HIERARCHICAL GRID STRUCTURES FOR STATIC GEOGRAPHIC DATA BASES

Andreas Kleiner and Kurt E. Brassel
Department of Geography
University of Zurich
Winterthurerstr. 190
CH-8057 Zurich, Switzerland

Abstract

Projects for the creation of large cartographic data bases and the advent of high volume optical disk technology create a need for structures for static mass data storage. This paper reviews existing approaches for structuring mass storage and proposes a scheme which combines the following concepts: a) separation of "background data" from "geographic object data"; b) use of hierarchical grid structures to define data buckets; c) coordinate shuffling to define keys for the spatial domain; d) use of efficient one-dimensional access structures (B-trees, Extendible Hashing); and e) pointer elimination due to the static properties of the data bases.

1. Introduction

Efforts are undertaken on both international and national levels to systematically collect base map information for digital data processing. The information contents of large area base map series are very high and the resulting data files are extremely voluminous. The use of adequate data base structures is thus a most important requirement for efficient utilization of large data bases. In recent years a multitude of data base models have been proposed. The present paper discusses some alternative storage approaches for data handling based on bulk memory spatial data bases. It is evident that storage efficiency cannot be defined in absolute terms, but it is dependent on the particular usage characteristics of the data. An example of such a large data base is the creation of a digital version of a worldwide 1:1 million map series, including hydrology, administrative boundaries, transportation features and relief (Bickmore 1986). It will be basically a worldwide multi-layered base for environmental sciences consisting of a mixture of point, line, area and volume types of geographic objects. As a product it is envisioned to be offered on one or multiple compact disks as a read-only store for easy distribution. Based on such a scenario we discuss some alternative storage structures for static base map elements of various characteristics. Dynamic map modification and updating is of no concern here; our aim is to provide a structure for fast search and retrieval of background elements. Though this information will be combined with a variety of "foreground" elements of the specific users, we shall disregard the aspects of foreground elements.

2. Existing Approaches for Structuring Mass Storage

Recent efforts of data handling in computer cartography were mainly concerned with data structures which did not need much space on external storage devices and were concentrated on efficient internal processing. Among them quadtree structures are of special interest; they appeared in the 1970s (see a review article by Samet 1984). Our discussion will also include the bintree or 2-d-trie (Knowlton 1980, Tamminen 1984, Orenstein 1982) which uses a binary rather than a quaternary tree structure, that is, each father node can have only two son nodes with alternate splits in each spatial direction. Two levels of a bintree correspond to one level of a quadtree.

Large data bases have to be stored on external storage media. Traditionally, bucket methods are used for this task. In the case of multidimensional data dealing with the geometric organization of space, bucket methods are also called cell methods (Tamminen 1981b). These are based on the principle of storing spatially adjacent objects in adjacent physical stores and pack the data into blocks or pages just large enough to read them into central memory by a single disk access. Various strategies exist to subdivide space into hierarchical buckets of cellular or grid shape.

Bucket methods can be grouped into two basic approaches (Nievergelt et al. 1984). One group of methods structures the data by trees of keys and searches on the basis of guiding discriminators ("signposts"), while the others partition the underlying (coordinate) space. The latter allow access to a directory or directly to the data by address computation. Some structures originally developed for internal processing may also be used for the management of data buckets. Initial research dealt with the access to one-dimensional data. The structure used most often is the B-tree, which is a generalization of the binary search tree as adapted for external block access operations (Bayer and McCreight 1972). A special variant suitable for range searches is the B+-tree (Comer 1979). Other schemes are Extendible Hashing (Fagin et al. 1979) which stores a bucket pointer for each potential hash value, and Linear Hashing (Litwin 1980) which allows hashing without a directory. Since multidimensional search problems are of special interest for the geo-sciences, efforts to develop multidimensional access methods were undertaken by both computer scientists and specialists in spatial data handling. Examples of structures include extensions of the B-tree such as the multidimensional B-tree (Scheuermann and Ouksel 1982) or the k-d-B-tree (Robinson 1981) as a combination of the B-tree and the k-d-tree (the multidimensional binary search tree; Bentley 1975). Proposed methods which organize data space are Multipaging (Merrett 1978, using directory scales), Dynamic Multipaging (Merrett and Otoo 1982), EXCELL (Tamminen 1981a; Extendible Hashing for multi-dimensional space), the Grid File (Nievergelt et al. 1984; similar to EXCELL, but with internal linear scales and an external directory) and methods of Linear Hashing (Ouksel and Scheuermann 1983, Burkhard 1983, Orenstein 1983). The Field Tree (Frank 1983b) as a quadtree structure is based on a regular partitioning of space and is accessed by tree traversal. Most of these structures are designed to manage dynamic data bases.

Some authors deal with the problem of storing spatial objects without splitting them

along cell borders. Most methods, however, split the objects and have to concatenate them each time they are extracted. Hinrichs (1985) proposed a method of storing objects through parameters (center point and half side-length of the enclosing box) in higher dimensional space. However, access becomes extensive due to conical search areas. Frank (1983b) and Abel and Smith (1983) used a quadtree organization of space to store objects in the smallest enclosing quadtree cells. Frank (1983a) introduced a derivation of the quadtree where the cell origins are systematically shifted for the respective hierarchical levels; this allows all objects to fit a bucket cell of appropriate size and avoids that small objects at high level quadrant borders have to be placed in high level cells due to the border coincidence. To our knowledge Frank did not follow up on this method due to the complexity of access processes (Frank 1983b).

Another important concept for multidimensional data handling is coordinate "shuffling", the operation of bitwise interleaving coordinate keys, resulting in so called Peano keys (Peano 1973). It was developed in the domain of GIS for the transformation of multidimensional space into one dimension and results in "linear quadtrees" (or linear bintrees; see Peuquet 1984, Samet 1984). Access on these keys may be provided by methods for one-dimensional data (Tamminen 1981a, Abel and Smith 1983). There has also been a growing interest in this concept in the domain of conventional data base research (Tropf and Herzog 1981, Ouksel and Scheuermann 1983, Burkhard 1983, Orenstein 1984).

3. Hierarchical Grid Structures for Static Geographic Data Bases

The purpose at hand is to design storage strategies for spatial data on read-only external mass storage devices. A first requirement is to minimize both access time and external storage space. Under the assumption that information used for mapping or spatial analysis is highly clustered, bucket methods are typically used for this task.

Another major issue is the definition of the objects to be stored in the grid cell storage buckets. Spatial objects by definition may be of point, line, area or volume type. The latter three may be arbitrarily cut into parts by cellular grid systems. Depending on the application this may or may not be acceptable. For the purpose of this discussion we distinguish between those objects which can be subdivided along grid borders and those that should be stored in their integrity within one single data bucket. We call the former "background data", the latter "geographic object data". Background data can be stored and retrieved more efficiently as long as objects are not to be reconstructed.

In our task of storing large volume spatial data on read-only memory we propose to combine the following concepts: a) separation of "background data" from "geographic object data" and use of specific spatial bucket or cell methods for both cases; b) coordinate shuffling in the spatial domain; c) use of efficient one-dimensional access structures; and d) taking advantage of the static properties of the data to reduce storage overhead and processing time.

In our presentation we shall proceed as follows: First we present the basic concepts and specific schemes for the background data (3.1), then explain the specific organization of the geographic object data (3.2), and close with a discussion of some alternatives for the organization of keys and adaptations for static data bases (3.3).

3.1 Hierarchical Bucket Methods for Storing Background Information

Background data include point features and elements of linear and area objects that are separated at grid cell borders. A reconstruction of object integrity and topology is not required. Information pertaining to adjacent objects should be stored whenever possible in one data bucket. For internal storage quadtree or bintree structures appear to be optimal. On the other hand these structures are not suited for external storage of the tree nodes because tree traversal is inefficient. Analog to the development of the B-tree from the binary search tree where the number of sons of each father is so big as to occupy one page, we could extend a quadtree to a hierarchical grid structure with a whole matrix of son cells for each father cell. This minimizes the number of disk accesses and reduces search time not only by physical clustering of adjacent data, but also by minimizing reading operations in the access structure. This concept is illustrated in Figure 1. While solving one problem we diminish spatial flexibility, reduce cell occupancy and thus processing efficiency. To alleviate these problems we separate the organization in the space domain from the organization of bucket access: Space is subdivided into hierarchical cells (e.g., quadtrees, bintrees) which are then managed by shuffled keys and one-dimensional access structures (e.g., B-trees). The specific cell units represent data buckets containing background elements in form of point and string data. The size of the cells is a function of the information density in space. The model used most frequently is the quadtree. However, in cases where other than raster type data are to be handled bintrees are even more flexible. If a bucket is overfilled, it is first split into two sons instead of four. Figure 2 shows the bintree for a hypothetical map. All cells are labeled by shuffled codes interleaving bits in x and y directions. The number of digits used implicitly indicates the cell size or hierarchical level. The shuffled codes of the data buckets are stored in B-trees (or other access structures) where the B-tree elements again are packed in pages or "index-buckets" (Figure 3a).

3.2 Storing Geographic Object Data

For the storage of spatially extended objects which have to be handled as integral entities we will use an adaptation of the cell organization proposed by Frank (1983a). In order to find a cell into which an object of arbitrary form and size can fit, cells of all levels of the hierarchy may be used in the same structure. A regular quadtree is chosen as proposed by Abel and Smith (1983), but with a displacement of the grid at each level. Small objects crossing high level borders now fit lower level cells, thus avoiding to force too many objects into the top cells. Instead of an implementation of rather complicated "trees" where sons can belong to two or four fathers (i.e., graphs), we propose to use bit interleaving of coordinates. As point of reference we

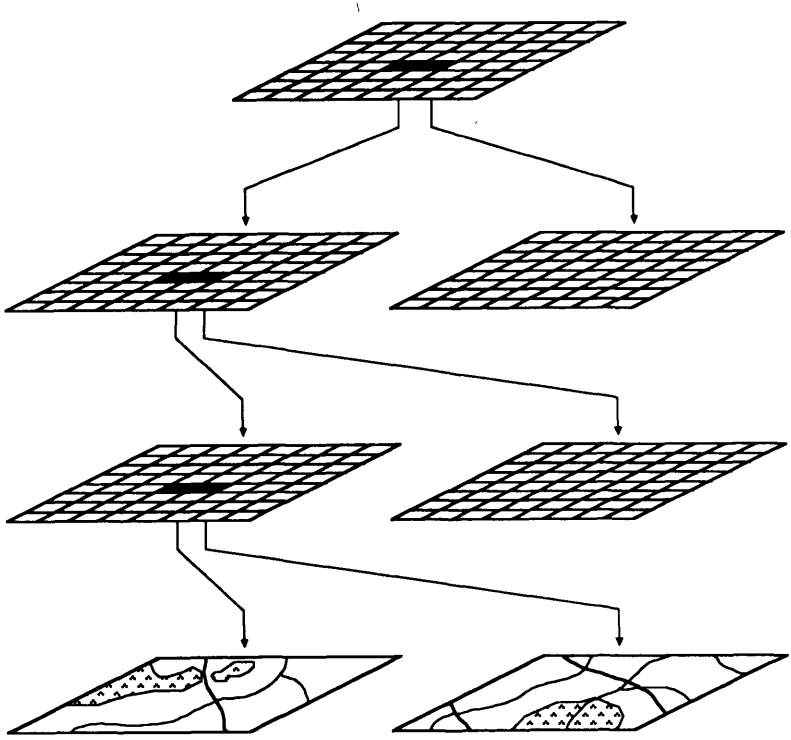

Figure 1: Hierarchical grid structure

either choose the lower left corner or the center of each cell with the addition of a suffix specifying the level and implicitly the displacement with respect to a regular matrix. In the former case each cell has a unique key which avoids the need for a level suffix. This scheme is illustrated in Figure 4. The area of interest is bound by heavy lines. In the example this is a single cell of Level 1; it is overlaid by cells of Levels 2 and 3. Objects that fit into cells of Level 3 are addressed by Level 3 keys, objects that cross Level 3 borders in cells of Levels 2 or 1. Cells at all levels include pointers to a data store where the sequence order is defined by the shuffled location keys of the lower left corner of the cells. Not all values of the domain of the keys are actually used. Each father has 9 sons, of which 4 sons are shared with one neighbor and 4 sons with three neighbors. Range searches are performed by the sequential traversal of the tree where each father is visited after the first three sons and before the remaining six sons.

The list of shuffled codes again is sorted and packed into B-tree pages. Each shuffled code is associated by a pointer to a data bucket. Each bucket may include a number of objects of various types: point, line area or volume objects with or without topological identifiers. For a polygonal network data buckets may include both arc

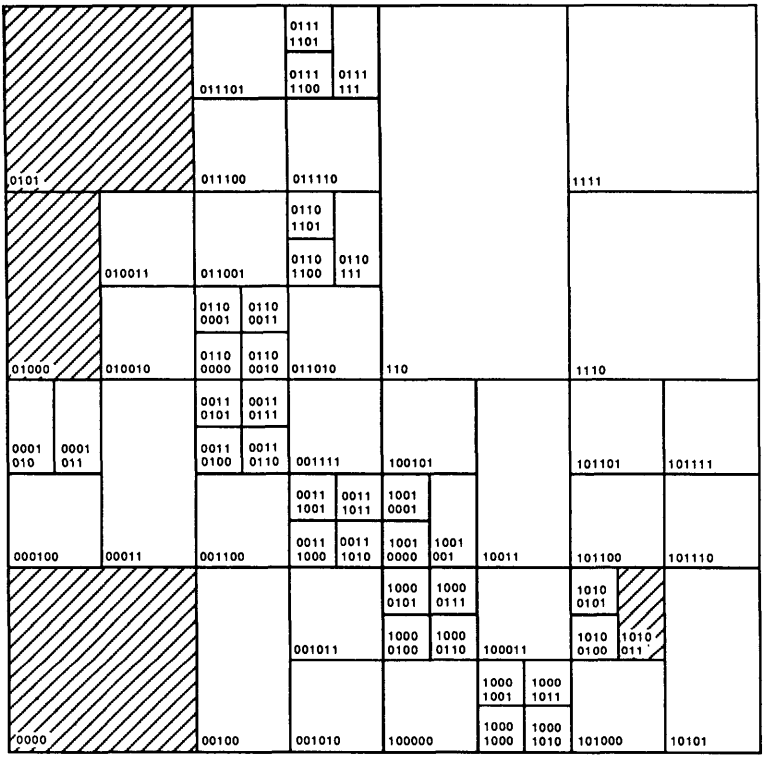

Figure 2: Bintree organization for background data (empty cells are shaded)

(1-cell) and polygon (2-cell) objects. Arc objects may be defined by a sequence of points and pointers to adjacent arcs and polygons, polygon objects by strings of arc identifiers, etc.

In general, a polygon description may not happen to be stored in the same bucket as its arcs. It may, therefore, be reasonable to define a unique object identifier for all objects of any type. This identifier consists of the bucket code plus a sequential counter within each bucket. Such a scheme enables searches based on objects independent of spatial criteria.

3.3 Alternative Key Organizations

In the previous discussion we have proposed B-trees as an index structure to the data buckets. Due to the static nature of the data base, the presorted buckets are stored at fixed physical locations on a mass storage device. Since the data are sorted by shuffled keys, this results in the important advantage that sequential searches in space

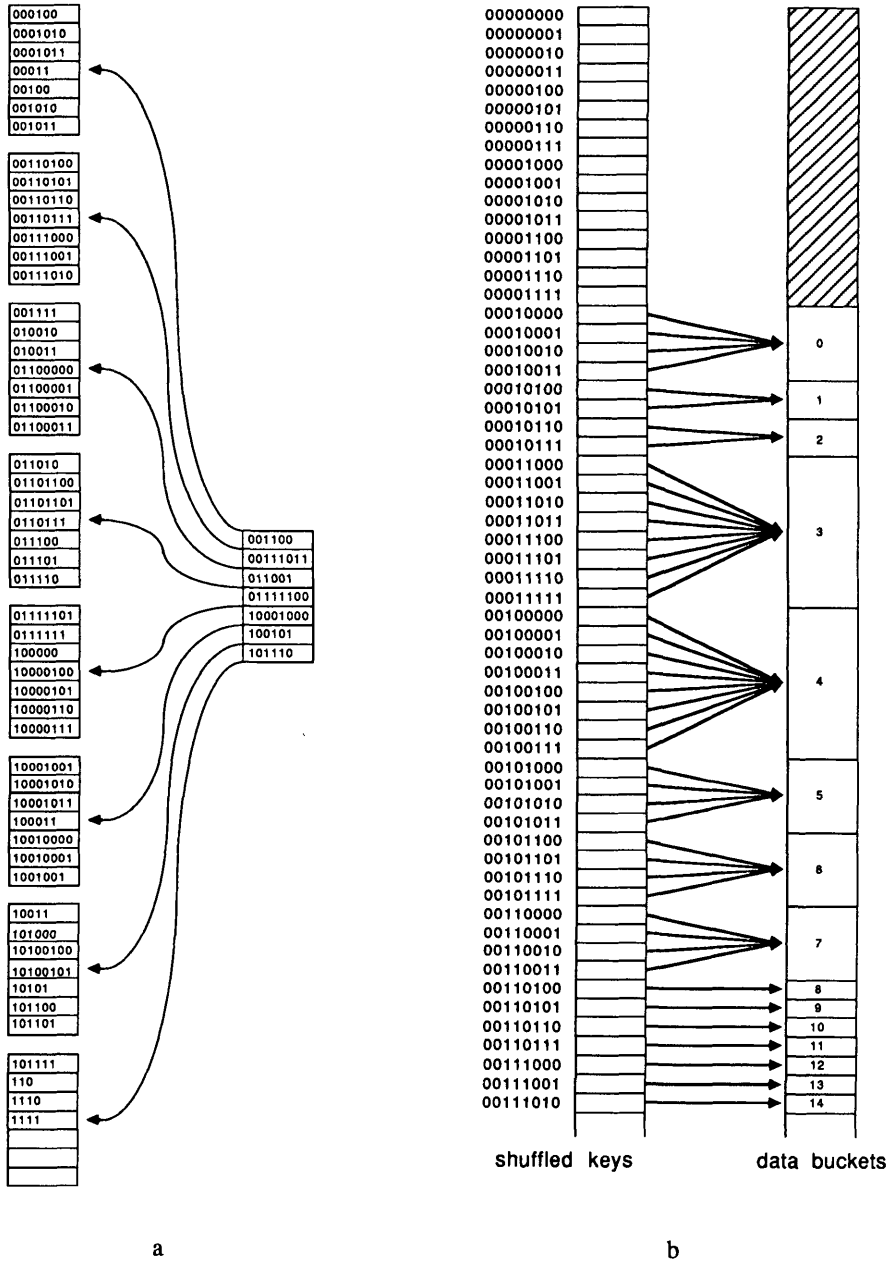

Figure 3: Access structures for the bintree in Figure 2: a) B-tree b) EXCELL

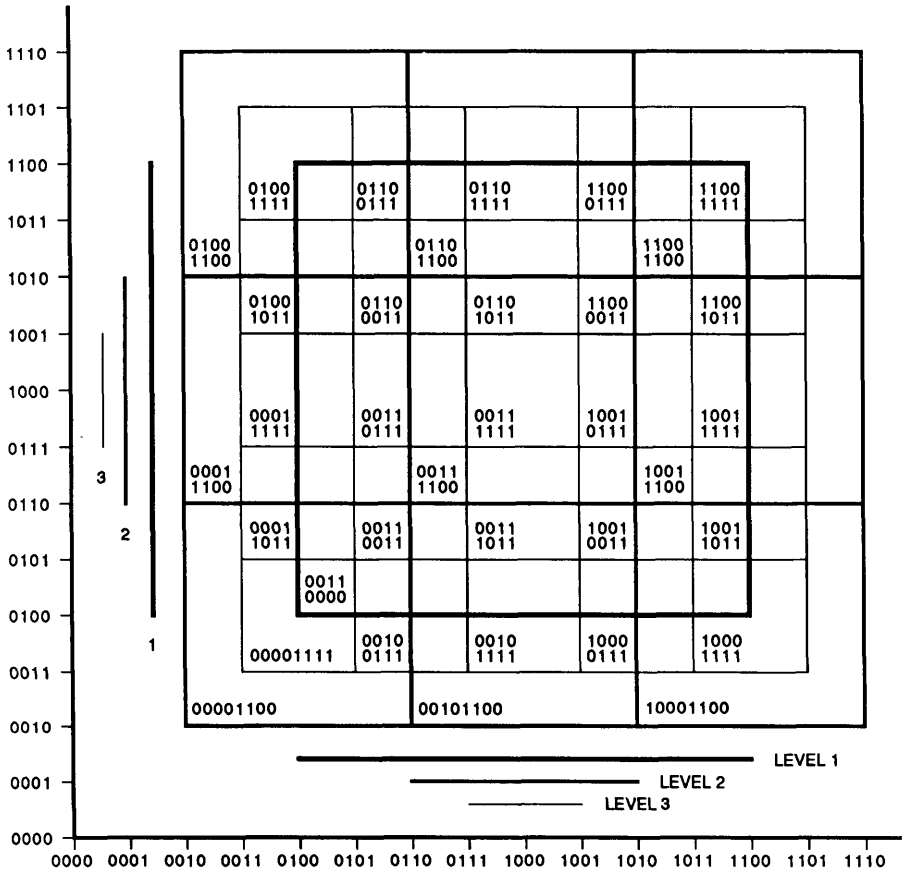

Figure 4: Hierarchically displaced quadtree structure for geographic object data

can be executed by sequential reading from disk. We therefore do not need a B+-tree for range search as proposed by Abel (1984). Also, if the structure of the B-tree is exhaustively defined, all pointers to B-tree pages and data buckets can be eliminated since their external storage position can be calculated from the position of the cell in the B-tree. Analog to "linear quadtrees" we call this structure a "linear B-tree".

Under certain circumstances the use of Extendible Hashing (EXCELL, Tamminen 1981a) may be a valid alternative. In contrast to the B-tree all area keys of smallest cell size are members of the directory. For higher level cells all pointers of its constituting subcells point to the same bucket. Area cells of any size can thus be addressed directly. For each area unit a pointer to the respective bucket location on mass storage is maintained. Figure 3b visualizes this alternative. The Extendible

Hashing option may be inferior to the B-tree option with respect to storage requirements but superior for random data access time. As we have pointed out above, not all values in the domain of keys are actually occupied in the case of geographic object storage. Since Extendible Hashing, however, uses entries for all potential keys, storage is inefficient. It is therefore advisable to compress the shuffled keys by a compression function. For each cell the compressed keys then indicate the position of the pointer to the bucket in mass storage.

4. Conclusions

We have discussed various storage structures for read-only spatial bulk data. We recommend to separate background data from geographic object data. Background data (points and spaghetti lines cut at cell borders) are preferably stored in spatial buckets defined by bintrees which are labeled by shuffled area codes and organized in a B-tree structure. Due to the static nature of our task, pointers to buckets are not required. A valid alternative is the use of Extendible Hashing (EXCELL). Geographic object data are packed in hierarchically displaced structures of quadtree cells, which are identified by shuffled area codes. The sorted codes are organized either for B-tree or Extendible Hashing access. In the latter case compression of shuffled keys is advisable. In a static environment all proposed structures allow for sequential retrieval of data buckets without the use of an index structure. Final decisions on the type of structures depend on the specifics of the map data to be recorded and the circumstances of their use. A major decision will be as to which elements will be stored as background or geographic object data. Other parameters are the level of detail to be recorded, i.e., the total volume of background or object data. Given this quantity for both types of stores, a decision has to be taken with respect to the cell resolution for background and object stores; this decision is related to the size of the data buckets used.

Future efforts shall be devoted to the determination of the specific parameters of a real-world project and the analysis of its specific usage profile. Additional work relates to the creation of algorithms for the construction of the static data base and to system implementation and testing.

Acknowledgements

This project is supported by a contract of the Swiss government. Mr. A. Herzog and Dr. H. Kishimoto have kindly reviewed the text. These contributions are gratefully acknowledged.

References

Abel, D. J. and J. L. Smith (1983): "A Data Structure and Algorithm Based on a Linear Key for a Rectangle Retrieval Problem", **Computer Vision, Graphics, and Image Processing,** Vol. 24, 1, pp. 1-13.

Abel, D. J. (1984): "A B+-Tree Structure for Large Quadtrees", **Computer Vision, Graphics, and Image Processing,** Vol. 27, 1, pp. 19-31.

Bayer, R. and E. McCreight (1972): "Organization and Maintenance of Large Ordered Indexes", **Acta Informatica,** 1, pp. 173-189.

Bentley, J. L. (1975): "Multidimensional Binary Search Trees Used for Associative Searching", **Communications of the ACM,** Vol. 18, 9, pp. 509-517.

Bickmore, D. P. (1986): "World Digital Database for Environmental Science", **Nachrichten aus dem Karten- und Vermessungswesen,** Series II, 44, pp. 13-14.

Burkhard, W. A. (1983): "Interpolation-Based Index Maintenance", **BIT,** 23, pp. 274-294.

Comer, D. (1979): "The Ubiquitous B-Tree", **Computing Surveys,** Vol. 11, 2, pp. 121-137.

Fagin, R., J. Nievergelt, N. Pippenger and H. R. Strong (1979): "Extendible Hashing - a Fast Access Method for Dynamic Files", **ACM Transactions on Database Systems,** Vol. 4, 3, pp. 315-344.

Frank, A. (1983a): **"Datenstrukturen für Landinformationssysteme - semantische, topologische und räumliche Beziehungen in Daten der Geo-Wissenschaften",** Mitteilungen aus dem Institut für Geodäsie und Photogrammetrie an der Eidgenössischen Technischen Hochschule Zürich, Nr. 34, Zürich: ETH, 116 pp.

Frank, A. (1983b): **"Probleme der Realisierung von Landinformationssystemen, 2. Teil: Storage Methods for Space Related Data: The FIELD TREE",** Eidgenössische Technische Hochschule Zürich, Institut für Geodäsie und Photogrammetrie, Bericht Nr. 71, Zürich: ETH, 63 pp.

Hinrichs, K. H. (1985): **"The Grid File System: Implementation and Case Studies of Applications",** Diss. ETH Nr. 7734, Zürich, 111 pp.

Knowlton, K. (1980): "Progressive Transmission of Grey-Scale and Binary Pictures by Simple, Efficient, and Lossless Encoding Schemes", **Proceedings of the IEEE,** 68, 7, pp. 885-896.

Litwin, W. (1980): "Linear Hashing: A New Tool for File and Table Addressing", **Proceedings Very Large Data Bases,** Montreal, pp. 212-223.

Merrett, T. H. (1978): "Multidimensional Paging for Efficient Data Base Querying", **Proceedings of the ACM ICMOD,** Milan, pp. 277-289.

Merrett, T. H. and E. J. Otoo (1982): "Dynamic Multipaging: A Storage Structure for Large Shared Data Banks", **Proceedings of the Second International Conference on Databases: Improving Usability and Responsiveness** (Ed. P. Scheuermann), New York: Academic Press, pp. 237-255.

Nievergelt, J., H. Hinterberger and K. C. Sevcik (1984): "The Grid File: An Adaptable, Symmetric Multikey File Structure", **ACM Transactions on Database Systems,** Vol. 9, 1, pp. 38-71.

Orenstein, J. A. (1982): "Multidimensional Tries Used for Associative Searching", **Information Processing Letters,** Vol. 14, 4, pp. 150-157.

Orenstein, J. A. (1983): "A Dynamic Hash File for Random and Sequential Accessing", **Proceedings Very Large Data Bases** (Ed. M. Schkolnick and C. Thanos), Florence, pp. 132-141.

Orenstein, J. A. and T. H. Merrett (1984): "A Class of Data Structures for Associative Searching", **Proceedings of the Third ACM SIGACT-SIGMOD Symposium on Principles of Database Systems,** Waterloo, Ontario, pp. 181-190.

Ouksel, M. and P. Scheuermann (1983): "Storage Methods for Multidimensional Linear Dynamic Hashing", **Proceedings of the Second ACM SIGACT-SIGMOD Symposium on Principles of Database Systems,** Atlanta, Georgia, pp. 90-105.

Peano, G. (1973): **"Selected Works",** Ed. H. C. Kennedy, Toronto, Toronto University Press.

Peuquet, D. J. (1984): "A Conceptual Framework and Comparison of Spatial Data Models", **Cartographica,** Vol. 4, pp. 66-113.

Robinson, J. T. (1981): "The K-D-B-Tree: A Search Structure for Large Multidimensional Dynamic Indexes", **Proceedings of the ACM SIGMOD,** pp. 10-18.

Samet, H. (1984): "The Quadtree and Related Hierarchical Data Structures", **ACM Computing Surveys,** Vol. 16, 2, pp. 187-260.

Scheuermann, P. and M. Ouksel (1982): "Multidimensional B-Trees for Associative Searching in Database Systems", **Information Systems,** 7, 2, pp. 123-137.

Tamminen, M. (1981a): **"The EXCELL Method for Efficient Geometric Access to Data"**, Acta Polytechnica Scandinavica, Mathematics and Computer Science Series No. 34, Helsinki: Univ. of Technology, 57 pp.

Tamminen, M. (1981b): "Order Preserving Extendible Hashing and Bucket Tries", **BIT**, 21, pp. 419-435.

Tamminen, M. (1984): "Comment on Quad- and Octrees", **Communications of the ACM**, 27, 3, pp. 248-249.

Tropf, H. and H. Herzog (1981): "Multidimensional Range Search in Dynamically Balanced Trees", **Angewandte Informatik**, 2, pp. 71-77.