Indices and Data Structures in Information Systems

Algimantas Juozapavièius, Richard E. Blake
(VU, Lithuania, NTNU, Norway)

Abstract

One of the key issues in information systems is to provide a fast and reliable access to data. This is true for conventional databases and information systems as well as for spatial or multimedia ones. Main tools for such aim are indices, among others. Search and reasoning operations using indices and structured data require a specific support on logical and physical level. A years of research have resulted in a great variety of multidimensional data structures and indices (access methods also). This paper overviews recent trend in the area of multidimensional, spatial, temporal indices and data structures discusses their principles and implementation issues. The special emphasis is given to research information systems, as well as to multimedia and spatial ones.

Keywords: information systems, multidimensional access methods, data structures, indices, search and reasoning, research and spatial data

Introduction

An information system (IS) may be defined as an integrated, user-machine system for providing information to support the operations, management and decision-making functions in an organization [8]. At the structural level, it is made up of a set of components or subsystems that captures, processes, stores, analyses, condenses, and disseminates information in various forms. Traditionally, information systems are text-oriented which provide reports, documents, and decision-making information for all levels of the hierarchy within an organization [11]. It is characterized by a text-in/text-out mode of operation, focusing primarily on structured fields and free text. However, this style of IS becomes obsolete since information is no longer text-based, but instead it is based on a combination of text, audio, video, image together with the semantic and spatio-temporal relationships among them.

Most recent information systems (knowledge-based IS, research IS, spatial or visual IS, multimedia IS, etc.) can be defined as an integrated, user-machine system for providing inter-related knowledge-based, visual and multimedia information to support the research or other operations, management and decision-making functions in an organization [3]. The inter-relationships between different multimedia data may signify relationships between the same media type (intra-media relationships) or between different media types (inter-media relationships). A distinction may be sometimes made between a visual IS and a multimedia system, where the latter tends to be more concerned with the system and support aspects [10]. A research or spatial IS, on the other hand, is concerned with the semantics, and possibly pragmatics, of multimedia information which occurs at a higher level much closer to the user.

Research IS are relevant to scientific and statistical data management. The operations include modeling and semantics, process models, query languages and user interfaces, data visualization, metadata management, knowledge discovery and data mining, probabilistic DBs, temporal data, spatial data, experiment data management, and tertiary storage management for scientific data [10]. Results address issues arising in specific scientific disciplines, as biological sciences, chemical sciences, earth and space sciences, environmental and climate sciences, geology, medicine, high energy physics, and social sciences.

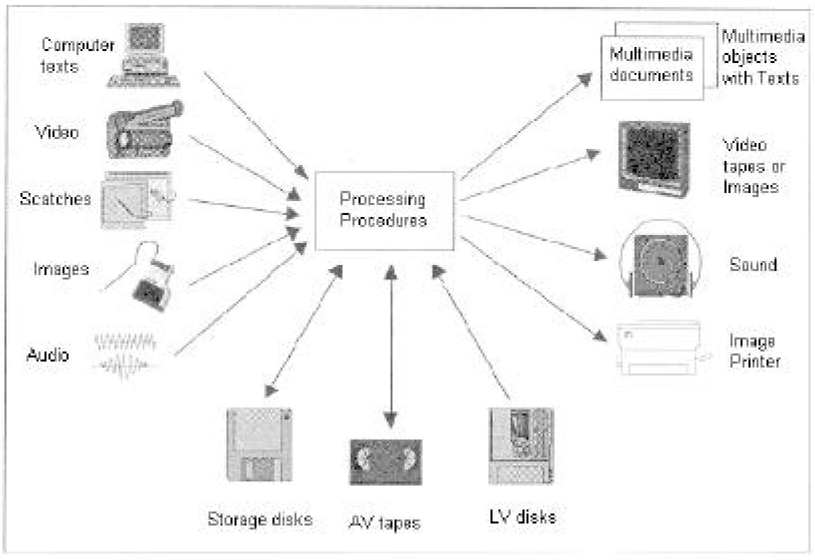

Fig.1. Visual information system

Figure 1 illustrates the functionality of a visual information system. In order to be a general-purpose IS the system must be able to handle a variety of information sources such as text, sound, graphics, images and video. In particularly, the contents of these sources ought to be properly indexed and easily accessible.

The indexing strategies for conventional information systems are well studied, many architectures of popular indices like dictionaries, catalogs, word-lists, etc. are explored and evaluated. The strategies for search procedures based on such indices are explained in details [11].

This is not the case for multimedia, visual or some other contemporary information systems. Methods of search and retrieval for multimedia information, as well as ones for spatial reasoning are just in their very development. The presence of multimedia platforms in IS, equipped with audio and video facilities, large memory, large disk storage and fast 1/0 for the effective handling of multimedia data, possibly based on multimedia chips or general-purpose chips with rich multimedia instructions, are affecting design and implementation of indices and fast search and reasoning procedures. The affect is valid for all the components of such indices, like data structures, logical schemes and algorithms.

The aim of this article is to provide a survey covering principles, methods and inventions in multidimensional, spatial data structures, content-based indices, and related algorithms. The necessary comparisons and conclusions for some approaches are included.

1. Data Structures and Indices

Data structures are used to design algorithms, especially for manipulations with data like sorting, searching, and editing of structured data, etc. [6]. There are other terms relevant to data structures: *abstract data types (ADT), and data types*. An *ADT* is defined as a mathematical model of the data objects that make up a data type, as well as the functions that operate on these objects. The operations that manipulate the data objects are included in the specification of the ADT. Then the term *data type* refers to the implementation of the mathematical model specified by ADT, so it is a computer representation of an ADT. The term *data structure* refers to a collection of computer variables that are connected in some specific manner. Classical examples of ADTs are stacks, queues, trees, heaps, etc. Data types like lists and matrices correspond to so-called *built-in data types*, while being implemented in some programming language. In many cases however a design of computer program will call for data types, so-called *user defined data types*, which can involve the construction of quite complicated data structures. User defined data types are main objects in constructions of data, designed for efficient procedures of information manipulations. Software engineering procedures are focusing on two different viewpoints for data structures: logical view and implementation view [6]. The logical view is used during program design and simply means model provided by the ADT specification. The implementation view of a data type considers the manner in which the data elements are represented in memory, and how the accessing functions are implemented. For an ADT there is only one logical view of a data type, and there may be many different approaches to implementing it.

However ADTs do not resolve all aspects of an algorithm design. They are usually unable to provide the fastest and reliable way for the whole procedure of data manipulation, like search and retrieval, etc. Additional objects have to be designed and implemented. For large information systems such objects usually are so-called *indices.* The indices are used to:

- provide a quick and easy access to data;
- save time and operations in editing, searching, inserting, deleting of data;
- provide additional services while designing queries, analyzing the content of data, etc.

In traditional databases indices correspond to dictionaries and catalogs.

The situation has been changed very much in recent years with the introduction or invention of hypermedia and multimedia data, with spatial and temporal database systems. Of course, the boom of computer resources available created the background for the hypermedia and multimedia to come into life. The hypermedia (nonlinear text), multimedia (integration of text, images, graphics, sound and video recordings), spatial and temporal data have been changing the notions of data structures and indices drastically. There were many new data structures suggested, especially for multidimensional, spatial and temporal data – data that can’t be sorted in a totally order. Indices were influenced and forced to change by many nonlinear search strategies appeared, as well as that techniques of designing and implementing indices have to be applied to data, quite different from the text – graphical, audio, visual ones.

Hypermedia systems as well as multimedia ones that store and present vast amounts of multimedia data are interconnecting them densely by a rich variety of hypertextual links. The multimedia database systems are reasonable to use when it is required to administrate a huge amounts of multimedia (MM) data objects with different types of data media (optical storage, video tapes, audio records, etc.). These data have to be used (that is, efficiently accessed and searched for) as many times as needed. Typical operations for the storage and retrieval of MM-data include:

- input of MM objects; composition of MM-objects; archive of data (in hardware and format independent way, these operations are typical for a storage of objects);
- support of complex search; efficiency (indices etc.); evaluation (aggregation, filtering); preview; also conversions (needed to gain or lead to hardware and format independence, these operations are typical for search and retrieval of objects).

Elements of multimedia data are identified as presented in table 2.

Medium	**Elements**	**Pattern**	**Typical size**	**Timing**	**Sense**
text	Printable Characters	Sequence	10 KB (5 pages)	no	visual/ acoustic
graphic	vectors, regions	Set	10 KB	no	Visual
raster image	Pixels	Matrix	1 MB	no	Visual
audio	sound/ volume	Sequence	600 MB (audioCD)	yes	Acoustic
video-clip	Raster image/ Graphics	Sequence	2 GB (30 min.)	yes	Visual

Table 2. Characterization of elements of multimedia data.

Comparison of multimedia databases (MMDBS) or MM information systems to the traditional ones reflects important differences:

- the search procedure converts to a reasoning of data – instead of receiving a strict and specific detailed answer from the search, just a kind of reasoning could be expected;
- the indices are no more hierarchical or consisting of objects totally sorted.

These differences originate from the nature of multimedia data – it's hard to apply a strict search procedure to a collection of images, where only approximate answers are available. The hypermedia and multimedia systems are also including basic layers (database layer, user interface and application layer, anchoring layer, some others), which are interacting to each other in a complicated way. For the design and implementation of MMDBS, new concepts are needed [8, 10]. Some of these concepts are proposed by *reference models*, defining the relationship between different layers and a way of interaction of them. Most popular of these models are:

- hypertext abstract machine (HAM);
- link engine/hypermedia engine/link service/hypermedia toolkit;
- hypermedia design model (HDM);
- Dexter hypertext reference model;
- Amsterdam hypermedia model (as an extension of Dexter hypertext reference model, adding time and context);
- Trellis hypertext reference model (or r-model);
- some others.

Perhaps the most popular reference model used as a background in numerous hypermedia and especially multimedia systems is the Dexter hypertext reference model, the main schematics is presented in the figure 3.

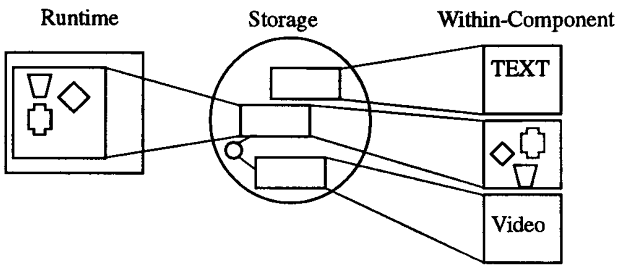

Fig. 3. The main schematics of Dexter reference model.

This model suggests three layers (runtime layer, storage layer, within-component layer) and two interfaces (presentation specification interface between the layers of runtime and storage, and anchoring interface between storage and within-component layers). According to this model the environment for user application programs may be specified independently from the data processing procedures. Using layered architecture, proposed by any of reference models above, a hypermedia system is basically providing support for navigation through the links that have been made between the multimedia documents, and for the presentation of the multimedia documents themselves. The models don't provide a *dynamic creation* of layers needed for navigation, conceptually they are considering just two levels (a level of *interface* and a level of *hyperbase*), and operations *present* and *navigate* to express the logic between these levels.

Many authors have proposed various schematics involving a distinctions to be made between the *data* (content of the node in a *hyperindex*) and the *paradata* (content of data used to index a content of a node). There are other terms used for the same purpose: *hyperbase/hyperindex*, *document network/concept network*, etc. The schema including special level for index in hypermedia databases possibly looks like in figure 4.

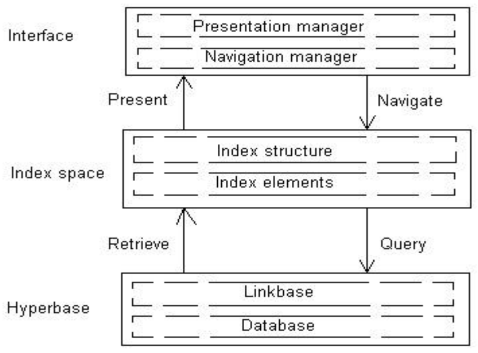

Fig. 4. The levels of hypermedia database with an indexing means

This basic schema of architecture of multimedia systems is usually more detailed in reference models and in numerous applications.

2. Index Structures

The concepts and algorithms to create an index of hypermedia information, while designing and implementing an application, have to follow specific principles and constrain. The diversity of multimedia data and their use constitute the role for an index structure:

- it has to capture and designate what the concept data are about;
- the index has to be defined what it can be used for;
- the index has to be described how it is used as a search retrieval structure.

An index structure encompasses:

- the internal structure defined for the concepts (their attributes, allowable attribute values. etc.);
- the external structure defined between these concepts (the logical and content relations that are defined between the concepts, how these relations are represented, etc.).

The design requirements that such a concept-based index structure has to fulfill are demanding:

- the index has to find a correct balance between representational power and practical usability, it should capture the useful semantics of the hypermedia information, while remaining understandable by the users;
- the index has to represent a model that is similar to the user's model of the hypermedia information, it should correspond with the user’s view on the contents, reducing the effort needed to understand and use it;
- the index has to be more intelligent than the content of the hypermedia information itself, it should be able to capture and represent every possible navigation path the user might want to take through the content.

Several concept-based index structures have been explored, each trying to address the demands in different ways. Some index structures are familiar from classic information retrieval (thesaurus, faceted thesaurus, concept lattice), some have been developed specifically to index hypermedia information (hyperindices, semantic hyperindices, etc.), and some have been derived from AI knowledge representation formalisms (inference network, semantic network, etc.) [1].

2.1. Thesaurus

A thesaurus is the most widely used index structure in conventional information retrieval systems. It consists of a set of concepts and a limited set of relationships between these concepts. Three types of interconcept relationship are represented: *equivalence* (preferred/nonpreferred equivalent concepts), *hierarchical* (broader/narrower concepts), *associative* (related concepts).

As a result, the thesaurus consists of a standardized, controlled vocabulary of concepts that are hierarchically structured into a single inheritance tree. The major advantages of a thesaurus-based index structure are its flexibility and its intelligibility. The major drawback is that a lot of efforts are needed for thesaurus construction and validation. Although tools and techniques have been developed for the computer-assisted creation of' thesauri, most existing thesauri have been carefully handcrafted (e.g.. the well-known ***MeSH*** thesaurus). There are just a few applications of thesauri to index hypermedia information.

2.2. Faceted Thesaurus

Faceted thesaurus is an indexing technique in which concepts are classified into separate hierarchical structures and in which each hierarchy captures a different viewpoint of the documents, as in example presented in figure 5. A faceted thesaurus consists of a number of different thesauri, and each thesaurus is used to index the documents with respect to some different knowledge domain. The main advantage of using faceted thesaurus is that this index structure allows for greater exhaustivity and precision in the hypermedia indexing process. The disadvantages are the same as for those of conventional thesaurus, that is the creation of the thesauri and the validation of their hierarchical structure is very time-consuming, and these procedures are difficult to make computer-aided ones.

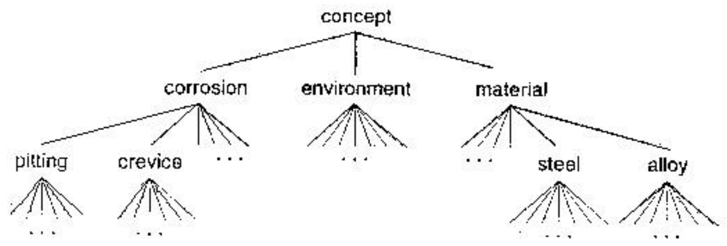

Fig. 5. The schematics of faceted thesaurus

2.3. Concept Lattice

A concept lattice is an extension (a powerful one) of the thesaurus index structure. This structure can be described mathematically – a concept lattice is a partially ordered set of concepts in which every pair of concepts has both a greatest lower bound (a unique narrower concept) and a least upper bound (a unique broader concept), as presented in figure 6.

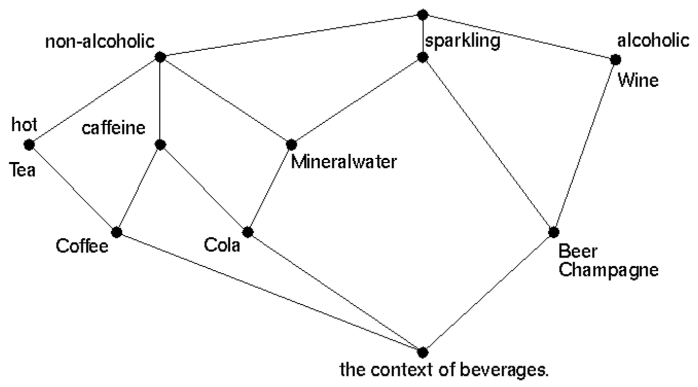

Fig. 6. The schema of a concept lattice for the concept of beverages

The notion of a concept lattice is very common in AI, in so-called *ontologies.* A principle approach called *formal concept analysis* provides a conceptualization called a *concept lattice*, and ontology is a specification of a concept lattice. In practice, ontology specifies the named part of a concept lattice called a *concept space.* Based upon the ontological relations, concept lattices and concept spaces are faceted, with each facet representing a distinct dimension of information. Then formal concepts with the subtype ordering form a class hierarchy, together with the named class of attributes. Every pair (in fact, any subcollection) of concepts has an *associated meet* concept corresponding to logical *and* an *associated join* concept corresponding to logical *or.* Tree hierarchies, for example file system hierarchies, are very special cases of lattices; they need just a bottom node to be added.

The major advantage of a concept lattice is that it can represent more flexible hierarchical structures than an ordinary or faceted thesaurus. There also exists a complete set of mathematical techniques that can be used *to create* concept lattices and check their internal consistency.

An example of information system, using concept lattice is given by the *WorldView* system [1], which is designed to process electronic news, articles, and abstracts of technical reports. Documents are automatically indexed and classified with respect to a lattice of concepts derived from *the IEEE Inspec* *thesaurus*. The WorldView retrieval engine interprets a user's query relative to this lattice of concepts, and then restricts the lattice to the sublattice relevant to the query. Using this sublattice, it can find the narrower concepts that can be used to extend the scope of the original query.

2.4. Hyperindices

Hyperindexing is an indexing technique that was specifically developed for hypermedia information. In this method, the content of a document is characterized by constructing an index expression (a set of index terms and connectors between these index terms) from the title of the document. From such an index expression one can derive the so-called *power index expression*, which forms a lattice like structure of index expressions that can then be used as a hypertext of indices. Each vertex in this lattice can be considered as a predefined query to the document space that can be enlarged (made less specific) or refined (made more specific) by moving respectively to the descendent or ancestor vertices in the lattice of index expressions. The example of a hyperindex is presented in figure 7.

The experiments have shown however, that for the retrieval purposes hyperindices are at least as effective as faceted thesauri. However hyperindices are believed to be superior to both:

- *collocation* (the degree to which the relevant index terms are near each other);
- *exhaustivity* (the degree to which the content of a document is reflected by index terms).

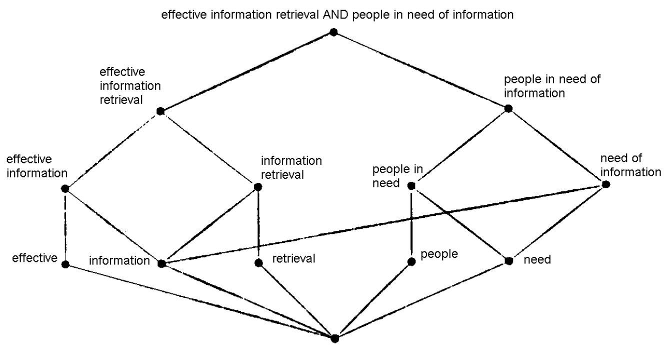
effective information retrieval AND people in need of information
effective information retrieval
people in need of information
effective information
information retrieval
people in need
need of information
effective
information
retrieval
people
need

Fig. 7. The hyperindex graph for titles of documents

2.5. Semantic Hyperindices

The strength of the hyperindexing technique lies in the fact that the lattice of hyperindices can be generated *automatically* from the concepts characterizing the node contents. However, when building these hyperindices the technique does not take into account how these concepts may possibly relate to each other semantically. To overcome this limitation, the semantic-aware version of hyperindices was suggested: so-called *semantic hyperindices.* The semantic hyperindexing technique introduces the use of associations or relationships between concepts belonging to different knowledge domains. These associations try to express which combinations of concepts are either inherently valid or potentially interesting from it usage point of view. They circumscribe the subsets of concepts that can be meaningfully taken together at the same time.

Domain-specific associations express which combinations of concepts are inherently valid with respect to the knowledge domains to which these concepts belong (e.g., certain combinations of concepts are excluded since they are not possible in theory or not pertinent in practice). *Usage-specific* associations express, which combinations of concepts should be considered together for specific kinds of readers and for specific kinds of tasks. This use of associations allows to fine-tune the lattice of hyperindices by excluding certain combinations of concepts that were generated by the hyperindexing technique and by including other combinations that would never have been generated by the hyperindexing technique. Using the semantic indices it is also possible to develop useful numerical metrics to characterize the degree of information overlap of the nodes' contents.

2.6. Inference Network

In an inference network as in figure 8, nodes represent concepts, and links represent dependence relations between these concepts. An inference network consists of two component networks:

- document network, representing the document collection;
- query network, representing the user’s information need.

The experiments with an inference network have shown that this index structure is very effective for hypertext information retrieval.

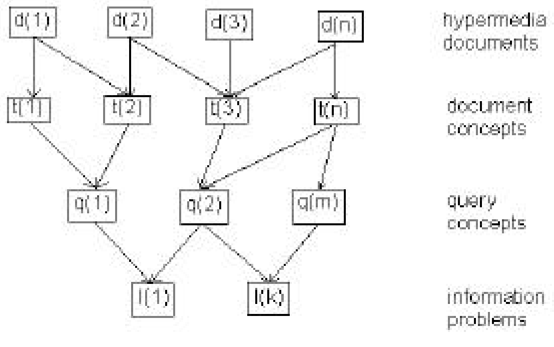

Fig. 8. The schematics of an inference network.

2.7. Semantic Network

In a semantic network nodes represent concepts, and links represent semantic relations between these concepts, as in figure 9. The indices based on semantic network are frequent in many applications.

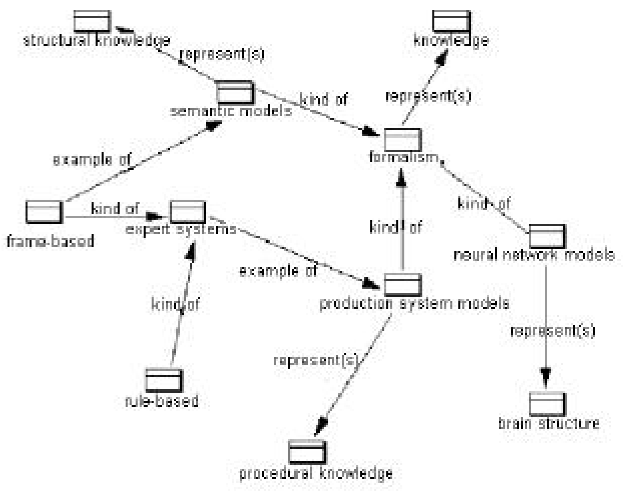

Fig. 9. The classes in semantic network of Mindtools

The system like Mindtools provide different, more constrained formalisms for representing personal knowledge [15]. Mindtools, like text, require that learners represent domain knowledge in a knowledge base with formal, constrained statements of relationships between the ideas in the knowledge base. However, the relationships and fundamental structures that are generated by each formalism vary. There are many classes of Mindtools, as presented in figure 9, and each class of Mindtools engages different critical thinking skills and involves different syntax and representational formalisms.

Semantic networks have the much richer internal organization of relations then thesauri. The node-link-node structure of a semantic network is conceptually close to the hypertext documents and links itself, so it supports browsing of hypertext documents in a natural way. It has been shown that using a semantic network it is possible to develop more robust and efficient retrieval mechanisms, provided the relations between the concepts are chosen with the user's typical retrieval tasks in mind. However, identifying the important concepts in a knowledge domain and the relations between these concepts is a challenging task. These concepts are organized into a semantic network, and the system allows a structured exploration of the resulting concept space:

- by matching a personal information representation against the concepts and relationships between the concepts;
- then by retrieving the corresponding citations.

Another example of utilization of semantic network is given by the *VISAR* system that is a part for the *Memex* system. Corresponding *Beyond web site* is a major research, educational, and collaborative web site, integrating the historical record of and current research in hypermedia. The site is very tightly interlinked through graphical, spatial, and textual representations of the relationships among the people, projects, institutions, publications, conferences, and themes that comprise the hypermedia community. *Memex* and *Beyond* is an outreach web site of the *NSF Graphics and Visualization Center*, which is NSF (National Science Foundation) Science and Technology Center. The Center is a consortium of five universities, including *Brown University, Caltech, University of North Carolina, University of Utah, and Cornell University*.

3. Index Acquisition

Many designs, approaches and implementations are considering three major steps while creating a concept-based index structure:

- extraction of index terms or notions;
- refinement of these terms (or notions);
- implementation of the relationship between these concepts.

There are a number of promising approaches for index structure creation and representation developed recently. Most of these approaches are computer-assisted ones, not fully automated.

- *Indexing in Context.* Some systems are indexing technical documents in a hypermedia framework, and the correspondence relations between the concepts and the documents they refer to are modified by using interactive user feedback to either reinforce or correct the system's knowledge in case of success or failure.
- *Question-Based Indexing.* In other systems question-based indexing refers to it hypermedia system that facilitates the indexing and retrieval of design documents in technical engineering. The system can acquire conceptual indices of text, graphics, and videotaped documents on the basis of the user's questions.
- *Conversational Indexing.* The conversational indexing refers to a large hypermedia system where the user is guided through hypermedia documents on the basis of a conversational model of hypertext navigation.
- *Agglomerative Hierarchic Clustering.* The agglomerative hierarchic clustering refers to an experimental information retrieval system that provides tools for textual analysis and concept clustering, and for which a hypertextual interface that uses concept cluster hierarchies to improve the navigational search process was built.
- *Interactive Taxonomic Classification.* The interactive taxonomic classification refers to a set-based hypermedia system designed to support taxonomic reasoning. Nodes are organized in sets on the basis of their similarity with respect to one or more attributes. The user can sort nodes into sets based on a particular number of attributes, examine the different sets of which that a node is a member and generate a new sets from old ones.

Such index structures as thesauri and similar ones are receiving some attention in the design of digital libraries. Other topics of indices are been explored in more theoretical way [1, 4].

4. Data Structures for Index Implementation

For the design, implementation and usage of index structures, some special support at the physical level is needed. The same is true for search operations as well. Such operations and requirements lead to access methods [4, 12, 13]. For the conventional databases the need includes well-known techniques for creating and maintaining of dictionaries, catalogs, and other similar means. The most popular data structure in this case (let's say a case of total-sorted data) is *a B-tree*. This structure can work without substantial differences with just the primary memory, as well as with the primary and secondary memories [12].

The hypermedia or multimedia systems are dealing with data, which are difficult to put into linear order. The index structures for such systems require essentially different access methods. For example, search operations in *spatial* databases include *point queries* (find all objects that contain a given search point) and the *region queries* (find all objects that overlap a given search region). Researches on-going in more then 20 years have resulted with a great variety of multidimensional access methods to support such operations [4].

One of the most popular access methods suggested for visual and graphics data is a *quad-tree;* the example is given in fig. 10. In order to organize several objects in a quadtree, each of them is taken into its minimum quadrant, and a spatial index for these objects is created, using alternatively a linear quadtree and a hierarchical quadtree. The index clearly reveals the different entities retrievable at a given spatial resolution. Within each unit, which consists usually of two pieces of data, the one value indicates the location, and the other is the alphanumerical identifier.

Sometime fractal points are regrouped into quadrants given a nice possibility to use quadtrees. This is also valuable because they provide the ability to store objects with different sizes. Consequently, let's say geographical objects of large areal extents will be located near the root of the tree and small objects in the terminal leaves.

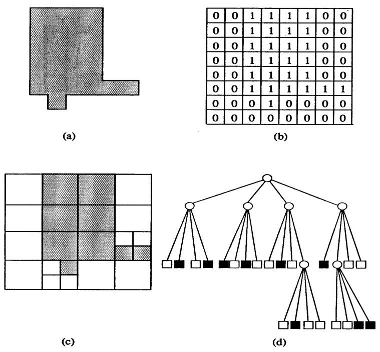

Fig. 10. The image (a), it's partiton and the quadtree (d)

The family of R-trees introduces other suitable and challenging data structures for spatial navigation, index structures and index acquisition. They are exploring a possibility for spatial indexing to use extents bounding spatial objects. One alternative is to use minimum-bounding rectangles, organized either in R-trees or in R'-trees, example of R-tree is presented in figure 11.

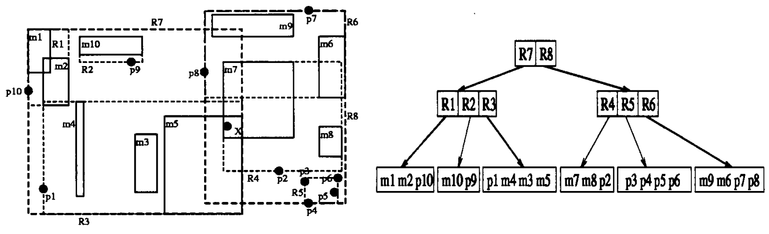

Fig. 11. The image and it's R-tree of the bounding boxes

For R-trees, objects are bounded by rectangles, usually of different size, and adjacent rectangles are regrouped within a bigger pseudo-rectangle. By repeating this operation, a hierarchy of rectangles is constructed with the result that the number of tests to access the point required vary with the logarithm of the number of objects.

To make the operations of grouping and retrieving objects more efficient, many branches of R-trees were suggested, as R$^+$-tree in figure 12 or R*-tree in figure 13, each of these trees are designed to fit the best way to spatial information with specific attributes. The examples of various R-trees below are designed for the same image as in figure 11.

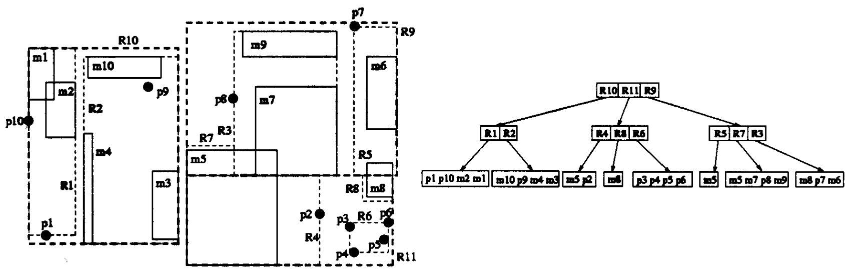

Fig. 12. R$^+$-tree for the image of figure 11.

The R-trees are sensible to the spatial distribution of rectangles, in their splitting and balancing operations. The R*-tree is always the most balanced. An

R*-tree encoding with the relational model of data will give the same relations as for the R-trees.

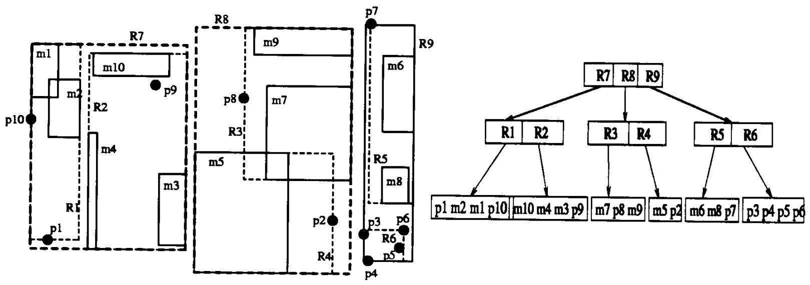

Fig. 13. R*-tree for the image of figure 11.

The main drawback of minimum-bounding rectangles for spatial information is that this way of spatial indexing is very sensitive to orientation. Some other methods have been proposed based on spheres and polygons, like trees enclosing objects by circles (or spheres at three dimensions) [4]. Even though it is often not easy to compute the circle, it is obvious that the extent of this geometric figure is not orientation sensitive. Moreover, this kind of spatial indexing is insensitive to orientation if the axes are rotated. Perhaps the main challenge is to find a method to determine automatically the bounding circle or sphere for any object; afterwards the addition or deletion of objects is not a problem. Cell trees give another possibility to index, when each object is bounded by a convex polygon. The main challenge is to determine rapidly the convex polygon for bounding the objects, especially the number of sides.

Faloutsos and Rong [4] have combined the R- tree and fractals by a so-called double transformation. A point in a four-dimensional space (the min-X, min-Y, max-X and max-Y) can represent rectangles, defined by minimum and maximum x and y; this represents the first transformation. Then, all 4D points representing rectangles are ordered by four-dimensional Hilbert or Peano keys, being the second transformation. Their results show that 4D Hilbert keys give the better performance for their criteria.

Today almost all database systems use B-trees as their main access method. One of the main drawbacks of the classical B-tree is, however, that it works well only for one-dimensional data, when data are totally ordered. R. Bayer, the inventor of B-tree, suggested a new access structure, called UB-tree (for universal B-tree) for multidimensional data. The UB-tree is balanced and has all the guaranteed performance characteristics of B-trees:

- it requires linear space for storage;
- it requires logarithmic time for basic operations of insert, search, and delete.

In addition the UB-tree has the fundamental property – it preserves clustering of objects w.r.t. to Cartesian distance. Then, the UB-tree shows its main strengths for multidimensional data. It has very high potential for *parallel processing*. With this method, a single UB-tree can replace an arbitrary number of secondary indexes. For updates this means that only one UB-tree must be managed instead of several secondary indexes. For queries and in particular range queries the UB-tree has multiplicative complexity instead of the additive complexity of multiple secondary indexes. The UB-tree is useful for geometric databases, datawarehousing and datamining applications.

5. Indexing in computer vision/object recognition

Indexing in recognition activities usually arises when part or all of an unknown or a reference is to be used to extract a list of reference classes that have at least some commonality with the data that led to the index.

Two concrete examples can be given: pose estimation and graph matching. Pose estimation, for example [9], seeks to answer the question: if the observed object were to be a member of class X then what is the most probable (list of) pose(s) that is most consistent with the processed image data? Given data structures that represent the appearance of an object of class X at different points on the viewing sphere, the processed image data of a real object is used as an index into this data structure to extract appearances similar to that of the object and their position on the viewing sphere can be obtained [2]. It is possible that, for each result of indexing, a region of the viewing sphere is obtained rather than a single point.

Pose estimation is a pattern recognition problem: one in which the output is a viewing specification rather than a class in the usual sense. In this case it is data from an unknown object that creates the index. Single or double subgraph matching of non-planar graphs is a NP-complete activity [5]. This means that in order for practical matcher to be based on the process an effective control strategy must be devised. This strategy can include the use of problem partitioning. The entire control and partitioning can be expressed as indexing. A decision strategy that arranges the selection of one out of a set of patterns as a sequence of binary choices implies a number of tests proportional to the logarithm of the total number of classes which are goal states.

It can be shown that by computing a sequence of indices, where each successive index represents an improved approximation to a goal state, the binary decision tree can be created as an inverted lattice. Problem partitions can be naturally included in this scheme as they are brought into play in successive indices. In this case it is data from reference objects that give rise to the indices.

6. Conclusions

As reflected in this article, research in spatial databases, and especially in hypermedia and multimedia information, has resulted in a wealth of various indexing approaches and principles, as well as in spatial access methods. This situation not only suggests variety of selections, but makes it difficult to recognize their merits and faults. Every new method seemed to claim superiority to at least one existing methods discovered previously. So the time and tests are needed to make the right decisions.

Despite of numerous efforts in research of indexing methods of spatial and multidimensional data, as a practical matter, only a few commercial spatial information systems today provide spatial indexing capabilities [7]. Some systems allow access to database objects via graphic cursor input for points or boxes or other shapes. Otherwise there is access via names or numerical identifiers in the attribute data tables. Sometimes topological neighborhoods provide a means of access, by following line segment or graph links for a specified polygon or line. Indexing capabilities are much rarer. If an information system makes indexing tools available, separate indices for both attributes and the spatial domain can be created, combining spatial approaches like adaptive grid-cells, with a binary searching mechanism, operating on data stored as modified binary (B-) trees.

The task of spatial indexing is very challenging. At present there are several techniques but none emerges as the best, although some form of hierarchical organization is generally advantageous. Moreover, two main secondary issues must also be solved: multi-layer indexing, and taking the physical disk structure into account. In several practical situations, spatial databases are split into several layers, each of them concerning a particular theme. But when it is desirable to work with several thematic layers within one cover area, then the layers must be combined adequately.

Evaluating researches that are presented in numerious papers, it has to be noticed there are so many different parameters that define optimality, and so many parameters that determine performance. First of all, the efficiency of an access method strongly depends on the data to be processed. An access method that performs reasonably well for rectangles may fail for arbitrarily oriented lines. Strongly correlated data may render an otherwise fast access method irrelevant for any practical application. Robustness against varying system parameters such as page size is also a desirable requirement. Furthermore, even large numbers of insertions and deletions should not deteriorate the structure. Likewise, an access method should provide efficient support for a broad range of queries. Insertions and reorganizations should be possible with little overhead. The access method should guarantee a certain minimum storage utilization, preferably independent of the dimension of the data. Further criteria an access method should meet are simplicity and scajability.

Hence, it is far from easy to compare or rank different access methods. Nevertheless, the implementation and experimental evaluation of access methods is essential as it often reveals deficiencies and problems that are not obvious from the design or a theoretical model.

Researches are using the Internet to test their models more and more often. Especially new trends in Internet-based scripting and programming techniques are of value. Many existing extensions of the WWW, may provide the right technological base for such a paradigm change and for getting an experience on the indexing and access methods proposed. This gives additional value to the Internet.

References

1. Arents H. C., Bogaerts W. F. L. (1996). Concept-Based Indexing and retrieval of Hypermedia Information. In Encyclopedia of Library and Information Sciences (supplement volumes), Vol. 58, Academic Press, New York, pp. 1-29.
2. Boros P., Blake R. E. (1995). The Calculation of the Aspect Graph from a CAD Model, Proceedings of the Second Asian Conference on Computer Vision.

3. Deryn G. (1997). Knowledge-Based Image Processing Systems. Springer-Verlag, London, 179pp.
4. Gaede V., Guenther O. (1995). Multidimensional Access Methods, preprint, http://www.wiwi.hu-berlin.de/~gaede/, 66pp.
5. [CC] Garey M. R., Johnson D. S. (1979). Computers and Intractability, a Guide to the Theory of NP-Completeness, Freeman Pub.
6. Heileman G. L. (1996). Data Structures, Algorithms, and Object-Oriented Programming. The McGraw Hill Companies, New York, 446pp.
7. Laurini R., Thompson D. (1995). Fundamentals of Spatial Information Systems. The APIC series, Num. 37, Academic Press, London, 680pp.
8. Leung Cl., ed. (1997). Visual Information Systems. Lecture Notes in Computer Science, Vol. 1306, Springer-Verlag, Berlin, Heidelberg. 287pp.
9. [AA] Matas J, Soh L. M., Kittler J. (1997). Object Recognition using a Tag. Proceedings of International Conference on Image Processing,
10. Nwosu K. C., Thuraisingham B., Berra P. B., ed. (1996). Multimedia Database Systems: Design and Implementation Strategies. Kluwer Academic Publishers, 381pp.
11. Salton G. (1988). Automatic Text Processing – The Transformation Analysis, and Retrieval of Information by Computer. Addison-Wesley Publishing Co., Reading, MA.
12. Samet H. (1990). Applications of Spatial Data Structures: Computer Graphics, Image Processing, and GIS. Addison-Wesley Publishing Co., Reading, MA, 507pp.
13. Samet H. (1990). The Design and Analysis of Spatial Data Structures. Addison-Wesley Publishing Co., Reading, MA, 507pp.
14. Shepherd B. (1993). Multimedia Hypermedia Model and Framework. In. Tutorial Notes from ACM Multimedia Conference, 59pp.
15. Yacci, M., Jonassen, D.H., Beissner, K., (1993). Structural knowledge: Techniques for assessing, conveying, and acquiring structural knowledge. Hillsdale, NJ: Lawrence.

Authors:

Algimantas Juozapavièius, associate professor, Faculty of Mathematics, Vilnius University, Lithuania, recently: visiting professor at Department of Computer

and Information Sciences, Norwegian University of Science and Technology, Trondheim, Norway
e-amil: juozapav@idi.ntnu.no or algimantas.juozapavicius@maf.vu.lt

Richard E. Blake, professor, Department of Computer and Information Sciences, Norwegian University of Science and Technology, Trondheim, Norway
e-mail: richard.blake@idi.ntnu.no