На правах рукописи

Чернов Андрей Фёдорович

АНАЛИЗ И РАЗРАБОТКА ИНДЕКСА ДЛЯ ПОИСКА ПОСЛЕДОВАТЕЛЬНОСТЕЙ ЭЛЕМЕНТОВ ПРОИЗВОЛЬНОГО ТИПА ПО ИХ ФРАГМЕНТАМ В РЕЛЯЦИОННЫХ БАЗАХ ДАННЫХ

05.13.11 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

АВТОРЕФЕРАТ

диссертации на соискание ученой степени кандидата технических наук

Санкт-Петербург – 2014

Работа выполнена в *федеральном государственном бюджетном образовательном учреждении высшего профессионального образования «Вологодский государственный университет» на кафедре автоматики и вычислительной техники*.

Научный руководитель:	*кандидат технических наук, доцент,* *Андрианов Игорь Александрович*
Официальные оппоненты:	*доктор технических наук, профессор,* *старший научный сотрудник, заведующий кафедрой вычислительных систем и информатики Государственного университета морского и речного флота имени адмирала С.О. Макарова,* *Марлей Владимир Евгеньевич*
	кандидат физико-математических наук, *доцент кафедры системного программирования Санкт-Петербургского государственного университета,* *Кознов Дмитрий Владимирович*
Ведущая организация:	*Санкт-Петербургский государственный политехнический университет*

Защита состоится «28» мая 2014 года в 15 часов 30 минут на заседании диссертационного совета Д 212.238.01 Санкт-Петербургского государственного электротехнического университета «ЛЭТИ» им. В.И. Ульянова (Ленина) по адресу: 197376 Санкт-Петербург, ул. Профессора Попова, д. 5.

С диссертацией можно ознакомиться в библиотеке СПбГЭТУ «ЛЭТИ» и на сайте www.eltech.ru

Автореферат разослан «28» ___марта___ 2014 г.

Ученый секретарь
диссертационного совета Д 212.238.01,
к.т.н. Щеголева Н.Л.

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность работы. В современном обществе огромную роль играют информационно-поисковые системы (ИПС). Информационные технологии проникают во все сферы человеческой деятельности. Постоянное увеличение накопленной информации в базах данных (БД) приводит к росту нагрузки и, как следствие, росту требований к поисковым системам, что подчеркивает актуальность и значимость исследований в области информационного поиска.

Для эффективного поиска в БД необходимо иметь соответствующее математическое и программное обеспечение. Для ускорения поиска или даже для создания возможности его выполнения необходима предварительная обработка данных с целью создания вспомогательных поисковых структур – *индексов*. Современные индексы базируются на таких поисковых структурах как битовые карты, B- и R-деревья, и их модификации, суффиксные структуры данных и др.

Вопросам проектирования структур данных и алгоритмов для информационного поиска посвящены работы Д. Хеллерстейна, А. Гуттмана, Р. Финкеля, М. Николà и других. Из отечественных публикаций можно выделить работы О. С. Бартунова, Т. Г. Сигаева, И. А. Андрианова, П. Г. Айткулова, Л. М. Бойцова и других. Особенно значимыми для данной диссертации явились работы Д. Хеллерстейна, О. С. Бартунова и Т. Г. Сигаева в области структур данных для информационного поиска.

Среди различных видов поиска важным и востребованным является поиск последовательностей элементов (числовые массивы, строки и т.п.). Наиболее востребованным является поиск таких данных по фрагменту. Подобный поиск поддерживается современными индексами с целым рядом ограничений: не всегда учитывается порядок следования элементов, большие требования к памяти и недостаточно эффективные алгоритмы построения индексов.

Кроме того, удобство/возможность индексации последовательностей элементов часто бывают ограничены по причине нормализации хранимой в БД информации. Однако в последнее время все чаще прибегают к изменению логической структуры БД для ее *сознательной денормализации*. То есть используется умышленное дублирование данных с целью ускорения поиска. К примеру, имеется БД кредитных договоров в коммерческом банке, по каждому договору в хронологическом порядке выполнено множество операций (выдача кредита, погашение, вынос на просрочку и т.п.). Необходимо ускорить поиск кредитных договоров в запросах типа: «найти все договоры, по которым после выдачи кредита было не менее 2 погашений подряд». Для ускорения и упрощения формулирования (например, на языке SQL) подобных поисковых запросов удобно хранить идентификаторы выполненных по договору операций не в отдельной связанной таблице БД, а в виде числового массива в той же таблице. Для полученных числовых массивов удобно построить индекс, ускоряющий их поиск по фрагменту. Это подчеркивает актуальность поиска последовательностей элементов. Однако базы данных, соответствующие третьей нормальной форме (стандарт де-факто при проектировании БД), не позволяют эффективно выполнить такой поиск даже с использованием индексов.

Подобные задачи поиска востребованы и в других предметных областях: информационные системы учебных заведений, системы сбора статистики и т.п.

Поэтому представляется достаточно актуальной задачей реализация индексного метода доступа (МД) к последовательностям элементов для их поиска по фрагменту.

Целью работы является создание индексного метода доступа к данным для эффективного поиска последовательностей элементов по их фрагменту. При этом в качестве последовательностей могут выступать числовые массивы, текстовые строки, бинарный программный код и др.

Для достижения поставленной цели в диссертации решены следующие **задачи**:

1. Анализ существующих подходов к построению индексов и выбор наиболее подходящего типа индекса для поиска последовательностей по фрагменту.
2. Модификация выбранной поисковой структуры и алгоритмов для ускорения поиска последовательностей по фрагменту.
3. Разработка программных реализаций модифицированной структуры индекса.
4. Получение экспериментальных данных, сравнение с известными аналогами.
5. Внедрение результатов в деятельность конкретных организаций.

Объектом исследования являются коллекции небольших и средних размеров (до нескольких десятков гигабайт), содержащие последовательности элементов произвольного типа.

Предметом исследования являются способы организации и алгоритмы работы с индексными структурами для эффективного поиска последовательностей по фрагменту.

Методы исследования. В работе использованы методы теории вероятностей, комбинаторики, теории графов, а также методы анализа алгоритмов.

Научная новизна полученных результатов заключается в следующем:

1. Разработана модификация структуры RD-дерева, предложенной Дж. Хеллерстейном и имеющей недостатки при поиске последовательностей в коллекциях размером несколько гигабайт и выше. В рамках модификации доработана структура внутренних узлов дерева, а также алгоритм поиска, в результате чего достигнуто существенное ускорение поиска последовательностей по фрагменту.
2. Доработан алгоритм построения RD-дерева путем добавления в него этапа анализа индексируемых данных, в результате чего сформированные на данном этапе актуальные для содержимого БД параметры построения индекса приводят к дополнительному ускорению поиска с использованием построенной структуры. Кроме того, доработка алгоритма построения индекса обеспечивает одинаковую высокую эффективность поиска, не зависимую от содержимого БД.
3. Предложен подход для уменьшения размера RD-дерева, в котором вместо сигнатур фиксированной длины применены сигнатуры переменной длины, из которых исключаются неиспользуемые младшие и старшие байты. В результате значительно (до нескольких раз) уменьшен размер индекса.

Практическая значимость. Практическую ценность имеют следующие полученные в диссертации результаты:

1. Разработанный индекс для СУБД PostgreSQL, основанный на модифицированном RD-дереве, применяемый для ускорения поиска числовых последовательностей по фрагменту и представляющий собой программный модуль для расширения СУБД.
2. Методика для ускорения и упрощения формулирования сложных, но достаточно востребованных, поисковых запросов, использующая сознательную денормализацию БД и поиск фрагментов в числовых массивах, тем самым обеспечивающая актуальность применения модифицированных

RD-деревьев для ускорения поиска числовых массивов.

3. Прикладное программное обеспечение (ПО), основанное на разработанных структурах и алгоритмах, позволяющее значительно ускорить поиск по сравнению с существующими программными средствами, используемыми для поиска последовательностей по фрагменту.

На защиту выносятся следующие основные результаты и положения:

1. Модификация RD-деревьев, предложенных Дж. Хеллерстейном, и поисковые алгоритмы, использующие ее для существенного ускорения поиска последовательностей по фрагменту.
2. Алгоритм формирования актуальных для содержимого БД параметров построения RD-дерева, используемый при анализе данных на начальном этапе построения индекса и приводящий к дополнительному ускорению поиска, а также обеспечивающий одинаковую высокую эффективность, не зависимую от содержимого БД.
3. Подход к построению RD-дерева, приводящий к значительному (до нескольких раз) уменьшению его размера, использующий в узлах дерева сигнатуры переменной длины, из которых исключаются неиспользуемые младшие и старшие байты.

Апробация работы. Основные результаты работы докладывались и получили положительную оценку на ряде международных, всероссийских и региональных конференций, в том числе: международная научно-практическая конференция «Вопросы образования и науки в XXI веке» (г. Тамбов, 2013 г.), международная научно-практическая конференция «Современное общество, образование и наука» (г. Тамбов, 2013 г.), седьмая международная научно-техническая конференция «Автоматизация и энергосбережение машиностроительного и металлургического производств, технология и надежность машин, приборов и оборудования» (г. Вологда, 2012 г.), всероссийская научная конференция «Молодые исследователи – регионам» (г. Вологда, 2009, 2010, 2011 гг.), научный семинар вологодского регионального отделения Научного совета РАН по методологии искусственного интеллекта (г. Вологда, 2013 г.).

Кроме того, материалы работы использованы в Федеральной целевой программе «Научные и научно-педагогические кадры инновационной России» на 2009-2013 годы: «Разработка методов формализации и верификации распределенных информационно-поисковых систем на основе сервис-ориентированной архитектуры».

Достоверность научных положений и результатов работы подтверждается математическими доказательствами и выкладками, результатами практических экспериментов по измерению основных параметров разработанного индекса, в том числе на реальных данных, а также апробацией основных положений работы на международных и всероссийских конференциях.

Реализация и внедрение результатов. Теоретические и практические результаты работы внедрены в программные продукты ЗАО «Эр-Стайл Софтлаб» и в учебный процесс кафедры автоматики и вычислительной техники Вологодского государственного университета:

- реализация эффективного поиска по фрагменту в числовых массивах, отражающих хронологию выполнения кредитных операций,
- значительное повышение производительности операции проверки банковских

клиентов на причастность к террористическим организациям,

- эффективная по скорости работы реализация поиска программного кода решений студентов в системе дистанционного лабораторного практикума по программированию.

Результаты работы используются в учебном процессе при преподавании курсов «Построение и анализ алгоритмов», «Алгоритмы и структуры данных» и на занятиях научного кружка «Программист».

Публикации. Основное содержание диссертационной работы опубликовано в 13 научных работах, среди которых 3 публикации в ведущих рецензируемых изданиях, рекомендованных ВАК, 1 монография и 9 докладов в материалах международных, всероссийских и региональных конференций.

Структура и объём диссертации. Диссертация состоит из введения, 4 глав, заключения, списка сокращений, библиографического списка и 3 приложений. Работа содержит 156 страниц машинописного текста, включая 49 рисунков и 8 таблиц. Библиографический список включает 65 наименований.

СОДЕРЖАНИЕ РАБОТЫ

Во **введении** обоснована актуальность диссертационной работы, ее научная новизна, определены цель и задачи исследования, сформулированы научные результаты, выносимые на защиту, с обоснованием их практической значимости, приведена информация об апробации и внедрении работы.

В **первой главе** определяются направления и конкретные задачи диссертации, выполняется анализ способов индексации последовательностей элементов.

В начале главы приводятся примеры конкретных практических задач, подтверждающие актуальность проводимого исследования. Первая задача заключается в ускорении *поиска по фрагменту в числовых массивах*, отражающих хронологию каких-либо событий. Задача ориентирована на поиск в коллекциях данных размером до нескольких десятков гигабайт. Значительно расширяет применение данной задачи поиска сознательная денормализация данных для хранения в БД числовых массивов с целью их эффективного поиска. При этом весьма актуальной задачей становится умение выполнять эффективный поиск массивов по фрагменту.

Вторая практическая задача, для которой применяются результаты диссертационного исследования, – *точный поиск по подстроке*. Задача ориентирована на поиск в текстовых полях БД для таблиц размером до нескольких десятков гигабайт, не накладывая ограничений на длину искомых подстрок и статичность индексируемых данных.

Далее выполняется анализ способов индексации данных и выбор типа индекса для последовательностей элементов.

Индексы на основе битовых карт разработаны для индексации данных, имеющих ограниченный набор возможных значений. Поэтому для часто изменяемых последовательностей данный тип индексов неприменим.

Хеш-индексы могут обрабатывать только простые запросы на точное равенство. Поэтому для поиска последовательностей по фрагменту этот МД также не подходит.

B-деревья* имеют множество достоинств. Однако существующие решения на базе B-деревьев, к примеру, для поиска текста по подстроке требуют достаточно много памяти, а эффективность алгоритмов их построения сильно зависит от длины индексируемых строк. Кроме того, требуется возможность упорядочивания элементов

по значению, что накладывает ограничения на тип элементов последовательностей. По этим причинам весьма затруднительно каким-либо образом адаптировать B-деревья для поиска в произвольных последовательностях по произвольным фрагментам.

Суффиксные структуры данных предназначены для поиска текста по подстроке, что является частным случаем поиска последовательностей по фрагменту. Однако их основной проблемой является их размер. Хотя асимптотика – *O(n)*, фактический размер индекса многократно превышает размер индексируемых данных. Поэтому данный тип индексов целесообразно применять только для коллекций небольшого размера (до нескольких гигабайт). Другой недостаток суффиксных поисковых структур в том, что даже незначительные изменения данных приводят к серьёзной перестройке индекса. Поэтому такие индексы не подходят для динамических данных. Такая разновидность суффиксных структур, как суффиксные массивы, использует сортировку суффиксов индексируемых строк в лексикографическом порядке, что накладывает ограничение на тип элементов (аналогично B-деревьям). По этим причинам суффиксные структуры неприменимы для достижения цели данного исследования.

R-деревья разработаны для индексации пространственных данных, поэтому не могут быть использованы для последовательностей элементов.

Модификация же R-деревьев – *RD-деревья* – разработана специально для индексации наборов элементов. Алгоритмы работы с RD-деревом не зависят от типа элементов в индексируемых наборах, поэтому одинаково эффективны для наборов элементов любого типа. RD-деревья обладают высокой скоростью поиска на реальных данных, а алгоритмы построения и обновления делают возможным их применение для динамических данных. Индексы данного типа очень компактные, благодаря использованию блюмовских фильтров в узлах дерева. Таким образом, данный тип индексов применимым для последовательностей элементов любого типа в коллекциях среднего размера (до нескольких десятков гигабайт).

RD-деревья являются разновидностью R-деревьев, изначально предложенных в 1984 году А. Гуттманом для индексации пространственных данных. В 1994 году Дж. Хеллерстейном предложена модификация R-деревьев для индексации наборов элементов – RD-деревья (пример на рис. 1). Для данных, хранимых в узлах RD-дерева, Хеллерстейном был введен термин *сигнатура*. Сигнатуры представляют собой *блюмовские фильтры* – битовые массивы, позволяющие ответить на вопрос о вхождении наборов элементов друг в друга. Каждому значению элементов в сигнатурах соответствует 1 бит, выбираемый хеш-функцией, – признак наличия в последовательности.

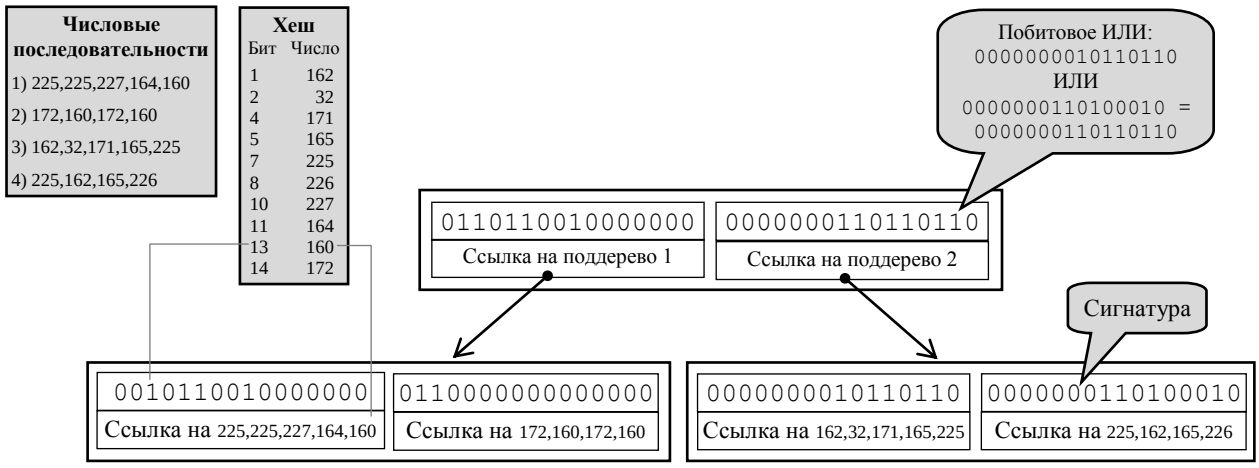


Рис. 1. Пример RD-дерева для индексации числовых последовательностей

RD-деревья имеют ряд недостатков:

- Эффективность поиска зависит от количества «ложных попаданий» – ошибочных обращений к внешней памяти из-за коллизий хеширования. Это ограничивает применимость только небольшими коллекциями данных.
- Поиск не учитывает порядок расположения элементов, что ограничивает применимость к хронологическим последовательностям, текстовым данным и пр.

Данные недостатки RD-деревьев не исключают потенциальной возможности, посредством ряда модификаций, их применения для эффективного поиска последовательностей по фрагменту. Поэтому для достижения цели диссертации выбирается индексный МД, основанный на RD-деревьях. Далее в работе проанализированы недостатки RD-деревьев и предложена их модификация, хорошо применимая для последовательностей элементов произвольного типа в коллекциях среднего размера (до нескольких десятков гигабайт).

Во **второй главе** выполняется асимптотический анализ алгоритмов работы с RD-деревом, более детально анализируются причины «ложных попаданий» при поиске последовательностей элементов, выполняется математическая оценка количества «ложных попаданий». В результате разработана модификация структуры RD-дерева, призванная минимизировать количество «ложных попаданий».

Асимптотический анализ показал, что временная сложность построения RD-дерева такая же, как у B- и R-деревьев, и для n последовательностей элементов в наихудшем случае составляет $O(n \cdot log(n))$. Требования к памяти у RD-деревьев тоже такие же, как у B- и R-деревьев, и составляют $O(n)$. Особенностью структур R- и RD-деревьев является то, что они не гарантируют приемлемой временной сложности поиска в наихудшем случае, но хорошо работают на реальных данных. Из-за «ложных попаданий», вызванных коллизиями хеширования, временная сложность поиска в наихудшем случае составляет $O(n)$, т.к. при этом придется обойти все дерево целиком. Такая асимптотика сложности поиска плохо отражает работу индекса на реальных данных, поэтому в диссертационной работе выполнен анализ сложности поиска для среднего случая, учитывая вероятность появления «ложных попаданий». Полученная асимптотика для среднего случая составляет $O(\frac{n}{\log(n)})$, что более точно отражает сложность поиска по RD-дереву на реальных данных, хотя всё же данная оценка несколько завышена.

В диссертации обосновывается, что основными причинами «ложных попаданий» при поиске последовательностей элементов на более высоком уровне (относительно коллизий хеширования) являются следующие:

- не учитывается количество одинаковых элементов в последовательностях,
- не учитывается порядок расположения элементов.

Далее во второй главе выполняется математическая оценка количества «ложных попаданий» при поиске по RD-дереву, т.е. количества «лишних» обращений к внешней памяти, которые придется сделать, чтобы найти одну запись.

$$K_{ложн\_попаданий\_на\_1\_успех} = \frac{P_{чтения\_стр}}{P_{попадания}} - 1, \quad (1)$$

где $P_{чтения\_стр}$ – вероятность чтения страницы внешней памяти в листовом узле дерева для проверки результата поиска и возможного исключения «ложного попадания»; $P_{попадания}$ – вероятность успешного исхода поиска (попадания).

Таким образом, для определения теоретического относительного количества «ложных попаданий» необходимо вычислить значения вероятностей $P_{чтения\_стр}$ и $P_{попадания}$. Для их определения были получены вспомогательные формулы, использующие предложенную математическую модель реальных данных.

Пусть имеются произвольные последовательности элементов длиной m. Каждый элемент может принимать k возможных значений. В каждой последовательности могут присутствовать одинаковые элементы (повторения). Для упрощения оценки принимаем допущение, что в последовательностях может повторяться только одно какое-либо значение в среднем r раз на каждую (в разных последовательностях может повторяться разное значение). Например, для числовой последовательности {32,**54**,12,45,**54**,6,1,**54**,9,0} длина m равна 10, а величина r (для значения 54) равна 3. В работе определено, сколько можно составить различных последовательностей (комбинаторных соединений), удовлетворяющих данным условиям – $A_k^{m,r}$. Отдельно рассмотрен случай, когда повторяющиеся элементы не являются соседними – A_k^{m,r^*} (это потребуется далее). В результате были получены следующие выражения:

$$A_k^{m,r} = A_k^{m'} + A_{k-1}^{m-r} \cdot V^{m,r} \cdot k,$$

$$A_k^{m,r^*} = A_k^{m'} + A_{k-1}^{m-r} \cdot V^{m,r^*} \cdot k, \text{ где}$$

где $A_k^m = \frac{k!}{(k-m)!}$, $A_k^{m'} = \begin{cases} A_k^m, если(k \geq m) \\ 0, если(k < m) \end{cases}$,

$V^{m,r} = \prod_{i=0}^{r-1} (\frac{1}{i+1} \cdot (m-i))$, $V^{m,r^*} = \begin{cases} V^{m-(r-1),r}, если(m-r \geq r-1) \\ 0, если(m-r < r-1) \end{cases}$.

Здесь A_k^m – количество размещений без повторений. Подобная математическая модель реальных данных принимается, как адекватная и применимая для теоретической оценки количества «ложных попаданий», поскольку она сравнительно хорошо соотносится с реальными данными.

С помощью данной математической модели определено теоретическое относительное количество «ложных попаданий» для случая поиска по RD-дереву в текстовом поле БД со следующими типовыми параметрами:

- средняя длина индексируемых последовательностей m = 50,
- количество возможных значений элементов k = 60 (например, это может быть русский алфавит в двух регистрах, кроме заглавных Ь, Ъ, и т.д.),
- среднее количество возможных повторений какого-либо значения r = 4 (например, это может быть два слова с удвоенным символом).

В результате, используя (1), установлено, что $K_{ложн\_попаданиц\_на\_1\_успех}$ в рассмотренном случае составляет порядка нескольких тысяч.

Далее во второй главе для снижения количества «ложных попаданий» выполнена *модификация структуры RD-дерева*. Для возможности модификации принято решение об увеличении той части сигнатуры, которая приходится на одно значение элементов, с 1 бита до 1 байта (выбрано из соображений программной реализации). Для этой однобайтовой части сигнатуры вводится определение – *сигнатурный элемент* (его структура приведена на рис. 2).

Рис. 2. Структура сигнатурного элемента после всех этапов модификации

Такая доработка структуры сигнатур не изменяет алгоритмов работы с RD-деревом, но увеличивает требования к оперативной и внешней памяти. Для компенсации увеличения размера индекса предложен подход, минимизирующий размер сигнатур RD-дерева. Для этого был проанализирован алгоритм построения сигнатур, предложенный Дж. Хеллерстейном. Данный алгоритм предполагает использование в сигнатурах битовых массивов фиксированной длины. Из данного диапазона хеш-функция выбирает позицию бита для конкретного значения элемента. Это приводит к тому, что сигнатуры часто содержат неиспользуемые младшие и старшие биты. Для того чтобы не хранить эти неиспользуемые нулевые биты предложенный в диссертации подход предполагает использование сигнатур переменной длины в узлах RD-дерева. Предложенный подход применен в четвертой главе при реализации индекса для СУБД PostgreSQL, в результате чего удалось не только полностью компенсировать увеличение размера индекса, но даже его уменьшить (до нескольких раз).

Этапы модификации структуры RD-дерева:

1. В индекс добавляется информация о количестве одинаковых элементов в последовательностях. Вместо признака наличия для каждого значения элементов в сигнатуре сохраняется *количество элементов с таким значением* (двоичное представление, рис. 2). Так как под количество используется 6 младших бит сигнатурного элемента, то максимальное учитываемое количество одинаковых элементов в последовательности будет равно 2^6, т.е. 64. Такого количества вполне достаточно для индексации последовательностей среднего размера (текстовые поля в БД, числовые массивы и т.п.), поскольку на практике количество повторений в последовательностях меньше, чем 64.

Данная модификация RD-деревьев увеличивает скорость поиска *фрагментов с повторениями*, например, {**172,160,172,160**}. Из поиска в таких случаях исключаются поддеревья, в сигнатурах которых для какого-либо значения элементов количество меньше, чем в сигнатуре искомого фрагмента.

После первого этапа модификации RD-деревьев выполнена математическая оценка количества «ложных попаданий», аналогично выполненной ранее оценке для оригинальных RD-деревьев. В результате теоретическое количество $K_{ложн\_попаданий\_на\_1\_успех}$ уменьшилось в среднем более чем на порядок.

2. В сигнатурный элемент добавляется *признак повторения подряд соответствующего значения элементов*. Технически для этого признака используется 1 бит каждого сигнатурного элемента (рис. 2).

Данный этап модификации RD-деревьев ускоряет поиск *фрагментов с соседними повторениями*, например, {**225,225**,227,164,160}. Из поиска в таких случаях исключаются поддеревья, в сигнатурах которых для какого-либо значения элементов сброшен бит

повторения, в то время как в сигнатуре искомого фрагмента этот бит установлен.

После второго этапа модификации RD-деревьев также выполнена математическая оценка количества «ложных попаданий». В результате теоретическое количество $K_{ложн\_попаданий\_на\_1\_успех}$ уменьшилось в среднем более чем в 4 раза по сравнению с первым этапом модификации и более чем в 50 раз по сравнению с оригиналом.

3. На третьем этапе модификации используется разбиение индексируемых последовательностей на части для учета порядка расположения элементов.

Для этого некоторые значения элементов выбираются в качестве *разделителей*, которыми индексируемые последовательности делятся на части – *фреймы*. Полученная информация о границах данных частей добавляется в сигнатуры узлов RD-дерева. Для этого в каждом фрейме выделяются крайние элементы – *граничные элементы*. При этом разделители не могут быть граничными элементами, т.к. не входят в состав фреймов, они лишь разбивают последовательности. Технически 1 бит каждого сигнатурного элемента интерпретируется как признак того, что элементы с соответствующим значением являлись граничными хотя бы раз (рис. 2).

Данный этап модификации RD-деревьев увеличивает скорость поиска фрагментов с граничными элементами, т.е. *фрагментов с фреймами*. Например, {162,**32**,171,165,225} при использовании значения 32 в качестве разделителя (здесь элементы 162 и 171 являются граничными). Из поиска в таких случаях исключаются поддеревья, в сигнатурах которых для какого-либо значения элементов сброшен бит наличия граничных элементов, в то время как в сигнатуре искомого фрагмента этот бит установлен.

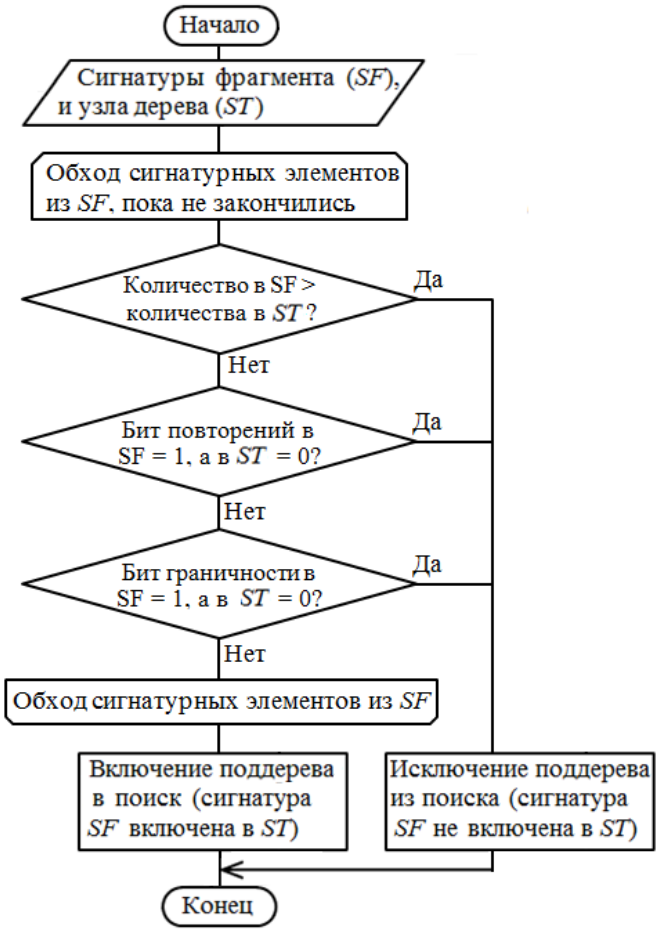


Рис. 3. Алгоритм проверки включения сигнатур друг в друга, используемый при поиске

Выполнить математическую оценку количества «ложных попаданий» после третьего этапа модификации RD-деревьев представляется затруднительным, так как данное количество напрямую зависит от выбранного набора разделителей фреймов. Ценность такой доработки для эффективности поиска доказывается эмпирически в последующих главах.

В результате трех этапов модификации RD-дерева алгоритм проверки включения сигнатур друг в друга, используемый при поиске для принятия решения об исключении поддеревьев из поиска, несколько усложнился (рис. 3). Однако благодаря выполненным модификациям теоретическое количество «ложных попаданий» значительно уменьшилось.

В **третьей главе** производится экспериментальная проверка скорости поиска по индексу и эффективности построения индекса при использовании модифицированной структуры RD-дерева. Доработан алгоритм построения индекса, который непосредственно перед созданием RD-дерева подготавливает актуальные для содержимого БД разделители фреймов. Выполняется оценка степени применимости выполненных модификаций.

Экспериментальная проверка выполнена на реальных данных: русскоязычный текст, англоязычный текст и числовые массивы в виде бинарного программного кода. Для получения итогового результата экспериментальные значения усредняются. Для текстовых данных в качестве разделителей фреймов использованы знаки препинания и служебные символы: «Space(){}[]<>'"?!.,:;^+-*/\». Для числовых массивов разделители фреймов выбраны случайным образом из часто встречающихся значений байтов в бинарном программном коде. Для проведения экспериментов использовалась собственная программная реализация индексов на основе структуры данных и алгоритмов, разработанных в предыдущей главе диссертации.

Проверка скорости поиска выполнена на БД размером 1 миллион записей (рис. 4).

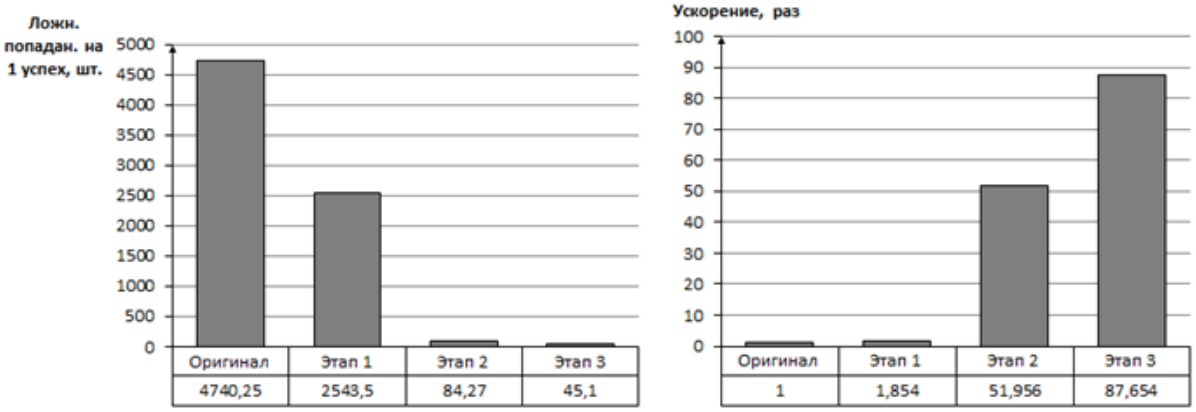


Рис. 4. Результаты скорости поиска фрагментов с соседними повторениями и фреймами

Для различных искомых фрагментов результаты эффективности поиска следующие:

- Фрагменты без повторений и фреймов – наблюдается снижение эффективности поиска не более чем на 3%.
- Фрагменты с повторениями – ускорение поиска составило около 65%.
- Фрагменты с соседними повторениями – скорость поиска возросла более чем на порядок (до 30 раз).
- Фрагменты с соседними повторениями и фреймами – скорость поиска возросла в несколько десятков раз (до 80 раз).

Далее в третьей главе получена экспериментальная *зависимость скорости поиска от размера БД*. На реальных данных убедились, что скорость поиска различных фрагментов в среднем растет достаточно медленно с увеличением размера БД (значительно медленнее асимптотики $O(\frac{n}{\log(n)})$). К примеру, скорость поиска на миллионе записей лишь в два раза выше, чем на 100 тысячах записей. Это подтверждает тот факт, что для RD-деревьев асимптотика сложности поиска недостаточно доказывает их эффективность.

Далее *выявлены недостатки третьего этапа модификации* RD-деревьев, который заключается в разбиении индексируемых последовательностей на фреймы. Проблема в том, что используется фиксированный набор разделителей, который:

- неудобен в использовании, поскольку его нужно каким-то образом задавать в зависимости от индексируемых данных,
- не всегда эффективен при изменяющемся содержимом БД.

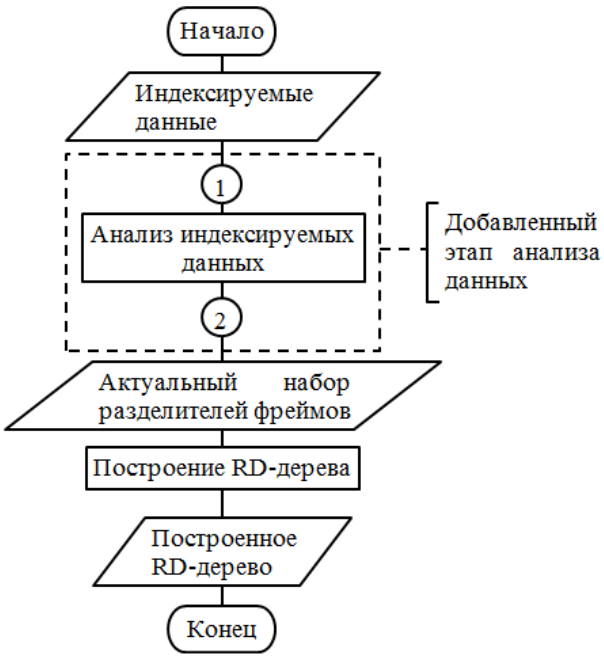


Рис. 5. Доработанный алгоритм построения RD-дерева

Для устранения данных недостатков модифицированных RD-деревьев в алгоритм построения индекса *добавлен этап анализа индексируемых данных* (рис. 5). Назначение данного этапа – сформировать актуальный набор разделителей фреймов для текущего содержимого БД. В общем случае разделителем может быть любое значение элементов, которое удобно использовать в качестве такового.

Актуальность набора разделителей на конкретном содержимом БД удобно определять, исходя из *доли разделителей в общем наборе элементов*. Набор разделителей является актуальным для конкретного содержимого БД (с точки зрения разбиения на фреймы), если все элементы в индексируемых последовательностях являются либо разделителями, либо граничными элементами, т.е. когда количество граничных элементов максимально. Данная ситуация является идеальной, и на реальном содержимом БД добиться такого сочетания разделителей и граничных элементов практически невозможно, но алгоритм формирования актуального набора разделителей должен к этому стремиться.

Обозначив актуальную долю разделителей $X_{акт}$, получили ее зависимость от средней длины индексируемых последовательностей m:

$$X_{акт} = \frac{m}{3 \cdot m - 2},$$
$$\lim_{m \to \infty} X_{акт} = 1/3 \qquad (2)$$

Далее выработан критерий отнесения конкретного значения элементов k к разделителям – это *доля граничных элементов при использовании в качестве разделителей только элементов со значением k*. Данная величина обозначена как Y_k. Чем больше Y_k, тем больше значение k подходит на роль разделителя.

Используя выражение (2) и выработанный критерий отнесения значения элементов к разделителям, сформулирован *алгоритм формирования актуального набора разделителей* для конкретного содержимого БД (рис. 6). Данный алгоритм положен в основу добавленного этапа анализа индексируемых последовательностей при построении RD-дерева (рис. 5).

Для сформулированного алгоритма выполнена экспериментальная проверка корректности на нескольких различных наполнениях БД. Принцип проверки в том, что полученный в результате использования алгоритма набор разделителей действительно создает максимальное количество фреймов.

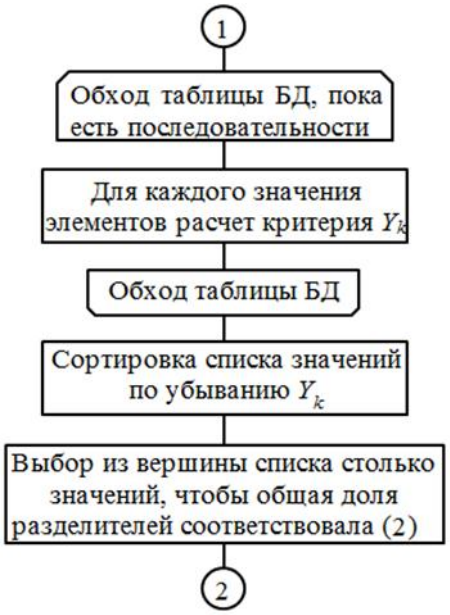


Рис. 6. Алгоритм анализа данных для формирования актуальных разделителей фреймов

Как видно из рис. 6, этап анализа индексируемых данных требует последовательного обхода соответствующей таблицы БД, этого же требует и построение RD-дерева для этой таблицы. Таким образом, для построения RD-дерева вместе с анализом данных *потребуется два обхода таблицы БД*, т.е. анализ займет около 50% от общего времени построения R-дерева, что может сделать эффективность построения индекса неприемлемой.

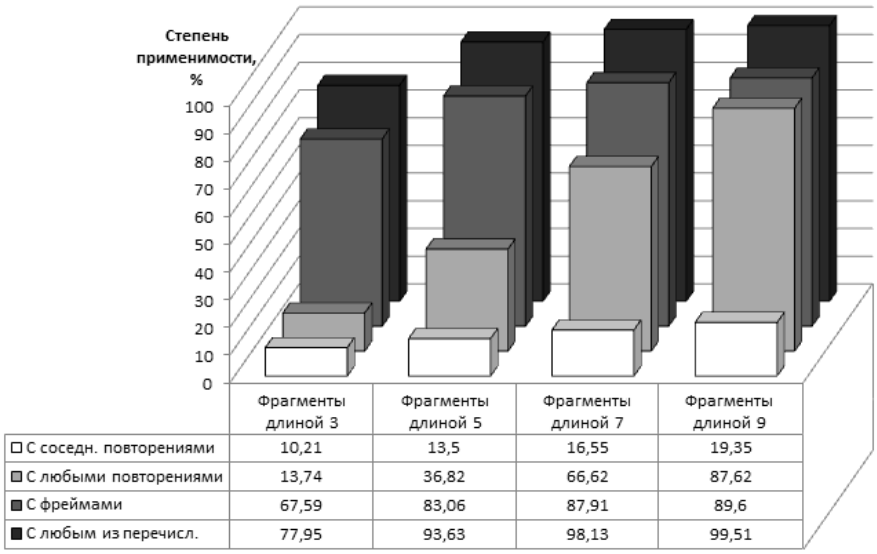
Для ускорения построения RD-дерева было предложено на этапе анализа обходить не всю таблицу целиком, а только определенную долю ее записей в произвольном порядке. На различном тестовом содержимом БД был экспериментально получен минимальный процент обходимых при анализе записей, при котором сформированный набор разделителей не отличается от набора, полученного полным обходом таблицы. Например: русскоязычный текст – 1%, англоязычный текст – 4%, числовые массивы – 6%. Следовательно, на этапе анализа вовсе не обязательно обходить всю таблицу целиком, достаточно в произвольном порядке обойти не более 10% записей. Оценка времени построения индекса показала, что такой этап анализа данных занимает около 11% от общего времени построения индекса. При такой эффективности этап анализа данных применим для коллекций последовательностей размером несколько десятков гигабайт.

Следует отметить, что при существенных изменениях содержимого в индексируемой таблице БД, желательно перестроить индекс, используя тем самым более актуальный набор разделителей фреймов.

В следующем вычислительном эксперименте была измерена скорость поиска последовательностей по фрагменту, используя предварительный анализ данных, а не фиксированный набор разделителей фреймов. Поиск по фрагменту дополнительно ускорился на 10-40% относительно уже трижды модифицированной структуры RD-дерева. Это обусловлено правильным выбором элементов в качестве разделителей для конкретного содержимого БД.

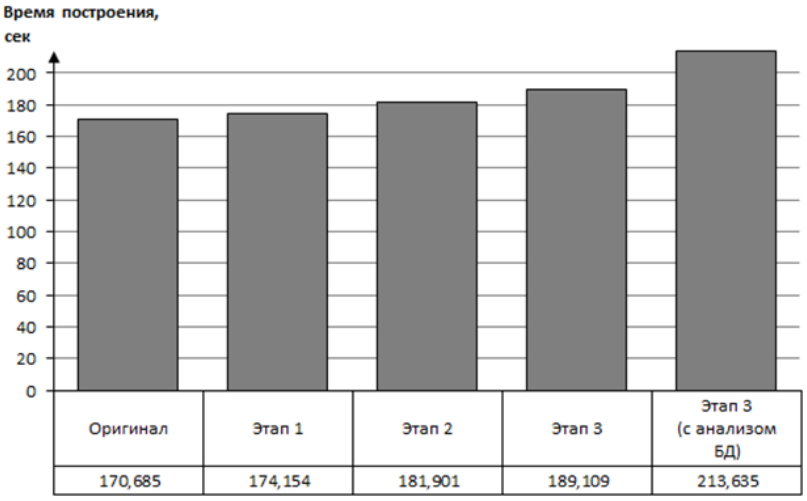
Далее в третьей главе выполнена *оценка степени применимости выполненной модификации* RD-деревьев к различным искомым фрагментам. Для этого на реальных данных размером один миллион записей была определена доля фрагментов, к которым применима модификация индекса, в общем объеме всех возможных искомых фрагментов (рис. 7). Как видим, степень применимости модификаций близка к 100% при достаточно большой длине фрагментов (7 и более).

В конце третьей главы произведена экспериментальная оценка влияния выполненной модификации RD-дерева на время построения индекса (рис. 8). Использовались реальные данные размером один миллион записей.



	Фрагменты длиной 3	Фрагменты длиной 5	Фрагменты длиной 7	Фрагменты длиной 9
С соседн. повторениями	10,21	13,5	16,55	19,35
С любыми повторениями	13,74	36,82	66,62	87,62
С фреймами	67,59	83,06	87,91	89,6
С любым из перечисл.	77,95	93,63	98,13	99,51

Рис. 7. Применимость модификаций RD-деревьев ко всем возможным искомым фрагментам



Оригинал	Этап 1	Этап 2	Этап 3	Этап 3 (с анализом БД)
170,685	174,154	181,901	189,109	213,635

Рис. 8. Влияние выполненной модификации на время построения RD-дерева

Видим, что время построения индекса даже после реализации всех изменений структуры RD-дерева увеличилось приблизительно на 25%, что не критично.

Кроме того, было экспериментально доказано, что рост времени построения индекса при увеличении размера БД соответствует асимптотике $O(n \cdot log(n))$.

В **четвертой главе** рассматривается применение полученных теоретических результатов к решению поставленных практических задач.

Первая задача заключается в ускорении *поиска по фрагменту в числовых массивах*, отражающих хронологию каких-либо событий. Посредством числовых массивов могут быть отражены, например, последовательности состояний, в которых искомый объект находился на протяжении своего существования, последовательности производимых объектом (либо с объектом) операций, и т.д.

Для решения подобных задач поиска *разработана специальная методика*, использующая сознательную денормализацию логической структуры БД и поиск фрагментов в числовых массивах. Данная методика заключается в следующем:

- Последовательности состояний/операций искомых объектов, хранимые в отдельных связанных таблицах БД, преобразуются в числовые массивы;
- В таблицу БД, где хранятся искомые объекты, добавляется новый столбец, в котором сохраняются числовые массивы состояний/операций;
- Для данного столбца таблицы строится индекс на основе модифицированных

RD-деревьев, который позволяет эффективно искать объекты по фрагменту их состояний/операций.

Данная методика позволяет упростить формулировку и существенно ускорить сложные, но при этом достаточно востребованные поисковые запросы. Задача поиска фрагментов в числовых массивах может быть востребована в различных предметных областях. Например, кредитные договоры и произведенные с ними операции, студенты и сданные ими экзамены, сотрудники организации и их должности, и т.п.

Проверка в реальных условиях реализации поиска числовых массивов по фрагменту выполнялась в ЗАО «Эр-Стайл Софтлаб». В подсистеме «Кредитование» интегрированной банковской системы (ИБС) RS-Bank была реализована возможность эффективного поиска кредитных договоров по фрагменту истории выполненных по ним операций. На специально подготовленной БД, содержащей свыше двухсот тысяч кредитных договоров, запросы на поиск последних по фрагменту массива выполненных операций ускорились в несколько раз с нескольких секунд до нескольких десятых секунды.

Вторая практическая задача направлена на *точный поиск текста по подстроке*. При этом текстовые данные рассматриваются как символьные последовательности. Задача ориентирована на поиск в текстовых полях БД для таблиц размером до нескольких десятков гигабайт. При этом искомые подстроки могут быть произвольной длины. На статичность данных ограничений не накладывается.

Для решения задачи эффективного поиска подстроки в тексте существуют различные индексные МД: суффиксные поисковые структуры, так называемые строковые B-деревья и др. Суффиксные структуры данных из-за своих недостатков, рассмотренных в первой главе диссертации, применимы только для небольших коллекций статичных данных. Для строковых B-деревьев время построения значительно больше, чем у обычных B-деревьев и сильно зависит от длины индексируемых строк. Кроме того, фактический размер индекса многократно превышает размер исходных данных (аналогично суффиксным структурам).

Модифицированные в диссертации RD-деревья не имеют ограничений на применение к динамическим коллекциям текстовых данных размером до нескольких десятков гигабайт. Поэтому для поиска таких данных выбор модифицированных RD-деревьев будет являться наиболее предпочтительным по сравнению с другими индексами, благодаря увеличенной скорости поиска, быстрым операциям обновления дерева и его компактности.

Проверка в реальных условиях реализации поиска текстовых данных по подстроке выполнялась в ЗАО «Эр-Стайл Софтлаб» и ВоГУ. В подсистеме «Расчетный банк» ИБС RS-Bank была существенно ускорена системная процедура проверки клиентов банка на причастность к террористическим организациям. На специально подготовленной БД, содержащей свыше 10 тысяч банковских документов и более 3 тысяч клиентов, длительность процедуры проверки снизилась с пяти минут до порядка 30 секунд.

В ВоГУ на кафедре автоматики и вычислительной техники модифицированные RD-деревья использованы для реализации поиска по программному коду в системе дистанционного лабораторного практикума по программированию. Результаты показали, что поиск различных подстрок во всем архиве решений выполняется менее пяти секунд для низкоселективных запросов и порядка полминуты для высокоселективных запросов. Данные результаты являются вполне приемлемыми, учитывая большое количество накопленных в архиве решений.

Как отдельная разновидность текстового поиска по подстроке был рассмотрен *полнотекстовый поиск с использованием хеширования слов* естественного языка в числовые значения. Данный подход предполагается применять для коллекций текстовых документов, содержащих десятки и сотни слов. При этом индексируемые текстовые документы рассматриваются как массивы целых чисел, для которых уже строится индекс на основе RD-дерева.

Далее в четвертой главе выполнена реализация индекса, основанного на модифицированном RD-дереве, для СУБД PostgreSQL. Для модификации был выбран встроенный в СУБД индекс для ускорения поиска числовых массивов (тип intarray). Данный индекс основан на R-деревьях, абстрагированных от пространственных данных с помощью обобщенного программного интерфейса. В результате встроенный индекс представляет собой оригинальное RD-дерево с блюмовскими фильтрами для реализации сигнатур. К этому индексу были применены модификации структуры и алгоритмов, описанные в предыдущих главах.

Результаты проверки эффективности показали ускорение поиска до 10 раз по сравнению с оригинальным встроенным индексом на низкоселективных запросах. На высокоселективных запросах использование индекса не даёт существенных преимуществ из-за многочисленных проверок для исключения «ложных попаданий». В целом выигрыш в эффективности несколько меньше, чем полученный в третьей главе на собственной программной реализации. Причина в кэшировании страниц внешней памяти в СУБД PostgreSQL. Однако результаты поиска согласуются с полученными в третьей главе, но с меньшим эффектом.

По сравнению со встроенным в PostgreSQL индексом время построения модифицированного индекса возросло не более чем на 15%.

Однако удалось уменьшить размер индекса, который получился соизмеримым с размером исходных данных. Этого удалось достичь благодаря предложенному подходу, использующему *сигнатуры переменной длины*, из младших и старших байтов которых исключаются пустые сигнатурные элементы. Во встроенном индексе позиции сигнатурных элементов выбирались хеш-функцией из фиксированного диапазона (остаток от деления на размер диапазона в реализации PostgreSQL). Для этого приходилось под каждую сигнатуру выделять фиксированное количество памяти, равное максимально возможной позиции сигнатурного элемента (252 байта в реализации PostgreSQL). Поэтому даже если последовательность не содержала элементов, которым соответствовали младшие/старшие байты в сигнатуре, последняя все равно была максимально возможного размера. Модифицированная хеш-функция для связи значения элемента последовательности с позицией в сигнатуре использовала *динамическую хеш-таблицу*, благодаря чему использовались только такие позиции сигнатурных элементов, которые соответствовали реально присутствующим в БД значениям. В результате предложенного подхода весь фиксированный размер (252 байта) стал требоваться не всегда, что привело к снижению размера RD-дерева почти в 4 раза. Данный факт подтвердил преимущество RD-деревьев в плане расходования памяти по сравнению с другими индексами.

В **заключении** подводятся итоги работы и формулируются основные полученные результаты.

ОСНОВНЫЕ РЕЗУЛЬТАТЫ РАБОТЫ

1. Разработана модификация структуры RD-дерева, хранящая во внутренних узлах количество одинаковых элементов в последовательностях, признак повторения элементов подряд и признак нахождения элементов на границе фреймов. Доработка позволяет ускорить поиск последовательностей по фрагменту до 10 раз. Ускорение поиска наблюдается на широком диапазоне искомых фрагментов:
   a. фрагменты с повторениями,
   b. фрагменты с соседними повторениями,
   c. фрагменты с фреймами.
2. Экспериментально доказана высокая степень применимости выполненных модификаций. Доля перечисленных выше типов искомых фрагментов от общего количества всех возможных фрагментов близка к 100% при достаточно большой длине фрагментов (7 и более). На более коротких искомых фрагментах применимость модификации несколько меньше.
3. Доработан алгоритм построения RD-дерева посредством добавления в него этапа анализа индексируемых данных. Для этого выработаны необходимые критерии и разработан вспомогательный алгоритм для формирования набора разделителей фреймов, актуального для содержимого БД. Время анализа при этом составляет около 11% от общего времени построения RD-дерева. Использование актуального для содержимого БД набора разделителей фреймов при построении индекса дополнительно ускоряет поиск на 10-40%.
4. Разработана методика для ускорения и упрощения формулирования сложных, но достаточно востребованных, поисковых запросов посредством сознательной денормализации БД и поиска фрагментов в числовых массивах, отражающих хронологию каких-либо событий.
5. В несколько раз уменьшен размер индекса, основанного на RD-дереве, благодаря использованию сигнатур переменной длины в узлах дерева. Этого удалось добиться, несмотря на добавление в узлы RD-дерева новых данных.
6. Разработан и внедрен новый индекс для конкретной СУБД с открытым исходным кодом для ускорения поиска последовательностей по фрагменту.

СПИСОК ПУБЛИКАЦИЙ

В изданиях, рекомендованных ВАК РФ:

1. Чернов, А.Ф. Ускорение поиска в индексах на основе R-деревьев // Программные продукты и системы, 2012. №2 (98), С. 46 - 50.

2. Чернов, А.Ф., Андрианов И.А. Оптимизация разбиения данных в индексах на основе R-деревьев // Системы управления и информационные технологии, 2011. №4 (46), С. 18 - 22.

3. Чернов, А.Ф. Модификация индексов на основе R-деревьев для ускорения поиска // Информационные системы и технологии, 2011. №6 (68), С. 10 - 18.

Монография:

4. Андрианов, И.А. Индексирование и поиск в последовательностях для больших баз данных: монография / И.А. Андрианов, А.Ф. Чернов. – Вологда, ВоГУ, 2013. – 165 с.

Материалы конференций:

Материалы конференций (всего 9 публикаций) приведены в библиографии диссертации.