Visual Feedback in Querying Large Databases

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

Abstract

In this paper, we describe a database query system that provides visual relevance feedback in querying large databases. The goal of our system is to support the query specification process by using each pixel of the display to represent one data item of the database. By arranging and coloring the pixels according to their relevance for the query, the user gets a visual impression of the resulting data set. Using sliders for each condition of the query, 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 a complex query, the user gets visual feedback for each part of the query and, therefore, will easier understand the overall result. The system may be used to query any database that contains tens of thousands to millions of data items, but it is especially helpful to explore large data sets with an unknown distribution of values and to find the interesting hot spots in huge amounts of data. The direct feedback allows to visually display the influence of incremental query refinements and, therefore, allows a better, easier and faster query specification.

1. Introduction

In very large databases with tens of thousands or even millions of data items it is often a problem to find the data a person is interested in. Scientific, engineering or environmental databases, for example, contain large amounts of data that, in many cases, are collected automatically via sensors and (satellite) monitoring systems. In querying such systems even a user who is experienced in using the database and the query system may have difficulties to find the interesting data spots. If the user does not know exactly 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 expected, at least more than the user is willing to deal with. The core of the problem in searching huge amounts of data is the process of query specification. With today's database systems and their query interfaces, a person has to issue queries in a one-by-one fashion. Generally, there are no possibilities to slightly change a query, to express uncertain or vague queries. Most important, the user gets no feedback on his query except the resulting data set containing either no data items and thus no hint for continuing the search or too many data items and thus too many to look at.

Many approaches have been made to improve the database query interface by providing a better feedback in cases of unexpected results. One approach are graphical database interfaces that allow the user to browse the data (e.g. BAROQUE [Mot86], FLEX [Mot90] or GRADI [KL92]). In general, these systems only support the user in browsing the resulting data and in finding errors made because of not knowing the database system, the data model and query language, and/or the schema of the database. Another approach are cooperative database interfaces [Kap82, ABN92] that try to give 'approximate answers' in cases where the query does not provide a satisfactory answer. Such systems use techniques like query generalization [Cha90] that is dropping or relaxing a selection predicate in cases where the original query fails, and statistical approximations or intensional responses instead of full enumeration in the cases of large results (key ideas are presented in [JKL77] for the first time). Cooperative systems mainly help the user to understand the result better and to refine erroneous queries but do not help to visualize huge amounts of data and to find interesting correlations between attributes and their values in the context of a specific query. Even if cooperative systems provide statistical distributions for the values of a specific attribute, there is no possibility to relate this distribution to distributions of other attributes. For relating the results of two independent selection predicates, a cooperative system may determine the correlation coefficient, but this may not be very helpful for a specific data item from a huge data set. Additionally, cooperative interfaces cannot give direct feedback for dynamically changing queries.

Our idea to improve the database query interface is to support the query specification process by visually representing the result. Many approaches to visualize multivariate, multidimensional data have been proposed for various purposes in different application contexts [PG88, Bed90, FB90, ID90, LWW90, MGTS90, MZ92]. In dealing with databases consisting of tens of thousands to millions of data items, our goal is to visualize as many 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 1024 x 1280 pixels it is about 1.3 million pixels. Our concept is to use each pixel of the screen to visualize the data set resulting from a query. By the visual feedback a better, easier and faster query specification can be achieved. Important is the interactiveness of such a system. The user should have the possibility to modify the query on-line and to see the changes of the visualized data set immediately. By playing with such a system, the user may learn more about the data than by issuing hundreds of queries. As we will show later, it is especially useful in the case of complex queries such as range queries on multiple attributes.

The rest of the paper is organized as follows: Section 2 elaborates on the new query paradigm which is the basis of our query system. Section 3 introduces our concepts for visualizing large data sets of multivariate data and elaborates on some of the possibilities in designing the system. Section 4 illustrates the features provided by our query and visualization interface. Section 5 summarizes our approach and points out some of the open problems for future work.

2. A New Query Paradigm

In today's database systems, queries are specified in a one-by-one fashion. This is adequate if the user of the database exactly specifies the desired data and accesses a clearly separated data set. For many application areas where databases are used on a regular basis, e.g. accounting, reservation systems, and so on, queries are often based on keys accessing exactly the desired data. If e.g. a person deposits money to a specific account or if all transactions for a specific account are searched for, the resulting data set is clearly separated and, therefore, in general, one query is sufficient to get the desired data. In other application areas, however, especially those with very large data volumes such as scientific, engineering or environmental databases, it is often difficult to find the desired data. Problems occur e.g. if the database contains data different to what the user expected or if the user does not know exactly what s/he is looking for. In the latter case, querying the database is like an inexact search. If a query does not provide the desired result, usually the database is queried by an other similar query differing in just one detail. While searching for the desired data, often many similar queries are issued before the user is able to find the desired result.

Many problems in querying a database arise if the user does not know the database system, the data model and query language, and/or the schema of the database. But even if the user has perfect knowledge in all these domains, i.e. all queries are completely correct (syntactically as well as semantically), queries may have results which do not correspond to the users intention. The reason is that the user does not know the specific data in the database. Therefore, in most cases, it is very difficult for the user to estimate the amount of data that will be retrieved, especially for complex queries with many selection predicates and for range queries. With our query interface, we focus on the problems arising from the vagueness of the answers for complex queries and try to give the users more feedback on their queries. If e.g. a researcher in environmental science is searching in a huge database of test series for significant values, s/he might be looking for some correlation between multiple parameters for some specific period of time and some geographic regions. Since none of the parameters for the query is fixed, in general, it is very difficult to get the desired information. The researcher would probably start to specify one query that corresponds to some assumption and after issuing many refined queries and applying statistical methods to the results, s/he might find some interesting correlations.

With our query interface which will be described in the next subsection, the query specification process would be much easier. In the beginning, the user still has to specify one query. Then, s/he gets the visual feedback on the query and may interactively change the query according to the impression from the visualized results. In exploring unknown data sets, the principle of incremental query refinement guided by visual feedback can be very helpful for the user to find the desired data. The described query specification strategy may be characterized as a top down strategy since the user first specifies a query and then applies sliders for all attributes used in the query to reduce or enlarge the data retrieved. Also a *bottom up* strategy is imaginable where the user first gets a visual representation of the whole database and uses sliders for the different attributes to focus on the desired data. In the following, we only describe the top down strategy which is implemented in our 'VisDB'-system.

3. Visualizing Large Data Sets

As already mentioned, the basic idea of our query and visualization interface is to present as many data items as possible at the same time 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 a visual feedback on the query and that s/he can easily focus on the desired data, understand the influence of various query components and find out why slightly different queries have completely different results. In the following, we are going to discuss the important aspects in designing our query and visualization interface: the screen layouts, the heuristics used to reduce the amount of data that is displayed, the query facilities for different data types including potential distance functions and the weighting function used to combine the different query parts of complex queries.

3.1 Screen Layouts

The principle idea for visually displaying the data on the screen is to sort them according to their relevance with respect to the query and to present different relevance factors in different colors. The sorting is necessary to avoid completely sprinkled images that would not help the user in understanding the data. One interesting question in designing the system was how to arrange the relevance factors on the screen. 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 one hundred percent correct answers are colored yellow in the middle and the rest are rectangular spiral-shaped around this region. The colors range from yellow over green, blue and red to almost black and denote the distance from the correct answers (c.f. section 3.2). The resulting window for the overall result is always similar to the upper left part of the visualization window presented in figures 2-4. Overall result windows of different queries only differ in the size of the areas with different colors and may even be completely yellow in cases where all the data is a completely correct result or almost black in cases where all the data is a completely wrong result. In figure 2, we display the overall result window for a query with the relevance factors being almost equally distributed.

Up to this point, the visualization of the overall result only contains information on the amount of data being retrieved and on the distribution of approximate answers for a specific query. This information may already be very helpful for the user; however, to really help the user in refining the query, 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 (c.f. figure 2-4). In these separate windows, we place the pixels for each data item in the same place as the overall result for the data item in the overall result window. Now, the user gets additional information about data that fulfills some of the conditions but fails to fulfill some other conditions. Only by the visual color impression of the single screens, the user gets information on how restrictive each of the conditions 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 data items 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 real 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 straightforward idea was to display the data in 2D or 3D with selected attributes assigned to the axis. With such kind of arrangements, however, we have 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 all 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 one of our goals, namely to present at 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 really makes sense, 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 exists, our representation can be of great value to provide visual feedback in querying the database. Stimulated by real 2D or 3D representations of the data, we got the idea to improve our interface by including some feedback on the direction of the distance. 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.

With this kind of representation which is currently being implemented, we do not represent the distance of data items directly by their location, but we denote the absolute value of the distance by their color and the direction by their location relative to the correct answers (colored yellow). The advantage of this kind of representation is that each data item may be assigned to one pixel and no overlay of data items with the same distance does occur. A problem may occur in some special cases if e.g. no data items exist that have a negative distance for both attributes but many data items that have a negative distance for one of them and a positive one for the other one. In this case, the bottom left corner of the window would be completely empty. In the worst case, two diagonally opposite corners of the window may be completely empty and, as a result, only half as many data items are presented to the user as possible. Even in this case, the user gets a valuable information on how s/he has to change the query to get more or less results. In summary, it may be noted that maximizing the number of data items conflicts with arrangements that have multiple attributes assigned to the axis.

3.2 Calculating the Distance

An important question in building our system was to design the distance functions used to calculate the distance of the data in the database from the desired value as specified in the query. The distance functions can be different for different data types and even for a single data type multiple distance functions may be useful. For number types such as *integer* or *real* and other metric types such as *date* the distance of two values is easily determined by their numerical difference. For non-metric types such as enumerations with a non-interpretable distance between different values (ordinal types e.g. *grades*) or even with non-comparable values (nominal types e.g. *professions*) there is no obvious way to determine the distance. For ordinal types the distance may be defined by some domain-specific distance function or by a distance matrix containing the distance for all pairs of values. A distance matrix may also be useful for nominal types but, in some cases, even a constant value may be an adequate distance. For the data type *string* there are many possibilities to calculate the distance. Depending on the application and the context of the retrieval, the user may want to choose between the lexicographical difference, characterwise difference, substring difference or even some kind of phonetic difference. In information retrieval applications that are using keyword or natural language information for querying the database, even a semantic distance may be used as, for example, in [KKL92] where the overall distance is deduced from the distances in the corresponding object-oriented verb and noun hierarchies. If dealing with queries consisting of multiple conditions, other specific distance functions may be

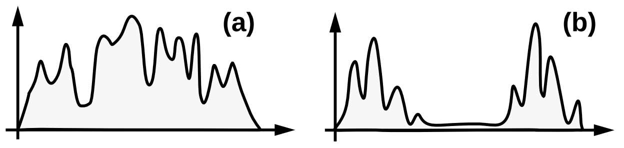

Figure 1: two density functions

used to combine the values of multiple attributes. Examples are the Manhattan distance or the Euclidean distance in n-dimensional space (see section 3.4).

3.3 Reducing 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 most exact way is to use a statistic 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)* the distribution and *f(x)* the density function.

Let r be the number of distance values that can be displayed on the screen and n be the number of data items in the data base, then only data items with an absolute distance in the range [0, (r/n)-quantile] are chosen to be presented to the user. If negative and positive distance values are used then the range of values presented to the user is given by [α_0*(1-r/n)-quantile, (α_0*(1-r/n) + r/n)-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 3.1), correspondingly the combined α-quantiles for two dimensions may be used. For screen layouts with separate windows for each selection predicate the number of data items that are displayed may be chosen lower to be able to display the windows for all selection predicates non-overlapping on the screen. In this case, the parameter r has to be chosen correspondingly lower. For the visualization to be useful to study specific data items or small groups of data items, several pixels can be used to represent the distance for one data item. A zoom out of the image may be helpful even if the amount of data that could be displayed is very large. In empirical tests with real data we found that using 4, 6, 12 or 16 pixels for each data item can be very helpful. This causes the number of presentable data items to be divided by 4, 6, 12 or 16 and again, the quantiles have to be adapted correspondingly.

The α-quantiles are the best choice if we want to present as many data items as possible. We found, however, that in many cases it is better to present less data items if the density function of the distance values has multiple peaks (see figure 1 for an example for two density functions). If we have e.g. two groups of distance values each being in different orders (see figure 1b), 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 the different colors. To implement this heuristics, first we define the range [r_{min}, r_{max}] for the number r of distance values that is intended to be 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. Then, we choose the data item with the highest s_i to be the last data item that is displayed.

3.4 Combining Distances into the Relevance Factor

In this section, we extend the calculation of the distance to queries consisting of multiple selection predicates. To determine the combined distance for a complex query, first the distances are calculated for each selection predicate of the query as described in subsection 3.2. The combination of distances for the different selection predicates can be a problem because the distances for the different selection predicates now 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 (c.f. section 3.2) 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 1000 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 selection predicates. Therefore, in general, it is necessary to obtain weighting factors (w_j , $j \in 1, .., \#selection\_predicates (\#sp)$) representing the order of importance of the selection predicates assigned by the user. This is somehow similar to techniques used in information retrieval that have been proposed to allow a ranking of the resulting data according to its relevance for the query. As we will show in section 4.1, the weighting factors may also be changed dynamically allowing to get a visual feedback on their impact on the result.

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 [0, d_{max}] (or [d_{min}, d_{max}] in case of bi-directional distances) for each selection predicate to a fixed range (e.g. [0, 255]). In 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 to first reduce the number of data items that are considered 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

since the less a selection predicate is weighted, the higher is the probability that data with a greater distance for this selection predicate is needed. Then, the data are normalized by transforming the range [0, d_{max}] (or [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 use 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

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

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

After calculating the combined distance for the whole condition, the relevance factor is determined as the inverse of that distance value. The *relevance factor* combines the information on how good a data item approximates the query into one value representing the relevance of the data item with respect to the query.

3.5 Coloration of the Relevance Factors

Visualizing the relevance factors using color, corresponds to the task of mapping a color scale to a single parameter distribution. The advantage of color over gray scales is that the number of just noticeable differences (JNDs) is much higher. The main task is to find a path through color space that maximizes the number JNDs but, at the same time, is intuitive for the application domain [LRR92].

In designing the system, we experimented with different colormaps varying in hue, saturation and value. We found that the coloration has a high impact on the system to be intuitive. The user, for example, may implicitly connect good answers with light colors and bad answers with dark colors or the user may be accustomed to green colors for good answers and red colors for bad answers (like the colors used for traffic lights). 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 value (intensity) and a hue (color) ranging from yellow over green, blue and red to almost black are a good choice to denote the distance from the correct answers. Still, the user may define different colormaps and use them instead of the standard colormap which has been used for our examples.

4. The Query Interface

In using our query and visualization system, the possibility to modify queries dynamically 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 condition and on special areas 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 understand the effects of query modifications quickly and to learn more about the data in the database, especially in the context of querying large databases with millions of data items. In the following, we will give a description of the querying process.

The first step is to specify a query. The user may use traditional query languages such as SQL or graphical user interfaces such as GRADI [KL92]. Since any existing query interface may be used in this step and since it has basically no impact on our system, we do not elaborate on this step but focus on the following steps: visualization of results and query modification. In figure 2, we present the interactive query interface of the 'VisDB' system that is currently being implemented. The screen is divided in the left portion, the 'Visualization' windows and the right portion, the 'Query Modification' window. In the 'Visualization' windows, the user gets a visual representation for the overall result and for each selection predicate (see figure 2). In the 'Query Modification' part, sliders for the selection predicates and weighting factors as well as some other options are provided. The color spectrum of each slider corresponds to the distribution of distances for the corresponding attribute. The yellow region which may be quite large like in the case of the third selection predicate in figure 2 indicates the number of data items fulfilling the selection predicate; the regions with other colors correspond to the amount of data that have specific distances for the selection predicate. Inside each slider, the lowest and highest value of the approximate answer 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' and finally the 'query range' and 'weighting factors'. In the following, we will describe how these parameters may help the user to explore the data 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 'selected color' fields. The possibility to get the values corresponding to some color or color range for each selection predicate makes it easier for the user to understand and interpret the visualization and to modify the query. In figure 2, for example, the blue region in the middle of the upper right window may be easily identified by the user. To understand the meaning of this region, however, the user needs additional information to relate colors and attribute values. In this special case, the user may easily observe that data items with values in the ranges of about 82,400 - 96,700 or 146,600 - 151,300 for attribute one are quite good overall answers for the query although their values for attribute one are quite bad as indicated by the blue color. Another help for the user to understand the visualization and to find interesting data spots is to select a specific data item in one of the visualization windows to get the data item highlighted in all visualization windows and the values for the attributes displayed in the 'selected tuple' field of the 'Query Modification' window. 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 windows. 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 in the visualization window that have the selected color for the considered attribute. In the other visualization windows the same data items are displayed allowing the user to easily compare the values for the other attributes of those data items.

The query is displayed with the query parameters represented graphically by the black lines in the sliders and by the value of the upper and lower limit in the 'query' 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 (see figure 2). For numbers, the user may also choose a different kind of slider where the medium value and some allowed deviation can be manipulated graphically (see attribute one in figure 2). Different kinds 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 nonmetric types (ordinal and nominal datatypes) may be, for example, enumerations of the possible values with the possibility to select each of the values or not. Special sliders may be designed for special datatypes and special distance functions, e.g. for strings with different distance functions (c.f section 3.2). Below the query parameter field, the weighting factors are represented graphically. Like the query parameters, the weighting factors may be directly changed using the mouse. The specification of weighting factors may be different if other functions to combine the distances are used.

On the left side of the query modification window, 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, the number of data items in the database, the number of data items being displayed in the visualization window (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 3.3 (c.f. figure 2). 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 re-calculates the visualization after each modification of the query. The user may also switch to an 'auto re-calculate off' mode where queries are only re-calculated on demand. This option is useful for large databases with many data items or if complex distance functions are used, because the re-calculation for each modification may need a considerable amount of time. Other menu options allow the user to add or delete selection predicates at any time, to extend the query or to issue a completely different query. Furthermore, to help the user in specifying queries, we envision advanced facilities that allow the user to identify regions in the visualization window with the system automatically generating queries providing the data items of the identified regions as results.

In figures 2-4, three visualizations of query results are displayed. The query used to generate the visualization in figure 3 only differs from the query in figure 2 in one selection predicate which has been set to a different range. It is interesting that the resulting visualizations for all selection predicates are completely different. The visualization presented in figure 4 results from the same query as displayed in figure 2 except that the weighting factors for selection predicates one and two have been set to 20 which is one dimension lower than the weighting factor of selection predicate three. Interesting are the clear identifiable regions of different colors which denote clusters of data items with a comparable distance, and the correlations between the windows for the different selection predicates. Also interesting, but not easily identifiable in the printed version of our visualizations are hot spots, i.e. single exceptional data items in regions which are otherwise homogeneous. Much of the information the user may get out of the visualization is related to the semantics of the data. Due to space limitations, in this paper we do not elaborate on these aspects, since we would have to introduce the schema and the instances of our data base in more detail.

5. Conclusions

A major problem in querying large databases is to find the interesting data and their properties, e.g. hot spots, clusters of similar data or correlations between different parameters. The user must have some advance knowledge about the data in the database to be able to find the interesting data in an acceptable amount of time. Even if the user has some knowledge about the data, it can be quite tedious to focus on the desired data. For complex queries, the user often obtains unexpected results that are difficult to interpret and contain no hints on how to modify the query. Our idea to improve the query specification process is to visualize the query result including a set of approximate answers. For all data items that are not fulfilling the query, we calculate the distance with respect to each of the selection predicates, combine the distances for all selection predicates into one value using the weighting factors obtained from the user and visualize the single and combined distances. With our interactive query modification interface, the user may dynamically change the query and receives immediate feedback by the visual representation of the resulting data set. Our query and visualization interface improves the possibilities to explore large databases allowing the user to find results which, otherwise, would remain hidden in the database. The system further helps to considerably reduce the time needed for exploring the database.

The visualizations presented in figures 2-4 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. We believe that our ideas are general enough to be readily applied to existing databases and database systems. Currently, we are exploring a large environmental database with about 180,000 data items. In interfacing to current commercial database systems, however, performance problems arise since no access to partial results of a query is available, no support for incrementally changing queries is provided and no multidimensional data structures such as R*-Trees [BKSS90] or Buddy-Trees [SK90] are used for fast secondary storage access. We are currently trying to improve the performance in directly interfacing with the database system. In the future, we plan to realize our query and visualization interface on a parallel machine to be able to support a really interactive query modification even for mid-size to large amounts of data and complex distance functions.

Inspired by using our prototype, we already have several ideas to extend our system. One extension is the automatic generation of queries corresponding to some specific region in one of the visualization windows. The region may be graphically identified by the user. Then, the system should try to find adequate selection predicates that provide the desired data items as a result. Another idea is a special handling of complex queries with tens to hundreds of parameters that are only used on smaller data sets. For this case, special screen layouts with one continuous area for each data item may be better suited to help the user to identify similarities. To further improve our system, we intend to apply it to many different application domains, each having its own parameters, distance functions, query requirements and so on. We believe that query and visualization systems like ours can also be useful in other application contexts. They may be the starting point for new ways to visually solve problems that have proven to be very difficult using conventional tools. Querying of very large databases is just one example.

Acknowledgments

At this point, we should mention that the idea for the work reported in this paper originated in the sessions and tutorials of last years *Visualization* conference. We are thankful to many people with whom we discussed first ideas during the conference especially to Jeff Beddow and Cliff Beshers who gave the tutorial on '*Designing a Visualization Interface for Multidimensional*, *Multivariate Data*' as well as to Georges Grinstein and Ben Shneiderman who both encouraged us to work on the ideas reported in this paper.

References

[ABN92] 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.

[Bed90] Beddow J.: *'Shape Coding of Multidimensional Data on a Mircocomputer Display'*, Visualization '90, San Francisco, CA., 1990, pp. 238-246.

[BKSS90] 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, N.J., 1990, pp. 322-331.

[Cha90] Chaudhuri S.: *'Generalization and a Framework for Query Modification'*, Proc. 6th Int. Conf. on Data Engineering, Los Angeles, CA., 1990, pp. 138-145.

[FB90] Feiner S., Beshers C.: *'Visualizing n-Dimensional Virtual Worlds with n-Vision'*, Computer Graphics, Vol. 24, No. 2, 1990, pp. 37-38.

[ID90] Inselberg A., Dimsdale B.: *'Parallel Coordinates: A Tool for Visualizing Multi-Dimensional Geometry'*, Visualization '90, San Francisco, CA., 1990, pp. 361-370.

[JKL77] 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.

[Kap82] Kaplan S. J.: *'Cooperative Responses from a Portable Natural Language Query System'*, Artificial Intelligence, Vol. 19, 1982, pp. 165-187.

[KKL92] Keim D. A., Kim K.-C., Lum V.: *'Intelligent Natural Language Processing for Media Data Query'*, Proc. 2nd Int. Golden West Conf. on Intelligent Systems, Reno, NEV., 1992, pp. 257-262.

[KL92] 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: Lecture Notes in Computer Science, Springer, pp. 95-112.

[LRR92] Levkowitz H., Robertson P., Rogowitz B.: *'Color Theory and Models for Computer Graphics and Visualization'*, Tutorial No. 5, Visualization '92, Boston, MA., 1992.

[LWW90] LeBlanc J., Ward M. O., Wittels N.: *'Exploring N-Dimensional Databases'*, Visualization '90, San Francisco, CA., 1990, pp. 230-239.

[MGTS90] 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.

[Mot90] 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.

[MZ92] 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.

[PG88] Pickett R.M., Grinstein G.G.: *'Iconographic Displays for Visualizing Multidimensional Data'*, Proc. IEEE Conf. on Systems, Man and Cybernetics, Beijing and Shenyang, China, 1988.

[SK90] 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.