Структура OPC-trie как новый тип индекса в СУБД НИКА

В.А. Тищенко[I,II]

[I] Федеральное государственное учреждение «Федеральный исследовательский центр «Информатика и управление» Российской академии наук», г. Москва, Россия
[II] Образовательное частное учреждение высшего образования «Православный Свято-Тихоновский гуманитарный университет», г. Москва, Россия

Аннотация. Построение индексов является стандартной процедурой в любой СУБД. Однако инвертированный индекс не обеспечивает оптимальный интерактивный доступ к ключевому массиву. Для оптимального доступа к ключам при организации интерактивного интерфейса предполагается использование сжатого префиксного дерева PATRICIA-trie в качестве индекса. Сама структура PATRICIA-trie не обеспечивает оптимального доступа к ключам и требует дополнительного сжатия префиксов для получения оптимального классификатора. Результатом, предлагаемым в данной статье, является использование оптимальной структуры OPC-trie в качестве индекса. OPC-trie создается посредством дополнительного сжатия по путям PATRICIA-trie и не требует динамического сжатия префиксов.

Ключевые слова: *PATRICIA-trie, OPC-trie, вариграммы, многоуровневый индекс по лексикографическому признаку.*

DOI: 10.14357/20790279210408

Введение

Использование сжатого префиксного дерева Моррисона PATRICIA-trie [1] и технологии прерывания ветвления [2], предложенной Сассенгатом, при построении интерактивного интерфейса в виде оптимального классификатора [3] или индекса предполагает дополнительное сжатие по путям этого дерева для достижения оптимальной структуры. Выполнение динамического сжатия по путям излишне усложняет саму процедуру организации интерактивного доступа и не эффективно, поскольку требует дополнительных операций в префиксном дереве. Оптимальной структурой многоуровневого индекса является оптимальное префиксное дерево OPC-trie, определенное в [4]. При построении структуры OPC-trie исходная модель определяется в виде формулы [5,(1)] для расчета функционала общего числа операций на одном уровне индекса $S_{\text{оп}}(n,n_g)$, зависящего от максимального числа вершин в классах n и числа вершин в группах n_g, на которые разбиваются классы. Над расчетным уровнем классификатора надстраивается однобуквенный или двухбуквенный верхний уровень индекса. В общем случае число уровней классификатора над ключевым уровнем массива может быть более двух. Оценку числа уровней можно делать на основе равномерного случая распределения ключей по префиксам [6].

1. Процедура построения оптимальной структуры OPC-trie

Для оптимизации цифрового поиска Сассенгат первый предложил использовать «смешанную стратегию» [2], т.е. двигаться по префиксному дереву с последующим переходом на список ключей посредством прерывания ветвления, которое он выбирал достаточно произвольно так, чтобы список состоял из шести ключей. В общем случае такой выбор точки прерывания ветвления не приводит к оптимальной структуре, определенной в [4]. Сжатое префиксное дерево PATRICIA-trie [1] также не оптимально в смысле [4] и является исходной точкой построения оптимального префиксного дерева OPC-trie посредством дополнительного сжатия префиксов по путям PATRICIA-trie. При построении очередного уровня многоуровневого индекса необходимо двигаться от листьев к корню дерева PATRICIA-trie. Точки ветвления в PATRICIA-trie, в которых выполнены неравенства:

$$n(s_0)>n^*,\ n(s)\le n^*, \qquad (1)$$

где префикс s является непосредственно подчиненным префиксу s_0, а $n(s_0)$ и $n(s)$ – это число листьев, подчиненных соответствующим префиксам, будут точками ветвления в дереве OPC-trie [3]. Все префиксы, подчиненные префиксам, в которых выполнено только второе неравенство из (1), переходят в OPC-trie без ветвления. Процедуру построения OPC-trie можно сформулировать в виде следующего алгоритма.

1 шаг. Проход от листьев PATRICIA-trie вверх до префиксов, в которых выполнено условие (1), определяющие точки ветвления в OPC-trie.

2 шаг. Перенос полученных точек ветвления в OPC-trie посредством добавления префиксов, подчиненных префиксам, в которых определены точки ветвления.

3 шаг. Отсечение ветвей PATRICIA-trie, начиная от префиксов, в которых выполнено условие (1).

4 шаг. Проверка условия, что число добавленных префиксов на Шаге 2 составляют один класс, т.е. их не более n^*.

5 шаг. Если условие на шаге 4 выполнено, то завершение алгоритма, в противном случае переход к Шагу 1.

В результате первой итерации ключевой уровень массива будет надстроен одним уровнем префиксов, для которых выполняется условие (1). Эти префиксы будут также точками прерывания ветвления с переходами на соответствующие списки ключей. Число ключей в списках ограничено величиной n^*. Далее описанная процедура выполняется итерационно. На очередной итерации берется за исходное тоже сжатое дерево PATRICIA-trie,, но только до узлов ветвления, в которых выполняется условие (1). Таким образом, эти узлы становятся листьями в PATRICIA-trie. В этом дереве аналогично берутся точки ветвления, удовлетворяющие условию (1) и переносятся в оптимальное дерево OPC-trie. В результате подчиненный уровень префиксного дерева OPC-trie будет надстроен верхним уровнем префиксов. Эти префиксы образуют классы ключей, включающие подклассы, образованные префиксами подчиненного уровня. Итерационная процедура построения OPC-trie завершается, когда число всех оставшихся префиксов, которым подчинены все ключи массива, не превышает n^* и эти префиксы образуют один базовый класс в OPC-trie. Оптимальное сжатое по путям префиксное дерево OPC-trie можно рассматривать как иерархию классов ключей, для каждого из которых выполнено условие (1) так, что каждый класс содержит максимальное, но не превышающее n^* число непосредственно подчиненных префиксов или ключей [3,4].

2. Число уровней в индексе OPC-trie

Число уровней в индексе определяется, исходя из процедуры построения индекса, на верхнем уровне которого префиксы составляют один класс, что и является условием завершения процедуры построения [3]. В равномерном случае распределения [6] длина префикса на последнем уровне индекса определяется как $k_a=\log_a N$ и $k_{n^*}=\log_{n^*} N$ для PATRICIA-trie и OPC-trie соответственно. Здесь $a=|A|$ – мощность алфавита A. При $a\le n^*$ длина префикса $k_a \ge k_{n^*}$. В неравномерном случае средние длины ключей будут не менее, чем k_a и k_{n^*}. В предположении, что в равномерном случае на каждом уровне индекса к префиксу добавляется по одной букве величины k_a и k_{n^*} будут соответствовать числу уровней в индексе в равномерном случае. При оценке числа уровней в неравномерном случае можно отталкиваться от этих величин.

3. Схема описания данных многоуровневого индекса

На рис.1 представлено описание данных для индекса AttributeArray с текстовым ключом key_A по атрибуту Attribute объекта el_o, являющегося элементом массива ObjectArray. Шаблон ObjectPattern осуществляет переход по ссылке на основной массив объектов ObjectArray. Рекурсивный шаблон CyclicLettersPattern позволяет разбить текстовый атрибут на отдельные буквенные префиксы, создавая тем самым префиксное дерево из произволь-

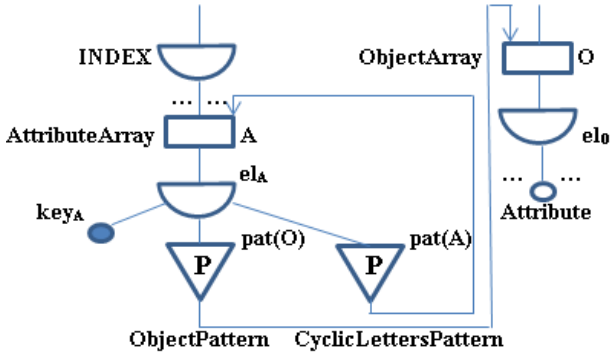

Рис.1. Концептуальная схема БД НИКА для многоуровневого индекса

Замечание. Терминальная вершина Attribute может находиться на любом подчиненном уровне массива ObjectArray.

ного переменного числа уровней. Массив A может быть описан в виде:

$$T(A)=\{[T(key_A),T(pat(O)),\mathrm{T}(pat(A))]\}. \quad (2)$$

$T(A)$ состоит из типа ключа $\mathrm{T}(key_A)$ и сложных типов: $T(pat(O))$ – ссылки на шаблон массива O и $T(pat(A))$ – ссылки на шаблон массива A. Такая схема описания данных подходит для любого префиксного дерева, в том числе, и для исходного массива (вырожденный случай префиксного дерева). Таким образом, схема на рис.1 и тип данных (2) в БД НИКА являются одинаковыми для сжатого префиксного дерева PATRICIA-trie и для оптимального префиксного дерева OPC-trie.

4. Отклонение от исходной модели

Над расчетным уровнем индекса надстраивается уровень, состоящий из одограмм или биграмм, в предположении, что мощность соответствующих классов префиксов не превышает значения оптимального параметра индекса n^*. Это предположение может нарушаться. В случае ключевого уровня массива по полю ФИО, например, N=34 657[1] число префиксов второго уровня OPC-trie на однобуквенный префикс “К” составляет $n_к^{(2)}$=187 (см. табл.1) и n^*=176. В этом случае $n_к^{(2)}>n^*$. Формально это означает, что на первом уровне классификатора вместо буквы “К” нужно вынести N-граммные префиксы: “Ка”,“Кв”,“Ке”,…,“Ксен”,…,“ Кьян ”, ...

Табл.1

Частоты однобуквенных префиксов $n_L^{(2)}$ на втором уровне OPC-trie для поля ФИО

А	Б	В	Г	Д	Е	Ж
90	99	99	98	64	25	10
З	И	К	Л	М	Н	О
36	59	187	85	118	77	26
П	Р	С	Т	У	Ф	Х
134	58	174	47	19	18	9
Ц	Ч	Ш	Щ	Э	Ю	Я
8	34	41	4	9	15	18

Из табл.1 видно, что для однобуквенных префиксов число непосредственно подчиненных префиксов изменяется в диапазоне $4\le n_L^{(2)}\le 174$ и $n_L^{(2}\le,n^*$, если не учитывать префикс “К”. Таким образом, на практике этим незначительным отклонением можно пренебречь. В вырожденном случае некоторые одограммы “Щ”($n_{щ}$=156≤,n^*),”Э”($n_э$=23≤,n^*),”Ю”($n_ю$=156≤,n^*) не требуют разделения на подклассы и образуют один класс сразу с переходом на ключевой уровень массива. В редких случаях на расчетном уровне могут получаться все (или почти все) префиксы одинаковой длины, например, биграммы на буквы “Х”,”Ц”. В остальных случаях расчетный уровень классификатора содержит вариграммы.

[1] В случае больших или меньших объемов характер отклонений будет таким же. Другие объемы рассмотрены в следующем пункте.

5. Влияние объема массива на процедуру построения оптимального классификатора

Табл.2

Глубина префиксного дерева при разных объемах ключевого массива

N	$S_{оп}^*$ / N	n^*	$\log_{n^*}$ N
1	2	3	4
34 657	14,57	176	2,02
103 823	17,95	333	1,99
923 915	31,15	619	2,14
4 216 674	44,74	1 078	2,18
11 655 046	58,40	1 259	2,28

В первой колонке табл. 2 указан объем массива, во второй – оптимальное среднее число операций на один ключ, в третьей – максимальное число ключей или префиксов в классе, в четвертой – глубина префиксного дерева в равномерном случае распределения ключей по префиксам. Неравномерный общий случай дает при том же объеме ключей глубину префиксного дерева не меньшую в среднем, чем в равномерном случае. С ростом числа ключей N в исходном массиве и максимального числа ключей в классе n^* в оптимальном случае наблюдается тенденция к возрастанию глубины префиксного дерева. Превышение значения 2 в десятых долях говорит о том, что среди классов префиксов на втором уровне классификатора будут появляться классы, в которых число префиксов превышает n^* и при объемах ключей более 1 млн следует использовать общую формулу для расчета функционала общего числа операций $S_{оп}$ (а не формулу для расчета одного уровня).

6. Решение вопроса о способе построения первого уровня классификатора

Глубина префиксного дерева OPC-trie – это значение для равномерного случая распределения. Реальные случаи распределения рассмотрены в табл. 3. В последней колонке приведены частоты префиксов первого уровня относительно второго, для которых нарушается модель: $n_L\le,n^*$.

Табл.3

Префиксы верхнего уровня классификатора с нарушением исходной модели

N	n^*	Префиксы: $n_к > n^*$
34 657	176	К:187
103 823	333	К:339
923 915	619	А:859, Б:1 133, Г:904, К:1 190, М:1 202, П:1 027, С: 1 522
4 216 674	1 078	А:2 256, Б:3 050, В:1 400, Г:2 637,, Д:1 546, К:5 460, Л:1 731,М:3 300, Н:1 151, П:2 753, Р:1 254, С:3 982, Т:1 652, Ш: 1 314
11 655 046	1 259	А:4 508, Б:6 945, В:3 092, Г:5 675,, Д:3 255, Е:1 921:, З:2 403:, И:1 617:, К:12 335, Л:13 853, М:6 533, Н:2 548, О:1 358:, П:6 121, Р:3 046, С:8 795, Т:3 410, Ф:2 077:, Х:1 763:, Ч:2 163:, Ш: 3 398

Из табл. 3 видно, что для объема массива порядка 1 млн ключей уже недостаточно надстраивать второй уровень однобуквенными префиксами. У семи префиксов наблюдается превышение значения n^*, причем для префикса “С” $n_к > n^*$ почти в 2,5 раза.. Такими отклонениями пренебречь нельзя. В данном случае возможно надстроить второй оптимизированный уровень классификатора двухбуквенными префиксами. Такой классификатор не сильно будет отличаться от полностью оптимального классификатора, т.к. случаев превышения n^* не много и они не очень велики. Для объема массива в несколько миллионов ключей префиксы с превышением оптимального значения n^* составляют около половины алфавита (14 букв). Здесь также можно взять в качестве первого уровня биграммы, но отклонения от оптимального случая уже возможны и здесь, например, для префикса “Ка”: $n_{Ка} > n^* = 1\ 078$. При объеме более 10 млн ключей префиксы с превышением оптимального значения n^* составляют более половины алфавита (21 буква) и их частоты превышают n^* в несколько раз: для префикса “К”– $n_к/n^* \approx 10$. Здесь требуется расчет оптимальных параметров с использованием функционала общего числа операций $S_{оп}$ с учетом всех уровней классификатора, а не только для одного уровня. Сказанное можно свести к табл. 4.

Табл.4

Способ построения верхнего уровня классификатора в зависимости от объема массива

Объем ключевого массива	Вид 1-ого уровня классификатора	
Менее 1 млн ключей	Одногаммы	Функционал $S_{оп}$ для одного уровня
Порядка 1 млн ключей	Биграммы	Функционал $S_{оп}$ для одного уровня
Несколько млн ключей	Биграммы	Функционал $S_{оп}$ для одного уровня
Свыше 10 млн ключей	Вариграммы[2]	Функционал $S_{оп}$ в общем случае

В первых трех случаях в табл. 4 применяется формула [5,(1)] функционала $S_{оп}$ в частном случае для расчета одного уровня классификатора, который надстраивается еще одним уровнем классификатора в виде префиксов из одногамм или биграмм. Этот уровень не является расчетным. В последнем случае (объем массива свыше 10 млн ключей) предполагает расчет функционала $S_{оп}$ по общей формуле [5,(5)]. Здесь все уровни классификатора являются расчетными и состоят из вариграмм.

Заключение

Среди современных применений префиксного дерева можно перечислить хеш-карты на основе префиксного дерева [7], метод защиты от атак посредством возвратно-ориентированного программирования с использованием графа префиксного дерева [8], построение префиксного хеш-дерева без блокировок [9], хранение множеств и запросы на множества на основе префиксного дерева множеств [10]. В работе [10] рассматривается эффективность алгоритмов для множества содержащих операций посредством теоретического и эмпирического анализа, получены верхние границы временной сложности алгоритмов. Практические эксперименты показывают, что операции в префиксном дереве быстрее, чем операции в инвертированном индексе на порядок. Наряду с перечисленными направлениями исследований, не менее актуальным является применение префиксного дерева для организации интерактивного доступа к ключевому массиву посредством использования индекса в виде OPC-trie, который позволяет работать с ключевым массивом быстрее, чем инвертированный индекс и неоптимальные классификаторы. Для иллюстрации этого в табл. 5 приведен пример нахождения ключа для различных классификаторов где тип классификатора – тип интерактивного интерфейса с пользователем, m_i – число нажатий или шагов пользователя для перехода на данный ключ, m_i/m^* – число раз, во сколько оптимальный классификатор быстрее, чем i-ый.

[2] N-граммы переменной длины.

Табл. 5

Сравнение разных классификаторов при нахождении ключа «Смирнов Константин Васильевич»

i	Тип классификатора	m_i	m_i / m^*
1	Список (инвертированный индекс)	1 462	292
2	Однобуквенный с 1-им уровнем	97	19
3	Однобуквенный с 2-мя уровнями	10	2
4	Двухбуквенный, многоуровневый	20	4
5	Оптимальный	5	1

В статье изложены результаты исследований, заключающихся в формализации, постановке задачи, разработке методов и алгоритмов решения задачи оптимизации интерактивного доступа к ключевому массиву [3–6], а также реализации данных методов в виде оптимальных классификаторов в рамках гипертекстовой системы NKWSystem для ООСУБД НИКА [3], позволяющей визуализировать, трансформировать и анализировать информацию на основе методов обработки информации таких, как методов представления ключевого уровня массива в виде префиксных деревьев trie.

Результаты исследований, изложенные в данной статье, применяются в системе о репрессированных в годы советской власти http://martyrs.pstbi.ru для построения классификаторов для индексов по строковым полям.

Литература

1. *Morrison, D.* PATRICIA-practical algorithm to retrieve information coded in alphanumeric / D. Morrison // J. ACM 15,4. 0ct. 1968. P. 514-534.
2. *Sussenguth E.H.* Use tree structures for processing files / E.H. Sussenguth // CACM 6. 1963. P. 272-279.
3. *Арлазаров В.Л.* Устройство отыскания информации по ключевым словам / В.Л. Арлазаров, В.А. Тищенко // Патент на изобретение № 2679967 C1 Российская Федерация. 2019. Бюл. № 5.
4. *Тищенко В.А.* OPC-trie: спецификация оптимального классификатора для СУБД НИКА // Труды ИСА РАН. 2021. Т. 71. Вып. 1. С. 67-71.
5. *Тищенко В.А.* Выбор оптимального алфавитного классификатора при минимизации общего числа операций // Труды ИСА РАН. 2018. Т. 68. № 1. С.54-57.
6. *Тищенко В.А.* Идеальный случай классификатора по лексикографическому признаку при равномерном распределении ключей по префиксам / В.А. Тищенко // Материалы XXXIV Международной научно-практической конференции “Eurasiascience”, 31 декабря 2020 года. С.108-109.
7. *Areias M.* On the correctness and efficiency of a novel lock-free hash trie map design / Areias M., Rocha R.// JPDC,150, April 2021. P. 184-195.
8. *Zhu A.Y.C.* ROP Defense Using Trie Graph for System Security / Zhu A.Y.C., Yan W.Q., Sinha R. // IJDCF, 13,6. 2021. P. 1-12.
9. *Moreno P.* On the Implementation of Memory Reclamation Methods in a Lock-Free Hash Trie Design / Moreno P., Areias M., Rocha R.// JPDC,155, 2021, P.1-13.
10. *Savnik I.* Data structure set-trie for storing and querying sets: Theoretical and empirical analysis / Savnik I., Akulich M., Krnc M. // PLoS ONE 16(2). 2021.

Тищенко Владимир Александрович. Федеральное государственное учреждение «Федеральный исследовательский центр «Информатика и управление» Российской академии наук», г. Москва, Россия. Научный сотрудник. Сотрудник кафедры Информатики ПСТГУ. Количество печатных работ: 25. Область научных интересов: средства создания и поддержки электронных библиотек и электронных изданий. E-mail: vtischenko@isa.ru

OPC-trie structure as a new type of index in NIKA DBMS

V.A. Tishchenko[I,II]

[I] Federal Research Center "Informatics and control" of The Russian Academy of Sciences, Moscow, Russia

[II] St. Tikhons"s Orthodox University, Moscow, Russia

Abstract. Index building is a standard procedure in any DBMS. However, an inverted index does not provide optimal interactive access to the key array. For optimal access to keys when organizing an interactive interface, the article [5] assumes the use of a compressed prefix tree PATRICIA-trie as an index. The PATRICIA-trie structure itself does not provide optimal access to keys and requires additional prefix compression to obtain an optimal classifier. The result suggested in this article is the use of an optimal OPC-trie structure as an index. OPC-trie is created by additional compression along PATRICIA-trie paths and does not require dynamic compression of prefixes.

Keywords: *PATRICIA-trie, OPC-trie, varigrams, multilevel lexicographic index.*
DOI: 10.14357/20790279210408

References

1. *Morrison, D.* PATRICIA-practical algorithm to retrieve information coded in alphanumeric / D. Morrison // J. ACM 15,4, 0ct. 1968, P.514-534.
2. *Sussenguth, E.H.* Use tree structures for processing files / E.H. Sussenguth // CACM 6, 1963, P.272-279.
3. *Arlazarov V.L.* Device for finding information by keywords / V.L. Arlazarov, V.A. Tishchenko // Patent for invention No. 2679967 C1 Russian Federation, 2019. Bul. No. 5
4. *Tishchenko V.A.* OPC-trie: specification of an optimal classifier for the NIKA DBMS // Proceedings of ISA RAS, 2021, 71, No. 1. P.67-71.
5. *Tishchenko V.A.* The choice of the optimal alphabetical classifier while minimizing the total number of operations // Proceedings of ISA RAS, 2018. V. 68. No. 1. P.54-57
6. *Tishchenko, V.A.* The ideal case of a lexicographic classifier with a uniform distribution of keys by prefixes / V.A. Tishchenko // Materials of the XXXIV International Scientific and Practical Conference "Eurasiascience", December 31, 2020 pp.108-109.
7. *Areias M.* On the correctness and efficiency of a novel lock-free hash trie map design / Areias M., Rocha R.// JPDC,150, April 2021, P. 184-195.
8. *Zhu A.Y.C.* ROP Defense Using Trie Graph for System Security / Zhu A.Y.C., Yan W.Q., Sinha R. // IJDCF, 13,6, 2021. P.1-12.
9. *Moreno P.* On the Implementation of Memory Reclamation Methods in a Lock-Free Hash Trie Design / Moreno P., Areias M., Rocha R.// JPDC,155, 2021, P.1-13.
10. *Savnik I.* Data structure set-trie for storing and querying sets: Theoretical and empirical analysis / Savnik I., Akulich M., Krnc M. // PLoS ONE 16(2), 2021.

Tishchenko Vladimir Alexandrovich. Researcher, ISA FRC CSC RAS. Employee of department of Informatics, PSTGU. Graduated from the MEPhI in 1993. Number of publications: 25. Research interests: means of creation and support of electronic libraries and electronic publications. E-mail: vtischenko@isa.ru