УДК 004.066
DOI: 10.25559/SITITO.16.202004.824-832

Оригинальная статья

Индексирование значений нечёткой переменной

Н. К. Самойлов
ФГБОУ ВО «Воронежский государственный университет», г. Воронеж, Российская Федерация
394018, Российская Федерация, г. Воронеж, Университетская площадь, д. 1
nk.samoylov@gmail.com

Аннотация
В настоящее время, как в научных работах, так и в программных проектах широко используются нечеткие данные, лингвистические переменные и т.д. Однако в современных СУБД отсутствует поддержка операций над нечеткими данными и средства индексирования этих данных. Предлагается решение задачи индексирования набора нечетких значений лингвистической переменной. Нечеткие переменные представляются в виде замкнутых плоских ломаных или полигонов. Для оптимизации обработки строится проекция полигона, которая будет храниться в иерархическом индексе. Предложена модифицированная структура R-дерева, которая оптимизирована для хранения нечетких значений. Внутренние узлы дерева хранят минимальный ограничивающий интервал для всех дочерних узлов, листовые узлы, кроме того хранят сам полигон нечеткого значения и полигоны соответствующие заданным альфа-уровням. Листовые узлы имеют ссылки на соседние листовые узлы для выполнения операций больше, меньше, неравно. Предложены модифицированные алгоритмы вставки и поиска в модифицированном R-дереве. Проведено тестирование предложенных алгоритмов, которое показало не худшее время выполнения по сравнению со стандартными алгоритмами R-дерева.

Ключевые слова: нечеткая переменная, лингвистическая переменная, замкнутая ломаная, R-дерево, алгоритм, листовой узел, внутренний узел, индекс.

Автор заявляет об отсутствии конфликта интересов.

Для цитирования: Самойлов, Н. К. Индексирование значений нечёткой переменной / Н. К. Самойлов. – DOI 10.25559/SITITO.16.202004.824-832 // Современные информационные технологии и ИТ-образование. – 2020. – Т. 16, № 4. – С. 824-832.

Original article

Indexing Values of a Fuzzy Variable

N. K. Samoylov
Voronezh State University, Voronezh, Russian Federation
1 Universitetskaya pl., Voronezh 394018, Russian Federation
nk.samoylov@gmail.com

Abstract

Currently, fuzzy data, linguistic variables, etc. are used. However, modern DBMS lacks support for operations on odd data and means of indexing this data. A solution to the problem of indexing a set of fuzzy values of a linguistic alternative is proposed. Fuzzy variables are represented as closed planar polylines or polygons. To optimize processing, a polygon projection is built, which will be stored in a hierarchical index. A modified R-tree structure is proposed, which is optimized for storing fuzzy values. Internal tree nodes store the minimum bounding spacing for all child nodes, leaf nodes, in addition, store the fuzzy value polygon itself and the polygons corresponding to the specified alpha levels. Leaf nodes have links to neighboring leaf nodes to perform more, less, unequal operations. Modified algorithms for insertion and search in a modified R-tree are proposed. Testing of the proposed algorithms was carried out, which showed not the worst execution time compared to standard R-tree algorithms.

Keywords: fuzzy variable, linguistic variable, closed polyline, R-tree, algorithm, leaf node, internal node.

The author declares no conflicts of interest.

For citation: Samoylov N.K. Indexing Values of a Fuzzy Variable. S*ovremennye informacionnye tehnologii i IT-obrazovanie* = Modern Information Technologies and IT-Education. 2020; 16(4):824-832. DOI: https://doi.org/10.25559/SITITO.16.202004.824-832

Введение

Подавляющее большинство СУБД используют языки запросов (язык SQL наиболее распространенный), которые определяют результирующий набор данных, исходя из условий, на соответствие которым проверяются данные в процессе поиска. Эти условия должны быть точно заданы, поскольку точность является основным требованием при определении условий в языках запросов. Например, покупатель автомобиля, ищущий дешевую машину, должен точно определить диапазон цен. Независимо от того как определены границы диапазона, автомобиль с ценой незначительно превышающей установленный лимит не будет соответствовать условиям запроса. Это показывает, что указанные ограничения являются результатом необходимости точного определения условий, которые изначально были выражены естественным языком и с неточными терминами. Проблему можно решить с помощью лингвистических терминов, которые моделируются лингвистическими переменными, и обрабатывается в запросах, адресованных в базу данных. Это так называемые нечеткие запросы. Более того, данные хранящиеся в базе данных также могут быть нечеткими, например, можно хранить значения лингвистической переменной. Эти значения могут быть представлены в виде многоугольников на плоскости и представлять пространственные данные.

Для индексирования пространственных данных в СУБД используются различные модификации R-дерева [1] (R-tree, R+-tree, R*-tree) или пространственные индексы. R-дерево предоставляет методы динамического доступа к данным, которые организованы в иерархию прямоугольников. Эта структура данных позволяет выполнять операции вставки, удаления и поиска в определенном прямоугольнике.

Несмотря на то, что R-дерево было описано во многих статьях и монографиях по базам данных, необходимо коротко описать его свойства. R-дерево предназначено для структурирования множества n-мерных геометрических объектов, используя для этого минимальный ограничивающий n-мерный прямоугольник [1] (minimum bounding rectangle - MBR). Каждому узлу в R-дереве соответствует MBR, который ограничивает все его дочерние узлы. Листовые узлы дерева содержат ссылки на конкретные объекты в БД вместо ссылок на дочерние узлы.

Нужно заметить, что минимальные ограничивающие прямоугольники, которые содержат различные узлы, могут перекрываться. Иначе говоря, MBR может содержаться в нескольких узлах (в геометрическом смысле), но принадлежать будет только одному узлу. Это значит, что в процессе пространственного поиска можно посетить множество узлов, прежде чем будет найден необходимый MBR. Также несложно увидеть, что представление геометрических объектов в виде MBR может привести в процессе поиска к получению некорректных данных в результирующей выборке. Следовательно, R-дерево должно обеспечивать механизм фильтрации выборки, чтобы уменьшить затраты на последовательные проверки геометрических объектов.

R-дерево порядка (m, M) [2] имеет следующие характеристики:

– каждый листовой узел (не корень) может содержать m записей, где $M < m \le \frac{M}{2}$. Каждая запись имеет вид (mbr, oid), где mbr - MBR, который ограничивает геометрический объект записи, a oid - идентификатор записи;
– каждый внутренний узел может содержать m записей, где $M < m \le \frac{M}{2}$. Каждая запись имеет вид (mbr, p), где p - ссылка на дочерний узел и mbr - MBR, который ограничивает все минимально ограничивающие прямоугольники дочернего узла;
– минимальное число записей в корневом узле 2, если корневой узел не лист (в этом случае, он может содержать 0 или 1 запись);
– все листовые узлы R-дерева находятся на одном уровне.

Пусть R-дерево хранит N прямоугольников, тогда максимальное значение высоты можно получить как: $h_{\max} = \lceil \log_m N \rceil - 1$. Максимальное число узлов можно представить как сумму максимально возможного числа узлов на каждом уровне: $\sum_{i=1}^{h_{\max}} \left\lceil \frac{N}{m^i} \right\rceil = \left\lceil \frac{N}{m} \right\rceil + \left\lceil \frac{N}{m^2} \right\rceil + \cdots + 1$

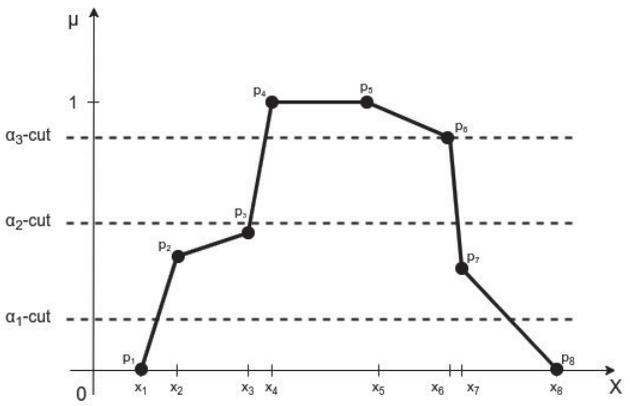
Значения лингвистической переменной [3] определяются характеристическими функциями. Для каждого значения характеристическая функция μ(x) определяется на интервале (x_1, x_2), значения характеристической функции лежат на отрезке [0, 1]. Таким образом, нечеткую переменную можно представить как аппроксимацию характеристической функции полигоном, одна из сторон которого лежит на оси OX. Подобное представление позволяет использовать R-деревья для индексирования нечетких переменных. Нужно отметить, что в настоящее время в СУБД отсутствуют механизмы для индексирования наборов нечетких переменных, что ограничивает их использование.

В данной статье предложена модификация структуры R-дерева для хранения и обработки нечетких переменных, представленных многоугольниками. Использование индексов в СУБД на основе модифицированной структуры R-дерева позволит не только хранить значения нечетких переменных, но и повысить производительность их обработки.

Цель исследования

Лингвистическая переменная (ЛП) характеризуется набором $\{X, T(X), U, G, M\}$ [4], в котором X - название переменной, T(X) - множество названий лингвистических значений переменной X, каждое из этих значений является нечеткой переменной $\tilde{x}$ со значениями из универсального множества U, G - синтаксическое правило порождающее названия $\tilde{x}$ для значений переменной X, M - семантическое правило, которое ставит в соответствие каждой нечеткой переменной $\tilde{x}$, ее смысл $M(\tilde{x})$, то есть нечеткое подмножество $M(\tilde{x})$ универсального множества[1] U [5]-[7]. Пусть $M(\tilde{x})$ - плоская замкнутая ломаная без самопересечений, любые два соседних звена которой не лежат на одной прямой, $M(\tilde{x}) = \{[p_i, p_j] \mid p_i = (x_i, y_i), p_j = (x_j, y_j), p_1 = p_{n+1}, i = 1 \ldots n+1\}$, рис. 1.

[1] *Рыжов А.П.* Элементы теории нечетких множеств и измерения нечеткости. М.: Диалог-МГУ, 1998.



Р и с. 1. Пример лингвистической переменной

F i g. 1. Linguistic variable example

Для хранения значений нечетких переменных, представленных как плоский полигон, в базе данных можно использовать тип данных GeoJSON[2]. Для выполнения запросов по GeoJSON данным (пространственные запросы) в СУБД используются различные виды R-tree индексов (пространственные индексы) [8], [9]. Как правило, СУБД предоставляют следующий набор операций поиска над пространственными данными [10]:

- выбор объектов, которые содержатся в заданном полигоне;
- выбор объектов, которые пересекаются заданным полигоном;
- выбор объектов, расстояние до которых от заданной точки или сферы не превышает заданную дистанцию.

Но этих операций недостаточно для выполнения поиска по набору нечетких значений.

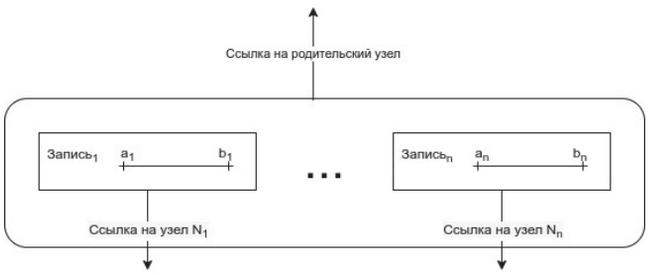
Индекс на основе R-дерева подходит для индексирования значений нечетких переменных, но нуждается в модификации.

Постановка задачи: разработать модель структуры дерева для индекса в базе данных и алгоритмы его функционирования. Индекс должен поддерживать следующие операции сравнения:

- нечетко-равно;
- нечетко-больше/меньше;
- нечетко-неравно.

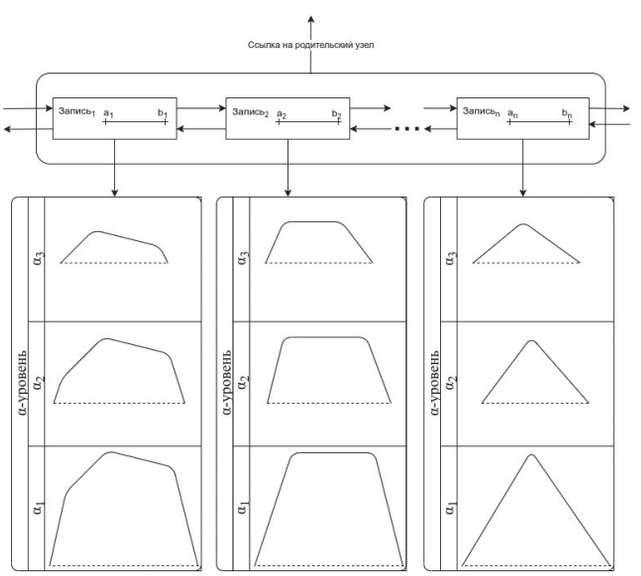
Дерево интервалов нечетких значений

Поскольку значения нечетких переменных располагаются в универсальном множестве $U = \{(x, y) | x \in R, 0 < y \le 1\}$, для их индексации можно не использовать каноническую реализацию R-дерева на плоскости, достаточно использовать версию R-дерева на прямой или дерево интервалов [11], [12] – R_1 - дерево. При этом вместо многоугольников дерево будет хранить отрезки, которые являются проекциями многоугольников на ось OX [13].



Р и с. 2. Внутренний узел R_1-дерева

F i g. 2. Internal node of the R_1-tree



Р и с. 3. Листовой узел R_1^{α} - дерева

F i g. 3. Leaf node of the R_1^{α} - tree

Пусть m – порядок R_1-дерева, тогда внутренний узел $Node_{\text{int}} = \{K_{\text{int}}, \text{Re}\, f_{parent}\}$, где $K_{\text{int}} = \left\{k_{\text{int}}^i \middle| \frac{m}{2} - 1 \le i \le m - 1\right\}$ – множество ключей внутреннего узла, $\text{Re}\, f_{parent}$ – ссылка на родительский узел. Пусть $\neq_{\tilde{x}}^i$ проекция $M(\tilde{x}_i)$ на OX, тогда ключ $k_{\text{int}} = \{\cup\pi_{\tilde{x}}^i, \text{Re}\, f_{\text{int}}^i\}$, где $\cup\neq_{\tilde{x}}^i$, - объединение всех проекций дочерних узлов или минимальный ограничивающий интервал [14], [15] (МОИ), $\text{Re}\, f_{\text{int}}^i$ – ссылка на i-тый дочерний узел, рис. 2. Пусть для $M(\tilde{x})$ задан набор α-уровней $A = \{\alpha_i | i = 0 \ldots n, \forall i > j, \alpha_i > \alpha_j, 0 \le \alpha_i < 1\}$, тогда R_1^{α} – дерево, которое позволяет хранить и обрабатывать нечеткие значения, для которых задан набор A. Пусть проекция $\pi_{\tilde{x}}^i = (x_{\min}, x_{\max})$, для каждого α_i определим прямоугольник $s_i = \{(x_{\min}, \alpha_i), (x_{\max}, \alpha_i), (x_{\max}, 0), (x_{\min}, 0)\}$, тогда $\tilde{q}_{\alpha}^i$ - отсечение $\tilde{x}$ уровня α_i, $\tilde{q}_{\alpha}^i = M(\tilde{x}) \setminus q_i$, $Q_{\tilde{x}} = \{\tilde{q}_{\alpha}^i | i = 0 \ldots n\}$ – множество всех отсечений $\tilde{x}$. Пусть $m(\pi_{\tilde{x}}^i) = \frac{x_{\min} + x_{\max}}{2}$ – середина

[2] *Butler H., Daly M., Doyle A., Gillies S., Hagen S., Schaub T.* The GeoJSON Format. RFC 7946. IETF Trust, 2016. [Электронный ресурс]. DOI: https://doi.org/10.17487/RFC7946

интервала проекции [16], [17].

Листовой узел R_1^α – дерева определяется как $Node_{leaf} = \{K_{leaf}, \mathrm{Re}\, f_{parent}\}$ рис. 3, где $K_{leaf} = \left\{ k_{leaf}^i \middle| \frac{m}{2} - 1 \le i \le m - 1 \right\}$ – множество ключей листового узла, $\mathrm{Re}\, f_{parent}$ – ссылка на родительский узел, рис. 3. Ключ $k_{leaf}^i = \{M(\tilde{x}_i), \pi_{\tilde{x}}^i, \mathrm{Re}\, f_{prev}, \mathrm{Re}\, f_{next}, Q_{\tilde{x}}\}$, где $\mathrm{Re}\, f_{prev}$ – ссылка предыдущий ключ, $\mathrm{Re}\, f_{next}$ – ссылка на следующий ключ. Множество K_{leaf} упорядочено, $k_{leaf}^i \succ k_{leaf}^j$, если $m(\pi_{\tilde{x}}^i) > m(\pi_{\tilde{x}}^j)$.

Модифицированный алгоритм для работы с -деревом

Модифицированный алгоритм обладает следующими особенностями:

1. Каждая запись во внутреннем узле дерева хранит не минимальный ограничивающий прямоугольник, а минимальный ограничивающий интервал;
2. Записи в узлах дерева упорядочены по возрастанию середин интервалов;
3. Записи в листовые узлы хранят, кроме минимального ограничивающего интервала, конечный набор полигонов нечеткого значения, которые соответствуют конечному набору различных значений альфа-уровня;
4. Записи в листовых узлах дерева образуют двусвязный список.

Алгоритм вставки полигона $M(\tilde{y})$ в R_1^α-дерево состоит из следующих шагов, рис. 4:

1. Получить проекцию $\neq_{\tilde{y}}$;

2. Рекурсивно найти листовой узел $Node_{leaf}$ для вставки нового ключа [18], [19], в котором $\exists k_{leaf}^i$, такой что $\left| m(\pi_{\tilde{y}}) - m(\pi_{\tilde{x}}^i) \right| \to \min$;
3. Создать объект ключа $k_{\tilde{y}}$, который содержит ссылки на объекты $M(\tilde{y}_i), \neq_{\tilde{y}}, Q_{\tilde{y}}$;
4. Вставить $k_{\tilde{y}}$ в $Node_{leaf}$:

– Найти позицию $p_{\tilde{y}}$ для вставки, в упорядоченном списке ключей;
– Вставить $k_{\tilde{y}}$ в позицию $p_{\tilde{y}}$;

– Проставить ссылки на соседние записи $\mathrm{Re}\, f_{prev}$ и $\mathrm{Re}\, f_{next}$ независимо от того, находятся ли они в узле $Node_{leaf}$ или в соседних узлах.

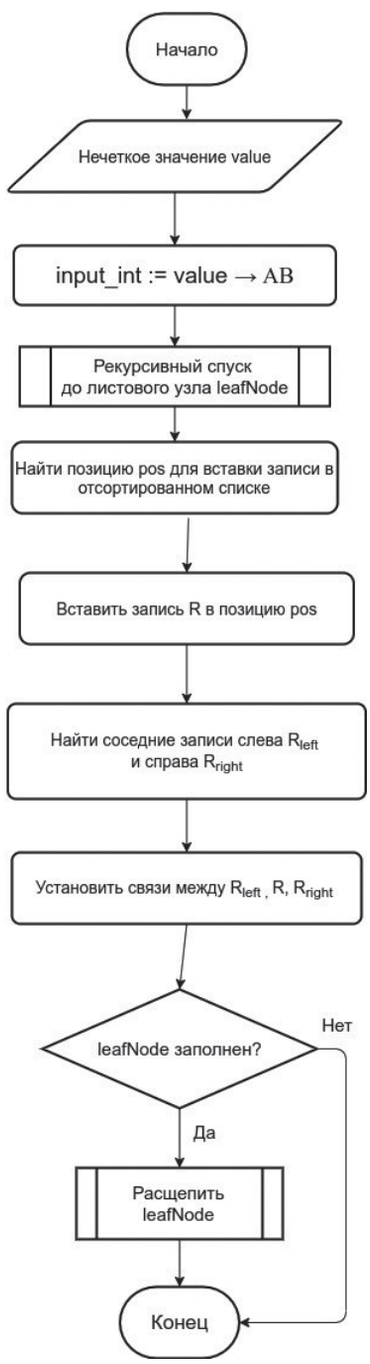
5. Если узел $Node_{leaf}$ заполнен, то выполнить процедуру расщепления.

Алгоритм поиска $M_\alpha(\tilde{v})$ - значения нечеткой переменной $\tilde{v}$ по уровню α в R_1^α-дереве состоит из следующих шагов, рис. 5:

1. Создать контейнер Δ для списка значений результата;
2. Получить проекцию $\neq_{\tilde{v}}$ полигона на ось OX;

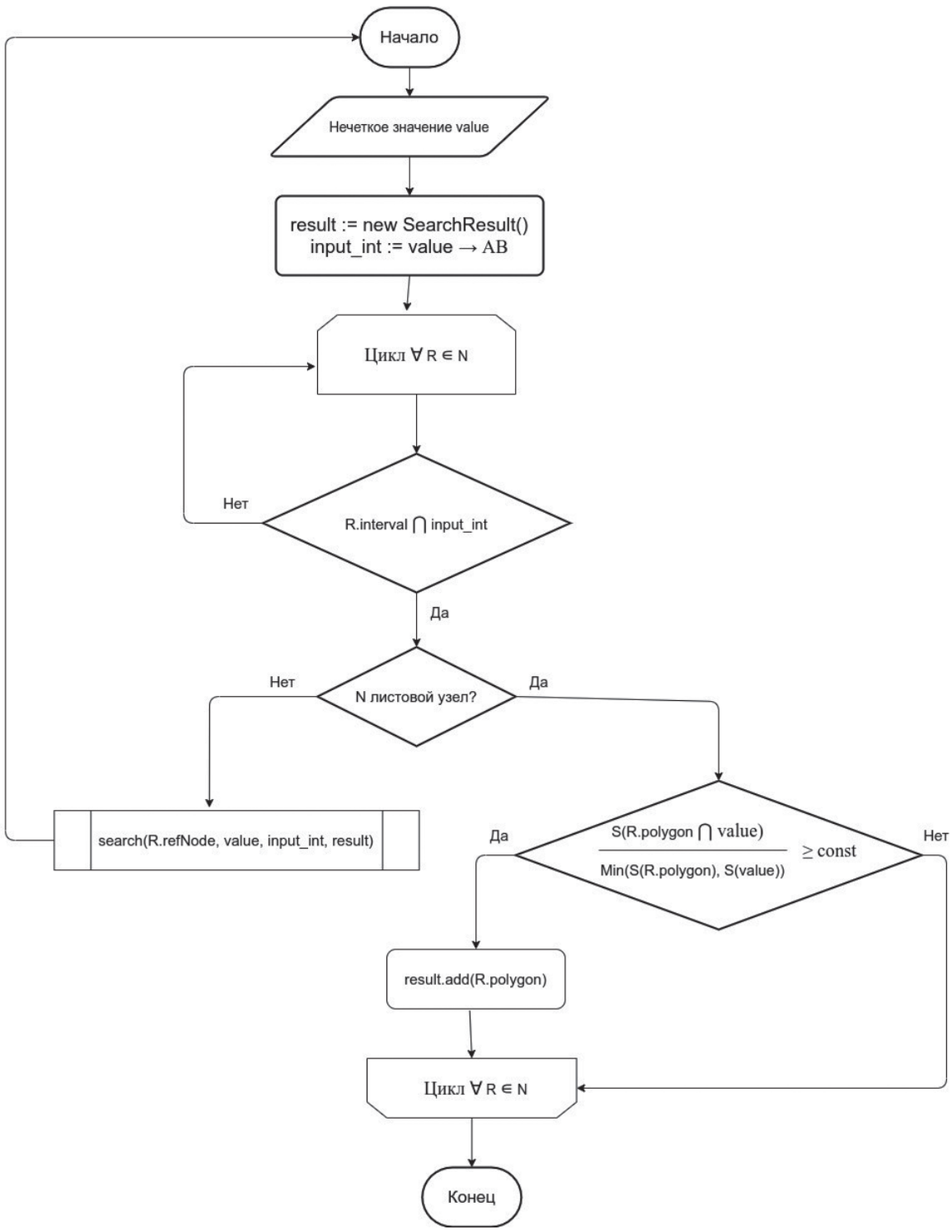
3.Для каждого ключа текущего узла выполнить проверку - есть ли общие точки у $\neq_{\tilde{v}}$ и МОИ ключа. Если общие точки есть, то

– если узел листовой, то если отношение $\frac{S(M_\alpha(\tilde{v}) \cap \tilde{q}_\alpha^i)}{\min(S(M_\alpha(\tilde{v})), S(\tilde{q}_\alpha^i))} \ge S_{\min}$ выполняется, то добавить $\tilde{q}_\alpha^i$ в контейнер Δ;
– иначе рекурсивно выполнить процедуру поиска [20]-[22] на дочернем узле текущего ключа.



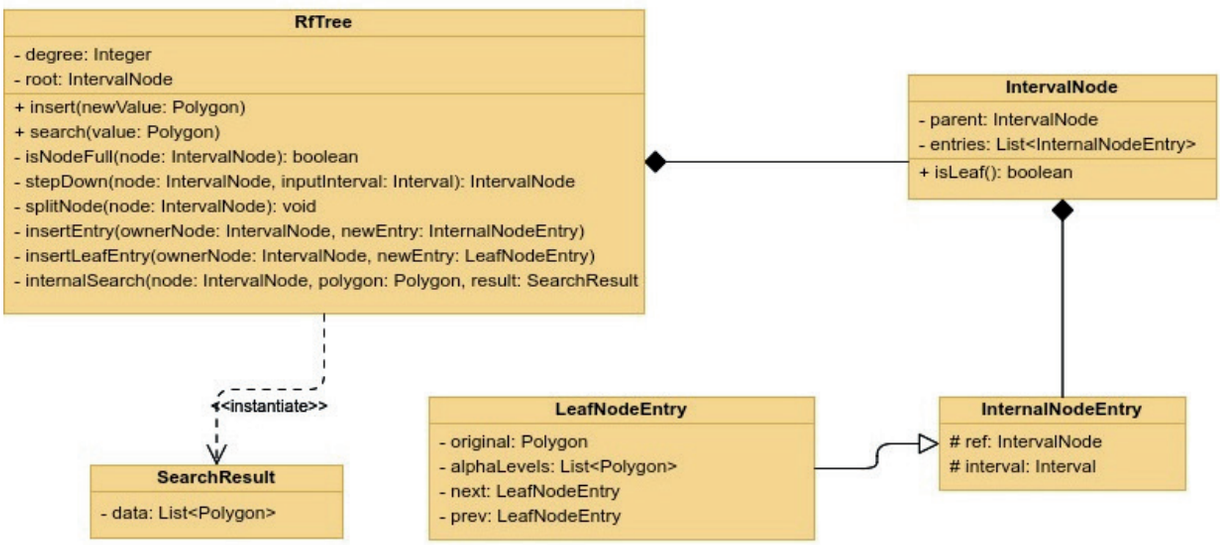
Р и с. 4. Блок-схема алгоритма вставки в R_1^α - дерево

F i g. 4. Block diagram of the algorithm for insertion into R_1^α - tree



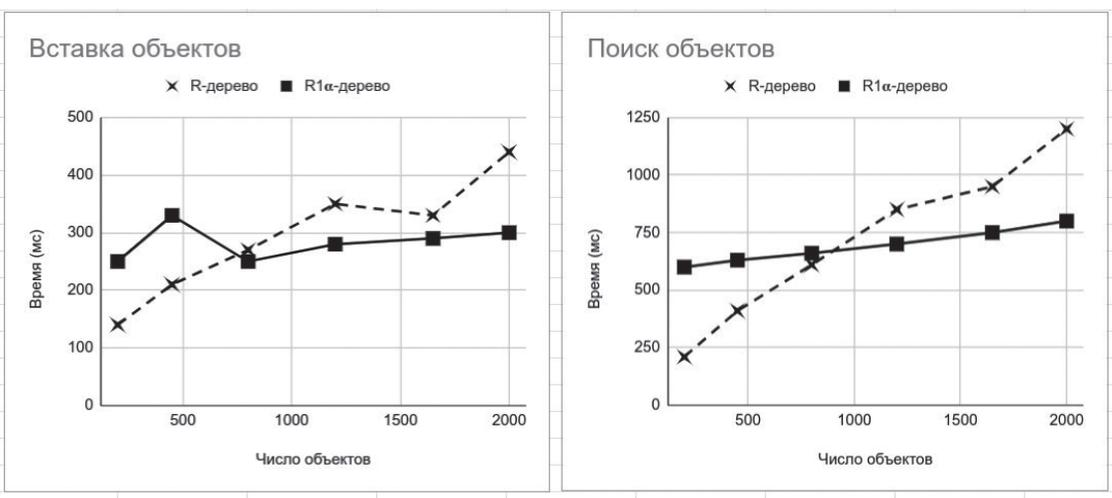
Р и с. 5. Блок-схема алгоритма поиска в R_1^{α} - дереве

F i g. 5. Block diagram of the search algorithm in R_1^{α} - tree



Р и с. 6. Структура классов приложения

F i g. 6. Application class structure



Р и с. 7. Сравнение производительности при вставке в дерево

F i g. 7. Performance comparison when inserting into a tree

Полученные результаты

В результате проделанной работы было разработано приложение реализующее алгоритмы работы с R_1^α -деревом на платформе Java, рис. 6.

Приложение имеет следующую структуру классов:

- Класс RfTree – реализация R_1^α -дерева, содержит как модифицированные методы R-дерева для обработки нечетких значений, так и стандартные для R-дерева методы;
- Класс IntervalNode – класс реализующий функционал внутреннего узла дерева;
- Класс LeafNodeEntry – наследник класса IntervalNode, реализует функционал листового узла дерева;
- Класс SearchResult – контейнер для результатов поиска.

Сравним производительность R^*-дерева и исследуемого R_1^α -дерева.

Для сравнения будем использовать набор из 2000 полигонов, которые представляют нечеткие переменные. Тестирование показывает, что с увеличением числа объектов в дереве производительность R^*-дерева уменьшается сильнее, чем производительность R_1^α -дерева. Это связано с тем, что в R^*-дереве используются минимальные ограничивающие прямоугольники, что усложняет операцию вставки в дерево [23]. А именно, процедура расщепления узла может иметь сложность $O\left(n^2\right)$ или $O\left(2^n\right)$. Также при выполнении операции поиска в R^*-де-

реве необходимо выполнять $O(2n)$ сравнений [24]-[25], в R_1^α -дереве $O(\log n)$ из-за того, что ограничивающие интервалы отсортированы.

Заключение

В данной работе предложена модификация R-дерева – R_1^α -дерево как средство организации индекса СУБД по набору нечетких переменных, предложены алгоритмы для работы с R_1^α -деревом. Индекс на основе R_1^α -дерева позволяет выполнять поиск с использованием операций сравнения: нечетко-равно, нечетко-неравно, нечетко-больше/меньше. Реализация предложенных алгоритмов выложена на портале Github[3] [26] и имеет публичный уровень доступа.

References

[1] Guttman A. R-trees: a dynamic index structure for spatial searching. *SIGMOD Rec*. 1984; 14(2):47-57. (In Eng.) DOI: https://doi.org/10.1145/971697.602266

[2] Manolopoulos Y., Nanopoulos A., Papadopoulos A.N., Theodoridis Y. R-Trees: Theory and Applications. *Advanced Information and Knowledge Processing*. Springer, London; 2006. (In Eng.) DOI: https://doi.org/10.1007/978-1-84628-293-5

[3] Zadeh L.A. The concept of a linguistic variable and its application to approximate reasoning - I. *Information Sciences*. 1975; 8(3):199-249. (In Eng.) DOI: https://doi.org/10.1016/0020-0255(75)90036-5

[4] Zadeh L.A. Fuzzy sets. *Information and Control*. 1965; 8(3):338-353. (In Eng.) DOI: https://doi.org/10.1016/S0019-9958(65)90241-X

[5] Klir G.J., Yuan B. Fuzzy Sets and Fuzzy Logic: Theory and Applications. Prentice Hall: Upper Saddle River, NJ; 1995. (In Eng.)

[6] Ryjov A.P. The measure of uncertainty of fuzzy set's collection: Definition, properties and applications. In: *1993 (2nd) International Symposium on Uncertainty Modeling and Analysis*. College Park, MD, USA; 1993. p. 51-54. (In Eng.) DOI: https://doi.org/10.1109/ISUMA.1993.366792

[7] Kaufmann A. Introduction to the Theory of Fuzzy Subsets. 1st ed. Academic Pr; 1975. (In Eng.)

[8] Vassilakopoulos M., Corral A., Karanikolas N.N. Join-Queries between Two Spatial Datasets Indexed by a Single R-Tree. In: Černá I. et al. (ed.) SOFSEM 2011: Theory and Practice of Computer Science. SOFSEM 2011. *Lecture Notes in Computer Science*. 2011; 6543:533-544. Springer, Berlin, Heidelberg. (In Eng.) DOI: https://doi.org/10.1007/978-3-642-18381-2_44

[9] Samet H. The Design and Analysis of Spatial Data Structures. Addison-Wesley Longman Publishing Co., Inc., New York, USA; 1990. (In Eng.)

[10] Copeland R. MongoDB Applied Design Patterns. O'Reilly Media, Sebastopol CA; 2013. (In Eng.)

[11] Wang X. Interval Tree and Its Application in Integer Factorization. *Journal of Mathematics Research*. 2019; 11(2):103-113. (In Eng.) DOI: http://doi.org/10.5539/jmr.v11n2p103

[12] Allen J.F. Maintaining knowledge about temporal intervals. *Communications of the Association for Computing Machinery*. 1983; 26(11):832-843. (In Eng.) DOI: https://doi.org/10.1145/182.358434

[13] Trudel A. Interval Algebra Networks with Infinite Intervals. In: *2009 16th International Symposium on Temporal Representation and Reasoning*. Bressanone-Brixen, Italy; 2009. p. 141-146. (In Eng.) DOI: https://doi.org/10.1109/TIME.2009.20

[14] Dawood H., Dawood Y. Universal Intervals: Towards a Dependency-Aware Interval Algebra. In: S. Chakraverty (ed.) Mathematical Methods in Interdisciplinary Sciences. Wiley; 2020. p. 167-214. (In Eng.) DOI: https://doi.org/10.1002/9781119585640.ch10

[15] Ligozat G. Qualitative Spatial and Temporal Reasoning. John Wiley & Sons, Inc., London; 2012. (In Eng.)

[16] Bhattacharya A. Fundamentals of Database Indexing and Searching. 1st. ed. Chapman & Hall/CRC, London; 2014. (In Eng.) DOI: https://doi.org/10.1201/b17767

[17] Yang Y., et al. LAZY R-tree: The R-tree with lazy splitting algorithm. *Journal of Information Science*. 2019; 46(2):243-257. (In Eng.) DOI: https://doi.org/10.1177/0165551519828616

[18] Andreev P.D., Bulygin A.I. On the Vertical Similarly Homogeneous R-Trees. *Lobachevskii Journal of Mathematics*. 2019; 40(2):127-139. (In Eng.) DOI: https://doi.org/10.1134/S1995080219020033

[19] Srividhya S., Lavanya S.R. Comparative Analysis of R-Tree and R -Tree in Spatial Database. In: *2014 International Conference on Intelligent Computing Applications*. Coimbatore, India; 2014. p. 449-453. (In Eng.) DOI: https://doi.org/10.1109/ICICA.2014.98

[20] Nakorn T.N., Chongstitvatana J. The RD-tree Allowing Data in Interior Nodes of the R-tree. In: *2006 IEEE Conference on Cybernetics and Intelligent Systems*. Bangkok, Thailand; 2006. p. 1-6. (In Eng.) DOI: https://doi.org/10.1109/ICCIS.2006.252290

[21] Al-Badarneh A.F. Yaseen Q., Hmeidi I. A new enhancement to the R-tree node splitting. *Journal of Information Science*. 2010; 36(1):3-18. (In Eng.) DOI: https://doi.org/10.1177/0165551509340360

[22] Al-Nsour E., Sleit A., Alshraideh M. SOLD: A node-Splitting algorithm for R-tree based on Objects' Locations Distribution. *Journal of Information Science*. 2019; 45(2):169-195. (In Eng.) DOI: https://doi.org/10.1177/0165551518785561

[23] Ang C.H., Tan T.C. New linear node splitting algorithm for R-trees. In: Scholl M., Voisard A. (ed.) Advances in Spatial Databases. SSD 1997. *Lecture Notes in Computer Science*. 1997; 1262:337-349. Springer, Berlin, Heidelberg. (In Eng.) DOI: https://doi.org/10.1007/3-540-63238-7_38

[24] Vu T., Eldawy A. R-Grove: growing a family of R-trees in the big-data forest. In: *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL '18)*. Association for

[3] Хостинг ИТ-проектов Github [Электронный ресурс]. URL: https://github.com/nk-samoylov (дата обращения: 14.09.2020).

Computing Machinery, New York, NY, USA; 2018. p. 532-535. (In Eng.) DOI: https://doi.org/10.1145/3274895.3274984

[25] Gaur G., Kalra S., Bhattacharya A. Patterns for Indexing Large Datasets. In: *Proceedings of the 23rd European Conference on Pattern Languages of Programs (EuroPLoP '18).* Association for Computing Machinery, New York, NY, USA; 2018. Article 5, p. 1-6. (In Eng.) DOI: https://doi.org/10.1145/3282308.3282314

Поступила 14.09.2020; одобрена после рецензирования 21.11.2020; принята к публикации 15.12.2020.
Submitted 14.09.2020; approved after reviewing 21.11.2020; accepted for publication 15.12.2020.

Об авторе:

Самойлов Николай Константинович, ассистент кафедры программирования и информационных технологий, факультет компьютерных наук, ФГБОУ ВО «Воронежский государственный университет» (394018, Российская Федерация, г. Воронеж, Университетская площадь, д. 1), ORCID: http://orcid.org/0000-0002-2945-3340, nk.samoylov@gmail.com

Автор прочитал и одобрил окончательный вариант рукописи.

About the author:

Nikolay K. Samoylov, Assistant of the Department of Programming and Information Technology, Faculty of Computer Sciences, Voronezh State University (1 Universitetskaya pl., Voronezh 394018, Russian Federation), ORCID: http://orcid.org/0000-0002-2945-3340, nk.samoylov@gmail.com

The author has read and approved the final manuscript.