DBMS Index for Hierarchical Data Using Nested Intervals and Residue Classes

Vladimir Volonkin

Institute of Information Technologies and Telecommunications of the North-Caucasus Federal University, Stavropol, Russia
vl.voronkin@raxperi.com

Abstract. In the work an index based on B+ tree and oriented to storage of tree which are coded by nested intervals method with usage of system of residual classes is described.

Keywords: nested intervals, residue classes, hierarchical data.

1 Introduction

A hierarchy in relational databases (RDB) in present time is implemented ineffectively. Existed methods have definite drawbacks and intent as usual on either quick data reading truckle to record or vice versa.

There are a few possible ways of an effectiveness increasing of a tree-type structures storage: an improvement of existed methods of a hierarchy storage, a development and a research of new methods or refusal of relational databases using in favour of NoSQL databases: graph, document-centric and other types.

All methods of a hierarchy structures presentation in RDB can be grouped into two main categories:

1. Methods of trees encoding.
2. Hierarchical / recursive SQL extensions.

We should note methods of hierarchy presentations, which are based on combining of several methods, for example: method, combining materialized path and adjacency list [1].

Till recently there were two main methods of hierarchy storage (graph) in group of trees encoding methods: nested sets [2] and materialized path [5].

In reference [1] Vadim Tropashko has suggested a modification of materialized path: a method of nested intervals for storage of tree-type structures. The offered method, based on conception of the materialized path in graphs and the continued fractions theory, eliminates a problem of a data redundancy in the materialized path.

Main problems of the method are following: a necessity of calculation with a great numbers, a significant complexity of an execution of partial operations in particular trees rearrangement during transference of subtrees.

To eliminate problems concerned with nested intervals authors [3] offered to encode nested intervals with using of systems of residual classes (SRC).

Using of SRC makes a number multiprocessing possible, relieves from the necessity to work with great numbers and bignums, but imposes some constraints on its using:

- a possibility to present only restricted amount of numbers;
- an absence of effective algorithms for comparison of numbers in SRC.

For effective presentation of trees in databases it is necessary to solve following problems:

- an elimination of working with great numbers and bignums, which are appearing by using of the nested intervals method;
  The problem solving is posed in [3]. A consequence of this method using is problem of increasing of number storage redundancy. This problem is following from nature of number, which is presented in SRC.
- high complexity of operations execution to rearrange tree;
  The problem can be partly solved by using of residue numbers. Interdependency between residues of numbers, which are expressed in SRC, that allows to execute arithmetics with residues simultaneously. For realization of an opportunity of parallel calculations with number residues of nested intervals expressed in SRC, without taking interdependency into consideration, numbers should be in particular order: a parent node should be described necessarily before a descendant node. An alternation of sibling nodes doesnt matter.
- an absence of indexing methods for numbers expressed in SRC;
  At the moment there are no any indexing methods for numbers expressed in SRC. An indexation of numbers expressed in SRC by using traditional algorithms is unfeasible because numbers expressed in SRC dont seem possible to arrange in series as in a decimal as in a SRC representation. This feature issues from the nature of number expressed in SRC.
- a significant performance penalty at storage of big trees in DB (more than 1 mln records);
  The problem is mainly conditioned by increasing of a key length during derivation of big trees and by a big quantity of calculations needed to an execution of tree derivation operations.

Offered in the work approach to the index derivation and processing allows to solve problems described above.

There is provided a structure of index based on the B+ tree and oriented on index multiprocessing in not relational DB.

2 Structure of the Index

2.1 Encoding of the Tree by the Method of Nested Intervals

The method of nested intervals is an expansion of nested sets model, using of continued fractions.

Lets present a tree:

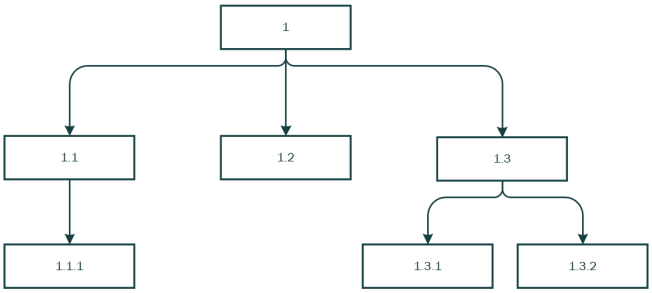

Fig. 1. An examle of tree

Encode the node 1.3.2 using of continued fractions 1:

$$1.3.2 = 1 + \frac{1}{3 + \frac{1}{2 + \frac{1}{x}}} = \frac{9x + 4}{7x + 3}; \tag{1}$$

A result 1 is an nested interval:

$$(\frac{9}{7}, \frac{4}{3}) \tag{2}$$

An interval (2) determines a range, where all descendant nodes will be encoded, and consists of two parts: 9/7 directly a node code, 4/3 the parents node code. For unique identification of the node it is enough to use only a node code, and to calculate the parents node code only if necessary.

2.2 Systems of Residual Classes in the Index

A presentation of number in the SRC is based on a notion of deduction and the Chinese remainder theorem. The SRC is determined of set of coprime modules which are called a basis. In the SRC [4] numbers are represented in following way:

$$A(\alpha_1, \alpha_2, ..., \alpha_n); \tag{3}$$

$$\alpha_i = A - [\frac{A}{p_i}]p_i, (\forall_i \in [1, n]), \tag{4}$$

$$P = \prod_{i=1}^{n} p_i \tag{5}$$

Where $p_1, p_2, ..., p_n$ - system modules, P - volume of system range. Assume $p_1 = 2, p_2 = 3, p_3 = 5, P = 30$.

Consider, how the node code is encoding in the SRC:
Encode a node 1.3.2 by using of formulas (3) and (4):

$$1.3.2 = (\frac{9}{7}, \frac{4}{3}) \tag{6}$$

We will use only the node code:

$$\alpha_{11} = 17 = (\alpha_1, \alpha_2, \alpha_3). \tag{7}$$

$$\alpha_1 = 9 - [\frac{9}{2}] * 2 = 1, \tag{8}$$

$$\alpha_2 = 9 - [\frac{9}{3}] * 3 = 0, \tag{9}$$

$$\alpha_3 = 9 - [\frac{9}{5}] * 5 = 4, \tag{10}$$

$$\alpha_{12} = 7 = (1, 1, 2). \tag{11}$$

It is following from the nature of the SRC that a number expression in the SRC imposes constraints on a number length.

A maximum quantity of numbers, which could be stored in a number expressed in the SRC, is always less than a quantity of numbers, which could be stored in the same quantity of bits as a decimal number.

So a maximum whole unsigned number, which can be stored in 4 bytes, equals:

$$N_{max} = 2^{32} = 4294967296 \tag{12}$$

It is known that modules of the system of residual classes $p_1, p_2, ..., p_n$ should be prime in pairs numbers. In this case there is a single non-negative decision modulo P an equation system which describes residue numbers:

$$x \equiv \alpha_1(mod(p_1)), x \equiv \alpha_2(mod(p_2)), , x \equiv \alpha_k(mod(p_k)) \tag{13}$$

Obviously the more equal P and N_{max}, the less of a redundancy of numbers storage in the SRC. Hence, to find a minimum pressure of numbers record in the SRC it is necessary to try maximally great prime in pairs residue number base.

As an example we will try optimum bases, which are maximally not exessive for number length of 4 bytes: 255, 254, 253, 251.

$$P = 255 * 254 * 253 * 251 = 4113089310 \tag{14}$$

Thus storing residues should be maximally great and prime in pairs numbers, which could be got into such quantity of bits that fits to one residue.

For an increasing of number quantity in index, which can be expressed in the SRC, a dynamic change algorithm of residues quantity in number expressed in the SRC.

In the case when a situation of number repletion, which is expressed in the SRC, appears (situation when a sum of several SRC is exceeded P) the number is added by additional residue and in that way the maximum quantity of stored numbers P increases.

In the case when a large quantity of residues exists, with the purpose of storage redundant decrease a converting of system bases is performed with increase of each residue bit length .

In the case of using 2 bytes instead of one we can store (15) numbers for one residue storage in 4 bytes.

$$P = 65535 * 65534 = 4294770690 \tag{15}$$

It follows that (15)has less data redundance in comparison with (14) and allows to express more numbers quantity in the SRC.

2.3 Structure of the Index

The main problem of using an approach, offered in [3], is an impossibility of index derivation by numbers expressed in the SRC with traditional methods, because it is impossible to compare directly numbers expressed in the system of residual classes. To compare such numbers it is required to execute certain arithmetic conversions. As usual to compare numbers expressed in the SRC we should convert each number to the radix numeration system for further compare. From the point of view of productivity this approach is not effective because of computation efforts to number conversion from the nonpositional notation to the radix numeration system and because of necessity to work with great numbers and bignums.

An algorithm of index derivation is offered to solve the problem concerned with of impossibility of index derivation by numbers expressed in the SRC with traditional methods.

Note a tree from the figure 1 as a line (16), arranging nodes in order of a tree traversal from left to right. The traversal is realized as in a left-side tree traversal in case of using of nested sets.

$$[1.1[1.1.1]1.2, 1.3[1.3.1, 1.3.2]] \tag{16}$$

For illustrative purposes descendant nodes are enclosed in brackets.

Encode each node of the tree using the method of nested intervals and convert numbers to the SRC. As a system of residual classes bases we use modules: 3, 5, 7.

To solve a problem of intervals cross-cups we use early known decision, which is based on chain fractions properties: refusal of using index ‘1‘ as an element of materialized path, and a root of tree equals “2.2“.

As it was said earlier for identification of tree node it is not necessary to calculate all interval, where descendant nodes are. It is enough to calculate an interval begin. In this case the interval begin identifies tree node by unique way and is node code.

Results of conversions are represented in table:

Initial materialized path	Modified materialized path	Node code (interval begin)	Node code in the SRC
1	2.2	5/2	(2,0,5)/(2,2,2)
1.1	2.2.2	12/5	(0,2,5)/(2,0,5)
1.1.1	2.2.2.2	29/12	(2,4,1)/(0,2,5)
1.2	2.2.3	17/7	(2,2,3)/(1,2,0)
1.3	2.2.4	22/9	(1,2,1)/(0,4,2)
1.3.1	2.2.4.2	49/20	(1,4,0)/(2,0,6)
1.3.2	2.2.4.3	71/29	(2,1,1)/(2,4,1)

Rewrite the line (16), replaced there elements of reified path by a value of node code in the SRC:

$$\frac{(2,0,5)}{(2,2,2)}\left[\frac{(0,2,5)}{(2,0,5)}\left[\frac{(2,4,1)}{(0,2,5)}\right]\frac{(2,2,3)}{(1,2,0)},\frac{(1,2,1)}{(0,4,2)}\left[\frac{(1,4,0)}{(2,0,6)},\frac{(2,1,1)}{(2,4,1)}\right]\right] \tag{17}$$

Note that the tree in the line (17) is represented in sorted-out state. And the line, which encodes the tree, may be unambiguously formed in a process of tree processing if a transactional integrity of execution of modification operation its tops and edges.

For illustrative purposes futher we will operate with elements of materialized path meaning element codes in the SRC.

Rewrite the line (16), removed from materialized path data about parent node:

$$1\big[1[1]2,3[1,2]\big] \tag{18}$$

Switch from string indication of the line (18) to binary. The figure 2 clearly shows intervals of encoded tree, which are stored in index.

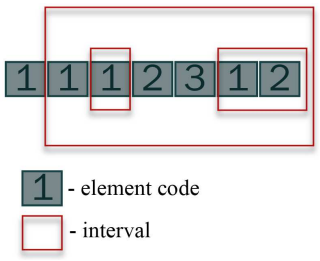

Fig. 2. Nested intervals of encoded tree

Offered index structure is the most productive when using in combination with highly productive data warehouse. As an example we will consider realization of this index in NoSQL DBMS MongoDB.

In binary form the index is stored as pages (documents) consecution with contiguous information. Besides node code each index record contains this document (or link to it) and reference to parent node. The index structure is presented in the figure 3.

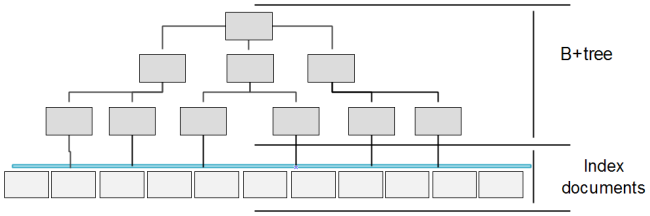

Fig. 3. Index structure

The index includes B+ tree and documents, which store data about nodes and relations between them.

A simplified diagram of index documents is presented in the figure 4. This scheme extends the figure 2 by adding new connections between records.

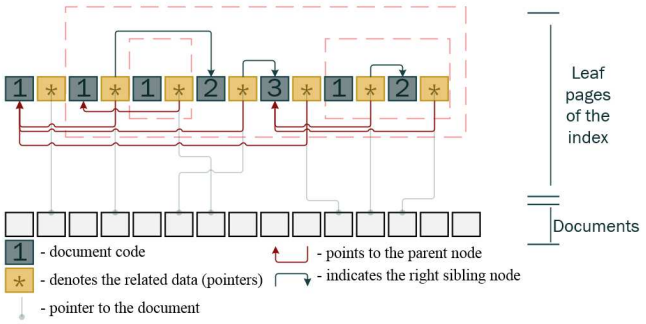

Fig. 4. A simplified diagram of index

Each record in index document store the following data: record code, link to the right sibling node, document corresponding tree node, or link to it, and link to parent node.

In spite of the fact that this scheme stores relations between tree nodes, search of required node is possible only by walkthrough of all records. For elimination of this problem it is necessary to derivate index for access of tree tops, which

is encoded in the SRC. As index we will choose structure of type B+ tree with sheet-like records, which references to top codes of the line corresponding to tree.

To solve the problem of comparing numbers expressed in the SRC it needs to derivate B+ tree according to numbers expressed in the SRC as ordinary array. Comparing of two numbers of the SRC in this case occurs as comparing of usual areas. This scheme allows to disregard from the SRC conception by adding of some redundance.

Sheet-like records of the tree point to corresponding records in index sheet pages.

A simplified diagram of index is represented on the figure 5.

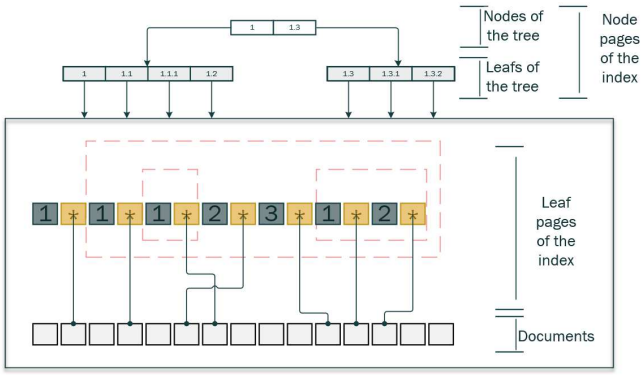

Fig. 5. Schematic representation of the index structure

However because of the fact that residues value in the SRC represented as array doesnt correspond to decimal value of number expressed in the SRC, elements in sheet pages of the tree would point to elements in sheet pages of index randomly (figure 6).

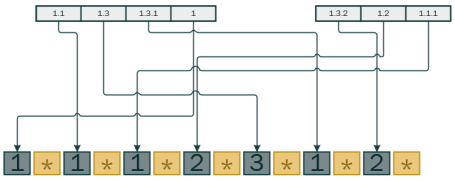

Fig. 6. Schematic representation of an accordance of records in sheet pages of the tree to records in sheet pages of the index

Encoded numbers in the SRC and switched from line representation to binary, we get the index structure, which is in the figure 7.

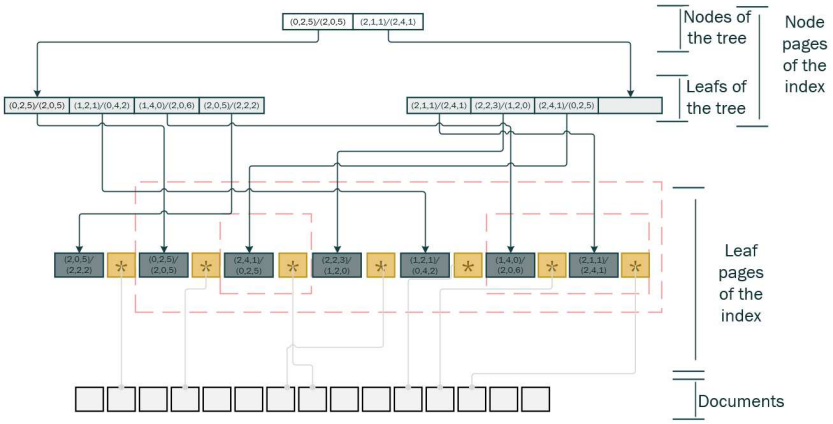

Fig. 7. Schematic map of the index structure

Such record allows to read trees and subtrees quickly and to perform operations of recording: an addition, a removal and displacement of nodes/subtrees.

Sheet-like index pages are replaced on the disc in the form of the double-linked list (figure 8)in the order of the tree nodes traversal. All links are using a multilevel addressing.

Fig. 8. A consecution of pages in the index file

High-speed operations of tree manipulation in this index are possible due to during data storage in the form of consecution of pages/documents a task of migration of data parts (for instance, during a subtree removal) comes to fragmentation/unification of boundary pages (which contain data partly) and to changing of links between pages with the purpose of maintenance of tree consecution arrangement in the index.

As result of the fact that each residue of number expressed in the SRC doesnt depend on other residues of the same number, we get an possibility of the high-speed converting of the subtree with using of parallel computations.

3 Experiments and Results

Testing results of presented in the article index structure in comparison with the realization of trees storage by the method of nested intervals and with the index of B+ tree type are shown in the table 1. As DBMS it was choosen NoSQL DBMS MongoDB. Offered index is realized as separate module, which expands

a functionality of MongoDB. For parallel computations the graphics processor NVIDIA and the technology CUDA were applied.

Table 1. Results of productivity comparasing

Characteristic	Offering index	Nested intervals + B+ tree
Time of node insertion	0.00005 sec	0.00007 sec
Time of node removal	0.00005 sec	0.000001 sec
Time of subtree moving (10000 nodes)(CPU)	28.8 sec	25.4 sec
Time of subtree moving (10000 nodes)(GPU)	0.0007 sec, without data copying (between CPU and GPU) 0.42 sec with data copying (between CPU and GPU)	A supporting of GPU in MongoDB is not realized

4 Conclusion

In the work the index structure for storage of hierarchies in DB is suggested. For derivation of the index it is used the B+ tree, which is necessary for high-speed finding of document locations on the disc, and array of documents which are the index base. Also the method of index derivation on numbers expressed in the SRC is represented.

Using of this index structure in common with methods, allowing to realize parallel computations, permits to increase speed of working with trees.

Offered index structure can be applied in databases of NoSQL style to increase of productivity of index structure processing.

This approach can be improved by adding of methods of vector residue compression for decreasing of overhead expenses on storage and increasing of data processing rate.

References

1. V. Tropashko: Nested Intervals Tree Encoding with Continued Fractions. ACM SIGMOD, Volume 34, Issue 2, 2005
2. J. Celko: Joe Celkos Trees and Hierarchies in SQL for Smarties. Morgan Kaufmann, 2004
3. A. Malikov, A. Turyev: Nested Intervals Tree Encoding with System of Residual Classes. ICEICE No.2, 2011
4. N. Chervyakov: Modular Parallel Computing Structures of Neuroprocces System. FIZMAT, 2003
5. V. Tropashko: Trees in SQL: Nested Sets and Materialized Path: [Electronic resource]. 2003. URL: https://communities.bmc.com/docs/DOC-9902