует вероятностную кластеризацию (Matteucci,2019). К общим недостаткам методов кластеризации (Matteucci,2019) можно отнести следующие: недостаточно удовлетворяют всем требованиям одновременно, при большой размерности и большом объеме данных возникают проблемы, связанные с временной сложностью, эффективность метода зависит от определения “расстояния”, при отсутствии очевидной меры расстояния необходимость определить её (особенно в пространствах с большой размерностью), результат алгоритма кластеризации может быть интерпретирован различными способами.

Актуальные проблемы кластерного анализа разбираются в обзорной статье (Бериков,2008). Проблема выработки критериев качества группировки с учетом экспертных знаний важна в связи с тем, что один и тот же набор объектов может классифицироваться по-разному. Недостаточность знаний об изучаемых объектах “затрудняет формулировку их математических моделей”. Следствием этого является, например, неприменимость вероятностных методов, реализованных, например, в виде EM-алгоритма (Dempster,1977). Большое число измерений “может привести к почти полной неразличимости точек“. Здесь важным вопросом является выбор метрики (Мандель, 1988).

Проблема нелинейности взаимосвязи компонент рассмотрена в статье (Scholkopf,1999). Ядерный (нелинейный) анализ главных компонент (PCA) позволяет выявлять сложные взаимосвязи между переменными. Предлагаемый метод посредством использования ядерных функций эффективно вычисляет главные компоненты в пространствах компонент с высокой размерностью, связанных с входным пространством посредством некоторого нелинейного отображения. Основная идея ядерного PCA в использовании нелинейных ядерных функций вместо стандартного скалярного произведения компонент для косвенного выполнения PCA в пространстве с высокой размерностью, которое нелинейно связано с входным пространством. Указанный нелинейный метод дает лучшие показатели распознавания, чем соответствующий линейный метод, а также производительность нелинейного метода может быть улучшена путем использования большего числа компонент. Вычислительная сложность PCA не возрастает с размерностью пространства компонент. При размерности входного пространства, меньшем, чем число объектов, ядерный PCA по вычислениям является дороже, чем линейный PCA. В сравнении с другими нелинейными методами преимущество данного метода состоит в отсутствии нелинейной оптимизации, что позволяет избежать опасности получить локальный минимум в результате кластеризации. Ядерный PCA путем выбора ядерных функций определяет априори вид нелинейных компонент, которые

являются главными компонентами в пространстве компонент. В методе допустим широкий класс нелинейностей. Нерешенной проблемой является выбор наилучшего ядра для данной задачи.

Другой актуальной проблемой кластерного анализа (Бериков,2008) является необходимость представления результатов в специальном виде, понятном только специалисту в данной предметной области. В статье (Michalsky,1983) предлагается метод концептуальной кластеризации и проводится сравнение этого метода с методами числовых таксономий. Основное различие — кластеризация основана не на математической мере близости объектов, а на основе “концепции членства”. Кластеры — группы объектов, характеризующиеся простыми “хорошо подогнанными” описаниями. Коллекции кластерных описаний оптимизируют заданные глобальные кластерные критерии качества. Концептуальная кластеризация дает более подходящее решение с точки зрения человеческой интерпретации, в то время как в случае методов, связанных с числовыми таксономиями, получаются произвольные кластеры (в 14 из 18 рассмотренных методов). Один из основных результатов исследования состоит в том, что (Michalsky,1983, с.406) “нет ни одной меры численного сходства, которая постоянно приводила бы к дендрограммам, имеющим простую концептуальную интерпретацию”. Концептуальная кластеризация также использует определенную меру подобия объектов, но совсем другого вида — эта мера учитывает не только расстояние между объектами (как в стандартных методах кластеризации), но также их связь с другими объектами, их связь с некоторыми предопределенными концепциями (“конъюнктивными описаниями”). Результатом использования такой “концептуально-зависимой” меры сходства является значительное возрастание вычислительной сложности метода, и, следовательно, времени выполнения программы. В сравнении с методами числовых таксономий время, затрачиваемое на алгоритм концептуальной кластеризации, больше в 100 раз. В методе могут быть использованы только номинальные или порядковые

переменные. Метод учитывает только те переменные, которые были включены во входные данные. В отличии от описанного метода метод (Назаров, 2014) рассматривает объекты, характеризуемые нечеткими параметрами. Отмечается, что существующие алгоритмы нечеткой кластеризации, например c-means, формируют кластеры, объекты которых могут с различной степенью принадлежности относиться к различным кластерам одновременно. При этом большинство алгоритмов не учитывает нечеткую природу самих объектов. Рассматриваемый метод может работать с параметрами, заданными в нечеткой форме в виде функций принадлежности. Наибольшее предпочтение отдается кусочно-линейным функциям принадлежности в виду их большей разделяющей способности, чем П-образным функциям принадлежности. Формула оценки полезности концептуальной кластеризации была модифицирована для объектов с нечеткими параметрами. Метод показал лучшую точность, чем методы EM (Dempster,1977) и g-means (Michalsky,1983), который является расширением метода k-means и автоматически пытается определить число кластеров посредством проверки на нормальность распределения. Важно отметить, что автор статьи (Назаров, 2014) подчеркивает, что “время выполнения программы, реализующей разработанный метод кластеризации объектов с нечеткими значениями параметров, растет экспоненциально относительно количества обрабатываемых объектов”.

В статье (Бериков,2008) выделяется проблема, связанная с нахождением глобального экстремума у критерия качества группировки. При этом отмечается, что классические методы кластерного анализа не гарантируют нахождения строго-оптимального решения, а алгоритм полного перебора различных вариантов разбиения объектов по группам имеет "трудоемкость, экспоненциально зависящую от размерности". Для поиска оптимального решения предлагается использовать генетический алгоритм и нейронные сети. Здесь возникают другие проблемы, связанные с оператором рекомбинации в случае генетического алгоритма. Это недействительность группировочных

решений, когда число кластеров в полученных в результате применения оператора рекомбинации вариантах группировок уменьшается, и нечувствительность к контексту, "когда одно и то же группировочное решение кодируется разными последовательностями", а также трудность интерпретации этого оператора с точки зрения предметной области. Основные недостатки нейронных сетей — сложная структура, плохая интерпретируемость и долгое время обучения. Здесь возникает та же проблема интерпретации параметров сети (весов) с точки зрения предметной области. Существуют проблемы "сходимости, устойчивости, локальный минимум и корректировка параметров" (Манжула,2011), в виду этого нейронные сети могут использоваться вместе с генетическим алгоритмом. К существенному недостатку (Манжула,2011) можно отнести то, что "окончательное решение зависит от начальных установок сети и его практически невозможно интерпретировать в традиционных аналитических терминах". Использование нечетких нейронных сетей имеет недостаток (Манжула,2011), состоящий в "априорном определении компонентов". Общим недостатком нейронных сетей является "сложность понимания процесса получения сетью результата".

Наконец, проблема устойчивости группировочных решений (Бериков,2008) приводит к идее комбинации различных методов. Конечный результат группировки может зависеть от "начальных условий, порядка объектов, параметров работы алгоритмов". Для улучшения результатов группировки, например, алгоритм k-средних может быть запущен несколько раз с различными начальными условиями. В статье (Fred, 2005) подчеркивается, что трудно выбрать единственный алгоритм кластеризации, учитывающий все виды кластерных форм, структур и размеров, среди сотни существующих в настоящее время таких алгоритмов, а также решить какой алгоритм является наилучшим для заданного множества данных. Рассматривается N возможных разбиений данных на кластеры $P = \{P^1, P^2,\ldots, P^N\}$ и необходимо найти “оптимальное” разбиение данных на кластеры P^*, используя доступную

информацию об N имеющихся разбиений данных на кластеры. Пусть k^* — число кластеров в P^*. Тогда P^* должно удовлетворять следующим свойствам: a) согласованность с кластерным ансамблем P; b) надежность кластеризации; c) хорошее соответствие известной информации, если такая информация доступна. Первое свойство обозначает согласование объединенного разбиения P^* с отдельными разбиениями P^i, i=1,…,N. Свойство надежности предполагает, что число кластеров и включение объекта в кластер для разбиения P^* существенно инвариантны к небольшим изменениям в ансамбле P. Последнее требование предполагает знание некоторой основной информации об объектах в предметной области, что может показаться противоречащим процессу кластеризации без обучения. Однако обозначения кластеров, если такая информация доступна, используются только как средство дополнительной проверки рассматриваемых методов. Относительно величин k_i (число кластеров разбиения P^i) и k^* не делается никаких предположений. Предполагается, что результат объединенного разбиения P^* будет лучше объяснять природу группирования в сравнении с результатами отдельных разбиений P^i. Идея метода состоит в объединении результатов многократных процессов кластеризации в одно разбиение на кластеры путем просмотра каждого результата как независимого правила для разбиения на группы. Здесь возникают проблемы, связанные с генерацией кластерного ансамбля, объединением правил разбиений и извлечением согласующегося разбиения на основании объединенного правила. Метод “накопления правил” определяет новую матрицу подобия объектов для кластерного ансамбля. В рамках этого метода могут быть использованы различные методы кластеризации. В данном случае исследовались иерархические агломеративные алгоритмы одиночкой связи (“ближайшего соседа”) и средней связи для получения объединенного разбиения данных. Для анализа полученного результата используется критерий оптимальности, основанный на взаимной информации. Анализируется метод “накопления правил” на основе метода k-средних (Arthur, 2006) в свете

предложенного критерия оптимальности. Полученные результаты для искусственных и реальных данных показали способность метода определять кластеры произвольных форм и размеров. В сравнении с аналогичными комбинированными методами данный метод дал лучшие результаты. Метод может дать еще лучшие результаты, если вместо k-средних (Arthur, 2006) использовать более мощные методы для разбиения сложных множеств данных. В статье (Strehl, 2002) группировочные решения представлены в виде гиперграфа. Вершины гиперграфа — это исходные объекты. Ребро гиперграфа — это кластер в одной из группировок. Ищется разбиение гиперграфа по минимальному числу гиперребер. В заключении статьи (Бериков, 2008) на основании перечисленных актуальных проблем кластерного анализа делается вывод о необходимости дальнейшего развития комбинированных методов кластеризации.

Важной проблемой кластеризации является выбор расстояния между объектами. Для данных большой размерности в качестве расстояния может быть взята метрика Миньковского (Мандель, 1988, с.31): $d_p(x_i,x_j) = (\Sigma|x_{ik}-x_{jk}|^p)^{1/p}$, k=1,...,d, где d является размерностью данных. При p=2 это расстояние представляет собой Евклидово расстояние. Не существует общего метода выбора расстояния для любого приложения. О выборе расстояния для кластеризации объектов см. статью (Бериков,2008). Мандель И.Д. на с.28 рассматривает проблему адекватности мер близости. Он определяет два понятия: расстояние (метрику) $d(x_i,x_j)$ и меру близости $\mu(x_i,x_j)$ ($0\leq\mu(x_i,x_j)\leq1$), затем приводит пример простого перехода от расстояния к мере близости: $\mu(x_i,x_j)=1/(1+ d(x_i,x_j))$. Дальнейшие рассуждения он проводит в терминах меры близости $\mu(x_i,x_j)$, как порожденной некоторым расстоянием. При этом он отмечает, что аксиома неравенства треугольника для расстояния $d(x_i,x_j)$ не является конструктивным требованием, “и в принципе пригодны измерители близости, не удовлетворяющие неравенству треугольника”. Пример расстояния, для которого не выполняется неравенство треугольника — это расстояние

Кульбака-Лейблера (Кульбак, 1967), которое является несимметричной мерой удаленности одного случайного распределения от другого с заданными функциями плотности вероятности $f_1(x)$ и $f_2(x)$: $I(1|2)=\int f_1(x)\log(f_1(x)/f_2(x))\mathrm{d}\lambda(x)$, где. $\lambda(x)$ — вероятностная мера, интеграл берется по всему выборочному пространству. Существует симметричный аналог расстояния Кульбака-Лейблера (Кульбак,1967): $J(1|2)=\int(f_1(x)-f_2(x))\log(f_1(x)/f_2(x))\mathrm{d}\lambda(x)$, являющийся мерой расхождения или "мерой трудности различения" (Кульбак,1967, с.14) соответствующих вероятностных мер, определенных на заданном выборочном пространстве. Основные трудности при определении близости — неоднозначность выбора способа нормировки и определения расстояния между объектами. Адекватной считается статистика, которая не меняется при допустимых преобразованиях шкал (например, коэффициент корреляции) или меняется контролируемым образом (например, среднее арифметическое). Формальное определение адекватности — адекватной называется такая мера сходства, которая удовлетворяет аксиомам: $\mu(x_i,x_j)$ — непрерывна; $\mu(x_i,x_j)=\mu(x_j,x_i)$; $0\le\mu(x_i,x_j)\le1$, $\mu(x_i,x_j)=1\leftrightarrow x_i=x_j$ и, если $\mu(x_i,x_j)\le\mu(x_k,x_l)$, то $\mu(x_i',x_j')\le\mu(x_k',x_l')$. Здесь x_i', x_j', x_k', x_l' — значения показателей, полученные из x_i, x_j, x_k, x_l при допустимом преобразовании шкал, т.е., таком, которое не меняет отношения нестрогого порядка на множестве пар объектов. Далее Мандель И.Д. на с.29-30 формулирует основной результат в виде теоремы. Пусть m — это число признаков. Тогда класс адекватных мер сходства для шкал порядка $m\ge1$, интервалов и отношений порядка $m\ge2$ пуст. "Поэтому универсальные правила выбора той или иной меры отсутствуют" (Мандель, 1988, с.30).

Кендалл и Стьюарт (Кендалл, 1976, с.467) предлагают два различных подхода для решения задачи кластеризации. Пусть объекты выборки представлены n точками в p-мерном пространстве переменных. Тогда можно рассмотреть n различных точек в p-мерном евклидовом пространстве. Если эти точки разбиваются на ясно различимые группы, соответственно некоторому приемлемому определению, можно сказать, что n элементов классифицированы

по этим группам. "Близость" элементов должна определяться некоторой функцией от их значений. Второй способ связан с группированием по компонентам, а не по элементам выборки, т.е. рассматриваются строки матрицы наблюдений pxn. Значения по каждой компоненте будут представлены n-мерными векторами. В задачу входит определение того, насколько группируются эти вектора. В первом случае непосредственно рассматриваются объекты выборки, во втором — компоненты случайного вектора, описывающего объект выборки. При этом может исследоваться вопрос, связанный с необходимостью компонент вектора и нахождением наиболее важной компоненты. Для этого в качестве расстояния между двумя произвольными компонентами можно выбрать коэффициент корреляции между ними, задать интервал для расстояния, при котором компоненты считаются “близкими” и могут быть отнесены к одной группе. Под расстоянием здесь понимается мера сходства. Пару компонент с коэффициентом корреляции $\rho \geq \rho_g$ ($0<\rho_g<1$) можно отнести к одной группе. Для проверки очередной компоненты необходимо вычислить коэффициенты корреляции для всех пар компонент, входящих в рассматриваемую группу, с учетом новой компоненты. Если средняя корреляция окажется больше либо равной ρ_g, то новая компонента добавляется в эту группу. Здесь важно отметить, что этот процесс группирования не является однозначным и зависит от порядка, в котором выбираются компоненты для группирования. При небольшом числе компонент можно пытаться группировать всеми возможными способами и исследовать группы при каждом разбиении. Этот метод не зависит от масштаба, в котором измеряются отдельные компоненты. Метод ориентирован на число компонент p. Этот метод распространим и на качественные данные. При кластеризации объектов выборки, представленных векторами, а не их компонентами, описанный метод не подходит, так как в этом случае “расстояния” сильно зависят от изменения масштаба какой-либо переменной.

Как видно из обзора существует множество различных подходов для

разделения объектов на классы. Применительно к процессу построения алфавитного классификатора как конечного результата удобнее использовать термин классификация с неизвестным заранее числом классов. Построение многоуровневого алфавитного классификатора или многоуровневого индекса для ключевого уровня массива базы данных (в частности СУБД НИКА) является малоизученной важной проблемой. В отношении к доступу к БД обычная практика такова, что БД хранения рассматривается отдельно от системы доступа. При этом обычно исследуются запросы, их оптимизация, среднее время, минимальное время, их минимизация. Остаётся не изученной проблема интерактивного доступа и проблема оптимизации такого доступа.

1.2. Постановка задачи построения оптимального классификатора

Применительно к базам данных кластеризация используется при оптимизации доступа к данным. Она заключается в наиболее близком физическом размещении на диске логически связанных между собой данных. Однако остаются не рассмотренными проблемы оптимизации интерактивного доступа к базам данным. Применения методов кластеризации для решения задачи перехода к текстовому ключу массива посредством классификатора, построенного по лексикографическому признаку, относится к этому разряду задач. Основная **цель** диссертационной работы заключается в разработке методов построения многоуровневого классификатора по лексикографическому признаку, адаптирующего структуру лучевого поиска Сассенгата для организации интерактивного доступа к ключевому массиву для повышения эффективности, надёжности и качества гипертекстовой системы. Для достижения цели выделяется класс многоуровневых классификаторов по лексикографическому признаку, среди которых выбирается оптимальный посредством префиксного дерева сочетаний (ПДС). Оптимальность здесь рассматривается в смысле общего числа операций перехода S_{on} в классификаторе в зависимости от максимального числа ключей в классе n и числа ключей в группе n_g на данный ключ. Задача состоит в минимизации

целевой функции, формализованной в виде функционала S_{on}: $S^*_{on} = \min_{n \in D(n), n_g \in D(n_g)} S_{on}(n, n_g)$, по которому определяются оптимальное максимальное число ключей в классе n^* и оптимальное число ключей в группе n^*_g. Другой важной характеристикой классификатора является средняя длина ключа классификатора *k*, зависящая от максимального числа ключей в классе *n*. Здесь представляет интерес решение задачи построения регрессионной зависимости между этими величинами.

1.3. Научный вклад в область интерактивных методов доступа к базам данных

Применение ПДС как универсальной структуры для построения классификатора по лексикографическому признаку на данный момент не имеет аналогов. Обзор по этой структуре данных даётся в разделе 2.1. Префиксное дерево известно давно и описывается в известной монографии Кнута (Кнут,т.1,т.3,с.530). Несмотря на это она используется только в системах доступа к данным и при поиске данных, но не существует ее применения в системе интерактивного доступа к данным и соответствующей оптимизации такого доступа, являющегося альтернативным способом доступа по отношению к поиску данных. Для решения этих задач был предложен метод оптимизации интерактивного доступа к данным с использованием лексикографического многоуровневого классификатора

1.4. Значимость предлагаемого метода

Метод организации интерактивного доступа в виде классификатора на основе префиксного дерева является *новаторским*. Создание гипертекстовых систем на основе сложноструктурированных баз данных больших объёмов, публикуемых в интернет, в которых доступ к данным осуществляется через программы обозревателей общего использования, ставит задачу удобного просмотра, навигации и визуализации объёмных индексов по текстовым полям БД. Многоуровневый алфавитный классификатор призван решить эту

проблему.

В случае просмотра БД через обозреватель интернет важно найти способ отображения текстовых ключевых массивов, представляющих собой индексы по различным полям. Постоянно растущее число таких электронных публикаций обуславливает **актуальность** поставленной проблемы. К электронным публикациям БД можно отнести такие поисковые системы в интернет как google (проиндексированы миллиарды страниц на 2019г.), yandex и т.д. Здесь важно отметить, что поле ввода для перехода на заданный ключ может оказаться недостаточным, т.к. для пользователя важно просматривать сам массив ключей, например, для того, чтобы увидеть какие ключи он содержит и какие не содержит, а не только ввести искомый ключ. А для *мобильных устройств*, неоснащённых клавиатурой, использование классификаторов является более предпочтительным, чем поле ввода. При объёмах массивов в несколько тысяч записей уже может оказаться недостаточным организация интерактивного доступа к массиву в виде листания обычного списка ключей, а при нескольких десятках тысяч становится необходимым более быстрый способ перехода на ключ. Префиксное дерево позволяет выделять отдельные классы ключей на основе буквенных префиксов, которые составляют классификатор. О преимуществах классификатора по лексикографическому признаку перед диапазонным сказано в следующем разделе.

Аналогичная проблема построения пользовательского интерфейса решается в виде классификатора при электронной публикации в интернет БД "Жертв политического террора в СССР" (3,1 млн. имён на 2018г.). В последнем случае применяется трёхуровневый алфавитный классификатор. Первый уровень — буквы алфавита. Второй уровень — диапазонный трёхбуквенный классификатор на выбранную букву. Третий уровень представлен списком фамилий (с инициалом для частотных фамилий) из выбранного диапазона на втором уровне. Этот список неудобен тем, что может быть очень длинным.

Такой классификатор не оптимизирован по числу операций и не даст оптимального числа переходов пользователя на искомый ключ.

Приведённый пример является характерным для понимания проблемы построения классификатора. Дальнейшее рассмотрение проблемы построения классификатора по лексикографическому признаку ставит задачу, связанную с оптимизацией числа переходов пользователя в рассматриваемом классификаторе при переходе на искомый ключ. Сюда же можно отнести задачу, связанную с определением средней длины ключа классификатора или среднего числа переходов в зависимости от среднего числа ключей в классе.

Удобство рассматриваемого интерфейса состоит также в том, что классификатор является “схемой” для обозрения ключевого уровня массива, состоящей из буквенных префиксов-“подсказок”. Электронная публикация БД больших объёмов в интернет началась сравнительно недавно, а системы общего поиска в интернет не предъявляют к интерфейсу пользователя таких требований, поэтому рассматриваемая проблема является **малоизученной**.

Наконец, резюмируя этот раздел, следует указать на преимущество *оптимального классификатора* в **сравнении** с существующими методами интерактивного доступа. К ним относится простой список, разделённый на равные группы ключей, используемый как способ представления результата в общих поисковых системах в интернет, или алфавитный классификатор, состоящий из нескольких уровней с ключами на последнем уровне, выбранных “по смыслу”. При объёмах массивов в несколько миллионов записей такой подход оказывается недостаточным и приводит к невозможности использования такого интерфейса, что и проявилось в случае БД о политических репрессиях, когда в новой электронной публикации версии БД был сделан полный отказ от такого классификатора и использована только запросная форма, в виде нескольких полей. Это означает, что проблема построения оптимального классификатора должна быть формализована и решена в виде математической модели, которая позволит гибко оптимизировать

пользовательский интерфейс в зависимости от количества ключей в массиве и частот различных префиксов.

1.5. Обоснование классификации уникальными алфавитными ключами

Рассмотрим “буквенные метки” в словарях или записных адресных книжках. Они позволяют по начальному сочетанию букв найти страницы с соответствующими словами. Наличие у страницы книги трёх свободных сторон позволяет сделать для словаря метки поиска до третьей буквы слова включительно. Следуя дальше идее побуквенных меток, можно получить схему поиска, основанную на повторном индексировании в виде луча — быстрого чётко определённого по направлению движения поиска. “Луч или ПДС — это М-арное дерево, узлы которого представляют собой М-арные векторы с компонентами, соответствующими цифрам или буквам. Каждый узел уровня l является набором всех ключей, начинающихся с определенной последовательности l символов, которая называется его префиксом; узел определяет разветвление на M путей в зависимости от l+1 символа… Узлы-векторы расположены в соответствии с кодами символов… Это означает, что “лучевой поиск” будет весьма быстрым…” (Кнут,2007,т.3,с.527). Это составляет технологию быстрого многоуровневого принятия решения по индексу. Абстрактная концепция луча (Кнут,2007,т.3,с.529) “для представления семейства строк была предложена Акселем Тью в статье о строках, которые не содержат смежных повторяющихся строк”. Если задано N ключей, представляющих собой исходные строки, то среднее время поиска для больших N включает около $\log_M N$ итераций при случайных входных данных. При этом необходимо “примерно $N/\ln M$” М-арных узлов для размещения N случайных ключей. Количество цифр или символов, проверяемых в ходе случайного поиска, составляет примерно $\log_M N$ (Кнут,2007,т.3,с.542). Для сжатого дерева, в котором все узлы с одним подчинённым узлом объединены в один узел, это также справедливо.

Можно рассмотреть два варианта классификаторов: на основе

равномерного разбиения на классы с одинаковым числом ключей (каждому классу будет соответствовать некоторый диапазон буквенных ключей) и на основе разбиения по лексикографическому признаку (каждому классу будет соответствовать некоторый уникальный буквенный ключ). Второй метод классификации на основе ПДС предпочтителен тем, что при неудачном поиске ключа его отсутствие обнаруживается сразу при просмотре очередного уровня классификатора, соответствующего данной букве ключа, которая отсутствует в ПДС для данного сочетания. При равномерной классификации с диапазонами ключей это можно обнаружить только на ключевом уровне массива, т.е. осуществляя проход через все уровни классификатора до последнего. Таким образом, равномерный классификатор не даёт никакой информации о наличии или отсутствии сочетаний внутри диапазона.

Кнут также упоминает о совместном использовании дерева и прохода по списку ключей (Кнут,2007,т.3,с.531). “Идея луча хороша только для первых нескольких уровней дерева. Повысить производительность можно за счёт комбинирования двух стратегий: луча для нескольких первых символов и переключение на другую стратегию — для оставшихся. Например, Сассенгат (Sussenguth,1963) предложил использовать посимвольную схему до достижения части дерева, в котором возможны, скажем, не более шести ключей, а затем проходить последовательно по этому списку… Такая смешанная стратегия позволяет уменьшить количество М-арных узлов луча примерно в 6 раз без существенного изменения времени работы.”

Таким образом, при построении классификатора на основе ПДС возникает задача, связанная с определением оптимальной структуры классификатора для работы пользователя с ключевым массивом. Такой классификатор позволит за минимальное число шагов найти искомый ключ или обнаружить его отсутствие в ключевом массиве.

Выводы по главе

Глава 1 включает в себя обзор методов классификации. Из обзора следует, что в настоящее время существует большое количество методов кластеризации и построение *многоуровневого иерархического классификатора по лексикографическому признаку* можно отнести к иерархической дивизивной кластеризации. Исходный упорядоченный массив ключей представляет собой один кластер или класс, который делится затем на другие кластеры или классы по лексикографическому признаку. Результатом работы такой процедуры является иерархия вложенности классов, представляя собой различные способы группировки ключей в классы. Эта иерархия классов может быть представлена в виде префиксного дерева сочетаний (ПДС). Результатом кластеризации является префиксное дерево, построения которого на множестве ключей не является сложной задачей, т.к. строки, начинающиеся на одинаковые префиксы, естественным образом разбиваются на кластеры или классы. Каждый префикс в ПДС обозначает класс всех ключей, начинающихся с этого буквенного префикса. Обзор публикаций по префиксным деревьям дан в начале следующей главы. Префиксное дерево, представляющее собой пространство состояний классификатора, порождает целый класс классификаторов. Ставится задача выбора оптимального классификатора в виде минимизации целевой функции, в качестве которой выступает функционал общего числа операций в классификаторе. Отмечается, что предлагаемый *интерактивный* метод доступа к ключевому массиву БД на основе префиксного дерева является новаторским. Показана актуальность метода в связи с появлением электронных публикаций баз данных в интернет для визуализации и навигации по текстовым ключевым массивам. В конце главы показывается преимущество классификации с помощью уникальных алфавитных ключей. Следующая глава 2 посвящена методам построения алфавитных классификаторов с различным максимальным числом ключей в классе, среди которых на основе ПДС выбирается оптимальный с точки зрения общего числа операций в алфавитном классификаторе, чему посвящена глава 3.

Глава 2. Методы построения алфавитного классификатора

Во **второй главе** проводится обзор публикаций по структуре лучевого поиска, называемой ПДС или деревом PATRICIA Моррисона (см.рис.2.2,a), проанализирована сложность операций в данной структуре. Вводятся понятие ПДС и понятие классификатора с использованием ПДС, рассмотрены различные виды многоуровневого классификатора, а также модельные распределения ключей по буквенным сочетаниям, разработаны характеристики классификатора в виде случайных величин: длины префикса в ПДС и числа ключей в классе на данный префикс, построена регрессионная зависимость этих величин.

Классификатор представляет собой сжатое по поддеревьям ПДС (отличие только в том, что любой префикс классификатора — это путь от корня ПДС) и использует структуру лучевого поиска, предложенную Сассенгатом. PATRICIA — сжатое префиксное дерево trie без однопутевых ветвей. Структура trie была предложена Брианде и Фредкиным. Сассенгат использует метод, комбинирующий несколько первых уровней дерева с прерыванием ветвления на некотором уровне и списки ключей на соответствующие префиксы (см. пример на рис.2.2,b).

2.1. Понятие префиксного дерева сочетаний

В этом пункте даются определения, связанные с рассматриваемой иерархической структурой, и рассматриваются некоторые ее свойства.

Дерево (Кнут,2002,т.1) — это конечное множество T одного или более узлов со следующими свойствами: a) существует один выделенный узел — корень; b) остальные узлы (за исключением корня) распределены среди $m \geq 0$ непересекающихся множеств $T_1,\ldots, T_m$ и каждое из этих множеств, в свою очередь, является деревом; деревья $T_1,\ldots, T_m$ называются поддеревьями данного корня. Если в пункте b имеет значение относительный порядок поддеревьев $T_1,\ldots, T_m$, то дерево называют упорядоченным. M-арное дерево — это конечное множество узлов, которое является пустым или состоит из корня и M

непересекающихся M-арных деревьев. M-арное дерево является упорядоченным.

В книге (Кнут,2007,т.3,с.470) рассматриваются оптимальные бинарные (M=2) деревья поиска. Для N упорядоченных в лексикографическом порядке, ключей K_i с заданными частотными вероятностями, а также заданными вероятностями попадания искомого ключа между парами ключей K_i и K_{i+1} (при неуспешном поиске) ищется наилучшее бинарное дерево поиска в смысле ожидаемого числа сравнений при поиске. Даны $2N+1$ вероятностей $p_1,p_2,\ldots, p_N$ и $q_0,q_1,\ldots,q_N$, где p_i — вероятность того, что аргументом поиска является K_i; q_i — вероятность того, что аргумент поиска лежит между K_i и K_{i+1}; q_0 — вероятность того, что аргумент поиска меньше, чем K_1; q_N — вероятность того, что аргумент поиска больше, чем K_N, причём $\sum_{j=1}^{n} p_j + \sum_{k=0}^{n} q_k = 1$ [*)]. Цена дерева определяется как ожидаемое количество сравнений, а дерево с минимальной ценой называется оптимальным. Необходимо найти оптимальное дерево, минимизирующее ожидаемое количество сравнений при поиске: $C = \sum_{j=1}^{n} p_j \left(l_j^{(in)} + 1\right) + \sum_{k=0}^{n} q_k l_k^{(ex)}$. Здесь $l_j^{(in)}$ — номер уровня j-ого внутреннего узла; $l_k^{(ex)}$ — номер уровня (k+1)-ого внешнего узла[**)]. Корню дерева соответствует нулевой уровень. Пусть $c(i,j)$ — цена оптимального поддерева с весами $(p_{i+1},\ldots,p_j;q_i,\ldots,q_j)$ и $w(i,j)= p_{i+1}+\ldots+p_j+q_i+\ldots+q_j$ — сумма этих весов. На примере индексирования ключевых слов в контексте (Luhn, 1960) строится оптимальное бинарное дерево поиска (из 35 ключевых слов). Также рассматривается зависимость поведение цены как функции корня k (см.рис.2.1).

[*)] Условие нормировки не является обязательным. Вместо вероятностей может быть любая последовательность весов.

[**)] Внутренние узлы соответствуют ключам K_i, а внешние — листьям полного бинарного дерева, до которого оно достраивается

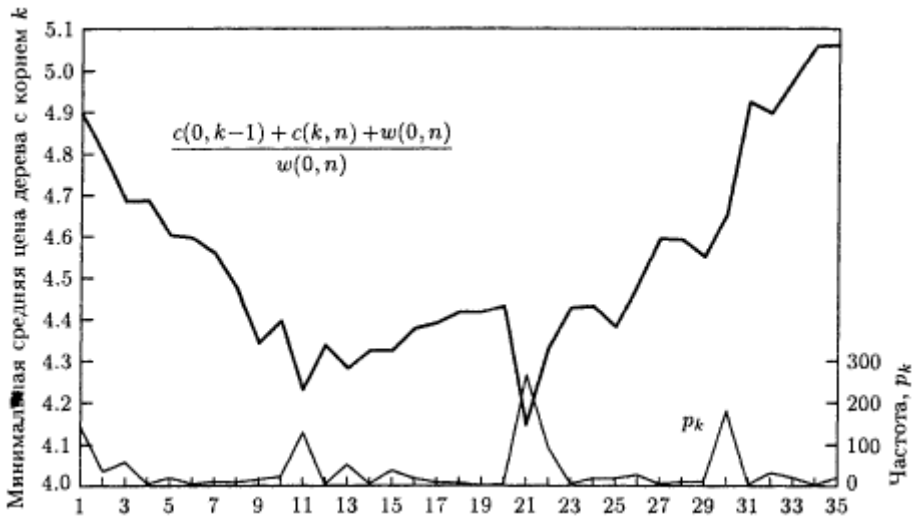


Рис.2.1. Поведение цены как функции корня k

Если построить график $c(0,k\text{-}1)+c(k,n)$ как функцию от k, то можно видеть из рис.2.1, что результат зависит от p_k. Оптимальное дерево требует в среднем 4,15 сравнений (глобальный минимум на графике). Полностью индекс приводится в (Youden, 1963). Исходя из приведённого примера, можно определить цену дерева в случае алфавитного классификатора, основанного на префиксном дереве сочетаний, определяемого ниже. Только в случае классификатора цена дерева будет представлять собой суммарное число операций по поиску всех ключей в иерархическом классификаторе, и задача будет сводиться к поиску оптимального классификатора с минимальным общим числом операций.

Луч или ПДС (Кнут,2007,т.3,с.527) — это M-арное дерево, узлы которого представляют собой M-местные векторы с компонентами, соответствующими цифрам или буквам. Каждый узел уровня l является набором всех ключей, начинающихся с определенной последовательности l символов, называемой его префиксом. Узел определяет разветвление на M путей в зависимости от l+1 символа.

Рассмотрим алгоритм “Лучевого поиска” (Кнут,2007,т.3,с.527). Дана таблица записей в форме M-арного луча. Алгоритм осуществляет поиск по заданному аргументу K. Узлы луча представляют собой векторы, индексы которых изменяются от 0 до M-1; каждый компонент такого вектора

представляет собой ключ или ссылку (возможно, пустую).

Шаг 1. Инициализация. Установить ссылочную переменную *P* так, чтобы она указывала на корень луча.

Шаг 2. Ветвление. Установить *k* равным следующему символу аргумента *K* слева направо. Если аргумент полностью просканирован, установить *k* равным нулю — пустому символу. Пусть *X* — *k*-й элемент в узле *P*. Тогда если *X* — ссылка, то выполнить шаг 3; если *X* — ключ, то выполнить шаг 4.

Шаг 3. Продвижение. Если $X \neq \Lambda$ (Λ — пустое слово), то установить *P* в *X* и вернуться к шагу 2; в противном случае алгоритм завершается неудачей.

Шаг 4. Сравнение. Если $X=K$, алгоритм завершается успешно; в противном случае алгоритм завершается неудачей.

При неудачном поиске будет найден элемент, более всего совпадающий с искомым. Время работы алгоритма сравнимо с бинарным поиском при успешном поиске ключа. Абстрактная концепция луча для представления семейства строк была предложена Акселем Тью в статье о строках. В описанном алгоритме “лучевого поиска” таблица может быть представлена в виде леса. Это было предложено Брианде (Briandais, 1959) для сохранения памяти при использовании связанного списка для каждого узла-вектора, поскольку большинство элементов вектора обычно пусто. Поиск в таком лесу осуществляется путем нахождения корня,

Рис.2.2. Пример сжатого префиксного дерева сочетаний: (a) без прерывания (b) с прерыванием ветвления

соответствующего первому символу, затем дочернего узла этого корня, соответствующего второму символу, и т.д. При этом каждый символ может быть представлен отдельным узлом в дереве. На рис.2.2,a, приводится пример сжатого префиксного дерева, в котором узлы с одной подчинённой вершиной “сжаты” в один узел, в одно буквенное сочетание. Предполагается, что узлы одного уровня упорядочены в алфавитном порядке. Для такого представления требуется больше памяти, чем для представления, в котором прерывается ветвление (см. рис.2.2,b), однако существенно упрощается работа с ключами переменной длины (Кнут,2007,т.3,с.530). Пусть N — это число ключей. Среднее время поиска посредством “лучевого поиска” или количество цифр или символов, проверяемых в ходе случайного поиска, составляет около $\log_M N$ итераций при случайных входных данных. Случайные данные означают, что M-ичные цифры равномерно распределены. При неравномерном распределении деревья будут существенно более асимметричными по сравнению с деревьями, полученными при случайных данных. Для бинарного луча M=2 известны более точные приближения: $\log_2 N$+1,33275 — для успешного поиска; $\log_2 N$-0,10995 — для неудачного поиска. Требуется примерно $N/\ln M$ узлов в M-арном луче с разветвлениями для различения N случайных ключей. Соответственно для различения групп из s ключей требуется $N/(s\ln M)$ узлов. Это приближение справедливо при больших N, малых s и малых M. Общее количество необходимой памяти пропорционально $MN/\ln M$. Для повышения эффективности предлагается использовать комбинированный метод. Идея луча подходит только для нескольких первых уровней дерева, а затем можно, например, проходить последовательно по списку ключей (Sussenguth,1963). Э.Г. Сассенгат берёт число ключей в этом списке не более шести. Такой комбинированный метод позволяет уменьшить число узлов дерева примерно в шесть раз. Кнут также вводит понятие “сжатого” дерева без однопутевых ветвей, когда узлы с одним путем объединяются с предыдущим узлом, перед которым было ветвление, в один узел, а буквы в этих узлах последовательно

составляют символьную последовательность данного объединенного узла.

Отдельным случаем построения описанного дерева является упорядочивание узлов одного уровня по частоте появления соответствующего буквенного префикса для рассматриваемого набора ключей. Все ключи считаются равновероятными. К префиксам можно применить рассуждения, которые Кнут применяет для неравномерного распределения ключей, если принять, что число обращений — это есть частота появления данного префикса. Он рассматривает этот вопрос в последовательном поиске по ключу. Среднее число сравнений в общем случае равно $\overline{C}_N = \sum_{i=1}^{N} i p_i$, где N — число ключей, p_i — их вероятности. Значение $\overline{C}_N$ минимально при $p_1 \geq p_2 \geq \ldots \geq p_N$, когда наиболее часто используемые записи находятся вначале. Далее он рассматривает различные случаи распределения вероятностей, и какой вклад каждое из них вносит в значение $\overline{C}_N$ при оптимальном размещении ключей (в порядке убывания частот). В случае равномерного распределения $p_1=p_2=\ldots=p_N=1/N$ получается, что $\overline{C}_N=(N+1)/2$. Если $p_i=1/2^i$, то $\overline{C}_N = 2 - 2^{1-N}$. В этом случае среднее количество сравнений меньше 2, если ключи расположены в порядке убывания вероятностей. Для клиновидного распределения вероятностей (Кнут,2007,т.3,с.432-433) $p_i = \frac{2(N-i+1)}{N(N+1)}, i = 1,\ldots,N$ получается величина $\overline{C}_N=(N+2)/3$. Это экономит около трети времени поиска при оптимальном размещении ключей в сравнении с равномерным распределением. Кнут указывает на то, что приведенные два последних распределения являются чисто теоретическими. Он утверждает, что в реальных ситуациях встречается ранговое распределение Ципфа. Объяснение закона Ципфа в современной интерпретации дается в статье (Kechedzhy,2004), где выводиться, что вероятность появления слова обратно пропорциональна его рангу $p \sim R^{-\zeta}$ в N-шаговой марковской цепи, в которой каждое следующее слово получается сдвигом на следующий символ в последовательности из N символов, состоящей

из нулей и единиц. Здесь $\zeta = \frac{1}{1+2n/L}$, где L обозначает длину слова, $n = \frac{N(1-2\mu')}{4\mu'}$, N — длина некоторой символьной последовательности: $L \leq N$, сила корреляций в последовательности: $-1/2 < \mu' < 1/2$, $\mu' = 0$ соответствует некоррелированной последовательности символов, положительное (отрицательное) μ' соответствует "притягиванию" ("отталкиванию") символов одного вида. Важно отметить, что закон Ципфа является чисто статистическим феноменом (Wentian, 1991). Ципф обнаружил, что n-ое по частоте слово языка встречается с частотной вероятностью "примерно обратно пропорциональной n" — $p_i = \frac{1}{iH_N}$, где H_N — N-ое гармоническое число. В случае такого закона распределения имеем $\overline{C_N} = N/H_N$ (Кнут,2007,т.3,с.427). Тогда поиск по ключу будет осуществляться примерно в $(1/2)\ln N$ раз быстрее, чем в равномерном случае. Другое распределение предполагает, что 80% запросов происходит к 20% ключей, т.е. $\frac{p_1 + p_2 + \ldots + p_{0.20n}}{p_1 + p_2 + \ldots + p_N} \approx 0{,}80$ для всех n. Это распределение также можно рассматривать как закон Ципфа, но совершенно в другой форме (Кнут,2007,т.3,с.427). При n, кратных 5, получается вероятностная функция $p_i = \frac{i^\theta - (i-1)^\theta}{N^\theta}$, $\theta = \log_2 0{,}80 / \log_2 0{,}20 = 0{,}1386$. Приближенно можно записать $p_i = \frac{1}{i^{1-\theta} H_N^{(1-\theta)}}$. Здесь $H_N^{(s)}$ обозначает N-ое гармоническое число порядка s. С изменением θ от 1 до 0 закон распределения изменяется (Кнут,2007,т.3,с.428) от равномерного к закону Ципфа. Среднее число сравнений для полученного закона равно $\overline{C_N} = H_N^{(-\theta)} / H_N^{(1-\theta)} \approx 0{,}122N$. На практике часто априори не известен закон распределения. Кроме того, необходимо хранить счетчики для каждой записи для того, чтобы упорядочивать ключи в оптимальном порядке. Кнут предлагает использовать метод (Кнут,2007,т.3,с.429), при котором найденный ключ перемещается в начало набора ключей. Таким образом, наиболее часто используемые ключи будут располагаться в начале набора. В этом случае

$\overline{C}_N - \tilde{C}_N = \frac{1}{2} + \sum_{1 \le i,j \le N} \frac{p_i p_j}{p_i + p_j}$. В случае распределения вероятностей по закону Ципфа предельная величина получается приблизительно равной $2N/\lg N$. При больших значениях N эта величина значительно меньше, чем $N/2$. Она превышает количество сравнений при оптимальном размещении записей только в 1,386 раза. Другой метод заключается в том, что найденный ключ меняется местами с предыдущим (McCabe,1965). Этот метод использует строго меньше сравнений, чем предыдущий при длительной работе, но сходится к предельному значению медленнее.

Необходимо сделать замечание относительно применимости закона Ципфа. Закон Ципфа представляет собой математическую модель распределения слов, которая имеет свои ограничения. Известно, что на небольших текстах закон Ципфа выполняется, а на длинных текстах, состоящих из самостоятельных отдельных частей, начинает нарушаться. Закон Ципфа плохо выполняется в области малых частот: число слов с малыми частотами оказывается меньше, чем получается по закону Ципфа (Маслов, 2006).

В дальнейшем будем рассматривать префиксные деревья сочетаний, подобные тому, которое изображено на рис.2.2, т.е. подобные “сжатому” варианту ПДС, когда имеются только узлы с ветвлением, а символы узлов без ветвления приписываются к ключу ближайшего узла с ветвлением, которому подчинены эти узлы.

Понятие префиксных деревьев trie вводится в литературе (Briandais,1959;Fredkin,1960). Данные структуры используются для представления массивов строк в памяти. Здесь обычно рассматриваются два аспекта — временная сложность и пространственная сложность. Исходя из оптимизации соответствующих параметров, с течением времени разрабатываются структуры, основанные на модификации исходной структуры trie. В конце 60-х годов Моррисон предложил сжатую структуру trie в виде дерева PATRICIA без однопутевых ветвей (Morrison,1968). Ещё ранее Эдвард

Сассенгат составил комбинированную стратегию цифрового поиска (Sussenguth,1963). Этим достигается одинаковое время поиска и обновления ключей. Она состоит из лучевого поиска до определённого уровня с прерыванием ветвления и последовательного прохода по списку ключей. Дальнейшие модификации используют перечисленные структуры. Более эффективная структура String B-tree при произвольной длине ключа предложена в статье (Ferragina,1999). Она комбинирует B-дерево и сжатое префиксное дерево PATRICIA. Эта идея развивается дальше в виде структуры BST— block string trie (Таранов,2011) или префиксного дерева, разделённого на блоки. В сравнении с аналогичными структурами, основанными на B^+-деревьях (Bayer,1972,1977;Walczuch,1999), они показывают такую же производительность и в тоже время в некоторых случаях могут использовать значительно меньше места. Наиболее близкими к структуре BST являются структуры, описанные в (Heinz,2002;Sample,2002;Askitis,2009). С точки зрения классификатора более важным является оптимизация среднего времени поиска ключа. В этом смысле интересна достаточно новая статья Резника (Reznik,2005). Он выделяет класс структур trie с адаптивным многозначным ветвлением. В отличие от обычного префиксного дерева и большинства его модификаций со средним временем доступа $O(\log n)$ (Кнут,Т.3;Sussenguth,1963;Fagin,1979;Flajolet,1986 и т.д.) структуры из этого класса имеют значительно более быстрое время доступа порядка $O(\log \log n)$. Резник изучил асимптотические свойства этого класса структур, к которому относятся LC-trie (Andersson,1993), т.е. сжатые по уровню структуры trie, “разряженные” LC-trie (Nilsson,1998,2002), структуры trie с логарифмическим ветвлением (Reznik,2002,2005). Исследуемый классификатор использует несколько технологий: дерево PATRICIA без однопутевых ветвей (Morrison,1968), комбинированную стратегию Сассенгата с прерыванием ветвления и затем проход по списку ключей (Sussenguth,1963), а также сжатие по поддеревьям. Как и в случае сжатия по уровням (Andersson,1993) сжатие по

поддеревьям приводит к многозначному ветвлению, что в свою очередь уменьшает среднюю высоту дерева классификатора. В нижеприведённом списке приводится средняя глубина дерева для разных случаев префиксного дерева, построенного для поля ФИО с числом ключей 34 657.

1. Префиксное дерево trie: 3,106
2. Оптимальный классификатор (Тищенко,2018): 1,991
3. Сжатое по уровню дерево LC-trie (Reznik,2005): 1,143

Отсюда видно, что оптимальный классификатор даёт глубину дерева, меньшую, чем обычное дерево trie, но большую, чем префиксное дерево сжатое по уровням, поскольку классификатор содержит только часть многозначных узлов, которые имеются в сжатом по уровням префиксном дереве.

В заключении этого пункта важно отметить, что префиксное дерево и его модификации востребованы в современных разработках и исследованиях. Для примера приведены несколько из них: перестройка строковой хеш-таблицы, структуры burst-trie для более эффективного использование кеша (Askitis,2010); блочное строковое префиксное дерево (Таранов,2011); эффективный поиск для очень большого множества строк (Fenz,2012); дисперсия числа узлов trie, исключая листья и узлы, которым непосредственно подчинены листья (Gaither,2013): выбор оптимальных по занимаемой памяти структур хеш-trie из соответствующего семейства структур (Steindorfer,2014); применение структур trie в базах данных в оперативной памяти для организации индексов (Truong,2015); применение структур trie для параллельной проверке свойств (Wen,2016); о реализация набора структур, основанных на trie, программных продуктах (Steindorfer,2016); предложена эффективная структура для работы с большим набором N-грамм (Pibiri,2017); предложена оригинальная структура хеш-trie с операциями над ней за постоянное время посредством параллельной организации вычислений (Prokopec,2018); использование структуры trie для генератора языка для расширенных регулярных выражений (Radanne,2018).

2.2. Различные виды многоуровневого классификатора на основе ПДС

2.2.1. Классификатор при равномерном распределении ключей по префиксам

В случае равномерного распределения ключей массива по буквенным префиксам можно применить формулу, определяющую число перестановок с неограниченными повторениями $U(a,k)=a^k$, где a — мощность алфавита, k — длина префикса. Отсюда $a^{k_m} = N$, где k_m — максимальная длина префикса, N — число ключей в массиве. Таким образом, можно записать.

$$k(n)=\log_a(N/n), \qquad (1)$$

где k — длина префикса,

N — общее число ключей на ключевом уровне массива,

a — мощность алфавита A: $a=|A|$,

n — число ключей в каждом из классов, на которые разбивается массив в алфавитном порядке следования ключей.

При n=1 получаем $k=k_m=\log_a N$. Это есть длина уникального префикса для данного числа ключей N. Длина ключа k задаёт разбиение исходного набора ключей по k-граммам. При k=0,1,2,…,k_m получаются соответствующие разбиения на классы (каждый класс определяется соответствующим уникальным префиксом длины k для каждого ключа из этого класса), содержащие n ключей: $n = a^{k_m}, a^{k_m-1}, a^{k_m-2}, ..., 1$. При k=0 получается вырожденный случай, когда в классе присутствует весь набор ключей, т.е. $n=N$. Каждый уровень содержит весь алфавит и число букв на уровне равно a. На каждом уровне добавляется по одной букве алфавита к буквенному префиксу. При $k=k_m$ каждый префикс будет задавать ровно один ключ. Для идеального случая распределения, т.е. равномерного распределения, следует брать число ключей в классе, являющееся степенью числа a мощности алфавита: $n=a^k$, k=0,1,2,…,k_m. Длина префикса будет равна степени числа a. Тогда число ключей в классе выражается через длину ключа как $n=N/a^k$. Отсюда $k(n)=\log_a(N/n)$.

2.2.2. Классификатор при частично равномерном распределении ключей по префиксам с равновероятными буквами одного уровня, начиная с определённого уровня

В общем случае распределение ключей по сочетаниям является неравномерным.

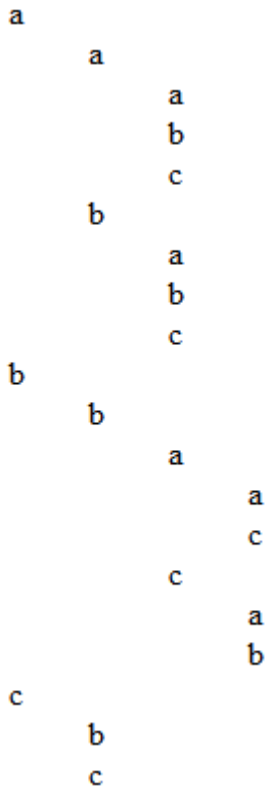


Рис.2.3. Пример ПДС

Классификатор — это иерархическая структура, в которой ключи могут быть составлены из сочетаний букв на основе ПДС. Отличие от ПДС состоит в том, что на каждом последующем уровне иерархии может добавляться *в среднем* по нескольку букв, даже в случае недостоверных переходов (достоверный переход — это переход с одной подчинённой вершиной). Кроме того, такой классификатор в зависимости от заданного числа ключей в классе n может охватывать ПДС не на полную глубину. Таким образом, среднее число уровней в классификаторе может быть меньшим, чем в ПДС. Пусть на каждом уровне классификатора может добавляться одна буква алфавита. Тогда каждый уровень классификатора может содержать любое подмножество букв алфавита $A_p \subseteq A$, $A_p \neq \varnothing$, $a_p = |A_P|$, $0 < a_p \leq a = |A|$. Возможные варианты множества A_p

определяются как $A_p \in B$=Bool(A)\{∅}, где Bool обозначает операцию взятия булеана, причем $|B|=2^a-1$. Величину a_p и множество A можно рассматривать как случайные, т.к. они зависят от сочетания начальных букв S_q , q=1,…,q_m, где индексы q присвоены в порядке левого обхода классификатора. Каждая буквенная позиция сочетания будет характеризоваться своим вероятностным распределением букв. Можно рассмотреть упрощённый вариант неравномерного случая, когда, начиная с некоторого уровня, все буквы на одном уровня равновероятны с частотными вероятностями $Pa_p(S_q)=1/a_p(S_q)$. Здесь $a_p(S_q)=|A_p(S_q)|$ и $A_p(S_q) \in B$ обозначает множество букв на заданное начальное сочетание S_q. На разных уровнях такого классификатора может быть различное множество букв. На рис.2.3 показан абстрактный пример ПДС с равномерными уровнями, начиная со второго уровня. На каждом уровне добавляется по одной букве к сочетанию. Сочетания внутри классов {aa, ab}, {bb}, {cb,cc} являются равновероятными. Сочетания из разных классов не равновероятны.

2.2.3. Случай числа ключей в классе, несовпадающего со степенью числа a=|A| (мощности алфавита A)

Пусть число ключей в классе n: $a^i<n<a^{i+1}$, k=0,…,k_m-1. Тогда структура классификатора будет совпадать с вышеописанным “идеальным” случаем, если принять, что $k=[\log_a(N/n)]+\delta(n)$, где $\delta(n)=1$, если $n<(a^i+a^{i+1})/2$, и $\delta(n)=0$, если $n\geq(a^i+a^{i+1})/2$. Получаем, что $\log_a(N/a^{i+1})<\log_a(N/n)<\log_a(N/a^i)$. Отсюда $[\log_a(N/n)]=\log_a(N/a^{i+1})=k_m-i-1$ и $k=k_m-i-1+\delta(n)$. Таким образом, в случае, если заданное число ключей в классе n меньше среднего арифметического между двумя соседними степенями a, ближайшими к n, то n принимается равным a^i. В противном случае n принимается равным a^{i+1}. Во втором случае число уровней в классификаторе будет на единицу меньше, чем в первом случае.

2.2.4. Классификатор, получаемый при числе ключей в классе, несовпадающим со степенью числа a=|A|

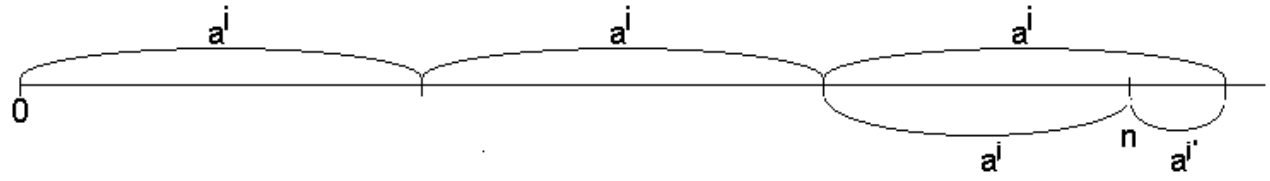


Рис.2.4. Число ключей в классе $a^i<n<a^{i+1}$

Классификатор, получаемый при n: $a^i<n<a^{i+1}$ позволяет приблизиться к рассмотрению неравномерного случая распределения ключей по буквенным сочетаниям, т.к. при определённых значениях n он будет иметь различное число букв в префиксе для различных классов. Пусть дано разбиение на классы величиной n: $a^i<n<a^{i+1}$. Тогда, если $n\geq(a^i+a^{i+1})/2$, то будем заменять его на разбиение a^{i+1}, а в случае $n<(a^i+a^{i+1})/2$ — на разбиение a^i. Здесь можно рассмотреть 2 случая. При n mod $a^i=0$ получим классификатор с длиной префикса на последнем уровне $k=[\log_a(N/n)]+1=\log_a(N/a^{i+1})+1=k_m-i$, а при n mod $a^i\neq0$ в классы размером n не будет вмещаться целое число подклассов размером a^i и некоторые классы a^i могут разбиваться числами cn, $c\in\mathbf{N}$ в общем случае на две неравные части. Эти классы требуют более мелкого разбиения на подклассы меньших степеней а, чем i. На рис.2.4 показан пример, когда n разбивает класс из a^i ключей на 2 подкласса — с числом ключей a^j и $a^{j'}$, где $j'<j<i$. Длины ключей у этих классов будут больше, чем у классов с a^i ключами на $i-j$ и $i-j'$ символов соответственно. Произвольный случай разбиения класса на подклассы показан на следующем рисунке.

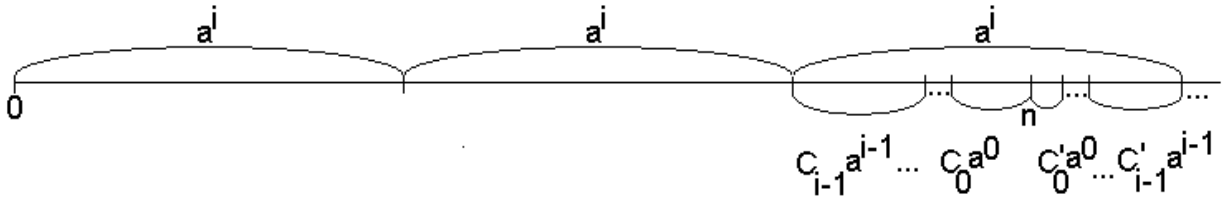


Рис.2.5. Произвольный случай разбиения класса на подклассы

2.2.5. Классификатор с “искусственной” неравномерностью и общий неравномерный случай

Классификатор с равномерным распределением ключей по префиксам позволяет понять общий характер зависимости длины ключа класса от числа ключей в классе. Классификатор, являющийся равномерным с некоторого заданного уровня ПДС, позволяет выяснить как начинает меняться данная зависимость, когда вноситься элемент случайности, и достоверные величины длина ключа и число ключей в классе становятся случайными. “Искусственная” неравномерность возникает, если взять число ключей в классе n, не являющееся степенью числа *a,* в случае равномерного ПДС (см. предыдущий пункт). В этом случае длины ключей классов будут повторяться с некоторым периодом *T*. В общем неравномерном случае нельзя выделить такого периода *T* повторяемости длин ключей. Однако внутри полупериода классификатор с “искусственной” неравномерностью “моделирует” общий неравномерный случай. В общем случае безусловная вероятность сочетания может быть вычислена по формуле умножения вероятностей: $P(a_{q1}a_{q2}\ldots a_{qk})=P(a_{q1})\ P(a_{q2}|a_{q1})\ \ldots\ P(a_{qk}|a_{q1}\ldots a_{qk\text{-}1})$, где $a_{q1},a_{q2},\ldots,a_{qk}\in A$. В этой формуле в произведении стоят условные вероятности, кроме первой. Эти вероятности определяются как частотные вероятности для заданного набора слов. Для описания неравномерного случая распределения букв по сочетаниям можно использовать такую же структуру, что и в случае равномерном — ПДС (Кнут,2007,т.3,с.527; Емельянов, Тищенко, 2010; Соловьёв, Тищенко, 2018). В неравномерном случае на каждом уровне такого дерева могут присутствовать не все буквы алфавита А. Нумеруя буквенные сочетания в порядке левого обхода (Богачева, Емельянов, 2003) ПДС, можно обозначить множество букв на данное сочетание s_i как $A_{si}\subset A$, $a(s_i)=|A_{si}|$. Множество A_{si} и величину $a(s_i)$ можно считать случайными. Если задать некоторое максимальное число ключей в классе $n_{\max}$, то можно получить классификатор, являющийся поддеревом исходного ПДС. Однако в силу неравномерности распределения ключей по буквенным сочетаниям он будет содержать произвольное подмножество букв алфавита А на каждом уровне классификатора. Для случайной величины длины

ключа можно определить среднее значение. Это характеристика полученного классификатора, которая при сравнении с равномерным случаем может служить показателем неравномерности. Для сглаживания этой неравномерности можно задать пороговое значение для числа букв на текущем уровне и при значении меньше порогового корректировать значение длины ключа в каждом конкретном случае, увеличивая значение на единицу до тех пор, пока не будет превышено пороговое значение или достигнута заданная максимальная длина ключа. В качестве порогового значения можно взять среднее число букв на уровне. В скорректированном классификаторе уровни будут содержать число букв, не менее порогового значения, за исключением тех уровней, для которых не существует число букв, превышающее пороговое значение.

Средняя длина ключа в классификаторе определяется по формуле $M\kappa(n) = \sum_{i=1}^{n} k_{si} n_{si} / N$. Сумма берётся по всем буквенным сочетаниям s_i, занумерованным в порядке левого обхода классификатора. Здесь n_{si} обозначает число ключей в классе с ключом s_i[*)], причём $n_{si} \le n$, n — заданное максимальное значение числа ключей в классе. Мк(1) соответствует случаю одного ключа в классе n=1 и n_{si}=1 для любого сочетания s_i. Можно сформулировать очевидное утверждение о том, что $k(1) \le M\kappa(n) \le M\kappa(1)$, где $k(1)=\log_a N$ — это длина ключа для равномерного случая. Это неравенство говорит о том, что неравномерный случай “не лучше” равномерного в смысле величин $k(1)$ и $M\kappa(1)$. Другое утверждение — для любых двух классификаторов, являющимися поддеревьями одного того же ПДС, выполнено неравенство $M\kappa(n_1) \le M\kappa(n_2) \Leftrightarrow n_1 \ge n_2$. Знак равенства достигается при $n_1=n_2$. Неравенство $M\kappa(n) \le M\kappa(1)$ говорит о соотношении средних длин ключа класса и ключа “единичного” класса (с одним ключом). Подобное неравенство справедливо и для конкретных длин

[*)] В качестве ключа многоуровневого классификатора s_i необходимо рассматривать сочетание букв от первого уровня ПДС до текущего уровня

ключа класса s_i и ключа s_is_j, содержащегося в этом классе: $\kappa(s_i,n)\leq\kappa(s_is_j,1)$. При s_j=е[*)] s_i является ключом класса из одного ключа. Из этого же неравенства следует, что длина ключа класса с ключом s_j не превосходит минимума длин ключей всех ключей, входящих в этот класс $\kappa(s_i,n)\leq\min\{\kappa(s_is_j,1)\}$ по всем ключам s_is_j. Аналогичное неравенство связывает длину ключа класса и среднюю длину ключа массива. Длина ключа s_i класса не превышает среднюю длину ключей, входящих в этот класс $\kappa(s_i,n)\leq M\kappa(s_i,1)$. Из рассмотренных неравенств следует, что для сравнения неравномерного классификатора с равномерным также можно использовать величину разности между средней длиной ключей в классе и средней длиной ключа класса $\Delta M\kappa(n)=M\kappa(1)-M\kappa(n)$. Соответствующая величина для равномерного случая $\Delta k(n)=k(1)-k(n)$. Тогда $\Delta k(n)\leq\Delta M\kappa(n)$ в силу того, что в неравномерном случае средняя длина ключа не меньше длины ключа в равномерном (см. выше). Разница между этими величинами может характеризовать степень отличия в среднем неравномерного случая от равномерного. Реальный пример показывает, что равномерный и неравномерный случаи могут отличаться достаточно сильно. Классификатор по полю ФИО[34 657] даёт среднюю длину ключа $M\kappa(1)\approx9{,}465$ с разбросом $\sigma_\kappa(1)\approx5{,}223$, в то время, как для равномерного случая получается величина $k(1)\approx3{,}073$.

2.3. Модельные распределения ключей по буквенным сочетаниям

2.3.1. Различные распределения ключей по буквенным сочетаниям

Вначале необходимо рассмотреть различные случаи распределения ключей по буквенным сочетаниям. Пусть имеется набор из N текстовых ключей, упорядоченных по алфавиту. Идеальным случаем распределения является равномерный случай, когда все буквенные сочетания присутствуют на любую глубину префиксного дерева сочетаний (ПДС) или при любой длине

*) е — пустая строка

ключа. Число ключей длиной k определяется как $N=a^k$, где a — это мощность алфавита. Отсюда можно получить логарифмическую зависимость максимальной длины ключа от общего числа ключей $k_m=\log_a N$. При этом каждый ключ длиной k_m определяет только один ключ исходного массива. Если рассмотреть классы по n ключей, то длина ключа класса вычисляется по формуле (1).

Таким образом, задавая необходимое число ключей в классе n, можно однозначно получить соответствующую длину ключа, которая определяет число уровней в ПДС для построения искомого многоуровневого классификатора. Для равномерного распределения слов величины k и n не являются случайными. Случай, противоположный равномерному распределению, является неравномерное распределение ключей по сочетаниям для любой длины ключа k. Тогда распределение случайной величины k является произвольным распределением, в общем случае, мультимодальным. Соотношение (1) для $k(n)$ переходит в некоторую регрессионную зависимость между средними величинами $\bar{k}$ и $\bar{n}$. Среди всех распределений ключей по сочетаниям можно выделить такие , для которых ключи распределены неравномерно на некоторое заданное число начальных уровней ПДС, а на остальных уровнях ПДС можно считать распределение слов равномерным. Это является абстрактным случаем, поэтому будем называть такие распределения ключей модельными.

Основной задачей является построение такого многоуровневого классификатора, который позволяет за оптимальное число шагов переходить к заданному ключу (или к наиболее близкому к заданному, если такового не существует) в упорядоченном по алфавиту наборе из N ключей.

2.3.2. Равномерный случай распределения ключей по буквенным сочетаниям

Если число букв в алфавите a=30, число ключей в массиве равно N=21.870.000.000, то для равномерного распределения из (1) получаем $k=k_m$=7 при n=1, т.е. все ключи состоят из 7 символов. При n=900 число уровней

классификатора из (1) равно k=5. При числе ключей в классе n, отличном от целых степеней a, k принимает дробные значения. В общем случае $a^{i-1} \le n \le a^i$, i=1,…,k_m. Тогда число уровней берется равное целой части [k] при $a^i/2 \le n \le a^i$, i=1,…,k_m и [k]+1 при $a^{i-1} \le n < a^i/2$ (см. пп. 2.2.3, 2.2.4). Легко понять, что равномерное распределение является чисто теоретическим (см. таблицу 2.1) и в реальной жизни такое распределение ключей не встречается.

Таблица 2.1

Длины ключей для равномерного распределения и для реальных индексов*)

Поле	N	k_{mp}	**M[κ_m]**	**σ[κ_m]**
ФИО	34 657	3,073404	9,465418	5,223059
ФИОРодства	1 775	2,198874	8,703662	4,103734
ФИОЗаявителя	1 780	2,199683	7,452809	5,435890

В таблице 2.1 k_{mp} — длина ключа при равномерном распределении для соответствующих N; M[κ_m] — мат. ожидание случайной величины максимальной длины ключа κ_m для одного ключа в классе n=1 для разных полей. В последней колонке указаны соответствующие среднеквадратичные отклонения. Данные в таблице 2.1 показывают, что величины M[κ_m] значительно отличаются от соответствующих значений k_{mp} для равномерного случая. Вследствие этого длина ключа для равномерного случая может служить лишь нижней границей для значений длины ключа в неравномерном случае.

2.3.3. Неравномерный случай распределения ключей по первой букве

Рассмотрим модельное распределение ключей по сочетаниям на первую букву ключа, а по остальным буквам будем считать распределение равномерным. Пусть P_i — вероятности встречаемости букв на первой позиции, $\sum P_i$=1, i=1,…,a. Тогда можно записать, что вероятность появления ключа в каком-либо классе из n ключей равна с одной стороны n/N, а с другой стороны

*) Статистические данные в таблице 2.1 посчитаны для соответствующих полей в базе данных “За Христа пострадавшие”.

$P_i(1/a)^{ki-1}$. Все вероятности появления букв дальше первой позиции считаются одинаковыми и равными $1/a$. Имеем, $k_i=1+\log_a(P_iN/n)$. Отсюда мат. ожидание $M[\kappa_{\mu l}]=\sum k_iP_i$ равно:

$$M[\kappa_{\mu_1}] = 1 + \sum_{i=1}^{a} P_i \log_a P_i + \log_a(N/n), \qquad (2)$$

где P_i — вероятности встречаемости букв на первой позиции, $i=1,\ldots,a$,

a — мощность алфавита,

N — число ключей в массиве,

n — число ключей в каждом из классов, на которые разбивается массив в алфавитном порядке следования ключей.

Величина $M[\kappa_{\mu l}]$ в формуле (2) отличается от $k_p=k(n)$ в формуле (1) для равномерного случая на величину $1+\sum P_i\log_aP_i$. Ниже будет показано, что эта величина не меньше 0. Это означает, что $M[\kappa_{\mu l}]\geq k_p$, т.е. неравномерный случай не лучше равномерного. Получаем, что дисперсия случайной величины длины ключа равна $D[\kappa_{\mu l}]=M[\kappa_{\mu l}^2]-M^2[\kappa_{\mu l}]=\sum P_i(1+\log_a(P_iN/n))^2-M^2[\kappa_{\mu l}]$, $i=1,\ldots,a$.

При рассмотрении верхнего уровня ПДС для алфавитного индекса по полю ФИО[*)] можно получить распределение ключей по буквам алфавита. Соответствующие частоты встречаемости ключей приводятся в следующей таблице для двух ПДС, построенных по полям ФИО$^{32\ 127}$ и ФИО$^{34\ 657}$.

Таблица 2.2

Пример неравномерного распределения ключей по первой букве для поля ФИО

Буква	**Част.1**		**Част.2**		**2 ур.**	**Буква**	**Част.1**		**Част.2**		**2 ур.**	
А	1892	0,059	2016	0,058	23	Р	1026	0,032	1088	0,032	12	
Б	2379	0,074	2462	0,071	11	С	3287	0,102	3538	0,102	16	17
В	1798	0,056	1905	0,055	16	Т	1129	0,035	1301	0,038	12	

[*)] Данные по индексу поля ФИО для БД “За Христа пострадавшие” версий 2008 и 2014гг. Числа в верхнем индексе у названия поля обозначают общее число вершин N на ключевом уровне.

Г	1548	0,048	1630	0,047	10	У	337	0,011	348	0,010	19	
Д	1185	0,037	1251	0,036	17	Ф	656	0,020	767	0,022	7	
Е	676	0,021	729	0,021	16	Х	312	0,010	376	0,011	9	
Ё	0		0	0	0	Ц	179	0,006	198	0,006	7	8
Ж	308	0,010	323	0,009	10	Ч	572	0,018	667	0,019	11	
З	789	0,025	828	0,024	16	Ш	732	0,023	892	0,026	12	
И	963	0,030	1041	0,030	20	Щ	126	0,004	154	0,004	4	
Й	0		0	0	0	Ь	0	0	0	0	0	
К	3694	0,115	3874	0,112	15	Ы	0	0	0	0	0	
Л	1372	0,043	1452	0,042	9	Ъ	0	0	0	0	0	
М	2262	0,070	2406	0,070	13	Э	20	0,001	22	0,001	8	9
Н	1229	0,038	1313	0,038	6	Ю	123	0,004	152	0,004	15	
О	655	0,020	701	0,020	18	Я	257	0,008	291	0,008	18	
П	2621	0,082	2932	0,085	15	∑	32127	1,002	34657	1,001	365	368

Колонки Част.1 и Част.2 обозначают частоты встречаемости ключей для вышеуказанных версий индекса по полю ФИО. В тех же колонках приводятся соответствующие частотные вероятности букв. Колонка 2 ур. обозначает число ключей на втором уровне ПДС. Если в этой колонке приводятся два числа, то они соответствуют различным версиям ПДС для поля ФИО. В случае одного числа значения для разных версий ПДС совпадают. В последней строке приводится сумма всех строк по колонкам. Частотные вероятности могут иметь ошибки округления, поэтому их сумма может несколько отличаться от 1. Из таблицы видно насколько неравномерно распределение слов по первой букве — вероятности лежат в диапазонах [0.001(Э);0.115(К)] и [0.001(Э);0.112(К)] для первой и второй версии ПДС соответственно. Эти вероятности отличаются от средних вероятностей букв русского алфавита (Ляшевская, 2009). Важно отметить, что расположение букв в порядке возрастания частот не изменяется в рассматриваемых версиях ПДС. Из формулы (2) при n=1 для данных таблицы 2 получаем: $M[\kappa_{\mu 1 m}{}^{(1)}]\approx 3{,}168$ и $M[\kappa_{\mu 1 m}{}^{(2)}]\approx 3{,}189$ для первого и второго ПДС

соответственно. Эти значения для модельного распределения мало отличаются от равномерного случая и значительно меньше мат. ожидания для неравномерного распределения (см. таблица 1.1): k_{mp}=3,073404<M[$\kappa_{\mu m}^{(2)}$]<< 9,465418=M[κ_m].

2.3.4. Неравномерный случай распределения ключей по двум и более начальным буквам

Из колонки ”2 ур.” числа букв на втором уровне ПДС (см. таблицу 1.2) видно, что по второй букве распределение ключей будет также неравномерным, т.к. отсутствует значительная часть букв алфавита на второй позиции (только для букв А,Д,И,О,С,У,Я присутствует более половины алфавита). Из общего числа начальных биграмм в таблице 2.2 (365 и 368 для двух версий ПДС) видно, что реализованных в языке сочетаний меньше возможного числа приблизительно в 2,5 раза (относится к начальным биграммам). Таблицы частот биграмм русского языка приводятся в книге Алфёрова (Алфёров, 2002, с.441-442). По аналогии с неравномерным распределением по первой букве для случая двух букв можно получить:

$$M[\kappa_{\mu_2}] = 2 + \sum_{i=1}^{a^2} P_i \log_a P_i + \log_a (N/n) \qquad (3)$$

Основное отличие формулы (3) в случае неравномерного распределения по двум первым буквам от формулы (2) в том, что все вероятности являются условными P_i=$P_i(\gamma_2|\gamma_1)$, где γ_1 и γ_2 обозначают события появления буквы на соответствующих позициях, а индекс i пробегает по всем возможным биграммам, число которых равно a^2. По аналогии можно записать общую формулу для любого неравномерного распределение на заданное число букв m. Число начальных m-грамм равно a^m.

$$M[\kappa_{\mu_m}] = m + \sum_{i=1}^{a^m} P_i \log_a P_i + \log_a (N/n) \qquad (4)$$

Формула (4) представляет собой лишь некоторую математическую модель. Такая модель хорошо показывает отличие неравномерного случая от

равномерного и связана с понятием, именуемым *энтропией*. Представление сути явления в наглядном виде и лаконичность этой модели являются основанием для её применения. Энтропия представляет собой ожидаемое число битов информации при наступлении случайного события (Кнут,2007,т.3,с.477): $H(P_1,P_2,...)=\sum_{i=1}^{a^m} P_i \log_a \frac{1}{P_i}$. С учётом формулы (1) получаем из (4):

$$M[\kappa_{\mu_m}] = m - H(P_1,P_2,...) + k(n) \qquad (5)$$

Из формулы (5) видно, что предлагаемая модель объясняет отличие мат. ожидания случайной величины длины ключа $M[\kappa_{\mu m}]$ при неравномерном распределении от длины ключа $k(n)$ в равномерном случае наличием энтропии $H(P_1,P_2,\ldots)$. В равномерном случае все вероятности P_i равны и энтропия $H(P_1,P_2,\ldots)$ принимает максимальное значение m (см. ниже). В этом случае $M[\kappa_{\mu m}]=k(n)$ и длина ключа становится достоверной величиной.

Из рассматриваемого примера (см. строка 1 в таблице 2.1) видно, что величина M для модельного распределения, полученная по формуле (4), приближается к величине $M[\kappa_m]=9{,}465418$ для общего неравномерного распределения по полю ФИО$^{34\,657}$ при $m=[M[\kappa_m]]$[*)], т.е. в данном случае при $m=9$ получаем $M[\kappa_{\mu 9}]\approx 9{,}334$. Это означает, что в общем случае слова неравномерно распределены по сочетаниям букв на всю глубину ПДС. Как выше было определено, величина $\Sigma P_i \log_a(1/P_i)$ в формуле (4) называется энтропией системы (Лапа,1974), представляющей собой классификатор: $H(S)=-\Sigma P(s_i)\log_a P(s_i)=I_{\text{ср}}(S)$, где S — множество начальных n-грамм, $P_i=P(s_i)$ — вероятность n-граммы $s_i \in S$. $H(S)$ также совпадает со средним количеством информации $I_{\text{ср}}(S)$, которое в среднем должно иметься для того, чтобы выделить начальную n-грамму из заданного множества n-грамм. n-граммы классификатора можно рассматривать как состояния системы, а вероятность P_i, $i=1,\ldots,a^m$, как вероятность одного из a^m возможных состояний системы. В

[*)] Внешние квадратные скобки обозначают целую часть от числа

случае отсутствия сочетания соответствующая вероятность P_i=0, а предел $P_i\log_a P_i$ равен 0 при P_i, стремящемся к 0. Этот предел получается, если применить правило Лопиталя (Корн,1970) для раскрытия неопределённости типа "∞/∞" к пределу функции $\ln x/(1/x)$ при x, стремящемся к 0. Энтропия множества символов равна количеству информации, которое может содержать какой-либо символ. Величина $M[\kappa_\mu]$ для модельного неравномерного распределения в формуле (4) должна быть не меньше величины $k(n)$ для равномерного случая в формуле (1), т.е. случай неравномерного распределения не лучше равномерного. Это означает, что $m+\Sigma P_i\log_a P_i\geq 0$. Это доказывает следующая теорема (Лапа,1974).

Теорема 1. Энтропия $H(S)$ удовлетворяет неравенству $H(S)\leq\log_a|S|$, где $|S|=a^m$, причём знак равенства есть только при $P(s_i)=1/|S|$, $\forall s_i\in S$.

Доказательство. Имеем:

$$H(S)-\log_a|S|=\sum_{i=1}^{a^m}P(s_i)\log_a(1/P(s_i))-\sum_{i=1}^{a^m}P(s_i)\log_a|S|=\sum_{i=1}^{a^m}P(s_i)\log_a(|S|P(s_i))^{-1}\leq$$

$$\leq\sum_{i=1}^{a^m}P(s_i)\left((|S|P(s_i))^{-1}-1\right)\log_a e=\sum_{i=1}^{a^m}(1/|S|-P(s_i))\log_a e=0$$

в силу неравенства $\ln x\leq x-1$. Только при x=1 это неравенство переходит в равенство, т.е. при $1/(|S|P(s_i))=1$, $P(s_i)=1/|S|$ $\forall s_i\in S$ $H(S)=\log_a|S|$.

Из доказанного неравенства следует, что для любого заданного алфавита символов количество информации, которое в среднем может содержаться в одном символе, достигает максимума, когда все символы равновероятны. В этом случае при $P(s_i)=1/|S|=a^{-m}$ из формулы (4) получаем равномерный случай распределения ключей по сочетаниям $M[\kappa_\mu]=m-\sum_{i=1}^{a^m}a^{-m}\log_a a^{-m}+\log_a(N/n)=k(n)$ (см. формулу (1)).

2.3.5. Общий неравномерный случай распределения ключей по буквенным сочетаниям

Формула (4) при m=[M[κ_m]] и n=1 соответствует общему неравномерному случаю распределения текстовых ключей по буквенным сочетаниям с одним ключом в классе. В этом случае распределение является равномерным только на последнем уровне ПДС. Каждый класс будет иметь уникальный ключ, длина которого в среднем равна M[κ_μ] (4). Если длина текстового ключа ограничена по длине некоторым значением $k_{\max}$, то M[κ_μ]$\leq k_{\max}$ и максимальное число уникальных ключей $n_{\max}$ равно a в степени $k_{\max}$.

Рассмотрим пример индекса по полю ФИО$^{34\,657}$. В таблице 2.1 строка 1 даёт длину ключа для класса из одного ключа для равномерного и неравномерного случаев распределения ключей по буквенным сочетаниям. Для неравномерного случая — это средняя величина M[κ_m]=9,465 со среднеквадратичным отклонением σ[κ_m]=5,223.

Применяя модельное распределение ключей, которое неравномерно на m начальных букв ключа, а далее равномерно, для расчёта средней длины ключа для поля ФИО$^{34\,657}$, получим, что наиболее близким к M[κ_m]=9,465 будет модельное распределение при m=[M[κ_m]]=9. В этом случае из формулы (4) мат. ожидание длины ключа равно M[κ_μ]=9,334. При этом число ключей в классификаторе 19720 меньше N=34657 по причине наличия классов, содержащих более одного ключа. В таблице 2.3 приведён пример ключей и их частот.

Таблица 2.3

Пример нескольких ключей классификатора с длиной ключа из 9 букв

Ключи на А	***n***	**Ключи на Б**	***n***	**…**	**Ключи на Ю**	***n***	**Ключи на Я**	***n***
Аарон (Ав	1	Б. Андрей	1	…	Ювеналий	1	Ябелов Ва	1
Аарон (Ко	1	Бабаев Гр	1		Ювеналия	2	Яблоков А	2
Аарон (Ку	1	Бабаев Ко	2		Ювенский	2	Яблоков Н	1
Абаимов В	1	Бабайцева	1		Ювженко И	1	Яблокова	1

Абакумов	1	Бабакаева	1		Юганов Ни	1	Яблонская	1
Абакумова	2	Бабанов М	1		Югов Алек	1	Яблонский	3
Абакумовс	1	Бабарчук	1		Югов Нико	1	Яворский	4
Абарников	1	Бабаскина	1		Югов Паве	1	Ягодин Ва	1
Абарнов И	1	Бабашкин	1		Югова Але	1	Ягодин Ев	1
Абашев Ал	2	Бабенко Е	1		Югова Евл	1	Ягодин Ст	1
Абашеев Кл ...	1	Бабенко И...	1		Югова Лид...	1	Ягодинска...	1

Из таблицы 2.3 видно, что большинство классов содержат 1 ключ.

На основании рассмотренных в данном пункте примеров можно сделать вывод о том, что модельные распределения ключей по буквенным сочетаниям дают результат для мат. ожидания длины ключа (4), который соответствует неравномерному распределению ключей по буквенным сочетаниям на всю длину ключа. Отсюда можно сделать вывод, что различные префиксы на любом уровне классификатора могут содержать различное случайное число ключей.

2.4. Проблемы построения многоуровневого алфавитного классификатора (на примере ключевого уровня массива ООСУБД НИКА)

Использование алфавитных классификаторов является известным способом организации быстрого перехода на искомое понятие. Такой переход является альтернативой к поиску данных в ключевом текстовом массиве через поле ввода и автозаполнение (Bast, 2006; Тищенко, 2013). Автозаполнение, основанное на том же подходе, что и алфавитный классификатор, было описано в статье (Тищенко,2013) и представляет собой некоторую “динамическую” форму классификатора. При большом числе ключей в ключевом массиве возникает проблема, связанная с построением оптимального с точки зрения числа переходов по уровням многоуровневого алфавитного классификатора. Эта проблема связана с построением регрессионной зависимости между средней длиной ключа в алфавитном классификаторе и средним числом ключей массива на этот ключ. Основу алфавитного классификатора составляет ПДС

(Кнут,т.1,т.3,с.530). Узлами такого дерева являются буквы или сочетания букв. Каждый из ключей массива представлен в таком дереве в виде полного пути от корня до листа. Число терминальных узлов в префиксном дереве совпадает с числом ключей на ключевом уровне массива. Каждый узел представлен очередной буквой или буквами из текущего ключа. Ветвление в дереве происходит в случае, когда у текущего ключа в данной позиции стоит буква, отличающаяся от буквы в той же позиции у предыдущего ключа. Процесс построения ПДС содержит следующие шаги.

Шаг 1. Переход к первому ключу массива.

Шаг 2. Берётся первая буква текущего ключа массива. Если среди узлов ПДС первого уровня отсутствует узел с этой буквой, то узел с этой буквой добавляется в дерево.

Шаг 3. Берётся следующая буква текущего ключа массива. Если среди узлов ПДС соответствующего уровня отсутствует узел с этой буквой, то узел с этой буквой добавляется в дерево.

Шаг 4. Если есть следующая буква, то переход к шагу 3.

Шаг 5. Переход к следующему ключу массива. Если он существует, то переход к шагу 2.

Полученное в результате построения дерево будет иметь во всех узлах по одной букве ключа массива. Такое дерево посредством объединения последовательно идущих узлов без ветвления в один узел даёт сжатое префиксное дерево, в котором узлы могут содержать более одной буквы. Будем рассматривать ПДС такого вида. Каждый путь от корня ПДС задаёт класс ключей массива, начинающихся с этого ключа. Этот путь является буквенным префиксом или ключом этого класса. В общем случае распределение ключей массива по буквенным сочетаниям является неравномерным. Это означает, что классы с ключами равной длины могут содержать различное число ключей массива. ПДС задаёт разбиение ключевого массива на классы ключей возможно различного размера с соответствующими ключами или префиксами. Такому

разбиению можно противопоставить равномерное разбиение, соответствующее равномерному распределению ключей массива по сочетаниям. В этом случае ключевой массив разбивается на классы равного размера. Можно рассмотреть наложение двух разбиений - неравномерного в виде ПДС на равномерное с классами равного размера *n*. Величина n принимается как максимальный размер класса в ПДС. Исходя из величины *n*, можно определить среднюю длину ключа *k*, необходимую для перехода к искомому ключу массива. Меняя величину *n* в заданных пределах можно построить регрессионную зависимость *k* от *n*. Предварительно будет рассмотрено разбиение на классы с помощью ПДС на примере поля ФИО, а также случайные распределения длины ключа класса и числа ключей в классе. В конце главы рассматривается нечёткая регрессионная модель зависимости.

Работа пользователя с многоуровневым классификатором на основе ПДС выглядит следующим образом. На каждом уровне классификатора пользователь выбирает искомое сочетание букв, добавляя эти буквы к ключу класса, и с последнего уровня классификатора переходит на класс ключей массива в виде традиционного представления ключевого массива, как списка, содержащего искомый ключ. Такая организация интерактивного доступа является альтернативной к поиску с помощью поля ввода. Другими словами, этот доступ может быть организован в виде меню с гиперссылками на следующий уровень классификатора, как альтернатива к набору текста на клавиатуре. Алфавитный классификатор удобно использовать на смартфоне, равно как и на любом виде устройств для доступа к ключевому уровню массива БД. Такой многоуровневый классификатор был реализован для ключевого уровня массива СУБД НИКА (Годунов, Емельянов, 1991; Тищенко, 2013). Все нижеприведённые статистические данные взяты из базы данных “За Христа пострадавшие”[*)].

[*)] БД “За Христа пострадавшие”, http://martyrs.pstbi.ru

2.4.1. Разбиение на классы с помощью ПДС

На рис.2.6 приводится фрагмент ключевого массива, а на рис. 2.7 — соответствующий фрагмент ПДС. В квадратных скобках указаны счётчики для префиксов следующего уровня и терминальных узлов в поддереве для данного ключа. Если принять, что число ключей в классе n_c не больше некоторого заданного значения $n_{max}=n$, т.е. $n_c \le n$, то ПДС можно обрывать на тех префиксах, для которых выполняется это условие. Полученное поддерево исходного ПДС можно рассматривать как основу классификатора. Однако здесь возникает проблема, связанная со сглаживанием неравномерности распределений ключей по начальным m-граммам. Неравномерность этого распределения порождает случайные распределения длины ключа класса и количества ключей массива в классе. При равномерном распределении эти величины являются достоверными и связаны строгой зависимостью, выраженной формулой (1).

Если взять максимальное число ключей в классе n=10, то фрагмент ПДС на рис. 2.7 будет развёрнут не на полную глубину (см. рис.2.8). На рис. 2.8 показан тот же фрагмент поддерева ПДС, что и на рис.2.7 Нетерминальные узлы в ПДС на рис.2.7, для которых выполнено условие $n_c \le n$=10, стали терминальными узлами в поддереве ПДС на рис.2.8. К ним относятся следующие сочетания: “Габ” (3,3), “Гавердовский” (12,2), “Гавриил (В” (10,2), “Гавриил (Г” (10,2), “Гавриил (И” (10,2), “Гавриил (К” (10,2), “Гавриила (” (10,2), “Гавриленко” (10,2), “Гаврили” (7,7). После буквенного сочетания приводится в скобках значение соответствующей двумерной случайной величины (κ, ν) длины ключа класса и числа ключей в этом классе. Нетрудно видеть, что в данном фрагменте все классы с числом ключей массива меньше 10 и длина ключа в ПДС $3 \le k \le 12$, что говорит о неравномерности распределений этих величин. Однако несколько классов имеют одинаковое число ключей массива в классе, а также встречаются классы с одинаковой длиной ключа. Более того, эти классы могут находиться рядом. Тогда возникают некоторые “участки равномерности” в случае присутствия

нескольких подчинённых букв в ПДС для данной m-граммы. Это хорошо видно в ПДС в случае, когда дерево раскрывается на одинаковую глубину на соседних узлах. Во фрагменте поддерева ПДС на рис.2.8 — это классы с ключами на “Гавриил (“):“Гавриил (А”, “Гавриил (В”, “Гавриил (Г”, “Гавриил (З”, “Гавриил (И”, “Гавриил (К”, “Гавриил (Л”, “Гавриил (П”, “Гавриил (Я”. Для этих классов выполнено условие $n_c \leq n$ и одновременно они имеют одинаковые длины ключей, отличающихся одной последней буквой. Следовательно, для данной начальной *m*-граммы, например, “Гавриил (“ можно говорить о “локальной” равномерности распределения слов по сочетаниям в смысле выполнения условия $n_c \leq n$.

Габриалович Вера Болиславовна
Габрияник Алексей Иванович
Габышев Степан Николаевич
Гаварин Николай Иванович
Гавердовский Владимир Львович
Гавердовский Николай Николаевич
Гавриил
Гавриил #1
Гавриил (Абалымов Николай Николаевич)
Гавриил (Владимиров Григорий Петрович)
Гавриил (Воеводин Григорий Дмитриевич)
Гавриил (Горбач Петр Дмитриевич)
Гавриил (Гур Гавриил Иванович)
Гавриил (Зверев Григорий Павлович)
Гавриил (Игошкин Иван Иванович)
Гавриил (Ильин /.../ Ефимович)
Гавриил (Кожухарев Гавриил Антонович)
Гавриил (Красновский Всеволод Витальевич)
Гавриил (Лихоманов Григорий Александрович)
Гавриил (Польшин Георгий Демьянович)
Гавриил (Яцик Григорий Петрович)
Гавриила (Гурцова Матрона Васильевна)
Гавриила (Чуркина Анисья Ивановна)
Гавриленко Андрей Елисеевич
Гавриленко Тимофей Дементьевич
Гаврилин Алексей Иванович

Рис.2.6. Фрагмент ключевого массива для поля ФИО[34 657]

2.4.2. Случайное распределение длины ключа класса

В общем случае распределение случайной величины æ длины ключа класса является мультимодальным, например, для индекса ФИО[32 127] при *n*=1

это распределение имеет 4 моды в точках 10, 5, 17, 29 в порядке убывания частотных вероятностей. Первые 3 моды соответствуют в среднем имени, фамилии и отчеству. Четвёртую моду можно объяснить тем, что могут быть указаны два имени для постриженных в монашество. Максимальная мода соответствует именам. Больше всего 10-буквенных ключей, содержащих фамилию и имя, идентифицирующих любой ключ из исходного набора. Индекс $ФИО^{34\,657}$ даёт такой же результат с такими же частотными вероятностями, только мода в точке 17 перемещается в точку 16. Случайное распределение числа уровней в ПДС, на которые разбивается ключ, также может служить характеристикой классификатора на основе данного ПДС. Длину ключа можно измерять не только в числе символов, но и в числе уровней в ПДС. Некоторые узлы в ПДС могут содержать более одной буквы (см.рис.2.7). Это связано с неравномерностью распределения ключей по сочетаниям. Узлы ПДС, содержащие сочетания букв, можно условно обозначить новыми буквами $\alpha_1,\alpha_2,\ldots,\alpha_c$. Полученное ПДС (назовём его обобщённым) имеет в каждом узле по одной букве и в этом смысле ближе к равномерному случаю, т.к. будут отсутствовать узлы, которым подчинена только один узел, как в случае “несжатого” исходного ПДС. Длина ключа в обобщённом ПДС имеет сглаженное распределение в сравнении с распределением в исходном ПДС. В рассматриваемых примерах индексов при n=1 это распределение унимодально с модой в точке 7. Здесь длина ключа измеряется в расширенном числе символов или уровнях ПДС. В исходном ПДС длина ключа может быть больше, т.к. одна буква в обобщённом ПДС может обозначать сочетание букв в исходном дереве. При n=10 рассматриваемое условное распределение длины ключа имеет 5 мод в точках 4, 9, 17, 20, 22 в порядке убывания частотных вероятностей для обеих версий индекса по полю ФИО. В случае обобщённого ПДС условное распределение длины ключа становится бимодальным с модами в точках 4 и 7 в порядке убывания частотных вероятностей.

Рассмотрим случайное распределение длины ключа, фиксируя размер класса n, как непрерывное, исходя из следующего рассуждения Крамера (Крамер,1975,с.192-193). “В приложениях к статистике величины непрерывного типа встречаются тогда, когда мы имеем дело с измерением величин, могущих принимать любое значение в некоторых пределах, например, цена товара, рост человека, размер урожая.

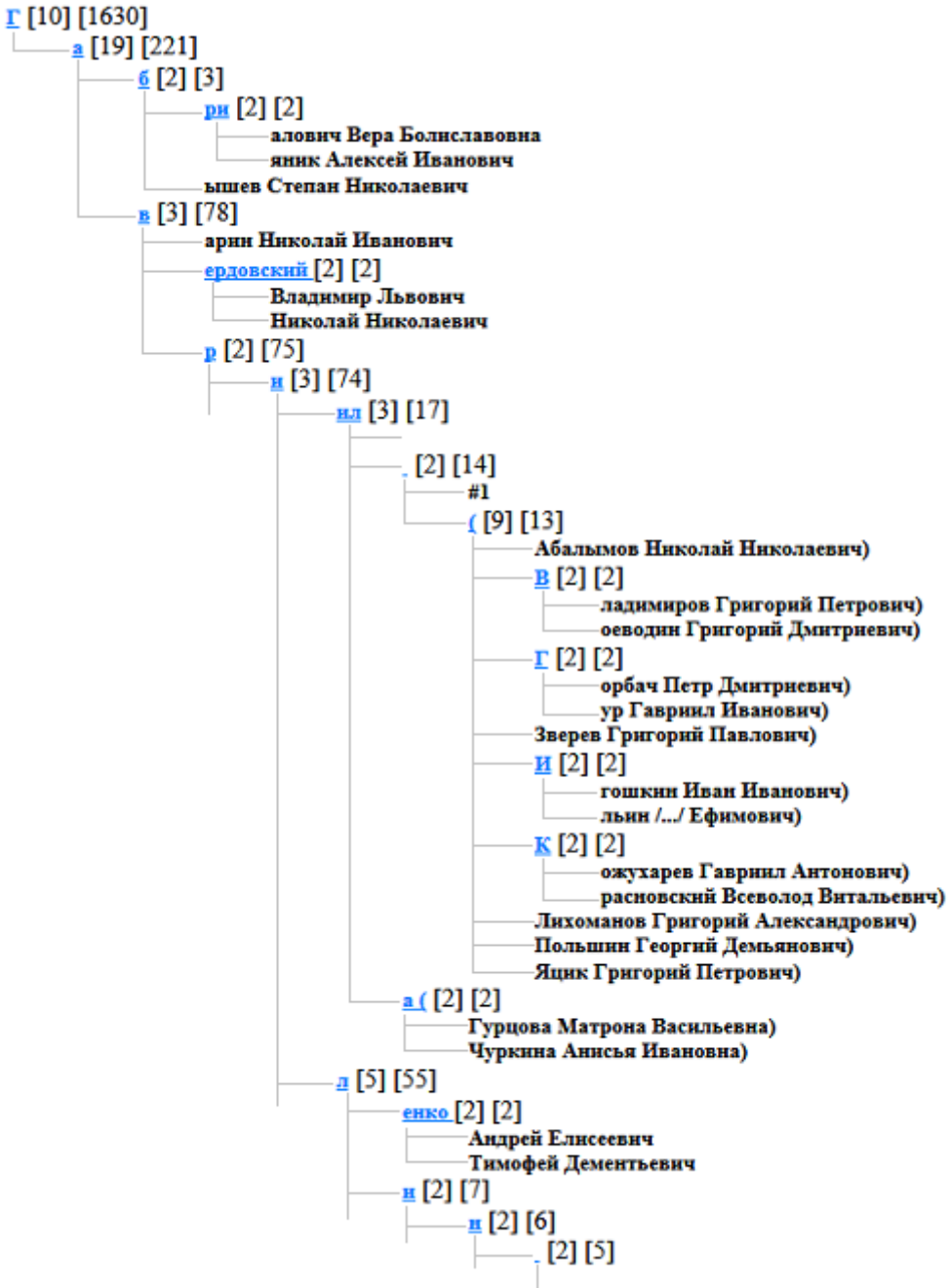


Рис.2.7. Фрагмент префиксного дерева сочетаний при одном ключе в классе для поля ФИО[34 657]

В этих случаях величины рассматриваются как непрерывные, хотя, строго говоря, фактические данные всегда разрывны, так как каждое измерение выражается целым числом, кратным наименьшей единице измерения данной величины. Так, цена выражается в денежных единицах, длина может быть выражена в сантиметрах, вес в килограммах и т.д. Таким образом, всегда, когда для теоретических целей величины этого рода рассматриваются как непрерывные, совершается некоторая математическая идеализация". Крамер приводит формулу для разложения в ряд Эджворда (Крамер, 1975,с.255) любой функции плотности. При этом он отмечает, что "на практике обычно не рекомендуется идти дальше третьего и четвертого моментов".

$$f(x) = \phi(x) - \frac{\gamma_1}{3!}\phi^{(3)}(x) + \frac{\gamma_2}{4!}\phi^{(4)}(x) + \frac{10\gamma_1^2}{6!}\phi^{(6)}(x) \qquad (6)$$

Здесь $\varphi(x)$ обозначает нормальную функцию плотности*), а $\varphi^{(3)}(x)$, $\varphi^{(4)}(x)$, $\varphi^{(6)}(x)$ — её производные 3, 4, 6 порядков соответственно, коэффициенты γ_1 и γ_2 — коэффициенты асимметрии и эксцесса соответственно. При этом предполагается, что исходная случайная величина нормирована. "При больших x выражение (6) иногда будет давать небольшие отрицательные значения для $f(x)$. Это, конечно, вполне согласуется с тем, что (6) даёт приближённое, но не точное выражение для функции плотности".

Кривые $\varphi^{(4)}(x)$, и $\varphi^{(6)}(x)$ симметричны относительно x=0, кривая $\varphi^{(3)}(x)$ "вводит в выражение (6) асимметричный элемент". Формулу (6) можно применить для аппроксимации функции плотности случайного распределения длины ключа к, подставив выборочные моменты нормированной случайной величины длины ключа. Для поля ФИО[34 657] при одном ключе в классе выборочное среднее $m_к \approx 10{,}966$ (ключ в среднем содержит 11 букв), среднеквадратическое отклонение $s_к \approx 5{,}487$. В этом случае для нормированной величины $x = (\kappa - m_к)/s_к$

*) $\phi(x) = \Phi'(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$

получаются следующие коэффициенты $\gamma_1 \approx 4{,}813$ и $\gamma_2 \approx 647{,}095$, а функция плотности распределения для случайной величины x имеет вид $f(x) = \phi(x) - 0{,}802\phi^{(3)}(x) + 26{,}962\phi^{(4)}(x) + 0{,}322\phi^{(6)}(x)$.

Рис.2.8. Фрагмент поддерева префиксного дерева сочетаний при 10 ключах в классе для поля ФИО[34 657]

2.4.3. Случайное распределение числа ключей в классе

Пусть n_c — число ключей в классе и n — максимальное число ключей в классе. Для поля ФИО[34 657] распределение числа ключей в классе близко к унимодальному с модой в точке n_c=1 при n<30, т.е. преобладают классы с одним ключом. “Близко” означает, что в точке n_c=1 – глобальный максимум, а в остальных точках могут быть незначительные колебания частотных вероятностей и небольшие локальные экстремумы. При дальнейшем увеличении n случайная зависимость “сглаживается” и перестаёт иметь характерный максимум в точке n_c=1, только наблюдается тенденция уменьшения частотной вероятности с ростом n_c. При n=100 наблюдаются

колебания частотной вероятности в произвольном виде. Дальнейшее увеличение n приводит к тому, что при значениях больших n_c~100 большая часть значений имеет нулевую частотную вероятность, т.е. классы с числом ключей более 100 встречаются значительно реже. По аналогии с распределением длины ключа класса можно записать функцию плотности распределения n_c, например, при n=1000: $f(x) = \phi(x) - 4{,}66\phi^{(3)}(x) + 113{,}673\phi^{(4)}(x) + 10{,}858\phi^{(6)}(x)$, где $x=(n_c-m_n)/s_n$, m_n=396,021 и s_n=238,249.

2.4.4. Регрессионная зависимость длины префикса от максимального числа ключей в классе

Строгая функциональная зависимость $k(n)$, определённая формулой (1), для равномерного распределения слов по начальным n-граммам представлена на рис.2.9.

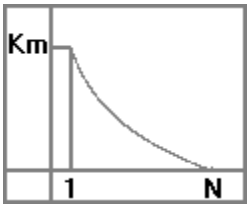


Рис.2.9. Зависимость $k(n)$

Здесь n — число ключей в классе, $1 \le n \le N$; k — длина буквенного ключа для класса, $0 \le k \le k_m$. Любое целое значение n из области определения задаёт соответствующее целое значение k из области значений. В обратную сторону также верно.

В произвольном случае длина ключа κ в классификаторе и число ключей в классе ν — случайные величины, а зависимость (1) переходит в регрессионную зависимость между мат. ожиданиями случайных величин Mκ(Mν). Для упрощения обозначения вместо букв мат. ожиданий будем использовать буквы k и n. Исходя из соотношения (1), регрессионную модель целесообразно выбрать в виде (7) (Емельянов,Тищенко,2010), заменив n на полином от n степени p.

$$k_r(n) = \ln\left(N / \left(\sum_{j=0}^{p} a_j n^j \right) \right) / \ln a \qquad (7)$$

Такая регрессионная модель может быть сведена к общей линейной (относительно параметров) модели (Дрейпер,1986) в случае, когда степень полинома p является известной. В данном случае степень полинома p считается неизвестной и также является параметром модели. В (Орлов,2006) отмечается, что в такой постановке требуется оценить объект нечисловой природы (p,a_0,a_1,a_2,…,a_p). Обычные методы оценивания для него неприменимы, т.к. p — дискретный параметр. Подчёркивается, что существующие методы оценивания степени полинома являются практическими и не дают обязательно оптимального значения p. Одним из таких методов является метод последовательного повышения степени полинома с проверкой адекватности модели по F-критерию Фишера. Этот метод используется в данной работе для построения полиномиальной регрессии на основе формулы (7), преобразованной так, чтобы в правой части равенства стоял полином.

Рассмотрим пример данных для величин n и k по полю ФИО[34 657].

Таблица 2.4

Данные для величин n и k по полю ФИО[34 657]

n	4,826	9,223	13,947	17,964	22,681	27,412	30,732	34,154	38,361	42,650
k	6,703	5,717	5,142	4,799	4,490	4,244	4,082	3,959	3,817	3,690

Для упрощения расчётов рассмотрим в качестве n максимальные значения вместо выборочных средних.

Таблица 2.5

Данные для величин $n \equiv n_{\max}$ и k по полю ФИО[34 657]

n	10	20	30	40	50	60	70	80	90	100
k	6,703	5,717	5,142	4,799	4,490	4,244	4,082	3,959	3,817	3,690

Как видно из таблицы 2.5 значения величины n являются целыми и увеличиваются с фиксированным шагом. В случае равноотстоящих значений удобно использовать регрессию на ортогональных полиномах (Дрейпер,2007), преобразовав выражение (7) к полиному от n.

$$y_r(n) = \sum_{j=0}^{p} a_j n^j / C' = \sum_{j=0}^{p} a_{cj} n^j \quad (8)$$

где y_r — преобразованная модельная переменная k_r из формулы (7) посредством формулы (9),

n — максимальное число ключей в классе,

a_j — полиномиальные коэффициенты, являющиеся параметрами модели,

p — степень полинома, являющаяся параметром модели,

C' — некоторая константа.

Экспериментальные значения переменной k можно преобразовать к переменной y посредством формулы $e^{\ln N - \ln ak(n)} / C = y(n)$ или

$$a^{k_m - k(n)} / C' = y;\ k = k_m - \log_a (C'y) \quad (9)$$

Здесь k_m=$\log_a N$. В рассматриваемом случае k_m=255 символов[*)], а число букв в алфавите принимается равным a=30. Константа C' выбирается так, чтобы y менялось в пределах от сотых до нескольких тысяч. В данном случае C'=10^{368}. Преобразование (9) приводит к полиномиальной регрессионной зависимости (8) $y_r(n)$. В таблице 2.6 приводятся данные для преобразованной переменной y в зависимости от x. Здесь

$$x = \frac{n}{10} - \frac{11}{2} \quad (10)$$

является свободной переменной ортогональных полиномов, значения которой получаются из таблиц значений ортогональных полиномов (Большев,1983) при 10 измерениях переменной y, вычисленной по формуле (9) при подстановке в качестве k_r значений k из таблицы 2.5.

Таблица 2.6

Данные для переменных x и y

[*)] В СУБД НИКА максимальная длина текстового ключа равна 255 символов

x	-9/2	-7/2	-5/2	-3/2	-1/2	1/2	3/2	5/2	7/2	9/2
y	0,058	1,664	11,764	37,776	108,055	249,469	432,824	657,653	1065,978	1641,884

Уравнение регрессии, выраженное через ортогональные полиномы $\Psi_j(x)$, принимает вид:

$$y_r(x) = \sum_{j=0}^{p} b_j \Psi_j(x) \tag{11}$$

Оценки неизвестных коэффициентов определяются методом наименьших квадратов по следующим формулам.

$$b_j = \frac{1}{S_j^2} \sum_{i=1}^{m} y(x_i)\Psi_j(x_i), \text{ где } S_j^2 = \sum_{i=1}^{n} \Psi_j^2(x_i) \tag{12}$$

Здесь $\Psi_j(x_i)$ обозначены значения ортогональных полиномов степени j (Большев,1983). Оценки коэффициентов b_j имеют дисперсию σ^2/S_j^2. Неизвестная дисперсия σ^2 имеет эффективную оценку в виде оценки остаточной дисперсии, определяемую формулой (Большев,1983):

$$s^2(p) = \frac{1}{m-p-1}\left(\sum_{i=1}^{m} y_i - \sum_{j=1}^{p} b_j^2 S_j^2 \right) \tag{13}$$

В случае неизвестной степени p аппроксимирующего полинома уравнения регрессии (11) в (Кобзарь,2006) рекомендуется определять p путём последовательных уточнений. Критерием для прекращения процедуры уточнения степени полинома является величина остаточной дисперсии (13). Если $s^2(p+1)>s^2(p)$, то в качестве регрессии принимается полином степени p. Значимость дисперсий $s^2(p+1)$ и $s^2(p)$ проверяется F-критерием Фишера при m-p и m-p-1 степенях свободы. В статье о методах снижения размерности (Орлов,Луценко,2016) приводится состоятельная оценка величины p, как выражение от среднего квадрата ошибки α_p: $p^* = Arg \min_p \{\alpha_{p+1} - 2\alpha_p + \alpha_{p-1}\}$.

Однако в (Орлов,2006) подчёркивается, что “излишнее усложнение статистических моделей вредно” и отмечается, что оценка остаточной дисперсии $s^2(p)$ будет колебаться около предела $\lim_{m->\infty} s^2(p) = \sigma^2$. Поэтому в качестве оценки неизвестной степени многочлена можно использовать,

например, первый локальный минимум оценки остаточной дисперсии $s^2(p)$, несмотря на то, что эта оценка степени полинома не является состоятельной и предельное распределение этой оценки является геометрическим (Орлов,2006). В результате применения формул (12) для определения коэффициентов ортогональных полиномов и формул для остаточных дисперсий можно получить следующую таблицу.

Таблица 2.7

Оценки коэффициентов уравнения регрессии

j	0	1	2	3	4	5	6	7
b_j	420,713	81,159	64,209	2,040	0,886	1,013	-0,243	-0,144
S_j^2	306380,499	72969,991	5649,474	637,557	316,042	194,783	246,675	66,074
R_j^2			12,916	8,861	2,017	1,623	0,790	3,733
$F_{0,95}$			3,73	4,21	4,95	6,26		

Здесь $R_j^2=S_{j-1}^2/S_j^2$. Из предпоследней строки таблицы для b_6 следует, что $S_5^2 / S_6^2 \approx 0{,}790 < 1$. Это означает, что в качестве оценки степени полинома можно взять пятую степень, т.к. она соответствует первому локальному минимуму остаточной дисперсии. С помощью F-критерия Фишера этот результат можно уточнить. Из последней строки видно, что $S_3^2/S_4^2 \approx 2{,}017 < F_{0,95}(m-p, m-p-1) = F_{0,95}(5,4) = 4{,}95$. Это означает, что остаточная дисперсия уменьшается незначимо при переходе от третьей степени к четвёртой. Следовательно, аппроксимирующий полином имеет третью степень. По критерию Стьюдента при уровне значимости α=0,05 все коэффициенты регрессии являются значимыми (см. неравенство (14) и таблицу 2.8).

$$| b_j | / S_{bj} > t(6;0{,}975) = 2{,}447 \qquad (14)$$

Таблица 2.8

Значимость коэффициентов регрессии и их 95%-ные доверительные интервалы

j	0	1	2	3
b_j	420,713	81,159	64,209	2,040
b_j/S_{bj}	52.690	58.390	29.216	7.485
$b_j \pm t(6;0{,}975)S_{bj}$	[401,2; 440,3]	[77,8; 84,6]	[58,8; 69,6]	[1,4; 2,7]

В результате получаем регрессионную зависимость:

$$y_r(x) = 420{,}713 + 81{,}159\Psi_1(x) + 64{,}209\Psi_2(x) + 2{,}04\Psi_3(x) \qquad (15)$$

В таблице 2.9 приводятся данные для y_r, полученные из модели (15).

Таблица 2.9

Данные по остаткам

y	0,058	1,664	11,764	37,776	108,055	249,469	432,824	657,653	1065,978	1641,884
y_r	-10,165	9.581	22.120	47.859	107.201	220.550	408.310	690.886	1088.682	1622.101
r	10,223	-7.916	-10.356	-10.083	0.854	28.919	24.513	-33.233	-22.704	19.783

Проверка статистик по остаткам (Дрейпер,2007) основывается на предположении, что остатки должны иметь сходство с наблюдениями из нормального распределения (для применения F-критерия) со средним, равным нулю. Среднее остатков равно $\sum r_i / m = 0$, также статистика $T_{11} = \sum r_i y_{ri} \approx 0$ (линейный член правильно включён в модель). В (Орлов,2005) отмечается, что модели на основе нормального распределения, как правило, неадекватны реальной ситуации, но позволяют глубже изучить суть рассматриваемого явления и пригодны для первоначального анализа. Следовательно, в рассматриваемом случае правильно принять не термин “адекватность модели”, а термин “модель с меньшей ошибкой” в смысле критерия Фишера в предположении, что остатки распределены нормально с нулевым средним и одинаковой дисперсией. Регрессионная зависимость описывает разброс данных относительно среднего $\bar{y}$=420,7 (k=4,091) на R^2*100%=99,86% (квадрат множественного коэффициента корреляции между y и y_r) . В результате проверки полезности регрессии (Дрейпер,2007) методом Бокса-Уэтса получаем $(y_{r\max} - y_{r\min}) / (ps^2 / n)^{1/2} \approx 4{,}048 \geq 4$ и практическое правило “четырёхкратного превышения” выполняется.

После подстановки выражений для ортогональных полиномов (Большев,1983) получаем зависимость $y_r(x) = 155{,}849 + 112{,}499x + 32{,}105x^2 + 3{,}401x^3$. При переходе (10) от переменной x к переменной n получаем зависимость $y_r(n) = 0{,}003n^3 - 0{,}240n^2 + 6{,}798n - 57{,}561$. Наконец, с помощью обратного преобразования (9) получаем регрессионную зависимость $k_r(n)$ и с помощью границ 95%-ого доверительного интервала зависимости $k_{rmin}(n)$ и $k_{rmax}(n)$ (см. последнюю строку в таблице 2.8)

$$k_r(n) = 5{,}867 - \log_{30}(0{,}0034n^3 - 0{,}240n^2 + 6{,}798n - 57{,}561) \text{ при } 14 < n \le 1000, \quad (16)$$

$$k_{r\min}(n) = 5{,}867 - \log_{30}(0{,}0029n^3 - 0{,}179n^2 + 5{,}889n - 100{,}258), \quad (16a)$$

$$k_{r\max}(n) = 5{,}867 - \log_{30}(0{,}0045n^3 - 0{,}397n^2 + 12{,}882n - 106{,}176), \quad (16b)$$

а также аналогичным методом получаем зависимость для среднего числа уровней алфавитного классификатора от максимального числа ключей в классе:

$$k_r(n) = 5{,}867 - \log_{30}(0{,}422n^2 - 17{,}306n + 178{,}517) \text{ при } 20 < n \le 1000. \quad (16')$$

Аппроксимирующий полином в (16') имеет меньшую степень, чем в (16). Это можно объяснить тем, что распределение среднего числа уровней имеет "сглаженное" распределение (меньшее число мод) в сравнении с распределением средней длины ключа.

Из формулы (16) получаем следующие данные для k_r (r_k=k-k_r).

Таблица 2.10

Экспериментальные и модельные значения длины ключа при различном максимальном числе ключей в классе

n	10	20	30	40	50	60	70	80	90	100
k	6,703	5,717	5,142	4,799	4,490	4,244	4,082	3,959	3,817	3,690
k_r	5,867	5,321	5,152	4,953	4,677	4,426	4,220	4,048	3,903	3,777
r_k	0,836	0,396	-0,010	-0,154	-0,187	-0,182	-0,138	-0,089	-0,086	-0,087

На рис.2.10 представлен график регрессионной зависимости $k_r(n)$ (16) в диапазоне $15 \le n \le 1000$. При этом средняя длина ключа уменьшается от 6,1 до 1,5. Схематично показана область*), соответствующая 95%-ному доверительному интервалу, внутри которой располагается регрессионная зависимость $k_r(n)$. Линия над зависимостью $k_r(n)$ соответствует $k_{r\min}(n)$ (16a), линия под зависимостью $k_r(n)$ соответствует $k_{r\max}(n)$ (16b).

*) Рисунок не отражает сужение доверительной области при увеличении n. Сужение происходит по причине того, что аппроксимирующий полином стоит под логарифмом

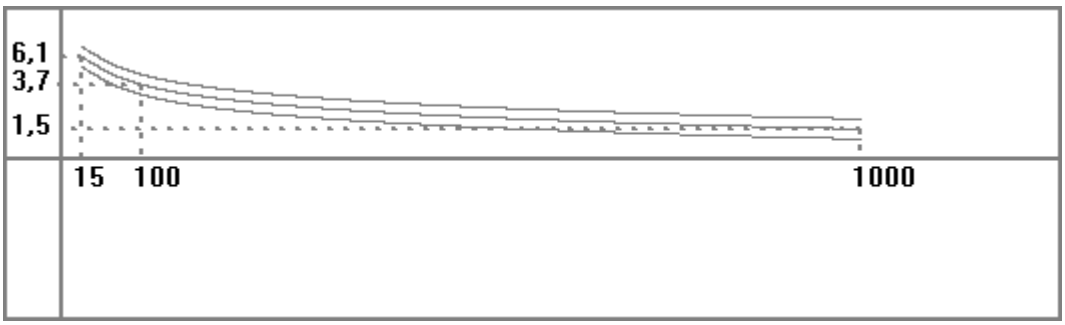


Рис.2.10. Регрессионная зависимость $k_r(n)$

Если округлять не до тысячных долей, а до миллионных, то можно достичь большей точности приближения. Значение k_r=5,867 при n=10 отличается от исходного на большую величину, чем остальные значения по причине того, что аппроксимирующий полином даёт значение -10,165 (см. таблицу 1.10), выпадающее из области определения обратного преобразования (11), т.к. выражение под логарифмом должно быть строго большим нуля. Значение под логарифмом в этой точке взято за единичное. Решение этой проблемы возможно с помощью использования метода наименьших квадратов при наличии ограничений (Дрейпер,2007). В каждой точке x_i, в которой оказывается отрицательное значение полинома $y_r(x_i)$, к искомому регрессионному уравнению (9) добавляется соответствующее ограничение $y_r(x_i)>0$. Однако представляется более эффективным в данном случае использовать более частный метод, который сводится к тому, чтобы рассмотреть другой диапазон и шаг, для построения регрессии. Например, можно взять $2\le n\le 20$ с шагом 2 и построить регрессионную зависимость (16'') отдельно для этих точек.

$$k_r(n) = 10{,}606 - \log_{30}(220{,}5625n^4 - 3288{,}5n^3 + 22127{,}12n^2 - 61711n + 55396{,}8) \text{ при } 3 < n \le 100 \qquad (16'')$$

Рассмотрим значения величин n и k, отличные от тех, по которым строилась регрессионная зависимость. Точки внутри заданного диапазона $10\le n\le 100$ будут давать значения величин k, близкие к экспериментальным, например, n=65 даёт k_r=4,318, что хорошо согласуется с экспериментальным значением k=4,163 и r_k=-0,155. Однако построенная зависимость не даёт правильных значений при n<15, т.к. при этих значениях n аппроксимирующий

полином, стоящий под логарифмом, принимает отрицательные значения.

Интересно рассмотреть точки вне заданного диапазона при n>100. В следующей таблице сведены данные для $100 \le n \le 1000$.

Таблица 1.11

Экспериментальные и модельные значения длины ключа вне диапазона регрессионной зависимости

n	100	200	300	400	500	600	700	800	900	1000
k	3,690	3,019	2,678	2,366	2,272	2,147	2,071	1.966	1.916	1.916
k_r	3,777	3,026	2,626	2,351	2,142	1,973	1,831	1.709	1.601	1.506
r_k	-0,087	-0,007	0,052	0,015	0,13	0,174	0,240	0,257	0,314	0,410

Величина ошибки r_k вначале является незначительной и близка к тем значениям, которые получаются внутри диапазона. Однако с n=500 величина r_k начинает заметно расти так, что построенной регрессионной зависимостью (16) не рекомендуется пользоваться при n>1000.

Зависимости $k_r(n)$, подобные (16), можно построить для других индексируемых полей БД с текстовым ключом, например, для ФИОЗаявителя^{1938} и ФИОРодства$^{2596*)}$. Используя описанный метод, основанный на ортогональных полиномах, получаются следующие зависимости. Для поля ФИОЗаявителя^{1938}:

$$k_r(n) = 5{,}867 - \log_{30}(34034{,}6n - 677668) \text{ при } 19 < n \le 300 \qquad (17)$$

$$k_r(n) = 5{,}867 - \log_{30}(33461{,}8n - 630351) \text{ при } 18 < n \le 300 \qquad (17')$$

и для поля ФИОРодства^{2596}

$$k_r(n) = 5{,}867 - \log_{30}(146{,}045n^2 + 1115{,}25n - 51661) \text{ при } 15 < n \le 300 \qquad (18)$$

$$k_r(n) = 5{,}867 - \log_{30}(147{,}055n^2 + 849{,}55n - 36691) \text{ при } 12 < n \le 300 \qquad (18')$$

Здесь для примера приводятся также зависимости (17') и (18') среднего числа уровней алфавитного классификатора от максимального числа ключей в классе. Эти зависимости близки к соответствующим зависимостям (17) и (18) для средней длины ключа. Эти зависимости начинают совпадать с (17') и (18') соответственно при $n \ge 40$ для обоих полей. Как видно из формул (16), (17), (18)

*) В верхнем индексе указано количество элементов на ключевом уровне соответствующего массива

степень полинома под логарифмом возрастает с увеличением числа ключей на ключевом уровне. Вероятно, это можно объяснить увеличением разброса случайной величины длины ключа при большем числе ключей. Соответственно и верхняя граница диапазона для случайной величины n максимального числа ключей в классе, при которой применимы регрессионные зависимости (17), (18), уменьшается с 1000 до 300. Нижняя граница величины n определяется как максимальное целое положительное значение n (меньшее верхней границы), при котором значение полинома, стоящего под логарифмом будет не больше нуля. Для получения регрессионных зависимостей при n, меньших нижней границы можно использовать указанный выше метод построения регрессии на ортогональных полиномах на соответствующем диапазоне $2 \le n \le 20$ с шагом 2. Из примера зависимости (16'') для этого диапазона видно, что на степень аппроксимирующего полинома может также влиять диапазон свободной переменной n и шаг изменения этой переменной. Особенностью зависимости среднего числа уровней (16') является то, что полином под логарифмом имеет экстремум в окрестности точки $n=21$ ($20 < n < 22$) и зависимость $k_r(n)$ в этой точке достигает максимального значения на целых значениях n. Таким образом, для того, чтобы сохранить убывающий характер зависимости $k_r(n)$, следует рассматривать $20 < n \le 1000$.

2.4.5. Уточнение регрессионной зависимости $k_r(n)$ на основе нечеткого регрессионного анализа

Полученные выше остатки (Дрейпер,2007) r в чётком регрессионном анализе принимаются как ошибка наблюдения, которая является случайной величиной. В нечётком регрессионном анализе (Могиленко,2004) остатки рассматриваются как обусловленные нечёткостью структуры модели. Среди рассмотренных в (Могиленко,2004) нечётких моделей была взята модель нечёткого регрессионного анализа по критерию минимальной нечёткости, комбинированного с методом наименьших квадратов (МНК), т.к. в качестве исходных данных эта модель может иметь результаты чёткого регрессионного

анализа, рассмотренного выше. А именно, в полученном полиноме все, коэффициенты могут быть рассмотрены как нечёткие коэффициенты с симметричной треугольной функцией принадлежности, причём в качестве нечётких центров выбираются значения коэффициентов из чёткой регрессионной зависимости, а значения нечётких разбросов определяются по критерию минимальной нечёткости. В результате применения критерия для рассматриваемого выше примера зависимости для поля ФИО$^{34\,657}$ получаем следующую нечёткую модель.

$$k_r(n) = (5{,}867;0) - \log_{30}((0{,}003;0)n^3 - (0{,}240;0)n^2 + (6{,}798;0)n - (57{,}561;33{,}233)) \qquad (16''')$$

Зависимость (16’’’), построенная на основе нечёткой регрессионной модели с нечёткими коэффициентами с симметричной треугольной функцией принадлежности, представляет собой альтернативу по отношению к доверительной области, ограниченной зависимостями $k_{\text{rmin}}(n)$ (14a) и $k_{\text{rmax}}(n)$ (16b).

2.4.6. Актуальность проблем построения алфавитного классификатора

Решение рассмотренных проблем построения многоуровневого алфавитного классификатора позволяет построить оптимальным образом работу с ключевым массивом. Использование интерактивного интерфейса на основе такого классификатора даёт возможность быстро находить искомый ключ массива без использования поля ввода. Для пользователя алфавитный классификатор представляет собой “путеводитель” или систему подсказок для перемещения к искомому ключу массива. Вопрос, связанный с актуальностью классификатора был более подробно освещён в разделе 1.4.

Выводы по главе

1. В главе 2 вводятся понятие ПДС и на его основе понятие классификатора.
2. Рассмотрены различные распределения ключей по буквенным префиксам. На основе модельного распределения на примере показано, что ключи распределены по префиксам *неравномерно* на всю длину ключа.

3. Построена *регрессионная зависимость* длины ключа класса от максимального числа ключей в классе.
4. Предварительно построены функции плотности соответствующих случайных величин.

Основным результатом данной главы является метод построения классификатора на основе регрессионной зависимости $k_г(n)$. Зависимость позволяет по заданному максимальному числу ключей в классе n получить среднюю длину ключа классификатора $k_г$. Ограничением метода является область определения n, на которой была построена зависимость. Зависимость $k_г(n)$ является актуальной для получения характеристик оптимального классификатора, рассматриваемого в следующей главе. Зависимость $k_г(n)$ задаёт класс классификаторов с различными параметрами $k_г$ и n, но не позволяет выбрать классификатор с оптимальным числом переходов. Этот выбор даёт разработанный автором метод, описанный в третьей главе.

Глава 3. Оптимизация функционала общего числа операций в алфавитном классификаторе

В **третьей главе** описан разработанный автором **метод построения оптимального классификатора по лексикографическому признаку**, в рамках которого рассматривается модель для оптимизации функционала общего числа операций. В отличии от метода, описанного во второй главе, данный метод позволяет построить классификатор с оптимальным числом переходов. Комбинированная структура префиксного дерева и списка ключей задаёт целый класс возможных классификаторов, среди которых необходимо выбрать оптимальный с точки зрения наиболее быстрого перехода на искомый ключ. При этом для каждого префикса определяется длина, при которой следует прерывать ветвление и переходить на линейный список, а также число уровней, на которые разбивается многоуровневый классификатор. Для этого решается задача оптимизации функционала общего числа операций $S_{\text{оп}}$. **Постановка задачи** имеет вид: $S_{on}^{*} = \min\limits_{n\in D(n), n_g \in D(n_g)} S_{on}(n, n_g)$. По оптимальному значению $S_{\text{оп}}^{*}$ определяются соответствующие аргументы – $(n^{*}, n_g^{*}) = \arg \min\limits_{n\in D(n), n_g \in D(n_g)} S_{on}(n, n_g)$ — оптимальное максимальное число ключей в классе n^{*} и оптимальное число ключей в группе n_g^{*}. Из регрессионной зависимости, описанной в главе 2, по значению n^{*} определяется оптимальная средняя длина ключа $k^{*}=k(n^{*})$ в оптимальном классификаторе.

3.1. Число операций в классификаторе при равномерном распределении ключей по префиксам

Пусть ключевой массив содержит N=810 000 строковых ключей, упорядоченных в алфавитном порядке. Кроме того, пусть все ключи равномерно распределены по начальным буквенным сочетаниям, которые являются префиксами строковых ключей. Каждый префикс определяет соответствующий класс ключей, начинающихся с этого префикса. В этом случае в ключевом массиве длина ключа равна $k_m = \log_a N = \log_{30} 810000 = 4$

буквам и ПДС состоит из 4 уровней. Каждый уровень ПДС содержит все буквы алфавита A и $a=|A|$. Каждый путь в ПДС от корня до листа задаёт соответствующий ключ в исходном массиве. Пусть также задано число ключей в классе n и в группе n_g соответственно, причём $n=a^k$, k — длина ключа $k \le k_m$, т.к. при равномерном распределении все классы ключей длины k одинакового размера с числом ключей a^k.

Тогда определим классификатор для данного ключевого массива с равномерным распределением ключей по буквенным префиксам как некоторую иерархическую структуру, построенную следующим способом. Каждый уровень классификатора состоит из всевозможных префиксов в ПДС, имеющих одинаковую длину, и, упорядоченных в лексикографическом порядке.

Определим число операций в классификаторе для ключевого массива с равномерным распределением буквенных ключей как функционал следующего вида.

$$S_{on}(n, n_g) = \sum_{h=1}^{h_m} n^{h-1} \left(\frac{m(m-1)}{2} n_g + \frac{n_g(n_g+1)}{2}(m-1) + \frac{r(r-1)}{2} \right) \qquad (19)$$

Здесь $h=1,\ldots, h_m$ — номер уровня в классификаторе; $h_m=[k_m/\Delta k]+1$ — число уровней в классификаторе (в это число также входит ключевой уровень массива, соответствующий уровню h_m.); Δk — длина n-грамм, добавляемых на каждом уровне; $n=a^{\Delta k}$ — число ключей в классе; $n_g \le n$ — число ключей в группе; $N= a^{km}$ — число ключей в массиве, k_m — длина ключа массива; $r = n \bmod n_g$ — число ключей в последней группе (при $r=0$ берётся $r=n_g$); $m = [n / n_g] + l(r)$, $l(r>0)=1$, $l(0)=0$ — число групп в классе[*)]. Коэффициент перед скобкой получен следующим образом: $N/(a^{km-(h-1)\Delta k})=n^{h-1}$. Любой префикс берётся от корня ПДС до уровня в ПДС, определяемого как $i_{ПДС} = i_K \Delta k$, где i_K — это номер уровня в классификаторе.

[*)] Символом m в главе 3 обозначено число групп ключей в классе. В главе 2 этот символ имеет другой смысл

В рассматриваемом выше примере можно посчитать число операций, например, с одним последним уровнем классификатора и второй уровень — это ключевой массив. Тогда формула (19) будет состоять из шести слагаемых без общей суммы. Если n=900 и n_g=40, то $S_{\text{опр}}$=900(10120+18040+210)+ 28370=25561370.

В формуле (19) сумма берётся по всем уровням классификатора h. Первое слагаемое — это сумма операций при проходе по всем группам ключей класса на уровне h. Второе — сумма операций при проходе по всем ключам групп класса на уровне h, кроме последней. Третье — сумма операций при проходе последней группы класса на уровне h. Множитель перед скобкой определяет число классов на уровне h. Здесь $m(h)$ — число групп по n_g ключей на уровне h и $r(h)\leq n_g$ — число ключей в последней группе класса.

3.2. Описание алфавитного классификатора на основе префиксного дерева сочетаний

Рассмотрим ключевой массив или индекс, состоящий из алфавитного списка ключей. Пусть этот индекс разбивается на *классы* по лексикографическому признаку посредством алфавитного классификатора (Емельянов,Тищенко,2010;Соловьёв,Тищенко,2018). Алфавитные ключи классификатора составляются из ключей префиксного дерева сочетаний (ПДС) для всевозможных путей в дереве. Каждый ключ — это соединение букв или буквенных сочетаний от корня до некоторого для каждого сочетания своего уровня. При этом длина алфавитного ключа класса выбирается минимальной настолько, чтобы соответствующий класс содержал не более, чем заданное количество ключей n. В каждый класс входят все ключи индекса, начинающиеся с алфавитного ключа класса. Каждый класс разбивается на *группы* по n_g ключей. Таким образом, в классе может содержаться несколько групп по n_g ключей, кроме последней группы, в которой может содержаться менее, чем n_g ключей (также и в случае одной группы).

3.3. Выбор оптимального алфавитного классификатора

Введём следующие обозначения. Пусть N — число ключей в индексе; n — максимальное число ключей в классе; n_g — число ключей в группе; k — число букв в алфавитном ключе, k=1,…,k_m, где k_m обозначает максимальную длину ключа; $n(k)$ — число ключей длины k. Предполагается, что ключи классификатора нумеруются по длинам ключей в порядке левого обхода (Емельянов,Богачёва,2001) ПДС. Пусть $n(k,i)$ — число ключей массива в классе с i-ым ключом классификатора длины k, причём $n(k,i)\le n$; $r(k,i)$ — число ключей в последней группе класса с i-ым ключом длины k: r(k,i)=m(k,i) mod n_g; $m(k,i)$ — число групп по n_g ключей с i-ым ключом длины k: $m(k,i)=[n(k,i)/n_g]+l(\{n(k,i)/n_g\})$, где квадратные и фигурные скобки — это целая и дробная часть числа соответственно, $l(0)=0$, $l(x)=1$ при $x>0$. Тогда общее число операций по поиску всех ключей массива определяется функционалом (20) (Тищенко,2018,2019).

$$S_{on}\left(n,n_g\right)=\sum_{i=1}^{k_m}\sum_{j=1}^{n(k)}\left(S_g(k,i,n_g)+S_{gk}(k,i,n_g)+S_{gr}(k,i,)\right)+S_k\left(n_g\right) \qquad (20)$$

Здесь первая сумма означает суммирование по всем длинам ключей от 1 до k_m, вторая сумма означает суммирование по всем ключам длины k от 1 до $n(k)$, где $n(k)$ — общее число ключей длины k. Первое слагаемое $S_g(k,i,n_g)$ — суммарное число операций прохода по группам ключей для класса с i-ым ключом длины k:

$$S_g(k,i,n_g)=\sum_{j=0}^{m(k,i)-1}jn_g=\frac{m(k,i)\left(m(k,i)-1\right)}{2}n_g \qquad (21)$$

Второе слагаемое $S_{gk}(k,i,n)$ — суммарное число операций прохода по ключам групп для класса с i-ым ключом длины k, кроме последней группы, в которой может быть менее чем n_g ключей:

$$S_{gk}(k,i,n_g)=\sum_{w=1}^{n_g}w\left(m(k,i)-1\right)=\frac{n_g\left(n_g+1\right)}{2}\left(m(k,i)-1\right) \qquad (22)$$

Третье слагаемые $S_{gr}(k,i)$ — суммарное число операций прохода по ключам последней группы числом $r(k,i) \le n_g$ для класса с i-ым ключом длины k:

$$S_{gr}(k,i) = \frac{r(k,i)\left(r(k,i)+1\right)}{2} \qquad (23)$$

Эти три слагаемых стоят под двойной суммой. Последнее слагаемое $S_k(n_g)$ — общее число операций для алфавитных ключей классификатора состоит из трёх слагаемых (см. формулу 24), подобных (21), (22) и (23). Однако весь уровень алфавитных ключей представляет собой один класс, разбитый на группы по n_g алфавитных ключей. Таким образом, у величин m_a числа групп по n_g алфавитных ключей и r_a числа алфавитных ключей, входящих в последнюю группу, соответствующих величинам $m(k,i)$ и $r(k,i)$, будут отсутствовать зависимость от величин k и i.

$$S_k\left(n_g\right) = \frac{m_a\left(m_a-1\right)}{2} n_g + \frac{n_g\left(n_g+1\right)}{2}\left(m_a-1\right) + \frac{r_a\left(r_a+1\right)}{2} \qquad (24)$$

Величины $m(k,i)$ и $r(k,i)$ являются случайными величинами и случайным образом зависят от максимального числа ключей в классе в силу неравномерности распределения текстовых ключей по буквенным префиксам.

В качестве оптимальных значений параметров алфавитного классификатора рассматриваются максимальное число ключей в классе n^* и число ключей в группе n_g^*, при которых достигается минимум функционала (20) $S_{on}(n^*,n_g^*)=S_{on}^*$ (Тищенко,2018). Ещё одним параметром алфавитного классификатора является среднее значение длины ключа k^*, определяемое из регрессионной зависимости $k(n^*)=k^*$ (Емельянов,Тищенко,2010). При $n=N$ длина алфавитного ключа $k=0$ и сумма (20) сводится к 3 слагаемым $S_{on}(N) = S_g(0,0,n) + S_{gk}(0,0,n) + S_{gr}(0,0)$ и $n(0,0)=N$.

3.4. Вид функционала общего числа операций в общем случае

Необходимо отметить, что формула (20) предполагает наличие одного уровня классификатора, который состоит из алфавитных ключей, ссылающихся на соответствующие классы ключей массива. При более общем рассмотрении

можно обобщить результат (20) для многоуровневого классификатора. При этом каждый более высокий уровень классификатора будет являться одноуровневым классификатором для подчинённого уровня. Формула (20) в этом случае перепишется в виде трёх слагаемых, соответствующих (21), (22) и (23), которые стоят под тремя суммами. Внешняя сумма будет соответствовать сумме по всем уровням многоуровневого классификатора, а две внутренние суммы будут соответствовать суммам в формуле (20). Также величины $m(k,i)$, $r(k,i)$ и $n(k)$ будут уже зависеть ещё от номера уровня в классификаторе h. Верхний уровень алфавитного классификатора будет состоять из одного класса. Число уровней классификатора $h_m = [\bar{k}/\Delta k]+1$. Здесь $\bar{k}$ обозначает среднюю длину ключа на последнем уровне классификатора, а Δk — среднее число букв, которое добавляется к ключу на каждом уровне классификатора. В величину h_m включается также ключевой уровень массива, поэтому добавляется 1. Таким образом, общая формула имеет вид (Тищенко,2018):

$$S_{\text{оп}}(n, n_g) = \sum_{h=1}^{h_m} \sum_{k=1}^{k_m} \sum_{i=1}^{n(h,k)} \left(S_g(h,k,i,n) + S_{gk}(h,k,i,n) + S_{gr}(h,k,i) \right) \qquad (25)$$

В рассмотренном примере применялась формула (20) для одноуровневого классификатора, т.к. ключевой массив состоит из относительно небольшого количества ключей — 34 657 и число префиксов в оптимальном алфавитном классификаторе (при n_g^*=20 и n^*=176) получается относительно небольшим порядка 1,5 тысяч. Для верхнего уровня классификатора классы ключей выделяются в виде однобуквенных префиксов, т.е. букв алфавита при n^*=176. Такой двухуровневый алфавитный классификатор над ключевым массивом ФИО$^{34\,657}$ согласуется с формулой (25). При построении пользовательского интерфейса классификатор с одним уровнем можно надстроить 2-3 уровнями алфавитных ключей и получить многоуровневый классификатор, например, первый уровень — однобуквенный классификатор, а второй 2-3-х буквенный классификатор.

Полученный качественный вид зависимости $S_{on}(n,n_g)$ с одним характерным глобальным минимумом можно считать достаточно общим. Например, в случае равномерного распределения текстовых ключей по сочетаниям при общем числе ключей 810 000 минимум S_{on}^*=24 732 450 достигается при n_g^*=35 или n_g^*=36 *и* n^*=900. С другой стороны, случайные распределения длины ключа классификатора и числа ключей в классе (Емельянов,Тищенко,2010) могут изменяться в зависимости от типа рассматриваемых полей, т.е., например, ключевой массив фамилий, ключевой массив адресов или ключевой массив должностей.

3.5. Алгоритм расчёта оптимального классификатора по лексикографическому признаку

В пункте выше предложен метод выбора оптимального алфавитного классификатора посредством минимизации функционала общего числа операций $S_{on}(n,n_g)$ в случае *неравномерного* распределения ключей по n-граммным префиксам. Параметрами классификатора являются n — максимальное число ключей на любой префикс, образующего класс ключей в классификаторе, и n_g — число ключей в каждой группе, с учётом того, что каждый класс разбивается на равные группы (в последней группе класса число ключей может быть меньше n_g). Для построения классификатора для ключевого массива строится префиксное дерево сочетаний (PATRICIA или ПДС) и каждому префиксу сопоставляется его частота встречаемости. Классификатор представляет собой префиксное дерево, сжатое по поддеревьям.

Для проведения расчётов, связанных с ПДС, более оптимальным способом следует распараллелить расчёт оптимального значения функционала общего числа операций $S_{on}^*= S_{on}(n^*,n_g^*)$ по формуле (20), для различных грамм и возможно биграмм. Из грамм или биграмм в зависимости от их частотности составляются группы с наиболее близкими средними частотами. Число групп равно числу каналов обслуживания. Для всех групп параллельно рассчитывается оптимальное значение функционала $S_{on}^*(j)$ на основании частот

в ПДС. Общее оптимальное значение функционала получается в виде суммы значений по группам:

$$S_{on}^{*} = \sum_{j=1}^{c} S_{on}^{*}(j) \qquad (26)$$

Величина *c* обозначает число каналов обслуживания. Нетрудно видеть, что для распределения грамм или биграмм по каналам обслуживания с наиболее близкими общими средними частотами необходимо решить задачу, связанную с минимаксным размещением объектов.

Существует точное решение этой задачи, но за субэкспоненциальное время $n^{O(\sqrt{P})}$ (Hwang,1993,Vol.9,no.1,no.4). Здесь P — число кластеров, по которым размещаются объекты. Приближённое решение существует в виде аппроксимации $1+\varepsilon$, например, за время $O(P^n)$ для малых P (Bādoiu,2002;Kumar,2010). Оба класса алгоритмов являются NP-трудными. Третий класс алгоритмов основан на выделении отдалённых точек (ВОТ) (Gonzalez,1985;Feder,1988). Гонзалез (Gonzalez,1985), в частности, доказал, что алгоритмы с точным решением и с аппроксимирующим коэффициентом меньшим двух являются NP-трудными. Поэтому непрактично искать решения посредством алгоритмов из указанных классов. Гонзалез предложил решение (Gonzalez,1985) методом ВОТ, который гарантирует удвоенное оптимальное решение за линейное время.

Алгоритм расчёта оптимального значения функционала S_{on}^{*} содержит следующие шаги:

1. Методом выделения отдалённых точек (или путём последовательного разделения алфавита) выделяется соответствующее число групп грамм или биграмм по числу каналов обслуживания с близкими средними частотами.
2. По каждому каналу параллельно производится расчёт функционала $S_{on}^{*}(j)$ по формуле (20).
3. По формуле (26) вычисляется общее значение S_{on}^{*} для всего алфавита.

Алгоритм расчёта оптимального классификатора был применён к ключевым массивам различных объёмов. Для параллельных расчётов использовались 4 канала обслуживания с частотой каждого ядра процессора 2,5ГГц (Core 2 Quad Q9300, DDR2 8GB, 120GB SSD). В данном случае получилось, что применение деления подряд на 4 части алфавита по наиболее близким частотам по процессам даёт более близкие средние частоты, чем применение метода выделения отдалённых точек.

Таблица 3.1

Проблема распределения буквенных сочетаний по каналам обслуживания

N	n_1 / t_1 (мс)	n_2 / t_2 (мс)	n_3 / t_3 (мс)	n_4 / t_4 (мс)
34 657	8 013	8 046	9 892	8 706
	905	920	1 107	889
103 823	22 945	25 875	27 077	27 926
	3 042	3 370	3 448	3 541
923 915	203 295	230 040	241 750	248 829
	36 942	43 181	45 677	46 223
4 216 674	931 042	1 042 443	1 100 402	1 142 787
	261 021	282 925	301 379	302 345
11 655 046	2 568 183	2 875 210	3 036 400	3 175 253
	843 701	934 134	957 721	1 007 096

В таблице 3.1 приводятся частоты n_i по четырём каналам и временные интервалы t_i (в миллисекундах) расчёта функционала для соответствующей последовательности букв s_i при различных объёмах N ключевого массива. Буквенные последовательности по каналам s_1=АБВГ, s_2=ДЕЖЗИЙК, s_3=ЛМНОПР, s_4=С–Я.

В таблице 3.2 приведены различные значения для минимума функционала $S_{on}(n^*,20)$ при различных объёмах массива N. Здесь также приводятся значения для времён создания ПДС T_c и обработки ПДС при расчёте общего числа операций T_o. Кроме того, даются значения для соответствующих размеров файлов ПДС V и оптимальных значений максимального числа ключей в классе n^*.

Таблица 3.2

Временные характеристики ПДС и оптимальные значения функционала при n_g=20

N	T_c, с	T_o, с	V, МБ	n^*	$S^*(n^*,20)$
34 657	5,4<1м	3,4<1м	9,6	176	505 080
103 823	18,0<1м	6,4<1м	23,5	333	1 864 063
923 915	211,1<4м	68,9<2м	215,9	619	28 775 243
4 216 674	1 155,1<20м	513,9<9м	954,1	1 078	188 637 291
11 655 046	8 187,1<137м	1 021,2<18м	2 892,8	1 259	680 654 436

На примере поля ФИО$^{34\,657}$ (данные взяты из базы данных “За Христа пострадавшие”[*)]) рассмотрим процесс нахождения оптимальных параметров алфавитного классификатора k^*, n^* и n_g^* в указанном выше смысле. Процесс нахождения минимального значения функционала (20) $S_{on}(n,n_g)$ сводится к полному перебору значений максимального числа ключей в классе n, например, от 1 до 1000. Предполагается, что число ключей в группе n_g также меняется в некотором диапазоне, например, $10 \le n_g \le 100$.

На рис.3.1 показаны графики зависимости максимального числа ключей в классе при минимальном значении S_{on} от числа ключей в группе $n^*(n_g)$ и суммарного числа операций, минимального по величине относительно n^*, от числа ключей в группе $S_{on}(n^*,n_g)$. Как видно из графика $S_{on}(n^*,n_g)$ минимум (при целых значениях n_g на диапазоне от 10 до 100 с шагом 10) достигается при n_g^*=20 и соответствующего значения n^*=176 (или n^*=177) $S_{on}^*=S_{on}(n^*,n_g^*)$=505 080. Фактически минимум S_{on}^* достигается при n_g=15. При этом величина $n^*(n_g)$ имеет сильные колебания.

В таблице 3.3 приводятся данные для n, κ, $S_{on}(n,n_g^*)$, где n_g^*=20, и на рис.3.2 показана соответствующая схематическая сглаженная зависимость $S_{on}(n,20)$. Как видно из таблицы 3.3 (строки с n=100,200,300) минимум S_{on} находится между значениями аргумента $100 \le n \le 300$ и соответствующих $2{,}678 \le \kappa \le 3{,}690$.

[*)] БД “За Христа пострадавшие”, http://martyrs.pstbi.ru

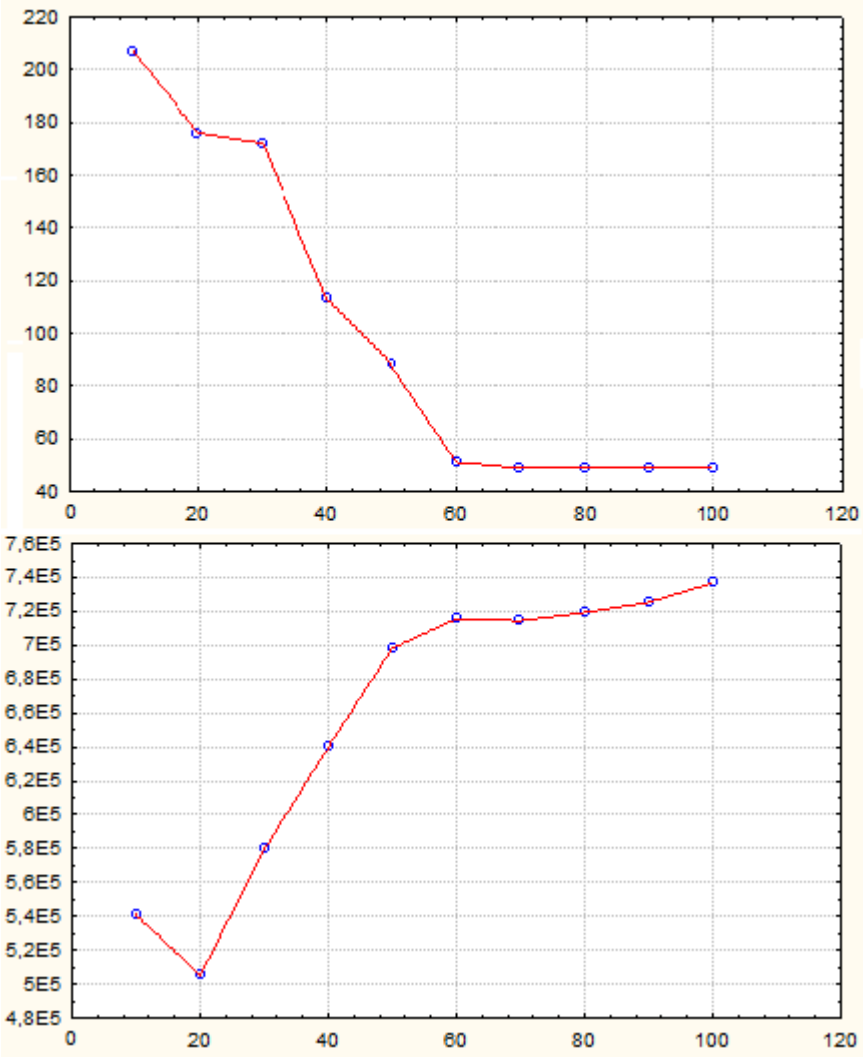


Рис.3.1. Графики $n^*(n_g)$ и $S_{on}(n^*,n_g)$

Таблица 3.3

Значение суммарного числа операций в зависимости от максимального числа ключей в классе

n	κ	$S_{on}(n,n_g^*)$
1	10,969	71 375 904
10	6,703	6 106 107
20	5,717	2 470 552
30	5,142	1 447 754
40	4,799	1 079 359
50	4,490	836 647
60	4,244	724 682
70	4,082	656 709
80	3,959	620 178
90	3,817	582 267
100	3,690	572 874
200	3,019	518 825

300	2,678	575 821
400	2,366	559 175
500	2,272	578 167
600	2,147	797 290
700	2,071	823 171
800	1,966	879 474
900	1,916	909 226
1000	1,916	909 226

В таблице 3.4 представлены длины префиксов при равномерном распределении ключей по префиксам K_p, средние длины префиксов при неравномерном распределении ключей по префиксам $K_н(1)$, соответствующее среднее число уровней для каждого объёма массива $K_o(n^*)$, средние длины префиксов $K_н(n^*)$ для оптимального значения максимального числа ключей в классе n^* и соответствующее среднее число ключей в классе $n_{ср}$. Таблица 3.4 составлена на основе соответствующих экспериментальных данных, на основе которых строятся регрессионные зависимости средней длины префикса от максимального числа ключей в классе $K_н(n^*)$. Здесь n_g — число ключей в группе бралось равным 20.

Таблица 3.4

Среднее число уровней и средние длины ключей оптимальных классификаторов

N	K_p	$K_н(1)$	$K_o(n^*)$	$K_н(n^*)$	n^*	$n_{ср}$
34 657	1,553	10,969	3,068	3,106	176	74,158
103 823	1,688	10,803	3,268	3,294	333	47,402
923 915	2,149	13,789	4,789	4,866	619	272,233
4 216 674	2,432	15,693	5,591	5,683	1 078	465,787
11 655 046	2,685	17,244	6,320	6,458	1 259	526,695

Таким образом, оптимальный классификатор в данном случае будет содержать максимальное число ключей в классе, равное 176, и средняя длина ключа класса в этом случае равна 3,106. На рис. 3.2 качественно показана зависимость $S_{on}(n,20)$ при $n_g=n_g^*=20$, т.е. для значения n_g, когда достигается оптимальное число операций $S_{оп}^*$. Минимум при $n^*=176$ условно показан при

n=200. Шаг делений по оси абсцисс также неодинаковый. Кроме того, необходимо отметить, что действительный минимум S_{on}^{*} достигается при n_g^{*}=15 и n^{*}=178 (или n^{*}=178) и равен S_{on}^{*}=487 117.

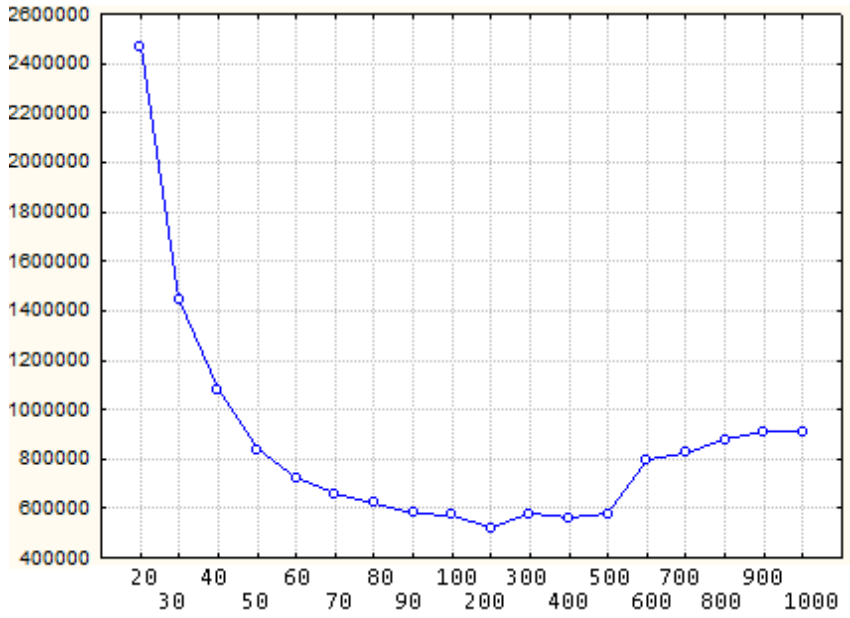


Рис. 3.2. Схематичная форма графика $S_{on}(n,20)$

Характер зависимости $S_{on}(n,n_g^{*})$ при различных N такой же как показано на рисунке 3.2, где N=34 657 — общее число ключей в массиве. Вид приведённой зависимости является типичным в смысле минимума функционала $S_{on}(n,n_g^{*})$.

Поиск оптимального значения функционала общего числа операций в алфавитном классификаторе S_{on}^{*} организован в виде полного перебора значений n и n_g на заданных диапазонах.

В результате по оптимальному значению максимального числа ключей в классе n^{*}, определённого по минимальному значению функционала общего числа операций $S_{on}(n,n_g)$, из регрессионной зависимости определяется оптимальная длина префикса классификатора $k(n^{*})$. Таким образом, оптимальный классификатор для индекса по фамилиям объёмом 34 657 в БД по репрессированным будет содержать максимальное число ключей в классе n^{*}=176 при числе ключей в группе n_g^{*}=20 и средней длине ключа класса k^{*}=3,106. Для других объёмов массивов N значения S_{on}^{*} (см. формулу (20)), n^{*} и $k^{*}=k(n^{*})=K_н(n^{*})$ приведены в таблицах 3.2 и 3.4. В результате применения метода к ключевому массиву строится соответствующий оптимальный

классификатор. Минимизация функционала общего числа операций S_{on} (см. формулу (20) или (25)) позволяет определить параметры оптимального классификатора n^* и n_g^*, по которому из регрессионной зависимости $k(n)$ определяется третий параметр k.

Выводы по главе

1. В главе 3 автор предлагает метод построения оптимального классификатора, характеристики которого определяются путём минимизации функционала общего числа операций. Эта глава содержит основную теорию для построения классификатора в качестве интерфейса для интерактивного доступа к ключевому массиву, что составляет *цель* диссертационной работы.
2. Оптимизация функционала даёт *параметры* оптимального классификатора — число ключей в классе и число ключей в группе. Из регрессионной зависимости (см. главу 2) по числу ключей в классе определяется средняя длина ключа оптимального классификатора.
3. Теоретический результат получен для расчёта функционала по всем уровням классификатора — формула (25). Однако на практике вполне достаточно пользоваться более простым видом функционала при объёме массива до одного миллиона ключей — формула (20).
4. Алгоритм расчёта функционала использует параллельные вычисления для увеличения скорости вычислений. При размещении букв по каналам обслуживания могут применяться *минимаксные алгоритмы* размещения объектов для уменьшения разницы средних частот букв по каналам обслуживания.

Следующие главы посвящены *практической реализации* оптимального классификатора с целью проверки разработанных теоретических положений и с использованием изложенных теоретических методов на основе гипертекстовой системы ООСУБД НИКА.

Глава 4. Теория, методы и средства построения гипертекстовой системы на основе СУБД НИКА

Теория построения классификатора по лексикографическому признаку изложена во второй и третьей главах. В четвёртой и пятой главе излагается практическая реализация классификатора и приводятся примеры его построения.

Естественной реализацией иерархического классификатора может служить объектно-ориентированная база данных НИКА.

“НИКА позволяет работать со сколь угодно сложными иерархическими объектами, имеющими произвольные сетевые связи. Модель данных СУБД НИКА является типово-полной, т.е. любая суперпозиция допустимых типов данных является допустимым типом. Отсутствие ограничений на структуру объектов в БД существенно облегчает процесс отображения объектов реального мира в БД: нет необходимости разбивать их на части для того, чтобы поместить в различные отношения, как это, например, требуется в реляционных СУБД … Манипулирование данными в системе НИКА трактуется шире, чем во многих других СУБД. Можно манипулировать не только данными, хранящимися в базе, но и данными, содержащимися в схеме описания данных. Это даёт возможность в случае необходимости динамически программным путём изменять схему БД, а также создавать программы, обрабатывающие БД произвольной структуры” (Годунов,1991,С.208).

НИКА в полной мере обладает необходимыми возможностями, связанными с основными понятиями ООБД, такими как: система поддержки типов, классы и объекты, идентификация объектов, наследование. Классы инкапсулируют функции или методы, в частности методы отображения объектов данного класса. В СУБД НИКА объекты — это нетерминальные вершины, а атрибуты объектов данного класса — это терминальные вершины. В качестве идентификатора объекта берётся координата данной вершины от корня БД. На основе СУБД НИКА строится гипертекстовая система, модель

которой близка к модели БД НИКА. В связи с этим, первый пункт данной главы содержит формальное описание модели СУБД НИКА, в качестве которого взято определение объектно-ориентированной базы данных у Абитебула (Abitebool,1988), и модели гипертекста в виде сетей Петри (Halasz,1994). Второй пункт — это формальное описание предметной области (ПО) в виде схемы базы данных и определение классификатора по лексикографическому признаку в виде схемы БД НИКА. Третий пункт содержит описание ядра гипертекстовой системы, представляющей собой спецификации или методы отображения вершин БД в виде гипертекстовых документов, соответствующих видам представления данных по Емельянову. Четвёртый — это описание надстройки над ядром с использованием расширяемого языка стилей XSL (Тищенко,2003) для уточнения отображения вершин. Описаны возможности отображения вершин в виде документов в форматах HTML, XML и PDF. В пункте приводятся примеры построения различных классификаторов.

Основные публикации по гипертекстовой системе для ООСУБД НИКА — (Емельянов,1987; Емельянов,Тищенко,1995,1996,1997,1999,2009,2010; Тищенко,2003,2013,2018,2019; Тищенко и др.,2019). Такая система соответствует направлению электронной публикации баз данных. Параллельным направлением исследований является публикация баз данных на бумажном носителе в виде справочников и книг с использованием издательских систем. Это направление развивалось отдельно (Емельянов,Соловьёв,1995,2000; Соловьёв,1998,2000).

4.1. Реализация гипертекстовой системы на основе СУБД НИКА

4.1.1. Принципы построения гипертекстовой системы на основе ООБД

4.1.1.1. Формальное описание модели СУБД НИКА

Для изучения проблем реструктуризации объектно-ориентированных баз данных (ООБД) S.Abitebool и R.Hull разработали формализм описания ООБД

(Abitebool,1988). Согласно этому формализму ООБД — граф — совокупность вершин и дуг, каждая вершина ООБД имеет один из трех типов: *, #, @. В (Abitebool,1988) следующим образом вводится понятие БД. К БД добавляется вершина “корень БД“, а также все дуги, связывающие эту вершину с корнями всех деревьев, описанных в БД. После этого преобразования любая БД без ссылок есть дерево. Введем понятие типа БД.

Определение: Тип объекта БД — это граф (V,E(V)), где V — множество вершин и E(V)$\subset V$xV – множество дуг, удовлетворяющих следующим условиям.

a) V делится на следующие подмножества:

V^* — терминальные вершины (атрибуты или *-вершины);

$V^{\#}$ — ассоциативные вершины (массивы или #-вершины);

$V^{@}$ — агрегативные вершины (структуры или @-вершины), кроме того, $V^{@}$ содержит корень БД V_0.

b) Для любого $v\in V$: $v\neq V_0$ существует и единственна дуга (v',v), входящая в v. Будем говорить. что v подчиняется v'. Для V_0 не существует входящих дуг.

c) Если $v\in V^*$, то множество E(v)={v':(v,v')$\in$E(V)} вершин, подчиненных v является пустым.

d) Если $v\in V^{\#}$, то существует и единственна дуга (v',v), выходящая из v: $v'\in V^{@}$. В этом случае v' называется элементом массива v и обозначается el_v. Множество E(el_v) должно содержать выделенный элемент, который является терминальной вершиной; этот элемент обозначается key_v.

e) Если $v\in V^{@}$, то существует по крайней мере одна дуга, выходящая из v.

f) Для любого $v\in V^*$ область d(v) его значений определяется как область значений некоторого базисного типа.

g) Для любого $v\in V$ имя вершины &v является допустимым именем в определенном алфавите. Кроме того, если v' и v'' подчинены одной и той же вершине v, то &v' отлично от &v''.

Замечания:

1) множества V^*,$V^{\#}$,$V^{@}$ не пересекаются;

2) пункт b) учитывает, что граф V является деревом с корнем V_0;

3) для любой вершины из V можно однозначно определить ее тип в смысле (Abitebool,1988), следующим образом. Если $v \in V^*$, то ее тип $T(v) = \&v: d(v)$. Далее, используя это определение можно рекурсивно определить типы всех $v \in V^{@}$ посредством $T(v) = \&v: \{T(v_1), ... ,T(v_n)\}$, где $v_1,...,v_n$ образуют $E(v)$. Для любого $v \in V^{\#}$ будем считать, что $T(v) = \&v: [T(el_v)]$. Типом БД определим тип ее корня $T(V_0)$.

Таким образом мы имеем простые базовые типы данных и конструкторы (массив и структура) новых типов на основе простых и прежде описанных типов. Тип вершины (объекта) определяется как совокупность всех подчиненных. Нет ограничений на сложность конструкций (суперпозицию типов). Следовательно эта модель является типово полной моделью данных.

Кроме вышеописанных типов вводятся вершины типа ссылки V^p и V^r.

V^p— ссылка на шаблон — повторное использование типа или создание нового типа той же структуры, но с другим именем.

V^r— ссылка на значение. Также повторное использование типа с другим именем, но с теми же значениями. Этот тип позволяет построение графа произвольной сложности, в частности, сети и графа с циклами.

Основные отличия объектно-ориентированного подхода это повышение семантической и функциональной выразительности (Замулин,1990). Семантическая выразительность обеспечивается введением сложных структур данных, а функциональная выразительность — интеграцией процедурных и информационных аспектов. Каждая вершина графа — объект может содержать методы его обработки (свойство инкапсуляции ООСУБД), в частности, метод его отображения на экране. В ООСУБД НИКА методы обработки хранятся в схеме БД.

4.1.1.2. Идентификация текущей точки

Рассмотрим проблему идентификации текущей точки в процессе движения по базе данных (Емельянов,Тищенко,1997). БД, построенная по

предложенной модели данных, является лесом — совокупностью выделенных деревьев, перевязанных ссылками. Выделенными являются те деревья-иерархии, которые образованы вложенными конструкторами типов — массивами и структурами. Следовательно, есть несколько выделенных объектов верхнего уровня, которые не входят ни в какие объекты более высокого уровня, и которые являются корнями выделенных деревьев. К любому объекту или его реквизиту (вершине БД) ведет от корня до него одна цепочка объектов выделенной иерархии, не проходящая через ссылочные типы. При просмотре ООБД бывает интересно следить за полными траекториями "блуждания" с учетом движения по ссылкам между иерархиями, в этом случае в траектории могут возникнуть циклы.

4.1.1.3. Просмотр объектов БД

Начинается просмотр БД показом списка всех корневых вершин, оформленных в соответствии с описанными ниже спецификациями как гипертекстовые ссылки. Выбрав нужный корень, можно перейти в соответствующий объект верхнего уровня. Текущая точка в БД отображается в виде полной траектории — цепочки имен, ведущей от одного из корней БД к данному месту в БД (включая и проходы по ссылкам). При этом показываются имена объектов (из схемы БД), построенных при помощи конструкторов массива, структуры и ссылки, а также имена ключей элементов массивов — идентификаторов конкретных объектов, через которые прошла траектория в процессе движения по БД.

Если в БД только одна выделенная иерархия, то гипертекстовая система представляет из себя (без спецификаций вершин схемы БД) часто применяемую (например, в справочных системах) линейно-иерархическую структуру, построенную на парадигме "оглавления книги" (Еременко,1996). Оглавление содержит линейную последовательность глав. Иерархия оглавления обеспечивает минимизацию пути к искомому разделу. Каждая страница гипертекста (текущий раздел) содержит ссылки:

– на раздел (разделы) более высокого уровня иерархии;

– на предыдущий раздел (или группу разделов) текущего уровня;

– на следующий раздел (или группу) текущего уровня;

– на первый раздел (или группу) подчиненного уровня.

При этом переход на раздел более высокого уровня осуществляется выбором одного из объектов траектории, оформленных как гипертекстовые ссылки. Переход на следующий (предыдущий) объект текущего уровня - выбором нужного объекта из списка объектов и входом в него по гипертекстовой ссылке с отображением на экране (с учетом спецификаций) объектов подчиненного уровня.

В общем случае, когда есть несколько корней БД и есть ссылки, отображение строится по метафоре "освещенного лабиринта" — показываются все возможные варианты движения на подчиненные структурные объекты из текущей точки, и значения всех терминальных вершин.

4.1.2. Модель сетей Петри

Сети Петри относятся к числу наиболее важных и распространённых математических моделей в области обработки информации. Они позволяют формально описывать вычислительные системы и порождённые ими процессы. С использованием динамической структуры строится модель гипертекста на основе сетей Петри. Указанная модель подробно рассматривается в (Scotts,1989). В этой модели компонента сети Петри расширяется дополнительными компонентами, связанными с конкретной реализацией гипертекста. Рассмотрим гипертекст в виде сети Петри (Кокин,2005):

$$H = <P, T, I, O, F, M, \delta> \qquad (27)$$

Здесь P обозначает конечное множество позиций или мест сети ($P \neq \varnothing$); T — конечное множество переходов ($T \neq \varnothing$); F — отношение инцидентности или множество дуг сети, $F \subset (P \times T) \cup (T \times P)$. Для сети $<P,T,F>$, представляющей собой двудольный граф с двумя типами вершин P и T, выполнены следующие условия:

1) $P \cap T = \varnothing$ — множества позиций и переходов не пересекаются;
2) $F \neq \varnothing$ и любой элемент сети инцидентен хотя бы одному элементу другого типа;
3) сеть не содержит пары позиций, которые инцидентны одному и тому же множеству переходов.

$I(t,p)$ — кратность дуги из позиции в переход; $O(t,p)$ — кратность дуги из перехода в позицию. $M:P \rightarrow N$ — разметка сети, т.е. функция, которая ставит в соответствие каждой позиции целое неотрицательное число $M(p)$. $\delta(M,t)=M'$ — функция изменения состояния, вызванного срабатыванием перехода t (в случае, если t разрешён при данной разметки M), M, $M' \in R(H)$, $R(H)$ — множество всех разметок сети H, достижимых от начальной разметки.

4.1.3. Интерпретация гипертекстового документного интерфейса к БД НИКА, в виде модели сетей Петри

Описанная модель объектно-ориентированной СУБД НИКА естественным образом вписывается в гипертекстовую систему. При отображении БД в гипертекстовые документы (например, html, xml) объект (структура, массив или ссылка) будет представлен в виде имени объекта, являющегося гипертекстовой ссылкой на атрибуты объекта и непосредственно подчиненные ему объекты в БД; атрибут (терминальная вершина) — в виде обычного текста: <имя_атрибута> = <значение_атрибута>. В соответствии с этим представлением объект базы данных будет представлен через гипертекстовую систему отдельным гипертекстовым документом, содержащим все атрибуты этого объекта и подчиненные объекты. Следовательно, структура базы данных НИКА через гипертекстовый интерфейс может быть описана соответствующим графом. Вершины графа - это гипертекстовые документы, что равносильно объектам БД, дуги графа - это гипертекстовые ссылки в документе на другие гипертекстовые документы (объекты), что соответствует дугам, выходящим из данного объекта. Таким образом, иерархия объектов в БД равносильна иерархии гипертекстовых документов. Гипертекстовая ссылка

содержит запрос к БД, в котором просто указывается путь в БД (цепочка вышестоящих объектов от корня БД) к запрашиваемому объекту.

Кроме самой структуры гипертекста сеть Петри (27) отображает работу пользователей с гипертекстовой системой посредством разметки M и функции изменения состояния δ. Пользователь выбирает в документе определенную вершину БД, представляющую собой гипертекстовую ссылку. Это действие представляется в сети Петри для данного гипертекста срабатыванием соответствующего перехода t и новой разметкой $M'=\delta(M,t)$. В рассматриваемом случае переход t представляет координату в БД, реализующую выбранную гипертекстовую ссылку. При этом метка перемещается из позиции p в позицию p', соответствующих исходному документу и документу, в который был осуществлен переход по ссылке, и, содержащих фрагменты БД. Текущая разметка M определяет возможные срабатывания переходов. Кроме описанных переходов сети Петри гипертекста, можно определить множество T_u дополнительных переходов, которые вместе с уже существующими переходами между позициями-вершинами связывают все позиции друг с другом. Эти переходы соответствуют заданию ссылки в адресной строке обозревателя, а также обновлению текущего документа . Все множество переходов связывает все позиции-вершины БД друг с другом и неявно задает ссылки с любого на любой уровень БД, которые возможно определить средствами и самой БД НИКА.

4.1.4. Двойственность структуры БД и структуры гипертекстовых документов

Важно отметить, что описываемое применение СУБД НИКА как гипертекстовой системы является правомерным в виду двойственности структуры БД описания гипертекстового документа в виде иерархии html/xml-элементов и структуры исходных html/xml-документов. В (Bogacheva,Emelyanov,2001) рассматривается проблема двойственности, подобная этой.

Обозначим в сети Петри множество отображенных вершин базы данных

и множество исходных иерархических и ссылочных связей между ними как V_{rd} и E_{rd} соответственно (Емельянов,Тищенко,2009). Для описания работы гипертекстовой системы необходимо также добавить позицию $p_н$ и дополнительные переходы $t_{нi}$, $t_{кi}$. Каждый переход $t_{нi}$ связан с соответствующей позицией из множества V_{rd}. Позиция $p_н$ есть некоторая начальная позиция, связанная со всеми переходами $t_{нi}$, $i=1,\ldots,|V_{rd}|$. Переходы $t_{кi}$ обозначают окончание работы с гипертекстовым интерфейсом СУБД НИКА и с каждым $t_{кi}$ связана соответствующая позиция-вершина из множества V_{rd}. Полученный из БД html/xml-документ может содержать также текущий путь (или координату) в БД, который определяет ссылки на вершины БД, начиная от корневой, которым подчинена данная вершина. Этим ссылкам соответствует подмножество переходов $T_{pi}\subset T_u$, $i=1,\ldots,|V_{rd}|$, которые связывают текущую позицию-вершину с позициями, соответствующими вершинам, которым подчинена данная. Формально описанные позиции и переходы можно определить следующим образом. $P=V_{rd}\cup P_н$, где $P_{н=}\{p_н\}$, $T=E_{rd}\cup T_u\cup T_н\cup T_к$, $T_н=(\cup\{t_{нi}\})$ $(i=1,\ldots,|V_{rd}|)$, где $\dot{p}_н=p_н\dot{}=T_н$, $M_0(p_н)=1$, $M_0(V_{rd})=\mathbf{0}$, $\mathbf{0}$ – нулевой вектор размерности $|V_{rd}|$; $\forall t_{uij}\in T_u$, $\dot{t}_{uij}=\{p_i\}$, $t_u\dot{}=\{p_j\}$, причем $p_i\in V_{rd}$, $p_j\in V_{rd}\backslash \mathrm{E}(v_{i1\ldots ik})$, где $i_1,\ldots,i_{1k}$ индексы вершин на соответствующих уровнях БД, начиная от корневой вершины, а $E(v)$ обозначает множество вершин БД, непосредственно подчиненных v; $\dot{t}_{нi}=\{p_н\}$, $t_{нi}\dot{}=\{p_н,p_i\}$, $\dot{t}_{кi}=\{p_i\}$, $t_{кi}\dot{}=\varnothing$, $p_i\in V_{rd}$, $i=1,\ldots,|V_{rd}|$. В результате получается сеть Петри, все переходы которой обладают свойством: при любой достижимой в сети разметки M всегда существует такая достижимая от M разметка M', что заданный переход может сработать. Это свойство называется свойством живости (Котов,1984) любого перехода в сети, а значит и всей сети. В соответствии с теоремой об эквивалентности проблемы живости и проблемы достижимости в ней произвольной разметки получаем, что не любая разметка в построенной сети Петри достижима, т.е. не достижимы все разметки из множества D_t всех t-тупиковых разметок, для которых $M_t \le M_{tm}$, $M_t\in D_t$, $M_{tm}\in D_{tm}$, где D_{tm} – конечное множество максимальных элементов множества $\mathrm{D_t}$.

В соответствии с теоремой о сводимости проблемы достижимости произвольной разметки к проблеме достижимости нулевой разметки (Котов,1984) получаем, что разметка **0**=(0,0,…,0) размерности |*P*| не достижима. Это следует также из того, что позиция $p_н$ всегда содержит одну метку, т.к. начальная разметка $M_0(p_н)=1$ и множество входных и выходных переходов для позиции $p_н$ совпадают $\dot{p}_н=p_н\dot{}=T_н$. Позиция $p_н$ моделирует первоначальные обращения к БД по заданной координате, при этом метка из позиции $p_н$ перемещается в позицию p_i, соответствующей заданной координате в БД, а также в соответствии с определением позиции $p_н$ помещается другая метка в позицию $p_н$: $M_0[t_{нi}>M$, $M_0(p_н)=1$, $M_0(p_i)=0$, $M(p_н)=1$, $M(p_i)=1$, где символ [*t*> обозначает отношение непосредственного следования разметок при срабатывании перехода *t*. Таким способом может быть описано любое обращение к БД по заданной координате $M_1[t>M_2$, $M_1(p_i)\geq 1$, $M_1(p_j)\geq 0$, $M_2(p_i)=M_1(p_i)-1$, $M_2(p_j)=M_1(p_j)+1$, $p_i,p_j\in V$, $t\in E_{rd}$: $p_i\in\dot{t}$, $p_j\in t\dot{}$. Завершение работы с БД моделируется посредством срабатываний переходов $t_{кi}$. Срабатывание перехода $t_{кi}$ приводит к удалению метки из соответствующей позиции p_i, если метка находится в позиции-вершине более заданного промежутка времени $\Delta t_к$: $M_1[t_{кi}>M_2$, $M_1(p_i)\geq 1$, $M_2(p_i)=M_1(p_i)-1$, $p\neq p_н$. Необходимо отметить, что координата в БД в терминах сетей Петри определяет некоторую последовательность срабатывания переходов, связывающих вершины, входящие в заданную координату в БД. Такие последовательности образуют подмножество свободного языка сети *L*(*H*). Обобщая описание сети Петри для БД НИКА, важно отметить, что схема БД НИКА (Годунов,Емельянов,1991) определяет класс эквивалентных сетей Петри, тип которых задан посредством схемы БД в смысле (Abitebool,1988).

4.2. Формальное описание предметной области в виде схемы БД

Модель данных в ООСУБД НИКА представляет собой сеть с выделенными иерархиями. Подобное представление данных является наиболее естественным при представлении сложно структурированной информации.

Разработка концептуальной схемы базы данных происходит в виде следующих этапов.

1. Структурирование предметной области, выделение основных понятий и их взаимосвязей.
2. Построение схемы базы данных в виде графа на основе выделенных понятий (вершины графа представляют собой понятия, дуги - взаимосвязи). Результирующий граф будет представлять собой иерархию понятий данной предметной области.
3. Создание схемы базы данных в среде объектно-ориентированной системы управления базы данных (ООСУБД) НИКА.

На рисунках 4.1 и 4.2 показаны условные обозначения вершин схемы БД в соответствии с определением типа БД в п.4.1.1.1 и приводится пример фрагмента произвольной схемы описания БД.

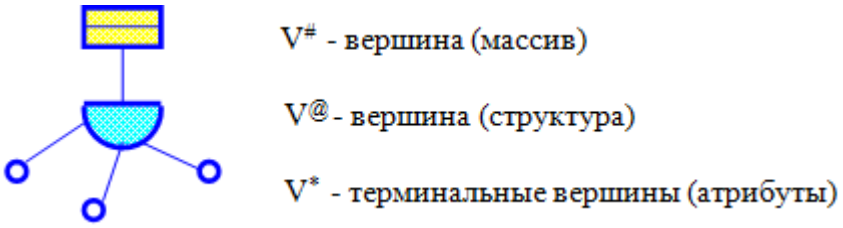


Рис.4.1. Графическое изображение вершин разных типов

Реальный пример описания данных показан на рисунке 4.2 (Емельянов,Тищенко,2009). Гипертекст, построенный с помощью языка HTML (XML), представляет собой граф, вершинами которого являются html/xml-документы (или выделенные части html/xml-документа). Каждый html-документ состоит из иерархии html-элементов. Такой гипертекст может быть отображён в базу данных НИКА с помощью соответствующей схемы данных. Эта схема может быть определена следующим образом в соответствии с определением типа базы данных в п.4.1.1.1. Вершина v1 представляет собой вершину верхнего уровня БД — массив документов, т.е. $v_1 \in V^{\#}$, $(V_0, v_1) \in E(V)$, где V_0 – корневая вершина и $E(V)$ — множество дуг графа, соответствующего типу объекта БД (см.рис.4.1). Элемент массива el_{v1} имеет множество подчиненных

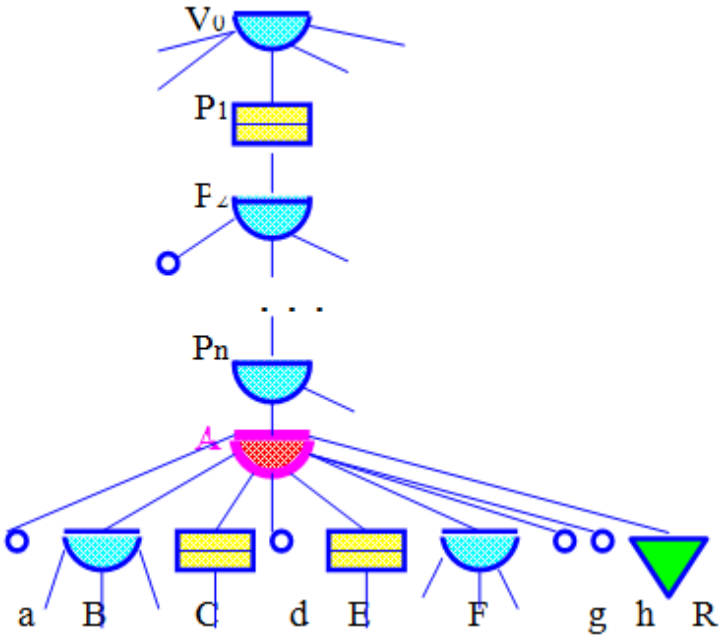


Рис.4.2. Пример фрагмента схемы описания данных

элементов $E(el_{v1})$, представляющего из себя структуру типа $T(elv1)=\{T(key_{v1}),T(v_2),T(v_3),T(v_4)\}$, состоящую из последовательности типов подчиненных вершин, где key_{v1}: $key_{v1} \in V^*$ — ключ элемента массива v_1, являющегося номером или идентификатором документа, v_2: $v_2 \in V^{\#}$ - массив html-элементов, v_3: $v_3 \in V^*$ — значение ключа корневого html-элемента или элемента с меткой html в массиве v_2, v_4: $v_4 \in V^r$, $v_4=ref_{v2}(v_3)$ — ссылочная вершина на значение, осуществляющая переход на корневой html-элемент в массиве v_2 по ключу v_3. Элемент массива el_{v2} имеет множество подчиненных элементов $E(el_{v2})$ и описывается типом $T(el_{v2}) = \{ T(key_{v2}), T(v_5), T(v_6), T(v_7), T(v_8), T(v_9), T(v_{10}), T(v_{11}) \}$, где key_{v2}: $key_{v2} \in V^*$ — ключ элемента массива $v2$, являющегося идентификатором html-элемента (уникальным для данного документа), v_5:$v_5 \in V^{@}$ — структура, содержащая значения метки и атрибутов для данного html-элемента, v_6: $v_6 \in V^*$ — текстовое содержимое для элементов, содержащих текст, v_7:$v_7 \in V^{\#}$ — массив строк — продолжение текста, определяемого вершиной v_6 (свыше 255 символов), v_8:$v_8 \in V^*$ — идентификатор первого подчиненного html-элемента данному элементу в массиве html-элементов v_2,v_9:$v_9 \in V^r$, $v_9=ref_{v2}(v_8)$ — ссылочная вершина на значение, осуществляющая

переход на заданный html-элемент в массиве v_2 по ключу v_8, v_{10}:$v_{10}\in V^*$ — идентификатор следующего html-элемента на данном уровне иерархии в массиве html-элементов v_2,v_{11}:$v_{11}\in V^*$, $v_{11}=ref_{v2}(v_{10})$ — ссылочная вершина на значение, осуществляющая переход на заданный html-элемент в массиве v_2 по ключу v_{10}. Окончательно получаем следующую структуру:

$$
\begin{aligned}
T(V_0) = \{\ \ [\ T(key_{v1}), \qquad & (28)\\
&\{\ [\ T(key_{v2}),\\
&\quad \{\ T(v_{12}),\ \{\ [\ T(key_{v13}),\ T(v_{14}),\ T(v_{15})\ ]\ \}\ \},\\
&\quad T(v_6),\\
&\quad \{\ [\ T(key_{v7}),\ T(v_{17})\ ]\ \},\\
&\quad T(v_8),\ T(ref_{v2}(v_8)),\ T(v_{10}),\ T(ref_{v2}(v_{10}))\ ]\ \},\\
&T(v_3),\\
&T(ref_{v2}(v_3))\ ]\ \},
\end{aligned}
$$

где квадратные скобки обозначают элемент массива, т.е. $T(v_i)$ =[$T(el_{vi})$], где $v_i\in V^{\#}$. Здесь $T(v_5)$ = { $T(v_{12})$, { [$T(key_{v13})$, $T(v_{14})$, $T(v_{15})$] }}, где v_{12}: $v_{12}\in V^*$ – метка данного html-элемента, key_{v13}:$key_{v13}\in V^*$ — ключ элемента нумерованного массива *v13* атрибутов и значений атрибутов html-элемента, v_{14}:$v_{14}\in V^*$ — атрибут html-элемента, v_{15}:$v_{15}\in V^*$ - значение атрибута html-элемента, $T(v_7)$ = { [$T(key_{v7})$, $T(v_{17})$] }, где key_{v7}: $key_{v7}\in V^*$ - ключ элемента нумерованного массива строк v_7, v_{17} – текстовая строка массива. Для реализации гипертекстовых ссылок средствами СУБД НИКА (а не только с помощью гипертекстового элемента "a" с заданным атрибутом "href" со значением ссылки в виде URL) полученную структуру (28) необходимо в элементе массива html-элементов el_{v2} дополнить ссылкой на значение элемента массива документов el_{v1}:

$$
\begin{aligned}
T(V_0) = \{\ \ [\ T(key_{v1}), \qquad & (29)\\
&\{\ [\ T(key_{v2}),\\
&\quad \{\ T(v_{12}),\ \{\ [\ T(key_{v13}),\ T(v_{14}),\ T(v_{15})\ ]\ \}\ \},\\
&\quad T(v_6),
\end{aligned}
$$

$\{ [T(key_{v7}), T(v_{16})] \},$

$T(v_8), T(ref_{v2}(v_8)), T(v_{10}), T(ref_{v2}(v_{10})),$

$T(v_{17}), T(ref_{v1}(v_{17})), T(v_{19})] \},$

$T(v_3),$

$T(ref_{v2}(v_3))] \},$

где v_{17}: $v_{17} \in V^*$ - номер или идентификатор html-документа в массиве v_1, v_{18}:$v_{18} \in V^*$, $v_{18}=\mathrm{ref}_{v1}(v_{17})$ - ссылочная вершина на значение, осуществляющая переход на заданный html-документ в массиве v_1 по ключу v_{17}, v_{19}:$v_{19} \in V^*$ - тип гипертекстовой ссылки $ref_{v1}(v_{17})$, задаваемый идентификатором html-элемента в документе с номером v_{17} (если не задан, то в качестве типа берется идентификатор корневого элемента). Ссылочная вершина v_{18} должна быть определена для элементов с меткой "a" и атрибутом "href" посредством соответствующего номера или идентификатора документа, заданного вершиной v_{17} (см.рис.4.3). Для получения массивов, соответствующих меткам html-элементов можно использовать индекс базы данных НИКА (Emelyanov,1994), определив вершину v_{12} как индексируемую. Средствами запроса СУБД НИКА можно также отобрать в отдельную ветку под вершиной INDEX все элементы, имеющие заданную метку и заданный атрибут (атрибуты). Если для атрибута задать значение, то можно сузить класс отобранных html-элементов. Таким образом мы получаем гибкий и простой в использовании механизм работы с элементами на заданном множестве html-документов. Средствами индекса СУБД НИКА можно под вершиной INDEX получить массив tags названий html-элементов, упорядоченных по алфавиту.

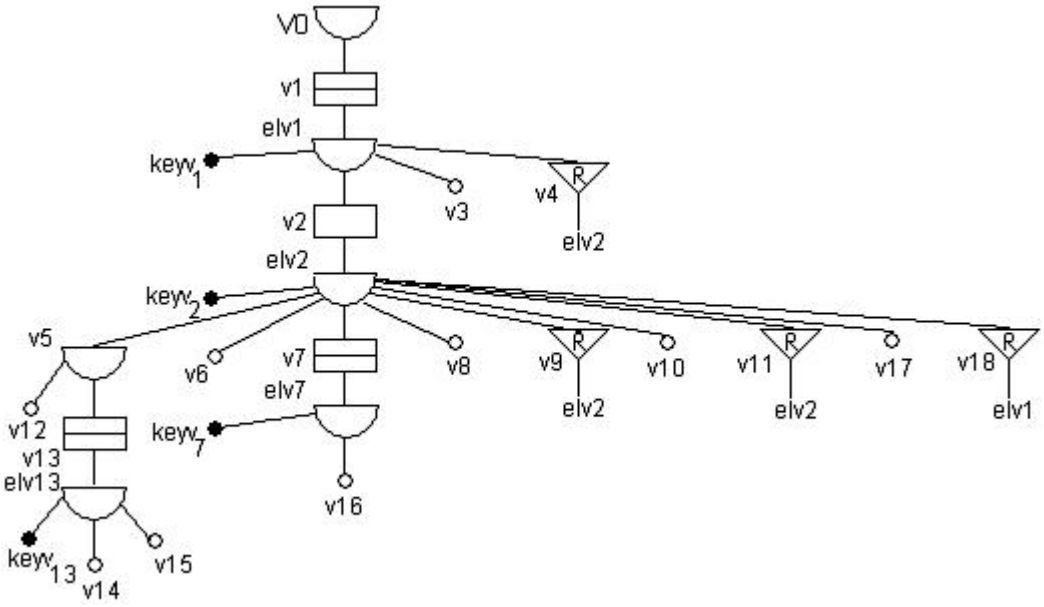


Рис.4.3. Описание данных для семантического представления документов средствами СУБД НИКА

Другой важный пример схемы описания данных (см.рис.4.4) — это база данных “За Христа пострадавшие” (http://martyrs.pstbi.ru/). База данных опубликована на сайте http://martyrs.pstbi.ru/ (www.pstbi.ccas.ru, http://kuz1.pstbi.ccas.ru/). Электронная публикация дублируется на зеркалах сайта: http://kuz3.pstbi.ru/bin/code.exe/frames/m/ind_oem.html/ans, http://sr.isa.ru/kuzn/frames/m/index.html. На момент написания этой работы в базе данных собрана информация о более, чем 36 тыс. пострадавших в годы гонений и свыше 6500 фотографий пострадавших. На основании базы данных издаётся многотомный биографический справочник, содержащий краткие биографический статьи в алфавитном порядке имён пострадавших (БС 1997, 2015). В приложении А приведена схема описания данных для массива ‘Дела’. Схема содержит около 100 реквизитов. Основные структурированные вершины — Родство, Конфессия, Дополнит.информ., Фотографии, ПЕРИОДЫ ЖИЗНИ, Образование, Рукоположение, Служение, МестаПроживания, Награды, Аресты, Осуждения, МестаЗаключения, Кончина, Канонизация, Реабилитация, Труды, Публикации, Заявитель, Документ.

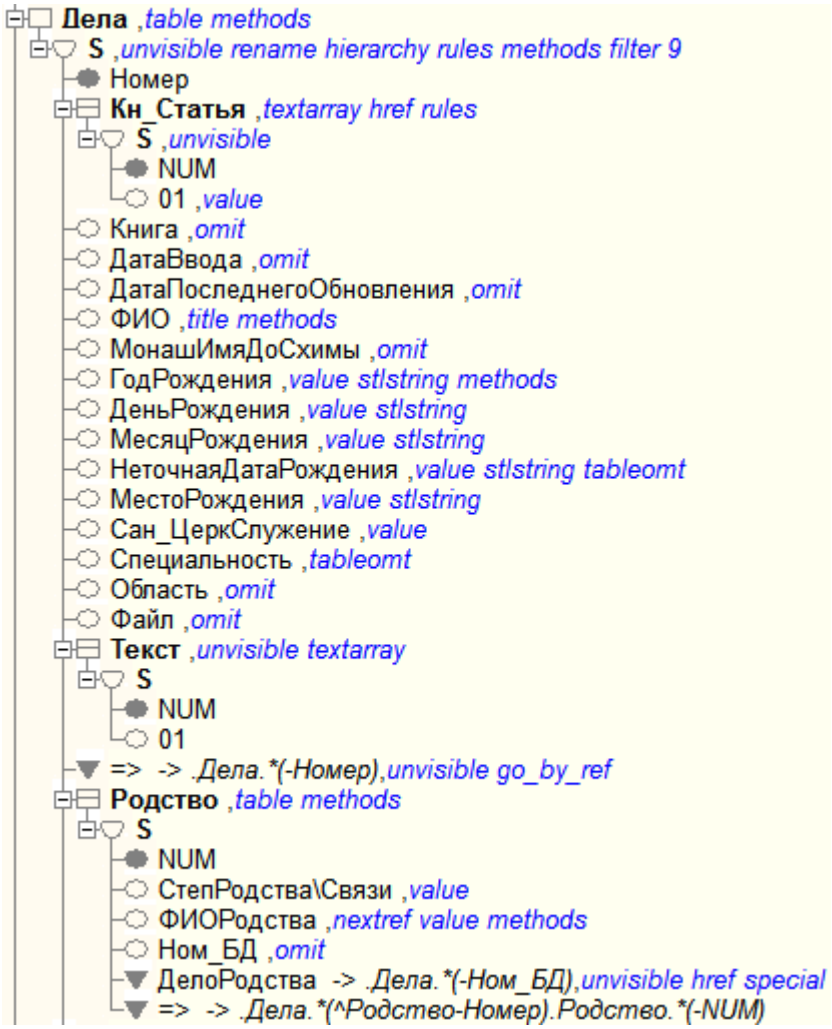


Рис.4.4. Фрагмент описания данных БД “За Христа пострадавшие”

Рассмотрим пример описания данных для построения многоуровневого классификатора для полей типа текст[*)], например, таких как ФИО, ФИОРодства, ФИОЗаявителя и т.д. (рис.4.5). “Буквы” — это массив с текстовым ключом *C*. *A* — это элемент массива, в котором задается циклическая ссылка на шаблон “Буквы”. Таким образом, элемент массива “*Буквы*” может содержать в себе произвольное число подчинённых вложенных массивов, имеющих такую же структуру, что и исходный массив “*Буквы*”. Это позволяет строить многоуровневый алфавитный классификатор любой степени сложности. Ключами массивов являются буквы или буквенные сочетания, соответствующие текущей позиции в текстовом ключе. Буквы, имеющие лишь

[*)] В СУБД НИКА поля типа текст ограничены длиной строки в 255 символов

одну подчинённую букву, могут объединяться в одно сочетание, присваиваемое ключу “Альфа” подчинённого массива.

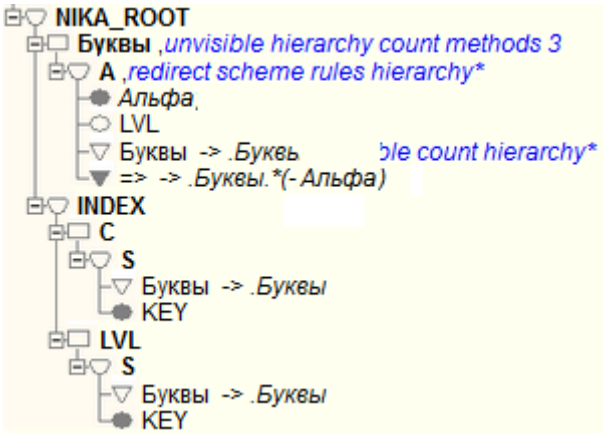


Рис.4.5. Описание данных для многоуровневого классификатора

Оптимальный классификатор определяется в смысле функционала $S_{on}(n,n_g)$, описанного в третьей главе. Для построения классификатора берутся оптимальные значения максимального числа ключей в классе n^* и числа ключей в группе n_g^*. Поскольку класс разбивается на группы, то $n_g^* \le n^*$. Каждый префикс исходного массива может быть получен из последовательности значений ключевого поля “Альфа” вложенных массивов “Буквы”. Классификатор строится, начиная с последнего уровня. Исходный массив на основе ПДС делится на классы с числом ключей, не превышающим n^*. Префиксы классов исходного массива составляют последний уровень классификатора. Полученный уровень классификатора разбивается на классы префиксов также как исходный массив. Итерационный процесс прекращается, когда число префиксов на текущем уровне классификатора будет не более n^*. Для каждого массива “Буквы” определён счётчик всех терминальных элементов в поддереве ПДС, который соответствует частоте встречаемости $\nu(s_i)$ данного префикса s_i. Префикс s_i добавляется на последний уровень классификатора, если $\nu(s_i) \le n^*$ и для всех префиксов префикса s_i их частота превышает n^*. При добавлении префикса на произвольный уровень классификатора частота

префикса корректируется, отсчитывая от предыдущего уровня префиксов, как от исходного массива (см.рис.2.7).

Будем рассматривать только ключевые уровни массива “*Буквы*” и вложенных массивов “*Буквы*” и гипертекстовые документы на основе этой схемы, не содержащие повторяющихся ссылок, в частности пути в БД. Тогда данная схема служит описанием данных для префиксного дерева сочетаний (ПДС). Такая схема определяет класс сетей Петри (Емельянов,Тищенко,2009), которые порождают регулярные языки. Сеть Петри с множеством переходов T называется *автоматной* (Котов,1984), если $\forall t \in T$: $|{\cdot}t|=|t{\cdot}|=1$, где символы ${\cdot}t$ и $t{\cdot}$ обозначают соответственно входное и выходное множества позиций для перехода t. Для сети Петри гипертекстовой системы БД НИКА на основе рассматриваемой схемы это определение выполняется. Это следует из того, что каждая вершина в иерархии вершин БД идентифицируется уникальной координатой в БД, которая и является гипертекстовой ссылкой на подчинённые вершины для нетерминальных вершин. Граф автоматной сети является связным, число меток в автоматной сети остаётся постоянным. При срабатывании перехода ровно одна метка из входной позиции перемещается в выходную позицию этого перехода. “Так как автоматная сеть ограничена, то её граф разметок конечен, следовательно, в классе автоматных сетей разрешимы проблемы достижимости разметки, проблемы живости, проблемы R-включения и R-эквивалентности, проблема эквивалентности по языкам… Автоматная сеть представляет собой сетевую форму задания конечных автоматов. Это непосредственно следует из конечности графа разметок любой автоматной сети. Этот граф представляет собой граф конечного автомата, в котором множество состояний образовано множеством достижимых в сети разметок, а алфавит — символами переходов в сети. Поэтому на автоматные сети распространяются все результаты теории конечных автоматов.” (Котов,1984) Проблема живости сети и проблема достижимости в ней любой разметки эквивалентны. Понятие живости перехода использует понятие достижимости из

данной разметки такой разметки, при которой данный переход может сработать. Живость сети — это живость любого её перехода. Если множество достижимых в сети разметок обозначить как $R(N)$, то R-включение для двух сетей (имеющих одно и то же множество мест) означает, что $R(N_1) \subseteq R(N_2)$, а R-эквивалентность — $R(N_1)=R(N_2)$. Рассматриваемая сеть гипертекстовой системы БД НИКА обладает свойством живости, т.е. для любого перехода достижима разметка, при которой этот переход может сработать. При работе сети предполагается, что в начальной позиции, соответствующей верхнему уровню алфавитного индекса, есть достаточное число меток для заданного максимального числа пользователей. При завершении работы пользователя соответствующая метка перемещается в начальную позицию, с которой связана любая позиция через переходы “завершения работы”. Таким образом, рассматриваемая сеть консервативна, т.е. общее число меток в ней постоянно.

4.3. Описание методов отображения вершин БД НИКА в гипертекстовые документы в виде спецификаций ядра гипертекстовой системы

Важным элементом любой базы данных являются индексы, построенные по различным полям БД. В этом смысле ПДС является некоторым более сложным индексом, чем просто алфавитный список ключей со ссылками на соответствующие записи БД. ПДС задаёт список ключей в виде иерархической структуры, элементы которой являются буквенные префиксы. Буквенные префиксы на одном уровне ПДС располагаются в алфавитном порядке. ПДС является основой для многоуровневого алфавитного классификатора для данного поля. Также ПДС может быть использовано для автозаполнения поля перехода по ключу на уровне (Тищенко,2013). Классификатор можно рассматривать как своеобразную «схему» ключевого уровня массива для данной предметной области. В этом смысле он является этапом в развитии технологии от базы данных к базе знаний (Арлазаров,2005). Авторы статьи, следуя М. Минскому, предлагают представлять знания в виде совокупности фреймов ”как структур данных для описания стереотипных ситуаций”. “С

программистской точки зрения — это интерфейс работы со знаниями… Главное, что фрейм имеет методы и интерфейсы: создание, корректировка, ввод/вывод порций разнородных знаний; выполнение запроса, который возвращает коллекцию объектов, удовлетворяющих условиям запроса; проверка непротиворечивости и целостности вводимых данных; запуск встроенных процедур в случае того или иного события; запись в журналы операций работы с данными и т. п.” Для визуализации БД, в частности алфавитного классификатора для заданного поля БД, были разработаны методы отображения вершин или объектов БД.

4.3.1. Типы отображения сложно структурированных данных

Прежде чем перейти к описанию методов отображения разных объектов БД, которые интегрированы с описанием схемы БД, проанализируем возможные способы представления сложно структурированных объектов. Все, выработанные человечеством, виды представления сложных структур на двухмерном листе бумаги (или экране дисплея) можно разделить на 4 группы (Емельянов.1987).

1. Последовательности или списки (одномерное представление):
    a) все данные представлены с их именами;
    b) семантическое представление (без имен со специально выделенными разделителями).
2. Таблицы (двумерное представление):
    a) с заголовками строк;
    b) с заголовками столбцов;
    c) с заголовками строк и столбцов;
    d) с заголовками строк, столбцов и разделов.
3. Иерархии (n-мерное представление):
    a) текст с заголовками (как правило с номерами разделов и уровней);
    b) древовидное (или графовое) представление (имена узлов используются для заголовков);

c) блок-схемное представление; семантические блоки изображаются элементами схемы (имя каждого блока является идентификатором подраздела);

d) ссылочное представление; более высокие семантические блоки ссылаются на подчиненные блоки (которые представляют собой отдельные документы).

4. Смешанное представление объединяет в себя комбинации первых трёх видов представления данных.

Для решения той или иной конкретной задачи выбирается одно конкретное представление данных, наиболее адекватное решаемой задаче. Последовательности замечательны компактностью представления данных, иерархии - наглядностью представления сложных структур, таблицы - возможностью просмотра по столбцам и строкам двумерных массивов. В рамках некоторого формализма, определяющего документ в виде семантических блоков, можно показать полноту предложенной классификации (Емельянов.1987). Доказывается, что “локально каждый семантический блок является последовательностью или таблицей, а документ в целом — последовательностью, таблицей, иерархией или комбинацией этих типов”. Описанная классификация видов представления данных, основанная на упомянутом формализме, применялась для построения различных систем на основе описанных видов представления данных (Emelyanov,Soloviov,1996; Emelyanov,Tishchenko,1996; Емельянов, Тищенко,1995,1997,1999).

4.3.2. Определения методов отображения и некоторые следствия

Введем следующее определения.

Определение 1. Методом отображения μ'' объекта O иерархической базы данных будем называть бинарное отношение, принадлежащее декартову произведению множества V всех вершин объекта O и множества всех подмножеств множества V: $\mu'' \in R = V \times 2^V$. Таким образом μ'': $V \to 2^V$.

Запись $\mu''(v) = S$, где $v \in V$, $S \in 2^V$, означает, что к вершине v применяется метод μ'', который отображает эту вершину в некоторое множество S подчиненных ей вершин. В объектной терминологии можно говорить, что объект (v,E(v)) *инкапсулирует* метод μ''.

Определение 2. Глубиной отображения n' объекта иерархической базы данных будем называть длину пути в графе $(V,E(V))$: $(v,v_1),(v_1,v_2),\ldots,(v_{n'-1},v_{n'})$, на которую применяется метод μ'' к вершине v.

В объектной интерпретации величину n' можно назвать "*глубиной наследования*" метода μ'' объектами, подчиненными $(v,E(v))$. Метод отображения μ'' с глубиной отображения n' будем обозначать $\mu_{n'}''$.

Определение 3. Метод отображения называется *частным,* если мощность множества S равна 1: $|S|=1$, т.е. множество S содержит один элемент, в противном случае, если $|S|>1$, метод называется *общим*. Частный метод отображения будем обозначать μ''^{p}.

Общие методы отображения относятся к четырем типам представления данных в виде последовательности, таблицы, иерархии или комбинации этих трех видов. Частные методы отображения используются совместно с общими и не изменяют общего вида представления, но модифицируют его. Совместное применение методов к объектам будем обозначать суммой $\mu''^{1}(v) + \mu''^{2}(v)$. В частном случае $\mu''^{1}(v) + \mu''^{2}(v) = S_1 \cup S_2$ (S_1 и S_2 - результаты действия методов μ''^{1} и μ''^{2}) или $\mu''^{1}(v) + \mu''^{2}(v) = S_1$, если результат отображения $\mu''^{2}(v)$ является $S_2 \equiv \varnothing$.

Определение 4. Общий метод отображения μ''^{s} называется *собственным*, если объект, к которому он применяется отображается только этим методом и не наследует других методов. Это достигается посредством совместного применения общего метода (общих методов) и метода sp (см. таблицу 1), т.е. $\mu''^{s}(v) = \mu''(v)+\mathrm{sp}(v)$.

Следствие 1. Все общие методы, применяемые к корневой вершине V_0 являются собственными.

Доказательство. Следует из определения базы данных, в соответствии с которым не существует дуг входящих в V_0, а значит не существует объектов, которым подчинен объект $(V_0,E(V_0))$, а также из определения собственного метода.

Следствие 2. К терминальным вершинам не применимы общие методы отображения, т.е. $\forall\, v\in V^* \Rightarrow \mu'' (v) \equiv \mu''^{p}(v)$.

Доказательство. Следует из определения базы данных (п.3.1), в соответствии с которым терминальные вершины не имеют подчиненных, т.е. $\forall v\in V^* \Rightarrow E(v)=\varnothing$.

Из этого утверждения следует, что общий тип представления задается методами нетерминальных вершинами.

Следствие 3. Пусть задан граф $(V,E(V))$ базы данных и вершина v не является корневой $v\neq V_0$ или терминальной $v\notin V^*$. Тогда если существует такая вершина v': $(v',v_1),(v_1,v_2),\ldots,(v_k,v)$ с общим методом отображения μ_n'': $n'>k'$ и объект $(v,E(v))$, соответствующий вершине v, не содержит собственных методов, то вершина v будет отображена методом $\mu_{n'-k'}''$.

Доказательство. Следует из определений 1 и 2.

В объектной терминологии можно говорить, что объект $(v,E(v))$ *наследует* метод μ'' от объекта $(v',E(v'))$.

4.3.3. Спецификации, управляющие отображением объектов

Как было сказано в п.4.3.2 существует четыре типа представления объекта на двумерной плоскости. Соответственно этим представлениям были реализованы методы отображения объектов СУБД НИКА. Эти методы применимы только к структурированным вершинам – массивам, структурам и ссылкам. Для массивов $V^{\#}$ характерны типы: последовательность – последовательность вершин текущего уровня, алфавитный указатель, текстовый массив; таблица подчиненных терминальных вершин; иерархия подчиненных вершин. Для структур $V^{@}$ характерны типы последовательность

подчиненных вершин и иерархия подчиненных вершин. Смешанный тип будет рассмотрен в п.6. Вторая группа методов отображения вершин не изменяет текущего представления, а влияет только на отображение данной вершины. Описание методов отображения объектов ООБД приводится в следующей сводной таблице (см. Приложения B,C для полного перечня спецификаций и их атрибутов).

Таблица 4.1

Описание спецификаций

Метод	Объект	Описание	Примечание
SQ **sequence**	Массив $V^{\#}$, Структура $V^{@}$	Список элементов текущего уровня в БД.	*Метод отображения объекта по умолчанию.*
SC **scale**	массив $V^{\#}$	Список элементов текущего уровня в БД в виде шкалы элементов в конце документа.	*Используется обычно совместно с другими методами (HRH,TAB) при сравнительно небольшом количестве элементов на уровне.*
SQA[n'] **alphabetical sequence**	Массив $V^{\#}$	Алфавитный указатель, содержащий первые ключевые элементы массива для каждой буквы алфавита. SQAn – для всех допустимых сочетаний **n** первых букв значений ключа.	*В настоящей версии обозначается как GRP, n'<=2.*
TXT **text**	Массив строк $V^{\#}$	Полнотекстовый документ.	—
TAB[n'] **table**	Массив $V^{\#}$ (вложенные массивы)	Таблица терминальных элементов структуры, являющейся элементом массива, ключевые элементы печатаются в виде гипертекстовых ссылок отдельным столбцом; TABn - таблица всех терминальных элементов на глубину n'.	*Метод отображения данных в виде первой нормальной формы или в виде вложенных таблиц.*

Метод	Объект	Описание	Примечание
HRH[n'] **hierarchy**	Массив $V^{\#}$, структура $V^{@}$	Фрагмент дерева, начиная с данной вершины до терминальных вершин (без гипертекстовых ссылок). HRHn- тоже что и HRH на глубину *n'*: массивы и структуры на уровне *n'* печатаются в форме гипертекстовых ссылок.	*Метод отображения данных в виде вложенных списков или с прорисовкой дерева.*[*)]
TIT **title**	Текст *V**	Значение терминальной вершины будет отображаться в виде заголовка (более крупным полужирным шрифтом).	*В html документе будет элементом:* * текст_знач *
IMG **image**	текст *V**	Значение терминальной вершины является именем файла изображения.	*В html документе будет элемент "изображение":* *<img src=имя_файла>*
MD **media**	текст *V**	Значением терминальной вершиной является имя видео/аудио файла.	*В html документе будет элемент "гипертекстовая ссылка":* *<a href= имя_файла >.*
HTTP	текст *V**	Значением терминальной вершины является гипертекстовая ссылка.	*В html документе будет "гипертекстовой ссылкой":* *<a href=гипер_ссылка>.*
OMT **omit**	любой объект *V**, *V#*, $V^{@}$	Вершина с этой спецификацией не будет отображена в html документе.	
TO **omit in table**	любая терминальная вершина *V**	Терминальная вершина не будет отображена в таблице в html документе.	*Метод используется совместно с методом таблица TAB, который инкапсулируется вышестоящим объектом типа массив.*
INO **omit in index**	любой объект *V**, *V#*, $V^{@}$	Вершина в индексе с этой спецификацией не будет отображена в html документе.	*Действует на вершины в индексе, расположенные выше ссылочной вершины '=>'.*

[*)] Примечание: далее в таблице приводятся частные методы отображения.

Метод	Объект	Описание	Примечание
VL **value**	любая терминальная вершина *V**	Будет отображено только значение терминальной вершины без ее имени.	
RN[имя] **rename**	любой объект *V**, *V#*, *V@*	Вершина будет отображена с указанным именем.	
RD[путь] **redirect**	массив *V#*, структура *V@*	Изменяет гипертекстовую ссылку на подчиненные вершины для нетерминальной вершины на указанный путь в базе данных.	*Новый путь в базе данных может быть только продолжением исходного пути, т.е. указывать на подчиненные вершины для данной нетерминальной вершины.*
STL[выраж] **style**	любая терминальная вершина *V**	Выражение для данного метода задает тип, размер, цвет шрифта и др. для отображения терминальной вершины в html документ.	*Отображает вершину посредством html-элемента:* * верш. *
HR **hypertext**	любая тер. верш. *V**	Присваивает ссылку следующей нетерминальной вершины.	*В html документе –"гипертек. ссылка".*
INR **index hypertext**	любая терминальная вершина *V**	Присваивает ссылку следующей нетерминальной вершины (если такая существует) данной терминальной вершине в индексе.	*Используется для ключей массивов. В html документе будет "гипертекстовой ссылкой":* * верш *
UNV **unvisible**	массив *V#*, структура *V@*	Не отображает имя нетерминальной вершины в иерархии.	*Используется совместно с методом иерархия HRH, который инкапсулируется вышестоящим объектом.*
VIS **visible**	терминальная вершина *V**, ключ	Отображает ключи массива.	*Используется совместно с методом иерархия HRH, который инкапсулируется вышестоящим объектом.*
HYP **hypertext reference**	массив *V#*, структура *V@*	Отображает нетерминальную вершину как гипертекстовую ссылку в иерархии.	*Используется совместно с методом иерархия HRH, который инкапсулируется вышестоящим объектом. По умолчанию все вершины в иерархии (кроме последнего уровня) печатаются как обычный текст с отступами.*

Метод	Объект	Описание	Примечание
NOH **no hypertext reference**	массив $V\#$, структура $V^{@}$	Отображает нетерминальную вершину как терминальную, т.е. без гипертекстовой ссылки.	
GO **go by reference**	ссылка на шаблон V^{P}, ссылка на значение V^{r}	Идет по ссылке ‘=>’ в индексе, т.е. осуществляет показ отобранных объектов	*Используется в случае, когда проиндексированный атрибут не непосредственно подчинен объекту верхнего уровня. Метод применяется для отображения типа иерархия.*
NU **no unodered list**	структура $V^{@}$	Печатает в иерархии на одной строке с данной вершиной подчиненные ей вершины.	*При отображении типа иерархия не вставляет html-элемента список после данной нетерминальной вершины.*
SP **special method**	массив $V^{\#}$, структура $V^{@}$	Позволяет отображать нетерминальный объект методом, который инкапсулируется именно этим объектом (таблица, иерархия на указанную глубину), а не методом, наследуемым от объекта, которому он подчинен.	*Используется совместно с методом иерархия HRH и таблица TAB.*

4.3.4. Практическое использование смешанных методов отображения

В практическом смысле наибольший интерес представляет смешанный тип. Рассмотрим следующие примеры. Пусть *m’* – число вершин непосредственно подчиненных вершине *v*.

1. **HRH(*v*) + SC(*v*)**. Представление вершины *v* в виде иерархии и последовательности (при небольших *m’*) .

2. **TAB(*v*) + SC(*v*)**. Представление вершины *v* в виде таблицы и последовательности (при небольших *m’*).

3. **SQA(*v*) + HRH(*v*).** Представление вершины v в виде алфавитного указателя и иерархии (при достаточно больших *m’*).

4. **SQA(*v*) + TAB(*v*).** Представление вершины v в виде алфавитного указателя и таблицы (при достаточно больших *m'*).

5. **HRH(*v*) + TAB^s(*v*$_1$') + SC^s(*v*$_2$').** Представление объекта (*v*,*E*(*v*)) в виде иерархии подчиненных объектов, причем подчиненный объект (*v*$_1$',*E*(*v*$_1$')) имеет представление таблица, а подчиненный объект (*v*$_2$',*E*(*v*$_2$')) имеет представление в виде последовательности всех подчиненных непосредственно ему объектов.

6. **HRH(*v*) + TXT(*v'*).** Представление объекта в виде иерархии. Подчиненная вершина *v'* отображается в виде текстового массива.

7. **TAB(*v*) + TXT(*v'*).** Представление объекта в виде таблицы. Подчиненная вершина *v'* отображается в виде текстового массива.

Приведем примеры совместного использования частных методов отображения.

1. **SQ(*v*) + OMT(*v'*); TAB(*v*) + OMT(*v'*); HRH(*v*) + OMT(*v'*).** Последовательность, либо таблица, либо иерархия с исключением подчиненной вершины *v*'.

2. **TAB(*v*) + TO(*v'*).** Таблица объектов, подчиненных *v* с исключением из нее вершины *v'*. В отличии от omt этот метод действует только для таблицы.

3. **HRH(*v*) + UNV(*v*).** Иерархия объектов, подчиненных *v* без указания имени вершины &*v*.

Замечание. Описанные примеры можно распространить на любое количество подчиненных вершин, например, **HRH(*v*) + OMT(*v*$_1$') + … + OMT(*v*$_k$')**, где вершины *v*$_1$',…, *v*$_k$' подчинены вершине *v*. Аналогичным образом применяются другие частные методы MD, HTTP, IMG и т.д.

4. **SQ(*v*) + RN[*v''*](*v'*) + IMG(*v''*).** Такое сочетание позволяет отобразить вершину *v* в виде последовательности подчиненных объектов, представляющих собой изображения. Метод RN переименовывает *v'* в

значение терминальной подчиненной вершины *v''*, в свою очередь к *v''* применяется метод изображения IMG.

4.3.5. Задание спецификаций в схеме ООБД

Описанные в п.3 спецификации задаются администратором http-сервера в вершинах схемы ООБД. Согласно стандарту ISO/IEC 10027[*)] описание информационного ресурса должно осуществляться теми же средствами, на которых построена работа с этим информационным ресурсом. Следовательно, описание спецификаций в схеме ООБД должно осуществляться средствами стандартных браузеров.

Схема ООБД отображается в виде дерева, каждой вершине которого соответствует окно выбора спецификаций, допустимых для данного типа вершины (см. Приложение А). Выбор спецификации обеспечивает соответствующую форму отображения ООБД http-сервером.

4.3.6. Дополнительные спецификации для отображения html-документа

Среди гипертекстовых html-документов будем рассматривать только правильные html-документы в смысле (Богачева,Емельянов,2001), чтобы исключить ошибки в html-документе с точки зрения семантики. Кроме того, важно отметить, что значения html-элементов "определяются только в контексте всего документа" (Богачева,Емельянов,2001). О том какие представления данных можно построить в частности на основе html-элементов было сказано выше в пункте 4.3.1. Относительно гипервершинной модели данных (Levene,1998) необходимо отметить, что в изложенной модели гипертекста посредством СУБД НИКА каждый элемент массива документов v_1 можно понимать как гипервершину, представляющую собой иерархию вершин el_{v1}, которая связана с другими гипервершинами того же массива документов v_1 посредством гипертекстовых ссылок, т.е. ссылок на значения $ref_{v1}(v_{17})$ (см.п.4.2, пример о представлении гипертекста в СУБД НИКА). Такая гипервершина

[*)] Information technology - Information Resource Dictionary System (IRDS) framework// ISO/IEC 10027: 1990 (E).

может инкапсулировать метод отображения иерархия (Емельянов,1987) для отображения html-документа, записанного в БД. Для отображения такого html-документа в программу html/xml сервера СУБД НИКА (Емельянов,Тищенко,2009) были добавлены дополнительные спецификации для отображения вершин БД (см. Таб.4.2).

Таблица 4.2

Дополнительные спецификации для отображения html-документа

Метод	Объект	Описание
HTMR **html root element**	$V^{@}$, $V^{\#}$, V^{r}	отмечает корневой элемент для иерархии html-элементов
HTML **html element tag**	$V^{@}$, v: $v \in V^{@}$, T(v)={ T(*tag*), { [T(key_{attrs}), T(*attr*), T(*val*)] }	отображает структуру как метку html-элемента в виде <tag[a_1=v_1…a_n=v_n]>...</tag>
NLB **no line break**	V^{*}	отменяет переносы строк для терминальных вершин

4.3.7. Спецификация шаблон TPL для отображения вершин посредством html/xml-шаблона

Для отображения вершин базы данных НИКА с использованием произвольного форматирования в html/xml-формате вводится спецификация шаблон (Тищенко,2012). Шаблоны хранятся в файле nkws.tpl. Каждый шаблон занимает блок памяти размером 4KB. Параметром к спецификации TPL является номер блока в файле nkws.tpl. В шаблоне используется обычный синтаксис html/xml-элементов, а также добавлены управляющие символы, которые приводятся в следующей таблице.

Таблица 4.3

Управляющие символы html/xml-шаблона

Обозначение	Отображение	Примечание
^&dn	ключ вершины с dod-номером dn	dn обозначает dod-номер
^*dn	значение вершины с dod-номером dn	—
^^dn	координата вершины с dod-номером dn	используется в гипертекстовой ссылке

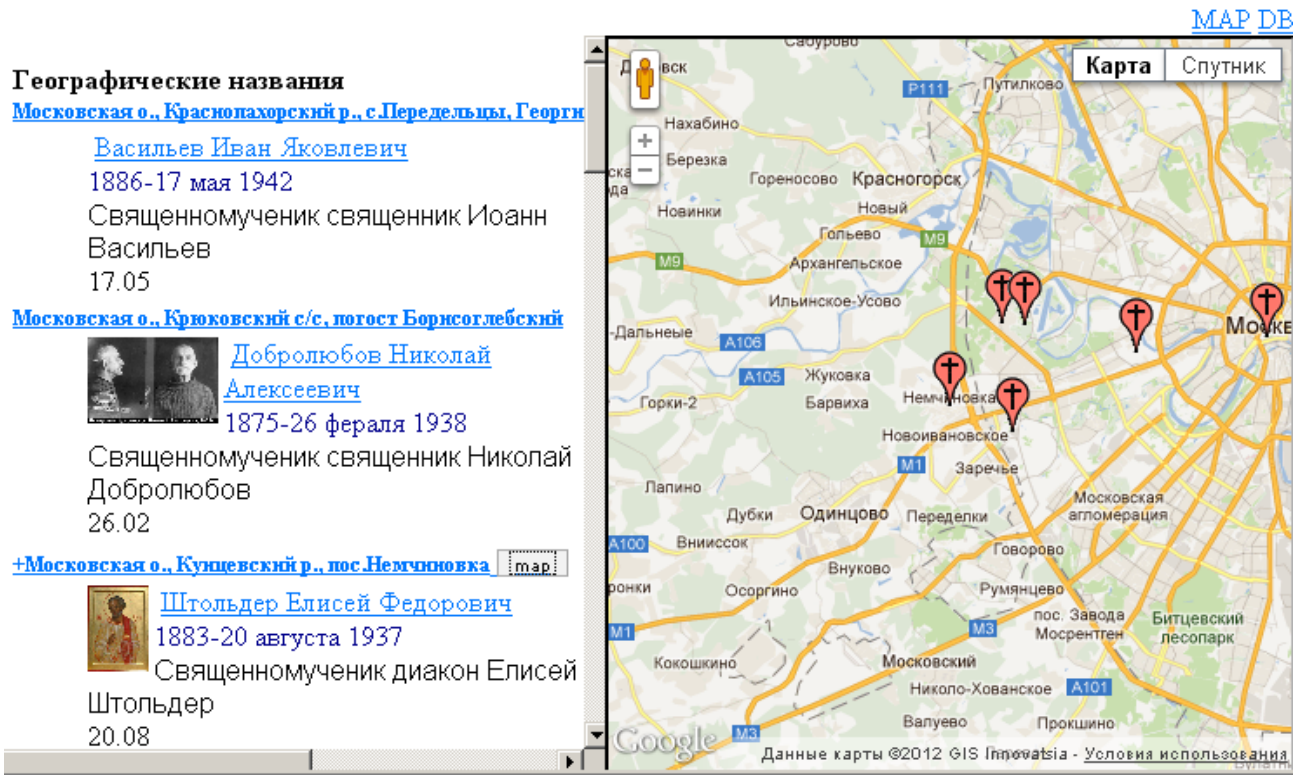


Рис.4.6. Фрагмент индекса по географическим наименованиям с использованием спецификации TPL и обозначением на карте google.

В шаблоне для заданной вершины в управляющих символах могут быть указаны dod-номера вершин, подчиненных заданной вершине базы данных. При этом путь в базе данных отсчитывается от текущей вершины, которой назначена спецификация TPL. Кроме dod-номеров подчиненных вершин может быть указан dod-номер текущей вершины. Недостатком такого подхода является то, что использование dod-номеров не дает возможности однозначно идентифицировать вершину в базе данных. Указанный недостаток устраняется посредством задания индексов или ключей массивов, в которых содержится вершина с данным dod-номером и просмотром сетевых связей.

При использовании географической карты можно воспользоваться спецификацией TPL для отображения вершин в произвольном виде с помощью html/xml шаблона (см.рис.4.6).

Из индекса видно, что название «Московская о., Кунцевский р., пос. Немчиновка», обозначенное на карте, было одним из мест служения сщмч. Елисея Штольдера (1883-20.08.1937).

4.3.8. Описание спецификации, отображающей данные в формате географического языка разметки

При отображении географических названий, представляющих собой строковый массив БД НИКА, с помощью картографического сервиса данному массиву назначается спецификация MAP (см.рис.4.7)

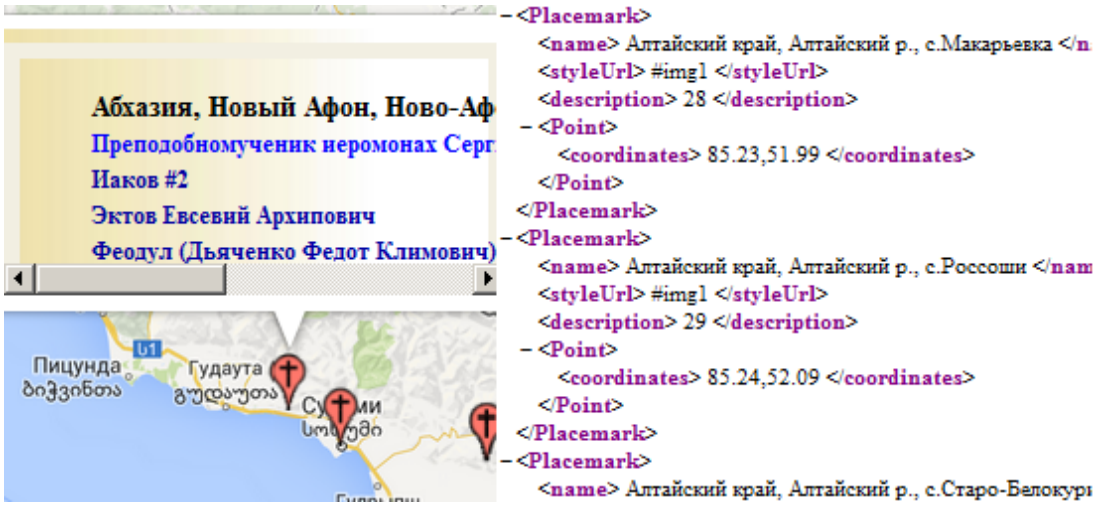


Рис.4.7. Информационное окно отображает список имен пострадавших. Фрагмент информационного слоя

Дальнейшим развитием этой спецификации является объединение географических названий в информационный слой. Спецификация FO (от англ. formatting objects) xml/html-сервера форматирует объекты БД НИКА в виде элементов географического языка разметки (см.рис.4.7). Как видно из примера для описания каждого географического объекта используется метка (Placemark). Элементы name, description, point задают название, идентификатор в БД, координаты географического объекта соответственно. Геометрически каждый объект представлен в виде точки с двумя координатами, задаваемыми элементом coordinates. Элемент styleUrl ссылается на описание значка для обозначения географического объекта на карте. В качестве обозначений на карте могут быть также использованы фотографии новомучеников. Совокупность блоков, представленных метками, определяющими объекты на карте, составляет xml-документ. Каждый блок этого xml-документа (Емельянов,1987) может быть отображен в виде информационного окна при выборе соответствующего названия. Переопределение события, связанного с

выбором названия места, дает возможность по идентификатору названия выполнить запрос к БД “Новомученики и исповедники” и сформировать посредством xml/html-сервера документ, содержащий список служивших в этом месте пострадавших со ссылками на их биографические справки.

4.3.9. Описание спецификации, реализующей автозаполнение

Автозаполнение (Bast,2006) представляет собой способ заполнения поля формы из предлагаемого списка вариантов слов и/или сочетаний слов, начинающихся с введенной в поле комбинации букв. Эта возможность используется в полнотекстовых поисковых системах или при заполнении форм. Здесь описание реализации автозаполнения для поля перехода на ключевом уровне массива БД НИКА. Xml/html-сервер СУБД НИКА создает страницы в соответствующем формате для построения списка автозаполнения. Страницы включают функции динамического изменения xml/html-документа без полной его перезагрузки. Список автозаполнения позволяет просмотреть наиболее часто встречающиеся в заданном поле БД варианты слов и ускорить нахождение требуемой вершины БД на ключевом уровне массива. Под ключевым уровнем массива будем понимать набор уникальных однотипных (в данном случае строковых) значений.

Основой спецификации автозаполнения (Тищенко,2013) служит, так называемое, префиксное дерево сочетаний, описанное в статье, посвященной построению многоуровневого индекса ключевого массива (Емельянов,Тищенко,2010). Это дерево содержит частоты встречаемости префиксных сочетаний букв, составляющих вершины рассматриваемого ключевого уровня массива. Таким образом, для любого сочетания букв можно определить список ключевых слов, начинающихся с данного сочетания, и соответствующие им частоты встречаемости этих слов на ключевом уровне, используя префиксное дерево сочетаний. Ключевое слово – это любое слово, входящее в произвольную вершину на ключевом уровне. Из полученного списка ключевых слов формируется список из 5 (или любого другого заданного

числа) наиболее часто встречающихся слов. Перед выборкой слов список упорядочивается в порядке убывания частот встречаемости. Результат выборки передается в ответ на запрос на автозаполнение. Интерфейсом или некоторой оболочкой реализации автозаполнения служит сценарий, встроенный в xml/html-документ, динамически создаваемый xml/html-сервером БД НИКА. Особенностью работы сценария служит задержка по времени перед запуском запроса на автозаполнение для того, чтобы уменьшить число таких запросов при быстром вводе исходного сочетания пользователем, например, при средней скорости набора 1 символ в 0,3с. В этом случае нет необходимости делать запрос при каждом набранном символе, а сразу после набора всего сочетания. Минимальная задержка между запросами на автозаполнение от одного пользователя равна 0,5с (при необходимости ее можно увеличить).

Далее приводится схема работы спецификации и описан процесс выполнения запроса на автозаполнение.

Условно можно выделить 2 шага в запросе на автозаполнение: загрузка xml/html-страницы, содержащей фрагмент ключевого уровня массива БД, в которую встроен сценарий, являющийся промежуточным звеном между пользовательским интерфейсом и xml/html-сервером БД НИКА, обрабатывающим запрос на автозаполнение (см. рис.4.8). Результатом работы системы, показанной на рис.4.8, является xml/html-документ с текущим фрагментом ключевого уровня массива БД с возможностью автозаполнения

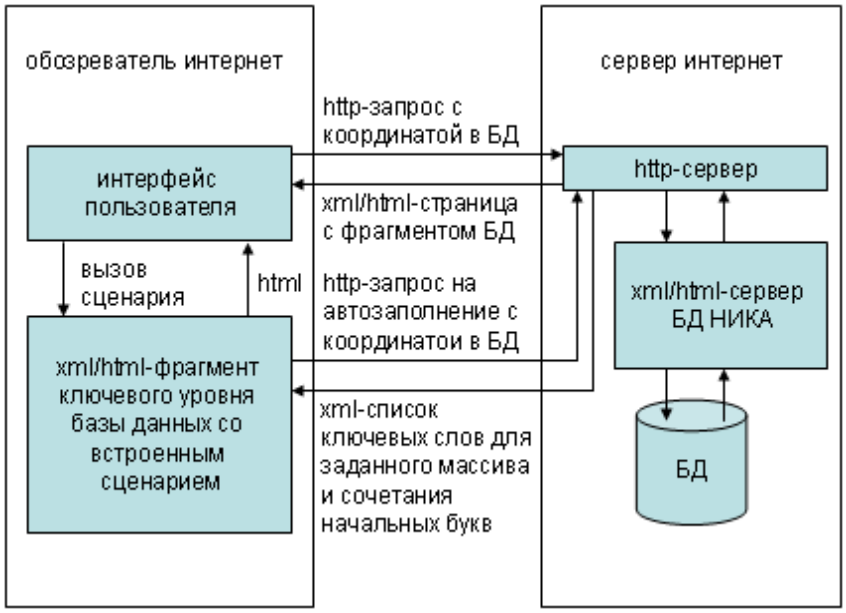


Рис.4.8. Обработка запроса на автозаполнение для заданного массива БД

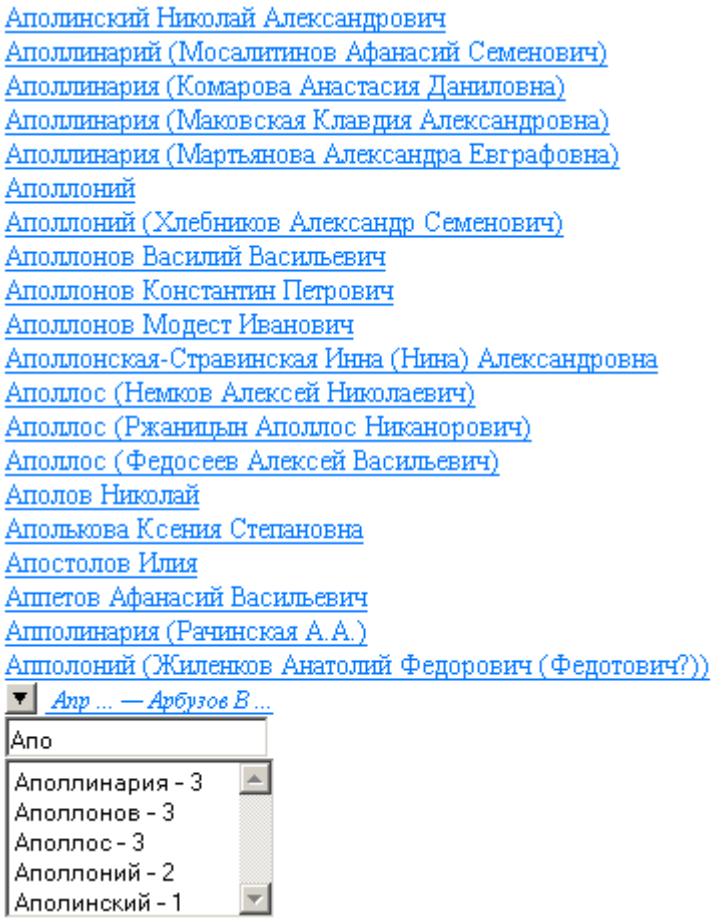


Рис.4.9. Пример фрагмента ключевого уровня массива ФИО с возможностью автозаполнения

поля перехода на текущем уровне БД. На рис.4.9 показан пример списка для заполнения поля перехода именами и фамилиями пострадавших, сформированного на основе индекса в виде префиксного дерева сочетаний по полю ‘ФИО’ БД «Новомученики и исповедники». При вводе каждого символа сочетания ‘Апо’ с промежутком времени не менее, чем в 0,5с происходит соответствующий запрос на автозаполнение. В списке представлены 5 наиболее встречающихся слов на ‘Апо’, отсортированных по частоте встречаемости.

4.4. Описание отображения фрагментов БД НИКА в xml-формате посредством языка XSL в виде надстройки над ядром гипертекстовой системы, используемой для тонкой настройки ядра

4.4.1. Расширяемый язык таблиц стилей как схема данных для динамически созданных xml-документов

Расширяемый язык таблиц стилей (extensable stylesheet language) состоит из следующих компонент: XSLT преобразование (XSL transformations) xml-элементов в другие xml-элементы, которые определяются заданным пространством имен; язык выражений XML path language (XPath), используемый XSLT, а также объекты форматирования, представляющие собой развитие ранее определенных стандартов стилей для xml и html класса CSS (cascading style sheet). Частным случаем XSL преобразования является преобразование xml-элементов в html-элементы. Язык XSL включает в себя возможности языка SGML описания типа данных (data type definition — DTD) на более низком уровне общности и определяет способы отображения элементов. Это позволяет использовать XSL для определения структуры новых элементов на основе уже определенных элементов, html-элементов и строк. Совокупность определенных элементов в виде xsl-шаблонов представляет собой тип xml-документа. Таким образом иерархия xsl-элементов может рассматриваться как схема xml-документа. Такая модель может быть хорошо использована для отображения вершин БД НИКА. Ранее при отображении использовался язык HTML, в котором отсутствовала схема данных. Поскольку БД НИКА имеет схему данных, то новая модель с использованием xsl-схемы

является более предпочтительной, чем html-документ. Фрагмент БД НИКА отображается в xml-документ, к которому применяется "стандартный" xsl-шаблон или xsl-шаблон, определенный пользователем. Преимущества схемы данных для объектно-ориентированного языка разметки OOML указываются в (Bogacheva,Emeljanov,1995) (хотя назначения OOML и XSL различно): логическая и физическая независимость данных, использование модели данных, интероперабельность, поддержка целостности. Конечным результатом применения xsl-преобразований к xml-документу для показа через стандартную программу просмотра, как и в первоначальной модели, является html-документ.

Наиболее общим видом xml-файла, динамически созданного посредством cgi-программы из БД НИКА, является вид, в котором названия элементов — это имена вершин БД и нетерминальные вершины отображаются в элементы, содержащие в себе подчиненные элементы, а терминальные вершины отображаются в соответствующие элементы, содержащие значения этих вершин. Такой вид соответствует виду dcm-файла, содержащего фрагмент БД НИКА, и используемого в макетном редакторе МАГИС (Bogacheva,Emeljanov,1995). Другим видом xml-файла, полученного из СУБД НИКА, является вид с использованием введенного в данной статье элемента list (список), определенного посредством стандартных элементов xsl. Элемент list отображает вершины БД НИКА в виде списка или иерархического списка (посредством вложенных элементов ul для формирования html-документа) и имеет атрибуты для управления отображением элементов на экране.

4.4.2. Построение отображения вершин базы данных НИКА в XML или HTML документ

Возможны различные способы построения отображения вершин базы данных НИКА в гипертекстовый документ посредством шаблона XSL. Один способ состоит в том, чтобы сразу динамически создавать XML-документ, используя XSL-элемент **list -** список**.** Для форматирования элементов списка можно использовать HTML-элементы, задавая их как атрибуты к элементам

списка **el** (см.ниже). Для отображения иерархии элементов можно использовать тот же самый элемент **list** с вложенными списками, т.е. в качестве элемента списка может быть использован элемент **list**. Элемент **list** позволяет отсортировать элементы списка в разном порядке. В случае табличного представления данных следует реализовать соответствующий элемент на языке XSL. О возможных видах представления данных в документе см. работу (Емельянов.1987). Пример элемента “список” для показа вершин базы данных НИКА. Элемент написан посредством XSL-шаблона.

Описание XSL-шаблона list.

```
<list  type="text|number|{user_defined_type}"
       order="+|-"
       kind="disc|decimal|none|delimiter"
       delimiter={string-value}
       first_delimiter={string-value}
       last_delimiter={string-value} help="1">

       (<rem>text string </rem>)
       (<att>text label for attributes print </att>)
       <el (type={html_tag} (value={html_attrs}))> element 1 </el>
       <el (type={html_tag} (value={html_attrs}))> element 2 </el>
       <el (type={html_tag} (value={html_attrs}))> element 3 </el>
       . . .
</list>
```

list - список

type = { ["text"] | "number" | qname-but-not-ncname } - тип списка (для сортированного списка является обязательным параметром);

order = { ["+"] | "-" }- xsl:sort order={ "ascending" | "descending" }порядок сортировки - по возрастанию или по убыванию;

kind = { ["disc"] | "decimal" | "none" | "delimiter" }- просмотр списка, как

ненумерованного, нумерованного, простого списка или списка с разделителями, определенными пользователем;

delimiter = { text-string }- разделитель элементов списка;

first_delimiter = { text-string }- разделитель перед первым элементом;

last_delimiter = { text-string }- разделитель после последнего элемента;

help = { "1" }- просмотр описания элемента list;

[**...**] - значение по умолчанию;

el - элемент списка

type = { html-tag-string }- a, img, h1 и т.д.

value = { html-element-attributes }- атрибуты html-элемента, перечисленные через точку с запятой:

ATTR1=VALUE1;ATTR2=VALUE2;ATTR3=VALUE3;**...**;ATTRn=VALUEn

Элементы или атрибуты, указанные в круглых скобках могут быть пропущены.

Другой способ включает в себя первый. Динамически создается XML-документ, содержащий фрагмент иерархии базы данных НИКА. Имена вершин - это элементы XML. Содержимое элементов - это значения для терминальных вершин и подчиненные вершины для структурированных вершин. Для показа этой иерархии различными типами отображения необходимо применить к полученной иерархии соответствующие XSL-элементы. Так для показа в виде списка необходимо применить элемент **list**. Для применения XSL-элемента к соответствующей иерархии вершин необходимо организовать проход по этой иерархии посредством рекурсивного обращения:

```
<xsl:template name="db_vertices_hrh_pass">
  <xsl:param name="vertices"/>
  ...
    <!-- для каждой подчиненной  вершины, задаваемой параметром vertices, относительно текущей вершины -->
```

```
    <xsl:for-each select="$vertices">
      ...
      <!-- рекурсивный вызов с параметром vertices="*": обработка всех
подчиненных вершин -->
      <xsl:call-template name="db_vertices_hrh_pass">
       <xsl:with-param name="vertices"  select="*"/>
      </xsl:call-template>
      ...
    </xsl:for-each>
  ...
</xsl:template>
```

Параметр vertices задает XPath выражение, соответствующее всем элементам, подчиненным текущему. Указанный параметр должен быть задан при первом обращении к элементу db_vertices_hrh_pass, т.к. в нем необходимо указать первую текущую (контекстную) вершину. Например, если верхним элементом иерархии элементов вершин является **doc**, содержащий все остальные элементы, то вызов должен выглядеть следующим образом:

```
<!-- обработать все вершины, подчиненные элементу doc -->
<xsl:call-template name="db_vertices_hrh_pass">
  <xsl:with-param name="vertices"  select="/doc/*"/>
</xsl:call-template>
```

Элемент **list** необходимо вызывать в процессе обработки; содержимое элементов вершин представляется с помощью элемента **el**.

Рассмотрим более простой пример отображения с использованием HTML-элемента <ul> — ненумерованный список. Тогда описанный выше XSL-элемент прохождения по иерархии элементов вершин базы данных НИКА

будет выглядеть следующим образом (шаблон db_vertices_hrh_pass вызывается без параметра непосредственно из шаблона doc — верхний элемент иерархии отображенных вершин):

exmpl.xsl

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">

<xsl:template match="doc"> <!-- верхний (корневой) элемент -->
 <html>
  <body>
   <xsl:call-template name="db_vertices_hrh_pass"/> <!--вызов шаблона для прохождения по-->
</body>                                    <!--иерархии элементов, содержащихся в doc,-->
 </html>                                    <!--вместо обычного вызова xsl:apply-templates-->
</xsl:template>

<xsl:template name="db_vertices_hrh_pass"> <!--шаблон для прохождения по иерархии-->
 <ul> <!--элементов и печати в виде html-списка-->
  <xsl:for-each select="*">
   <xsl:variable name="vert_name" select="name(current())"/>
   <xsl:variable name="vert_prn">
    <xsl:choose>
     <xsl:when test="starts-with($vert_name,'v-')">
      <xsl:value-of select="substring-after($vert_name,'v-')"/>
     </xsl:when>
     <xsl:when test="starts-with($vert_name,'n-')">
      <xsl:value-of select="substring-after($vert_name,'n-')"/>
```

```
    </xsl:when>
    <xsl:otherwise>
     <xsl:value-of select="$vert_name"/>
    </xsl:otherwise>
   </xsl:choose>
  </xsl:variable>
  <xsl:choose>
   <xsl:when test="current()/*"> <!-- структурированная вершина -->
    <li><b><xsl:value-of select="$vert_prn"/></b></li> <!-- название вершины -->
    <xsl:call-template name="db_vertices_hrh_pass"/> <!-- обработка подчиненных
-->
   </xsl:when>
   <xsl:otherwise> <!-- терминальная вершина -->
    <li>
     <xsl:choose>
      <xsl:when test="starts-with($vert_name,'v-')">
       <xsl:choose>
        <xsl:when test="$vert_prn='1'">          <!-- только значение -->
         <xsl:value-of select="current()"/>
        </xsl:when>
        <xsl:when test="$vert_prn='2'">          <!-- изображение -->
         <img>
          <xsl:attribute name="src"><xsl:value-of select="current()"/></xsl:attribute>
         </img>
        </xsl:when>
        <xsl:when test="$vert_prn='3'">          <!-- заголовок -->
         <h3><xsl:value-of select="current()"/></h3>
        </xsl:when>
       </xsl:choose>
```

```
      </xsl:when>
      <xsl:otherwise>                                    <!-- название = значение -->
       <xsl:value-of select="name(current())"/> = <xsl:value-of select="current()"/>
      </xsl:otherwise>
     </xsl:choose>
    </li>
   </xsl:otherwise>
  </xsl:choose>
 </xsl:for-each>
</ul>
</xsl:template>

</xsl:stylesheet>
```

Для проверки этого шаблона можно рассмотреть xml-файл:

exmpl.xml

```
<?xml version="1.0" encoding="windows-1251"?> <!-- указание кодировки -->
<?xml-stylesheet href="exmpl.xsl" type="text/xsl"?> <!-- ссылка на xsl-шаблон -->
<doc> <!-- верхний элемент -->
 <a1>
  <a2> text 2 </a2>
  <a3> text 3 </a3>
 </a1>
 <a4> text 4 </a4>
 <a5>
   <a6> text string 1 </a6>
   <a7> text string 2 </a7>
   <a8>
    <a9> вершина 9 </a9>
```

```
    <a10> вершина 10 </a10>
    <a11> вершина 11 </a11>
    <a12>
      <a13> vertex text </a13>
      <a14> vertex text string </a14>
    </a12>
   </a8>
 </a5>
 <a15> test vertex </a15>
</doc>
```

В результате на экране должно быть отображение элементов в виде вложенных html-списков:

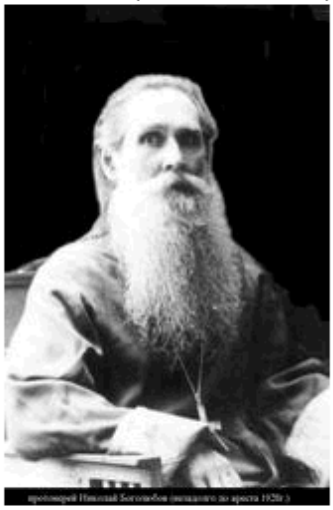
- **a1**
  - a2 = text 2
  - a3 = text 3
- a4 = text 4
- **a5**
  - a6 = text string 1
  - a7 = text string 2
  - **a8**
    - a9 = вершина 9
    - a10 = вершина 10
    - a11 = вершина 11
    - **a12**
      - a13 = vertex text
      - a14 = vertex text string
- a15 = test vertex

Рис.4.10. Отображение иерархии вершин

Используя тот же самый xsl-шаблон в случае базы данных "Новомученики и исповедники Русской Православной Церкви XXв." можно получить следующий результат:

Боголюбов Николай Михайлович

- ГодРождения = 1872
- ДеньРождения = 8
- МесяцРождения = 5
- МестоРождения = Нижегородская губ., Ардатовский у., с.Павловское
- протоиерей
- Сын священника и отец трех сыновей-академиков



протоиерей Николай Боголюбов

- **ПЕРИОДЫ ЖИЗНИ**
  - 1
    - Период жизни=1886-1909гг.
    - **Образование**
      - 1
        - Лысковское духовное училище
        - ГодОкончания = 1886

Рис.4.11. Отображение биографической справки

. . .

Исходным XML-файлом является следующий.

```
<?xml version="1.0" encoding="windows-1251"?>    <!-- указание кодировки -->
<?xml-stylesheet href="m.xsl" type="text/xsl"?>    <!-- ссылка на xsl-шаблон -->
<doc>
 <v-3>Боголюбов Николай Михайлович</v-3>
 <ГодРождения>1872</ГодРождения>
 <ДеньРождения>8</ДеньРождения>
 <МесяцРождения>5</МесяцРождения>
```

<МестоРождения>Нижегородская губ., Ардатовский у.,
с.Павловское</МестоРождения>
<v-1>протоиерей</v-1>
<v-1>Сын священника и отец трех сыновей-академиков</v-1>
<Фотографии>
<n-1>
<v-2>oni-10.jpg</v-2>
<v-1>протоиерей Николай Боголюбов 1886-1909гг.</v-1>
</n-1>
</Фотографии>
<ПЕРИОДЫ_ЖИЗНИ>
<n-1>
<Период_жизни>1886-1909гг.</Период_жизни>
<Образование>
<n-1>
<УчебЗаведение>Лысковское духовное училище</УчебЗаведение>
<ГодОкончания>1886</ГодОкончания>
</n-1>
. . .
</doc>

Исходным XSL-шаблоном служит тот же самый шаблон db_vertices_hrh_pass, помещенный в файл m.xsl.

4.4.3. Схема работы гипертекстовой системы БД НИКА в режиме XSL-надстройки с использованием БД спецификаций

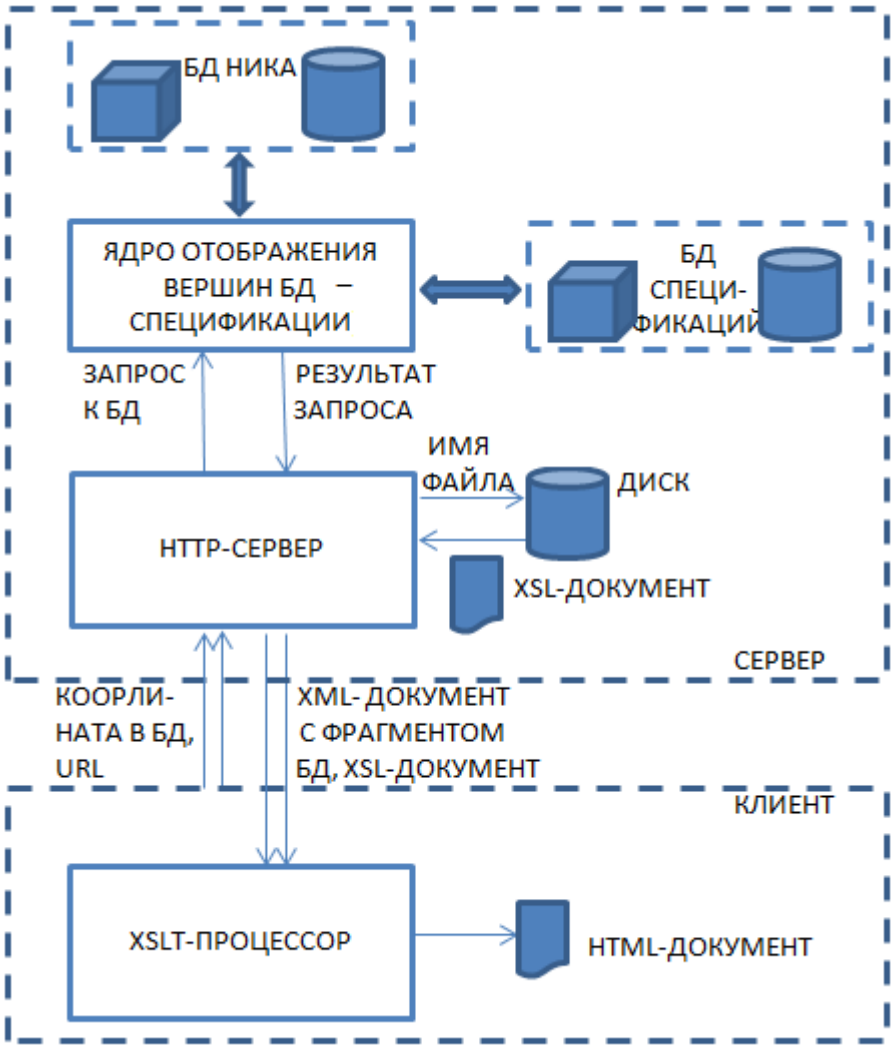


Рис.4.12. Функционирование гипертекстовой системы БД НИКА в режиме XSL-надстройки

Отличительной особенностью работы системы в режиме XSL-надстройки является работа с использованием базы данных спецификаций, выполненной в рамках общей технологии теми же средствами, что и основная БД. В отличие от предыдущего пункта здесь нет преобразования исходных html-элементов отображённых вершин в xml-элементы для того, чтобы получить возможность дополнительного уровня для манипуляции элементами и их отображением. БД спецификаций служит основой отображающего ядра и представляет собой некоторый шаблон в виде дерева xml-элементов со специальными управляющими конструкциями, описывающими вершины основной БД. В

первом приближении БД спецификаций, используемую на стороне сервера, можно считать некоторой альтернативой для манипуляции способом отображения вершин по отношению к xsl-шаблону, который может быть применён как на стороне клиента, так и на стороне сервера. БД спецификаций может быть по-прежнему дополнена применением xsl-шаблона путём определения ссылки на файл в адресной строке. БД спецификаций позволяет изначально формировать фрагмент БД в нужном виде, задавая форматирование вершин БД на уровне xml-элементов. В сочетании с xsl-шаблоном это позволяет формировать на основе данных из БД электронные документы произвольного вида.

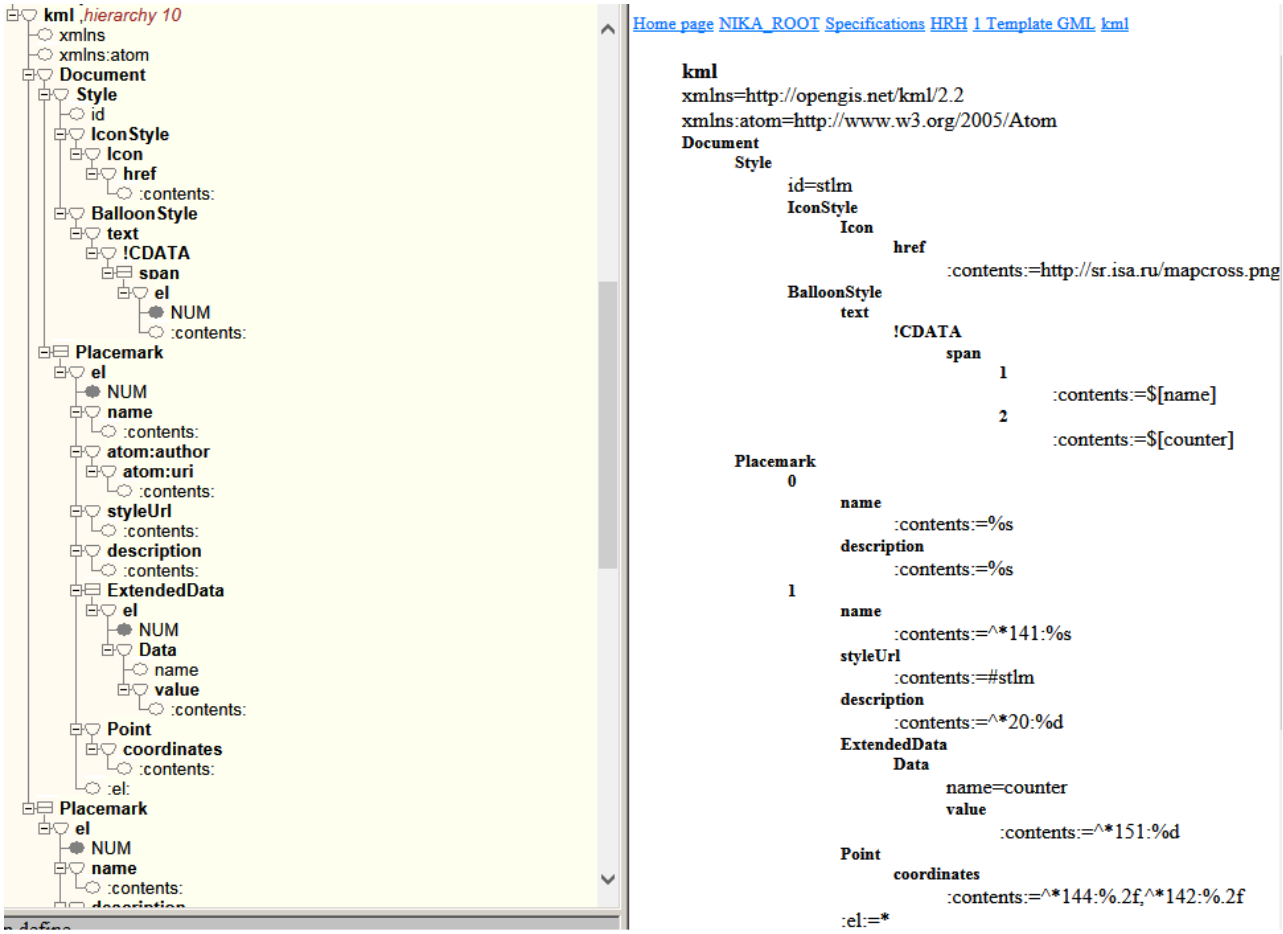


Рис.4.13. Пример фрагмента БД спецификаций

Функционирование системы в режиме с использованием БД спецификаций происходит следующим образом. При задании в адресной строке управляющего ключа xslfo отображающее ядро использует БД спецификаций. При этом в адресной строке задаётся номер спецификации в БД.

Отображающее ядро осуществляет обход элементов в соответствующем дереве элементов, определяющих данную спецификацию. Управляющие конструкции задают вершины основной БД, которые участвуют в отображении. Некоторые управляющие конструкции описаны в следующей таблице.

Таблица 4.4

Управляющие символы БД спецификаций

Обозначение	Отображение	Примечание
^&dn	ключ вершины с dod-номером dn	dn обозначает dod-номер
^*dn	значение вершины с dod-номером dn	—
^^dn	координата вершины с dod-номером dn	используется в гипертекстовой ссылке

Дерево элементов, определяющих данную спецификацию, можно рассматривать как объекты форматирования вершин БД, заданных с помощью управляющих конструкций. Пример спецификации FO (formatting objects) был приведён в пункте, посвящённому спецификации в формате географического языка разметки. В этом примере эта спецификация отображает данные в формате географического языка разметки с помощью БД спецификаций. На рис.4.13 в окне обозревателя приводится некоторый пример определения спецификации с помощью БД спецификаций. В правом фрейме приводится описание данных для отображения в формате географического языка разметки, а в левом фрейме соответствующие данные, определяющие шаблон спецификации, по которому будут отображаться данные из основной БД. Пример результата отображения показан на рис.4.14 в xml-формате.

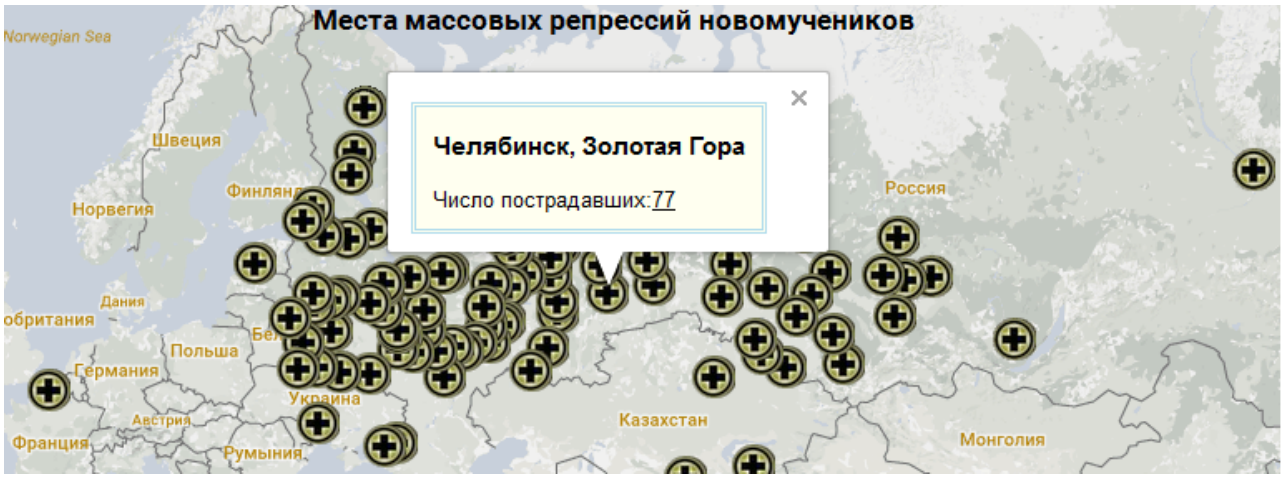


Рис.4.15. Отображение географических данных из БД на карте

```
- <kml>
  - <Document>
    - <Style id="stlm">
      - <IconStyle>
        - <Icon>
            <href>http://sr.isa.ru/mapcross.png?expirecount=10</href>
          </Icon>
        </IconStyle>
      - <BalloonStyle>
          <text> <span>$[name]</span> <span>$[counter]</span> </text>
        </BalloonStyle>
      </Style>
    - <Placemark>
        <name>%s</name>
        <description>%s</description>
      </Placemark>
    - <Placemark>
        <name>Алтайский край, г.Барнаул</name>
        <styleUrl>#stlm</styleUrl>
        <description>:%d</description>
      - <ExtendedData>
        - <Data name="counter">
            <value>107</value>
          </Data>
        </ExtendedData>
      - <Point>
          <coordinates>66.25,55.65</coordinates>
        </Point>
      </Placemark>
    - <Placemark>
        <name>Алтайский край, г.Бийск</name>
```

Рис.4.14. Пример фрагмента отображения вершин БД НИКА с помощью БД спецификаций

Данный xml-документ отображает картографический сервис на карте (рис.4.15).

4.4.4. Пример отображения биографической справки в виде версии для печати с использованием БД спецификаций

На рисунках 4.16, 4.17 даётся пример стилевого шаблона и пример его применения — отображение биографической справки в виде версии для печати. Посредством опции xslfo задействуется БД спецификаций. Специальная управляющая конструкция в результирующем дереве элементов отображает вершину основной БД в виде xml-документа, содержащую подчинённую данной вершине иерархию вершин из исходной иерархии в заданном порядке.

Результирующая иерархия вершин из основной БД

```
- <xsl:template match="/dcmt">
  - <html>
    - <head>
        <link rel="stylesheet" type="text/css" href="/nika_css/nmprint.css"/>
        <style> .vertClass { text-align: justify; } </style>
      </head>
      <xsl:apply-templates select="/dcmt/cnt[1]"/>
    </html>
  </xsl:template>
- <xsl:template match="/dcmt/cnt[1]">
  - <body class="pageMargins">
    - <b>
        <xsl:value-of select="*[@key='true']"/>
      </b>
      <!-- base level key -->
      <xsl:apply-templates select="_ФИО"/>
    </body>
  </xsl:template>
- <xsl:template match="_ФИО">
    <xsl:apply-templates select="/dcmt/cnt[1]/_Фотографии/_el[@key='1']" mode="FIO"/>
    <xsl:apply-templates select="_l"/>
  - <xsl:if test="/dcmt/cnt[1]/_Сан_ЦеркСлужение">
    - <font face="Calibri">
      - <b>
          <xsl:text>, </xsl:text>
          <xsl:value-of select="normalize-space(../_Сан_ЦеркСлужение)"/>
        </b>
      </font>
```

Рис.4.16. Фрагмент xsl-шаблона nm.dela.delo.xsl

определяется в БД спецификаций в управляющей конструкции посредством указания имён вершин и их подчинённости из схемы описания основной БД. Порядок следования вершин на одном уровне иерархии в БД спецификаций в управляющей конструкции может отличаться от порядка следования вершин в основной БД. Результирующий xml-документ содержит иерархию элементов с именами, заданными в БД спецификаций, и значения для терминальных вершин из основной БД. Такой подход позволяет отображать в xml-документ только те вершины, которые должны присутствовать в документе при его визуализации, а не всё поддерево. Стили представления xml-элементов на

экране могут быть заданы посредством xsl-шаблона. XSLT-преобразование может быть применено как на стороне клиента (см. п.4.4.3), так и на стороне

ЧЕРНЯЕВА Татьяна Михайловна

1873 — родился в населённом пункте: Москва

Из дворян, дочь известного генерала — героя освобождения Балкан в 1876-77гг. Имела сестру-близнеца Надежду. Окончила гимназию в Москве. Затем семья переселилась в Петербург. В начале 1-ой Мировой войны окончила курсы сестер милосердия и всю войну провела на фронте, работая в санитарном поезде

Служение. Петроград

После революции вступила в одну из многочисленных религиозных общин

1922 — Петроград, ул.Кирочная, 48/13 — 5/12

До войны 1914г. — сестра милосердия Свято-Троицкой общины. На момент ареста — без определенных занятий. На жизнь зарабатывала уроками музыки

1922 — арестован. Петроград

1922, 15 сентября — осужден. Петроградский губотдел ОГПУ за "антисоветская деятельность". Приговор — 3 года высылки в Тюменскую губ..

"Постановлением заседания Комиссии НКВД по административным высылкам от 27.12.1922г. постановление ПГО утвердить"

дата кончины. Причина смерти — умерла в ссылке. Место смерти — Тюмень.

Была сослана в Тюмень, где вскоре умерла от паратифа на руках сестры-близнеца Надежды, последовавшей за ней в ссылку

** *Архив УФСБ по Санкт-Петербургу и Ленинградской обл. Д.П-89251.*

Синодик гонимых, умученных, в узах невинно пострадавших православных священно-церковнослужителей и мирян Санкт-Петербургской епархии: XX столетие. СПб., 1999.С.117.

Синодик гонимых, умученных, в узах невинно пострадавших православных священно-церковнослужителей и мирян Санкт-Петербургской епархии. XX столетие. СПб., 2002.С.252.

Санкт-Петербургский мартиролог. СПб.: Изд-во "Миръ", "Общество святителя Василия Великого", 2002. 416с.С.252.

Щелкачев Владимир Николаевич

Андреевы Анна Федоровна и Мария Федоровна (воспитанницы)

Рис.4.17. Динамически созданный гипертекстовой системой СУБД НИКА xml-документ со стилями nm.dela.delo.xsl

сервера. Если в качестве клиента используется обозреватель интернет, то результатом XSLT-преобразования на стороне клиента служит html-документ. Использование XSLT-преобразования на стороне сервера позволяет получать документы различных форматов, в частности html и pdf. Для выполнения преобразования в pdf-документ необходимо использовать соответствующие

программы или библиотеки с поддержкой объектов форматирования XSL FO на стороне сервера, т.к. обозреватели поддерживают преобразование только в html с использованием css-стилей.

4.4.5. Построение отображения многоуровневого классификатора в виде гипертекста и версии для печати с использованием БД спецификаций

Приведённые примеры показывают широту и гибкость механизма ядра отображения на основе БД спецификаций. Вышеизложенные возможности отображения объектов БД посредством БД спецификаций можно применить к рассматриваемому в данной работе алфавитному классификатору на основе ПДС, которое строится также в рамках используемой технологии на базе СУБД НИКА. Значит все рассмотренные механизмы БД спецификаций применимы при отображении ПДС в виде многоуровневого алфавитного классификатора для заданного поля. Следующая глава посвящена рассмотрению примеров алфавитных указателей по различным текстовым полям БД “За Христа пострадавший”. В этом пункте ниже перечислены практические рекомендации, которые должны предъявляться к алфавитному классификатору на основе ПДС с использованием БД спецификаций и xsl-шаблона.

1. Число уровней алфавитного классификатора должно определяться на основе регрессионной зависимости средней длины ключа в ПДС от среднего или максимального количества ключей, входящих в класс. С учётом этой зависимости и заданного максимального числа ключей в классе определяется средняя длина ключа алфавитного указателя. Для задания алфавитного классификатора, соответствующему объекту типа массив с текстовым ключом, назначается спецификация GRP.
2. Число уровней в иерархическом классификаторе может не совпадать со средней длиной ключа в случае, если на каждом уровне классификатора может добавляться более одной буквы. Среднее число букв, добавляемых на каждом уровне, может быть определено из отношения среднего числа уровней в ПДС к средней длине ключа, определённой для групп с одной

вершиной. Число уровней и число букв, добавляемых на уровне, задаётся специальным атрибутом к спецификации GRP.

3. Условием перехода из классификатора на ключевой уровень БД — это значение числа ключей в классе, меньшее заданного порогового значения для данного сочетания букв. Условие проверяется в спецификации GRP для ПДС.
4. Пользовательский интерфейс классификатора на любом уровне должен быть оснащён однобуквенным классификатором верхнего уровня со ссылками на следующие уровни и на ключевой уровень массива основной БД. Можно воспользоваться xsl-шаблоном.
5. Пользовательский интерфейс классификатора на любом уровне должен расширяться двух или трёхбуквенным классификатором на выбранную букву посредством xsl-шаблона.
6. При выборе двух или трёхбуквенного ключа в классификаторе на данное сочетание отображается список всех сочетаний букв до пробела, возможно с указанием числа вершин на данное сочетание посредством xsl-шаблона и ПДС.
7. Сочетание различных уровней классификатора в одном документе должно обеспечиваться некоторым стандартным набором спецификаций и их атрибутов.

Следующий этап развития алфавитного классификатора предусматривает возможность использования диапазонов буквенных сочетаний ключей, поэтому в спецификациях необходимо предусмотреть такую возможность. Пример спецификации, использующей диапазоны — RNG (range).

Выводы по главе

1. Модель ООСУБД НИКА, описанная в главе 4, *очень близка* к модели гипертекста и поэтому является наиболее подходящей для реализации гипертекстовой системы, включающую в себя классификаторы по лексикографическому признаку.

2. В рамках такой системы возникает *двойственность* структуры БД и структуры гипертекстовых документов.
3. Спецификации (методы), составляющие *ядро* системы, реализуют различные способы отображения вершин БД в гипертекстовые документы или элементы документа.
4. Надстройка над ядром позволяет с помощью расширяемого языка таблиц стилей произвести *тонкую настройку* ядра.
5. Такой гипертекстовый интерфейс позволяет *наиболее просто* реализовать классификатор в виде отдельной БД НИКА, к которой будут применены соответствующие спецификации. “Упрощённый” неоптимальный классификатор реализован в виде спецификации GRP, в которой задаются приращение длины ключа на каждом уровне и число уровней в классификаторе. При этом все ключи на определённом уровне имеют одинаковую длину. Для построения оптимального классификатора необходимо использовать префиксное дерево сочетаний (ПДС) с частотами префиксов, построенное для данного ключевого массива. ПДС позволяет задать классификатор с различными длинами префиксов на одном уровне в зависимости от частоты данного префикса.

В пятой главе приводится практическое применение разработанного автором классификатора.

Глава 5. Практическое применение многоуровневого алфавитного классификатора на основе ПДС

В пятой главе приводятся практические примеры применения спецификаций отображения вершин БД гипертекстовой системы ООСУБД НИКА, описанной в четвёртой главе, для построения классификаторов для ключевых массивов. Данные отображаются в виде HTML/XML-документов. При реализации классификатора автором был использован опыт, накопленный профессором Емельяновым с сотрудниками при разработке языка OOML, подобного SGML, для СУБД НИКА. OOML служит прототипом для SGML. OOML был реализован в системе МАГИС, которая является документным интерфейсом для СУБД НИКА. В этом смысле реализация гипертекстовой системы для СУБД НИКА с использованием HTML/XML подобна реализации языка OOML, но только с добавлением гипертекстовых ссылок. Существует также несколько публикаций, посвящённых применению гипертекстовой системы для ООСУБД НИКА (Емельянов,Садовский,1997; Чернозуб,Емельянов,Тищенко,2012; Воробьёв,Емельянов,Тищенко,97-07-90055; Емельянов,2000).

5.1. Спецификация RNG для разбиения текущего уровня на диапазоны ключей, соответствующие группам ключей массива одинакового размера

Равномерное распределение текстовых ключей по буквенным сочетаниям является теоретическим. На практике группы с разными ключами будут отличаться по числу ключей. При разбиении массива ключей на группы одинакового размера как в случае равномерного распределения каждой группе в общем случае соответствует диапазон ключей, а не единственный уникальный ключ группы как в случае разбиения по ключам посредством префиксного дерева сочетаний (ПДС).

5.1.1. Спецификация RNG для задания диапазонов ключей в массиве

Спецификация RNG (range) присваивается элементу массива — структуре. Отображает в виде диапазонов ссылки на соответствующие ключи указанного массива. По умолчанию отображает только нижнюю границу диапазона, т.е. ключ (или первые буквы этого ключа), следующего за последним ключом текущего диапазона ключей. Посредством атрибута 238 задается число отображаемых букв ключа [1;255]. Если задать значение атрибута 238 равным нулю, то отображается диапазон значений, соответствующий ключам, следующим по порядку за последним ключом текущего диапазона. Длина ключей, обозначающих диапазон, вычисляется следующим образом. Первый ключ диапазона сравнивается с последним текущего диапазона, а последний ключ диапазона сравнивается с первым, следующим за диапазоном, для которого определяются ключи. Длина ключа в обозначении диапазона берется равной числу одинаковых начальных символов в сравниваемых ключах с добавлением единицы так, чтобы ключи отличались от предыдущего и последующего соответственно. Например, для поля “МестоСлужения” последний ключ текущего диапазона: (1) “Таврическая губ., Бердянский у., с. Юрьевка”, первый ключ следующего диапазона: (2) “Таврическая губ., Днепровский у., с.Б.Благовещенка, Покровская церковь”, последний ключ следующего диапазона: (3) “Таврическая губ., г.Алушта, храм св. Феодора Стратилата”, первый ключ диапазона, следующего за диапазоном, для которого определяется длина ключей: (4) “Таврическая губ., г.Бердянск, Вознесенский собор”. В результате сравнений (1) и (2) ключей, (2) и (3) ключей, а также (3) и (4) ключей получаем следующее обозначение для ссылки на диапазон: “Таврическая губ., Д… — Таврическая губ., г.А…”. Диапазон печатается как ссылка “next” на следующую группу ключей массива. Спецификация может быть распространена на несколько диапазонов.

5.1.2. Описание атрибутов спецификации RNG

Свойства спецификации RNG могут быть заданы с помощью атрибута — вершины 238 в описании данных. Четырехбайтное значение атрибута содержит следующие характеристики.

0 байт. Задаёт длину ключа при отображении границы диапазона.

1 байт. Задаёт число диапазонов ключей, следующих за текущим диапазоном, отображаемых спецификацией.

2 байт. Задаёт число предыдущих диапазонов ключей, включая текущий диапазон.

3 байт. Определяет следующие состояния спецификации RNG.

0 бит. Отображать нижние и верхние границы диапазонов.

1 бит. Отображать текущий диапазон в списке диапазонов.

2 бит. 1-й байт задает минимальную длину ключа границы диапазона.

3 бит. Печать списка диапазонов в конце ключевого уровня массива. В этом случае фрагмент БД может содержать число вершин, меньшее заданного числа вершин в группе.

4 бит. Печать первого и последнего диапазонов в списке диапазонов для данного массива текстовых ключей.

Необходимо отметить особенности работы спецификации RNG в сочетании со спецификацией LVN (см.Прил.В), с помощью которой динамически задается число отображаемых уровней иерархии БД. При отображении списка или иерархии на глубину 1 в списке диапазонов указываются обе границы для каждого диапазона. При задании глубины иерархии, большей 1, список диапазонов содержит только нижние границы. В настоящей версии спецификации расчет диапазонов производится только с помощью ключевого уровня массива без учета отображаемой иерархии. Таким образом, список диапазонов содержит ссылки на вершины, рассчитанные для иерархии глубины 1. Ссылка на диапазон, следующий непосредственно за

текущим диапазоном, ссылается на вершину, которая следует непосредственно за последней вершиной текущего диапазона. Далее предполагается оптимизировать расчет диапазонов посредством префиксного дерева сочетаний (ПДС).

5.2. Спецификация GRP для группирования по лексикографическому признаку

Основой классификатора в данной работе служит ПДС. Более простой вариант классификатора без ПДС не даёт возможности определить числа вершин на данное сочетание, т.к. эти статистики записываются в ПДС, а прямой подсчёт вершин на данное сочетание замедлит выполнение программы. По ПДС также легко определяются сочетания, которые не изменяют частоты появления вершин при добавлении очередной буквы или нескольких букв к данному сочетанию. Такие сочетания могут быть определены путём сравнения текущей и следующей вершин на ключевом уровне массива. Содержащиеся в ПДС частоты также позволяют построить работу с диапазонами ключей, а не только с отдельными уникальными ключами группы. Регрессионная зависимость средней длины ключа от числа вершин в группе (среднего или максимального) может дать среднюю длину ключа $M_к$ в классификаторе, если задать среднее или максимальное число ключей в классе. Для разбиения на уровни можно воспользоваться регрессионной зависимостью среднего числа уровней m_γ в ПДС от среднего или максимального числа ключей в классе. Отношение характеристик $m_к/m_\gamma$ даёт среднее число букв на уровне.

Спецификация GRP определяет классификатор в виде буквенных сочетаний. Атрибут 210 задаёт необходимые параметры классификатора — число уровней и число букв, добавляемых на каждом уровне, а также параметр, связанный с использованием ПДС. Методом, который приведен в статье (Емельянов,Тищенко,2010), можно создавать различные алфавитные указатели для массивов с текстовыми ключами в зависимости от длины (числа букв) ключей в алфавитном указателе и числа уровней. Рассмотрим реальный пример

индекса по полю ФИО с числом вершин на ключевом уровне 32 127. В этом случае имеется следующий результат. Обозначения: n – максимальное число ключей в классе; M_n – среднее значение ключей в классе в алфавитном указателе; $M_к$ – среднее значение длины буквенного ключа.

Таблица 5.1

Средние значения числа ключей в классе и соответствующие длины ключей для ФИО$^{32\ 127}$

n	10	20	30	40	50	60	70	80	90	100
M_n	4,848	9,284	13,909	17,990	22,433	26,839	30,099	34,194	38,436	42,352
$M_к$	6,614	5,628	5,054	4,722	4,422	4,188	4,041	3,895	3,758	3,657

Таким образом, в случае максимального значения 20 ключей в классе длину ключа в алфавитном указателе следует брать 5-6 букв со средним значением 9-10 ключей в классе. Если брать ~20 ключей в классе в качестве среднего значения, то длину ключа алфавитного указателя следует брать 4-5 букв (в среднем). Максимальное значения числа ключей в классе для этой длины ключа будет порядка 50.

Для групп из 20 вершин отличие между величинами средней длины ключа $m_к$ и средним числом уровней m_γ является очень небольшим и не превышает одной буквы, а если брать 20 вершин как среднее значение, то указанная разница не превышает 0,5 буквы (в среднем). Таким образом, в этом случае (и в большинстве случаев для поля ФИО) для числа вершин в группе более чем 20 эти величины можно считать одинаковыми для указанного общего числа вершин на уровне.

Для сравнения величин $M_к$ и M_γ ниже приводится таблица со значениями для этих величин в случае поля ФИО$^{32\ 127}$. Обозначения: n – максимальное ключей в классе; M_γ – среднее значение числа уровней в алфавитном указателе; $M_к$ – среднее значение длины буквенного ключа; $\Delta m = M_к - M_\gamma$.

Таб.5.2

Сравнительна таблица числа m_γ и длины ключа $m_к$ для ФИО$^{32\ 127}$

n	10	20	30	40	50	60	70	80	90	100
M_γ	5,523	4,912	4,533	4,273	4,066	3,901	3,786	3,678	3,572	3,504

$M_к$	6,614	5,628	5,054	4,722	4,422	4,188	4,041	3,895	3,758	3,657
Δm	1,091	0,716	0,521	0,449	0,356	0,287	0,255	0,217	0,186	0,153
$M_к$/$M_γ$	1,198	1,145	1,115	1,105	1,088	1,074	1,067	1,059	1,052	1,043

Рассматривая величину длины ключа $M_к$, следует учитывать также значение величины числа уровней в алфавитном указателе $M_γ$: $1 \le M_γ \le M_к$. При совпадении этих величин число уровней в многоуровневом алфавитном указателе совпадает с длиной ключа $M_к$. Каждый новый уровень образует свой алфавитный указатель, начинающийся на заданное сочетание букв. В идеальном случае, т.е. при наличии всех слов на все буквенные сочетания, число уровней тождественно совпадает с длиной ключа $M_γ \equiv M_к$ при любом значении длины ключа. На практике это не выполняется и $M_γ \le M_к$, причем при уменьшении длины ключа (и соответственно увеличении числа вершин в группе) уменьшится разница между $M_к$ и $M_γ$, при $M_к=1$ неравенство переходит в равенство. При различии $\Delta m>1$ алфавитный указатель будет содержать, по крайней мере, 1 уровень. Для буквенных сочетаний с большим числом вершин могут возникать дополнительные уровни алфавитного указателя. Важно отметить, что при таком делении на уровни алфавитные указатели, соответствующие различным начальным сочетаниям букв, будут содержать различное число букв или буквенных сочетаний в силу неравномерности распределения слов по буквенным сочетаниям. ($M_к$ / $M_γ$) – среднее число букв, добавляемое к сочетанию на каждом уровне.

5.3. Спецификация автозаполнения ключевого поля массива СУБД НИКА

По смыслу автозаполнение — это система подсказок, позволяющая пользователю полуавтоматически заполнять определённое поле формы. Автозаполнение (Bast,2006) представляет собой способ заполнения поля формы из предлагаемого списка вариантов слов и/или сочетаний слов, начинающихся с введённой в поле комбинации букв. Спецификация (Тищенко,2013), реализующая автозаполнение для ключевого поля СУБД НИКА, имеет обозначение FKW и атрибут 242.

5.3.1. Формальная модель автозаполнения

Процесс автозаполнения может быть описан посредством формальной модели в виде *абстрактного автомата со случайным входом*[*)]. Запрос на автозаполнение есть некоторое входное слово w автомата H_n'' в алфавите A_α (Тищенко А.В.,2012). Обработать запрос значит вычислить значение функции перехода состояний $t_n''(c_0'',w)=c_{nw}''$, где $w \in R \subset A^*$, $c_{nw}'' \in C_n''$, причём $c_{nw}''=\{wu_j | wu_j \in R \subset A^*, 0<j<n_j<n\}$. c_{nw}'' есть результат последовательного сужения множества слов $c_{w1}'' \supset c_{w2}'' \supset \ldots \supset c_w''$. Индекс n в обозначении состояния c_{nw}'' означает, что берутся первые n наиболее частотных слов из множества c_w''. Предполагается, что всем дугам графа перехода состояний автомата H_n'' со случайным входом приписаны некоторые вероятности перехода из состояния c_i в состояние c_j для дуги (c_i,c_j), $c_i,c_j \in C_n''$. В качестве вероятностей переходов берутся частотные вероятности соответствующих сочетаний при данном состоянии, т.е. если $c_j=t_n''(c_i,u)$, то берётся условная частотная вероятность u при переходе из c_i в c_j: $P_{u|c\text{i}}$. Интерпретацией автомата H, входом которого являются буквы алфавита А, а состояниями входные слова, может служить ввод сочетаний букв в поле ввода, т.е. формирование запроса на автозаполнение. Автомат H_n'', входом которого являются слова естественного языка, реализует переходы с помощью функции t_n'' между упорядоченными множествами слов естественного языка.

Префиксное дерево сочетаний (ПДС) может быть построено на основе множества состояний C автомата H. Сочетания, которые входят в префиксное дерево, выделяются из множества слов, которое ограничено набором слов, составляющих вершины заданного поля БД. В качестве вершин дерева берутся элементы множества C, т.е. буквосочетания естественного языка, но без повторения префикса.

5.3.2. Описание работы спецификации автозаполнения

Спецификация основана на префиксном дереве сочетаний (ПДС), построенном для данного ключевого уровня массива. ПДС содержит частоты встречаемости заданного префикса слова (т.е. начальной последовательности букв слова) на данном ключевом уровне массива. Спецификация автозаполнения FKW (frequent keywords) назначается вершине типа массив. Спецификация может работать в двух режимах: в режиме буквенных сочетаний и в режиме ключевых слов, который реализован во многих известных поисковых системах в интернет. Режим буквенных сочетаний позволяет получать первые m наиболее часто встречающихся сочетаний на ключевом уровне массива. Аналогично режим ключевых слов позволяет получать первые m наиболее часто встречающихся слов на ключевом уровне массива. Значение параметра m задается в атрибуте 242 спецификации. Результирующий список сочетаний или слов упорядочивается в порядке убывания частот встречаемости. Если для заданного сочетания букв существует префикс, меньшей длины (чем это сочетание), для которого получается тот же самый список, то для заданного сочетания и его префикса (префиксов) в качестве результата выдается один и тот же список сочетаний или слов. Например, сочетание “Михайл” для поля “ФИО” в БД “Новомученики и исповедники” дает следующий результат при m=10.

Михайлов	(42)–Александр Викторович, Александр Григорьевич и т.д.
Михайловский	(16)
Михайлова	(15)
Михайленко	(2)
Михайловская	(2)
Михайлина	(1)
Михайловская-Шлодгауэр	(1)

Тот же самый список будет выдан для префикса “Михай”, поскольку других вершин на это сочетание нет (в индексе по полю ФИО). Число слов в списке равно 7, т.е. не превышает m. Числа в скобках обозначают частоты встречаемости и выводятся при задании флага в атрибуте 242 к спецификации. В случае режима сочетаний букв сочетание “Михайл” дает следующий список сочетаний.

Михайлов	(76)
Михайленко	(2)
Михайлина	(1)

Здесь сочетание Михайлов (76) содержит следующие сочетания: Михайлов (42), Михайловский (16), Михайлова (15), Михайловская (2), Михайловская-Шлодгауэр (1). Видно, что список сочетаний 2 содержит префиксы слов, содержащихся в списке 1.

5.4. Описание работы с классификатором и построение оптимального классификатора для поля “ФИО”

С помощью спецификации GRP, задавая определённые значения параметров, в сочетании с другими спецификациями можно получить требуемый вид классификатора.

На рис.5.1 показан пример входной страницы для классификатора по полю “ФИО”. Двухбуквенные сочетания, выделенные в виде гипертекстовых ссылок, позволяют перейти на подчинённые уровни иерархического классификатора или непосредственно на список. Классификатор может быть дополнительно снабжён для удобства гипертекстовой алфавитной панелью, содержащей алфавит и двух/трёхбуквенные сочетания на выбранную букву алфавита. Это позволяет быстро осуществлять навигацию по ключевому уровню массива на искомый ключ. На рис.5,2 представлен второй уровень классификатора на биграмму “Би”. Он состоит из 20 четырёхбуквенных N-грамм. Поле ввода с возможностью автозаполнения (система подсказок) предоставляет альтернативный способ перехода на заданный ключ.

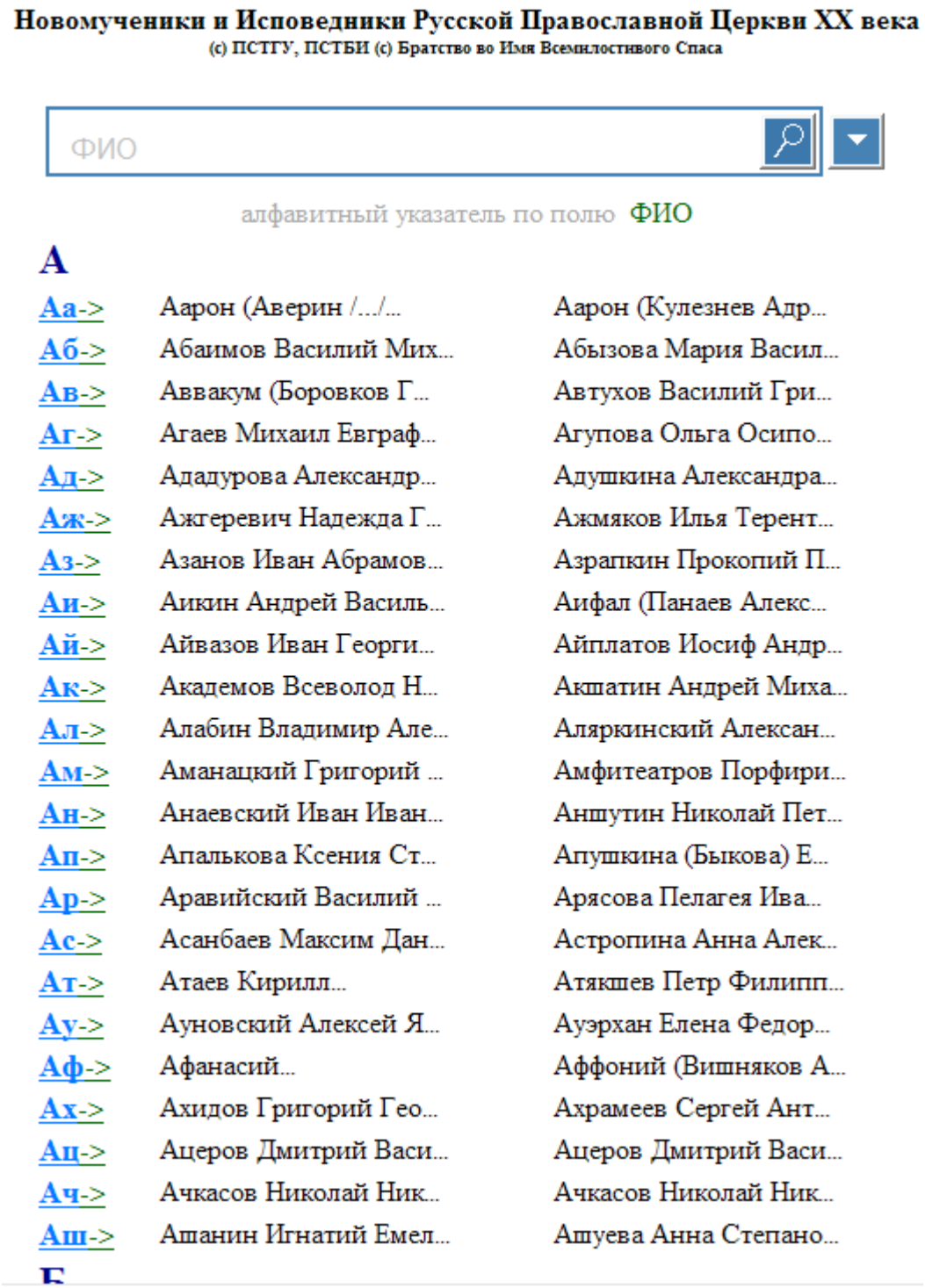


Рис.5.1. Пример страницы входа в классификатор по полю “ФИО”

В обоих случаях (многоуровневый классификатор и система подсказок) может быть использовано построенное для данного поля ПДС, содержащее статистику частот вершин ключевого уровня по сочетаниям. Такая статистика позволяет, например, задавать пороговое значение числа вершин на сочетание,

при котором нужно переходить от иерархического классификатора к списку вершин ключевого уровня на искомое сочетание.

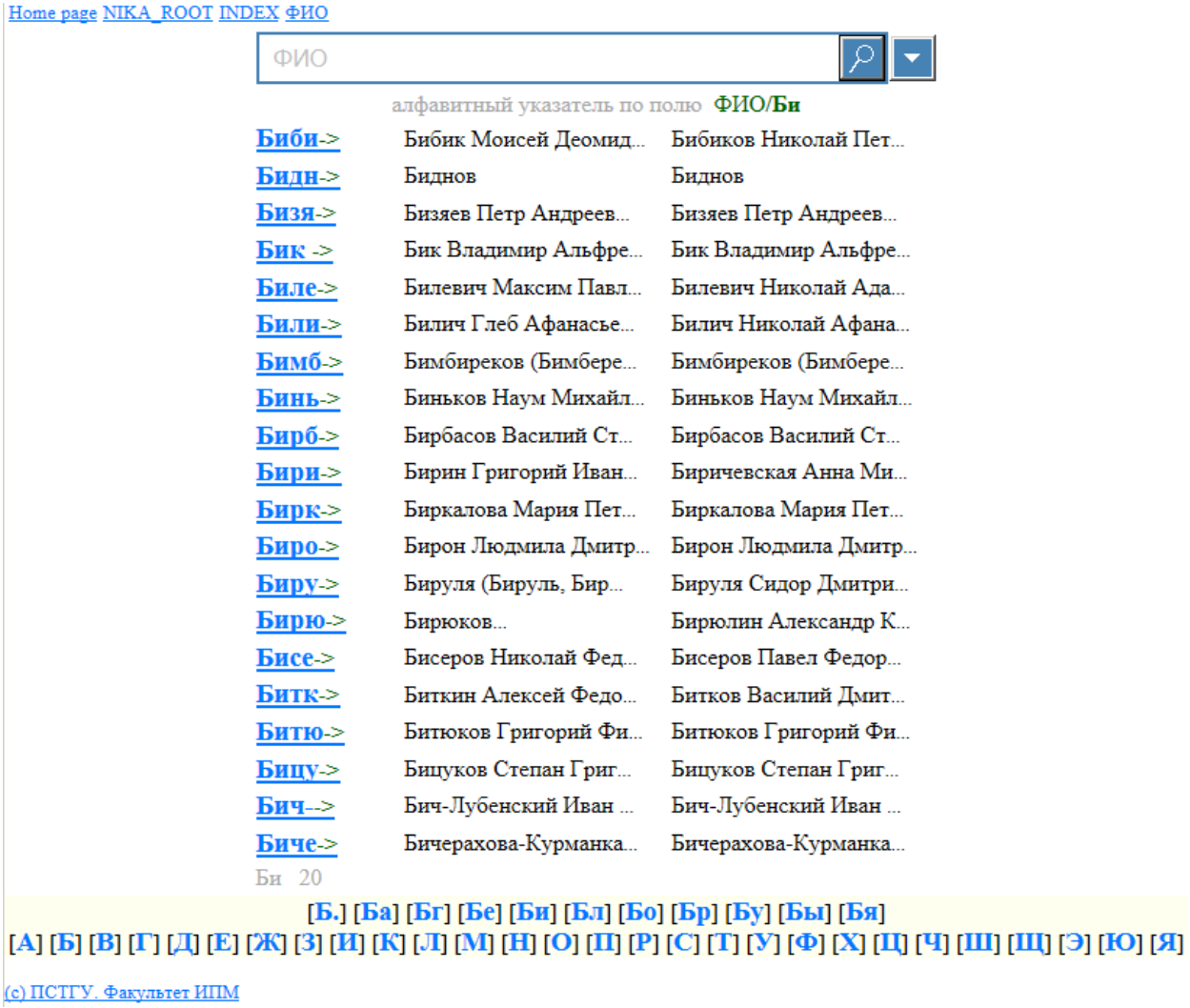


Рис.5.2. Второй уровень классификатора по полю “ФИО” на биграммиу “Би”

Совместно со спецификацией алфавитного классификатора GRP может быть использована спецификация шкала SC (scale). Эта спецификация относится к первому виду отображения данных (Емельянов,1987) в виде последовательности. Назначается вершинам типа массив. Особенностью работы спецификации заключается в том, что она всегда используется совместно со спецификациями иерархия HRH, таблица TAB или с представлением документа по умолчанию, т.е. с последовательностью. Таким образом, в сочетании с HRH или TAB спецификация SC всегда даёт смешанное представление (Емельянов,1987). Например, в БД о новомучениках вершине “ГодРождения” в индексе могут быть назначены следующие спецификации

HRH-3-SC или TAB-SC. Тогда после иерархии или таблицы в конце документа выдаются в виде последовательности ключи массива, заключённые в квадратные скобки $[k_1]…[k_n]$. Каждый ключ представляет собой ссылку на соответствующий элемент массива, позволяющий переходить в нужное место иерархии или таблицы. Атрибут 260 спецификации SC задаёт шаг, с которым выдаются элементы массива, максимальное число выдаваемых элементов. Тип атрибута является бинарным. При выборе соответствующего элемента шкала также выдаётся, начиная с этого элемента, если содержит не все элементы. Если в качестве шага задана -1, то спецификация SC переходит в специальный режим алфавитного указателя. В этом случае спецификация SC вызывает спецификацию GRP, выдавая в конце текущего уровня путь в ПДС и шкалу на заданную начальную N-грамму. В случае пустой начальной N-грамме выдаётся шкала букв алфавита со ссылками на соответствующие вершины ключевого уровня массива. Для начальной биграммы “Ко” для поля ФИО будет выдана шкала со следующими 19 трёхбуквенными N-граммами: “Коб”, “Ков”, “Ког”, “Код”, “Кож”, “Коз”, “Кой”, “Кок”, “Кол”, “Ком”, “Кон”, “Коп”, “Кор”, “Кос”, “Кот”, “Коф”, “Кох”, “Коч”, “Кош”. В шкале присутствуют все начальные триграммы на “Ко”, встречающиеся в ключах массива поля “ФИО”. Со второго уровня классификатора (см. выше рис.5.2) происходит переход на список ФИО на выбранную N-грамму. При выборе в шкале N-граммы происходит переход на вершину ключевого уровня массива, начинающегося на эту N-грамму. В результирующем документе в конце будет выдана соответствующая шкала на выбранную триграмму, например на “Коз” получается 11 четырёхбуквенных N-грамм: “Коза”, “Козе”, “Кози”, “Козл”, “Козм”, “Козн”, “Козо”, “Козу”, “Козы”, “Козь”, “Козю”. Аналогично происходит при выборе N-граммы произвольной длины. Шкала содержит N-граммы, встречающиеся в качестве начальных для вершин ключевого уровня массива, которые длиннее заданной N-граммы на одну букву. Процесс выбора N-грамм в шкале может проходить вплоть до N-граммы, которая совпадает с одной из вершин ключевого уровня.

На рис.5.3 ниже показан список ФИО при выборе N-граммы “Бирю”. Спецификация GRP в сочетании со спецификацией SC добавляет в конце списка шкалу выбора возможных N-грамм на “Бирю” (алфавитную шкалу и шкалу биграмм). В данном случае — это две N-граммы “Бирюк” и “Бирюл”. Для перехода на фамилию Бирюлин можно воспользоваться переходом по N-грамме “Бирюл”.

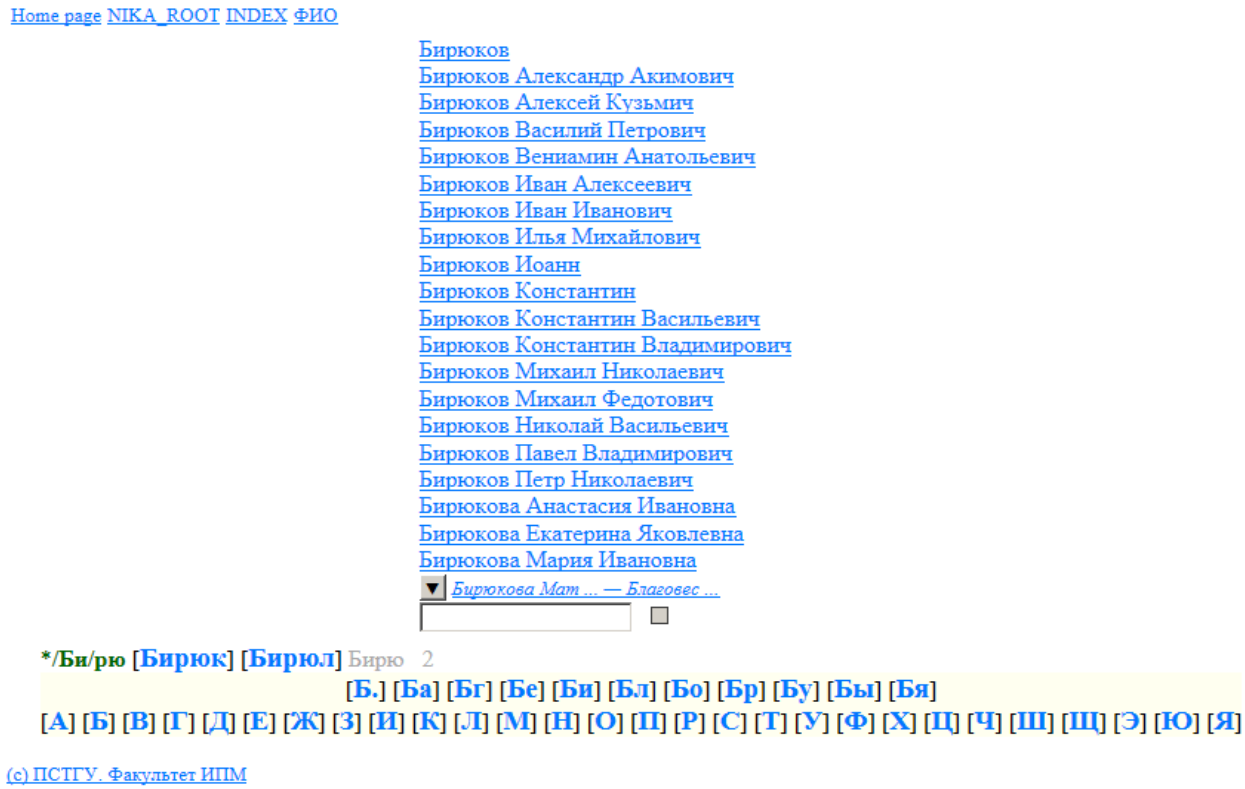


Рис.5.3. Список ФИО на N-грамму “Бирю”

Назначенные спецификации на вершине ФИО — это MVF-MVK-KPN-CNT-MTH-GRP-SC-2 (см. Прил. А). Атрибут 210 спецификации GRP равен gr_lvl=2;gr_alphn=2 (gr_alphn задаёт число букв на каждом уровне), что определяет двухуровневый четырёхбуквенный классификатор. Число вершин в списке в отображённом документе равно 20. Из таб.5.1 для поля ФИО п.5.2 можно определить, что при n=20 среднее число вершин в группе равно M_n=9,284 и средняя длина ключа классификатора равна $M_к$=5,628≈6, т.е. шестибуквенный классификатор. Из таб.5.2 следует, что среднее число уровней классификатора равно $M_γ$=4,912≈5. Это означает, что один из уровней классификатора, например, первый будет добавлять две буквы к ключу, а

остальные уровни по одной букве. В общем случае в этом классификаторе необходимо осуществить 5 переходов при поиске заданной вершине на уровне. Однако с точки зрения оптимальности классификатора в смысле величины $M_к$ при минимизации общего числа операций в классификаторе (см. п.2.5) ближе к оптимальному будет классификатор со средним числом вершин в группе равным около 20. Из таб.5.1,5.2 находим, что при среднем числе вершин в группе M_n=22,433 максимальное число вершин в группе *n*=50, а средняя длина ключа равна $M_к$=4,422≈4, что даёт четырёхбуквенный классификатор, который ближе к оптимальному случаю, чем шестибуквенный. Среднее число уровней равно M_y=4,066≈4, т.е. четырёхуровневый классификатор. С точки зрения удобства использования можно уменьшить число уровней классификатора при том же значении длины ключа, исходя из практических соображений. Максимальное число биграмм не может превысит 1089. В естественном языке реализуется значительно меньшее число биграмм. Например, в случае поля ФИО число начальных биграмм равно 368. Добавление биграммы к исходной N-грамме будет давать число различных сочетаний, начинающихся с этой N-граммы, не превышающее указанное число начальных N-грамм. Например, для поля ФИО буква “К” является наиболее частотной (см.п.2.3.3). В букве “К” биграмма “Ко” содержит наибольшее число четырёхбуквенных N-грамм равное 155. Этот пример приводит к выводу, что добавление биграмм на каждом уровне классификатора не даёт очень большого числа N-грамм на заданное сочетание и не превышает 400 биграмм. В случае поля ФИО можно использовать двухуровневый четырёхбуквенный классификатор. Необходимо отметить, что в среднем в таком классификаторе заданный ключ будет найден за два перехода. В отдельных случаях необходимо воспользоваться шкалой N-грамм в конце списка как, например, в случае N-граммы “Бирюл” (см.рис.5.3). В этих случаях число шагов будет более двух, в случае “Бирюл” — 3 шага. Основной недостаток описанного подхода с использованием среднего значения длины ключа в классификаторе заключается в том, что существуют частотные

N-граммы большой длины. Частотные N-граммы — это N-граммы, для которых число вершин превышает число вершин в списке, помещающимся в один гипертекстовый документ (см. рис.5.4).

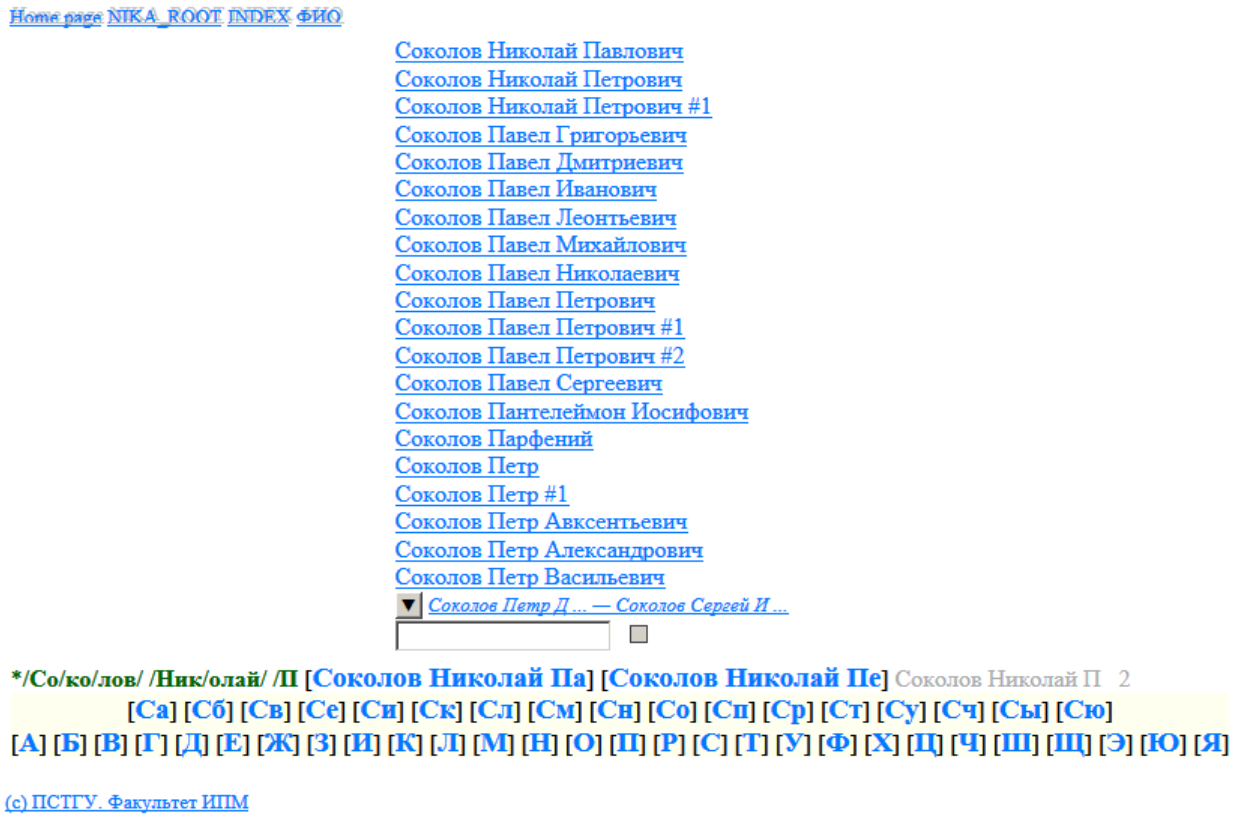


Рис.5.4. N-грамма с большим числом букв

Фамилия и имя Соколов Николай являются часто встречающимися. В БД зарегистрировано 32 пострадавших с такой фамилией и именем. Фамилия Соколов является ещё более частой — 251 пострадавший есть в БД с такой фамилией. Получаем частотные N-граммы из 15 и из 7 букв соответственно. На рис.5.4 видно, что путь в ПДС состоит из 8 шагов (каждый шаг отделяется символом наклонной черты) при переходе на N-грамму “Соколов Николай П”. Проблема, связанная с частотными N-граммами большой длины, может быть решена через добавление дополнительной панели длинных частотных N-грамм и их частот, а также посредством добавления биграмм к исходной N-грамме и перечислением в шкале всех N-грамм, получающихся из исходной путём добавления всех имеющихся биграмм. В данном примере — это N-граммы, представляющие собой фамилии, и их частоты. Частотные характеристики N-

грамм могут быть взяты из ПДС. Таким образом, при выборе буквы в однобуквенной шкале в двух/трёхбуквенной шкале отображаются N-граммы на эту букву и в шкале фамилий — фамилии на первую N-грамму с их частотами. Выбор другой N-граммы отображает соответствующую шкалу фамилий. Наконец, выбор фамилии даёт переход на заданную группу вершин.

Альтернативой к алфавитному классификатору служит поле ввода для перехода по ключу, дополненное системой подсказок. На рис.5.5 показан список ФИО на N-грамму “Сок”.

Рис.5.5. Отображение списка на триграмму ‘'Сок’' (a) с панелью 4-буквенных N-грамм, (b) с панелью фамилий

Переключение между панелью N-граммам (a) и панелью ключевых слов (b) осуществляется с помощью элемента управления флаговая кнопка. Для реализации такого подхода элементу массива “ФИО” присваивается спецификация SWC: RNG-RUL-UNV-HRH-RN-SWC-16 (см. Прил. A,B). Эта спецификация переключает между двумя групповыми спецификациями, присвоенными массиву ФИО, с псевдонимами, указанными в атрибуте 250: 2;;/spc_fkwords;keywords. Здесь разделителем является точка с запятой. Числовое значение задаёт тип форматирования элемента управления, пустое значение — это псевдоним первой групповой спецификации (спецификация по

умолчанию из атрибута 200 под системным номером вершины), /spc_fkwords — псевдоним второй групповой спецификации под вершиной 302, keywords — поясняющая строка. Для панели N-грамм (рис.5.5,a) — это спецификации, заданные в виде групповой спецификации по умолчанию. Для ключевых слов (рис.5.5,b) — это групповая спецификация в данном случае, с псевдонимом /spc_fkwords, которая определяется посредством спецификации FKW (см.п.5.3.2.). В панели ключевых слов отображены первые m наиболее частотных фамилий с соответствующими частотами для m=5 на заданную N-грамму. Наиболее частотной является фамилия — Соколов*). Данная статистика берётся из ПДС. Затем весь список ключевых слов на данную N-грамму упорядочивается по убыванию частот. Далее из списка берутся первые m слов.

На рисунке 2.7 рассматривается пример фрагмента ПДС для индекса по полю ФИО$^{34\,657}$ в БД по репрессированным. В верхнем индексе указано общее количество имён в индексе. На рисунке 6 в квадратных скобках указаны соответствующие частоты встречаемости префиксов. Первая величина — частота встречаемости относительно подчинённого уровня, вторая величина — частота встречаемости относительно поддерева.

В результате расчёта оптимального классификатора для поля ФИО$^{34\,657}$ получается минимум функционала общего числа операций $S_{оп}^{*}$=505 080 при максимальном числе ключей в классе n^{*}=176, числе ключей в группе n_g^{*}=20 и средней длине ключа класса k^{*}=3,106. Эти параметры задаются в качестве входных к спецификации, реализующей алфавитный классификатор в гипертекстовой системе СУБД НИКА. В результате на основе ПДС формируется двухуровневый классификатор для поля ФИО$^{34\,657}$. Со второго уровня классификатора, фрагмент которого показан на рисунке 7, осуществляется переход на ключевой уровень исходного массива на искомый префикс при выборе соответствующей гипертекстовой ссылки. Вторая цифра в

*) Частота начальной N-граммы Соколов — 233 отличается от вышеупомянутой 251, т.к. эти значения взяты в разных версиях БД.

квадратных скобках — это частота встречаемости данного префикса, которая не может превышать n^*=176. Фрагмент классификатора на рисунке 7 состоит из первой группы на префикс “А” с числом префиксов n_g^*=20. Общее число префиксов классификатора составляет 1519.

Рис.5.6. Фрагмент классификатора для поля ФИО[34 657]

Второй уровень классификатора для поля ФИО[34 657] надстраивается ещё одним уровнем, состоящим из букв алфавита с гипертекстовыми ссылками на соответствующие префиксы второго уровня (поскольку число сочетаний на втором уровне на каждую букву, кроме “К”, не превышает n^*). С расчётного уровня классификатора на рис.5.6 при выборе префикса осуществляется переход на группы ключей массива на данный префикс.

При больших объёмах массива число уровней классификатора может быть большим и в общем случае расчёт числа операций производится по формуле (25).

Построенный оптимальный классификатор позволяет находить ключ в среднем за 8 переходов по ссылкам: 4 — по префиксным ссылкам при выборе класса ключей (1 переход с однобуквенного уровня классификатора на основной и 3 при переходе на искомый префикс) и 4 — по ссылкам при переходе по группам на уровне ключевого массива, в худшем случае — за 16 переходов (на последнюю группу на префикс “Кра”), в лучшем случае — за 1 переход, например, на первую группу для букв “Щ”, ”Э”, ”Ю”, в которых все ключи вмещается в один класс. Такой классификатор оптимизирует доступ к массиву ключей, отображаемого в виде групп, заданного размера. Очевидно, что такой классификатор качественно превосходит последовательный список, разбитый на группы, с возможностью перехода на следующую группу. В отличии от равномерного классификатора с диапазонами ключей такой классификатор даёт возможность определения отсутствия ключей на заданный префикс на префиксных уровнях, не требуя перехода к самим ключам массива на данный префикс. Наконец, среди всех возможных классификаторов с различными параметрами такой классификатор оптимизирован по общему числу операций. В отношении к поиску посредством поля ввода является альтернативным методом и удобен для использования на мобильных устройствах, не оснащённых клавиатурой.

Алфавитные классификаторы служат одним из основных элементов справочно-информационной системе по репрессированным и доступны для использования в свободном режиме по адресу http://martyrs.pstbi.ru.

Выводы по главе

1. В главе 5 описано отдельное применение в виде спецификации автозаполнения, которую можно рассматривать как “динамический” классификатор.
2. Описаны применения в виде равномерного классификатора (спецификация RNG) и неоптимального классификатора (спецификация GRP).
3. Основной результат заключается в том, что в отличии от классификаторов RNG и GRP приложение, описывающее оптимальный классификатор на основе ПДС для ключевого массива ФИО$^{34\,657}$ позволяет пользователю выполнить переход на искомый ключ в среднем за минимальное количество шагов, что подтверждено опытом практического применения. Такая организация интерактивного интерфейса даёт возможность оптимальным образом осуществлять навигацию и просмотр ключей в массиве.

Заключение

Решением поставленных задач является оптимальный классификатор по лексикографическому признаку в смысле минимального в среднем числа переходов к искомому ключу применительно к ключевым массивам сложноструктурированной БД. Для построения оптимального классификатора были решены следующие задачи: проанализирована неравномерность распределения ключей по префиксам методом модельных распределений, исследован вид случайных величин длины префикса класса и числа ключей в классе посредством соответствующих функций плотности, исследована зависимость средней длины ключа префикса класса от максимального числа ключей в классе на основе регрессионной модели, среди возможных классификаторов выбран оптимальный посредством минимизации общего числа операций в классификаторе и реализована программа, выполняющая указанные задачи для заданного ключевого массива. Главным результатом диссертации являются методы построения классификаторов по лексикографическому признаку. Среди них выделяется метод построения оптимального классификатора. Эти методы позволяют исследовать различные характеристики классификаторов и выделить класс алфавитных классификаторов, к которому применяется метод оптимизации функционала общего числа операций. В результате получены следующие основные результаты.

1. Разработан метод модельных распределений для анализа неравномерности распределения ключей массива по n-граммным префиксам на основе префиксного дерева сочетаний. Для модельных распределений получена средняя длина ключа в виде формулы с использованием энтропии, которая показала, что при практическом применении ключи распределены по буквенным сочетаниям неравномерно на всю длину ключа.

2. Получены функции плотности распределений длины префикса класса $f(\kappa)$ и числа ключей в классе $f(n)$ для алфавитного классификатора путём кумулянтного разложения в ряд Эджворта. Аналитические выражения функции плотности распределения интересны с точки зрения выделения различных типов индексов по полям БД. Например, для индекса ФИО[32 127] при одном ключе в классе это распределение имеет 4 моды в точках 10, 5, 17, 29 в порядке убывания частотных вероятностей. Первые 3 моды соответствуют в среднем имени, фамилии и отчеству.
3. Разработан метод построения классификатора по лексикографическому признаку на основе регрессионной зависимости $\kappa(n)$ средней длины префикса алфавитного классификатора κ от максимального числа ключей на любой префикс n методом ортогональных полиномов Чебышева. Зависимость $\kappa(n)$ позволяет определить среднюю длину ключа классификатора при заданном максимально числе ключей в классе и используется также и в случае оптимального классификатора.
4. Разработан метод построения оптимального классификатора по лексикографическому признаку с использованием префиксного дерева сочетаний на основе оптимизации функционала общего числа операций в классификаторе $S_{on}^{*}=S_{on}(n^{*},n_g^{*})$, что позволяет формализовать интерактивный способ доступа к ключевому массиву. С помощью приведённого в разделе 3.5 алгоритма рассчитывается величина S_{on}^{*} для заданного ключевого массива. Из регрессионной зависимости получается параметр оптимального классификатора средней длины префикса $\kappa^{*}=\kappa(n^{*})$. Полученные характеристики применяются как параметры спецификаций, отображающих классификатор.
5. Разработана с использованием полученных результатов гипертекстовая система на основе ООСУБД НИКА NKWSystem (c) ИСА ФИЦ ИУ РАН, которая функционально состоит из ядра спецификаций отображения объектов БД, и надстройки над ядром в виде расширяемого языка стилей,

для тонкой настройки отображения объектов БД. Одним из применений указанного функционала является отображение объекта типа массив в виде оптимального классификатора по лексикографическому признаку с использованием спецификаций отображения префиксного дерева сочетаний с оптимальными параметрами.

Важно также привести вывод из раздела 2.1, что классификатор по лексикографическому признаку — сжатое по поддеревьям префиксное дерево trie — играет роль интерфейса с пользователем, поэтому в нём существенна временная, а непространственная сложность. Временные показатели классификатора получились быстрее приблизительно в 1,6 раза обычного дерева trie, но медленнее приблизительно в 1,7 раза сжатого по уровням дерева *LC-trie*, которое не применимо в виде интерфейса с пользователем.

Внедрение результатов работы. В качестве практического применения перечисленных результатов можно рассматривать различные внедрения автором диссертации гипертекстовой системы ООСУБД НИКА. В 1996 году она была внедрена в виде информационно-поисковой системы по репрессированным на базе сайта кафедры информатики ПСТГУ, в 1997 году внедрена в виде системы “СУБД про СУБД” на выделенном ресурсе ИСА РАН и зарегистрирована в Государственном регистре баз данных, в 1998 году внедрена в виде информационной системы на основе Ежегодника “Системные исследования” в рамках проектов РФФИ (Емельянов,Садовский,1997), в 2009 году внедрена в виде информационного ресурса редакции журнала “Вопросы философии” для организации полнотекстового доступа (Чернозуб,Емельянов,2012). Основу информационной системы по репрессированным составляет электронная публикация БД “За Христа пострадавшие”, которая доступна по адресу http://martyrs.pstbi.ru (Емельянов,2000; БС,1997; БС,2015). Для приведённых внедрений к диссертации прилагаются *справки о внедрении*. Также получено свидетельство о государственной регистрации программы *“Гипертекстовая система для*

ООСУБД НИКА” (Тищенко и др., 2019). Важным результатом является получение *патента на изобретение* (Арлазаров,Тищенко,2019).

Практическое внедрение результатов позволило проверить теоретические результаты на примере построенных оптимальных классификаторов для конкретных индексов БД, рассчитав оптимальные значения функционала общего числа операций, а также построить более оптимальный алгоритм его расчёта на основе параллельных вычислений.

Перспективы развития классификатора по лексикографическому признаку формулируются в виде следующих направлений развития.

- Выделение типов различных классификаторов на основе функций плотности распределения длины префикса класса, соответствующих различным полям БД, по которым строятся индексы, например, “ФИО“, “Адрес“, “Должность“.
- Применение префиксного дерева для автозаполнения поля ввода формы.
- Развитие пользовательского интерфейса классификатора в направлении, связанном с одновременным использованием различных панелей n-грамм для навигации по префиксам и ключам массива.
- Применение минимаксных методов размещения объектов для составления равночастотных в среднем буквенных последовательностей для параллельной обработки

БИБЛИОГРАФИЯ

1. Алферов, А.П. Основы криптографии / А.П. Алферов, А.Ю. Зубов, А.С. Кузьмин, А.В. Черемушкин — М.: Гелиос АРВ, 2002. — 480 с.

2. Арлазаров, В.Л. Объекты, формы, содержание (от баз данных к базам знаний) / В.Л. Арлазаров, Н.Е. Емельянов // 1-я Международная конференция «Системный анализ и информационные технологии» САИТ-2005 (12 – 16 сентября 2005 г. Переяславль – Залесский, Россия): Труды конференции в 2 т. Том 2. – М.КомКнига, 2005. С. 250 –255.

3. Арлазаров, В.Л. Устройство отыскания информации по ключевым словам / В.Л. Арлазаров, В.А. Тищенко // Патент на изобретение № 2679967 С1 Российская Федерация, 2019. Бюл. № 5.

4. Прот. Воробьёв, В.Н. "За Христа пострадавшие: Гонения на Русскую Православную Церковь 1917-1956: Биографический справочник. Книга первая А-К." / Прот. В.Н. Воробьёв, Г.В. Воронцов, Н.Е. Емельянов и др. М.: Издательство Православного Свято-Тихоновского Богословского Института, Москва, 1997. - 704 с.

5. Прот. Воробьёв, В.Н. "За Христа пострадавшие": Гонения на Русскую Православную Церковь 1917-1956. Биографический справочник. Кн.1 (А). / Прот. В.Н. Воробьёв, Л.А. Головкова, Н.Е. Емельянов и др. М.: Издат. ПСТГУ. 2015. - с.

6. Бериков, В.С. Современные тенденции в кластерном анализе / В.С. Бериков, Г.С. Лбов // Всероссийский конкурсный отбор обзорно-аналитических статей по приоритетному направлению «Информационно-телекоммуникационные системы», 2008. — 26 с.

7. Богачева, А.Н. "Семантическая модель документа" / А.Н. Богачева, Н.Е. Емельянов // Системные исследования. Ежегодник 2001, "Едиториал УРСС", М.2003, С. 360-375.

8. Большев, Л.Н. Таблицы прикладной статистики. / Л.Н. Большев, Н.В.

Смирнов. М.: Наука, 1983. С.416.

9. Информационная система по истории христианства в России в XX веке: Отчет о НИР № 97-07-90055 (Российский фонд фундаментальных исследований) / Прот. В.Н. Воробьев и др. — ПСТБИ — 1997.

10. Годунов, А.Н. СУБД НИКА / А.Н. Годунов, Н.Е. Емельянов, А.Н. Косьмынин, В.А. Солдатов // Системы управления базами данных и знаний. М.: "Финансы и статистика", 1991. С.208-249.

11. Дрейпер, Н. Прикладной регрессионный анализ. В 2-х кн. / Н. Дрейпер, Г. Смит — М.: Финансы и статистика, 1986.

12. Дрейпер, Н. Прикладной регрессионный анализ. / Н. Дрейпер, Г. Смит — М.: Диалектика, 2007. 912 с.

13. Емельянов, Н.Е. Некоторые факты, установленные по данным информационной системы "Новомученики и исповедники Русской православной Церкви XX века" / Н.Е. Емельянов // Материалы Юбилейной X Ежегодной богословской конференции ПСТГУ (20 - 22 января 2000г.) — Москва, 2000г. — С.348-353.

14. Емельянов, Н.Е. Теоретический анализ документного интерфейса / Н.Е. Емельянов. М., препринт ВНИИСИ, 1987, 40 с.

15. Развитие теории, методов и средств индексации, поиска и отображения объектов в сложных структурах и документах: Отчет о НИР № 96-01-01840 (Российский фонд фундаментальных исследований) / Н.Е. Емельянов и др. — ИСА РАН — 1996.

16. Емельянов, Н.Е. СУБД НИКА и гипертекстовые информационные системы в INTERNET / Н.Е. Емельянов, И.В. Муханов, В.А. Тищенко // Телематика-96. —1996.

17. Емельянов, Н.Е. WWW-сервер на основе СУБД НИКА / Н.Е. Емельянов, И.В. Муханов, В.А. Тищенко // 3-я Международная конференция «Развитие и применение открытых систем»: сб. тр., издание Международного центра научно-технической информации. — Москва,

1996.

18. Емельянов, Н.Е. Построение web-сервера для периодических изданий на материале ежегодника “Системные исследования” / Н.Е. Емельянов, В.Н. Садовский, В.А. Тищенко, И.Б. Чернышева // Системные исследования. Методологические проблемы. Ежегодник 1997, изд-во "Эдиториал УРСС”, 1997, с. 313-323.

19. Емельянов, Н.Е. Использование СУБД в издательской деятельности / Н.Е. Емельянов, А.В. Соловьев // Материалы VI Международной конференции «Применение новых технологий в образовании» 29 июня — 2 июля 1995 г. Троицк. — С. 123-125.

20. Емельянов, Н.Е. Средство конечного пользователя для генерации документов по базам данных / Н.Е. Емельянов, А.В. Соловьев, Д.В. Соловьев // Сборник трудов Института системного анализа РАН. Под ред. д.т.н., проф. В.Л. Арлазарова и д.т.н., проф. Н.Е. Емельянова — М.: Эдиториал УРСС, — 2000.

21. Емельянов, Н.Е. “Использование баз данных в составе BBS” / Емельянов Н.Е., Тищенко В.А. // материалы конференции “Информационные системы в науке - 95” — Москва, 1995.

22. Емельянов, Н.Е. Методология построения многоуровневого индекса ключевого массива по лексикографическому признаку на основе метода регрессионного анализа на примере СУБД НИКА / Н.Е. Емельянов, В.А. Тищенко // Обработка информационных и графических ресурсов / Сб. трудов ИСА РАН. Т.58. Под ред. чл.-корр. РАН В.Л. Арлазарова — М. 2010. С. 6-17.

23. Емельянов, Н.Е. Методы отображения объектов для построения web-сервера объектно-ориентированной базы данных. Развитие безбумажных технологий в организационных системах / Н.Е. Емельянов, В.А. Тищенко // / Сборник трудов ИСА РАН. Под ред. д.т.н. проф. В.Л. Арлазарова и д.т.н. проф. Н.Е. Емельянова — М.: URSS. 1999. С. 96-109.

24. Емельянов, Н.Е. Представление гипертекста в СУБД НИКА. Технология программирования и хранения данных / Н.Е. Емельянов, Тищенко В.А. // Сб. трудов ИСА РАН. Т.45. Под ред. чл.-корр. РАН В.Л. Арлазарова и д.т.н. проф. Н.Е. Емельянова — М. 2009. С. 17-36.

25. Емельянов, Н.Е. Принципы построения web-сервера на основе объектно-ориентированной базы данных / Н.Е. Емельянов, В.А. Тищенко // Информационные технологии и вычислительные системы. — 1997. — №4. — С.90-99.

26. Еременко, С.И. Принципы организации гипертекста на WWW-сервере ИВИБ / С.И. Еременко // Телематика-96, Санкт-Петербург, Республиканский научный центр компьютерных телекомуникационных сетей высшей школы, 1996, с. 89-90.

27. Замулин, А.В. "Системы программирования баз данных и знаний" / А.В. Замулин — Новосибирск Наука,1990, — с.227.

28. Кендалл, М. Многомерный статистический анализ и временные ряды / М. Кендалл, А. Стьюарт — М.:Наука, 1976, С.736.

29. Ким, Дж.-О. Факторный, дискриминантный и кластерный анализ / Дж.-О. Ким, Ч. У. Мьюллер, У. Р. Клекка и др. — М.:Финансы и статистика, 1989. 215 с.

30. Кнут, Д. Э. Искусство программирования. Том 1. Основные алгоритмы = The Art of Computer Programming. Volume 1. Fundamental Algorithms / Д.Э. Кнут, под ред. С. Г. Тригуб (гл. 1), Ю. Г. Гордиенко (гл. 2) и И. В. Красикова (разд. 2.5 и 2.6). — 3. — Москва: Вильямс, 2002.—Т. 1.—720 с.

31. Кнут, Д. Э. Искусство программирования. Том 3. Сортировка и поиск = The Art of Computer Programming. Volume 3. Sorting and Searching / Д.Э. Кнут, под ред. В. Т. Тертышного (гл. 5) и И. В. Красикова (гл. 6). — 2-е изд. — Москва: Вильямс, 2007. — Т. 3. — 832 с.

32. Кобзарь, А.И. Прикладная математическая статистика. Для инженерных и научных работников / А.И. Кобзарь — М.: Физматлит, 2006. С.816

33. Кокин, А.Г. Сети Петри. Моделирование / А.Г. Кокин — Курган. 2005. С.93.

34. Корн, А. Справочник по математике для научных работников и инженеров / А. Корн, Т. Корн — М.:Наука, 1970, С.720.

35. Котов, В.Е. Сети Петри / В.Е. Котов — М: Наука, 1984. — 160 с.

36. Крамер, Г. Математические методы статистики / Г. Крамер. Пер. с англ. — М.: Мир, 1975. 648 с.

37. Кульбак, С. Теория информации и статистика / С. Кульбак — М.:Наука, 1967, С.408.

38. Лапа, В.Г. Математические основы кибернетики / В.Г. Лапа — Киев:Вища школа, 1974, С.452.

39. Ляшевская, О.Н. Частотный словарь современного русского языка (на материалах Национального корпуса русского языка) / О.Н. Ляшевская, С.А. Шаров — М.: Азбуковник, 2009.

40. Мандель, И.Д. Кластерный анализ / И.Д. Мандель — М.: Финансы и статистика, 1988. С.176.

41. Манжула, В.Г. Нейронные сети Кохонена и нечеткие нейронные сети в интеллектуальном анализе данных / В.Г. Манжула, Д.С. Федяшов // Фундаментальные исследования. – 2011. – № 4. – С. 108-115.

42. Маслов, В.П. О законе Ципфа и ранговых распределениях в лингвистике и семиотике / В.П. Маслов, Т.В. Маслова // Матем.заметки, 2006, том 80, выпуск 5, С.718-732

43. Микони, С.В. О классе, классификации и систематизации / С.В. Микони // Онтология проектирования 2016. N1 (19). С.67-80.

44. Могиленко, А.В. Теория нечётких множеств. Нечёткий регрессионный анализ. / А.В. Могиленко. —Томск: Печат. Мануфактура, 2004. С.61.

45. Назаров, А.О. Модель и метод концептуальной кластеризации объектов, характеризуемых нечеткими параметрами / А.О. Назаров // Фундаментальные исследования. – 2014. – № 9-5. – С. 993-997.

46. Орлов, А.И. Прикладная статистика. Учебник / Орлов, А.И. — М.: Экзамен, 2006. С.671.

47. Орлов, А.И. Методы снижения размерности пространства статистических данных / А.И. Орлов, Е.В. Луценко // Политематический сетевой электронный научный журнал Кубанского государственного аграрного университета. 2016. N119.

48. Соловьев, А.В. Представление технология автоматизированного издания справочников / А.В. Соловьев // Научно-методический сборник тезисов докладов VIII Международной конференции «Информационные технологии в образовании». 3 — 6 ноября 1998 г. М.:МИФИ, — 1998. — С. 93–94.

49. Соловьев, А.В. Четыре концепции печати информации баз данных / А.В. Соловьев // Материалы XI Международной конференции «Применение новых технологий в образовании» 28 июня — 1 июля 2000 г. Троицк. — 2000. — С. 150–151.

50. Тищенко, В.А. Проблемы построения многоуровневого алфавитного классификатора (на примере ключевого уровня массива СУБД НИКА) / А.В. Соловьёв, В.А. Тищенко // Информационные технологии / Сб. трудов ИСА РАН. Т.68. Вып.1. Под ред. чл.-корр. РАН В.Л. Арлазарова - М. 2018. С. 63-73.

51. Таранов, И.С. Использование префиксного дерева для хранения и поиска строк во внешней памяти / И.С. Таранов // Труды ИСП РАН, Т.20, 2011, С.283-295.

52. Тараскина, А.С. Нечеткая кластеризация по модифицированному методу с-средних и ее применение для обработки микрочиповых данных / А.С. Тараскина // Проблемы интеллектуализации и качества систем информатики. Сборник ИСИ СО РАН под ред. чл.-корр. РАЕН В.Н. Касьянова. Новосибирск 2006. С.217-228.

53. Тищенко, А.В. Математические основы информатики. Часть 2. Теория

формальных языков и машина Тьюринга. Лекции для студентов, обучающихся по направлению «Прикладная математика и информатика» (программа подготовки магистров «Количественные методы в финансах и экономике») / А.В. Тищенко – М.: Финансовый университет, кафедра «Математика», 2012. – 62 с.

54. Тищенко, В.А. Выбор оптимального алфавитного классификатора при минимизации общего числа операций / В.А. Тищенко // Информационные технологии. Сб. трудов ИСА РАН. Т.68. Вып.1. Под ред. чл.-корр. РАН В.Л. Арлазарова - М. 2018. С. 54-57.

55. Тищенко, В.А. и др. Гипертекстовая система для ООСУБД НИКА / В.А. Тищенко. Свидетельство о государственной регистрации программы для ЭВМ №2019612352, 18 февраля 2019 г.

56. Тищенко, В.А. Инструментальные средства построения информационных систем / Н.Е. Емельянов, А.С. Богданов, И.В. Муханов, А.В. Соловьев, В.А. Тищенко, С.А. Хабарова, И.В. Щелкачёва // Отчет о НИР № 96-07-89394-в (Российский фонд фундаментальных исследований).

57. Тищенко, В.А. Организация интерактивного доступа к ключевому массиву на основе классификатора по лексикографическому признаку / В.А. Тищенко // Материалы XVIII Международной научно-практической конференции “Advances in Science and Technology”, 31 января 2019, С.108-111.

58. Тищенко, В.А. Применение автозаполнения для перехода по ключевым словам на искомые значения в массиве СУБД НИКА / В.А. Тищенко // материалы XXIII Ежегодной богословской конференции ПСТГУ, т.1, — 2013 — С.325-328.

59. Тищенко, В.А. Применение языка XSL для отображения БД НИКА / В.А. Тищенко // Организационное управление и искусственный интеллект. Сборник трудов ИСА РАН. Под ред. д.т.н. проф. Арлазарова В.Л. и д.т.н.

проф. Емельянова Н.Е. М.: URSS. 2003. С. 149-175.

60. Тищенко, В.А. Реализация функции географического позиционирования с использованием БД НИКА (на примере индекса по местам служений новомучеников и исповедников) / В.А. Тищенко // материалы XXII Ежегодной богословской конференции ПСТГУ, т.2 — М.: Издат. ПСТГУ, 2012г. С.218-223.

61. Чернозуб, С.П. О создании информационной системы «Философия и методология науки в журнале „Вопросы философии“ / Чернозуб С.П., В.И. Тищенко, Н.Е. Емельянов, Д.И. Сергеев, В.А. Тищенко и др. // Системные исследования. Методологические проблемы: Ежегодник 2011--2012. Вып.36/2011—2012. Под ред. чл.-корр. Ю.С. Попкова, д.филос.н В.Н. Садовского, к.филос.н. В.И. Тищенко. М.: URSS, 2012. С.239-247.

62. Abitebool, S. Restructuring hierarchical database objects / S. Abitebool, Hull R. // Theoretical Computer Science — 1988 — v.62 — P.3-38.

63. Andersson, A. Improved Behaviour of Tries by Adaptive Branching / A. Andersson, S. Nilsson // Information Processing Letters — 1993 — №46 — P.295–300.

64. Arthur , D. "How Slow is the k-means Method?" / D. Arthur, S. Vassilvitskii // Proceedings of the 2006 Symposium on Computational Geometry (SoCG). 2006.

65. Askitis, N. B-tries for disk-based string management / N. Askitis, J. Zobel // VLDB J. — 2009 — №18(1) — P.157-179.

66. Askitis, N. Redesigning the string hash table, burst trie, and BST to exploit cache / N. Askitis, J. Zobel // Journal of Experimental Algorithmics (JEA), 15, 2010.

67. Bādoiu, M. Approximate clustering via core-sets / M. Bādoiu, S. Har-Peled, P. Indyk // Proceedings of the thirty-fourth annual ACM symposium on Theory

of Computing. — 2002. — P. 250–257.

68. Bast, H. Type less, find more: fast autocompletion search with a succinct index / H. Bast, I. Weber // Proc. of SIGIR'06 conference. August 6-11, 2006. P. 364-371.

69. Bayer, R. Organization and Maintenance of Large Ordered Indices / R. Bayer, E.H. McCreight // Acta Inf. — 1972 — №1 — P.173-189.

70. Bayer, R. Prefix B-Trees / R. Bayer, K. Unterauer // ACM Trans. Database Syst. — 1977 — №2(1) — P.11-26.

71. Bazoobandi, H.R. A Compact In-Memory Dictionary for RDF Data / H.R. Bazoobandi, S. Rooij, J. Urbani, A. Teije, F. Harmelen, H. Bal // Proceedings of the 12th European Semantic Web Conference on The Semantic Web. Latest Advances and New Domains, May 31-June 04, 2015.

72. Bezdek, J.C, "Pattern Recognition with Fuzzy Objective Function Algorithms" / J.C. Bezdek — New York, Plenum Press, 1981.

73. Bogacheva, A.N. Object Oriented Markup Language and Restructuring Hierarchical Database Objects / A.N. Bogacheva, N.E. Emeljanov, A.P. Romanov // Proceeding ADBIS '95 Proceedings of the Second International Workshop on Advances in Databases and Information Systems. pp. 137-142, June 27 - 30, 1995.

74. Briandais, R. File Searching Using Variable Length Keys / R. Briandais // Proc. AFIPS Western Joint Computer Conference, San Francisco, California, USA, 15, March 1959. P. 295-298.

75. Clement, J. Dynamic sources in information theory: A general analysis of trie structures / J. Clement, P. Flajolet, B. Vallee // Algorithmica — 2001 — №29 (1/2) — P.307–369.

76. Dempster, A.P. "Maximum Likelihood from Incomplete Data via the EM algorithm" / A.P. Dempster, N.M. Laird, D.B. Rubin // Journal of the Royal Statistical Society, Series B — 1977 — vol. 39, 1 — P.1-38.

77. Devroye, L. A Note on the Average Depths in Tries / L. Devroye // SIAM J.

Computing — 1982 — №28, P.367–371.

78. Emelyanov, N. Indexing of Objects in Complex Structures / N. Emelyanov, A. Godunov, A. Romanov // Japan-CIS Symposium on Knowledge Based Software Engineering - JCKBSE-94.

79. Emelyanov, N.E. “Web server on the basis of NIKA DBMS” / N.E. Emelyanov, I.V. Muhanov, V.A. Tishchenko // Proceedings of the third international workshop on “Advances in databases and information systems”, ACM SIGMOD, Moscow, sep. 10-13, 1996, Vol.2, p.58-59.

80. Emelyanov, N.E. Classification of Structured Data Representations / N.E. Emelyanov, A.B. Soloviov, I.V. Schelkacheva // Proccedings of the Third International Workshop on Advances in DB and IS, ASM SIGMOD, ADBIS-96, Moscow, September 10-13, 1996, Vol.2, pp.4-8.

81. Fagin, R. Extendible Hashing – A Fast Access Method for Dynamic Files / R. Fagin, J. Nievergelt, N. Pipinger, H. R. Strong // ACM Trans. Database Syst. — 1979 — №4(3). — P.315–344.

82. Feder, T. Optimal algorithms for approximate clustering / T. Feder, D. Greene // Proceedings of the twentieth annual ACM symposium on Theory of Computing. — 1988. — P. 434–444.

83. Fenz, D. Efficient similarity search in very large string sets, Proceedings of the 24th international conference on Scientific and Statistical Database Management / D. Fenz, D. Lange, A. Rheinlander , F. Naumann , U. Leser // June 25-27, 2012, Chania, Crete, Greece, Pages 262-279.

84. Ferragina, P. The String B-tree: A New Data Structure for String Search in External Memory and Its Applications / P. Ferragina, R. Grossi // J. ACM, — 1999 — №46(2) — P.236-280.

85. Flajolet, P. Digital Search Trees Revisited / P. Flajolet, R. Sedgewick // SIAM J. Computing — 1986 — №15 — P.748–767.

86. Fred, A. Combining multiple clusterings using evidence accumulation / A. Fred, A.K. Jain // IEEE Tran. on pattern analysis and machine intelligence —

2005 — V.27 — P.835-850.

87. Fredkin, F.H. Trie memory / F.H. Fredkin // Communication of the ACM — 1960 — 3 — P.490-500.

88. Gaither J. The variance of the number of 2-protected nodes in a trie / J. Gaither, M.D. Ward // Proceedings of the Meeting on Analytic Algorithmics and Combinatorics, p.43-51, January 06-06, 2013, New Orleans, Louisiana.

89. Goldberg, D.E. Genetic Algorithms in Search / D.E. Goldberg // Optimization, and Machine Learning. Addison-Wesley, Reading, Mass., 1989.

90. Gonzalez, T. Clustering to minimize the maximum intercluster distance / T. Gonzalez // Theoretical Computer Science. — 1985. — Vol. 38. — P. 293–306.

91. Halasz, F. The Dexter hypertext reference model / F. Halasz, M. Schwartz // Communications of the ACM — 1994 — Vol.37(2) — P. 30-39.

92. Hamerly, G. Learning the k in k-means / G. Hamerly, C. Elkan // NIPS, 2003.

93. Heinz, S. Burst tries: a fast, efficient data structure for string keys / S. Heinz, J. Zobel, H. Williams // ACM Trans. Inf. Syst. — 2002 — 20(2) — P.192-223.

94. Hwang, R. Z. The slab dividing approach to solve the Euclidean p-center problem / R.Z. Hwang, R. C. T. Lee, R.C. Chang // Algorithmica. — 1993 — Vol. 9, no. 1. — P. 1–22.

95. Hwang, R.Z. The generalized searching over separators strategy to solve some NP-Hard problems in subexponential time / R.Z. Hwang, R.C. Chang, R.C.T. Lee // Algorithmica. — 1993. — Vol. 9, no. 4. — P. 398–423.

96. Kechedzhy, K.E. Rank distributions of words in additive many-step Markov chains and the Zipf law (англ.) / K.E. Kechedzhy, O.V. Usatenko, V.A. Yampol'skii // Phys. Rev. E. — 2004. — Vol. 72. — P. 046138(1)-046138(6).

97. Kumar, P. Almost optimal solutions to k-clustering problems / P. Kumar, P. Kumar // International Journal of Computational Geometry & Applications. 2010. — Vol. 20, no. 4.

98. Levene, M. “On the information content of semi-structured databases“ / M.

Levene // Acta cybernetica. — 1998. — Vol. 13. N 3. — P.257-276.

99. Luhn, H.P. Key word-in-context index for technical literature (KWIC index) / H.P. Luhn // American Documentation. — 1960.— Vol.11(4) — P.288-295.

100. McCabe, J. On serial files with relocatable records / J. McCabe // Operations Research — 1965. — Vol. 13. — P.609-618.

101. MacQueen, J.B. Some Methods for classification and Analysis of Multivariate Observations / J.B. MacQueen // Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability — Berkeley, University of California Press, 1967, 1 — P.281-297.

102. Matteucci, M. A Tutorial on Clustering Algorithms. [Electronic resource] / M. Matteucci // Dipartimento di Elettronica, Informazione e Bioingegneria. Politecnico di Milano. Access mode: http://home.deib.polimi.it/matteucc/Clustering/tutorial_html/index.html. — 25.03.2019.

103. Michalski, R.S. Automated construction of classifications: conceptual clustering versus numerical taxonomy / R.S. Michalski, R.E. Stepp // IEEE Transactions on Pattern Analysis Machine Intelligence. Apr;5(4), 1983, P.396-410.

104. Morrison, D. PATRICIA-practical algorithm to retrieve information coded in alphanumeric / D. Morrison // J. ACM 15,4(0ct. 1968), 514-534.

105. Nilsson, S. Implementing a Dynamic Compressed Trie / S. Nilsson, M. Tikkanen // Proc. 2nd Workshop on Algorithm Engineering (Saarbruecken, Germany, 1998) 25–36.

106. Nilsson, S. An experimental study of compression methods for dynamic tries / S. Nilsson, M. Tikkanen // Algorithmica 33 (1) (2002) 19–33.

107. Patt, Y.N. Variable length tree structures having minimum average search time / Y.N. Patt // Communications of the ACM, v.12 n.2, p.72-76, Feb. 1969

108. Pibiri, G.E. Efficient Data Structures for Massive N-Gram Datasets / G.E. Pibiri, V. Rossano // Proceedings of the 40th International ACM SIGIR

Conference on Research and Development in Information Retrieval, August 07-11, 2017, Shinjuku, Tokyo, Japan.

109. Prokopec, A. Cache-tries: concurrent lock-free hash tries with constant-time operations / A. Prokopec // ACM SIGPLAN Notices, v.53 n.1, p.137-151, January 2018.

110. Radanne, G. Regenerate: a language generator for extended regular expressions GPCE 2018 / G. Radanne, P. Thiemann // Proceedings of the 17th ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences P.202-214

111. Reznik, Y.A. Analysis of a class of tries with adaptive multi-digit branching / Y.A. Reznik // Proceedings of the 9th international conference on Algorithms and Data Structures, August 15-17, 2005, Waterloo, Canada.

112. Reznik, Y.A. On the Average Density and Selectivity of Nodes in Multi-Digit Tries / Y.A. Reznik // Proc. 7th Workshop on Algorithm Engineering and Experiments and 2nd Workshop on Analytic Algorithmics and Combinatorics (ALENEX/ANALCO 2005) (SIAM, 2005).

113. Reznik, Y.A. Some Results on Tries with Adaptive Branching / Y.A. Reznik // Theoretical Computer Science 289 (2) (2002) 1009–1026.

114. Sample, N. Managing Complex and Varied Data with the IndexFabric(tm) / N. Sample, B. Cooper, M. Franklin, G. Hjaltason, M. Shadmon, L. Cohe // ICDE, pages 492-493, 2002.

115. Scholkopf, B. “Kernel Principal Component Analysis” / B. Scholkopf, A. Smola, K. Muller // Advances in Kernel Methods - Support Vector Learning, 1999.

116. Scotts, P.D. Petri net based hypertext: document structure with browsing semantics / P.D. Scotts, R. Furuta // ACM transaction systems. Vol.7(1). Jan 1989. P.3-29.

117. Stanfel, L.E. Tree Structures for Optimal Searching / L.E. Stanfel // Journal of the ACM (JACM), v.17 n.3, p.508-517, July 1970.

118. Steindorfer, M. J. Towards a software product line of trie-based collections / M.J. Steindorfer, J.J. Vinju // Proceedings of the 2016 ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences, October 31-November 01, 2016, Amsterdam, Netherlands.

119. Steindorfer, M. J. Code specialization for memory efficient hash tries (short paper) / M. J. Steindorfer, J. J. Vinju // Proceedings of the 2014 International Conference on Generative Programming: Concepts and Experiences, September 15-16, 2014, Vasteras, Sweden.

120. Strehl, A. Clustering ensembles — a knowledge reuse framework for combining multiple partitions / A. Strehl, J. Ghosh // The journal of machine learning research. 2002. V.3. P.583-617.

121. Sussenguth, E.H. Use tree structures for processing files / E.H. Sussenguth // CACM 6, 1963, P.272-279.

122. Szpankowski, W. Average Case Analysis of Algorithms on Sequences / W. Szpankowski. — John Wiley& Sons, New York, 2001.

123. Emelyanov, N.E. "Web server on the basis of NIKA DBMS" / N.E. Emelyanov, I.V. Muhanov, V.A. Tishchenko // Proceedings of the third international workshop on "Advances in databases and information systems", ACM SIGMOD, Moscow, sep. 10-13, 1996, Vol.2, p.58-59.

124. Truong, T. Transparent inclusion, utilization, and validation of main memory domain indexes / T. Truong , T. Risch // Proceedings of the 27th International Conference on Scientific and Statistical Database Management, June 29-July 01, 2015, La Jolla, California.

125. Youden, W.W. Index, Volumes 1--10 (1954--1963) / W.W. Youden // JACM, V. 10, P.583-646.

126. Walczuch, N. Using individual prefixes in B^+-trees / N. Walczuch, H. Hoeger // Journal of Systems and Software, 47(1):45-51, 1999.

127. Wen J., Yang G. Staged Symbolic Execution for Parallel Property Checking / J. Wen, G. Yang // ACM SIGSOFT Software Engineering Notes, v.41 n.6,

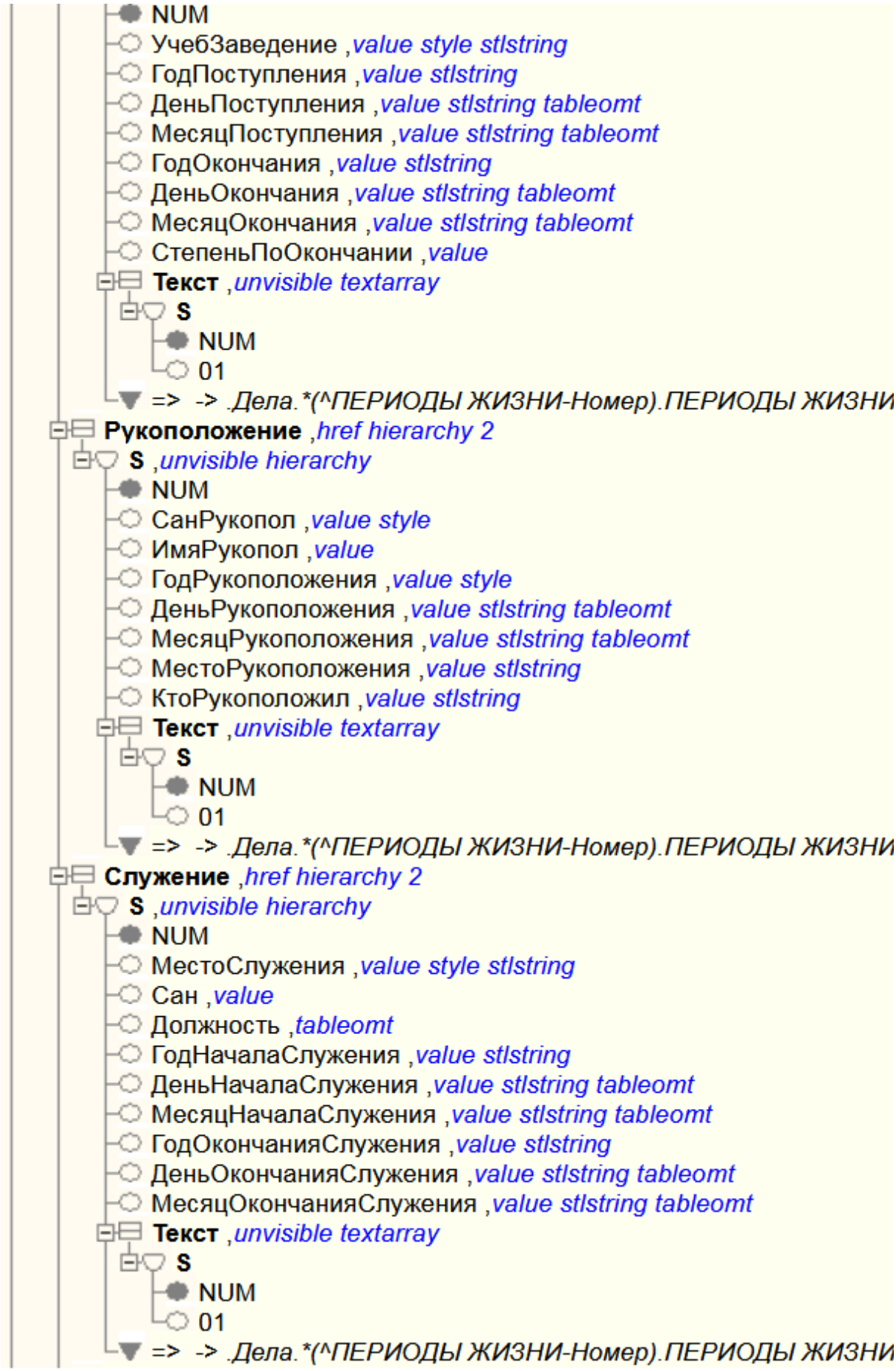
November 2016.

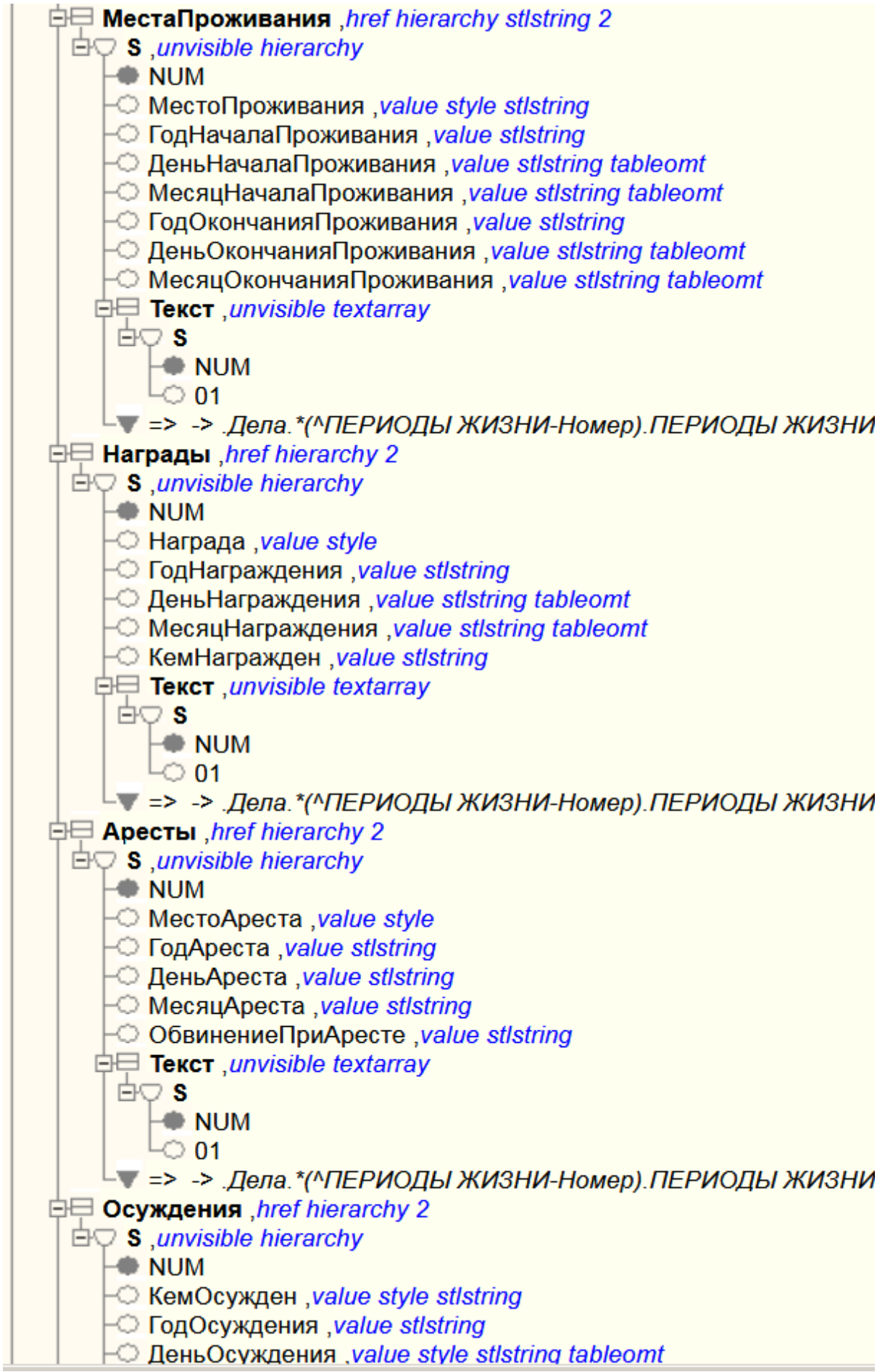
128. Wentian, L. Random Texts Exhibit Zipf's-Law-Like Word Frequency Distribution : paper [Electronic resource] / L. Wentian // Santa Fe Institute, 1991. - С. 8. Access mode: https://santafe.edu/research/results/working-papers/random-texts-exhibit-zipfs-law-like-word-frequency — 25.03.2019.

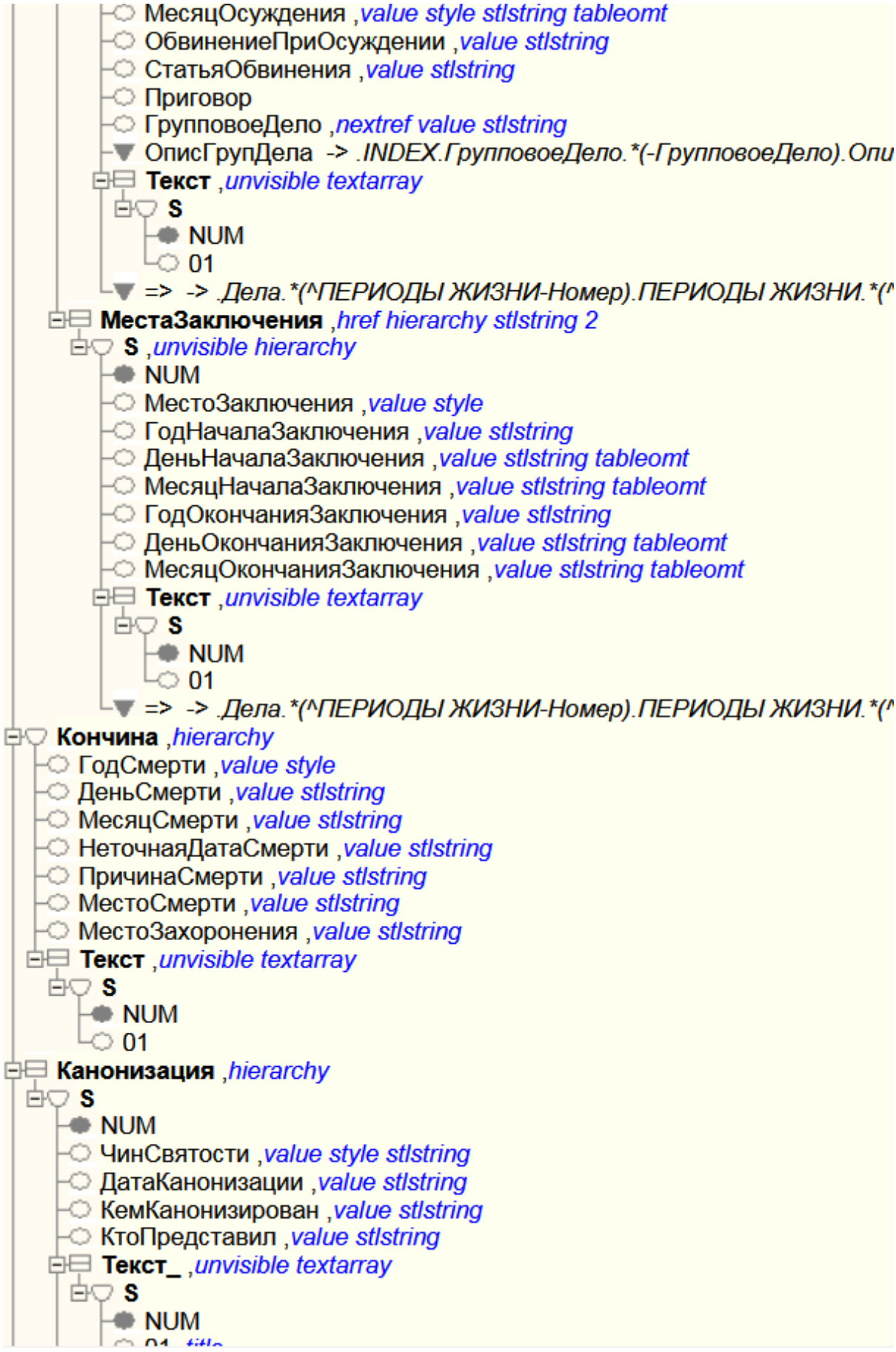
Список принятых сокращений

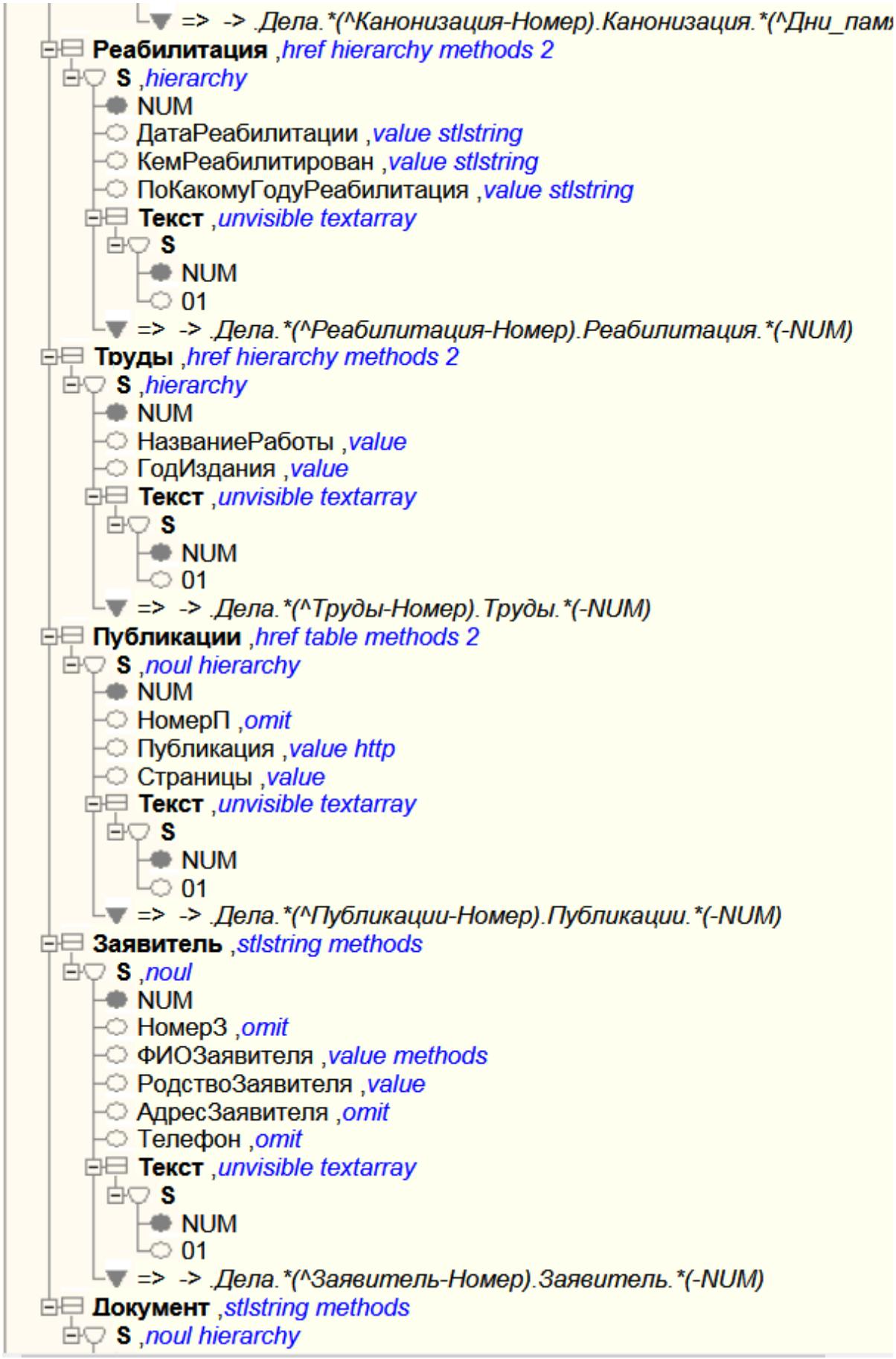
БЗ	база знаний
луч	по***луч***ение (информации)
МАГИС	макетный генератор информационных систем
МНК	метод наименьших квадратов
НИКА	система Новой Информационной Комплексной Автоматизации
ООБД	объектно-ориентированная база данных
ПДС	префиксное дерево сочетаний
ПО	предметная область
dcm	document
DTD	data type definition
CSS	cascading style sheets
HTML	hypertext markup language
LC-trie	level compressed trie
OOML	object-oriented markup language
PATRICIA	practical algorithm to retrieve information coded in alphanumeric
pdf	portable document format
RDF	resource description framework
SGML	standard generalized markup language
trie	re***trie***val
XML	e***x***tensible markup language
XPath	XML path language
XSL	e***x***tensible stylesheet language
XSLFO	XSL formatting objects
XSLT	XSL transformations

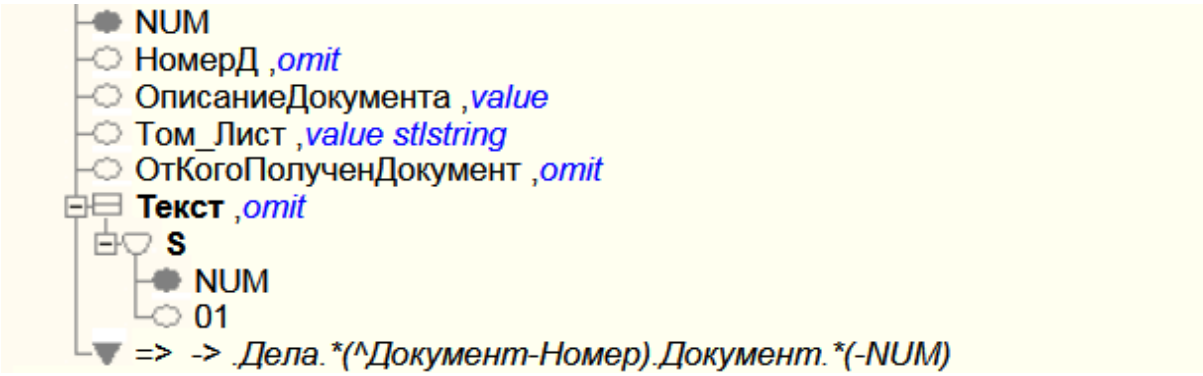
Приложение А. Схема описания данных для массива “Дела”


NUM
УчебЗаведение ,value style stlstring
ГодПоступления ,value stlstring
ДеньПоступления ,value stlstring tableomt
МесяцПоступления ,value stlstring tableomt
ГодОкончания ,value stlstring
ДеньОкончания ,value stlstring tableomt
МесяцОкончания ,value stlstring tableomt
СтепеньПоОкончании ,value
Текст ,unvisible textarray
S
NUM
01
=> -> .Дела.*(^ПЕРИОДЫ ЖИЗНИ-Номер).ПЕРИОДЫ ЖИЗНИ
Рукоположение ,href hierarchy 2
S ,unvisible hierarchy
NUM
СанРукопол ,value style
ИмяРукопол ,value
ГодРукоположения ,value style
ДеньРукоположения ,value stlstring tableomt
МесяцРукоположения ,value stlstring tableomt
МестоРукоположения ,value stlstring
КтоРукоположил ,value stlstring
Текст ,unvisible textarray
S
NUM
01
=> -> .Дела.*(^ПЕРИОДЫ ЖИЗНИ-Номер).ПЕРИОДЫ ЖИЗНИ
Служение ,href hierarchy 2
S ,unvisible hierarchy
NUM
МестоСлужения ,value style stlstring
Сан ,value
Должность ,tableomt
ГодНачалаСлужения ,value stlstring
ДеньНачалаСлужения ,value stlstring tableomt
МесяцНачалаСлужения ,value stlstring tableomt
ГодОкончанияСлужения ,value stlstring
ДеньОкончанияСлужения ,value stlstring tableomt
МесяцОкончанияСлужения ,value stlstring tableomt
Текст ,unvisible textarray
S
NUM
01
=> -> .Дела.*(^ПЕРИОДЫ ЖИЗНИ-Номер).ПЕРИОДЫ ЖИЗНИ


МестаПроживания ,href hierarchy stlstring 2
S ,unvisible hierarchy
NUM
МестоПроживания ,value style stlstring
ГодНачалаПроживания ,value stlstring
ДеньНачалаПроживания ,value stlstring tableomt
МесяцНачалаПроживания ,value stlstring tableomt
ГодОкончанияПроживания ,value stlstring
ДеньОкончанияПроживания ,value stlstring tableomt
МесяцОкончанияПроживания ,value stlstring tableomt
Текст ,unvisible textarray
S
NUM
01
=> -> .Дела.*(^ПЕРИОДЫ ЖИЗНИ-Номер).ПЕРИОДЫ ЖИЗНИ
Награды ,href hierarchy 2
S ,unvisible hierarchy
NUM
Награда ,value style
ГодНаграждения ,value stlstring
ДеньНаграждения ,value stlstring tableomt
МесяцНаграждения ,value stlstring tableomt
КемНагражден ,value stlstring
Текст ,unvisible textarray
S
NUM
01
=> -> .Дела.*(^ПЕРИОДЫ ЖИЗНИ-Номер).ПЕРИОДЫ ЖИЗНИ
Аресты ,href hierarchy 2
S ,unvisible hierarchy
NUM
МестоАреста ,value style
ГодАреста ,value stlstring
ДеньАреста ,value stlstring
МесяцАреста ,value stlstring
ОбвинениеПриАресте ,value stlstring
Текст ,unvisible textarray
S
NUM
01
=> -> .Дела.*(^ПЕРИОДЫ ЖИЗНИ-Номер).ПЕРИОДЫ ЖИЗНИ
Осуждения ,href hierarchy 2
S ,unvisible hierarchy
NUM
КемОсужден ,value style stlstring
ГодОсуждения ,value stlstring
ДеньОсуждения ,value style stlstring tableomt


МесяцОсуждения ,value style stlstring tableomt
ОбвинениеПриОсуждении ,value stlstring
СтатьяОбвинения ,value stlstring
Приговор
ГрупповоеДело ,nextref value stlstring
ОписГрупДела -> .INDEX.ГрупповоеДело.*(-ГрупповоеДело).Опи
Текст ,unvisible textarray
S
NUM
01
=> -> .Дела.*(^ПЕРИОДЫ ЖИЗНИ-Номер).ПЕРИОДЫ ЖИЗНИ.*(^
МестаЗаключения ,href hierarchy stlstring 2
S ,unvisible hierarchy
NUM
МестоЗаключения ,value style
ГодНачалаЗаключения ,value stlstring
ДеньНачалаЗаключения ,value stlstring tableomt
МесяцНачалаЗаключения ,value stlstring tableomt
ГодОкончанияЗаключения ,value stlstring
ДеньОкончанияЗаключения ,value stlstring tableomt
МесяцОкончанияЗаключения ,value stlstring tableomt
Текст ,unvisible textarray
S
NUM
01
=> -> .Дела.*(^ПЕРИОДЫ ЖИЗНИ-Номер).ПЕРИОДЫ ЖИЗНИ.*(^
Кончина ,hierarchy
ГодСмерти ,value style
ДеньСмерти ,value stlstring
МесяцСмерти ,value stlstring
НеточнаяДатаСмерти ,value stlstring
ПричинаСмерти ,value stlstring
МестоСмерти ,value stlstring
МестоЗахоронения ,value stlstring
Текст ,unvisible textarray
S
NUM
01
Канонизация ,hierarchy
S
NUM
ЧинСвятости ,value style stlstring
ДатаКанонизации ,value stlstring
КемКанонизирован ,value stlstring
КтоПредставил ,value stlstring
Текст_ ,unvisible textarray
S
NUM


=> -> .Дела.*(^Канонизация-Номер).Канонизация.*(^Дни_пам
Реабилитация ,href hierarchy methods 2
S ,hierarchy
NUM
ДатаРеабилитации ,value stlstring
КемРеабилитирован ,value stlstring
ПоКакомуГодуРеабилитация ,value stlstring
Текст ,unvisible textarray
S
NUM
01
=> -> .Дела.*(^Реабилитация-Номер).Реабилитация.*(-NUM)
Труды ,href hierarchy methods 2
S ,hierarchy
NUM
НазваниеРаботы ,value
ГодИздания ,value
Текст ,unvisible textarray
S
NUM
01
=> -> .Дела.*(^Труды-Номер).Труды.*(-NUM)
Публикации ,href table methods 2
S ,noul hierarchy
NUM
НомерП ,omit
Публикация ,value http
Страницы ,value
Текст ,unvisible textarray
S
NUM
01
=> -> .Дела.*(^Публикации-Номер).Публикации.*(-NUM)
Заявитель ,stlstring methods
S ,noul
NUM
НомерЗ ,omit
ФИОЗаявителя ,value methods
РодствоЗаявителя ,value
АдресЗаявителя ,omit
Телефон ,omit
Текст ,unvisible textarray
S
NUM
01
=> -> .Дела.*(^Заявитель-Номер).Заявитель.*(-NUM)
Документ ,stlstring methods
S ,noul hierarchy


NUM
НомерД ,omit
ОписаниеДокумента ,value
Том_Лист ,value stlstring
ОтКогоПолученДокумент ,omit
Текст ,omit
S
NUM
01
=> -> .Дела.*(^Документ-Номер).Документ.*(-NUM)

Приложение В. Описание спецификаций

Спецификация	Тип объекта*)	Описание	Атрибут	Примечание
AKEY	V#	предел частоты сочетания		максимум частоты
ALN	V#	выравнивание: r, l, c	К**)	DBP_PROP: -1, -2, -3
ANS	V*	перекодировка в ANSI		с HTTP для ссылок
APP	V#@pr*	пользовательская специф.	202	xml: атрибут appspec
ARn	V#	см. HRHn		устар.
AUD	V*,t	ссылка на аудиофайл		значок aud.gif
AUT	V#@pr	авторизация	248	см. список атрибутов
BN	V#	печать границ массива		
BV	V#@r	изменяет базовую вершину	262	в координате в БД
CLN	V@	календарь	254	параметр спец.
CMB	V@	N-граммы ключ. уровня		
CNR	V#@pr*	опред. элемент-контейнер		вершина внутри конт.
CNT	V#	отображение счётчиков	401	также 402 и 403
CNV	V*	преобразование значения	228	
CRO	V@	объект циклич. ссылки		идёт по ссылке
CTX	Vpr	определяет контекст		через ссылку на мас.
DBC	V#@*	печатает координату в БД		в виде одной строки
DBP		см. DBC		тоже, что и DBC
DFL	V*	значение по умолчанию	206	при отсутствии знач.
DT		изменяет имя схемы		не реализована
DTF		формат даты		не реализована
ELM	V@	печатает посл. элемент		для элемента массива
ELP	V*	эллипсис после n-го симв.		
ERH	V#@pr*	задаёт обработчик ошибок		
EXC	V#@pr*	не отображать вершину		запр. ссылки на подч.
FKW	V#	список ключевых слов	242	наиболее частотных
FLD	V*	терм. верш. в виде поля		форма ввода вершин
FLT	V@	условная фильтрация верш.	304	назнач. элем. массива
FNF	V#	селективная таблица	300	см. TAB
FORM	V#@r	добавляет форму в докум.		запр. назн. на подчин.
FRM	V*	список поле=значение		поле списка hidden

) Типы объектов: v# — массив, v@ — структура, vp — ссылка на шаблон, vr — ссылка на значение, v — терминальная вершина

**) Указывается в конфигурации гипертекстовой системы СУБД НИКА

FST	V#	отобр.только 1-й элемент		
GIF	V*	имя файла изображения		в формате GIF
GO	Vpr	проход по ссылке		в иерархии
GRPn	V#	алфавитный классификатор	210	парам. : длина ключа
HR	V*	присв.ссылку след. нетерм.		терм. как нетерм.
HRHn	V#@r	n-мерная иерарзия	201	парам.: глубина иерар.
HTML	V@	отображает как html-элем.		запись html в БД
HTMR	V#@r	задаёт корневой элемент		
HTTP	V*	uиперссылка		http://;https://
HTTPB	V#@rp*	гиперссылка как кнопка	210	
HYP	V#@r	нетерм. как гиперссылка		в иерархии
IMG	V*	отобр. как элемент изобр.		в виде img-элемента
INC	V#@rp	включение файла в докум.		<алиас_БД.inc>
INO	V#@rp*	пропуск верш. в индексе		устаревш., см. CRO
INR	V*	присв.ссылку след. нетерм.		в индексе, см. HR
JPG	V*	имя файла изображения		в формате JPG
KPN	V#	шаблон для поиска по кл.		на ключевом уровне
L_K	V#@rp*	перейти по ключу	204	на тек. уровне
L_N	V#@rp*	перейти на след. верш.		на тек. уровне
LIM	V#@*	определяет разделитель	216	по умолч.
LVN	V#	выбор макс.уровня иерар.	236	список номеров уров.
MAP	V#@r	определяет геофрейм		использ.карт.сервис
MLNT	V*,float	долгота:градусы в пиксели	264	проекция Меркатора
MLTD	V*,float	широта: градусы в пиксели	264	проекция Меркатора
MSG	V#@rp	сообщение при отсут.верш.	224	в атр. номер сообщен.
MTH	V#@rp	групповая спецификация	302	атт.: мас.груп.специф.
MVF	V#	переход по ключу в массиве		через поле ввода
MVK	V#	переход по ключу в массиве		через путь в БД
NEX	V#@rp	не раскрывать иерархию		под ссылкой
NEXT	V#@r	разделитель в конце пути		текущая координата
NL	V#@r	подчин.термин.без раздел.		
NLP	Vrp	цикл.ссылки с однокр. прох.		не реализована
NLR				не реализована
NMR	V@	нумератор для шкалы		см. спец. SC (scale)
NOH	V#@r	печатает нетерм.как текст		без ссылки на подчин.
NU	V#@r	не добавл.после нетерм. ul		
NXT	V*	терм.со ссылк.след.нетерм.		
OMT	V#@rp*	пропускает вершину		
OPC	V#@r	интерак.скрыв./показ.элем.	256	параметры спец.
OW	V*	открыв.ссылку в новом окне		совместно с NXT

PFX	V@	отобр.путь в БД как префикс		совместно с DBP
PNL	V#	панель кнопок в массиве		
RD	V#@rp	изменяет путь в БД	218	на подчин.объект
REFT	Vrp	ссылка на другое дерево БД	298	,299
RF	V#@rp	иерархия всех подчинённых		
RN	V@,el*)	переименование эл. массива	220	список формат. dn**)
RNG	V@,el	выбор диапазонов ключей	238	параметры спец.диап.
ROW	V#@r	число строк как в структуре		
RSL	V#@r	автом.проход по гиперссыл.		атр. name=resolve
RUL	V#@rp*	выполнение сценария		nkws.dod, nkws.tre
SC	V#	спец. шкала	260	параметры спец.
SH	V#@r	переход на заданную схему	214	схема опис. данных
SKP	V#@r	задаёт ключ в PATH_INFO	232	/spc_1_<key>
SLI	V#	спецификация “слайдер”	258	параметры спец.
SMC	V#	макс. знач. сумм. счётчика	226	
SP	V#@r	применяет иерарх. и таб.		к текущ. вершине
SPT	V@	разделяет строку на подстр.	252	
SRT	V#@*	выполняет сортировку		
STL	V#@*	задаёт стиль отображения	222	указывает стиль
STR	V#@pr*	задаёт строку отображения	212	задаёт строку
SUBJ	V#, t	поле задаёт тему письма		электронной почты
SWC	V@, el	переключает между груп.сп.	250	названия груп. спец.
TABn	V#	вложенная/селектив.таблица	300	задаёт столбцы
TAG	V*	определяет элемент для вер.	230	задаёт тег
TIP	V#	всплывающая подсказка	222	задаёт текст
TIT	V*	заголовок		
TO	V*	пропускает вершину в табл.		
TPL	V#@r	задаёт шаблон	234	номер шаблона
TSC	V*, t	транскрипция вершины		английскими буквами
TXT	V#	текстовый массив		
TXTN	V#	текстовый массив		с нумерацией предл.
UNV	V#@rp	пропускает название верш.		
VIS	key_{el}	печатает ключ массива		
VL	V*	печатает только значение		без ключа вершины

*) элемент массива

**) номер вершины в дереве описания данных или системный номер

Приложение С. Описание атрибутов спецификаций

Атрибут	Тип	Спецификации	Описание
200			
201	V#@r	HRHn	глубина иерархии
202	V#@pr*	APP	xml: атрибут appspec
204	V#@rp*	L_K	перейти по ключу на тек. уровне
206	V*	DFL	значение по умолчанию
210	V#@rp*	GRPN, HTTPB	длина ключа
212	V#@rp*	STR	задаёт строку отображения
214	V#@r	SH	задаёт файл схемы данных
216	V#@*	LIM	определяет разделитель
218	V#@rp	RD	задаёт путь в БД
220	V@,el	RN	задаёт ключ(и) вершины
222	V#@*	STL, TIP	стилевая строка
224	V#@rp	MSG	номер сообщения
226	V#	SMC	макс. знач. сумм. счётчика
228	V*	CNV	тип преобразования
230	V*	TAG	задаёт тег
232	V#@r	SKP	/spc_1_<key> задаёт key
234	V#@r	TPL	номер шаблона в файле nkws.tpl
236	V#	LVN	список номеров уровней
238	V@,el	RNG	параметры диапазонов
242	V#	FKW	число ключевых слов
250	V@, el	SWC	названия груп. спец.
252	V@	SPT	задаёт разделитель
254	V@	CLN	параметр спец.
256	V#@r	OPC	параметры спец.
258	V#	SLI	параметры спец.
260	V#	SC	параметры спец.
262	V#@r	BV	задаёт базовую вершину в коор.
264	V*,float	MLNT, MLTD	уровень увеличения
270		не реализован	псевдоним
299	Vrp	REFT	параметры спец.
300	V#	FNF, TABn	массив столбцов таблицы
302	V#@rp	MTH	массив групповых спецификац.
304	V@	FLT	условие фильтрации вершин
401	V#	CNT	формат счётчика по поддереву
402	V#	CNT	формат счётчика по уровню
403	V#	CNT	формат суммарного счётчика

Приложение D. Фрагмент оптимального классификатора по полю ФИО (34 657 биографических справок)

Ниже приводятся фрагменты алфавитного классификатора: 1-ый уровень, 2-ой уровень, 3-ий уровень, соответствующий ключевому массиву. В квадратных скобках приводятся число сочетаний на данный префикс (в префиксном дереве) и частота встречаемости данного префикса.

1-ый уровень классификатора

А *Аарон (Аверин /.../ Гаврилович)---Ашуева Анна Степановна* [23] [2016]
Б *Б. Андрей---Бялыницкий-Бируля Борис Андреевич* [11] [2462]
В *Вавaев Николай Арсеньевич---Вячеслова Лидия Николаевна* [16] [1905]
Г *Габриалович Вера Болиславовна---Гущо Анна Макаровна* [10] [1630]
Д *Давид (Бекетов Дмитрий Николаевич)---Дятловский Федор Илларионович* [17] [1251]
Е *Ева (Павлова Акилина Васильевна)---Ешмеков Василий Васильевич* [16] [729]
Ж *Жабров Василий Михайлович---Жюно Мария Люсиеновна* [10] [323]
З *Забавин Борис Иванович---Зятьков Никита Николаевич* [16] [828]
И *Иадор (Ткаченко Иларион Афанасиевич)---Ия (Репина Ольга Ивановна)* [20] [1041]
К *Кабанкина Мелания Ивановна---Кюппар (Кюпар) Иван Кириллович* [15] [3874]
Л *Лабазов Петр Владимирович---Ляшков Иван Михайлович* [9] [1452]
М *Мавра (Богатова Мария Дмитриевна)---Мячина Мария Андреевна* [13] [2406]
Н *Набережная Елизавета Владимировна---Нюхин Егор Семенович* [6] [1313]
О *Обердусс---Ошурков Филипп Кузьмич* [18] [701]
П *Паведская Мария Александровна---Пяшкевич Федор Алексеевич* [15] [2932]
Р *Работинская Мария Васильевна---Ряхов Григорий Савельевич* [12] [1088]
С *Сабаев Андрей Васильевич---Сюсюкайло Илья Романович* [17] [3538]
Т *Табаков Иосиф Сергеевич---Тяросова Анисья Федоровна* [12] [1301]
У *Уар (Шмарин Петр Алексеевич)---Ушков Федор Васильевич* [19] [348]
Ф *Фабриков Лаврентий Петрович---Фуфаев Илья Матвеевич* [7] [767]
Х *Хабаев Павел Иванович---Хутынский Александр Константинович* [9] [376]
Ц *Цабина Наталья Федоровна---Цюпа Николай Иванович* [8] [198]
Ч *Ч. Валентина---Чучунов Николай Иванович* [11] [667]
Ш *Шабаев Александр Григорьевич---Шушунова Прасковья Дмитриевна* [12] [892]
Щ *Щавлева Афимия Тимофеевна---Щучалин Николай Алексеевич* [4] [154]
Э *Эверметидов Василий Николаевич---Эшкеев Михаил Васильевич* [9] [22]
Ю *Ювеналий (Литвиненко /.../ Алексеевич)---Юшков Федор Яковлевич* [15] [152]
Я *Ябелов Василий Михайлович---Яшунин Григорий (Герасим?) Петрович* [18] [291]

2-ой уровень классификатора (фрагмент)

Аарон (*Аарон (Аверин /.../ Гаврилович)---Аарон (Кулезнев Адриан Михеевич)* [2] [3]
Аб *Абаимов Василий Михайлович---Абызова Мария Васильевна* [12] [71]
Ав *Аввакум (Боровков Григорий Антонович)---Автухов Василий Григорьевич* [9] [97]
Аг *Агаев Михаил Евграфович---Агупова Ольга Осиповна* [8] [84]
Ад *Ададурова Александра Николаевна---Адушкина Александра Ивановна* [6] [26]
Аж *Ажгеревич Надежда Григорьевна---Ажмяков Илья Терентьевич* [2] [4]
Аз *Азанов Иван Абрамович---Азрапкин Прокопий Петрович* [5] [24]
Аикин Андрей Васильевич *Аикин Андрей Васильевич---Аикин Андрей Васильевич*
Ай *Айвазов Иван Георгиевич---Аймаков Николай Семенович* [2] [2]
Ак *Академов Всеволод Николаевич---Акутин Василий Егорович* [6] [93]
Ала *Алабин Владимир Александрович---Алашев (Алашеев?) Михаил Иванович* [6] [10]
Алдо *Алдокимова Антонина Федоровна---Алдошкин Трофим Епифанович* [2] [2]
Алебастров Николай Михайлович *Алебастров Николай Михайлович---Алебастров Николай Михайлович*
Алевтина (*Алевтина (Васильнина Софья Михайловна)---Алевтина (Овчинникова Мария Александровна)* [3] [3]
Алеев *Алеев Василий Павлович---Алеев Михаил Иванович* [2] [2]
Алейников *Алейников Петр Евлампиевич---Алейникова Олимпиада Назаровна* [2] [3]
Алекина Екатерина *Алекина Екатерина---Алекина Екатерина*
Алекса *Алексагин Никита Тимофеевич---Алексапольский Василий Алименьевич (Алинич?)* [3] [134]
Алексе *Алексеев (Аскольдов-Алексеев) Сергей Александрович---Алексенцева Домна Егоровна* [3] [90]
Алекси *Алексий---Алексия (Тимашева-Беринг) Александра Григорьевна* [3] [46]
Алекторский Михаил Васильевич *Алекторский Михаил Васильевич---Алекторский Михаил Васильевич*
Алеманов В *Алеманов Василий Михайлович---Алеманов Виктор Александрович* [2] [2]
Ален *Аленин Иван Никитич---Аленкин Никита Петрович* [2] [2]
Алеутов Павел Андреевич *Алеутов Павел Андреевич---Алеутов Павел Андреевич*
Алефиренко *Алефиренко Михаил Дмитриевич---Алефиренко Пелагея Григорьевна* [2] [3]
Алех *Алеханова Мария Абрамовна---Алехина Мария Тимофеевна* [2] [7]
Алеш *Алешин Даниил Дмитриевич---Алешковский Константин Петрович* [2] [11]
Али *Алимов Александр Алексеевич---Алиференкова Милитина Васильевна* [4] [15]
Алкеев Филипп Прокопьевич *Алкеев Филипп Прокопьевич---Алкеев Филипп Прокопьевич*
Алма *Алмазов Александр---Алманова Феодосия Алексеевна* [2] [23]
Ало *Алоев Александр Иванович---Алонтьев Кирилл А.* [3] [8]
Алпатский Афанасий Михайлович *Алпатский Афанасий Михайлович---Алпатский Афанасий Михайлович*
Алтухов *Алтухов Александр Петрович---Алтухова Анна Евсеевна* [2] [4]
Алфе *Алфеев (Ялфеев) Александр Васильевич---Алферьев Сергей Васильевич* [2] [28]
Алхимович *Алхимович Аркадий Михайлович---Алхимович Сергей Романович* [2] [2]
Алы *Алыков Евгений Яковлевич---Алышев Василий Федорович* [2] [3]
Аль *Альбанов Николай Петрович---Альшанов Данил Григорьевич* [5] [18]
Алю *Алюминарская Ольга Михайловна---Алюхин Алексей Алексеевич* [2] [2]
Аля *Аляевдин Александр Никандрович---Аляркинский Александр Александрович* [5] [19]
Ам *Аманацкий Григорий Федорович---Амфитеатров Порфирий Иванович* [9] [64]
Ана *Анаевский Иван Иванович---Анаховская Агния Михайловна* [5] [71]
Ангел *Ангелина---Ангелова Елена Михайловна* [2] [17]
Анд *Андарах Евдокия Николаевна---Андрющенко Михаил Алексеевич* [2] [158]

3-ий уровень — первая группа ключей на префикс “Алех”

Алеханова Мария Абрамовна
Алехин Андрей Степанович
Алехин Владимир Алексеевич
Алехин Владимир Павлович
Алехин Трофим Яковлевич
Алехина Акилина Назаровна
Алехина Мария Тимофеевна
Алешин Даниил Дмитриевич
Алешинский /.../ Николаевич
Алешинский Иван Николаевич
Алешинский Николай Иванович
Алешкин Аким Андреевич
Алешкин Сергей Спиридонович
Алешкина Вера Андриановна
Алешкова Мария Андреевна (Алексеевна)
Алешковский Григорий Васильевич
Алешковский Иван Григорьевич
Алешковский Константин Петрович
Алимов Александр Алексеевич
Алимов Дмитрий Емельянович
▼ *Алимова ... — Алмазов ...*
Алех

* Алеха Алехи Алех 2