Modification of Query Processing Methods in Distributed Databases Using Fractal Trees

Olha Svynchuk[1], Andrii Barabash[2], Serhii Laptiev[3] and Tetiana Laptieva[4]

[1,2]*National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute», Politehnichna str. 56, 5, Kyiv, 03056, Ukraine*
[3,4]*Taras Shevchenko National University of Kyiv, Volodymyrska str. 64/13, Kyiv, 01601, Ukraine,*

Abstract

Today in database management systems there is an acute problem of searching for data in large data sets. To solve this problem, we propose a modified search tree and its improvement using a fractal index search tree with a multilevel structure. Each level in such a structure is a separate fractal tree. Algorithms for data processing in DBMS RAM by modified methods are described. These methods can be used to search for the same data from different tables. Increased the minimum filling of the node, which reduces the height of the tree. The symmetry of the fractal tree helps to execute the query quickly and, as a result, reduce the number of requests to the disk subsystem. Also, due to the self-similarity property, the most frequently used indexes will be loaded into the DBMS RAM much faster after selection. This will speed up the process of finding the information you need for the request. Loading data indexes into RAM based on statistics on the frequency of use of indices and index size weights will reduce the number of indexes that are loaded into RAM, in contrast to the classic loading where the loading of indexes occurs during their use and after filling the memory, it is deleted. Another big advantage is that indexes that are almost never used will not be loaded into RAM. The proposed approach with fractal trees also has an important scaling property, as fractal trees are divided into a large number of smaller trees, which is especially true in the era of multicore modern computer systems. To study the effectiveness of the use of indexes based on a modified fractal search tree in the database and select the best system for hosting the database server, we measured the speed of information retrieval in tables for the Windows 10 operating system. During the experiments it was shown that the search speed on the modified trees in comparison with the modified fractal search tree is reduced by 12%.

Keywords

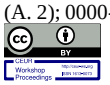
database, data search, B + -trees, modified trees, fractal trees, indexes

1. Introduction

In today's world we can see a rapid increase in information, which complicates the process of its storage and management. Therefore, for its organization and quick search using databases (DB), which are organized according to the concept that describes the characteristics of this data. In modern information systems for high-quality work with databases use DBMS database management systems that provide the ability to create, store, update and search for the necessary information. DBMSs also provide a number of useful services: schema to control data semantics, query language to organize access to part of the

III International Scientific And Practical Conference "Information Security And Information Technologies", September 13–19, 2021, Odesa, Ukraine

EMAIL: 7011990@ukr.net mailto: (A. 1); andrew.barbsh@gmail.com (A. 2); salaptiev@gmail.com (A. 3); tetiana1986@ukr.net (A.4)
ORCID: 0000-0001-9032-6335 (A. 1); 0000-0001-8433-2827 (A. 2); 0000-0002-7291-1829 (A. 3); 0000-0002-5223-9078 (A. 4)



database, data granulation, data integrity management, compression to reduce database size, indexing to speed up query processing. However, the integration of different databases into the production process at enterprises and other institutions has a number of shortcomings associated with the organization of their management and monitoring of events in databases [1-5].

In modern databases, an important element is the search for data in tables that contain many rows and columns and are not always ordered. Therefore, to implement a quick search, indexes are created that are formed from the values of one or more columns and pointers to the corresponding rows. Indexes allow you to avoid sequential or step-by-step browsing of the file in search of the desired data. They are ordered, each element of the index contains the name of the searched object and a pointer-identifier of its location. The more indexes, the better the performance of database queries, but a very large number of indexes does not guarantee high performance [5-10].

Many databases use different trees and their modifications to build such indexes. However, if the tree has an insufficient number of nodes and their fullness, the data search time increases [11-13]. The disadvantages may also be the use of identical indexes for different tables and sending to the RAM of indexes that are rarely used [14-15].

The base trees in index construction and data retrieval are B-trees, namely their type B + trees. These trees easily implement the independence of the program from the structure of the information record, have the ability to sequential access and all key data are contained only in the sheets. The main disadvantages of such trees are the compactness of filling and the number of levels of trees [16-18].

You can also select K-trees, which contain all the characteristics of the B + tree, but have a better strategy of splitting and merging nodes. Also, these trees have more elements at the root of the node and the fullness of the node is ¾. All this saves hard disk space and increases the speed of access to information [19-20].

However, the index structures used in modern databases have some limitations due to the long process of restructuring the index structure in the case of adding or removing new data. Accordingly, this leads to a slow process of searching for information in a database with large data sets.

The aim of the article is to improve the process of processing indexes in databases using fractal trees and speed up query execution.

2. Modified search tree

The existing mechanisms of data modification in the tables have a certain feature - the change of keys in the corresponding nodes of the tree is performed with the subsequent restructuring of the index. This significantly affects the speed of writing information to the database and, accordingly, is an important factor in increasing the number of queries to the database. You also need to store only the most frequently used indexes, then, accordingly, the access time to the data storage location will be reduced. Therefore, the existing methods need to be improved, which will allow to find the necessary information faster [21-22].

In the + tree we will improve as follows:

- increase the minimum filling of the node, which will reduce the height of the tree;
- change the rule of separating the nodes of the tree - splitting the node with its two neighbors into four new nodes;
- change the rule of connecting tree nodes - connecting four nodes into three new nodes;
- in the tree leaf we will store records of links to the same fields in different tables, which will increase the time of receipt of links to data in the tree and speed up the search.

We describe the search for data using indexes. Indexes are loaded into the RAM of the database after receiving the request. Next, a list of data is formed, which contains the necessary information, and the found data is sent to RAM. However, in the classical algorithm for loading indexes in the RAM are indexes that are almost not used, and, accordingly, take place until they are replaced by other indexes. Therefore, it is necessary to improve the procedure for processing indexes in the RAM of the database (Picture 1) by:

- reducing the specific storage in the RAM of indexes that are little used;
- processing little-used indexes by reading them from disk.

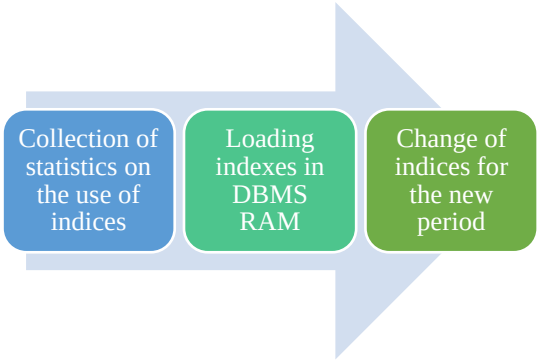


Figure 1: Algorithm for loading indexes in the DBMS RAM by hashing

Here is an algorithm for loading indexes into memory based on the index hashing method:

- DBMS loads indexes into RAM according to the classical algorithm and collects statistics on the number of used indices during Δt;
- after collecting statistics, the DBMS loads into RAM only those data that were used most often during the time period Δt;
- if there is no data in the RAM during the query, the search is performed by reading nodes from the disk index space of the database;
- if the time Δt has expired, then in RAM are loaded those indexes that are used most often and have not been loaded before.

This algorithm is implemented in two stages:

1. statistics are collected on the number of used indices for the corresponding period Δt;
2. the indices that were most often used in the previous time interval Δt are loaded into RAM.

We have a formula for calculating time:

$$\Delta t = \frac{(\sum_{i=1}^{n} k_i W_i)t}{(\sum_{i=1}^{n} W_i)k}, \qquad (1)$$

where k_i – the number of used i-th index, k – the number of indexes used, W_i – the weighting factor of the i-th index, t – time of statistics collection.

Loading data indexes into RAM (figure 1) based on statistics on the frequency of use of indices and index size weights leads to a decrease in the number of indexes that are loaded into RAM, in contrast to the classic loading, where the loading of indexes occurs during their use, and after filling the memory, it is deleted. Another big advantage is that indexes that are almost never used will not be loaded into RAM.

3. A modified method of searching for queries using fractal trees

Recently, fractals are increasingly being used in various areas of our lives. Fractals can be used to model and describe various phenomena in the fields of radio engineering and electronics, digital information processing, and computer graphics [23].

The concept of «fractal» was proposed by the French-American mathematician Benoit Mandelbrot. In 1977, he published Fractal Geometry of Nature, describing repetitive drawings from everyday life. According to him, many geometric shapes consist of smaller shapes, which when enlarged accurately repeat a large shape. After research, he also found that fractals have chaotic behavior, fractional infinite dimension and can be described mathematically using simple algorithms.

Fractal in a more general sense means an irregular, self-similar structure, set, subsets and elements of which are similar to the set itself. Fractals can be deterministic or stochastic. They can also be classified according to self-similarity. There are three types of self-similarity in fractals: exact self-similarity (looks the same at different magnifications); almost self-similarity (fractal looks approximately (but not exactly) self-similar at different magnifications); statistical self-similarity (fractal has numerical or statistical measures that persist with magnification). Examples of fractals are the Cantor set, the Lyapunov fractal, the Serpinsky triangle, the Serpinsky carpet, the Menger sponge, the Apollonia grid, the dragon curve, and the Koch curve. Also recently, attention is paid to fractal trees: from each branch depart smaller, similar to it, from them - even smaller (figure 2). By a separate branch of mathematical methods can describe the properties of the whole tree.

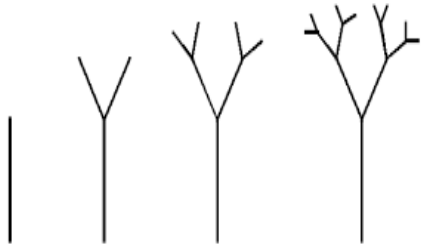


Figure 2: An example of constructing a fractal tree

To construct the structure of the indices will be used Pythagorean fractal tree - a flat fractal, consisting of interconnected right triangles of squares built on the legs and hypotenuse (figure 3).

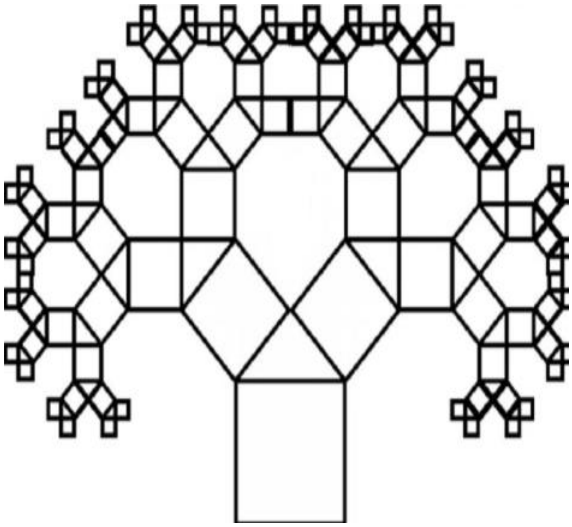


Figure 3: Pythagorean tree

The Pythagorean tree with N levels is a trunk and two Pythagorean trees with N-1 levels depart from it symmetrically, so that the length of their trunks is 2 times less and the angle between them is 90 degrees (figure 4).

The Pythagorean tree is divided into subtree blocks, where each tree is a full-fledged fractal tree. We present this subtree in the form of a new horizontal level, which complements the vertical structure of the original tree. If the new horizontal level is too large, then in order to fit into one block of the disk, it is divided into two blocks and indexed in the third horizontal level.

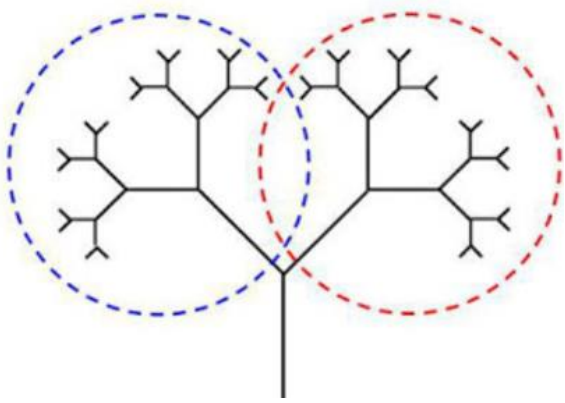


Figure 4: Pythagorean tree for 6 levels

These indexes can be easily used for large databases. The structure of such indexes is presented in the form of arrays with a length equal to powers of number 2. This structure is easily scalable for a large number of keys, and is not sensitive to the content of the entered queries.

The main advantage of using fractal trees is that the resulting structure is symmetrical and internally balanced. Symmetry helps to execute the request quickly and, as a result, there will be much fewer requests to the disk subsystem. Also, due to the self-similarity property, the most frequently used indexes will be loaded into the DBMS RAM much faster after selection. This will speed up the process of finding the information you need for the request.

The index uses a new multi-level approach - additional levels of the tree allow you to search in the data block that contains the information on request. Each request accesses the same number of levels, which provides balanced access to the index and disk subsystem.

Updating, inserting and deleting indexes can be done very efficiently. The update is performed as a sequential deletion of the old key, followed by the insertion of a new key value. Inserting a key into a fractal tree involves adding one new node or adding an edge to an existing node. Inserting requires changes to only one block at level 1. First, look for a block to update - if the block is crowded, it must be divided, and this leads to the creation of a new node in level 2. Separation of blocks is very rare and does not affect performance.

To study the effectiveness of indexes based on a modified fractal search tree in the database and choose the best system for hosting the database server, we measured the speed of information retrieval in tables for Windows 10. Experiments show that the search speed of modified trees

compared to modified fractal search tree is reduced by 12% (Picture 5).

The average error of the result for the modified search tree is 0.91%, and for the modified fractal search tree is 0.89%. Therefore, the experiments are performed correctly and provide the results with a given accuracy.

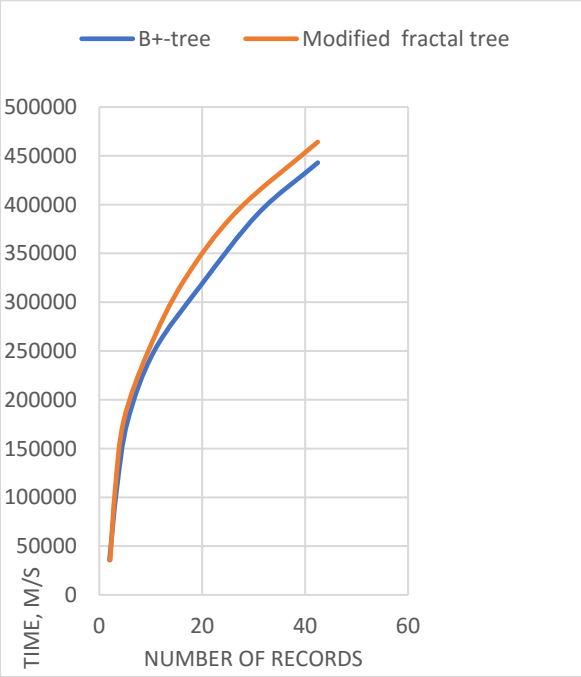


Figure 5: Comparison of query search speed for modified tree and modified fractal tree

4. Conclusions

New methods of index processing in databases for speeding up information processing are offered. The developed modified methods differ from the known methods of processing queries in databases in that they can be used for a large amount of information. Loading data indices into RAM based on statistics on the frequency of use of indices and index size weights leads to a decrease in the number of indexes that are loaded into RAM. A modification of the data processing algorithm in RAM has been performed, which has made it possible to exclude indexes that are rarely used in memory. The resulting structure is balanced and optimized for storage in the disk subsystem, reduces the number of I / O operations to a minimum. The method of constructing indexes based on a modified fractal tree allows to increase the data search speed by 12% compared to the modified method of index search based on a classic B + tree. The proposed approach also has an important property of scaling, as fractal trees are divided into a large number of smaller trees, which is especially true in the era of multicore modern computer systems.

Prospects for further research are seen in the creation of new methods for processing queries in distributed databases based on index hashing using fractal trees.

5. References

[1] V.A. Mashkov, O.V. Barabash, Self-Testing of Multimodule Systems Based on Optimal Check-Connection Structures. Engineering Simulation. Amsterdam: OPA, 13 (1996) 479-492.

[2] V.A. Mashkov, O.V. Barabash, Self-checking and Self-diagnosis of Module Systems on the Principle of Walking Diagnostic Kernel. Engineering Simulation. Amsterdam: OPA, 15 (1998) 43-51.

[3] O. Barabash, G. Shevchenko, N. Dakhno, O. Neshcheret, A. Musienko, Information Technology of Targeting: Optimization of Decision Making Process in a Competitive Environment. International Journal of Intelligent Systems and Applications. Hong Kong: MECS Publisher, 9 (12) (2017) 1-9.

[4] O.V. Barabash, P.V. Open'ko, O.V. Kopiika, H.V. Shevchenko, N.B. Dakhno, Target Programming with Multicriterial Restrictions Application to the Defense Budget Optimization. Advances in Military Technology, 14(2) (2019) 213-229.

[5] V. Sobchuk, O. Barabash, A. Musienko, O. Svynchuk, Adaptive accumulation and diagnostic information systems of enterprises in energy and industry sectors. 1st Conference on Traditional and Renewable Energy Sources: Perspectives and Paradigms for the 21st Century (TRESP 2021), Volume 250, 09 April 2021. doi.org/10.1051/e3sconf/202125008002

[6] H. Zhenbing, V. Mukhin, Ya. Kornaga, O. Herasymenko, Y. Bazaka, The scheduler for the grid system based on the parameters monitoring of the computer components. Eastern European Journal of Enterprise Technologies, 1 (2017) 31-39.

[7] V. Savchenko, O. Ilin, N. Hnidenko, O. Tkachenko, O. Laptiev, S. Lehominova, Detection of Slow DDoS Attacks based on User's Behavior Forecasting. International Journal of Emerging Trends in Engineering Research (IJETER) 8(5) (2020) 2019-2025.

[8] O. Laptiev, O. Stefurak, I. Polovinkin, O. Barabash, V.Savchenko, O. Zelikovska. The method of improving the signal detection quality by accounting for interference. 2020

IEEE 2nd International Conference on Advanced Trends in Information Theory (IEEE ATIT 2020) Conference Proceedings Kyiv, Ukraine, November 25-27, pp.172-176.

[9] V. Tkachov, V.Tokariev, Y. Dukh, V. Volotka, Method of Data Collection in Wireless Sensor Networks Using Flying Ad Hoc Network. 2018 5th International Scientific-Practical Conference Problems of Infocommunications. Science and Technology, October 9-12, 2018 Kharkiv, Ukraine, pp. 197-201.

[10] K. Smelyakov, S. Smelyakov, A. Chupryna Advances in Spatio-Temporal Segmentation of Visual Data. Chapter 1. Adaptive Edge Detection Models and Algorithms. Series Studies in Computational Intelligence (SCI), volume 876, publisher Springer, Cham, 2020, pp. 1-51.

[11] S. Yevseiev, R. Korolyov, A. Tkachov, O. Laptiev, I. Opirskyy, O. Soloviova, Modification of the algorithm (OFM) S-box, which provides increasing crypto resistance in the post-quantum period. International Journal of Advanced Trends in Computer Science and Engineering (IJATCSE) 9(5) (2020) 8725-8729.

[12] O. Barabash, O. Laptiev, O. Kovtun, O. Leshchenko, K. Dukhnovska, A. Biehun, The Method dynavic TF-IDF. International Journal of Emerging Trends in Engineering Research (IJETER), 8(9) (2020) 5713-5718.

[13] O. Laptiev, V.Savchenko, S. Yevseiev, H. Haidur, S. Gakhov, Spartak Hohoniants, The new method for detecting signals of means of covert obtaining information. 2020 IEEE 2nd International Conference on Advanced Trends in Information Theory (IEEE ATIT 2020) Conference Proceedings Kyiv, Ukraine, November 25-27, pp.176 –181.

[14] V. Sobchuk, V. Pichkur, O. Barabash, O. Laptiev, I.Kovalchuk, A. Zidan, Algorithm of control of functionally stable manufacturing processes of enterprises. 2020 IEEE 2nd International Conference on Advanced Trends in Information Theory (IEEE ATIT 2020) Conference Proceedings Kyiv, Ukraine, November 25-27, pp. 206-211.

[15] V. Savchenko, O. Laptiev, O. Kolos, R. Lisnevskyi, V. Ivannikova, I. Ablazov, Hidden Transmitter Localization Accuracy Model Based on Multi-Position Range Measurement. 2020 IEEE 2nd International Conference on Advanced Trends in Information Theory (IEEE ATIT 2020) Conference Proceedings Kyiv, Ukraine, November 25-27, pp.246-251.

[16] Z. Hu, V. Mukhin, Ya. Kornaga, O. Herasymenko, Y. Mostoviy, The Analytical Model for Distributed Computer System Parameters Control Based on Multi-factoring Estimations. Journal of Network and Systems Management, 27(2) (2019) 351-365.

[17] O. Barabash, O. Laptiev, V. Tkachev, O. Maystrov, O. Krasikov, I. Polovinkin, The Indirect method of obtaining Estimates of the Parameters of Radio Signals of covert means of obtaining Information. International Journal of Emerging Trends in Engineering Research (IJETER), 8(8) (2020) 4133-4139.

[18] S. Yevseiev, R. Korolyov, A. Tkachov, O. Laptiev, I. Opirskyy, O. Soloviova, Modification of the algorithm (OFM) S-box, which provides increasing crypto resistance in the post-quantum period. International Journal of Advanced Trends in Computer Science and Engineering (IJATCSE) 9(5) (2020) 8725-8729.

[19] M. Pratsiovytyi, O. Svynchuk, Spread of values of a Cantor-type fractal continuous nonmonotone function. Journal of Mathematical Sciences, 240(3) (2019) 342-357. doi.org/10.1007/s10958-019-04354-0

[20] O. Laptiev, G. Shuklin, O.Stefurak, O. Svynchuk, O. Urdenko, S. Hohoniants, Method of the increasing the detection system and recognition of digital radiosignals. Wschodnioeuropejskie Czasopismo Naukowe, East European Scientific Journal, 2 (54) (2020) 4-16.

[21] O. Svynchuk, O. Barabash, J. Nikodem, R. Kochan, O. Laptiev, Image compression using fractal functions Fractal and Fractional, 5(2), (2021) 31.

[22] O.V. Barabash, A.P. Musienko, V.V. Sobchuk, N.V. Lukova-Chuiko, O.V. Svynchuk, Distribution of Values of Cantor Type Fractal Functions with Specified Restrictions. Chapter in Book "Contemporary Approaches and Methods in Fundamental Mathematics and Mechanics". Editors Victor A. Sadovnichiy, Michael Z. Zgurovsky. Publisher Name: Springer, Cham, Switzerland AG 2021, pp. 433-455. doi.org/10.1007/978-3-030-50302-4_21

[23] S. Toliupa, N. Lukova-Chuiko, O. Oksiuk. Choice of Reasonable Variant of Signal and Code Constructions for Multirays Radio Channels. Second International Scientific-Practical Conference Problems of Infocommunications. Science and Technology. IEEE PIC S&T 2015. pp. 269 – 271.