Жигалова Мария Александровна – Национальный исследовательский университет «Высшая школа экономики»; e-mail: mashuzden@mail.ru; 614070, г. Пермь, бул. Гагарина, 37а; тел. +73422545608; кафедра информационных технологий в бизнесе; студентка.

Сухов Александр Олегович – e-mail: ASuhov@hse.ru; кафедра информационных технологий в бизнесе; к.ф.-м.н., старший преподаватель.

Zhigalova Maria Aleksandrovna – National Research University “Higher School of Economics”; e-mail: mashuzden@mail.ru; 37a, Gagarina street, Perm, 614070, Russia; phone: +73422825372; the department of information technologies in business; student.

Sukhov Alexander Olegovich – e-mail: ASuhov@hse.ru; the department of information technologies in business; cand. of phys.-math. sc.; senior lecturer.

УДК 004.652

А.В. Чернов, В.И. Янц, Е.В. Карпенко

ПРИМЕНЕНИЕ RADIX ДЕРЕВЬЕВ ДЛЯ ИНДЕКСАЦИИ СЛАБОСТРУКТУРИРОВАННЫХ ДАННЫХ*

В системах управления слабоструктурированными данными остро стоит проблема осуществления поиска в больших наборах данных. При осуществлении поиска выполняется значительно число дисковых операций, т.к. для работы с любыми видами данных требуется извлечь их из постоянного хранилища, при этом, на сегодняшний день, дисковая подсистема является самым медленным элементом компьютера. Как следствие, рост количества дисковых операций неминуемо приводит к быстрому исчерпанию вычислительных ресурсов СУБД. Для решения данной проблемы предлагается использовать Radix деревья с модифицированной многослойной структурой для построения поискового индекса. Каждый слой в такой структуре представляет собой дополнительное Radix дерево. Внутри слоя, ветви дерева помещены в структуры соответствующие размеру дискового блока. Переход между блоками осуществляется при помощи двух видов ссылок – маркированных и немаркированных. Ссылки связывают слои и позволяют переходить либо к конкретному, результирующему элементу в слое, либо к блоку данных содержащему ссылку на результирующий элемент дерева в следующем слое. Такая структура позволит выполнять не более 1-й дисковой операции на каждый слой индекса. При этом количество слоев в полученной структуре будет зависеть от объема данных и степени разнородности и лишь в редких случаях превысит число 3. Данная многоуровневая структура легко поддается кэшированию и позволяет сократить количество дисковых операций до одной на один поисковый запрос. Полученная структура позволяет нивелировать недостатки несбалансированных Radix деревьев и получить быстрый и компактный поисковый индекс для применения в широком круге систем управления слабоструктурированными данными, обеспечивая высокую скорость работы с наборами неоднородных, длинных и сложных строк.

Слабоструктурированные данные; индекс; поиск; базы данных, нагруженные деревья, radix деревья, сжатие данных; системы управления базами данных.

A.V. Chernov, V.I. Yants, E.V. Karpenko

ADAPTATION OF RADIX TREES FOR INDEXING SEMISTRUCTURED DATA

In semi-structured data management systems there is acute problem of searching in large data sets. During search implementation, considerable number of disk operations is being performed, which inevitably leads to a rapid depletion of computing resources database with the increase of queries number. To solve this problem, we propose to use Radix tries with a modi-

* Работа выполнена при финансовой поддержке РФФИ, проекты 13-01-00325 А, 15-01-03067 А.

fied multi-layer structure for building the search index. Such structure will allow performing not more than one disk operation for each index layer. The number of layers in the resultant structure will depend on the amount of data and the degree of heterogeneity and only in rare cases will exceed 3. Multi-level structure can be easily cached and allow to reduce the number of disk operations to one per one search query. The resulting structure allows avoiding disadvantages of unbalanced Radix tries and getting a quick and compact search index for use in wide range of semi-structured data management systems.

Semi-structured data; index; searching; database; laden tries; radix tries; data compression; database management system.

Введение. В современном мире, с широким распространением и развитием информационных технологий, доступ к информации стал серьезной проблемой. При этом объем обрабатываемой информации неукоснительно растет, что требует создание новых инструментов для работы с большими массивами данных. Усилия многих специалистов направлены на улучшение моделей извлечения информации, путем поиска новых подходов в работе со слабоструктурированными данными [1]. Это позволит принимать во внимание содержание и структуру слабоструктурированных документов для повышения точности результатов.

Слабоструктурированные данные – представляют собой форму организации данных, при которой структура документа не может быть однозначно классифицирована, допускать исключения и неточности, а также изменяться в течение эксплуатации [2]. Примером таких данных может служить каталог продукции, где данные от нескольких поставщиков (каждый из которых оперирует своей схемой данных) должны быть интегрированы в одну систему, которая позволит покупателям подбирать необходимые им товары посредством одного запроса.

Чаще всего слабоструктурированные данных представляются в виде графа, в котором элементы данных соединены маркированными связями [20]. В традиционных реляционных система эти связи и представляют собой схему данных. В современных системах управления слабоструктурированными данными поддерживать такую структуру не представляется возможным из-за высоких накладных расходом на ее модификацию и поддержку, а также потребности в формализации исходных данных. Тем не менее, графовая или древовидная структура прекрасно подходит для построения быстрых и эффективных индексов.

Один из способов управления слабоструктурированными данными – это хранить и обрабатывать их реляционными СУБД. Данные должны быть сконвертированы для хранения в таблицах, например, используя инструменты от компании Oracle [15]. Данный процесс требует предварительного извлечения схемы данных. Чаще всего конвертация становиться весьма не тривиальной задачей [16]. Если схемы данных не существует, и она не может быть сформирована динамически, тогда данные могут быть сохранены в виде набора элементов вложенных друг в друга [10]. Выполнения поиска на таких структурах крайне неэффективно.

Альтернативный способ – э то построение специализированной системы, которая создана именно для управления слабоструктурированными данными. Существует такие проекты как Lore [14] , Tamino [17] и XYZFind [18]. Все они столкнулись с проблемой сложности осуществления операций поиска в хранилище слабоструктурированных данных - наличие большого количества непредсказуемых связей между элементами приводит к необходимости многократно запрашивать индекс, что очень быстро загружает дисковую подсистему [13]. Чтобы решить данную проблему необходимо разработать инструмент, который позволит сократить количество операций ввод-вывода до минимума, при этом учитывая тот факт, что многие распределённые системы управления базами данных, не гарантируют согласованности данных [19]

Цель данной работы является разработка эффективного поисковый индекса оптимизированного для слабоструктурированных документов, который позволит индексировать большой набор строк в компактной и быстрой структуре. При этом разрабатываемая структура должна быть сбалансирована, чтобы доступ к индексу не требовал большого количества операций ввода/вывода. Это позволит работать с большими и сложными наборами слабоструктурированных данных, даже на медленных дисковых подсистемах.

В этой статье приводиться структура нового поискового индекса и как он может быть использован для оптимизации поиска по слабоструктурированным базам данных. В частности, будут рассмотрены следующие аспекты:

- Использование индекса на длинных и сложных ключах, закодированных в виде строк.
- Применение Radix деревьев для организации поискового индекса.
- Процесс вставки, удаления и обновления ключей в поисковом индексе.
- Сравним полученное решение с существующими подходами.

Анализ существующих решений. Проблема хранения, индексации и поиска слабоструктурированных данных получила большее внимание в научных работах [3, 5, 6, 13].

В одном из исследований [16] была использовано DTD схема для сопоставления XML-данных в реляционные таблицы. Система хранения извлекала схему из самих данных с использованием интеллектуального анализа данных [9].

В работах также отмечается, что достаточно трудно работать с данными, нерегулярной или переменной структуры. В одной из работ было изучено хранение XML-данных в СУБД в виде набора атрибутов и ребер, используя незначительные знания о структуре документа [10]. В других работах, предлагаемые системы хранения данных изначально используют модель слабоструктурированных данных [14, 17, 28]. Процесс поиска в этих системах, как правило, также требует нескольких просмотров индекса [13].

Индекс слабоструктурированных данных. Предлагаемый в данной статье индекс – это особая структура, которая легко масштабируется для большого количества ключей, и нечувствительна к длине или содержанию введенных строк. Эти особенность крайне важна в слабоструктурированных СУБД, т.к. данные там, как правило, задано в виде строке.

Индекс построен на основе нагруженного дерева со сжатием – Radix. Пример Radix дерева показан на рис. 1. Узлы дерева помечены их глубиной. Ребра помечены символами, позволяющими определять, к какому элементу далее производить переход.

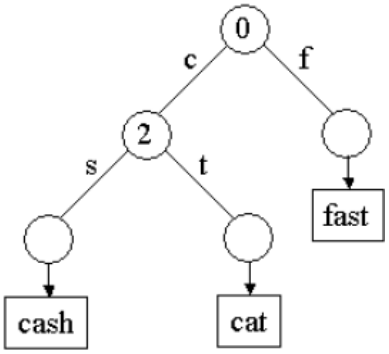

Рис. 1. Radix дерево

Одно из главных преимуществ Radix – размер дерева не зависит от длины вставленных ключей. Причем, каждый новый ключ добавляет не более одного звена и узла к индексу, даже если ключ имеет большую длину. Размер индекса построенного на Radix дереве будет расти крайне медленно, благодаря алгоритму сильного сжатия с потерями.

Один из главных недостатков применения Radix деревьев состоит в том, что получаемая структура несимметрична и внутренне не сбалансирована. Несимметричные структуры довольно сложно использовать для поиска данных, т.к. для выполнения запрос может потребоваться значительное количество обращений к дисковой подсистеме.

Разработанный индекс имеет все положительные свойства деревьев Radix, но сбалансирован и оптимизирован для быстрого доступа к данным. Индекс используется новый, многоуровневый подход: дополнительные слои (уровни) дерева позволяют осуществлять поиск именно в том в блоке данных, в котором может содержаться ответ на запрос. Каждый запрос обращается к одинаковому количеству слоев, что обеспечивает сбалансированный доступ к индексу и дисковой подсистеме.

Базовое Radix дерево разделено на блоки-поддеревья, где каждое поддерево является полноценным Radix деревом. Можно представить данное поддерево в качестве нового горизонтального слоя, дополняющего вертикальную структуру исходного дерева. Если новый горизонтальный слой слишком большой, чтобы уместиться в один блок диска, то он разделяется на два блока, и индексируются в третьем горизонтальном слое.

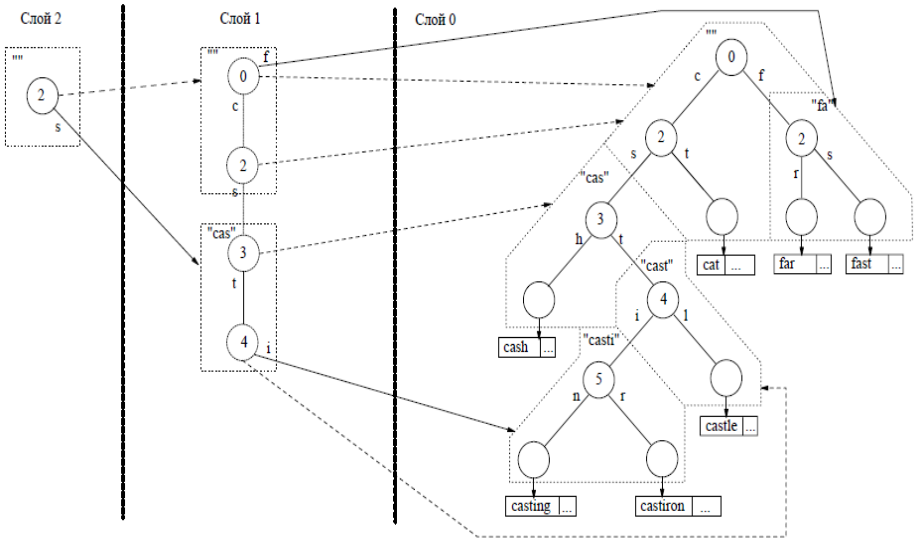

Рис. 2. Многослойный индекс

Внутри индекса определены два вида ссылок из слоя i к слою i-1: маркированные ссылки ⟶ и немаркированные ссылки ⇢. Маркированные ссылки ничем не отличаются от обычных ребер в дереве, не считая того, что маркированная ссылка соединяет узел в одном слое с поддеревом в следующем слое. Немаркированная ссылка связывает узел в одном слое, с конкретным узлом содержащем тот же префикс в следующем слое.

Таким образом, на рис. 2, узел отмеченный как "3" в слое 1 соответствует префиксу "cas" и соединен с поддеревом в слое 0 при помощи немаркированной ссылки.

Поиск. Процесс поиска начинается в корневом узле блока в левом горизонтальном слое. В конкретном блоке, поиск протекает нормально, сравнивая символы в поисковом ключе с ребрами. Если помеченное ребро является маркированной ссылкой, поиск переходит по горизонтали к другому блоку в следующем слое направо. Если немаркированное ребро совпадает с соответствующим символом поискового ключа, поиск следует горизонтально напрямую к ребру нового блока в следующем слое. Поиск переходит от слоя к слою, пока нижний слой (слой 0) не будет достигнут, и требуемые данные не будут найдены. Во время поиска в слое 0, если непомеченное ребро не совпадает с соответствующим символом поискового ключа, то это означает, что искомый ключ не существует, и поиск прекращается. В противном случае, поиск приведет к искомым данным. По завершению поиска крайне важно убедиться, что найденные данные соответствуют поисковому ключу, т.к. из-за сжатия с потерями данных применяемого в Radix деревьях возможно ложное совпадение.

Процесс поиска рассматривает один блок на слой, и всегда рассматривает одинаковое количество слоев. Если блоки по размеру соответствуют дисковым блокам, то это означает, что поиску может потребоваться только одна операция ввода/вывода в каждом слое, конечно, если необходимый блок не находится в оперативной памяти. Одно из преимуществ использования структуры Radix является то, что ключи хранятся очень компактно, и многие из них могут уместиться в один блок. Таким образом, блоки имеют очень высокую производительность. На практике это означает, что индекс, хранящий большое количество ключей (например, 1 миллиард) в общем случае требует всего трех слоев. Причем слой 0 можно хранить только на диске, а слои 1 и 2 дополнительно кэшировать в оперативной памяти.

Обновление индекса. Обновления, вставки и удаления, равно как и поиск, могут быть выполнены очень эффективно. Обновление осуществляется последовательным удалением старого ключа и последующей вставкой нового значение ключа. Вставка ключа в дерево Radix включает в себя либо добавление одного нового узла или добавление ребра к уже существующему узлу. В общем случае, вставка требует изменения только одного блока в слое 0. Первым шагом необходимо найти блок для обновления. Если этот блок переполнен, то он должен быть разделен, что требует создание нового узла в слое 1. Это изменение также ограничивается одним блоком. Разделения блоков возникают довольно редко, и не оказывают сильного влияния на производительность. Если блок в крайнем левом горизонтальном слое должен быть разделен, то создается новый горизонтальный слой. Алгоритмы разделения и вставки хорошо изложены в широком круге научных работ [8,11] и в данной статье не рассматриваются.

Заключение. Полученная структура позволяет индексировать значительные объемы информации при этом, не теряя в производительности поиска. Отличие полученных результатов от существующих научных работ, состоит в том, что полученная структура сбалансирована и оптимизирована для хранения в дисковой подсистеме, сокращает количество операций ввода-ввода до минимума. Благодаря многоуровневой структуре, представляется возможность организовать эффективное кэширование путем помещения одного или нескольких слоев в оперативную память. Помещение одного слоя в оперативную память сократит количество обращений к диску при каждом запросе на 1.

Предложенный метод подходит для создания индексов в высокопроизводительных системах, обслуживающих большой наборов неоднородных, длинных и сложных строк. В то же время механизмы индексации СУБД или слабоструктурированных хранилищ данных могут обеспечить некоторую оптимизацию, но они

имеют трудности в достижении высокой производительности, которая возможна с использованием предложенного метода, особенно, если запрос является сложным или разветвленным или получает доступ к необычным объемам данных. Очевидно, что разработанный индекс представляет собой эффективный способ управления слабоструктурированными данными.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Бутакова М.А., Климанская Е.В., Янц В.И.* Мера информационного подобия для анализа слабоструктурированной информации // Современные проблемы науки и образования. – Пенза, 2013.
2. *Климанская Е.В., Чернов А.В., Янц В.И.* Методы обработки слабоструктурированных данных в автоматизированных системах на железнодорожном транспорте // Известия высших учебных заведений. Северо-Кавказский регион. Серия технические науки. – 2013.
3. *Abiteboul S.* Querying semi-structured data // In Proc. ICDT, 2001.
4. *Bertino E.* Index configuration in object-oriented databases // VLDB Journal. – 1994. – № 3 (3). – P. 355-399.
5. *Buneman P. et al.* A query language and optimization techniques for unstructured data // In Proc. SIGMOD, 1996.
6. *Chamberlain D., Robie J. and Florescu D.* Quilt: An XML query language for heterogeneous data sources // In Proc. WebDB Workshop, 2000.
7. *Comer D.* The ubiquitous B-tree // Computing Surveys. – 1979. – № 11 (2). – P. 121-137.
8. *Cooper B. and Shadmon M.* The Index Fabric: A mechanism for indexing and querying the same data in many different ways. Technical Report, 2000. DBLP Computer Science Bibliography.
9. *Deutsch A., Fernandez M. and Suciu D.* Storing semistructured data with STORED // In Proc. SIGMOD, 1999.
10. *Florescu D. and Kossmann D.* A performance evaluation of alternative mapping schemes for storing XML data in a relational database. INRIA Technical Report 3684, 1999.
11. *Alon Itai.* The JS Data Structure. Technical report, 1999.
12. *Lee W.C. and Lee D.L.* Path Dictionary: A New Approach to Query Processing in Object-Oriented Databases // IEEE TKDE. – May/June 2004. – № 10(3). – P. 371-388.
13. *Jason McHugh and Jennifer Widom.* Query Optimization for XML // In Proc. 25th VLDB, 1999.
14. *McHugh J. et al.* Lore: A Database Management System for Semistructured Data // SIGMOD Record. – 1997. – № 26 (3). – P. 54-66,
15. *Oracle Corp*. Oracle 9i database. http://www.oracle.-com/ip/deploy/database/9i/index.html.
16. *Shanmugasundaram J. et al.* Relational databases for querying XML documents: Limitations and opportunities // In Proc. 25th VLDB, 2003.
17. *Software AG.* Tamino XML database. http://www.-softwareag.com/tamino/.
18. XYZFind. XML Database. http://www.xyzfind.com.
19. *Кузнецов С.* Транзакционные параллельные СУБД: новая волна // Труды Института системного программирования РАН. – М.: Институт системного программирования РАН, 2011. – Т. 20.
20. *Zargayouna H.* Context and semantic of semi-structured documents indexing // In First COnference in Information Retrieval and Applications (CORIA'O4), March 2004.

REFERENCES

1. *Butakova M.A., Klimanskaya E.V., Yants V.I.* Mera informatsionnogo podobiya dlya analiza slabostrukturirovannoy informatsii [The measure of information of similarity for the analysis of semi-structured information], *Sovremennye problemy nauki i obrazovaniya* [Modern problems of science and education]. Penza, 2013.
2. *Klimanskaya E.V., Chernov A.V., Yants V.I.* Metody obrabotki slabostrukturirovannykh dannykh v avtomatizirovannykh sistemakh na zheleznodorozhnom transporte [Methods semi-structured data processing in automated systems of railway transport], *Izvestiya vysshikh uchebnykh zavedeniy. Severo-Kavkazskiy region. Seriya tekhnicheskie nauki* [Izvestiya vuzov. Severo-kavkazskii region. Technical Sciences, 2013.

3. *Abiteboul S.* Querying semi-structured data, *In Proc. ICDT*, 2001.
4. *Bertino E.* Index configuration in object-oriented databases, *VLDB Journal*, 1994, No. 3 (3), pp. 355-399.
5. *Buneman P. et al.* A query language and optimization techniques for unstructured data, *In Proc. SIGMOD*, 1996.
6. *Chamberlain D., Robie J. and Florescu D.* Quilt: An XML query language for heterogeneous data sources, *In Proc. WebDB Workshop*, 2000.
7. *Comer D.* The ubiquitous B-tree, *Computing Surveys*, 1979, No. 11 (2), pp. 121-137.
8. *Cooper B. and Shadmon M.* The Index Fabric: A mechanism for indexing and querying the same data in many different ways. Technical Report, 2000. DBLP Computer Science Bibliography.
9. *Deutsch A., Fernandez M. and Suciu D.* Storing semistructured data with STORED, *In Proc. SIGMOD*, 1999.
10. *Florescu D. and Kossmann D.* A performance evaluation of alternative mapping schemes for storing XML data in a relational database. INRIA Technical Report 3684, 1999.
11. *Alon Itai.* The JS Data Structure. Technical report, 1999.
12. *Lee W.C. and Lee D.L.* Path Dictionary: A New Approach to Query Processing in Object-Oriented Databases, *IEEE TKDE*, May/June 2004, No. 10(3), pp. 371-388.
13. *Jason McHugh and Jennifer Widom.* Query Optimization for XML, *In Proc. 25th VLDB*, 1999.
14. *McHugh J. et al.* Lore: A Database Management System for Semistructured Data, *SIGMOD Record*, 1997, No. 26 (3), pp. 54-66,
15. *Oracle Corp*. Oracle 9i database. Available at: http://www.oracle.-com/ip/deploy/database/9i/index.html.
16. *Shanmugasundaram J. et al.* Relational databases for querying XML documents: Limitations and opportunities, *In Proc. 25th VLDB*, 2003.
17. *Software AG*. Tamino XML database. Available at: http://www.-softwareag.com/tamino/.
18. XYZFind. XML Database. Available at: http://www.xyzfind.com.
19. *Kuznetsov S.* Tranzaktsionnye parallel'nye SUBD: novaya volna, *Trudy Instituta sistemnogo programmirovaniya RAN*. Moscow: Institut sistemnogo programmirovaniya RAN, 2011, Vol. 20.
20. Zargayouna H. Context and semantic of semi-structured documents indexing, In First COnference in Information Retrieval and Applications (CORIA'O4), March 2004.

Статью рекомендовал к опубликованию д.т.н., профессор В.И. Финаев.

Карпенко Екатерина Владимировна – Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Ростовский государственный университет путей сообщения»; e-mail: vereskunekaterina@rambler.ru; 344038, г. Ростов-на-Дону, пер. Ашхабадский 6/2, кв. 18; тел.: +79085128029; экономист первой категории.

Чернов Андрей Владимирович – Государственное бюджетное образовательное учреждение высшего профессионального образования «Ростовский государственный строительный университет»; e-mail: avche@yandex.ru; 344038, г. Ростов-на-Дону, ул. Ленина 93/2, кв. 8; тел. +79185812714; кафедра прикладной математики и вычислительной техники; д.т.н., профессор.

Янц Владимир Игоревич – e-mail: v.i.yants@pmvt.ru; 344052, г. Ростов-на-Дону, ул. Социалистическая, 162; тел.: 88632019014; кафедра прикладной математики и вычислительной техники; аспирант.

Karpenko Ekaterina Vladimirovna – Federal State-Owned Educational Establishment of Higher Vocational Education «Rostov State Transport University»; e-mail: vereskunekaterina@rambler.ru; 6/2-18, Ashkabadsky, Rostov-on-Don, 344038, Russia; phone: +79085128029; economist 1-st category.

Chernov Andrey Vladimirovich – State-Owned Educational Establishment of Higher Vocational Education «Rostov State University of Civil Engineering»; e-mail: avche@yandex.ru; 93/2-8, Lenina street, Rostov-on-Don, 344038, Russia; phone +79185812714; department of Applied Mathematics and Computing; dr. of eng. sc.; professor.

Yants Vladimir Igorevich – e-mail: v.i.yants@pmvt.ru; 162, Socialisticheskay street, Rostov-on-Don, 344052, Russia; phone: +78632019014; the department of applied informatics and computer science.