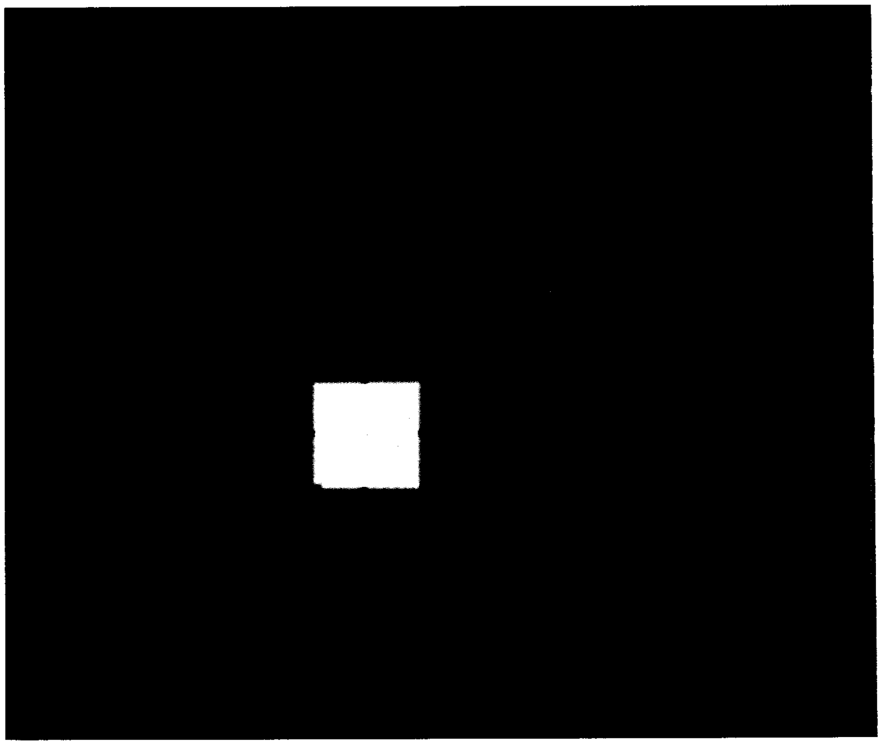
VisDB: Database Exploration Using Multidimensional Visualization

Daniel A. Keim and Hans-Peter Kriegel
University of Munich



In this system, each display pixel represents one database item. Pixels are arranged and colored to indicate the item's relevance to a user query and to give a visual impression of the resulting data set.

Scientific, engineering, and environmental databases can contain data collected automatically and continuously via sensors and, for example, satellite monitoring systems. Finding the data you want in these very large databases—with tens of thousands or even millions of data items—can be very difficult. Even researchers experienced in using a database and query system have trouble "mining" these databases for the interesting data sets. Users who do not know the data and its distribution exactly might need dozens of queries to get started.

The query specification process is the core of the problem. With today's database systems and query interfaces, users must issue queries one at a time. Generally, you cannot change a query slightly or express vague queries. Most importantly, you get no feedback on the query except in the form of the resulting data set—which may contain either no data items, and thus no hint for continuing the search, or too many data items to browse efficiently.

Researchers have developed many approaches to improve the database query interface so that it gives better feedback when the query yields unexpected results. For example, graphical database interfaces let users browse the data visually.[1,2] These interfaces either sort the data for users or provide direct hypertext-like access to more detailed versions of it. Another approach uses cooperative database interfaces[3] that try to give "approximate answers" when the original queries do not provide a satisfactory answer. This approach employs such techniques as query generalization, which drops or relaxes a selection predicate when the original queries fail, and statistical approximations or intensional responses—instead of full enumeration—when the query results are very large. Joshi, Kaplan, and Lee[4] first presented the key ideas for these techniques. Cooperative systems mainly help the user understand the results and refine erroneous queries; they do not help find interesting properties of the data such as functional dependencies, local correlations, or exceptional data items.

Information retrieval is another area that relates to our work. A lot of research in this area has been done to improve recall and precision in querying databases of unstructured data such as full text. For example, one approach employs user-provided relevance assessments of results to rerank the results or rerun adapted queries.[5]

The VisDB system supports the query specification process by representing the result visually. The main idea behind the system stems from the view of relational database tables as sets of multidimensional data where the number of attributes corresponds to the number of dimensions. In such a view, it is often unclear which dimensions are independent and which are dependent. In most cases, only a limited number of the dimensions are of interest in a certain context. In the VisDB system, we therefore restrict the number of visualized dimensions to those that are part of the query. In other words, the dimensionality of our visualizations corresponds to the number of selection predicates.

Researchers have proposed many approaches to visualizing arbitrary multivariate, multidimensional data. The well-known books of Bertin[6] and Tufte[7] include many examples. More recent techniques include shape coding,[8] worlds within worlds,[9] parallel coordinates,[10] iconic displays,[11,12] dimensional stacking,[13] hierarchical plotting,[14] and dynamic methods.[15] In developing a system to handle databases consisting of tens of thousands to millions of data items, our goal is to visualize as many data items as possible while, at the same time, giving 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 on the order of one to three million pixels. For example, our 19-inch displays with a resolution of 1,024 × 1,280 pixels has about 1.3 million pixels.

To explore large data sets efficiently, a system must be interactive. Empirical studies show that interactive, slider-based interfaces considerably improve efficiency and accuracy in accessing databases.[16] Equally important is the possibility of getting immediate feedback on the modified query. By playing with such a system, users can learn more about the data than they can by issuing hundreds of queries.

A new query paradigm

In today's database systems, users specify queries in a one-by-one fashion. This is adequate if you can specify the desired data exactly and access a clearly separated data set. Many database applications work this way. For example, accounting and reservation systems base their queries on keys that access the desired data exactly. If you search transactions for a specific account, the resulting data set is clearly separated. Therefore, one query generally suffices to get the desired data.

In other application areas, however, especially those with very large data volumes such as scientific, engineering, and environmental databases, it is often difficult to find the desired data. Problems occur if the database contains data different from what the user expects or if the user does not know exactly what to look for. In the latter case, querying the database is like an inexact search. If a query does not provide the desired result, the user must query the database again, usually by a similar query that differs in just one detail. Often, the user must issue many similar queries before finding the desired result.

Users have problems in querying a database when they do not know the database system, the data model and query language, or the database schema. But even if they have perfect knowledge in all these domains—that is, their queries are both syntactically and semantically correct—users can still get query results that do not correspond to their intentions, simply because they do not know the specific data in the database. In this case, they will find it very difficult to estimate the amount of data that will be retrieved, especially for range queries and complex queries with many selection predicates.

The VisDB query interface uses visualization techniques to give users more feedback on their query results. For example, environmental scientists searching a huge database of test series for significant values might be looking for some correlation between multiple dimensions for a specific period of time and geographic region. Since none of the query parameters is fixed, finding the desired information is generally very difficult. Therefore, researchers might start by specifying one query that corresponds to some assumption, but they might not find an interesting correlation until they issue many refined queries and apply statistical methods to the results.

The VisDB system simplifies the query specification process. To begin, users still have to specify one query. Then, guided by the visual feedback, they can interactively change the query according to the impression they get from the visualized results. In exploring very large databases, the visualization of results coupled with the means of incrementally refining the query offer an effective way to find interesting data properties.

The key idea of VisDB is to use the phenomenal capabilities of the human vision system for analyzing midsize amounts of data efficiently and immediately recognizing patterns that would be difficult, even impossible, for computers to find. One major research challenge is to find visualization techniques that support the user in analyzing and interpreting large amounts of multidimensional data.

Visualizing large data sets of multidimensional data

The basic idea underlying our visualization techniques is to use each pixel of the screen to visualize the data items resulting from a query. The query results thus give the user not only the data items fulfilling the query but also a number of data items that approximately fulfill the query. The approximate results are determined by calculating distances for each selection predicate and combining them into the *relevance factor*. The distance functions are data-type and application dependent, so the application must provide them. Example distance functions include the numerical difference (for metric types), distance matrices (for ordinal and nominal types), and lexicographical, character, substring, or phonetic difference (for strings). The sidebar titled "Calculating the relevance factors" describes important aspects of this procedure.

Basic visualization technique

Our basic technique for visually displaying the data on screen is to sort them according to their relevance with respect to the query and to map the relevance factors to colors. The sorting is necessary to avoid completely sprinkled images that would not help the user understand the data. One question in designing the VisDB system was how to arrange the relevance factors on the screen. We tried several arrangements such as top-down, left-

Calculating the relevance factors

The procedure for calculating relevance factors addresses the issues described here.

Calculating the distance. The first step is to determine the distance between the attribute and the corresponding query values for each data item. The distance functions used in this step are data-type and application dependent. In some cases, even for a single data type, multiple distance functions can be useful.

For number types such as integer or real and other metric types such as date, we can determine the distance of two values easily by their numerical difference. For nonmetric types such as enumerations with a noninterpretable distance between values (ordinal types such as grades) or with noncomparable values (nominal types such as professions), there is no obvious way to determine the distance. For ordinal types, the distance might be defined by some domain-specific distance function or by a distance matrix containing the distances for all pairs of values. A distance matrix can also be useful for nominal types, but even a constant value can be an adequate distance in some cases. There are many possibilities for calculating the distance of string data types. Depending on the application and the retrieval context, the user might want to choose between lexicographical, character, substring, or even some kind of phonetic distance.

Combining distances into the relevance factor. The next step combines the independently calculated distances of the different selection predicates. This is not straightforward, however, because we must consider the distances for the different selection predicates with respect to the distances of the other selection predicates.

One problem is that the relative importance of the multiple selection predicates is highly user and query dependent. Only user interaction can solve this because only the user can determine the priority of the selection predicates. Therefore, the user must provide weighting factors w_j, representing the order of importance of the selection, where $j \in 1, \ldots,$ #sp, and #sp is the number of selection predicates.

A second problem is that the values calculated by the distance functions may be in completely different orders of magnitude. For example, in a medical application, a distance of 1 gram per deciliter for hemoglobin may be very high, and a distance of 1,000 erythrocytes per deciliter may be very small. We solve this problem by normalizing the distances. We can define a simple normalization as a linear transformation of the range $(d_{\min}, d_{\max})$ for each selection predicate to a fixed range such as (0, 255).

For combining the independently calculated and normalized distances of multiple selection predicates into a single distance value, we use numerical mean functions such as 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

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

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

After calculating the combined distance for the whole condition, we determine the relevance factor as the inverse of that distance value. The relevance factor thus combines information on how well a data item approximates the query into one value representing a data item's relevance with respect to the query.

Note that special applications may use other specific distance functions, such as the Euclidean, L^p, or Mahalanobis distance in n-dimensional space, to determine the distances of multiple selection predicates.

Reducing the amount of data to be displayed. Since the number of data items in the database may be much higher than the number of data items that can be displayed on screen, we had to find adequate heuristics to reduce the amount of data and to determine the data items whose relevance should be displayed. The most exact way uses a statistical parameter, namely, the α-quantile. The α-quantile is defined as the lowest value ξ_α such that

$$F(\xi_\alpha) = \int_{-\infty}^{\xi_\alpha} f(x)dx = \alpha$$

where $0 \le \alpha \le 1$, $F(x)$ is the distribution function and $f(x)$ is the density function.

Let r be the number of distance values that fit on the screen, #sp 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 $[0, r/(n * (\#sp + 1))$-quantile$]$ are presented to the user. If negative and positive distance values are used, then the range of values presented to the user is given by $[\alpha_0 * (1 - p)$-quantile, $(\alpha_0 * (1 - p) + p)$-quantile$]$ where $p = r/(n * (\#sp + 1))$ and α_0 is determined by α_0-quantile $= 0$.

In the special case of two dimensions assigned to the two axes, we can use the combined α-quantiles for two dimensions. For the grouping arrangement, the number of data items that can be displayed on the screen is lower since each data value requires multiple pixels.

to-right, and centered, and found that arrangements with the highest relevance factors centered in the middle of the window seemed the most natural. As shown in Figure 1, we color the 100-percent-correct answers yellow and place them in the middle of the visualization with the approximate answers creating a rectangular spiral around this region.

The colors range from yellow to green, blue, red, and almost black to denote increasing distance from the correct answers. We chose this color scale empirically (see the sidebar "Coloration of the relevance factors"). To relate the visualization of the over-

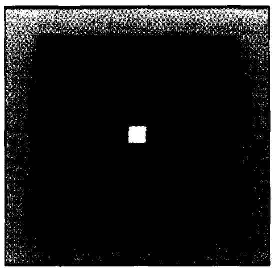


Figure 1. Spiral-shaped arrangement of one dimension.

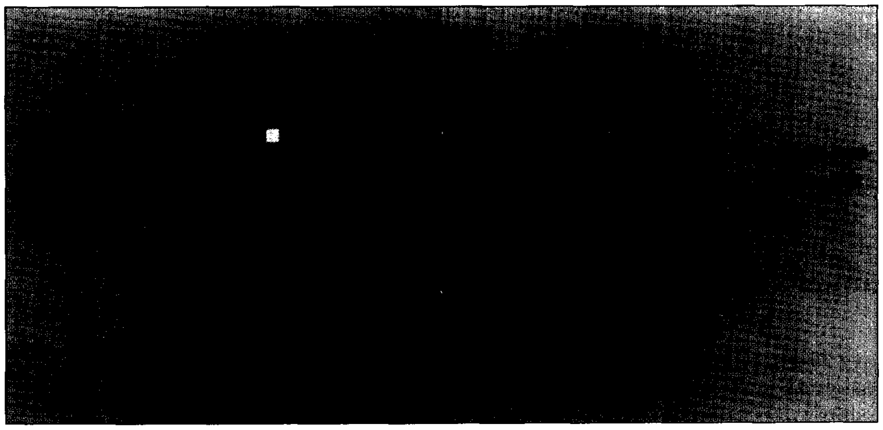


Figure 2. Arrangement of windows for displaying five-dimensional data.

all result to visualizations of the different selection predicates (dimensions), we generate a separate window for each selection predicate of the query and arrange the windows next to each other, as shown in Figure 2. In the separate windows, we place the pixels for each data item at the same relative position as they appear for that data item in the overall result window.

All the windows together make up the multidimensional visualization. By relating corresponding regions in the different windows, the user can perceive data characteristics such as multidimensional clusters or correlations. Additionally, the separate windows for each selection predicate provide important feedback to the user, for example, on the restrictiveness of each selection predicate and on exceptional data items.

Mapping two dimensions to the axes

We also experimented with other screen arrangements of the data items. One straightforward idea was to display the data in 2D or 3D with selected dimensions assigned to the axes. Such arrangements, however, can cause many data items to concentrate in one screen area while other areas remain virtually empty. These arrangements can also cause some data items to be superimposed on others, thereby making the latter items invisible.

Although 2D or 3D visualizations might be helpful in cases where the data have some inherent 2D or 3D semantics, we did not pursue this idea for several reasons. First, in most cases, the number of data items that can be represented on the screen at the same time is quite limited. This conflicted with our goal of presenting as many data items as possible on screen. Second, in most cases where a 2D or 3D arrangement of the data really makes sense, systems using such arrangements have already been built. For example, a 2D visualization is obviously the best support for spatial queries of 2D data, and basically all geographical information systems provide such visual data representations.

However, for all cases where no inherent 2D or 3D semantics of the data exists, our visualization technique can provide valuable visual feedback when querying the database. We decided to improve our interface by including some feedback on the direction of the distance into the visualization. We assigned two dimensions to the axes and arranged the relevance factors according to the direction of the distance. As shown in Figure 3 on the next page, we arrange negative distances to the left and positive distances to the right for one axis dimension; for the other

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 of JNDs but is, at the same time, intuitive for the application domain.[1]

In designing the VisDB system, we experimented with different color maps. We found that coloration has a high impact on the system's intuitivity. The user may, for example, implicitly connect good answers with light colors and bad answers with dark colors, or green colors with good answers and red colors with bad answers (like the colors used for traffic lights). We tried many variations of the color map to enhance the usefulness of our system and selected a color map with constant saturation, increasing value (intensity), and hue (color) ranging from yellow over green, blue, and red to almost black to denote the distance from the correct answers.

The color model used in the VisDB system is a variation of the hue, saturation, value (HSV) model. Instead of the hexcone in the HSV model, we use a circular cone with the intensity defined as the Euclidean distance to the black axis and the saturation defined as the Euclidean distance to the gray axis. In the HSV model, both parameters are determined by using the maximum of (r, g, b). In contrast to color scales generated according to the HSV model, our model provides color scales whose lightness ranges continuously from light to dark colors.

Since the usefulness of color maps varies depending on the user and the application, the VisDB system lets users define their own color maps and employ them instead of our standard color map.

Reference

1. G.T. Herman and H. Levkowitz, "Color Scales for Image Data," *IEEE CG&A*, Vol. 12, No. 1, Jan. 1992, pp. 72-80.

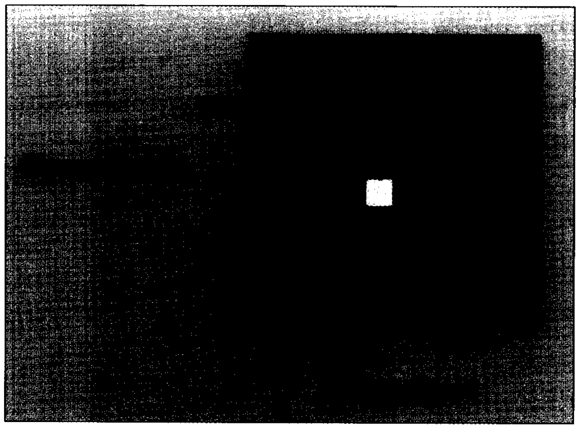


Figure 3. 2D arrangement of one dimension

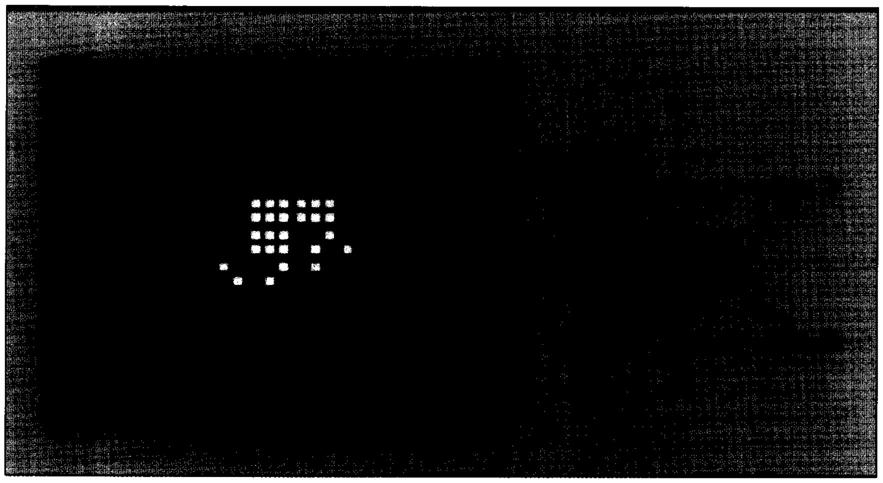


Figure 4. Grouping for five-dimensional data.

axis dimension, negative distances are to the bottom and positive ones to the top.

With this kind of representation, we do not show the distance of data items directly by their location. Instead, we denote the absolute value of their distance by color and their direction by the location relative to the correct answers (colored yellow). Thus, each data item can be assigned to one pixel, and no overlay occurs between data items with the same distance. A problem may arise in some special cases, for example, if there are no data items having a negative distance for both axis dimensions but there are many data items having a negative distance for one axis dimension and a positive distance for the other. 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 could be completely empty and, as a result, only half as many data items as possible presented to the user. Even in this case, the user gets valuable information on how to change the query to get more or fewer results.

Grouping the dimensions

In both the original and 2D arrangements, the pixels corresponding to the different dimensions of one data item are distributed in different windows for each dimension. In contrast, the grouping arrangement places all dimensions for one data item in one area. The idea of grouping the dimensions into one area is similar to the shape-coding approach described in Beddow.[8] In our approach, however, we do not focus on shape to distinguish the data items, and we manage the criterion and arrangement of the data items differently. As shown in Figure 4, we arrange each area in a rectangular spiral shape according to the combined relevance factor of the considered data items. The coloring of distances for the different dimensions can be the same as in the original or 2D arrangement. The generated visualizations, however, are completely different.

Preliminary experiments show that the grouping arrangement requires more pixels per data value. In the original and 2D arrangements, we used one pixel per dimension per data item. Empirical tests show that the grouping arrangement requires an area of at least 2 × 2 pixels per dimension per data item for the visualization to provide useful results (3 × 3 or 4 × 4 pixels provide better results). This implies that only one-fourth (or even one-ninth or one-sixteenth) of the data items can be displayed on screen at one time, making the grouping arrangement suitable only for a focused search on smaller data sets. Note that this arrangement also calls for additional pixels in the area surrounding each data item. Otherwise, it would be impossible to know which pixels belong to which data item.

Even though it may visualize fewer data items, the grouping arrangement provides more useful visualizations for data sets with larger dimensionality. In the original and 2D arrangements, the pixels for each dimension of the data items are related only by their position. For relatively small dimensionality (fewer than eight dimensions), humans seem to relate the different portions of the screen quite easily. The higher the dimensionality, the more difficult it becomes. The grouping arrangement does not require the user to make these correlations and therefore seems advantageous for larger dimensionalities.

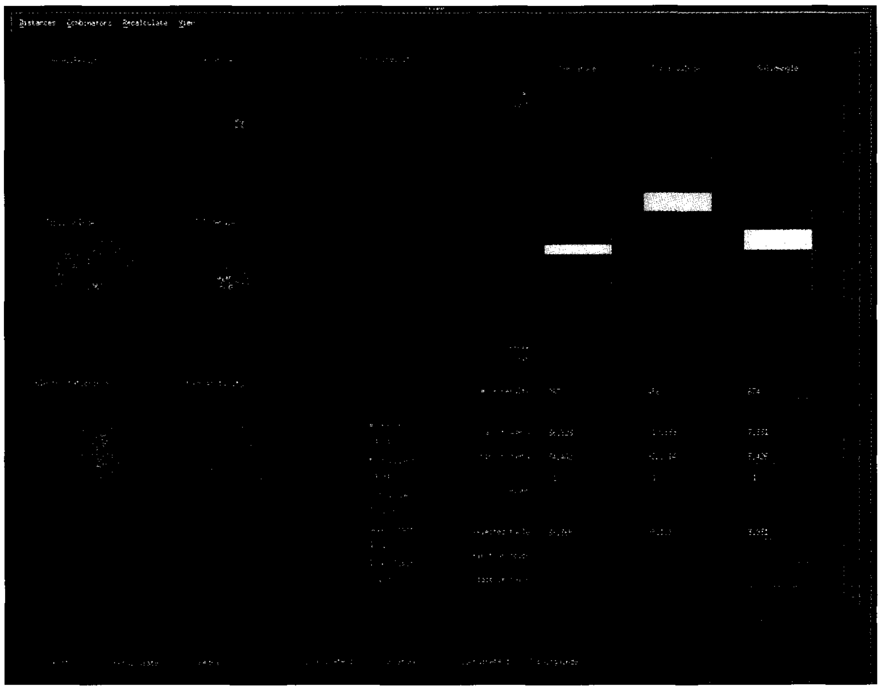


Figure 5. The VisDB system interface.

For example, sliders for nonmetric types (ordinal and nominal data types) can display enumerations of the possible values and allow users to select the values. Users can design special sliders for special data types and distance functions, such as strings with different distance functions.

Below the three sliders on the far right, the interface lists several parameter fields for each selection predicate, namely, the number of results, query range (min and max), weighting factors, data values of a selected tuple, and data values corresponding to a selected color range (first and last). The possibility of correlating data values to some color or color range for each selection predicate might help the user understand the visualization and modify the query accordingly. Users can focus on sets of data items with a specific color by selecting a color range with one of the sliders. VisDB will then retrieve only those data items in the corresponding visualization window that have the selected color for the considered attribute. The other visualization windows will also display these same data items, allowing the user to easily compare the values of the other attributes of those data items.

Also helpful in understanding the visualization and finding interesting data spots is the possibility of selecting a specific data item in one of the visualization windows, highlighting it in all visualization windows, and displaying the values for the attributes in the selected tuple field. This option lets the user focus on exceptional data items or get an example of a data item from an interesting region in one of the windows.

Below the color spectrum for the overall result, there are fields for the number of data items in the database, the number of data items displayed in the visualization window (absolute value and percentage), and the number of resulting data items presented to the user. Using a slider, the user can change the percentage of data displayed or the allowed range. (In the latter case, the percentage is determined using the heuristics described in the sidebar "Calculating the relevance factors.") Changing the percentage of data displayed can 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 query modification. The user also has the option of having the system recalculate queries only on demand. This option is useful for large databases with many data items or for complex distance functions that take a considerable amount of time to recalculate. Other menu options let the user choose dif-

Interactive data exploration

To give users immediate feedback on query changes, VisDB needs dynamic query modification capabilities. The visualizations provide feedback on the amount of data retrieved, the restrictiveness of conditions, the distribution of distances for each condition, and special areas the user might be interested in. For example, if the yellow region in the middle of each window grows larger, it means that more data items are fulfilling the condition (and vice versa). If a window becomes darker, the corresponding selection predicate is becoming more restrictive. If the overall structure changes, the distribution of distances for the corresponding selection predicate is changing, and so on. These visual indicators help users understand the effects of query modifications quickly and learn more about the data in the database, especially in the context of large databases with millions of data items.

In the VisDB system, users initially specify their queries through graphical user interfaces such as Gradi[2] or traditional query languages such as SQL. As a result of this query, they get the VisDB interactive query and visualization interface shown in Figure 5. The interface features a "Visualization" portion on the left and a "Query Modification" portion on the right. The Visualization portion displays the data set resulting from the query, including a certain percentage of approximate answers, by using one of the three visualization methods described in the previous section.

The Query Modification portion provides sliders for modifying the selection predicates and weighting factors as well as some other options. Different kinds of sliders are available for different data types and distance functions. Sliders for numbers, for example, allow graphical manipulations of either the lower and upper limits or the medium value and some specified deviation. Sliders for discrete types reflect the discrete nature of the data by allowing only discrete movements of the slider.

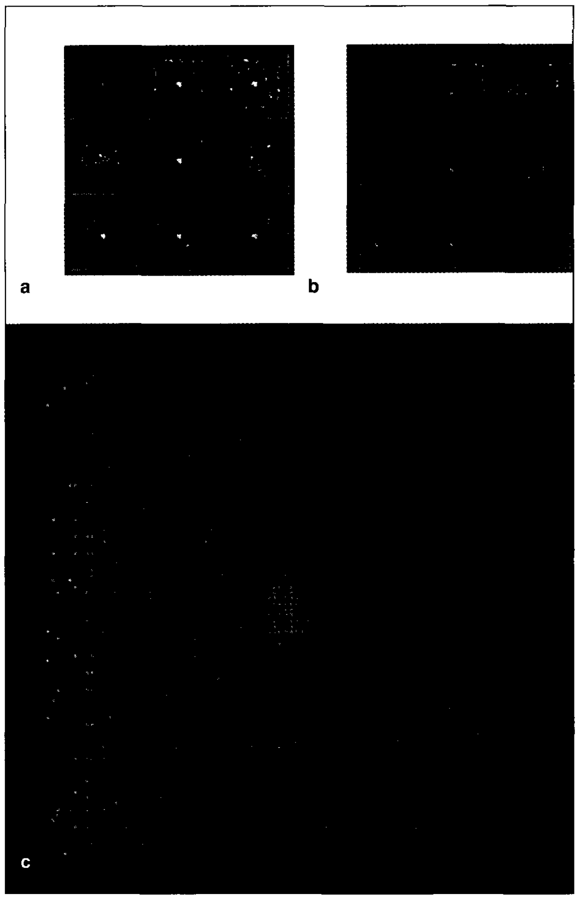
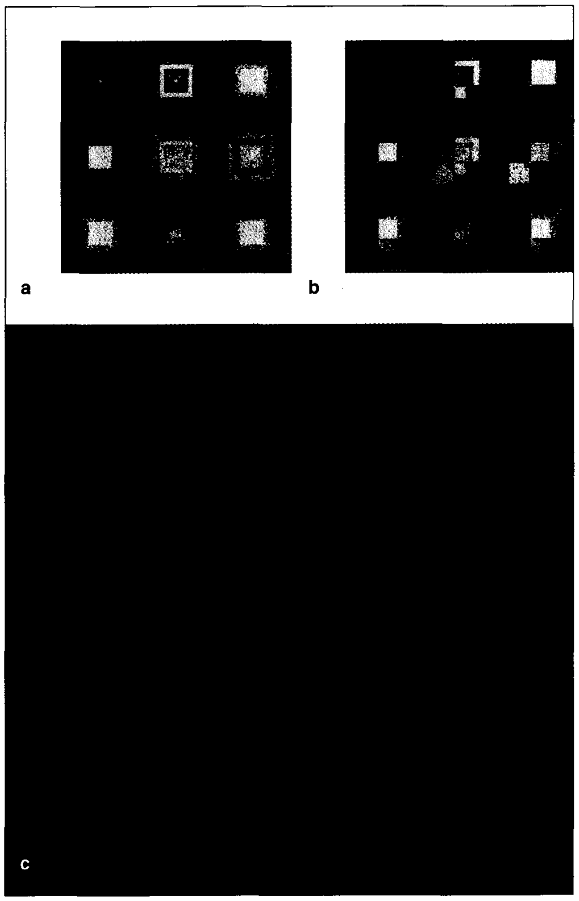


Figure 6. Eight-dimensional data displayed with the three different visualization methods (1,000 data items): (a) basic visualization technique, (b) 2D arrangement, and (c) grouping arrangement.

Figure 7. Eight-dimensional data displayed with the three different visualization methods (7,000 data items): (a) basic visualization technique, (b) 2D arrangement, and (c) grouping arrangement.



ferent distance or combinator functions, select a different visualization technique or slider type, add or delete selection predicates, extend the query, or issue a new query.

Examples

Figures 5 through 8 display several visualizations of query results. We generated the visualization in Figure 5 by using surface point data from a large molecule complex (subtilisin carlsberg with eglin). In our molecular biology project, we have used the VisDB system to find regions where molecules can dock by identifying sets of surface points with distinct characteristics.

In evaluating our visualization techniques, we currently explore other data sets including a large database of geographical data, a large environmental database, a NASA earth observation database, and artificially generated data sets. Artificial data sets are crucial for comparing different visualization techniques to find their strengths and weaknesses.[17] They let us vary the number of data items, number of dimensions, and data properties (for example, the distribution of each dimension and the number and size of clusters) for controlled comparisons.

The visualizations displayed in Figures 6 through 8 use artificially generated data consisting of a uniformly distributed base data set and multiple clusters. The data set used for Figure 6 consists of 1,000 eight-dimensional data items with five clusters. The data set used for Figure 7 is similar, except it consists of 7,000 data items. Figure 8 is generated from a database with 100,000 five-dimensional data items containing five clusters. In the visualizations, many regions of different colors are clearly identifiable and denote clusters of data items with a comparable distance. There are interesting correlations between the windows for different selection predicates. For example, regions that have a specific color in the window for one selection predicate have a different color in the window for another selection predicate. Corresponding regions with different colors denote clusters of data items with similar characteristics. The color of the region for some dimension corresponds to the cluster's distance from the reference region in that dimension.

In Figure 8, the red region for selection predicate 3 corresponds to the green region for selection predicate 2, which means there is a cluster of data items with distinct characteristics for these two dimensions. The colors denote the distance from the specified values for both dimensions, which is higher in the case

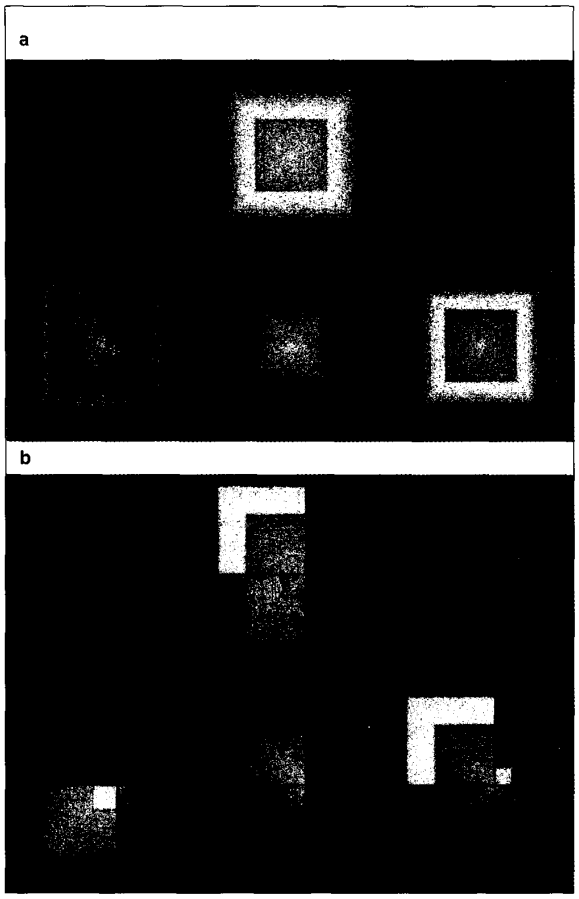


Figure 8. Five-dimensional artifically generated data items (100,000 data items): (a) basic visualization technique and (b) 2D arrangement.

of selection predicate 2, as indicated by the darker color (red).

Another interesting observation shows up in comparing the visualizations in Figure 7a and b and Figure 8a and b. The colored regions of the basic visualization technique often cluster in one quadrant of the 2D arrangement (for example, see the brown region for selection predicate 8 in Figure 7a and b). This provides additional information on the position of the cluster with respect to the two dimensions assigned to the axes. It might also help the user in modifying the query. Also interesting, but not easily identifiable in the printed version of our visualizations, are hot spots—that is, single exceptional data items in otherwise homogeneous regions.

Much of the information users get from the visualization is related to the data semantics. Due to space limitations, we do not elaborate on these aspects here. To do so would require introducing the schema and instances of the databases used. In the case of the artificial data sets, this would mean at least a specification of the base data set and all clusters.

The VisDB system is useful not only for such data mining tasks as finding hot spots, groups of similar data, and correlations among different dimensions. It also addresses such tasks as similarity retrieval and finding adequate query parameters and weighting factors. For example, in large CAD databases of 3D parts, it is not obvious how to formally describe similarity. Usually, there are many parameters describing the parts (in one real-world mechanical engineering application, we had 27 parameters), and each parameter might be important for a part to be similar. In CAD databases, you would issue a query searching for similar parts by using fixed allowances for some of the parameters. As a result, the query would get only the information concerning whether or not a data item fulfills all allowances. However, you could miss a part that exactly fits in all but one parameter. Therefore, in similarity retrieval, it seems important to provide approximate responses and let the user adjust the allowances and weighting parameters. Our system provides features that exactly support these tasks, making it a promising candidate for use in similarity retrieval.

Our system also helps find corresponding data items in multiple independent databases. If the user can define a distance function for the two attributes to be joined, our system may help identify closely related data items and find adequate parameters for approximately joining the databases.

Visualizing the results of complex queries

In addition to supporting simple one-table queries, where all selection predicates are connected by the same Boolean operator, our visualization techniques also support complex queries, such as queries with arbitrarily connected selection predicates (nested ANDs and ORs), multitable queries, and some types of nested queries (for details, see Keim, Kriegel, and Seidl[18]). VisDB uses multiple layers of windows for different parts of these queries. This gives users visual feedback for each part of the query and helps in understanding the overall result.

Multiple layers of windows are sufficient for queries with nested Boolean operators, but to support multitable and nested queries requires a mechanism for joining tables and dealing with the cross product. To support multitable queries, VisDB considers all data items of the cross product that approximately fulfill the join condition. The user obtains a separate window for the join condition wherein all data items of the cross product fulfilling the join condition are yellow and all others are colored according to their distance.

If tables are connected by foreign keys, it does not make sense to consider approximate results because the distances on foreign keys may not have any semantics. In such cases, VisDB considers only those data items that fulfill the join condition; it generates no visualization for the join condition. In many other cases, however, it is helpful to consider data items that approximately fulfill join conditions. For joins on numerical attributes, for example, we can use the numerical difference between the considered data items of the two relations as an approximation of the join condition to be fulfilled. In a similar way, we can determine the distances for non-equijoins ($a1 < a2$) or parametrized (non-equi)joins ($a1 - a2 < c$).

In the case of nested queries, VisDB provides separate visualizations for each selection predicate, including the subqueries involved. In the visualization corresponding to the overall result of a subquery, the user sees yellow when the subquery condition is fulfilled and otherwise the color corresponding to the distance of the data item most closely fulfilling the subquery condition. We determine the data item most closely fulfilling the subquery condition by the minimum distance resulting from an approximate join of the inner and the outer relation(s).

Instead of displaying a single value for the whole subquery, the system might give users the option of selecting a single data item and getting the complete subquery with all its selection predicates, including the join of inner and outer relation(s) presented in a separate window.

Implementation

The visualizations presented in Figures 5 through 8 are screen dumps from working with the VisDB system. We have implemented the system in C++/Motif running under X Windows on HP 7xx machines. The current version is main memory based and supports interactive database exploration for databases containing up to 50,000 data items (on HP 735 workstations). We find this performance very encouraging since we have not yet optimized our algorithms. The implementation runs into performance problems, however, when interfacing with current commercial database systems because they offer no access to partial query results and no support for incrementally changing queries. Nor do they use multidimensional data structures for fast secondary storage access.

We are currently working on improving the performance in directly interfacing with database systems. In the future, we plan to implement the VisDB system on a parallel machine that will support interactive query modifications even for midsize to large amounts of data and complex distance functions.

Future extensions

Inspired by our prototype, we have several ideas to extend the VisDB system. One extension is automatic generation of queries that correspond to some specific region in a visualization window. The user will identify the region graphically. The system will then find adequate selection predicates to provide the desired data items. Another idea is to generate time-series visualizations corresponding to queries changed incrementally. By changing the query, different portions of multidimensional space can be visualized, allowing even larger amounts of data to be displayed. 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. In addition to real-world data, we will also use artificially generated data sets that allow controlled studies on the effectiveness of our visualization techniques.

Conclusions

Visualization techniques can help researchers explore very large amounts of arbitrary, multidimensional data and find interesting data sets—hot spots, clusters of similar data, or correlations between different dimensions. Our approach to these "data mining" tasks combines traditional database querying and information retrieval with new data visualization techniques. The VisDB system can visualize at once the number of data values equal to the number of pixels on current displays, providing valuable feedback on the database query and helping users find results that would otherwise remain hidden. The system's interactivity lets users focus on interesting data.

We believe that query and visualization systems like ours are valuable for many applications. They may be the starting point for new visual solutions to problems that have proved very difficult. Querying large databases is just one example. ❑

Acknowledgments

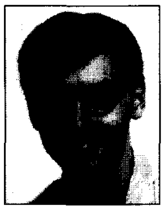
We want to thank all the people who contributed to the VisDB system, especially Thomas Seidl, who implemented the first prototype of the system, and Juraj Porada, who implemented the current version.

References

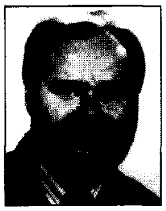
1. A. Motro, "Flex: A Tolerant and Cooperative User Interface to Databases," *IEEE Trans. on Knowledge and Data Engineering*, Vol. 2, No. 2, 1990, pp. 231-246.
2. D.A. Keim and V. Lum, "Gradi: A Graphical Database Interface for a Multimedia DBMS," *Proc. Int'l Workshop on Interfaces to Database Systems*, in *Lecture Notes in Computer Science*, Springer, London, 1992, pp. 95-112.
3. S.J. Kaplan, "Cooperative Responses from a Portable Natural Language Query System," *Artificial Intelligence*, Vol. 19, 1982, pp. 165-187.
4. A.K. Joshi, S.J. Kaplan, and R.M. Lee, "Approximate Responses from a Data Base Query System: Applications of Inferencing in Natural Language," *Proc. 5th Int'l Joint Conf. on Artificial Intelligence*, 1977, pp. 211-212.
5. G. Salton and M.J. McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill, New York, 1983.
6. J. Bertin, *Graphics and Graphic Information Processing*, Walter de Gruyer & Co., Berlin, 1981.
7. E.R. Tufte, *The Visual Display of Quantitative Information*, Graphics Press, Cheshire, Connecticut, 1983.
8. J. Beddow, "Shape Coding of Multidimensional Data on a Microcomputer Display," *Proc. Visualization 90*, IEEE Computer Society Press, Los Alamitos, Calif., 1990, pp. 238-246.
9. S. Feiner and C. Beshers, "Visualizing *n*-Dimensional Virtual Worlds with *n*-Vision," *Computer Graphics*, Vol. 24, No. 2, 1990, pp. 37-38.
10. A. Inselberg and B. Dimsdale, "Parallel Coordinates: A Tool for Visualizing Multidimensional Geometry," *Proc. Visualization 90*, IEEE CS Press, Los Alamitos, Calif., 1990, pp. 361-370.
11. R.M. Pickett and G.G. Grinstein, "Iconographic Displays for Visualizing Multidimensional Data," *Proc. IEEE Conf. on Systems, Man and Cybernetics*, IEEE Press, Piscataway, N.J. 1988, pp. 514-519.
12. R.D. Bergeron, L.D. Meeker, and T.M. Sparr, "Visualization-Based Model for a Scientific Database System," in *Focus on Scientific Visualization*, H. Hagen, M. Miller, and G. Nielson, eds., Springer, Berlin, 1992, pp. 103-121.
13. J. LeBlanc, M.O. Ward, and N. Wittels, "Exploring *N*-Dimensional Databases," *Proc. Visualization 90*, IEEE CS Press, Los Alamitos, Calif., 1990, pp. 230-239.
14. T. Mihalisin et al., "Visualizing Scalar Field on an *N*-dimensional

Lattice," *Proc. Visualization 90*, IEEE CS Press, Los Alamitos, Calif., 1990, pp. 255-262.

15. F. Marchak and D. Zulager, "Effectiveness of Dynamic Graphics in Revealing Structure in Multivariate Data," *Behavior, Research Methods, Instruments and Computers*, Vol. 24, No. 2, 1992, pp. 253-257.
16. B. Shneiderman, "Dynamic Queries for Visual Information Seeking," to appear in *IEEE Software*, 1994.
17. D. Bergeron, D.A. Keim, and R. Pickett, "Test Datasets for Evaluating Data Visualization Techniques," in *Perceptual Issues in Visualization*, G.G. Grinstein and H. Levkowitz, eds., Springer, Heidelberg, 1994.
18. D.A. Keim, H.-P. Kriegel, and T. Seidl, "Supporting Data Mining of Large Databases by Visual Feedback Queries," *Proc. 10th Int'l Conf. on Data Eng.*, IEEE CS Press, Los Alamitos, Calif., 1994, pp. 302-313.



Daniel A. Keim is a teaching research assistant in the Institute for Computer Science at the University of Munich, Germany. His research interests include visualization of statistical data, visual support for querying databases, database support for visualization systems, interfaces to database systems, and interoperability of heterogeneous databases. Keim received his diploma (equivalent to an MS degree) in computer science from the University of Dortmund in 1990. Currently, he is finishing his PhD in computer science.



Hans-Peter Kriegel is a professor for database and information systems in the Institute for Computer Science at the University of Munich, Germany. His research interests are in spatial database systems, particularly query processing, performance issues, and parallel operations. Data exploration and data mining in very large spatial databases led him to the area of visualization. Kriegel received his MS and PhD in 1973 and 1976, respectively, from the University of Karlsruhe, Germany.

Readers can contact the authors at the Institute for Computer Science, University of Munich, Leopoldstr. 11B, D-80802 Munich, e-mail {keim, kriegel}@informatik.uni-muenchen.de.

Reader Service Number 1