First publ. in: Database issues for data visualization : proceedings / IEEE Visualization '93 Workshop, San Jose, California, USA, October 26, 1993. - Berlin [u.a.] : Springer, 1994. - pp. 210-229. - (Lecture Notes in Computer Science ; 871). - ISBN 3-540-58519-2

Using Visualization to Support Data Mining of Large Existing Databases

Daniel A. Keim, Hans-Peter Kriegel

Institute for Computer Science, University of Munich
Leopoldstr. 11B, D-80802 Munich, Germany
{keim, kriegel}@informatik.uni-muenchen.de

Abstract. In this paper, we present ideas how visualization technology can be used to improve the difficult process of querying very large databases. With our VisDB system, we try to provide visual support not only for the query specification process, but also for evaluating query results and, thereafter, refining the query accordingly. The main idea of our system is to represent as many data items as possible by the pixels of the display device. By arranging and coloring the pixels according to the relevance for the query, the user gets a visual impression of the resulting data set and of its relevance for the query. Using an interactive query interface, the user may change the query dynamically and receives immediate feedback by the visual representation of the resulting data set. By using multiple windows for different parts of the query, the user gets visual feedback for each part of the query and, therefore, may easier understand the overall result. To support complex queries, we introduce the notion of 'approximate joins' which allow the user to find data items that only approximately fulfill join conditions. We also present ideas how our technique may be extended to support the interoperation of heterogeneous databases. Finally, we discuss the performance problems that are caused by interfacing to existing database systems and present ideas to solve these problems by using data structures supporting a multidimensional search of the database.

Keywords: Visualizing Large Data Sets, Visualizing Multidimensional Multivariate Data, Data Mining, Visual Query Systems, Visual Relevance Feedback, Interfaces to Database Systems

1. Introduction

The need for database system support visualization systems has been widely recognized and has been a main focus of two previous workshops which were held in conjunction with the *SIGGRAPH '90* and *Visualization '91* conferences. The main question that has been dealt with is how database technology can adequately support visualization systems. In this context, researchers are working on extending current object-oriented database management systems, designing adequate database schemas and formats that allow storing and accessing the large amounts of data which are needed by visualization systems. The question of database support for visualization, however, is only one side of the coin; the other side is visual support for databases. In this paper, we take up the question, how to visually support database users in accessing, analyzing and understanding the growing amount of data that is stored in the computer.

The progress made in hardware technology allows today's computer systems to store very large amounts of data. The available storage space is easily filled with data that is often automatically recorded via sensors and monitoring systems. Today, even simple transactions of every day life, such as paying by credit card or using the telephone, are typically recorded by using computers. Even larger amounts of data are generated by automated test series in physics, chemistry or medicine and satellite observation systems are expected to collect one terabyte of data every day in the near future [FPM 91]. The data of all areas mentioned so far is collected because people believe that it is a potential source of valuable information providing a competitive advantage (at some point). Querying and analyzing the databases to uncover the valuable information hidden in them, however, is a difficult task. Traditional database query languages such as SQL [ISO 92] allow people to query the databases, but finding the data a person is interested in is often a problem. Even experienced database users may have difficulties to find the interesting hot spots. Since, in general, the user does not exactly know the data and its distribution, many queries may be needed to find the interesting data sets. The result for most queries will contain either less data than expected, sometimes even no answers, so-called 'NULL' results, or more data than the user is able to deal with. With today's database systems and their query tools, it is only possible to view quite small portions of the data. If the data is displayed textually, the amount of data that can be displayed is in the range of some one hundred data items but this is like a drop in the ocean when dealing with millions of data items. Having no possibility to adequately query and view the large amounts of data that have been collected because of their potential usefulness, the data becomes useless and the database becomes a data 'dump'.

The need for supporting the process of querying and analyzing databases has been widely recognized and was even ranked one of the most important topics of database research for the 90s [SSU 90]. The US government, for example, sponsors large projects, such as the Sequoia 2000 project, to develop advanced data analysis techniques for very large databases. Many companies also recognized the potential of analyzing their databases. Banks and retail stores, for example, analyze their transaction records to understand customer habits better and thus, tailor their marketing promotions accordingly. Banks also analyze loan and credit history to improve their loan approval policies. Over the last several years, many tools and algorithms for data analysis have been developed. It seems, however, that advanced techniques for data analysis are not yet mature - at least for the flood of data we are facing today. Since, on the other hand, the technology for generating, collecting and storing data is available, the gap between the amount of data that has to be analyzed and the amount of data that can be analyzed is growing.

Visualization technology seems to provide important potentials to improve the process of querying, analyzing and understanding the data. With visualization techniques, larger amounts of data can be presented at the same time on the screen, colors allow the user to instantly recognize similarities or differences of thousands of data items, the data items may be arranged to express some specific relationships and so on. To our knowledge, up to this point visualization techniques are only used in databases in the rare cases where the data has some inherent two- or three-dimensional semantics. In geographic databases, for example, 2D visualizations are used to adequately support spatial queries and basically all Geographic Information Systems (GIS) provide such visual representations of the data. In most application areas, however, the data does not lend itself to an easy visualization and, therefore, in most cases no visual support for querying the database is provided. We believe that in dealing with very large amounts of data, visual support allowing database users to profit from the progress made in visualization hard- and software, is essential to support the process of exploring the data.

Most of the data collected in the past is stored in relational database management systems. In particular, large (unstructured) data sets are usually stored in relational systems. However, for visualization purposes relational systems are not well suited, and often there is no other way to access the data than to completely extract the whole database. There are many reasons for using relational systems instead of better suited systems such as object-oriented database management systems. One of them is the proliferation of relational systems. Another reason is, as recent studies show, that object-oriented database technology is not yet mature for really large amounts of data. In the context of this paper, we focus on available databases and therefore, we mainly refer to the relational data model. However, most of the ideas that will be presented in this paper also work in conjunction with object-oriented or other database systems.

The rest of the paper is organized as follows: Section 2 elaborates on the tasks that are important to querying large databases. The term 'data mining' is introduced and discussed with respect to related areas of research. Section 3 gives an overview of our ideas to visually support the querying of large databases and describes the query specification and visualization interface of the VisDB system. In section 4, some of the problems in interfacing with (traditional) database systems are described and potential solutions are discussed. Section 5 summarizes our approach and points out some of the open problems for future work.

2. Data Mining

The process of searching and analyzing large amounts of data is also called 'data mining'. The large collections of data are the potential lodes of valuable information but, like in real mining, the search and extraction can be a difficult and exhaustive process. Therefore, for the mining to be successful, adequate and efficient mining tools are essential. In some sense, data mining is like the work of radiologists. It is like scanning the database to identify phenomena that need to be looked at, showing the regular structure of the data but also helping to find anomalies. Before we describe the differences of data mining to related research areas, we will first try to define the term 'data mining'.

2.1 Definition of Data Mining

'Data Mining' can be defined as the (non-trivial) process of searching and analyzing data, in particular to find implicit but potentially useful information. Let $D = \bigcup_{i=1}^{m} D_i$ with $D_i = \{d_1, ..., d_{n_i}\}$ be the data set to be analyzed. Note that the D_i do not need to be stored in the same database management system. The process of data mining can be described as the process of finding a subset D' of D and hypotheses H_U (D', C) about D' that a user U considers *useful* in an application context C. The hypothesis may describe properties of the data set D', it may identify relationships between subsets of D' or it may be a combination of both. Our definition can be further formalized, e.g. by defining a hypothesis description language. a context description formalism and so on. The user and his/her notion of 'usefulness', however, can hardly be formalized since 'usefulness' not only depends on the changing knowledge of the user and the application domain, but it also includes some notion of creativity and users may not be able to define their usefulness criteria. On the other hand. if a data mining tool helps the user to find useful D' and to find and verify hypotheses, then it may not be important to have the hypothesis, the context and so on formally specified. All these aspects are present in the mind of the user who will also be able to express and communicate his/her ideas towards other humans.

2.2 Related Research Areas

Our definition of data mining is a quite broad definition and relates to a wide range of other research areas including statistics (data analysis, cluster analysis), artificial intelligence (knowledge discovery, machine learning), database interfaces (data browsing, cooperative database interfaces) and information retrieval. In the following, we give a brief overview of these areas.

Simple statistical parameters such as average, variance or correlation coefficients only allow special kinds of hypotheses. namely those with D' = D or D' = D_i in the case where D is partitioned into relations $D_1, ..., D_m$. More complex statistical methods such as multidimensional cluster analysis and mathematical taxonomy [DE 82] try to find hypotheses about real subsets of the database ($D' \subset D$ with $|D'| \ll |D|$ and $|D'|$ sufficiently large). An exhaustive cluster analysis of multidimensional data would require checking of relationships between all combinations of dimensions for all subsets of data items which is computationally intractable for large data sets. Therefore, most cluster analysis algorithms use some kind of heuristics to reduce the search space (e.g. [Hub 85, GSSK 87, GPP 90]). Still. for really large databases with millions of data items, cluster analysis is not feasible without human guidance. Additionally, statistical methods are not suited for nominal and structured data and often lead to results that are difficult to interpret. Furthermore, statistical methods do not help to find single exceptional data, so-called hot spots. In our context, we talk about hot spots if $D' \subset D$ and $|D'| = 1$ or sufficiently small when compared to |D|.

In artificial intelligence, researchers are working in related fields of knowledge discovery and machine learning which can also be considered to be data mining. Among the AI techniques used in data mining are inverted expert system approaches, probabilistic theories, bayesian statistics, neural networks and genetic algorithms. In contrast to our approach, in knowledge discovery the hypotheses are usually rules or facts which are formally specified in some high-level language [FPM 91]. In some cases, the knowledge is even not made available to the user to support hypothesis finding since it is extracted from samples, either unsupervised or supervised by experts and directly used for decision making in similar cases (well-known examples are [Qui 86, RM 86, HHNT 86]).

Another area related to data mining is database query interfaces. The ability to extract data satisfying a common condition is like data mining in its ability to produce interesting and useful hypotheses. Traditional query interfaces, however, do not help the user in finding interesting data. The usefulness of the results largely depends on the user's a priori knowledge and intuition. In many cases, however, it is like the search of a needle in a haystack. Many approaches have been made to improve the database query interface. One approach is graphical database interfaces that allow the user to browse the data (e.g. FLEX [Mot 90] or BAROQUE [Mot 86]). Another approach is cooperative database interfaces [Kap 82, ABN 92] that try to give 'approximate answers' in cases where the query does not provide a satisfactory answer. Such systems use techniques like query generalization [Cha 90], that is, dropping or relaxing a selection predicate in cases where the original query fails, and statistical approximation or intensional responses instead of full enumeration in the cases of large results (key ideas are already presented in [JKL 77] for the first time).

In the area of information retrieval, a lot of research has been done to improve recall and precision in querying databases of unstructured data such as (full) text. The search is usually supported by some kind of 'descriptor' that may be extracted automatically or assigned by the user. In this context, distance functions for text, strings or descriptors [HD 80], ranking functions [NMK 81, FM 91] and weighted queries [SB 88] have been examined. To improve the effectiveness of information retrieval systems, the notion of relevance feedback (using relevance assessments provided by the user) and approximate matching algorithms have been proposed [Sal 88, FS 91]. Although the work in information retrieval mainly focuses on (full) text databases, we believe that it is an essential prerequisite of our research.

2.3 The Tasks Involved in Data Mining

For querying and analyzing a database, first a query needs to be specified. In most database applications, query specification is restricted to predefined, possibly parameterized queries which have been designed and implemented by the database designer. If the user wants to issue other queries than the predefined ones, usually an interactive interface for ad hoc queries is available. The queries, however, have to be expressed in specific database query languages such as SQL which represents a quasi-standard. The

SQL query language with its linear syntax was developed two decades ago and has not changed substantially since then. Major problems are that queries need to be issued in a one-by-one fashion providing no possibilities to slightly change a query, to express uncertain or vague queries and to intuitively specify complex queries. As we will show in the next section, visualization technology may help to improve the query specification process considerably.

After issuing the query, the user gets the resulting data set fulfilling the given query conditions. However, as already explained in the introduction, in many cases the result does not provide adequate feedback helping the user to find interesting hypotheses or to refine the query. This is true for very large data sets consisting of multiple attributes with a flat structure, e.g. data generated by automated test series (physics), periodical observations (weather) or statistical recordings (credit card payments). For large structured data sets, it is even more difficult to find interesting hypotheses. In the relational data model, for example, related data is usually stored in multiple relations which need to be joined in order to find relationships between the data. In cases, where relations are joined using (foreign) keys which are specially designed for connecting multiple relations. the join can be executed efficiently providing exactly the desired tuple. In data mining, the goal is to find new relationships and, therefore, (foreign) keys do not provide any help. Often, such relationships are vague and thus, possibilities to express vague or approximate joins are needed. Additionally, it would be helpful, if the results of such joins provide feedback on the closeness in fulfilling the approximate join condition, which is also not provided by current database systems. In the next section, we will present our ideas to use visualization technology to provide better feedback for querying large flat (c.f section 3.2) as well as structured data sets (c.f section 3.3).

If the data is stored in multiple independent databases, the task of finding interesting hypotheses about the data is even harder to solve. In addition to the problem of having different data models, query languages. database management systems and so on, the problems of incompatible structures. incompatible instances, inconsistencies, etc. have to be solved. Again, as we will describe in section 3.4, our visualization technique may be helpful to solve some of the problems in dealing with multiple databases.

At this point, we want to draw the reader's attention to a problem arising in performing data mining on large existing databases, namely the poor support by existing database management systems which are used to store the data. This problem is especially relevant when trying to visualize large amounts of multidimensional data. Current database management systems support neither incrementally changing nor uncertain queries. Furthermore, they do not adequately support range conditions on multiple attributes which require a fast multidimensional search of the database. We believe that existing database systems need to be extended to better support the requirements of data mining and visualization systems. In section 4, we will describe the mentioned problems in more detail and discuss potential solutions.

2.4 Key Characteristics of Data Mining Tools for Large Databases

Before we describe our ideas to support data mining, in this subsection we briefly mention the characteristics of data mining tools that we consider to be the most important ones: interactiveness and efficiency.

For data mining of really large databases to be successful in the near future, we believe that it is essential to make the human being part of the data analysis process. It will be important to combine the best features of humans and computers. The unmatchable intelligence, creativity and perceptual abilities of humans need to be supported by computers, which are best suited to do searching and number crunching. A major research challenge is to find human-oriented forms of representing large amounts of information. In today's systems. the perceptual abilities of humans are only used to a very limited extent. Only a few systems use vision and sound to help the user in data analysis (see [SBG 90] for an example). In the future, data mining tools need to be built in a human-centered way supporting an effective interaction between the user and the system.

A second important characteristic of data mining tools is efficiency. Efficiency is important for the algorithms to scale up well enough when dealing with very large data volumes. Although there is no universally agreed definition of 'efficient', it has been stated that algorithms whose computational requirements are of the same order as sorting [O(n logn)] or better can be considered efficient [FPM 91]. Given that hardware improvements will continue at the same rate as in the past, it is unlikely that algorithms with a complexity that is substantially higher than O(n logn) will be useful in dealing with data volumes in the range of terabytes.

3. Visual Support in Querying Databases

The basic idea of our query and visualization interface is to present as many data items as possible at the same time on the screen with the number of data items being only limited by the number of pixels of the display. Our goal is to visualize the data in a way that the user gets visual feedback on the query and thus can easily explore the database, understand the influence of various query components and find out why slightly different queries have completely different results. In the following, we will give a brief overview of our VisDB system: query specification component (subsection 3.1), visualization of large flat data (subsection 3.2), visualization of complex data (subsection 3.3) and ideas for extending the system to partially solve the problems that arise in dealing with multiple independent databases (subsection 3.4).

3.1 Query Specification

In exploring very large databases, an easy-to-use query interface for 'ad hoc' queries is important. Most of the currently available 'ad hoc' query interfaces, such as SQL, are not user friendly and, in particular, not suitable for data mining. Most of them only provide line-driven, textual interfaces allowing queries to be issued in a one-by-one fashion only. Furthermore, they do not allow interactively modifying queries nor do they adequately support the specification of complex queries. Our idea to solve these problems is to visually support the query specification process allowing even inexperienced users to retrieve data from the database without knowing a specific query language. In contrast to most of today's database management systems where the user is forced to think in terms of the data model and the query language, with our system the user can interactively construct the query and, using the visual representation of the query provided by the system, the user can understand and modify the query more easily.

The main ideas of our visual database query interface have been presented in [KL 92]. Although the visual query interface has been developed in the context of a multimedia database management system, it is generally useful for specifying SQL-like queries. Note, that the query specification interface is largely independent from the rest of the VisDB system. For the purpose of query specification, the user may also use traditional query languages such as SQL or other graphical user interfaces such as QBE [Zlo 77], Visual SQL [TC 90] or Iconic Query [PC 93] instead of our visual query specification interface. However, as we will see in subsection 3.3, our query specification interface is especially useful in conjunction with our technique for visualizing complex query results since it allows direct access to all parts of complex queries.

To briefly explain the query specification process, let us go through an example that will be used in subsection 3.3 to demonstrate the visual feedback in case of complex queries. Assume a user of an environmental database with local weather parameters (temperature, humidity, direction and speed of the wind, solar radiation, precipitation, etc.) and air pollution values (CO, SO_2, NO_2, ozone, etc.) wants to find a correlation between temperature, solar radiation and humidity on one hand and the ozone level on the other hand. According to his/her assumption that there is a correlation between the parameters with a time delay of two hours, the user may specify the following query:

> *'Select the temperature, solar radiation, humidity and ozone level if at the same location the temperature is higher than 15°C or the solar radiation is higher than 600 watt/m^2 or the humidity is lower than 60% and between recording temperature and ozone there is a time difference of two hours.'*

In specifying this query, the user starts by selecting the tables needed for the query from the *Tables* window, namely *Weather* and *Air-Pollution*. As a result, all attributes of the *Weather* and the *Air-Pollution* table are displayed in separate windows. To specify the projection, the user may then move the attributes *Temperature, Solar-Radiation, Humidity* and *Ozone* into the *Result List* window. The next step is to specify the condition. Assume, the user wants to start with the *at-same-location* part. By clicking to *Cond* in the *Tool Box*, s/he gets an empty condition box in the *Query Representation* window and by clicking to the connection *'Air-Pollution at-same-location Weather'* the desired join condition may be specified. 'Connections' are joins which are defined and named by the database designer (or the user) prior to their actual use. Next, the user may want to specify the 'OR'-part of the condition. For each of the three attributes *Temperature, Solar-Radiation* and *Humidity*, the user selects *Cond* from the *Tool Box*, the attribute from the *Weather* table, the desired comparison operator and finally, s/he types in the corresponding value of the limit. Then, all three parts are selected and by clicking to 'OR' in the *Tool Box*, the boxes get 'OR'-connected. The last part that needs to be specified is the time related join condition. Since a join with the intended meaning is already predefined, the join may easily be specified by selecting *Cond* from *Tool Box*, *'Air-Pollution with-time-diff(min) Weather'* from the *Connections* window and typing in the desired time value of 120 minutes. The last step is to combine the conditions into the final result. This is done by selecting all separate parts and by clicking to the logical operator 'AND' from the *Tool Box*. The final result of the query specification is shown in figure 1. The details of the query specification interface are beyond the scope of this paper and are given in [KL 92].

3.2 Visual Feedback in Querying Large Flat Data Sets

As indicated by our definition, we view data mining as an interactive hypotheses-generation process. Our goal is to get the data to ask questions, rather than asking questions to the data. In contrast to most other approaches to data mining (c.f. section 2.2), our idea is to use the phenomenal abilities of the human vision system which is able to analyze compact to midsize amounts of data very efficiently and immediately recognizes patterns in images which would be very difficult (in some cases even impossible) and at least very time-consuming if done by the computer. The research challenge is to find adequate ways of visually presenting multidimensional data to support the user in analyzing and interpreting the data.

Visualization of data which have some inherent two- or three-dimensional semantics has been done even before computers could be used for visualization, and since computers have been used for this purpose, a lot of interesting and efficient visualization techniques have been developed by researchers working in the graphics field. Visualization of large amounts of arbitrary multidimensional data, however, is a relatively new research area. Researchers in the graphics/visualization area are currently exploring techniques in different application domains [Bed 90, FB 90, ID 90, LWW 90, MGTS 90, MZ 92]. In most of the approaches proposed so far, the number of data items that can be visualized on the screen at the same time is quite limited (in the range of 100 to 1,000 data items), but it is a declared goal of the visualization community to push this limit [Tre 92]. In dealing with databases consisting of tens of thousands to millions of data items, our goal is to visualize as many data items as possible at the same time to give the user some kind of feedback on the query. The obvious limit for any kind of visualization is the resolution of current displays which is in the order of one to three million pixels, e.g. in case of our 19 inch displays with a resolution of 1,024 x 1,280 pixels it is about 1.3 million pixels. Our idea is to use each pixel of the screen to give the user a visual feedback on the query allowing him/her to easily focus on the desired data, understand the influence of various query components and find out why slightly different queries have completely different results.

The basic idea of our visualization technique for large flat data sets is described in [KKS 93]. In our approach, as a result for a query the user does not only get the data items fulfilling the query but also a number of data items that approximately fulfill the query. The approximate results are determined using distance functions for each of the selection predicates which are combined into the relevance factor. The distance functions are datatype- and application-dependent and must be provided by the application. Examples of distance functions are numerical difference (for metric types), distance matrices (for ordinal and nominal types), lexicographical, character-wise, substring or phonetic difference (for strings) and so on. Having calculated the distances for each of the selection predicates, the distances are combined into the relevance factor. Important aspects such as normalizing and weighting the different selection predicates, the formulas we use to calculate the relevance factors and the heuristics used to reduce the number of data items that are displayed are described in [KKS 93]. The relevance factors are then sorted resulting in a one-dimensional distribution ranking the approximate responses according to their relevance. The basic idea for visualizing the relevance factors is to map them to colors and represent each data item by several pixels colored according to the relevance of the data item. The colored relevance factors are displayed on the screen with the highest relevance factors (yellow) centered in the middle of the window and the approximate answers with colors ranging from green over blue and red to almost black rectangular spiral-shaped around this region (c.f. figures 2-5). To relate the visualization of the overall result to visualizations of the different selection predicates, we generate a separate window for each selection predicate of the query. In these separate windows, we place the pixels for each data item at the same relative position as the overall result for the data item in the overall result window. The separate windows for each of the selection predicates provide important additional feedback to the user, e.g. on the restrictiveness of each of the selection predicates and also on single exceptional data items. After having the visual feedback, the user may interactively change the query according to the impression from the visualized results. Using highlighting of corresponding pixels in different windows or a projection of the visual representation to specific color ranges, the user may further explore the data helping him/her to relate the relevance factors in the different windows. By being able to get the attribute values corresponding to some specific color, the user may better understand and interpret the visualizations. According to the discoveries made during this process, the user may then incrementally change the query using sliders provided for each of the selection attributes (c.f. figures 2-3).

As already indicated in the previous section, our approach to data mining largely differs from the techniques used in statistics, artificial intelligence, database interfaces and information retrieval. The most obvious difference is that we are using visualization and coloration to support the data mining process. In our approach, we try to adequately support the excellent vision capabilities of humans whom we believe to be the most important factor in data mining. Additionally, our technique is fast enough to be used in very large databases. For simple queries and standard distance functions the complexity is O(n logn) with n being the number of data items. Obviously, query processing time is dominated by the time needed for sorting. Furthermore, our technique is completely application-independent, and, in contrast to most other approaches to data mining, with our approach it is possible to find single exceptional values which are difficult, maybe even impossible, to find with traditional cluster analysis or knowledge discovery methods.

3.3 Visual Support in Querying Large Structured Data Sets

Up to this point, we have only considered the simplest types of queries, namely queries on flat data sets with all selection predicates being connected by the same Boolean operator. In this subsection, we briefly describe how complex queries, i.e. queries with the selection predicates being arbitrarily connected (nested 'AND's and 'OR's), multiple table queries and some types of nested queries may be supported visually in our system.

In dealing with complex conditions that consist of arbitrary Boolean expressions of selection predicates, in the first step the user gets only the visualization of the top level of selection predicates. In terms of the graphical representation of the query in the query representation window, it is the leftmost logical operator with the corresponding selection predicates. If one of the selection predicates itself consists of a Boolean expression, then the user may not understand how the visualization of that part is generated since only one visualization with the overall result for that part is displayed. To be able to explore the impact of any query part, in the VisDB system the user may get visualization and query modification windows for arbitrary subparts at any level of the Boolean expression by simply double clicking the corresponding Boolean operator in the query representation window. The query representation window is available to the user during the whole process of data mining to provide an overview of the actual query, reflecting all changes made by direct modifications and to allow access to all parts of the query. In general, the arrangement of data items in the upper left part of the visualization representing the overall result of the corresponding query part is the same arrangement as for the overall result of the whole query. However, the user may also examine the query part independently and use an option to get the data items arranged according to the relevance factors calculated for the query part only. In our example query (c.f. subsection 3.1), in the first step the visualization consists of four parts: one for the overall result of the query and three for the three parts connected by 'AND' (see figure 2). If the user wants to see visualizations for each of the selection predicates connected by 'OR', s/he might double click on the 'OR'-box in the query representation window and, as a result, s/he will get another query visualization and modification window for this subpart (see figure 3).

Another type of complex queries are multi-table queries which, in general, involve some kind of join. The totality of data items that need to be considered in this case is the cross product of all tables involved. Our idea for visually supporting multi-table queries is to consider all data items of the cross product that approximately fulfill the join condition. As for all other selection predicates, the user gets a separate window with the data items of the cross product that fulfill the join condition being yellow and the others being colored according to their distance. In some cases, e.g. if the tables are connected by foreign keys which are designed to connect related data items. this may not be helpful since the distances on foreign keys may not have any semantics. In such cases, only those data items that fulfill the join condition should be considered and no visualization for the join condition needs to be generated. In most other cases, however, it is quite helpful to consider data items that approximately fulfill join conditions. In our example from environmental science, for example, we have a time- and a location-related join condition both of which may well be considered as vague ones. Such approximative joins may even be crucial to find the desired results if, for example, the time interval for measuring the weather and air pollution parameters is different or if the weather and the air pollution measurement station are not at the same but at close-by locations. In these cases, join conditions requiring time or location equality would provide only very few or even no results though they would be quite helpful. Again, the distance functions used to determine the distance of the join tuples are user and application dependent (c.f. section 3.2). For joins on numerical attributes, for example, the numerical difference between the considered data items from the two relations might be used as an approximation of the join condition to be fulfilled. In a similar way, the distance functions for non-equijoins ($a1 < a2$) or parametrized (non-equi)joins ($a1 - a2 < c$) may be determined. Special joins, e.g. to relate geographical locations (c.f. example query), require more complex distance functions. In a different context, other distance functions may be helpful, e.g. if the user is only interested in one relation and in the number of join partners that each data item of this relation has with another relation, the user might use the inverse of that number as the distance.

In the last part of this subsection, we briefly describe how our visualization technique may support the user in dealing with nested queries. As an example, we describe the case of nested queries where the subquery is connected by using 'exists' or 'in'. In dealing with such types of queries, the user may choose the outer relation(s) to be the basis for displaying the relevance factors of the results. Again, the user will get a separate visualization part for each of the (top level) selection predicates. In the visualization part corresponding to the overall result of the subquery, the user gets yellow in case the subquery condition is fulfilled and otherwise the color corresponding to the distance of the data item most closely fulfilling the subquery condition. The data item most closely fulfilling the subquery condition can be determined by the minimum distance in performing an approximate join of the inner and the outer relation(s). Using this single value to be displayed for the whole subquery, the user gets no feedback on the distribution of distances for the approximative join and on the other selection predicates that may be involved in the subquery. For this reason, we provide the possibility to select one single data item in the visualization window and to get the complete subquery with all its selection predicates including the join of inner and outer relation(s) presented in a separate visualization and modification window. This way, the user is viewing the impact of the subquery in the context of a single data item from the outer relation(s). If the user is more interested in the connections between inner and outer relation(s), s/he might use the cross product of inner and outer relation(s) as a basis for displaying the relevance factors. In this case, the user gets a better feedback on the amount and distribution of distances for data items that only approximately fulfill the join of inner and outer relation(s). However, since we are dealing with the cross product, the totality of data items that are considered is much larger and the percentage that can be displayed is correspondingly lower.

Note, that in most cases where negations are used (negated conditions, NOT IN, NOT EXISTS etc.), no distance values may be obtained and hence no coloring is possible. Exceptions are negated comparison operators [*not (a1 op a2)* with *op* $\in \{>, <, \geq, \leq\}$] where the comparison operator may be inverted. The problem of not having distinguishable values in case of negations is similar to the problem of negations in logic programming.

3.4 Visual Support for Interoperating Multiple Independent Databases

In this section, we will give a brief overview of some ideas to use our visualization technique to perform data mining on data that is stored in multiple independent databases. Many of the problems arising in this context are related to schema enrichment, transformation and resolution as well as query decomposition, translation and optimization. In our papers [KKM 93a-b, KKM 94], we propose algorithms to solve part of these problems for relational and object-oriented databases. Our algorithms and most other algorithms proposed in this context only use techniques that work at the schema or query level but do not consider the data instances. Many problems, however, such as finding corresponding or conflicting data items, can only be solved by comparing the data instances of the different databases and we believe that our visualization technique may be helpful to solve such problems. In addition to using our visualization technique for joining relations from multiple independent databases which can be performed in the same way as multiple table joins within one database (c.f. section 3.3), our technique may be used to identify similar or related data, to discover relationships between tables from multiple databases, to find relations representing the same data items, to find conflicting or incorrect data (typing errors, wrong data entries, lost updates) and so on. Similar patterns in the visualizations, for example, which may easily be identified provide hints for relating sets of data items or even whole relations. Special distance functions may be defined to find related data items in multiple databases (e.g. by mapping the keys of different databases). Approximate joins between relations from different databases may help to find out whether the relations contain similar or related data. Specific differences in the visualizations of two relations may provide hints for conflicting data, and single exceptional data items may help to find incorrect data instances. Up to this point, we did not have a change to extend the VisDB system to be able to evaluate the mentioned ideas but we believe that visualization technology in general and our visualization technique in particular will be of great help in querying, analyzing and comparing large amounts of data as it is required in interoperating multiple independent databases.

4. Extending Existing Database Systems

Our query and visualization interface is designed to work on top of commercially available relational database systems. We are focusing on the relational data model since it is widely used especially to handle large amounts of flat data. A similar query interface, however, may be built to query databases with different data models and query languages. In the following, we describe the problems in interfacing with relational database systems and present ideas on how to realize an on-line interface that is fast enough to directly work on the data stored in the database system.

4.1 Problems in Interfacing with Relational Database Systems

The most important drawback of currently available relational systems is that they do not provide the performance needed to support a query and visualization interface like ours. Most systems are optimized to support high transaction rates but do not adequately support the retrieval and transfer of large continuous ranges of data items. In particular, queries with range conditions on multiple attributes require a fast multidimensional search which is not adequately supported in current systems. Another problem is the poor support for queries that are changed incrementally by the user. In current systems, each query is processed separately and, for a sequence of similar queries having only minor differences, there is no way to incrementally retrieve the changes of the resulting data. For our query and visualization interface, however, an adequate support of incremental queries is crucial to allow interactive querying of the database guided by visual feedback.

One additional problem in accessing the database with complex queries is that we need an independent result for each selection predicate or subquery. This is necessary to display the separate windows corresponding to the different parts of the query. To get the necessary information, we either have to issue one query for each window or get the necessary information automatically while the database system is processing the query. For performance reasons, the first alternative is out of the question since it requires multiple scans of the data; the second alternative is not supported by today's database systems.

4.2 Improving the Performance in Interfacing with the Database System

Since the performance of retrieving data from relational systems is by far too poor to be useful for our system, the current prototype used to evaluate our ideas does not interface directly to a database system. We think that currently the only way to achieve a dynamic visualization of the results that is fast enough for interactively changing queries is to keep the relevant part of the database or at least their distance values in main memory. Keeping data in main memory without re-querying the database means working with a possibly non-actual version of the database. This however implies that after slightly changing the query, the visualization may not be completely correct because some data items in the database might have changed after retrieving the data from the database. For most applications, this is not a problem since we are dealing with very large databases that are changing only gradually and, since we are not presenting the data items themselves but only their relevance factors, minor changes of the data have almost no impact on the visualization.

Another important restriction of keeping the data in main memory is the limitation of the amount of data that can be handled. This is a serious restriction since we are dealing with very large databases with millions of data items. Even for simple datatypes with only 4 bytes per data item about 5 megabytes of main memory are needed to store the about 1.3 million data items that may be displayed on the screen at one moment. If we think of the database as being at least 100 times the amount of data that is retrieved as the result for a query, we would need about 500 megabytes to keep the data in main memory. For datatypes such as strings that require more bytes per data item, there is no possibility to keep the data in main memory. Therefore, for large databases it will be necessary to directly interface to the secondary storage based database system that additionally guarantees consistency, integrity and recoverability in a multi-user environment.

Our idea to make interactive query modification possible, is to retrieve more data than necessary in the beginning and to get only the additional data needed for the modified query later on. It will be important to find adequate heuristics that determine the superset of the data which is retrieved in the beginning. Since a modification of the different selection predicates as well as the weighting factors may cause major changes of the resulting data set, in general it cannot be avoided that additional data must be retrieved from the database. In this case, our idea is to generate 'delta'-queries that only retrieve the additionally needed data which, in most cases, can be done quickly. Additionally, to support a fast (real time) access to the database, data structures that support range queries on multiple attributes may be used. Multidimensional data structures such as R*-Tree [BKSS 90] or Buddy-Tree [SK 90] are important to find the data fulfilling the query criteria without searching the whole database. With these ideas, we support some limited kind of an incremental query processing strategy with one longer processing time in the beginning and shorter processing times later on (when interactiveness is important), hopefully providing access to the database that is fast enough to allow interactive modifications of the queries.

5. Conclusions

Data mining in very large databases is one of the big challenges that researchers in the database area are currently facing. The task is to efficiently find interesting data sets, i.e. hot spots, clusters of similar data or correlations between different parameters. Our approach to support the data mining process combines traditional database querying and information retrieval techniques with new techniques of visualizing the data. Our 'VisDB' system can visually represent the largest amount of data that can be displayed at one point of time on current display technology, providing valuable feedback in querying the database and allowing the user to find results which, otherwise, would remain hidden in the database. The interactivity of the system supports focusing on interesting data providing a promising way to efficiently explore the database. Our approach is independent from any specific application area and requires no knowledge of the application other than the distance and weighting functions. In contrast to traditional cluster analysis or knowledge discovery algorithms, no complete analysis of the data resulting in facts or rules in a high-level language is done by the system. The user with his/her perceptual capabilities and general knowledge is responsible for doing the analysis and interpretation. As a result. the performance of our approach is better than in most other approaches to data mining, making it fast enough to be used for very large amounts of data.

The visualizations presented in figures 2 - 5 are generated by a prototype of our 'VisDB' system. The prototype has been implemented to evaluate the concepts and design of our query and visualization interface. The implementation of some parts of the interface, especially the interactive modification of queries and the visualization in case of nested queries, is not yet completed. Furthermore, we are working on improving the performance in interfacing to traditional database systems. The ideas presented in section 4 are our starting point, but additional ideas will be necessary to allow the VisDB system to work fast enough even for midsize to large amounts of data.

In this paper, we have shown that for exploring large data sets the principle of *incremental query refinement guided by visual feedback* can be very helpful for the user to discover interesting data sets and to derive and verify hypotheses about them. Our VisDB system, being built around this principle, provides a simple and elegant but remarkably powerful way of supporting data mining in very large databases.

References

[ABN 92] Anwar T. M., Beck H. W., Navathe S. B.: '*Knowledge Mining by Imprecise Querying: A Classification-Based Approach*', Proc. 8th Int. Conf. on Data Engineering, Tempe, AZ, 1992, pp. 622-630.

[Bed 90] Beddow J.: '*Shape Coding of Multidimensional Data on a Mircocomputer Display*', Visualization'90, San Francisco, CA. 1990, pp. 238-246.

[BKSS 90] Beckmann N., Kriegel H.-P., Schneider R., Seeger B.: '*The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles*', Proc. ACM SIGMOD Int. Conf. on Management of Data, Atlantic City, NJ, 1990. pp. 322-331.

[Cha 90] Chaudhuri S.: '*Generalization and a Framework for Query Modification*', Proc. 6th Int. Conf. on Data Engineering, Los Angeles, CA, 1990, pp. 138-145.

[DE 82] Dunn G., Everitt B.: '*An Introduction to Mathematical Taxonomy*', Cambridge University Press, Cambridge, MA, 1982.

[FB 90] Feiner S., Beshers C.: '*Visualizing n-Dimensional Virtual Worlds with n-Vision*', Computer Graphics, Vol. 24, No. 2. 1990, pp. 37-38.

[FM 91] Frei H. P., Meienberg S.: *'Evaluating Weighted Search Terms as Boolean Queries'*, Proc. GI/GMD-Workshop. Darmstadt 1991. in: Informatik-Fachberichte, Vol. 289, 1991, pp. 11-22.

[FPM 91] Frawley W. J., Piatetsky-Shapiro G., Matheus C. J.: *'Knowledge Discovery in Databases: An Overview'*, in: Knowledge Discovery in Databases, AAAI Press, Menlo Park, CA, 1991.

[FS 91] Frei H. P., Schäuble P.: *'Determining the Effectiveness of Retrieval Algorithms'*, Information Processing & Management, Vol. 27, No. 2, 1991.

[GPP 90] Geiger D., Paz A., Pearl J.: *'Learning Causal Trees from Dependence Information'*, Proc. 8th National Conf. on Artificial Intelligence, 1990. pp. 771-776.

[GSSK 87] Glymour C., Scheines R., Spirtes P., Kelly K.: *'Discovering Causal Structure'*, Academic Press, San Diego. CA. 1987.

[HD 80] Hall P. A., Dowling G. R.: *'Approximate String Matching'*, Proc. 6th Annual Int. SIGIR Conf., in: SIGIR, Vol. 17, No. 4, 1983. pp. 130.

[HHNT 86] Holland J. H., Holyoak K. J., Nisbett R. E., Thagard P. R.: *'Induction: Processes of Inference, Learning, and Discovery'*, MIT Press. Cambridge, MA, 1986.

[Hub 85] Huber P. J.: *'Projection Pursuit'*, The Annals of Statistics, Vol. 13, No. 2. 1985, pp. 435-474.

[ID 90] Inselberg A., Dimsdale B.: *'Parallel Coordinates: A Tool for Visualizing Multi-Dimensional Geometry'*, Visualization'90. San Francisco, CA. 1990, pp. 361-370.

[ISO 92] ISO/IEC: *'Database Language SQL'*, ISO/IEC 9075:1992 (German Standardization: DIN 66315).

[JKL 77] Joshi A. K., Kaplan S. J., Lee R. M.: *'Approximate Responses from a Data Base Query System: Applications of Inferencing in Natural Language'*, Proc. 5th Int. Joint Conf. on Artificial Intelligence (IJCAI). Boston, MA, 1977, pp. 211-212.

[Kap 82] Kaplan S. J.: *'Cooperative Responses from a Portable Natural Language Query System'*, Artificial Intelligence. Vol. 19, 1982. pp. 165-187.

[KKM 93a] Keim D. A., Kriegel H.-P., Miethsam A.: *'Integration of Relational Databases in a Multidatabase System based on Schema Enrichment'*, Proc. 3rd Int. Workshop on Interoperability in Multidatabase Systems (RIDE-IMS), Vienna, Austria, 1993, pp. 96-104.

[KKM 93b] Keim D. A., Kriegel H.-P., Miethsam A.: *'Object-Oriented Querying of Existing Relational Databases'*, Proc. 4th Int. Conf. on Database and Expert Systems Applications (DEXA), Prague. Czech Republic, 1993, in: Lecture Notes in Computer Science, Vol. 720, Springer, 1993, pp. 325-336.

[KKM 94] Keim D. A., Kriegel H.-P., Miethsam A.: *'Query Translation Supporting the Migration of Legacy Databases into Cooperative Information Systems'*, Proc. Int. Conf. on Cooperative Information Systems. Toronto, Canada, 1994.

[KKS 93] Keim D. A., Kriegel H.-P., Seidl T.: *'Visual Feedback in Querying Large Databases'*, Proc. Visualization'93, San Jose, CA, 1993, pp. 158-165.

[KL 92] Keim D. A., Lum V.: *'Visual Query Specification in a Multimedia Database System'*, Proc. Visualization'92, Boston. MA. 1993, pp. 194-201.

[LWW 90] LeBlanc J., Ward M. O., Wittels N.: *'Exploring N-Dimensional Databases'*, Visualization'90, San Francisco. CA. 1990. pp. 230-239.

[MGTS 90] Mihalisin T., Gawlinski E., Timlin J., Schwendler J.: *'Visualizing Scalar Field on an N-dimensional Lattice'*, Visualization'90, San Francisco, CA, 1990, pp. 255-262.

[Mot 86] Motro A.: *'BAROQUE: A Browser for Relational Databases'*, ACM Trans. on Office Information Systems. Vol. 4. No. 2, 1983, pp. 164-181.

[Mot 90] Motro A.: *'FLEX: A Tolerant and Cooperative User Interface to Databases'*, IEEE Trans. on Knowledge and Data Engineering, Vol. 2, No. 2. 1990, pp. 231-246.

[MZ 92] Marchak F., Zulager D.: *'The Effectiveness of Dynamic Graphics in Revealing Structure in Multivariate Data'*, Behavior. Research Methods, Instruments and Computers, Vol. 24, No. 2, 1992. pp. 253-257.

[NMK 81] Noreault T., McGill M., Koll M. B.: *'A Performance Evaluation of Similarity Measures, Document Term Weighting Schemes and Representations in a Boolean Environment'*, in: Information Retrieval Research. Butterworths, London. 1981.

[PC 93] Parsaye K., Chignell M.: *'Intelligent Database Tools & Applications'*, John Wiley & Sons. New York, 1993.

[Qui 86] Quinlan J. R.: *'Induction of Decision Trees'*, in: Machine Learning, Vol. 1, No. 1, 1986, pp. 81-106.

[RM 86] Rummelhart D. E., McClelland J. L.: *'Parallel Distributed Processing'*, MIT Press, Cambridge. MA. 1986.

[Sal 88] Salton G.: *'A Simple Blueprint for Automatic Boolean Query Processing'*, Information Processing & Management, Vol. 24. No. 3. 1988, pp. 269-280.

[SB 88] Salton G., Buckley C.: *'Term-Weighting Approaches in Automatic Text Retrieval'*, Information Processing & Management. Vol. 24, No. 5. 1988, pp. 513-523.

[SBG 90] Smith S., Bergeron D., Grinstein G.: *'Stereophonic and Surface Sound Generation for Exploratory Data Analysis'*. Proc. Conf. Special Interest Group in Computer and Human Interaction (SIGCHI), 1990. pp. 125-131.

[SK 90] Seeger B., Kriegel H.-P.: *'The Buddy Tree: An Efficient and Robust Access Method for Spatial Databases'*, Proc. 16th Int. Conf. on Very Large Data Bases, Brisbane, Australia, 1990, pp. 590-601.

[SSU 90] Silberschatz A., Stonebraker M., Ullman J. D.: *'Database Systems: Achievements and Opportunities'*, Technical Report. No. TR-90-22, Dept. of Computer Sciences, University of Texas at Austin, 1990.

[TC 90] Trimble J. H., Chappell D.: *'A Visual Introduction to SQL'*, John Wiley & Sons, New York, 1990.

[Tre 92] Treinish L. A., Butler D. M., Senay H., Grinstein G. G., Bryson S. T.: *'Grand Challenge Problems in Visualization Software'*. Proc. Visualization'92, Boston, MA, 1992, pp. 366-371.

[Zlo 77] Zloof M. M. *'Query-By-Example: A Data Base Language'*, IBM Systems Journal, Vol. 4, 1977, pp. 324-343.

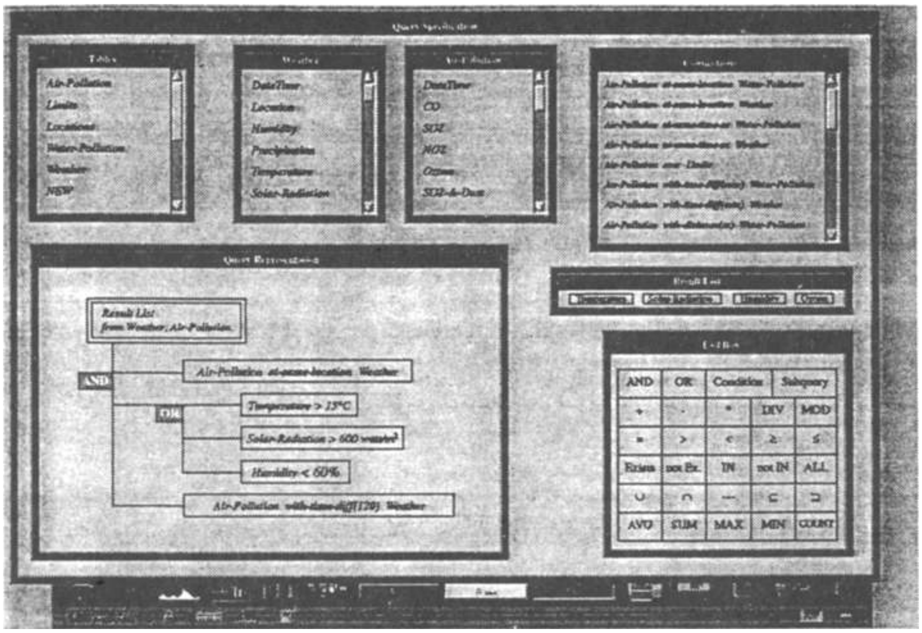
Tables
Air-Pollution
Limits
Locations
Water-Pollution
Weather
NEW
DateTime
Location
Humidity
Precipitation
Temperature
Solar-Radiation
DateTime
CO
SO2
NO2
Ozone
SO2-&-Dust
Air-Pollution at-same-location Water-Pollution
Air-Pollution at-same-location Weather
Air-Pollution at-same-time-as Water-Pollution
Air-Pollution at-same-time-as Weather
Air-Pollution over Limits
Air-Pollution with-time-diff(min) Water-Pollution
Air-Pollution with-time-diff(min) Weather
Air-Pollution with-distance(m) Water-Pollution
Query Representation
Result List from Weather, Air-Pollution
AND
Air-Pollution at-same-location Weather
OR
Temperature > 15°C
Solar-Radiation > 600 watt/m²
Humidity < 60%
Air-Pollution with-time-diff(120) Weather
Result List
Temperature
Solar-Radiation
Humidity
Ozone
AND
OR
Condition
Subquery
+
-
*
DIV
MOD
=
>
<
≥
≤
Exists
not Ex.
IN
not IN
ALL
∪
∩
—
⊆
⊇
AVG
SUM
MAX
MIN
COUNT

Fig. 1. Query Specification Window

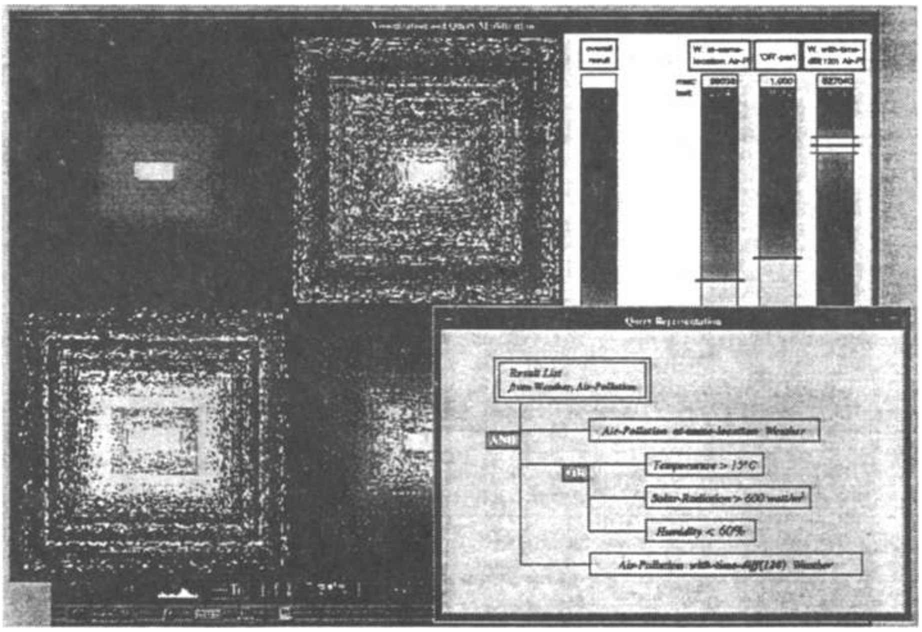
overall result
W. at-same-location Air-P
'OR'-part
W. with-time-diff(120) Air-P
max
last
1.000
Query Representation
Result List from Weather, Air-Pollution
AND
Air-Pollution at-same-location Weather
OR
Temperature > 15°C
Solar-Radiation > 600 watt/m²
Humidity < 60%
Air-Pollution with-time-diff(120) Weather

Fig. 2. Query Visualization and Modification Window

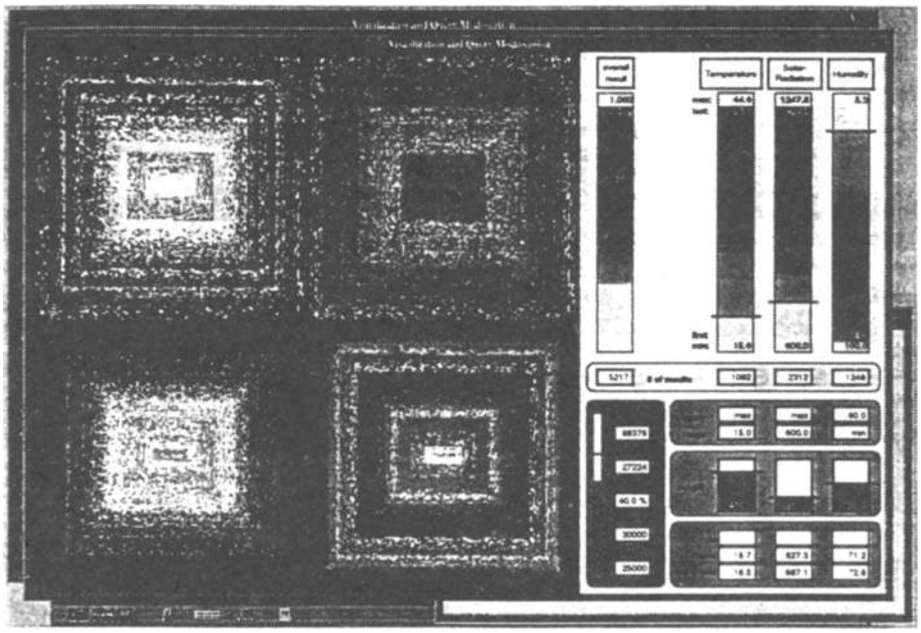

Fig. 3. Visualization of the 'OR'-Part of the Query

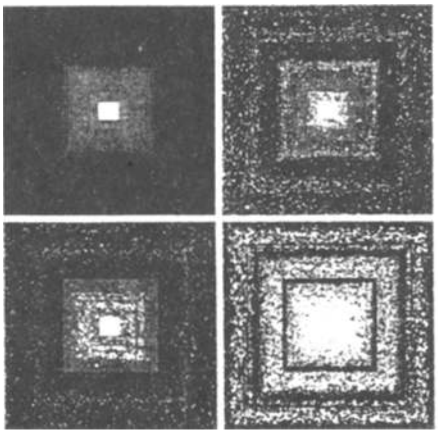

Fig. 4. Query Example 2

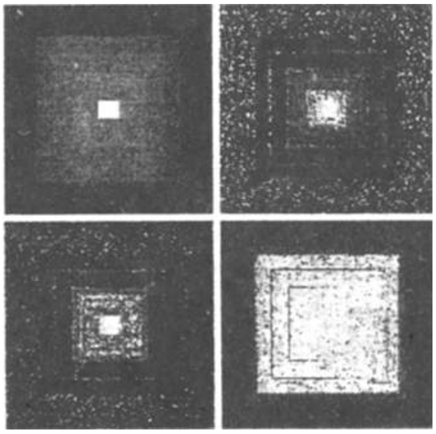

Fig. 5. Query Example 3