Multidimensional data structures usage in adaptive data storages

D R Potapov
Department of Applied Mathematics, Informatics and Mechanics, Voronezh State University, 394006, Universitetskaya pl., 1, Voronezh, Russia
E-mail: potapovd36@gmail.com

Abstract. The article is dedicated to the analysis of methods for construction of multidimensional data storages. This topic is important because of the increasing demand for these containers in today’s world. Containers can be used for working with multidimensional vectors in such fields of knowledge as economics, physics, biology, political science, medicine and technology, for solving various tasks in computer graphics, multimedia databases and animation, for working with spatial data. The paper analyzes and reviews hierarchical and hashed containers, space-filling curves, different hybrid and high dimensional containers. The article also identifies the main fields of application of multidimensional structures that can be used in implementation of adaptive storages.

1. Introduction
Most of objects in the real world are multidimensional or have a lot of parameters that can be transformed to multidimensional variables. Until recently, processing of multidimensional objects has been executed only by transforming them to one-dimensional objects. This approach decreases processing and analysis efficiency. Recently, however, the situation has changed and a lot of new algorithms and data structures have appeared [1] as a result of the growing demand in various fields of science and production.

Multidimensional data structures [2] and algorithms are used for processing multidimensional vectors in physics (e.g. computational hydrodynamics, electromagnetism), biology (e.g. to search for incomplete coincidences in protein and DNA sequences), economy, political science, medicine and technology. These containers are also widely used in computer graphics and animation, multimedia databases, geographic information systems, virtual reality systems.

Classification of multidimensional data structures [3]:

- Location:
  - RAM
  - External memory
- Data access method [4]:
  - Point access method (PAM)
  - Spatial access method (SAM)
- Partitioning method:
  - Space driven partitioning
The present article analyzes multidimensional data storages (classified by construction method, refer to figure 1) and their usage in adaptive data storages.

Figure 1. Classification Of Multidimensional Data Structures.

2. Hierarchical Representation of Multidimensional Information
Hierarchical multidimensional data structures can be divided into two large groups:

- **R-like** trees. R-tree [6] (R by region/rectangle) divides space into rectangles (in space with dimensionality equal two) and into paralleloptopes (in multidimensional space) so that regions can intersect and form a hierarchy (refer to figure 2). In this tree, objects that are closely located should be placed in the same node of the tree. In addition, there are various modifications and variations that have been created to improve R tree performance. The most famous R-like trees are **R*-tree** [7], **R+ tree** [8], **X-tree** [9], **M-tree** [10], **SS-tree** [11], **SR-tree** [12], **VAM-split R-tree** [13].
• **BSP-like** trees. BSP (binary space partition) tree [14] — a hierarchical data structure, each internal node of which contains splitting hyperplane that divides space into two subspaces and leaves contain objects. In this tree, if an object is located in the positive half-space (relatively to the dividing plane), then it is the right son, otherwise, it is the left one (refer to figure 3). Thus, this tree corresponds to some binary space partition. In addition, there are a lot of BSP-tree modifications that determine how the splitting planes are defined. The most famous BSP-like trees are k-d tree [15][16], k-d-b tree [17], hb-tree [18], LSD-tree [19], BIH/SKD-tree [20], quadtree [21], octree [22], VP tree [23], BD tree [24].

![Figure 2. R-tree.](image2)

![Figure 3. BSP-tree.](image3)

3. **Multidimensional Hashing**
In addition to the hierarchical approach to construct multidimensional data containers, hashing is used. In this case, space is divided into cells by scales \([x_0, x_1, x_2]\) and \([y_0, y_1, y_2]\) in such a way
so that one cell contains no more than \( N \) objects. \( N \) is a parameter of data structure. It defines maximum count of objects that can be placed in one memory block (e.g. hard drive segment). There are two methods to identify cell address by coordinates:

- Using directory. Directory is a two-dimensional array that corresponds to appropriate space partitioning. Each cell of this array contains hard drive block address, which contains appropriate cell objects (one block can contain several cells). Figure 4 illustrates this schema. Containers that use directory are Gridfile [25] and its modifications (Twin Grid file [26], Two-layer Grid file [25], Multilevel Grid file [27]), EXCELL [25], R-file [28].

- Without directory. In this case, special hash function is used to identify cell address by coordinates. This improvement allows avoiding one access to external memory. In addition, linear hashing is used instead of linear scales along the axes, e.g. for two-dimensional case \( l_0 = H_0(K_0), l_1 = H_1(K_1) \), where \( K_0, K_1 \) are the object’s keys. The MOLHPE data structure (multidimensional order preserving linear hashing with partial expansions) [29] implements this approach. Other container — PLOP (Piecewise linear order preserving hashing scheme) [30] uses binary trees instead linear hashing and supports chain of blocks (refer to figure 5).

---

**Figure 4.** Grid-file schema with \( N=3 \). Left — space dividing by Gridfile, center - mapping space cells to external blocks, right — Gridfile directory.

**Figure 5.** PLOP.
4. Space-filling Curves
Another method of constructing multidimensional data containers is space-filling curves. The main idea behind this approach is to order all multidimensional points. So these points are transformed to one-dimensional points, to which one-dimensional containers can be applied [31]. There are several well-known curves that can order points (see figure 6). Figure 6(a) illustrates “row by row” curve, figure 6(b) — ”snake” curve, figure 6(c) — spiral curve, figure 6(d) — Cantor curve [32], figure 6(e) — Peano curve, figure 6(f) — U-index curve, figure 6(g) — Z-mirror curve, figure 6(h) — Gray curve [33] and figure 6(i) — Hilbert curve [34].

![Space-filling curves](image)

Figure 6. Space-filling curves.

5. Hybrid Methods
There are a lot of methods combinations in addition to those listed above. They can be split into three groups:

- Hybrid of hashing and hierarchical approaches. The main idea behind this method is to order blocks (from hashing) as tree (hierarchical). BANG-file [35], buddy-tree [36], HR Tree and G-tree are examples of this approach.
- Hybrid of hierarchical access method and space-filling curves. First, objects are sorted by curves and then saved in a tree. For example, UB-tree [37], Hilbert R-tree [38].
- Hybrid of two hierarchical approaches — R-tree and BSP-tree or R-tree and quadtree. For instance, R*Q Tree, Q+R Tree, hybrid tree, SH-tree.

6. Multidimensional Containers For High-dimensional Space
According to the research [2], all data structures mentioned above are suitable for dimensionality that does not exceed 15. For handling objects in spaces where dimensionality is more than 15, other containers were invented; these containers use the filtering approach also known as vector-approximation. Such data structures split the data space into rectangular cells, assign a unique bit-string of each cell and approximate the data vector that falls into a cell by this bit-string. This approach allows limiting access to external memory by reading a sequentially small approximations file instead of the file with real vectors on the first step and extract from external memory just those vectors that were filtered by approximation. A vector approximations file is
much smaller than the original data file and hence far more efficient than direct sequential scan and the variants of R-tree (see figure 7). The most well-known containers for high-dimensional space are VA-File, VA+-File, LPC-File, A-Tree, GC-Tree, RA-Blocks, IQ-tree, SA-tree, A-tree, iDistance [39].

Figure 7. VA-file.

7. Analysis

There are a lot of multidimensional data structures, so there is an opportunity to choose an optimal data structure for usage in a self-adaptive storage almost for any case. When choosing, it is necessary to take into account both the general conditions of the container operation (the detailed analysis is presented in this article [40]) and conditions specific for multi-dimensional containers.

When working in spaces with large dimensionality, there is necessary to use special containers (refer to Multidimensional Containers For High-dimensional Space).

When the amount of data is small and it fits in the RAM, k-d tree, BSP tree, BD tree, quad tree, R-tree and their various modifications can be used. In addition, there are special (cache conscious) data structures that consider the peculiarities of the processor caches, which improve performance. Such containers include the CR-tree [41] and its various options, MR-tree [42], DR-tree.

For situations where data is used for a long time without modification, it is necessary to use structures for static data, such as k-d tree, Packed Hilbert R-tree and VAMSplit trees. For working with large amounts of data, when there is not enough RAM, other containers must be used. According to the research [2], the following structures are the best in terms of performance (this list doesn’t include containers, comparative analysis for which was not published):

- Hilbert R-tree [38];
- cell tree with oversize shelves [43];
- buddy (hash) tree [36];
- KD2B-tree [44];
- PMR-quadtree [45];
- R+-tree [8];
- R*-tree [7].

However, to choose even among these structures is not an easy task, because at the moment there is no best data structure by all parameters. First of all, this is due to the fact that there are many optimality criteria, so the best container by one parameter may be worst by another parameter. For example, storage’s time and space efficiency depend on data, queries, in many
respects from hardware equipment characteristics (for example the size of the page of memory). On the other hand, a container, which is good with iso-oriented splitting hyperplanes, may have very bad efficiency with arbitrary splitting hyperplanes. It is also worth considering that the point access methods can be very ineffective for handling spatial objects.

In general, a data structure that will be used in the external memory for a self-adaptive data storage should be selected based on reliability and simplicity of implementation. This approach is applied in many commercial products, for example, using quadtrees in SICAD and Smallworld GIS, R-trees in Informix, and Z-ordering in Oracle [2]. In addition, based on the statistics of queries to self-adaptive storage, it is possible to analyze multidimensional data structures and select the best container for this load by the trial and error method.

8. Conclusion
In the article, the existing methods of constructing multidimensional data containers are considered and an overview of such containers is presented. The article analyzes multidimensional data structures and highlights the scope of application of these containers in the implementation of self-adapting associative data containers.

References
[1] Gulakov V K 2010 Multidimensional data structures (Bryansk: Bryansk State Technical University Press) [in Russian]
[2] Gaede V and Gunther O 1998 Multidimensional Access Methods ACM Comput. Surv. 30(2) 170–231
[3] Averchenkov V I, Gulakov V K, Trubakov A O, Trubakov E O and Matyushin V N 2014 ANALYSIS AND CLASSIFICATION OF DATA ACCESS METHODS Bulletin of the of computer and information technologies 12 48–55 [in Russian]
[4] Vaishnavi V K 1984 Multidimensional height-balanced trees IEEE Transactions on Computers 33(4) 334–343
[5] Samet H 1990 Applications of Spatial Data Structures (Boston: Addison-Wesley Longman Publishing Co.)
[6] Kronacker M and Banks D 1995 High-concurrency locking in R-trees Proc. of the 21th Int. Conf. on Very Large Data Bases (San Francisco: Morgan Kaufmann Publishers Inc.) pp 134–145
[7] Beckmann N, Kriegel H, Schneider R and Seeger B 1990 The R*-tree: an efficient and robust access method for points and rectangles Proc. of the 1990 ACM SIGMOD Int. Conf. on Management of data (NY: ACM) pp 322–331
[8] Sellis T, Roussopoulos N and Faloutsos C 1987 The R+-tree: a dynamic index for multidimensional objects Proc. of the 13th Int. Conf. on Very Large Data Bases (San Francisco: Morgan Kaufmann Publishers Inc.) pp 507–518
[9] Berchtold S, Keim D A and Kriegel H 1996 The X-tree: An Index Structure for High-Dimensional Data Proc. of the 22th Int. Conf. on Very Large Data Bases (San Francisco: Morgan Kaufmann Publishers Inc.) pp 28–39
[10] Ciaccia P, Patella M and Zezula P 1997 M-tree: An Efficient Access Method for Similarity Search in Metric Spaces Proc. of the 23th Int. Conf. on Very Large Data Bases (San Francisco: Morgan Kaufmann Publishers Inc.) pp 426–435
[11] White D A and Jain R 1996 Similarity indexing with the ss-tree Proc. of the Twelfth Int. Conf. on Data Engineering (Washington: IEEE Computer Society) pp 516–523
[12] Katayama N and Satoh S 1997 The sr-tree: an index structure for highdimensional nearest neighbor queries Proc. of the 1997 ACM SIGMOD Int. Conf. on Management of data (NY: ACM) pp 369–380
[13] Jain R and White D A 1996 Similarity Indexing: Algorithms and Performance Proc. SPIE Storage and Retrieval for Image and Video Databases IV vol 2670 ed I K Sethi, R Jain (San Jose: SPIE) pp 62–75
[14] Fuchs H, Kedem Zvi M and Naylor B F 1980 On Visible Surface Generation by A Priori Tree Structures Proc. of the 7th annual Conf. on Computer graphics and interactive techniques (NY: ACM) pp 124–133
[15] Bentley J L 1975 Multidimensional binary search trees used for associative searching Communications of the ACM 18(9) 509–517
[16] Bentley J L and Friedman J H 1979 Data structures for range searching ACM Computing Surveys 11(4) 397–409
[17] Robinson J T 1981 The K-D-B-tree: A search structure for large multidimensional dynamic indexes Proc. of the 1981 ACM SIGMOD Int. Conf. on Management of data (NY: ACM) pp 10–18
[18] Lomet D B and Salzberg B 1989 The hB-tree: A robust multiattribute search structure Proc. of the Fifth Int. Conf. on Data Engineering (Washington: IEEE Computer Society) pp 296–304
[19] Henrich A, Six H-W and Widmayer P 1989 The LSD tree: Spatial access to multidimensional point and non-point objects Proc. of the 15th Int. Conf. on Very Large Data Bases(San Francisco: Morgan Kaufmann Publishers Inc.) pp 45–53
[20] Ooi B, McDonell K J and Sacks-Davis R 1987 Spatial KD-tree: An Indexing Mechanism for Spatial Databases Proc. of the IEEE Int. Computer Software and Applications Conf. (Washington: IEEE Computer Society) pp 433–438
[21] Finkel R A and Bentley J L 1974 Quad Trees: A Data Structure for Retrieval on Composite Keys Acta Informatica 4(1) 1–9
[22] Zeng M, Zhao F and Zheng J 2013 Octree-based fusion for realtime 3D reconstruction Graph Model 75(3) 126–136
[23] Yiamilos P N 1993 Data structures and algorithms for nearest neighbor search in general metric spaces Proc. of the fourth annual ACM-SIAM Symposium on Discrete algorithms (Philadelphia: Society for Industrial and Applied Mathematics) pp 311–321
[24] Ohsawa Y and Sakauchi M 1983 BD-Tree: A New N-dimensional Data Structure with Efficient Dynamic Characteristics Proc. of the IFIP 9th World Computer Congress (Amsterdam: North-Holland/American Elsevier) pp 539–544
[25] Nievergelt J, Hinterberger H and Sevcik K C 1984 The Grid File: An Adaptable, Symmetric Multikey File Structure TODS 9(1) 38–71
[26] Hutflesz A, Six H-W and Widmayer P 1988 Twin grid files: Space optimizing access schemes Proc. of the 1988 ACM SIGMOD Int. Conf. on Management of data (NY: ACM) pp 183–190
[27] Whang K-Y and Krishnamurthy R 1991 The Multilevel Grid File - A Dynamic Hierarchical Multidimensional File Structure Proc. of the 2nd Int. Symp. on Database Systems for Advanced Applications (Singapore: World Scientific Press) pp 449–459
[28] Hutflesz A, Six H-W and Widmayer P 1990 The R-file: An efficient access structure for proximity queries Proc. of the Sixth Int. Conf. on Data Engineering (Washington: IEEE Computer Society) pp 372–379
[29] Kriegel H-P and Seeger B 1986 Multidimensional order preserving linear hashing with partial expansions Proc. Int. Conf. on Database Theory (Berlin: Springer) pp 203–220
[30] Kriegel H-P and Seeger B 1988 PLOP-hashing: A grid file without directory Proc. of the Fourth Int. Conf. on Data Engineering (Washington: IEEE Computer Society) pp 369–376
[31] Potapov D R, Artemov M A, Baranovskii E S and Seleznov K E 2017 EXISTING METHODS OF KEY-VALUEstorages construction for using in adaptive data storages REVIEW Cybernetics and programming 5 14–45 [in Russian]
[32] Orenstein J and Merritt T H 1984 A class of data structures for associative search Proc. 3rd ACM SIGACT-SIGMOD Symp. on Principles of Database Systems (NY: ACM) pp 181–190
[33] Faloutsos S 1986 Multiatribute hashing using Gray-codes Proc. of the 1986 ACM SIGMOD Int. Conf. on Management of data (NY: ACM) pp 227–238
[34] Hilbert D 1984 Uber die stetige Abbildung einer Linie auf ein Flachenstiick Mathematischen Annalen 38 459–460
[35] Freeston M 1987 The BANG file: A new kind of grid file Proc. of the 1987 ACM SIGMOD Int. Conf. on Management of data (NY: ACM) pp 260–269
[36] Seeger B and Kriegel H-P 1990 The buddy-tree: An efficient and robust access method for spatial database systems Proc. of the 16th Int. Conf. on Very Large Data Bases (San Francisco: Morgan Kaufmann Publishers Inc.) pp 590–601
[37] Bayer R 1997 The universal B-tree for multidimensional indexing Proc. of the Int. Conf. on Worldwide Computing and Its Applications (London: Springer-Verlag) pp 198–209
[38] Faloutsos S and Kamel I 1994 Hilbert Rtree: An improved R-tree using fractals Proc. of the Twentieth Int. Conf. on Very Large Data Bases (San Francisco: Morgan Kaufmann Publishers Inc.) pp 500–509
[39] Moene-Lococo N 2005 High-Dimensional Access Methods for Efficient Similarity Queries Computer Vision and Multimedia Laboratory Technical Report N:0505 (University of Geneva)
[40] Potapov D R, Artemov M A and Baranovskii E S 2017 REVIEW ADAPTATION CONDITIONS OF ADAPTIVE ASSOCIATIVE DATA STORAGES Bulletin of the Voronezh State University. Series “System analysis and information technologies” 1 112–119 [in Russian]
[41] Kim K, Cha S K and Kwon K 2001 Optimizing multidimensional index trees for main memory access Proc. of the 2001 ACM SIGMOD Int. Conf. on Management of data (NY: ACM) pp 139–150
[42] Kim K-C and Yun S-W 2004 MR-Tree: A Cache-Conscious Main Memory Spatial Index Structure for Mobile GIS Proc. of the 4th Int. Conf. on Web and Wireless Geographical Information Systems (Heidelberg: Springer-Verlag Berlin) pp 167–180
[43] Gunther O and Gaede V 1997 Oversize shelves: A storage management technique for large spatial data objects Int. Journal of Geographical Information Science 11(1) 5–32
[44] Oosterom P 1994 *Reactive data structures for geographic information systems* (NY: Oxford University Press)

[45] Nelson R and Samet H 1987 A population analysis for hierarchical data structures *Proc. of the 1987 ACM SIGMOD Int. Conf. on Management of data* (NY: ACM) pp 270–277