First publ. in: Proceedings / Tenth International Conference on Data Engineering, February 14 - 18, 1994, Houston, Tex. pp. 302-313

Supporting Data Mining of Large Databases by Visual Feedback Queries

Daniel A. Keim, Hans-Peter Kriegel, Thomas Seidl
Institute for Computer Science, University of Munich
Leopoldstr. 11B, D-80802 Munich, Germany
{keim, kriegel, seidl}@informatik.uni-muenchen.de

Abstract

In this paper, we describe a query system that provides visual relevance feedback in querying large databases. Our goal is to support the process of data mining by representing as many data items as possible on the display. By arranging and coloring the data items as pixels according to their relevance for the query, the user gets a visual impression of the resulting data set. 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. Furthermore, by using multiple windows for different parts of a complex query, the user gets visual feedback for each part of the query and, therefore, may easier understand the overall result. Our system allows to represent the largest amount of data that can be visualized on current display technology, provides valuable feedback in querying the database, and allows the user to find results which, otherwise, would remain hidden in the database.

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

1. Introduction

The total amount of information in the world is estimated to be doubling every 20 months and the size and number of databases is probably growing even faster. As computers affect more and more aspects of modern society, one by-product is the growing amount of information that is captured in a computer-readable form. The automation of activities in all areas including business, engineering, science, and government produces an ever-increasing stream of data, because every day new applications for computers arise, and even simple transactions, such as paying by credit card or using the telephone, are typically recorded using computers. Automated test series in physics, chemistry and medicine generate large amounts of data that are collected automatically via sensors and monitoring systems. Even larger amounts of data are collected by satellite observation systems which are expected to generate one terabyte of data every day in the near future [FPM 91].

The data of all areas mentioned so far are collected because people believe that it is a potential source of valuable information providing a competitive advantage (at some point). Querying and analyzing the data to uncover the valuable information hidden in the databases, however, is a difficult task. The growth in the size and number of existing databases far exceeds human abilities to analyze the data. If all the data are to be analyzed at all, computer supported data analysis will have to play an important role. Today, already most of the data is stored in computers and an increasing amount of the data is managed by database management systems. Their query languages allow people to query the databases, but finding the interesting data often remains a problem. Even experienced database users may have difficulties to find the hot spots. Since the user does not know exactly the data and its distribution, many queries may be needed to find them. The result for most queries will contain either less data than expected, sometimes even no answers, so-called 'NULL' results, or more data than expected, at least more than the user is willing to deal with. Thus, the data which was once collected because it might be useful now may sit useless in 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 [SFD 93] 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. On the one hand, over the last 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 is to be analyzed and the amount of data that can be analyzed is growing.

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, adequate and efficient mining tools are essential for the mining to be successful. In some sense, data mining is like the work of radiologists. It is like scanning the database to identify phenomenons that need to be looked at, showing the regular structure of the data, but also helping to find anomalies.

2.1 Definition of Data Mining

'Data Mining' can be defined as the (non-trivial) process of searching and analyzing data, helping to find implicit but potentially useful information. Let $D = \{d_1, ..., d_n\}$ be the data set to be analyzed. Then, 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. This 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 user's mind 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 quite a 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 allow only special kinds of hypotheses, namely those with D' = D or D' = R_i in the case where D is partitioned into relations $R_1, ..., R_m$ ($D = \bigcup_{i=1}^{m} R_i$). 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 the relationships between all combinations of dimensions for all subsets of data items which is computationally intractable for large data sets. Although most cluster analysis algorithms use some kind of heuristics to reduce the search space (e.g. [Hub 85]), for very large databases with millions of data items, cluster analysis is not feasible without human guidance. 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 the related fields of knowledge discovery and machine learning which can also be considered as data mining. Among the AI techniques used in data mining are decision tree approaches, inverted expert system approaches, probabilistic theories, Bayesian statistics, neural networks and genetic algorithms. In contrast to our work, in knowledge discovery the hypotheses are usually rules or facts which are formally specified in some high-level language [FPM 91].

Another area related to data mining are 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. The usefulness of the results, however, largely depends on the user's a priori knowledge and intuition. Additionally, today's query interfaces only allow queries to be issued in a one-by-one fashion providing no possibilities to incrementally change a query, to express uncertain and vague queries. Approaches to improve the query interface include graphical database interfaces that allow the user to browse the data (e.g. FLEX [Mot 90]) and cooperative database interfaces [Kap 82, ABN 92] that try to give 'approximate answers' in cases where the query does not provide a satisfactory answer (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. In this context, distance functions for text, strings or descriptors [HD 80], ranking functions [NMK 81] 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]. 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 Key Characteristics of Data Mining Tools

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 very 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 intelligence, creativity and perceptual abilities of humans which are unmatchable 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 sys-

tems, the perceptual abilities of humans are only used to a very limited ext.5end. Only few systems use vision and sound to help the user in data analysis (see [SBG 90] for an example).

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 hardware improvements 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.

2.4 Outline of the paper

The rest of the paper is organized as follows: Section 3 presents our idea to provide visual feedback in querying large databases. Section 4 illustrates the query specification and visualization interface and introduces the notion of approximate joins. Furthermore, a larger example is presented illustrating the handling of complex queries. To be able for the reader to mathematically understand how our visualizations are created, in section 5 we briefly describe distance functions, calculation of the relevance factors and the heuristics used in the system. Section 6 summarizes our approach and points out some of the open problems for future work.

3. The Basic Idea

As indicated by our definition, we view data mining as an interactive hypotheses generation process. Our goal is to challenge 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 mid-size amounts of data very efficiently. It is able to immediately recognize 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 users in analyzing and interpreting the data.

Visualization of data which has some inherent two- or three-dimensional semantics has been done even longer than computers exist. Since using computers for this purpose, a lot of interesting and efficient visualization techniques have been developed by researchers working in the graphics field [EW 92]. Visualization of large amounts of arbitrary multidimensional data, however, is a pretty new research area. Researchers in the graphics/visualization area are currently exploring techniques in different application domains [FB 90, ID 90, LWW 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 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 the best possible 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 - or more general, to support a better, easier and faster query specification. The interactiveness of such a system is important. The user should have the possibility to modify the query on-line and to see the changes of the visualized data set immediately. By exploring the data with such a system, the user may learn more about the data than by issuing hundreds of queries.

Now, let us explain our ideas using a real world example. In environmental science, researchers want to find correlations between local weather parameters such as temperature, humidity, direction and speed of the wind, solar radiation, precipitation and the air pollution by CO, SO_2, NO_2, ozone, etc. They have large series of the weather and pollution parameters being measured every hour or even every few minutes at multiple locations resulting in data volumes of about 10 MByte even if only measured for one year at one location. Some obvious correlations between parameters, e.g. a positive correlation of temperature and solar radiation, can be easily found by calculating average or sum values for specific periods of time as well as their correlations. Other interesting aspects, however, such as a time-lagged increase of temperature and ozone, or single exceptional values are difficult to find with traditional analysis methods. Finding such interesting data by directly querying the database is also very difficult, since in general, none of the parameters for the query can be fixed in advance.

Using our query and visualization system, the user still has to specify a query using a graphical query specification tool. As a result of the 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 for distance functions are the numerical difference (for metric types), distance matrices (for ordinal and nominal types), lexicographical, character-wise, substring or phonetic difference (for strings) and so on. In case of our environmental database, we simply use numerical differences.

Having calculated the distances for each of the selection predicates, the distances are normalized and weighted be-

fore they are combined into the relevance factor. Relevance factors may be calculated for all data items, but in general, we do not want (or may not be able) to present all the data on the screen. Therefore, a threshold restricting the number of data items that are represented on the screen needs to be determined. This can be done by simply presenting as many data items as fit on the screen, by presenting a user given percentage of the data or by more intelligent reduction algorithms (c.f. section 5.1). Then, the relevance factors are sorted resulting in a one-dimensional distribution, ranking the approximate responses according to their relevance. The principle idea for visualizing the relevance factors is to map them to colors and represent each data item by several pixels being colored according to the relevance of the data item. Since screens are typically two-dimensional displays, we had to find an adequate way of arranging the colored relevance factors. We tried several arrangements such as top-down, left-to-right, centered, etc. and found that arrangements with the highest relevance factors centered in the middle of the window seem to be the most natural. The absolutely correct answers are colored yellow in the middle and the approximate answers with colors ranging from green over blue and red to almost black are rectangular spiral-shaped around this region (c.f. figure 1a). In cases where distance functions provide positive and negative distances, we also allow different arrangements (c.f. figure 1b).

The resulting window for the overall result is always similar to the upper left part of the visualization window presented in figures 4 and 5. It only provides feedback on the amount of data fulfilling the query and on the distribution of the approximate answers. This information may already be very helpful for the user; however, to better support the data mining process, it is necessary to relate the visualization of the overall result to visualizations of the different selection predicates. Therefore, we generate a separate window for each selection predicate of the query. In these 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 high-lighting 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 having the possibility 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.

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 which 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.

4. The Query Specification and Visualization Interface

The basic idea of our query and visualization interface is to present as many data items as possible to the user to provide visual feedback on the query and to allow easy exploration of the database, to understand the influence of various query components, and to find out why slightly different queries have completely different results. In the following, we give a brief overview of our VisDB system: query specification and visualization components, facilities provided to modify queries interactively, processing of complex queries and examples for the visualization of results for different queries.

4.1 Query Specification

The query specification interface we use for specifying queries is a derivative of the GRAphical Database Interface (GRADI) [KL 92]. Although GRADI was developed in the context of a multimedia database management system, it is generally useful for specifying SQL-like queries. GRADI has the advantage of allowing direct access to all parts of complex queries. For the purpose of query specification, the user may also use traditional query languages such as SQL, or other graphical user interfaces. Since the query specification is largely independent from the rest of the VisDB system, we describe it only briefly in the following.

When starting the VisDB system, first the user has to select the database s/he wants to work with. After getting the Query Specification window, the next step is to select the tables to be used in the query. For each selected table a list with all attributes will be displayed in a separate window and all ‘connections’ involving at least one of the selected tables will appear in the *Connections* window. ‘Connections’ are joins which are defined and named by the database designer (or the user) prior to their actual use. It may have parameters. To specify the result list (projection), the user has to move the desired

attributes and operators (avg, sum, max, min, count) to the *Result List*. Now, only the condition part of the query remains to be specified. Using connections, attributes of the selected tables, and operators provided by the *Tool Box*, the query may be built interactively using the mouse. To support an incremental query specification process, we allow the user to specify all parts of the query independently and to combine them at a later stage. In the *Query Representation* window the query is displayed graphically. Each part of the query is represented by a small box, simple conditions by a single, subqueries by a double box, and the connecting lines are labeled with the type of connection used. The *Tool Box* allows fast access to all functions supported by the system. The functions are divided into six groups: logical operators and basic elements, arithmetic operators, comparison operators, nesting operators, set operators, and aggregate operators. The basic elements *Condition* and *Subquery* are necessary for the incremental query specification process. To allow the user to express the relative importance of each of the selection predicates, weighting factors may be defined by selecting condition or subquery boxes and assigning weighting factors to them.

To further explain the query specification process, let us go through an example. Assume, a user of the environmental database (c.f. section 3) 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 2 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.'*

The final result of the query specification for this query is shown in figure 3. The details of the query specification process are beyond the scope of this paper and are given in [KL 92].

4.2 Visual Feedback

The principle idea of the visualization of results has already been described in section 3. As a result for a query, not only the absolutely correct answers are retrieved, but also approximate ones. These are determined by calculating a distance for each of the selection predicates and combining them into the relevance factor. Instead of displaying the data itself, we represent each data item by one, four or sixteen pixels of which the color represents the relevance of the data item. Mapping the relevance factors to colors corresponds to the task of finding an adequate color scale for a single parameter distribution. The advantage of color over gray scales is that the number of just noticeable differences (JNDs) is much higher [LRR 92]. The main task in coloring the relevance factors is to find a path through color space that maximizes the number of JNDs, but, at the same time, is intuitive for the application domain. In designing the system, we tried many variations of the colormap to enhance the usefulness of our system and found experimentally that for our application, a colormap with quite constant saturation, an increasing luminosity (intensity) and a hue (color) ranging from yellow over green, blue and red to almost black is a good choice to depict the distance from the correct answers.

To get a useful visual representation, the relevance factors are sorted in descending order and arranged in a window, the highest relevance factors centered in the middle and the approximate ones in a rectangular spiral-shape around this region (see figure 1a). The sorting is necessary to avoid completely sprinkled images that would not help the user in understanding the data. Overall result windows of different queries only differ in the size of the areas with different colors. They may even be completely yellow in cases where all the data represent completely correct results or almost black in cases where all the data are completely wrong results. To give the user more feedback than the amount of correct answers and the distribution of approximate answers, we additionally provide visualizations of the distances for each of the (top level) selection predicates (c.f. figure 4). However, in contrast to the overall result window, we do not sort the distances, but keep the same ordering of data items as in the overall result window. As a result, the pixels of the windows are implicitly related by their position. To be more specific, for every data item the colors representing the distances for the different selection predicates are at the same relative position in each of the windows. The separate windows for the different selection predicates provide important additional information to the user. By the visual color impression of the single screens, the user gets information on how restrictive each of the selection predicates is, i.e. how many data items fulfill a condition, how many fail to fulfill a condition, and how close the data items are to fulfill each of the conditions. Using the correspondence of pixels between the separate windows denoted by their position, the user may also study specific data items. If in one of the windows there is a color spot in an area of different color, the user might check for this specific data item in the other windows, or s/he might even retrieve the values for the corresponding data item out of the database.

In designing our system, we also experimented with other arrangements of the data items on the screen. One idea was to display the data in 2D or 3D with selected attributes assigned to the axis. However, with such kinds of arrangements, we had the problem that on the one hand many data items may be concentrated in some area of the screen while other areas are virtually empty, and on the other hand many data items are superposed and therefore not visible. Although 2D or 3D visualizations may be very helpful, e.g. in cases where the data have some inherent two- or three-dimensional semantics, we did not pursue this idea for several reasons: One reason is that in most cases the number of data items that can be represented on the screen at the same time

is quite limited. This was in contrast to our goal of providing feedback for as many data items as possible on the screen. A second reason is that in most cases where a 2D or 3D arrangement of the data is straightforward, systems using such arrangements have already been built. For spatial queries on two-dimensional data, for example, a 2D visualization is obviously the best support for querying the database, and basically all Geographical Information Systems provide such visual representations of the data. For all cases, however, where no inherent two- or three-dimensional semantics of the data and therefore no straightforward visualization exists, our representation can be of great value to provide visual feedback in querying the database. Stimulated by 2D or 3D representations of the data, we got the idea to provide an optional second method of visualization that includes some feedback on the direction of the distance for distance functions that provide positive and negative values. The basic idea is to assign two attributes to the axis and to arrange the relevance factors according to the direction of the distance; for one attribute negative distances are arranged to the left, positive ones to the right and for the other attribute negative distances are arranged to the bottom, positive ones to the top. Inside the regions, the data items with the relevance factors sorted in an descending order are arranged from the middle (yellow region) to the edges of the window (see figure 1b).

With this kind of representation, we do not represent the distance of data items directly by its locations, but we denote the absolute value of the distance by its color and the direction by its location relative to the correct answers (colored yellow). An advantage is that each data item may be assigned to one pixel and no overlays of data items with the same distances occur. In summary, it may be noted that maximizing the number of visualized data items conflicts with arrangements that have multiple attributes assigned to the axis.

4.3 Interactively Modifying the Query

When using our system, the possibility to dynamically modify queries is important. Since modifications have a direct impact on the visualization, the user will get an immediate feedback on the effects of the changes. The visualization provides feedback on the amount of data retrieved, on the restrictiveness of the conditions, on the distribution of the distances for each selection predicate and on special data sets the user might be interested in. For example, if the yellow region in the middle of each window is getting larger (shrinking), more (less) data items fulfill the condition; if a window is getting darker (brighter), the corresponding selection predicate is getting more (less) restrictive; if the overall structure of a window is changing, the distribution of distances for the corresponding selection predicate is changing and so on. These visual indicators are a valuable help to quickly understand the effects of query modifications and to learn more about the data in the database, especially in exploring large databases with millions of data items.

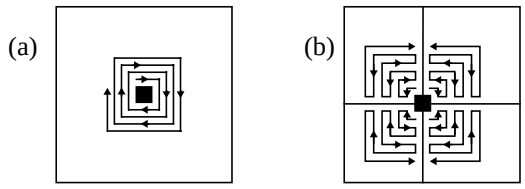

Figure 1: (a) normal arrangement, (b) 2D-arrangement

In figures 4 and 5, the query visualization and modification window of the 'VisDB' system is displayed. The window is divided into the left portion, the 'Visualization' part and the right portion, the 'Query Modification' part. In the 'Visualization' part, the user receives a visual representation for the overall result and for each selection predicate. The 'Query Modification' part consists of sliders for the selection predicates and weighting factors as well as some other options. The color spectrum of each slider is just a different arrangement of the colored distances and corresponds to the distribution of distances for the corresponding attribute. Inside each slider, the lowest and highest value of the visualized data items for the corresponding selection predicate are displayed. Outside the color spectrums the minimum and maximum value of the attribute in the database are displayed to give the user a feeling for useful query values or query ranges.

Below the sliders, several parameters are listed for each attribute, namely the 'number of results', the attribute values of a 'selected tuple', the attribute values corresponding to some selected color range ('first' and 'last of color') and finally the 'query range' and the 'weighting factors'. In the following, we will describe how these parameters may help the user to explore the database and to modify the query. Using the mouse, the user may choose a specific color or color range in any of the sliders to get the corresponding values of the attribute in the 'first' and 'last of color' fields. The possibility to get the specific values corresponding to the different colors for each selection predicate makes it easier for the user to understand and interpret the visualization. In figure 5, for example, the red region in the lower right window may be easily identified by the user. To understand the meaning of this region, however, the user needs additional information relating colors and attribute values. In this special case, the user easily observes that data items with values in the range of about 71% - 73% for Humidity are quite good overall answers although their values for Humidity are quite distant as indicated by the red color. An additional help for the user to understand the visualization and to find hot spots is to select a specific data item in one of the visualizations to get the data item highlighted in all visualization parts and the values for the attributes displayed in the 'selected tuple' field. The user may use this option to focus on an exceptional data item or to get an example for a data item from an interesting region in one of the visualization parts. To focus on sets of data items with a specific color, it is possible to select some color range in one of the sliders to get only those data items displayed that

have the selected color for the considered attribute. In the other visualizations the same data items are displayed allowing the user to easily compare the values of the other attributes.

To allow an interactive query modification, the query parameters are represented graphically by the black lines in the sliders and by the value of the upper and lower limit in the 'query range' field. The user may use the sliders to roughly modify lower and upper limit of the query or s/he may directly change the values in the 'query' field (c.f. figure 5). For numbers, the user may also choose a different type of slider where the medium value and some allowed deviation can be manipulated graphically (see rightmost slider in figure 4). Different types of sliders are provided for different datatypes and different distance functions. Sliders for discrete types, for example, reflect the discrete nature of the data by allowing only discrete movements of the slider. Sliders for non-metric types (ordinal and nominal datatypes) may be, for example, enumerations of the possible values with the possibility to select each of the values. Special sliders may be designed for special datatypes and special distance functions, e.g. for strings with different distance functions (c.f section 3). Below the query parameter field, the weighting factors are represented graphically. Like the query parameters, the weighting factors may be directly manipulated using the mouse.

On the left side of the query modification part, there is a color spectrum for the overall result. Since the combined distance values have no inherent meaning, no values are assigned to the different colors. Instead of fields for a selected tuple, selected colors, or the query values, the number of data items in the database, the number of data items being displayed in the visualization (absolute value and percentage), and the number of resulting data items are presented to the user. Using a slider, the user may change the percentage of data being displayed or the allowed range, in case the percentage is determined using the heuristics described in section 5.1 (see figure 5). Note that changing the percentage of data being displayed may completely change the visualization since the distance values are normalized according to the new range.

In the normal mode, the system recalculates the visualization after each modification of the query. The user may also switch to an 'auto recalculate off' mode where queries are only recalculated on demand. This option is useful for large databases or if complex distance functions are used, because the recalculation for each modification may need a considerable amount of time. Another menu option provides the possibility to switch back to the query specification process, thereby allowing structural changes or extension of the query, and to specify completely different queries.

4.4 Complex Queries & Approximative Joins

Up to this point, we have only considered the simplest types of queries, namely one table queries with all selection predicates being connected by the same boolean operator. In this subsection, we will 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 in our system.

In dealing with complex conditions that consist of arbitrary boolean combinations of selection predicates, in the first step the user gets only the visualization of the top level of the boolean expression. 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 the part is displayed. To be able to explore the impact of any query part, in the VisDB system the user has the possibility to get a visualization and query modification window for arbitrary subparts at any level of the boolean expression by simply double clicking to 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 4.1), 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 4). 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 5).

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 corresponds to the cross product of all tables involved. Our idea to visually support 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 for the join condition with all 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 many 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 which both may well be considered as vague ones. Such approximative joins may even be crucial to find the desired results if e.g. 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). 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 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 only 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.

4.5 Examples and Applications

In this subsection, we discuss some example visualizations and applications. The visualizations presented in figures 4 and 5 have been produced by our prototype system using real world data taken from a large database of geographical information.

The starting point of our description is the query example presented in figure 3. In the visualization part of the query modification and visualization window (c.f. figure 4), it can be easily seen by the different colors dominating the visualizations that each of the selection predicates has a distinct impact on the overall result. The visualization of the third selection predicate displayed in the lower right part of the visualization window, for example, is dominated by dark colors which means that the selection predicate is quite restrictive. The visualizations of selection predicate one and two are much brighter. In case of the lower left window, this is clear because it corresponds to the overall result of the 'OR'-part and all data items fulfilling one of the three selection predicates are colored yellow. Note, that the corresponding window (lower left of figure 4) is identical with the upper left window of figure 5. Figure 5 visualizes each of the three or-connected selection predicates with the arrangement of data items being the same as in figure 4.

Our query and visualization system is not only useful for data mining tasks such as finding correlations between different attributes, finding groups of similar data, and finding hot spots, but also for other tasks such as similarity retrieval, finding adequate query parameters and weighting factors, and finding correspondences in different databases. Finding similar parts in a large CAD database is an example for the first two of these tasks. In a CAD database of 3D-parts, it is not obvious how similarity can be formally described. Usually, there are quite many parameters (in a concrete application in mechanical engineering we had 27 parameters) describing the parts, and each of them might be important for a part to be similar. In searching for similar parts in traditional CAD databases a query is issued using fixed allowances for some of the parameters.

As a result of the query, the user only gets the information whether a data item fulfills all allowances or not. However, the user might miss a part that exactly fits in all except one parameter and just misses to fulfill the allowance of that single parameter. Therefore, in similarity retrieval, it seems to be important to provide approximate responses and to allow the user to adjust the allowances and weighting parameters. Our system provides features that exactly support these tasks making it a promising candidate to be used in similarity retrieval. Another example for an interesting application of our system are multi-database systems where it is often a problem to find corresponding data items in multiple independent databases. If a distance function for the two attributes to be joined can be defined, our system will help the user to identify closely related data items of the two databases and to find adequate parameters for approximately joining the databases.

5. Mathematical and Statistical Foundations

In this section, we briefly describe the mathematical and statistical foundations of our visualization technique. As we will see, the formulas used in calculating the relevance factors are of high importance for the visualizations to be useful. Some issues such as the distance and weighting functions are highly application dependent and the examples presented in this paper are given such that the reader is able to understand how we derived the images presented in figures 4 and 5.

5.1 Heuristics to Reduce the Amount of Data to be Displayed

Since the number of data items that can be displayed on the screen is limited by the number of pixels, we had to find adequate heuristics to reduce the amount of data and to determine the data items of which the distance shall be displayed. The exact way is to use a statistical parameter, namely the α-quantile. The α-quantile is defined as the lowest value ξ_α with $F(\xi_\alpha) = \int_{-\infty}^{\xi_\alpha} f(x)\,dx = \alpha$, where $0 \le \alpha \le 1$, *F(x)* is the distribution and *f(x)* the density function.

Let r be the number of distance values that can be displayed on the screen, #sp be the number of selection predicates and n the number of data items in the database, then only data items with an absolute distance in the range [0, p-quantile] are chosen to be presented to the user where p equals r/(n*(#sp+1)). If negative and positive distance values are used, the range of values presented to the user is given by [α_0*(1-p)-quantile, (α_0*(1-p) +p)-quantile] where α_0 is determined by α_0-quantile = 0. In the special case of two attributes assigned to the two axis (c.f. section 4.2), correspondingly the combined α-quantiles for two dimensions may be used. In the case, when several pixels are used per data item, the number of presentable data items needs to be divided by the corresponding factor (4 or 16) and the quantiles need to be adapted correspondingly.

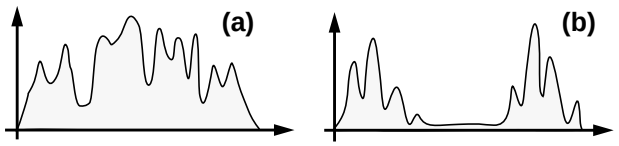

Figure 2: two density functions

More important than the number of data items that are displayed is the effect of the highest and lowest value of the considered part of the data on the normalization. The α-quantiles are only the best choice if we want to present as many data items as possible. Depending on the distribution of values, in many cases it will be better to present less data items, especially if the density function of the distance values has multiple peaks (see figure 2 for an example for two density functions). If we have e.g. two groups of distance values each being in different orders (see figure 2b), it may be helpful to present only the values of the lower group to the user since, in this case, the graduate differences within this group are better enhanced by different colors. To implement this heuristics, first we define the range [r_{min}, r_{max}] for the number r of distance values that the user would like to get displayed. Suppose, the data items x_i are sorted according to their distance d_i. Then, for each $x_i \in \{x_{r_{min}}, \ldots, x_{r_{max}}\}$ we calculate $s_i = \sum_{j=i-z}^{i+z} |d_i - d_j|$, with z being a heuristically determined data dependent constant in the range $2 < z \ll r_{max} - r_{min}$. Then, we choose the data item with the highest s_i to be the last data item that is displayed. Although at the first glance, the complexity of this heuristics seems to be in the order of $z*(r_{max}- r_{min})$, the algorithm can be easily optimized to a complexity in the order of $(z+r_{max}- r_{min})$ by successively calculating the s_i.

5.2 Combining the Distances into the Relevance

To determine the combined distance for a complex query, first the distance values are calculated for each selection predicate of the query by the application dependent distance functions as described in section 3. The combination of distances for the different selection predicates, however, is not straightforward because the distances for the different selection predicates have to be considered with respect to the distances of the other selection predicates, and the combined distance must be defined and meaningful globally. One problem is that the values calculated by the distance functions may be in completely different orders of magnitude (e.g. in a medical application, a distance of 1g/dl for Haemoglobin may be very large and a distance of 1,000 per dl for Erythrocyte may be very small). A second problem is that the relative importance of the multiple selection predicates is highly user and query dependent.

The second problem can only be solved by user interaction since only the user is able to determine the priority of the se-

lection predicates. Therefore, in general, it is necessary to obtain weighting factors (w_j , $j \in 1, \ldots, \#sp$) representing the order of importance of the selection predicates from the user.

The first problem can be solved by a normalization of the distances. A simple normalization may be defined as a linear transformation of the range $[d_{min}, d_{max}]$ for each selection predicate to a fixed range (e.g. [0, 255]). When experimenting with this normalization, we found that in some cases it may cause misleading results. A single data item, for example, with an exceptionally high or low value may cause a completely different transformation, even if the combined distance of this data item is too high to be displayed. As a consequence of the normalization, however, the corresponding selection predicate may have little or no impact on the overall answer resulting in a set of approximate answers with a completely misleading visualization. Our idea to improve the normalization is first to reduce the number of data items to be displayed for each selection attribute to a number that is proportional to $\frac{r}{n \times w_j}$. The inverse proportionality to w_j ($w_j \in [0, 1]$) is important for the following reason: The less a selection predicate is weighted, the higher is the probability that data with a greater distance for this selection predicate are needed. Then, the data are normalized by transforming the range $[d_{min}, d_{max}]$ of the remaining data items to a fixed range as described above.

In order to combine the independently calculated and normalized distances of multiple selection predicates into a single distance value, we successively calculate combined distances for all subparts of the query, according to the structure of the query. In this step, we use e.g. the weighted arithmetic mean for 'AND'-connected condition parts and the weighted geometric mean for 'OR'-connected condition parts. More exactly, for each data item x_i the combined distance is calculated as:

$$\text{Combined Distance}_i = \sum_{j=1}^{\#sp} w_j \times d_{ij} \quad \text{in case of 'AND',}$$

$$\text{Combined Distance}_i = \prod_{j=1}^{\#sp} d_{ij}^{w_j} \quad \text{in case of 'OR'.}$$

Before a calculated combined distance is used as a parameter for combining other distances, it is also normalized as described above. After calculating the combined distance for the whole condition, the relevance factor is determined as the inverse of that distance value. At this point it should be mentioned, that for special applications other specific distance functions such as the Euclidean, L^p or the Mahalanobis distance in n-dimensional space may be used to combine the values of multiple attributes.

6. 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 allocate 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 allows to visualize 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 allows to focus on interesting data providing a promising way to explore the database efficiently. Our approach is independent from any specific application area and requires no knowledge on 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 query performance 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 4 and 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 screen layouts with two attributes assigned to the axis, is not yet completed. Furthermore, in interfacing to traditional database systems, we found that tasks such as multidimensional search and incremental changes of queries which are important for our system to work fast enough, are not adequately supported. Additionally, current systems do not provide access to the preliminary results of query subportions. We are currently working on techniques that allow our system to work fast enough despite these problems. Our idea is to retrieve more data than necessary in the beginning and to retrieve only the additional portion of the data that is needed for a slightly modified query later on. Additionally, multidimensional data structures that support range queries on multiple attributes will be essential to improve query performance.

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.

Note

For technical reasons, it was not possible to publish the screen dumps in color in this proceedings. If you are interested, we would gladly forward the color pages to you.

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.

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

[EW 92] Earnshow R. A., Wiseman N.: *'An Introduction Guide to Scientific Visualization'*, Springer, 1992.

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

[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, 1991.

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

[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.

[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.

[KL 92] Keim D. A., Lum V.: *'GRADI: A Graphical Database Interface for a Multimedia DBMS'*, Proc. Int. Workshop on Interfaces to Database Systems, Glasgow, England, 1992, in: Workshops in Computing, Springer.

[LRR 92]Levkowitz H., Robertson P., Rogowitz B.: *'Color Theory and Models for Computer Graphics and Visualization'*, Tutorial No. 5, Visualization'92, Boston, MA., 1992.

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

[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.

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

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

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

[SFD 93] Stonebraker M., Frew J., and Dozier J.: *'The Sequoia 2000 Architecture and Implementation Strategy'*, Sequoia 2000 Technical Report 93/23, University of California, Berkeley, CA., 1993.

[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.

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

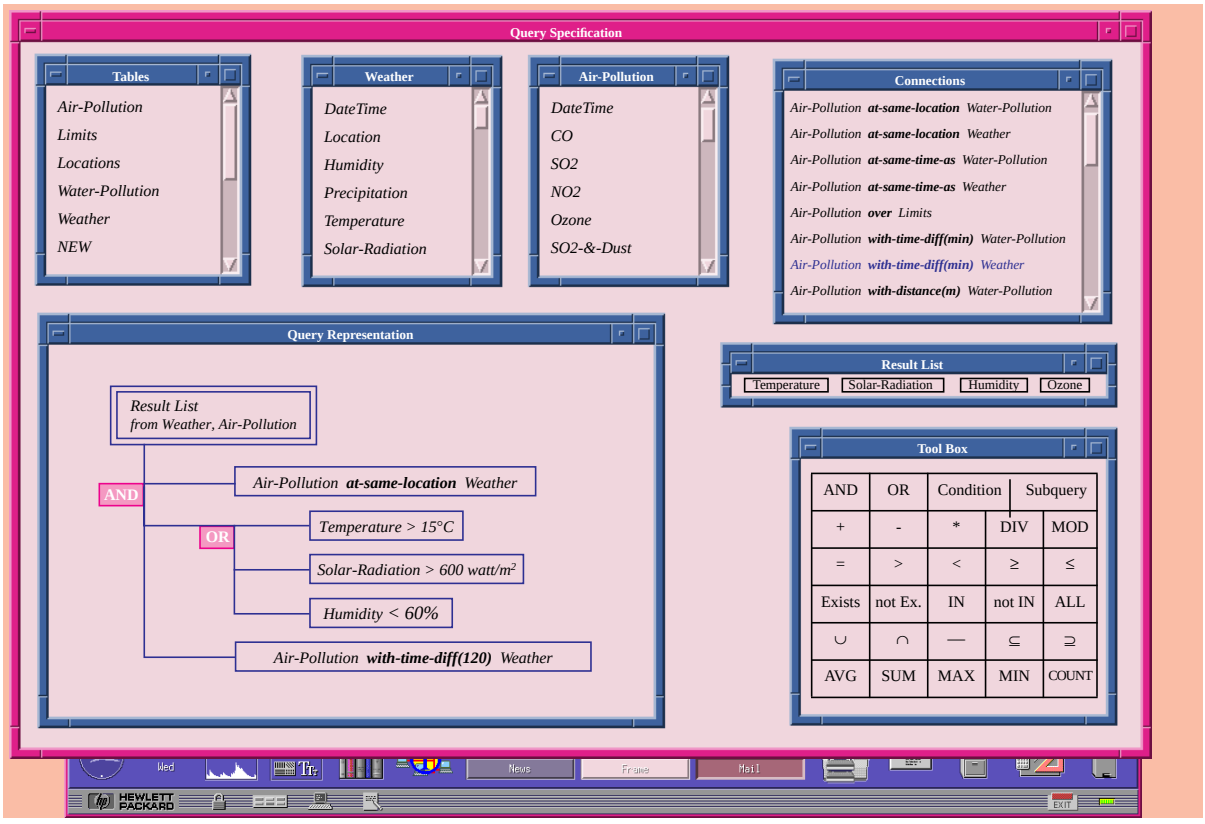

Figure 3: Query Specification Window

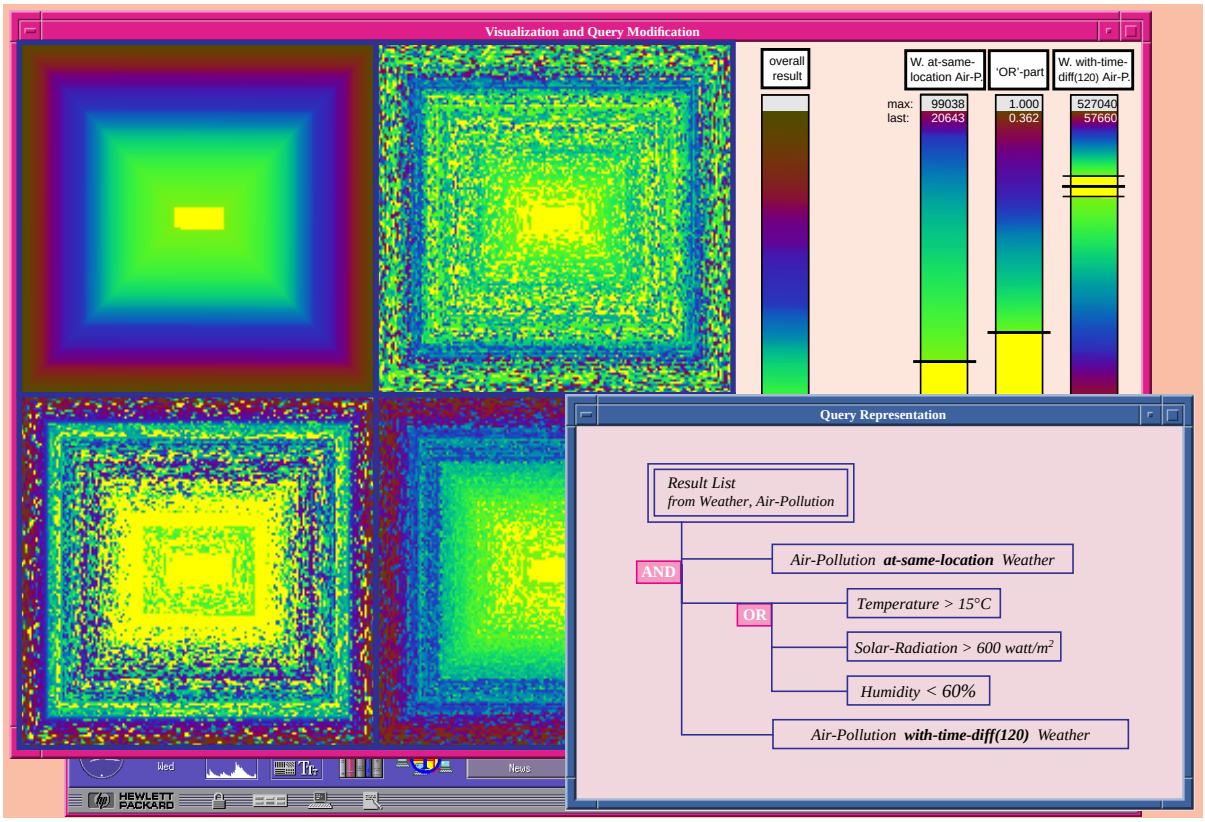

Figure 4: Query Visualization and Modification Window

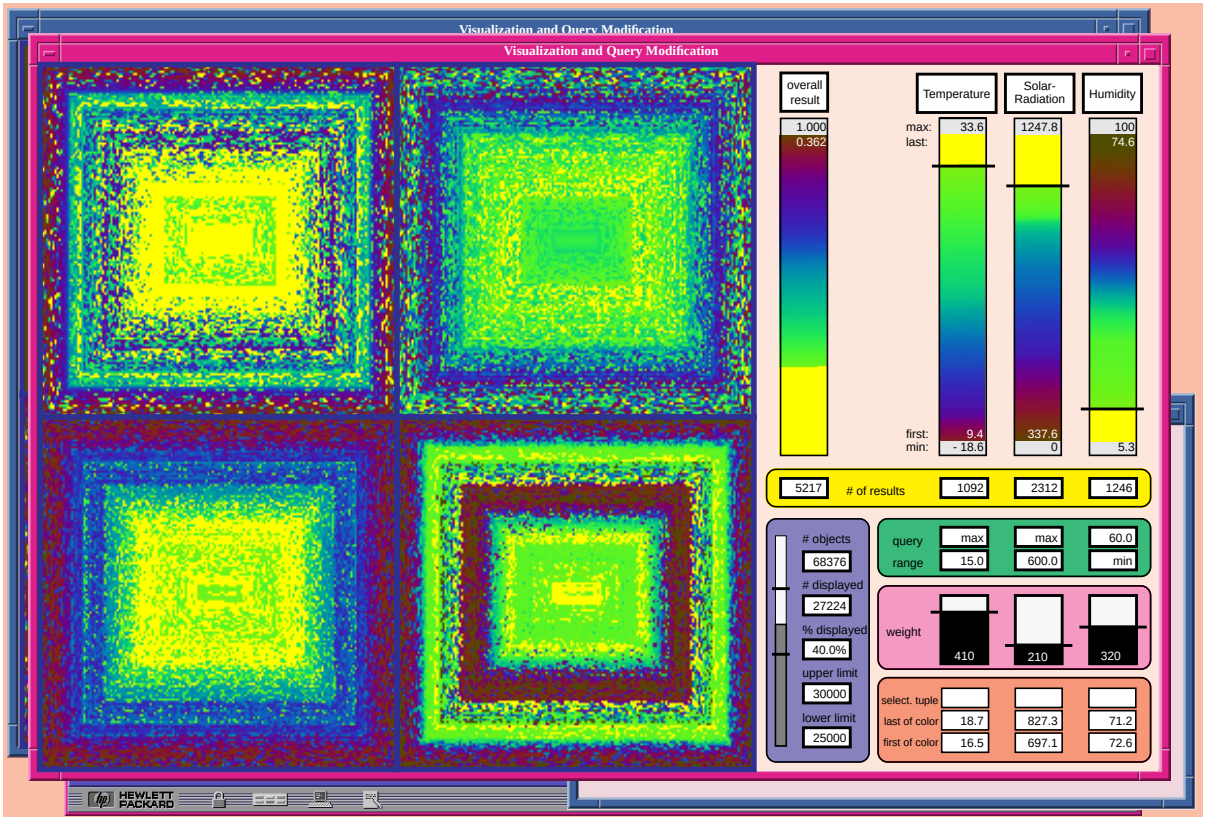

Figure 5: Visualization of the 'OR'-Part of the Query