A FRAMEWORK FOR KNOWLEDGE-BASED, INTERACTIVE DATA EXPLORATION

Jade Goldstein, Steven F. Roth, John Kolojejchick, and Joe Mattis
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
(412) 268-8818
Jade.Goldstein@cs.cmu.edu, Steven.Roth@cs.cmu.edu

ABSTRACT

In this paper, we propose a framework that combines the functionality of data exploration and automatic presentation systems to create a knowledge-based, interactive, data exploration system. The purpose of a data exploration system is to enable users to uncover and extract relationships hidden in large data sets. The purpose of an automatic presentation system is to reduce the need for users and application developers to have graphic design expertise and to spend much time interacting with graphics packages to view their data. Previous work on data exploration was limited to query mechanisms that were often complex to learn and difficult to use, data manipulation mechanisms that did not provide complete coverage of the operations needed by users (especially the ability to form ad hoc groupings of data), and graphics that were restricted to a small set of predefined visualizations. Automatic presentation research, although addressing these issues, has been limited to the display of small data sets. This research has also not developed approaches to combine interactive, user-directed processes of design and data manipulation with automatic presentation mechanisms. We propose a framework that overcomes these limitations of current data exploration systems and integrates new interactive capabilities with automatic presentation components. This approach to supporting data exploration integrates recent work on SageTools, an environment for interactive and automatic presentation design, with a prototypical interactive data manipulation system called IDES. In this paper, we present our work on the IDES data manipulation capabilities and discuss requirements for coordinating them with automatic presentation of large data sets.

KEYWORDS: Data Exploration, Data Visualization, Intelligent Interfaces, Automatic Presentation Systems, Graphic Design, Computer-supported Design, Large Data Sets.

1. INTRODUCTION

The widespread use of databases and computers is requiring growing numbers of people to use and understand increasing amounts of information. These databases contain diverse data, including combinations of quantitative, temporal, categorical, hierarchical, geographic, and other types of information. Users of these large data repositories will not be limited to scientists and technically oriented professionals. Thus, there is a need for software that assists users with diverse levels of expertise in their data exploration tasks without substantial training and/or effort. The tasks users need to perform with information go beyond retrieving simple facts and answering focused questions. Instead, the tasks involve solving problems and making decisions based on the current state of the data, which is often repeatedly refined and extracted. These decisions depend upon the user's understanding of the relationships latent within the data. Once an interesting relationship is discovered, the user can use it to guide the next instruction to the system. This iterative and interactive process, which Brachman [5] calls data archaeology, is initiated and controlled by people. In contrast, data mining [10] emphasizes the use of automatic mechanisms to search for patterns.

We propose a framework for building a *knowledge-based interactive data exploration* system that will support the data archaeology process. Doing so requires understanding the interactive and iterative processes of data exploration. Specifically, tools must support three kinds of exploration subtasks:

1. *data visualization operations,* which include finding or designing and creating effective graphics.
2. *data manipulation operations,* which include selecting data for display, focusing on particular attributes of the data, and grouping or reorganizing data.
3. *data analysis operations*, which include statistical testing, summarization, and transformation for understanding properties of the data.

Of course, these subtasks are interdependent and overlapping. For example, a user may wish to select data directly from a visualization (a data manipulation operation performed on a visualization). Data exploration systems will need to employ other user interface techniques that provide easy interaction and communication with the components of the system that generate displays, such as direct manipulation [11]. They also need to employ query mechanisms that allow the user to focus on their data and results and not on the process of creating a query.

Data Visualization. Automating portions of the data exploration process requires specific knowledge relevant to each of the three exploration subtasks. For data visualization, research has focused on systems that can automatically generate a display of data composed of graphics and text or developing new techniques customized to specific tasks or types of analyses. The intent of

automatic presentation systems is to relieve users and application programmers of the need for graphic design knowledge and of the task of designing and specifying displays. This lets users concentrate on their goals for viewing information. Furthermore, for complex combinations of data, automatic presentation systems have the ability to generate graphics that users might not even consider. However, research on existing automatic presentation systems has been narrow, primarily focusing on representing knowledge of graphic design and not on interactive mechanisms for performing design. Recent work [19] has applied this technology to create computer supported data-graphic design tools, in which users can interactively specify and/or search and choose from a library of previously created graphics. This system, called SageTools, builds on an automatic presentation system called SAGE [16, 17, 18] and provides an approach to situating automatic technology in an environment which supports iterative and interactive data-graphic creation.

Previous automatic presentation systems have also been limited to the display of small data sets - those that fit well in a single computer window [6, 12, 14, 16, 17, 19]. These systems are unable to support many applications that use hundreds or even thousands of data elements. When dealing with such large data sets, it is no longer sufficient for the presentation system to create only a display. The system must also provide mechanisms for interactive manipulation and analysis of large data sets.

Data Manipulation. The functionality needed for large data set manipulation and analysis in an automatic presentation system is an extension of that needed in a conventional data exploration system. Thus, it is useful to first explore standard data exploration systems in which there are several popular approaches to providing interactive data manipulation techniques. Data base query systems, statistical packages and electronic spreadsheets provide various levels of support for accessing, modifying and reorganizing data. However, these are not well integrated with effective graphics techniques, nor do they provide flexible, low-effort tools that can be used by a broad cross-section of users. Query systems have great flexibility but require learning programming skills and lack convenient tools for organizing and summarizing information. Spreadsheets have greater intuitive appeal but lose the flexibility of query languages for selecting data. They also have a limited ability to rapidly reorganize information. Attempts to provide spreadsheets with these capabilities have often resulted in new programming environments rather than effective interface mechanisms.

In exploring the nature of data manipulation techniques, we classify data manipulation goals into three categories: *controlling the scope* (selecting desired portions of data), *choosing the level of detail* (creating and decomposing aggregates of data), and *selecting the focus of attention*

(concentrating on the attributes of data that are relevant to current analysis). We have used this classification to evaluate the functionality of existing data manipulation interface techniques.

Based on these results, we have expanded an interface mechanism called the Aggregate Manipulator (AM) [15] and combined it with Dynamic Queries (DQ) [1] in a prototype system called IDES (Interactive Data Exploration System). We use the results of our experience with IDES to propose extensions to SageTools to handle large data sets. The goal of these extensions and the integration of IDES with SageTools is a knowledge-based interactive data exploration system, a tool for users who are investigating relationships in large data sets.

In this paper, we concentrate on an overview of a framework for accomplishing this goal and the details of the IDES system. Section 2 discusses the proposed integration of SAGE and IDES. Section 3 discusses the data manipulation operations: controlling the scope, choosing the level of detail, and selecting the focus of attention. Section 4 explains how we selected the interface mechanisms used in IDES. Section 5 gives an overview of the design of IDES. Section 6 provides examples of how IDES is an effective tool for covering the data manipulation operations, due to the complementary nature of AM and DQ, as well as IDES's inherent flexibility in methods of exploring large data sets. This is demonstrated in two domains: shipping and real estate. Section 7 highlights the extensions required for SAGE as illustrated by the properties of large data sets in IDES.

2. FRAMEWORK FOR KNOWLEDGE-BASED INTERACTIVE DATA EXPLORATION

In this section, we propose a framework for a knowledge-based interactive data exploration system. Figure 1 shows the conceptual architecture that will be discussed in this section. The framework is composed of system modules, communication protocols, and knowledge of data and task characteristics and graphic design. Users communicate through direct manipulation interfaces, which generate appropriate directives to other components. The components access stored knowledge relevant to their functionality and the data. The following subsections will expand on these three concepts.

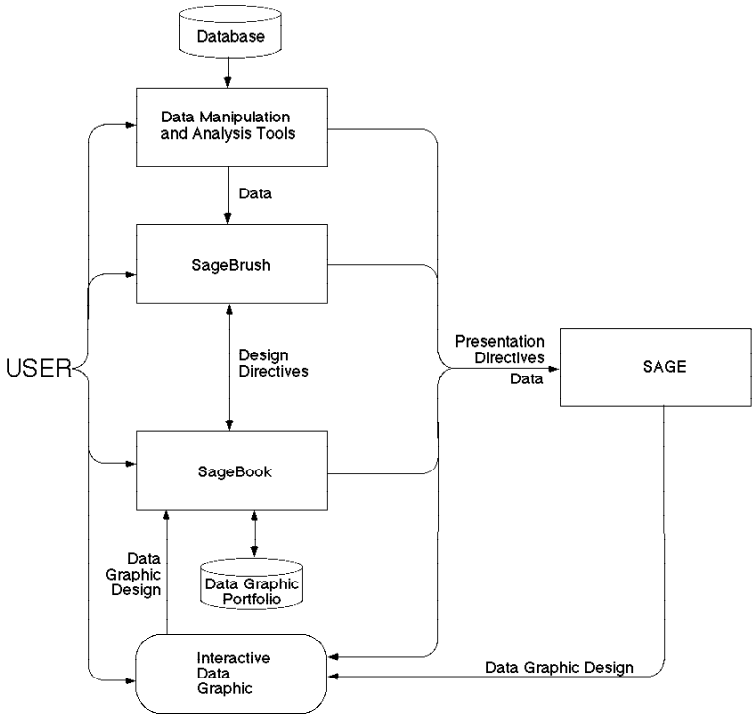


Figure 1: Proposed architecture for knowledge-based interactive data exploration.

2.1 System Components

All the components in the architecture provide one or more mechanisms for supporting data visualization, data manipulation, or data analysis. SAGE [16, 17, 18] is an automatic presentation system containing many features of related systems like APT, BOZ, and ANDD [12, 6, 14]. SAGE uses a characterization of data [16] to be visualized and a user's data viewing goals to design graphics. Design operations include selecting techniques based on expressiveness and effectiveness criteria and composing and laying out graphics appropriate to the data and user goals.

SageTools [19], an extension of SAGE, goes beyond previous presentation systems in several ways. SAGE can create graphics when users completely specify their designs (using SageBrush) as well as when they provide no specifications at all. Most importantly, it can accept *partial* specifications at any level of completeness between these two extremes and finish the design in a reasonable manner. User specifications generate *design directives*, which constrain the path of a search algorithm that selects and composes graphics to create a design. The ability to accept partial specifications is due to a rich *object representation* of the components of graphic displays, including their syntax (i.e., their spatial and structural relationships) and semantics (i.e., how they indicate the correspondence between data and graphics). The representation allows SAGE to produce combinations of a wide variety of 2D graphics (e.g., charts, tables, map-like coordinate

systems, text-outlines, networks). It also enables SageBook to support search for previously created pictures with graphical or data elements specified by users. A detailed discussion of automatic design capabilities can be found elsewhere [6, 12, 16, 17, 18, 19].

SageBrush is a tool with which users sketch or assemble graphical elements to create designs and map them to data. SageBrush provides users with an intuitive and efficient language for sketching their designs, then translates these sketches into a form that can be interpreted by SAGE. The assumption is that any interactive design interface that attempts to provide complete coverage of graphics will require a knowledgeable system behind it to be successful.

New designs begin with a user's selection of a partial prototype. As illustrated in Figure 2, SageBrush's interface (left window) consists of a design work area (center) into which users drag prototypes (top), graphemes (left), and data attribute names (bottom). *Prototypes* are partial designs, each with a spatial organization, graphemes, and/or encoders that commonly occur together. Encoders are frames of reference for interpreting properties of graphemes. For example, axis encoders enable us to interpret (i.e., derive a data value from) the position of a bar (a grapheme) in a chart (a spatial framework).

Prototypes are extended by adding graphemes and selecting properties of them to assign to data attributes (e.g., their color, shape, size, position, etc.). While the chart and map prototypes have no graphemes, dragging them into the design work area creates an *encoding space* which supports new design choices. The encoding space of a chart or map is defined by the interior of the two axes or coordinate-frame, respectively. Dragging a mark grapheme into a chart's encoding space results in directives to SAGE to include these grapheme types in the design, with their positional properties interpreted relative to the chart's coordinate system.

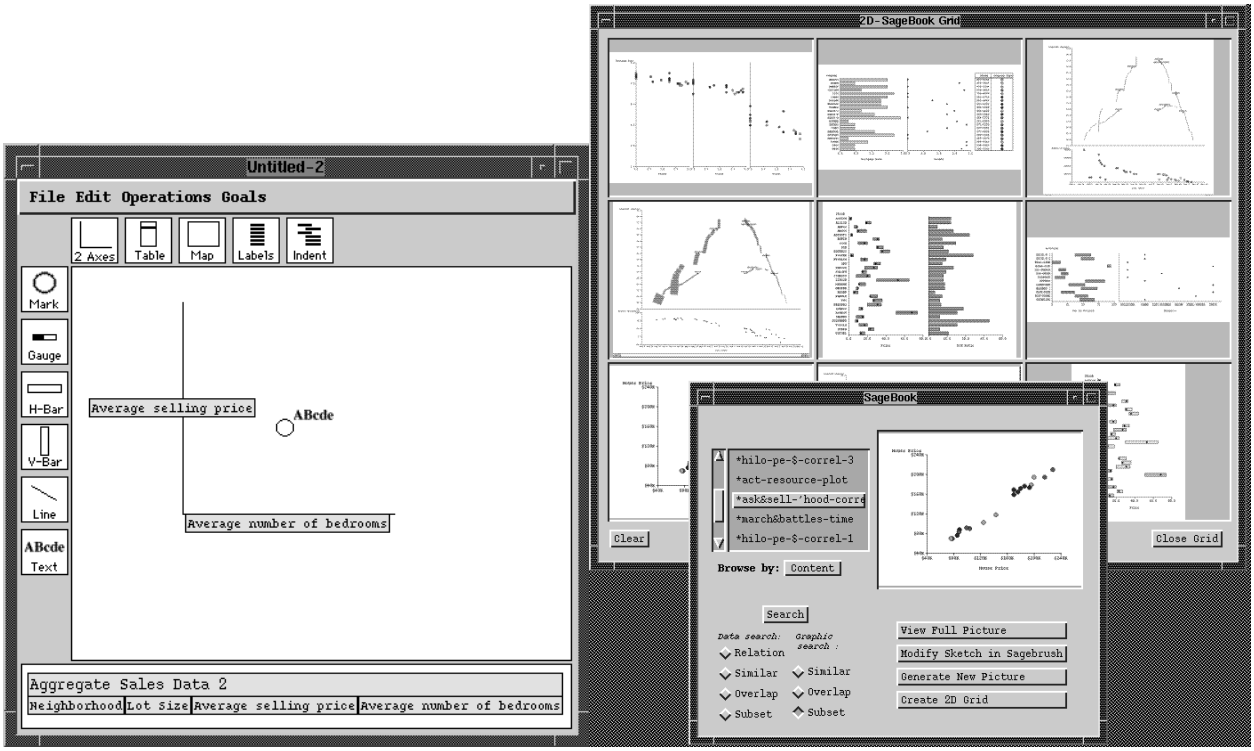


Figure 2: The SageBrush interface (on left) and the SageBook interface (on right)

SageBook (Figure 2, right windows) is a tool for browsing and retrieving previously created pictures (i.e. complete, rendered designs) and utilizing them to visualize new data. SageBook supports an approach to design in which people remember or examine previous successful visualizations and use them as a starting point for designing displays of new data. After selecting a visualization, users extend or customize it as needed. Our experiences in graphic design suggest that search and reuse of prior cases with customization is a common process. Therefore, our goal is to provide methods for searching through previously created pictures based on their graphical properties and/or the properties of the data they express. A picture found in this way can optionally be modified in SageBrush prior to sending it to SAGE, which creates a similar style graphic for the new data. SageBook capitalizes on the explicit representation of designs and data characteristics by SAGE to provide a vocabulary for expressing search criteria.

SAGE and SageBrush will need to be extended to handle large data sets. For example, SAGE will need to create visualizations that group portions of data to control level of detail (e.g., given screen constraints and user direction, SAGE might choose the picture in Figure 4b rather than in Figure 4a). Section 7.2 will briefly discuss how the syntactic and semantic representation of graphics and data characteristics will need be extended to create displays containing these data groupings. SageBrush will need to include new interactive graphemes, such as the aggregate gateway grapheme, the graphical representation for a grouping of data.

Most of all, SageTools will need a Data Manipulation component to provide a variety of interface mechanisms for grouping, filtering, transforming and performing other operations on data. Data manipulation operations are the focus of IDES, and Section 7 will discuss their role in the broader framework. Sections 3-6 explain the purpose and capabilities of IDES in detail.

2.2 Communication

Communication between the user and the components of the exploration system occurs via directives, which are messages that can result from user actions. Directives are generated through interactions with a variety of user interfaces, including menu commands, dialog boxes and direct manipulation techniques, such as sliders, button selection, and drag and drop mechanisms. There are several types of directive:

Presentation directives are the means by which users communicate their intent for the creation of graphics. There are four types of presentation directives: task, style, aesthetic, and data. Task directives are used to communicate a user's information-seeking goals (e.g., the data must be viewed very accurately). Style directives are display goals which affect the types of graphic techniques that are used to encode data (e.g., use saturation to show neighborhood as in Figure 4). Aesthetic directives influence the choice of *values* for graphical techniques (e.g. the color values, scale of axes, shape of objects). Data directives, discussed in subsequent sections, are the means by which the user communicates with the data manipulation and analysis component of the system. Following are some examples of each types of presentation directive.

Task Directives:

- indicate specific information-seeking goals for particular data attributes, e.g., looking up accurate values, detecting correlations, locating needs, scanning for differences.
- prioritize data attributes, e.g., the "cost" data attribute should be displayed more prominently than the "quantity" attribute, and the information-seeking goals for cost have a higher priority than the information-seeking goals for quantity.
- will need to include focus of attention and level of detail directives, because the choice of number of attributes and the groupings of data affect the choice of graphical techniques.

Display Directives:

- Style Directives: Convey combinations of graphical and textual techniques. For example, users may specify a display with two bar charts side by side, whose vertical axes are identical, or that they don't want to use color to encode a data attribute. Design directives will also need to convey interface objects to be included in displays to perform data manipulation.

- Aesthetic Directives: Inform the presentation system of the user's preferences for the appearance of the picture, e.g., use red, blue, and white, or draw the picture in a 3" x 5" window.

Previous automatic presentation systems did not provide interfaces for users to convey directives. In SageTools, task, style and aesthetic directives are generated through SageBrush and SageBook specifications. For an interactive data exploration system, we need to expand the task and display directives to include operations associated with large data sets. We also need to introduce data directives for communication with the data manipulation and analysis component of the system. Section 7.1 will briefly discuss the data directives.

2.3 Data Characterization and Design Knowledge

A knowledge-based interactive data exploration system must be application-independent, yet able to interpret data sufficiently in each new application. It is able to do this because of a general vocabulary for describing the data and task characteristics of domains. It uses these characteristics along with design knowledge to generate graphics [16, 17, 18, 19].

The data characterization is provided by an application developer or user and can be modified or updated by the user through a data characterization generator. It is based on an underlying vocabulary for describing the semantic and structural properties of the data (actual and derived) that are relevant to presentation design [4, 16]. A complete set of data characteristics enables appropriate mappings between the data and graphics. All current work on automatic presentation is founded on a strong definition of the data characteristics. Data characteristics include:

- *sets of data objects,* including distinctions among quantitative, ordinal, and nominal scales of measurement; whether objects represent coordinates or amounts, where a coordinate is a point or location temporally, spatially or otherwise (e.g., calendar-date) and an amount is a value not embedded in a frame of reference (e.g., weight or number-of-days); ordering conventions among objects, etc.
- *attributes of objects,* e.g., whether an attribute or relation maps from one object to one or more other objects; whether missing values are meaningful; the arity of a relation; etc.
- *higher order relationships* among attributes, e.g., the semantics underlying the relation between the beginning, end and duration of a time interval or the relationship between longitude and latitude in a two-dimensional spatial representation.
- *domain of membership*, which refers to whether data measures time, space, temperature, currency, mass, etc., which enables a system to preserve such standard conventions as time displayed along a horizontal axis, and East-West appearing right to left.

- *algebraic dependencies* among database elements, e.g., each bar in the stacked bar chart displays the sum of its parts.

The data characterization is interpreted by using application-independent design knowledge. Design knowledge contains two components: a library of presentation techniques and mechanisms for selecting and combining techniques. The library of presentation techniques consists of: (1) graphical and textual techniques for displaying the components of various kinds of tables, charts, maps and network diagrams; (2) information describing the types of data for which the technique is suitable (e.g., the graphical technique color is suitable for nominal data with six or fewer items); and (3) the syntactic or structural relations among elements used in graphics (e.g., axes, lines, points, labels). The presentation design knowledge contains information about which techniques best satisfy which goals, how techniques can be combined, and which combinations of techniques create the most effective presentation for the users' data, according to their information seeking and display goals. It also has knowledge about how to structure, organize, and lay out displays and their components. Extensions to this knowledge and to the data and task characterization language are needed to support large data set exploration and are discussed in subsequent sections.

This section provided an overview of the components of a framework for data exploration environments. It was based on prior research on SageTools, which was an approach to computer supported data-graphic design and IDES, an experimental data manipulation tool for large data sets. Our future research will involve extending SageTools to incorporate the features of IDES so that visualizations created with SageTools include appropriate data manipulation operations. Sections 3-6 will discuss the implementation and functionality of IDES, and Section 7 will discuss the extensions necessary for large data set functionality.

3. DATA MANIPULATION: CONTROLLING SCOPE, FOCUS OF ATTENTION, & LEVEL OF DETAIL

The exploration goals that a user will have are clearly task dependent. These goals are also dynamic, changing as the user views various data and displays. Data manipulation is one of the processes that users perform in data exploration. Springmeyer [20] performed an extensive empirical analysis of the processes that scientists use when performing data analysis. Her category “data culling” is most similar to that of data manipulation. We have analyzed the data manipulation process in detail for object-attribute data (data which consists of an object such as a house and several attributes for that object, such as selling-price and number of bedrooms) and

have identified three types of exploration operations: controlling the scope, selecting the focus of attention, and choosing the level of detail.

Controlling *scope* involves restricting the amount of data one wishes to consider. There are two ways this can be accomplished: (1) selecting a subset of values of a data attributes, such as only cities with a population over 2 million, or (2) disjunctively join subsets of the data, such as data for cities where the population is over 1 million (set 1) or cities which have a population between 2 million and 10 million and are in the Eastern US (set 2)

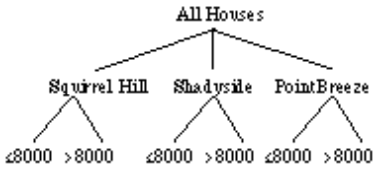
The second class of goals addresses *focus of attention,* which involves choosing the attributes of data one wishes to view in displays, to use for scope operations, and to use to control the level of detail. For example, a database of cars may consist of various attributes (car-model, year, company, cost, miles-per-gallon, repair-rating), but a user may wish to just focus on the average miles-per-gallon (for all car-models of a given company).

There is one specialized focus operation, the creation of *derived attributes*, which are attributes that do not occur in the original data and are defined by the user. For example, for our car data, we can create a derived-attribute called manufacturing-location (with values of American, European and Asian) by assigning a value to each car based on its manufacturer. The result is three groups, that can be displayed visually by coloring the cars on a display based on their manufacturer. Referred to as brushing [7] or painting [13], this technique control focus of attention using color. Another way to create derived attributes is to transform existing attributes by some filter [9], for example, create a binary attribute, fuel-efficient, from the car attribute miles-per-gallon by filtering the data by miles-per-gallon greater than 30.

The third type of goal is choosing the *level of detail*, which involves changing the granularity of the data that the user wants to examine, either by *aggregation* (combining data into meaningful groups) or by *decomposition*: (breaking a larger data group into smaller groups). The process of aggregation is sometimes referred to as *composition*, when the process involves combining two data groups. Suppose we have house-sale data with the following attributes: number-houses-sold, total-sale-price, and date, where date represents a day of 1992. This involves 365 data points. The user may wish to change the level of detail by grouping dates into months and displaying the total sales per month. This reduces a display of 365 data points to one of 12 (aggregation). On any resultant aggregate, users might want to do data analysis operations, which involve examining *derived properties* of the data. These include defining *summary statistic*s, which are statistics that can be computed on the values of attributes (e.g., sum or average), such as the average total-sale-price for 1992.

Decomposition involves reducing a data group into smaller groups based on the same or different attributes of the included data objects. Figure 3a gives an example of how decomposition could occur for a real estate sales database with the following attributes: house-selling-price, neighborhood, number-of-bedrooms, lot-size. The neighborhood attribute has the values {Squirrel Hill, Shadyside, Point Breeze}. The grouping of "All Houses" is decomposed by neighborhood into three sub-groups of houses, one for each neighborhood. Each neighborhood group is partitioned further by lot size: lot sizes less than or equal to 8000 ft. and lot sizes greater than 8000 ft. Figure 3b shows the representation of the house data in the aggregate manipulator's outliner (described further in section 6).

(a)



(b)

	count	average # of Bedrms	min-max # of Bedrms	average Lot Size	min-max Lot Size
AllHouses	292	4	2,8	4793	945,15730
Point Breeze	52	4	2,6	4911	1349,11543
under8000	47	4	2,6	4392	1349,7800
over8000	5	5	3,6	9789	8100,11543
Shadyside	47	4	2,6	3373	945,8784
under8000	46	4	2,6	3255	945,7500
over8000	1	5	5,5	8784	8784,8784
Squirrel Hill	193	4	2,8	5107	1066,15730
under8000	164	4	2,6	4172	1066,7975
over8000	29	5	2,8	10393	8050,15730

Figure 3: Decomposition Representations: (a) Decomposition Tree (b) Aggregate Manipulator representation.

We could also perform the same aggregation using data visualizations. For the same real estate data (as in Figure 3b), Figure 4a shows the scatter plot depicting houses, where the number of bedrooms attribute is on the x-axis and price is on the y-axis. If we aggregate all the individual data points by neighborhood, we obtain the scatter plot in Figure 4b which consists of three data groups, each of which is plotted using the average bedroom and average price. Figure 4c shows the plot where each neighborhood aggregate has been decomposed into two lot-size partitions (under8000 and over8000).

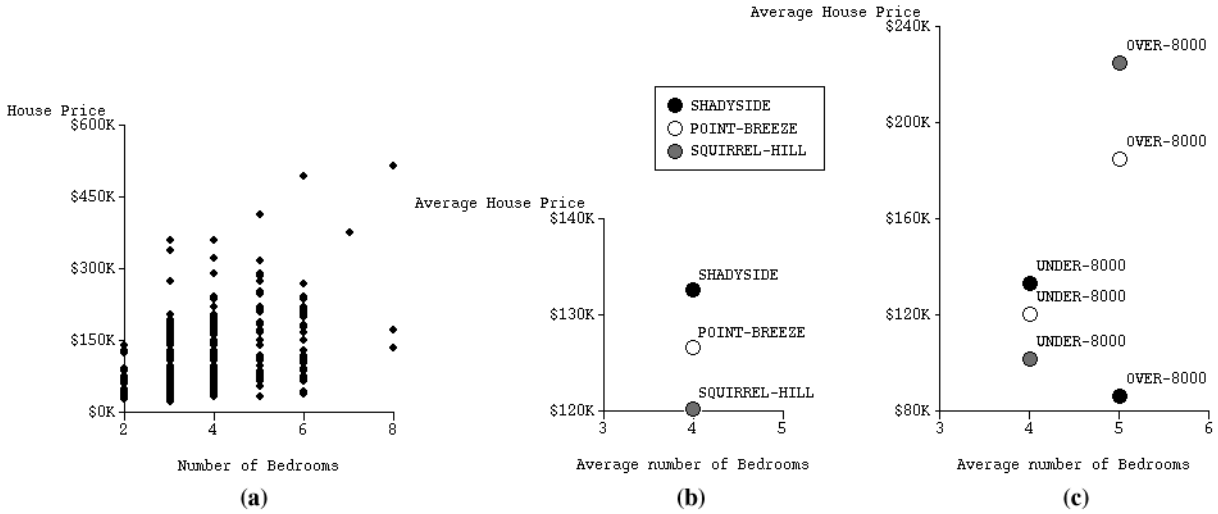


Figure 4: Example of Aggregation in Visualization. (a) The raw data. (b) Aggregating the raw data by neighborhood. (c) Decomposing the neighborhood aggregates by the lot-size attribute.

As we can see from Figure 3a, the process of decomposition forms a *hierarchy*, which structures data into meaningful groupings specified by the user. We have identified four classes of decomposition:

- user-defined or pre-defined natural groupings: These can be defined interactively by the user or in advance as built-in knowledge. A possible pre-defined natural grouping is time, e.g., years -> quarters -> months. An example of a user-defined grouping is data on crimes, where each crime data object has a date attribute. An analyst may decide to break the year into holiday days and non-holiday days (as illustrated in Figure 5a).
- element frequency divisions: Divisions are computed by the system to have the same number of elements (e*qui-frequency*). For example, if the user wants 20 divisions (partitioned by time) of the 1000 crimes committed in 1991, then there will be 50 crimes per time interval and each time interval can have a different size (Figure 5b).
- set interval divisions : Divisions are computed by the system to have the same interval size (*equi-interval.*). In the above example, if the user wants weekly divisions of the data, the system would divide the data into weeks: 1/1-1/7, 1/8-1/14, etc. (Figure 5c).
- system-provided statistical methods: The system can use clustering statistics or other methods to partition the data into groups.

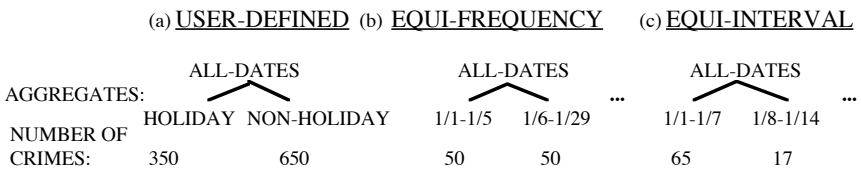


Figure 5: Types of decomposition.

4. SELECTING INTERFACE MECHANISMS

In the last section, we discussed a categorization of data manipulation operations - methods of selecting, grouping, and transforming data. In previous work [8], the authors have evaluated data exploration software according to this classification and discussed their advantages and disadvantages; the results are summarized in Table 1.

Dynamic Query or Queries (DQ) is an interactive technique which allows the user to manipulate sliders to control the amount of data displayed [1]. Each slider as shown in Figure 11, corresponds to a data attribute. The Aggregate Manipulator mechanism of Webs [15] was designed for level of detail operations, that of aggregation (grouping) and decomposition (partitioning groups), along with the display of group summary statistics. Iconographer [9] uses directed graphs that are programmed visually by the user to control scope, level of detail, and focus of attention operations. Powerplay [3] allows decomposition in pre-defined hierarchical structures. Excel allows level of detail operations through an "outline" mechanism in which the user can create groupings of data sets through a cumbersome process of individually linking cells in the database. SQL and other database query interfaces provide the most expressive means for specifying control of scope, but require learning complex programming languages to perform these and other operations.

DATA MANIPULATION OPERATIONS		Dynamic Query	Aggregate Manipulator	Iconographer	Powerplay	Excel
SCOPE	- filter data using attribute(s)	xx	x	x		
	- select multiple disjunctive subsets		xx	x		x
FOCUS OF ATTENTION	- select attribute(s) for viewing operations	xx	x	xx	x	x
	- select attribute(s) for level of detail operations		xx	x	x	x
	- derive attribute(s) from existing attributes		xx	x		x
LEVEL OF DETAIL	- predefined aggregation & decomposition		xx	x	x	x
	- flexible aggregation & decomposition		xx	x		x

Table 1: The data exploration operations provided by different software or techniques. If the software (technique) allowed the operation in a simple, straightforward manner, we assigned the value "xx". If the operation involved non-intuitive operations, lots of steps, or steps bordering on programming, we assigned the value "x".

Table 1 suggests that the aggregate manipulator provides complete coverage of desired data manipulation operations. However, the AM does not perform the scope operations of filtering

data or selecting attributes for viewing operations as well as DQ does. For filtering data, the AM requires creating user-defined partitions, which might have to be re-created for a slightly different choice of data (e.g., if users decide they want to view data for houses which sold for $125,000-$200,000 instead of $100,000-$200,000). Furthermore, if the user partitions the data set several times, it can be confusing what portion of the data (i.e., what range of values for the various attributes) is being displayed. In the case of DQ, determining these values is straightforward, since each attribute has its own slider or selector mechanism. However, DQ does not have the ability to disjunctively combine sets (e.g., display houses which sold for $50,000-$100,000 in the neighborhood Squirrel Hill and those that sold for $50,000-$150,000 in Shadyside) without creating methods that have multiple sets of dynamic queries linked to the same display. Thus, there is a need for a mechanism such as the AM to perform this and other operations (e.g. displaying summary statistics). Examples of the interactions between DQ and the AM will be given in Section 6.

In order to integrate AM and DQ in a prototype for exploration of large data sets IDES [8], we needed to extend them to function for many types of data. For the AM, this required exploring the types of operations that users would want to do with their data and then extending the AM so it could perform these types of decompositions and summary statistics based on a general data characterization rather than application-specific mechanisms. For DQ, we needed to be able to create a slider on demand and to have a method to select elements of nominal (non-numeric, unordered elements) data rather than just ranges of quantitative (numeric) data. For nominal data we use a scrolling list of elements (see Figure 11) and allow the user to select multiple elements of the list. Since the combination of these new versions of AM and DQ is not data specific, it is easily generalized to any new object-attribute data set.

5. SYSTEM DESIGN

The main functionality and dataflow of IDES is summarized in Figure 6. Decoupling the display area and the AM (which were linked in Webs) has the advantage that the user can explore and manipulate the data in the display area or the AM without affecting the other workspaces. This allows maintaining multiple perspectives on the data at different levels of detail. However, this has the drawback that aggregates that appear in the AM may or may not appear in the Display Area and vice versa. Whether or not this causes problems for users will be evaluated in user studies.

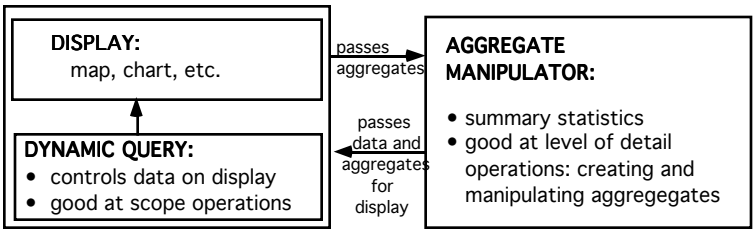


Figure 6: Data Flow for the AM & DQ.

The AM, DQ, and display comprise three of the four IDES workspaces (see Figure 11). The AM is a workspace for creating, decomposing, and directing the display of aggregates in other areas. The Display Area is both a work area for creating aggregates and a place to view the elements of the aggregates created by the AM. Dynamic query sliders are always connected to the current display. Changing the sliders changes the portion of the data that is displayed. Above the Display Area are menus that allow the user to create an aggregate, show an aggregate, clear the Display Area, and perform related functions. New aggregates can be created in the Display Area by selecting data points (represented by icons or graphical symbols) individually or as groups. Selected icons can be composed into a new aggregate, by the "Create Aggregate" command from the Display options. The user gives the aggregate a name, and the points representing the individual data objects are replaced by an icon representing an the aggregate data object, which we call an *aggregate gateway*. Users can move aggregates into the AM to be worked on further and optionally transferred back to the Display Area. If the aggregate is selected in the Display Area and a corresponding aggregate exists in the AM, both are highlighted. Lastly, users can decompose an existing aggregate into its components in the Display Area by double-clicking on the aggregate gateway object. The number of objects displayed reflects the bounds of the existing dynamic query sliders. This change in detail is not mirrored in the AM.

The last workspace consists of the data detail area (lower right corner), which is used for showing selected attributes of subsets of aggregates or individual data objects.

6 APPLICATIONS

We have implemented this design for a shipping domain and a real estate domain. We will show advantages of using multiple visualizations, aggregation of data, and iterative processing to allow the user to find answers to data exploration questions. We will also show the usefulness of the AM for combining groupings of data, in particular joining sets disjunctively, and we will show the advantages of combining both DQ and the AM mechanisms.

6.1 Shipping

Consider a scenario in which the U.S. government is sending emergency supplies worldwide. Transportation planners use a database of shipments, where each record represents a description of a single order to be transported. An order has the following attributes: shipment ID, origin port, destination port, mode of transportation (air/sea), priority, quantity (shipment weight in tons), scheduled arrival date (FAD), due date (LAD), and lateness (FAD - LAD). The initial state of IDES is a scatterplot display (Figure 7 shows the scatterplot after some operations have been performed) and a single aggregate of all individual records, called ALL-SHIPMENTS, in the aggregate manipulator. All points above the diagonal line in Figure 7 are late. We want to know which destination seaports have large quantities of high priority late shipments, so that we can send additional personnel and equipment to assist with the situation.

We first want to display the number of items in ALL-SHIPMENTS and the total quantity (in tons) of shipments in the AM. We obtain these values by pressing on the top column header area of the AM table and selecting the summary statistic "count". This gives the total number of items, i.e. 265 (Figure 7). For the second column we select the summary statistic "total", then obtain a list of the possible attributes that can be used with "total" and from this list select "quantity". This procedure could just as easily have been done in reverse, choosing the attribute "quantity" and then selecting from a list of possible summary statistic options. Because the system has built-in knowledge data characteristics for each attribute (refer to Section 2.3), the choices are limited to those appropriate for the attribute. For example, since "total" is not a valid summary statistic for the attribute "due-date" (dates are coordinates rather than quantities and hence can't be totaled) it wouldn't be on the list if the "due-date" attribute were already chosen.

We then decompose the ALL-SHIPMENTS aggregate by mode, by pressing on the ALL-SHIPMENTS aggregate in the aggregate manipulator, which gives a pop-up menu of attributes, the decomposition options. We choose "mode" which creates two aggregates, AIR and SEA (Figure 7). These new aggregates are indented from the initial group ALL-SHIPMENTS in the *outliner* (the leftmost column) of the AM. We want to display only the sea shipping data, so we clear the display, select and move the SEA aggregate from the AM into the display We then double click on the SEA aggregate (i.e. gateway) in the display area to display the individual shipment data.

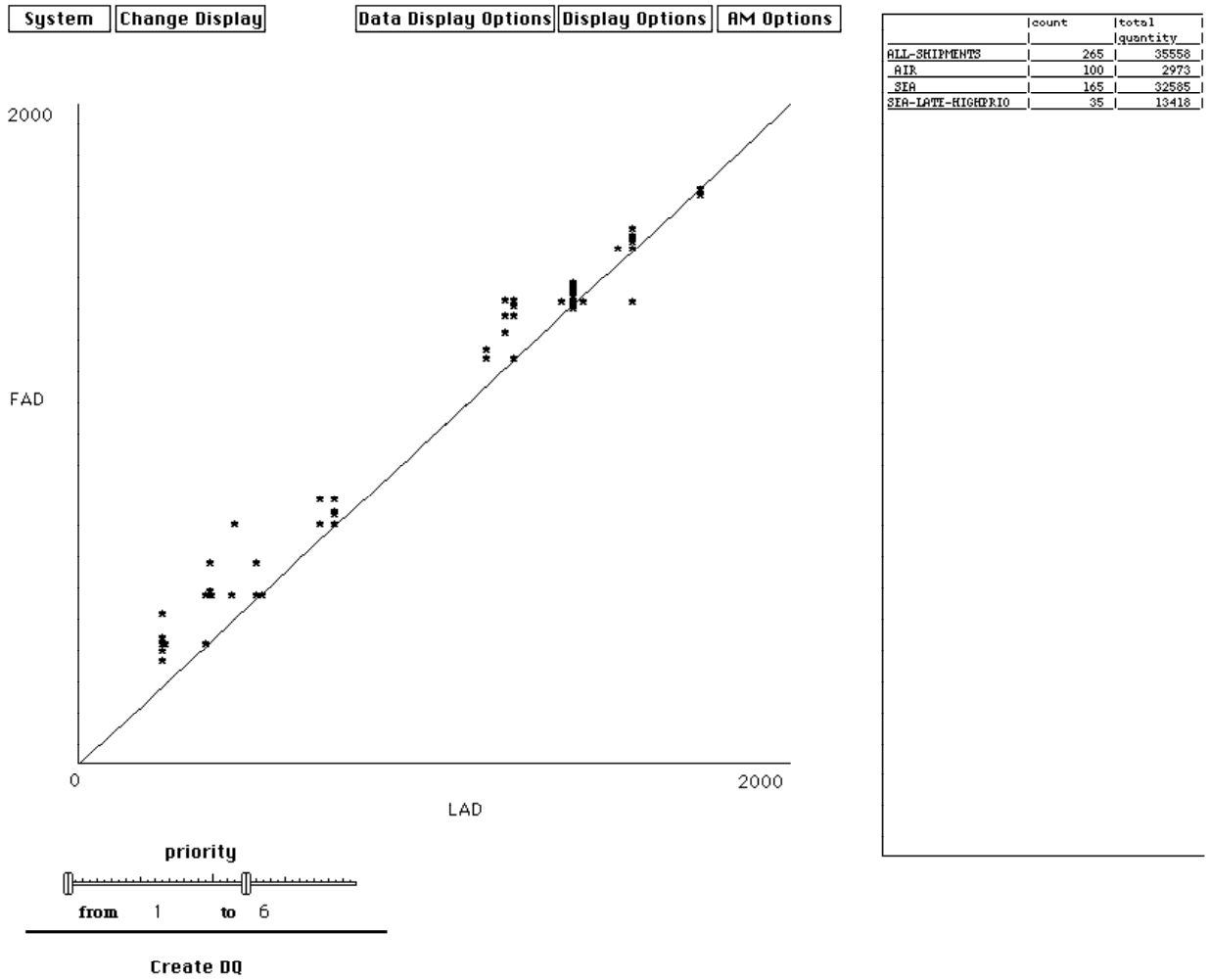


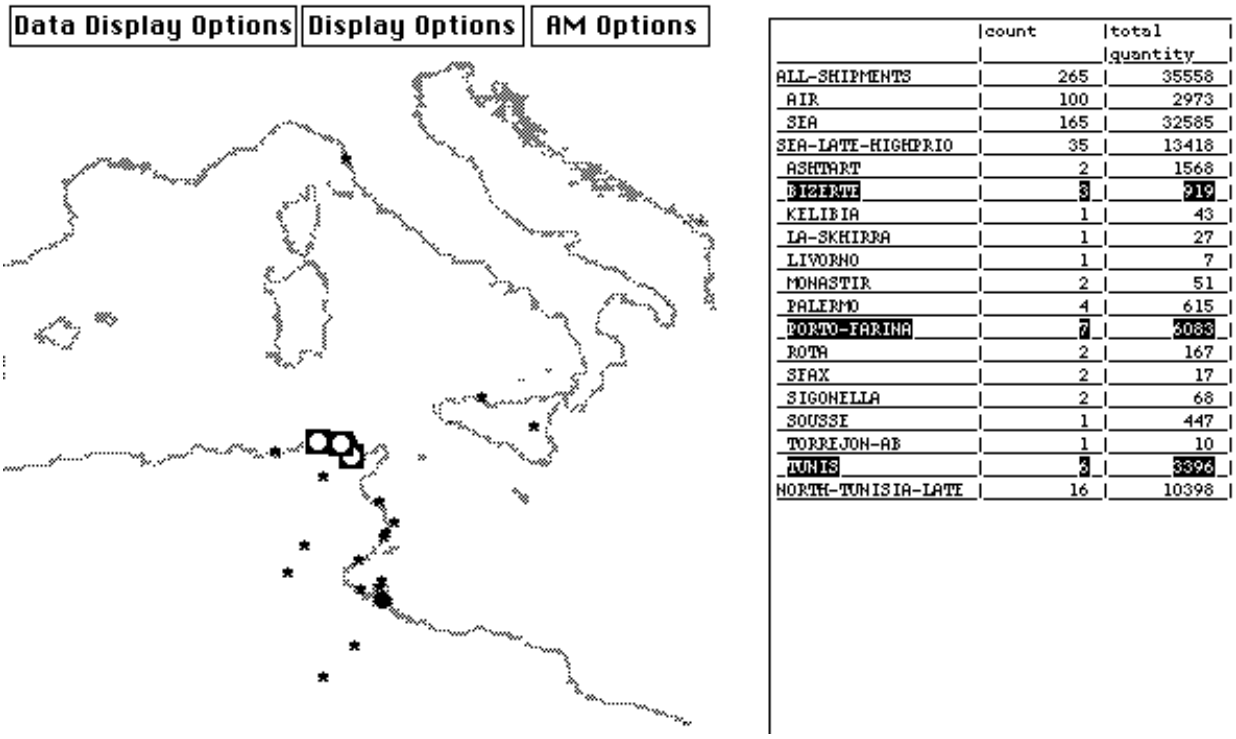
Figure 7: Scatterplot display after a series of operations. In the upper right hand corner is the outliner portion of the AM and in the upper left hand corner are the option buttons. Pressing on an option button gives a pop-up menu of available commands.

We now create a slider bar for priority by pressing on "Create DQ" in the DQ area. We choose the attribute "priority" from the list of options and use the resulting slider to limit the shipments displayed to those with priority 1 to 6. Figure 7 shows the results of these operations. Additional sliders for other attributes could also be created.

We decide to focus only on the first cluster of the two late shipment clusters (the lower left region of the display in Figure 7). We select all shipments above the diagonal line using bounding boxes . When we have the desired group, we choose the command Create Aggregate from the Display Options and name the aggregate SEA-LATE-HIGHPRIO. We then display this

aggregate in the aggregate manipulator and decompose it by all values of the attribute destination port (Figure 8).

We change to the map display, then select the four aggregates in the AM that represent ports with the largest total quantity of late shipments. We display them on the map (their representation is a larger graphical object than the individual shipments) to see their location. We notice that three of them are in North Tunisia, and we select those three aggregates on the display, which highlights (selects) them in the AM. We then create an aggregate in the AM and name it NORTH-TUNISIA-LATE. Figure 8 shows the results of these operations. We can see the total quantity (in tons) of late shipments, which gives us an idea of the extent of the problem and what resources we might need to allocate to the situation.



	count	total quantity
ALL-SHIPMENTS	265	35558
AIR	100	2973
SEA	165	32585
SEA-LATE-HIGHPRIO	35	13418
ASHTART	2	1568
BIZERTE	3	919
KELIBIA	1	43
LA-SKHIRRA	1	27
LIVORNO	1	7
MONASTIR	2	51
PALERMO	4	615
PORTO-FARINA	7	3083
ROTA	2	167
SFAX	2	17
SIGONELLA	2	68
SOUSSE	1	447
TORREJON-AB	1	10
TUNIS	3	3396
NORTH-TUNISIA-LATE	16	10398

Figure 8: The map display and AM after creating an aggregate NORTH-TUNISIA-LATE out of the ports in North Tunisia with the largest weight of late shipments (Bizerte, Porto-Farina, and Tunis).

This example has shown the value of multiple visualizations for aggregate creation and decomposition to support the user in the exploration process. Formulating SQL-like queries for this situation would be difficult because the user is unaware at the beginning where the problem destination ports are located and whether or not these ports are close enough geographically to be grouped together. For example, the set of shipments represented by the NORTH-TUNISIA-LATE aggregate would have been retrieved by a query like: "the set of shipments, whose mode

is SEA, priority is between 1 and 6, lateness is positive, LAD (i.e. due-date) is ‘among the lowest values’, and which are delivered to ports where ‘there are a lot of late shipments and some of which are close geographically’."

One of the features of the AM is its ability to create (compose) an aggregate by just selecting and combining multiple aggregates in the AM. For example, suppose we are interested in the combination of high-priority (<4) sea shipments and all air shipments (we assume air is high priority). We can decompose SEA into different priority levels (or choose only one priority level if we so desire) and then compose the high-priority sea aggregate (SEA-HIGHPRIO<4) with the aggregate representing all air shipments (AIR) in the AM (see Figure 9) to form a new aggregate (HIGH-PRIO). Composed aggregates, such as HIGH-PRIO are placed in the outliner one step to the left of their leftmost child aggregate to avoid confusing them as parts of other aggregates.

Since AIR and SEA are distinct categories their combination contains no overlapping elements. Suppose we decide that North Tunisia is a high priority area and thus any shipment with a destination port there is considered high priority. We could aggregate all shipments in North Tunisia on the map, name it NORTH-TUNISIA-PRIO and move the aggregate to the AM. We then compose NORTH-TUNISIA-PRIO with HIGH-PRIO to form INTEREST-SHIPMENTS (Figure 9). There may be overlapping shipments between these two sets. The system is aware of any overlapping shipments and makes sure that the resulting aggregate references each appropriately. Notice that the count for the INTEREST-SHIPMENTS aggregate is less than the sum of HIGH-PRIO and NORTH-TUNISA-PRIO.

	count	total quantity
ALL-SHIPMENTS	265	35558
AIR	100	2973
SEA	165	32585
SEA-HIGHPRIO<4	64	16050
HIGH-PRIO	164	19023
NORTH-TUNISIA-PRIO	109	16067
INTEREST-SHIPMENTS	206	24708

Figure 9: Combining aggregates.

Note that the integration of AM and DQ supports the formation of disjunctive queries which are impossible with DQ alone. Forming complex queries in the aggregate manipulator is intuitive -- users do not even have to recognize that they are creating queries. Such a straightforward mechanism helps the users to concentrate on their data and goals, rather than on the formation of queries.

These examples have shown how the AM is a useful mechanism for discovering properties of information in a large data set and how the use of various visualizations can assist the user in this process. In the real estate example, we will elaborate on how AM and DQ synergistically operate in our system and how they are useful for different purposes.

6.2 Real Estate

Our real estate data consists of attributes from an actual real estate database of houses sold. There are 27 attributes (Figure 10) with varied data types: quantitative (e.g., selling price), nominal (e.g., neighborhood), and interval (e.g., date of sale). The attributes of the house data have three natural hierarchical relationships. City can be decomposed into neighborhoods or zip codes. Companies can be decomposed into offices (selling or listing), and offices can be decomposed by agents. There are many possible user-defined partition options as discussed in Section 3.

• address	• number of rooms	• lot size	• selling price	• selling office
• neighborhood	• number of bedrooms	• living room size	• asking price	• listing office
• city	• number of bathrooms	• dining room size	• date of sale	• listing agent
• zip code	• style of house	• kitchen size	• assessment	• company
• age of house	• fireplace	• master bedroom size	• tax	• days on market
• type of house	• garage			

Figure 10: Attributes of the real estate data set.

In this section, we will discuss two scenarios. The first shows the weaknesses of using the AM alone. The second scenario shows how using DQ alone would require substantial work for the user. For both these cases, we show how using the combination of the AM and DQ is most effective.

Consider the following scenario. Jennifer is new to the Pittsburgh area and has the following goals for a house: (a) in the price range $100,000 to $150,000, (b) a lot size of at least 5000 sq. ft., because she wants a nice back yard, (c) at least 4 bedrooms, and (d) close to Carnegie Mellon University (i.e. in the neighborhoods of Shadyside, Squirrel or Pt. Breeze).

Jennifer would like to see houses which match these criteria and their map locations. The initial state of the system has the aggregate “AllHouses” in the AM and all houses displayed on the map (Figure 11). First, Jennifer creates dynamic query sliders for the attributes Selling Price, Lot Size and Neighborhood. After she selects the appropriate ranges or values for these queries, the map displays houses in Shadyside, Squirrel Hill and Point Breeze, with a price between 100-150K and lot size over 5000 sq. ft. Jennifer then selects all the data on the map and creates a new aggregate “Sq-Shady-PB”.

Note that if Jennifer wanted a group with more or fewer houses, she could change the sliders until she had approximately the number she desired. This is an awkward procedure in the AM

because it requires creating a new user-defined partition for each revision and then looking either at the summary statistics or displaying the new partition on the map.

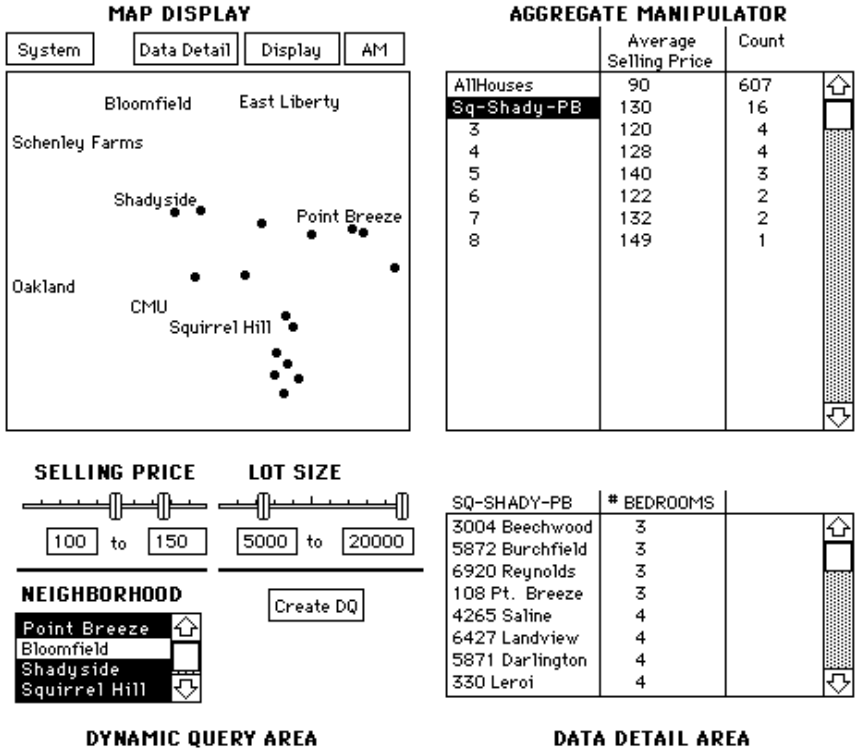
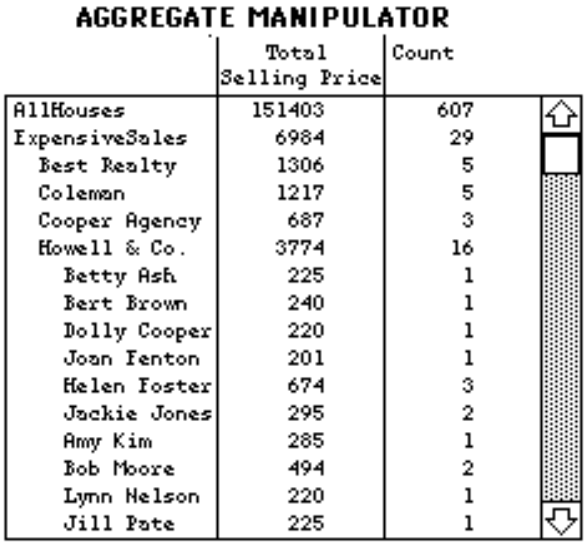


Figure 11: A readable representation of the interface as a result of using the AM and DQ to partition house data. There are four workspaces with various options accessible via pop-up menus from the buttons in the upper left hand corner.

Figure 11 shows the results of the operations and summary statistics and data for the aggregate "Sq-Shady-PB". To partition "Sq-Shady-PB", she presses on it in the outliner of the AM, which gives a pop-up menu of attributes and selects the attribute "# Bedrooms." She chooses to partition the attribute "# Bedrooms" into individual values. All non-empty aggregate groups are displayed in the AM (in this case 3-8) along with summary statistics for any specified columns (in this case Average Selling Price and Count). To get the display in the Data Detail Area, she selects the aggregate "Sq-Shady-PB", uses an AM option to move it to the Detail Area and then chooses the attribute "# Bedrooms" from a column header pop-up menu in the same manner as that for the summary statistics.

The second scenario involves another situation in which we show the combination of the AM and DQ is superior to either method alone. John wants to sell his house and is looking for possible real estate agents. He believes his house will sell for around $250,000. He wants to know which company and then which sales agent has sold the most houses in the price range

$200,000-$300,000 in his neighborhood in the last year. DQ alone is quite awkward to use because John would have to select all combinations of company and sales agents and then count the number of houses that appear on the map. However, DQ is easy to use for simple selection of the neighborhood and ranges for the price and date. From this he creates an aggregate ExpensiveSales (Figure 12). He then partitions this aggregate by the attribute company and selects two summary statistics: "Total" (for the attribute "Selling Price") and "Count". After finding that Howell & Co. sold the most houses, he decomposes this aggregate of houses by sales agent. The result of this decomposition is that John can quickly see that Helen Foster sold the most houses.



	Total Selling Price	Count
AllHouses	151403	607
ExpensiveSales	6984	29
Best Realty	1306	5
Coleman	1217	5
Cooper Agency	687	3
Howell & Co.	3774	16
Betty Ash	225	1
Bert Brown	240	1
Dolly Cooper	220	1
Joan Fenton	201	1
Helen Foster	674	3
Jackie Jones	295	2
Amy Kim	285	1
Bob Moore	494	2
Lynn Nelson	220	1
Jill Pate	225	1

Figure 12: The AM as a result of decomposing the aggregate ExpensiveSales by the attribute company and decomposing the partition Howell & Co. by the attribute sales agent.

These examples have shown how AM and DQ synergistically operate in our system and how they are useful for different purposes. The process of aggregation provides control over the level of detail of data shown. Slider mechanisms allow rapid filtering of the data. We will now discuss how these techniques can be used in our framework and how the components, knowledge, and directives modules need to be extended.

7. EXTENSIONS REQUIRED FOR LARGE DATA SETS

The IDES system and the examples above show how effective data manipulation tools can assist users in the data exploration process. For a computer-supported data graphic design system to handle these capabilities, the directives and knowledge must be extended and a data manipulation and analysis tool must be provided. In addition, we propose adding a data characterization

generator, which allows the user to specify their own hierarchical decompositions. This section will discuss these extensions.

7.1 Presentation Directives

As discussed in section 2.2, presentation directives are the means by which users communicate their intent for the creation of graphics. In order to support interactive data exploration, presentation directives must be extended to allow users to communicate their data exploration intent as well.

Additional data directives must be generated when users communicate with the data manipulation and analysis component of the system to control their data. Examples of data directives include specifying the scope of the data to be displayed and specifying desired summary statistics for a group of data.

Task directives must be extended to allow users to specify when control over scope, focus of attention or level of detail is needed in a visualization. For example, to allow control over focus of attention a directive specifying the users need to actively filter visualized data based on values of a specified attribute could be generated. Such a request might result in the design of a visualization with built-in dynamic query sliders.

To allow control over level of detail, a directive might allow users to specify a particular data hierarchy to be used when data needs to be aggregated. For example, the user might specify when presenting data on people that if there are too many individuals to display, the system should display groups of people by income, age or other breakdown which accomplishes a directive (like minimizing overlap or achieving better distribution of attribute values). Such directives must be expressed in a vocabulary that is sufficiently general and atomic to enable their use in interface mechanisms which control the addition and removal of visible data as part of the process of controlling scope or expanding aggregates.

7.2 Data Characterization for Aggregates

Exploration of large data sets requires two features with which we have experimented in IDES: aggregates and hierarchies. Aggregates represent groupings of data and abstractions or summaries of their attributes. Hierarchies help define and organize aggregates. Each aggregate data object has an associated set of data elements, a data characterization that describes the set, and representative data values for each attribute based on global properties of the set. A system can use representative values and display them using aggregate gateways, which are the graphic

representation of aggregate data objects that enable interactive expansion to greater detail. A system selects representative values of attributes and visualizes them as gateways appropriately is due to its knowledge of aggregate data characteristics.

An aggregate data object is an abstraction and grouping of similar individual data objects and its characteristics can be derived partly from those of the data elements from which it is composed. For example, an aggregate of shipment data objects will express abstractions of shipment attributes: weight, port, and date. Weight is a quantitative attribute that can be summarized by a mean, mode, total, range, or a distribution frequency (i.e. number of elements with each value or the number of different values). In contrast, port is a nominal attribute, which can only be summarized with a distribution-count (i.e. the frequency of different values in the set, or possibly just the most frequent value). Dates are interval attributes, which can be represented with the same summary statistics as quantitative attributes except total.

Knowledge of data characteristics can also guide the choice of an attribute to partition data elements into aggregates, independent of the manner in which their attributes are summarized. For example, nominal attributes can be used to group data objects which have the same value (e.g. shipments destined for the same port). Quantitative and temporal attributes can be used to group data objects with common values, but they can also group data based on intervals (i.e. ranges). We discussed the use of equi-frequency and equi-interval methods of grouping data into aggregates in the section on IDES.

These observations enable a knowledge-based system to automatically select (or support user selection of) attributes and values to define groupings to create aggregates when the level of detail must be reduced. They also enable a system to select appropriate graphic techniques to visualize groups and the representative values which summarize their attributes. For example, they enable a system to select bar lengths for totals or medians of quantitative attributes but not for the *medians* of sets of dates - a median of a set of dates is still a date and must be expressed as a coordinate rather than an amount [16]. Similar knowledge enables interval bars to be selected to express quantitative or temporal ranges.

In addition to knowledge of aggregate characteristics derived from element data, a data exploration system requires knowledge of strategies for hierarchically structuring data aggregates. There are three types of knowledge of hierarchical relationships:

- Domain independent: knowledge of universal hierarchical relationships, e.g., that days are grouped into weeks or calendar months, and then into years.

- Domain dependent: knowledge specific to an application, e.g. that dates in a college's academic year can be aggregated by months with semesters (Fall, Spring and Summer semesters); dates in a business calendar might be aggregated by quarter (Jan-Mar, Apr-Jun, etc.).
- User-defined: knowledge that a user provides for a particular data set. For example, for population data, the user may want to use a new partition of the states which divide the states into coastal and non-coastal ones or alphabetically organized groups.

Just as a system can choose attributes to group data elements and summarize other attributes, it can use hierarchical structures like these to control the level of detail. Hierarchies provide intuitive methods for moving flexibly across levels of detail, enabling users to control aggregation and decomposition and keep track of the relationships among aggregates at different levels (i.e. parents and siblings in a tree).

A mechanism which must be added to support manipulation is a data characterization builder. Such a tool would enable users to interactively create and store new hierarchical structures for grouping data and controlling level of detail (e.g. to provide the ability to create the holiday/non-holiday breakdown of all-dates depicted in Figure 5a). It updates the data characterization so that the presentation system will be aware of the new hierarchy for aggregation purposes. The ability to quickly and easily create hierarchies allows users to group their data in partitions they want to explore repeatedly and in ways that are meaningful for specific instances of data.

7.3 Extensions to the Design Knowledge

Besides being used to display data, the IDES display area can be used to create and manipulate aggregates as we have shown. If the level of detail is too great (i.e. there are too many overlapping data points), the presentation system can choose to group and aggregate data by using the aggregate gateway graphical object. Once displayed, the user can expand an aggregate gateway to increase detail. If the aggregate gateway contains too much data to fit into a newly generated display, the presentation system must choose between three options: permit overly dense and overlapping data representations, change the scale and use scrolling controls to expose portions of the data, or reduce the visible data by creating new aggregates (i.e., those that do not overlap as much or that create an intermediate number of aggregates).

In addition, there are several possibilities for the graphic resulting from the expansion of the aggregate gateway:

- expand the aggregate data into the same display.

- create a separate display for the more detailed data, using the same graphic techniques as the parent display.
- create a separate display, using new techniques which shows only the detailed data.
- create a new display with new techniques which enable both the expanded and unexpanded aggregate data from the original display to coexist effectively.
- create a new display following some presentation directives that the user provides. For example, the user might want to emphasize a particular dimension of the aggregate differently from the previous display.

Design knowledge can also be used to select interface techniques in the data exploration tools. For example, instead of the outliner form of the aggregate manipulator as shown in Figure 11, a hierarchical graph (node-link diagram) might be used to show additional relations between aggregates or a Tree Map [21] to better show quantitative attributes of aggregates. Another example is the case where the data includes nominal attributes with a large number of values. Based on knowledge of the task and data characteristics, a system could select a form of DQ called the Alphaslider [2], which allow users to rapidly choose nominal values.

8. SUMMARY AND CONCLUSION

One important component in the design of user interfaces for exploring large data sets is that of data manipulation techniques. In this paper we explored these techniques with respect to a classification for data manipulation user goals, that of scope, focus of attention, and level of detail. We integrated into our interactive data exploration system, IDES, the technique of dynamic query, whose strength is scope operations, with the aggregate manipulator, whose strength is control of level of detail. We demonstrated how the combination of these tools can enable people to efficiently answer questions that are typical in data exploration.

Another important component for exploration large data sets is data visualization techniques. Automatic presentation systems are useful in that they relieve the user of the need to design and construct pictures. However, they must have features that allow users to construct graphics (as with SageBrush) and find previously constructed graphics to reuse or modify (as with SageBook).

In this paper, we have proposed a framework for knowledge-based, interactive data exploration. We have discussed the components that such a system should have in terms of our research on IDES and SageTools. Implementing this framework would require addressing all the issues we have presented:

- Developing richer directives that refer to characteristics of data, tasks, design choices, and aesthetic preferences.
- Extending characterization vocabulary to describe both the hierarchical structure of data as well as the data exploration tasks that users perform.
- Developing approaches to visualizing information using aggregate graphical objects, to serve both as useful representations of data and as mechanisms to explore data, i.e. as a gateway to more detailed data.
- Developing and testing interface mechanisms that support data exploration, including filtering, dynamic query, painting, hierarchy expansion and contraction, scrolling, and mechanisms for structuring, partitioning, and aggregating data.
- Developing interface mechanisms for interacting with automatic presentation systems, such as SageBrush and SageBook, which enable users to communicate presentation directives in a natural way.
- Developing and understanding the processes of data manipulation, data analysis and data visualization and their relationships.

In addition, future research will involve performing user studies to ascertain how well people are able to use the various components of these systems. In regard to IDES, we plan to explore how our object-attribute paradigm needs to be expanded to relational data that does not fall in this paradigm. We also plan to incorporate other visualization techniques, such as brushing, which supports coordination of attributes across multiple displays.

REFERENCES

1. Ahlberg, C., Williamson, C. and Shneiderman, B. Dynamic Queries for Information Exploration: An Implementation and Evaluation, in Proceedings of the CHI '92 Conference (May 1992), ACM Press, pp. 619-626.

2. Ahlberg, C. and Shneiderman, B. The Alphaslider: A Compact and Rapid Selector, in Proceedings of the CHI '94 Conference (Boston, April 1994), ACM Press, pp. 365-371.

3. Barr, R. Using Graphs to Explore Databases and Create Reports, in SIGCHI Bulletin (July 1990), p. 24-27.

4. Bertin, .J. *Semiology of Graphics*, The University of Wisconsin Press 1983.

5. Brachman, R.J. et. al. Intelligent Support for Data Archaeology, in proceedings of Workshop on Intelligent Visualization Systems, IEEE Visualization'93 Conference (San Jose, October 1993) , p. 5-19.

6. Casner, S. A Task-Analytic Approach to the Automated Design of Graphic Presentations, in ACM Transactions on Graphics, 10, 2 (April 1991), 111-151.

7. Cleveland, W.S. and McGill, M.E. *Dynamic Graphics for Statistics,* Wadsforth Inc., Belmont, CA 1988.

8. Goldstein, J. and Roth, S.F. Using Aggregation and Dynamic Queries for Exploring Large Data Sets, in Proceedings of the CHI'94 Conference (Boston, April 1994), ACM Press, pp. 23-29.

9. Gray, P.D., Waite, K.W., and Draper, S.W. Do-It Yourself Iconic Displays, in *Human-Computer Interaction - INTERACT '90*, D. Diaper et al., Elsevier Science Publishers B.V., 1990, pp. 639-644.

10. Holsheimer, M. and Siebes, A. Data Mining: The Search for Knowledge in Databases, Report CS-R9406, ISSN 0169-118X, Amersterdam, The Netherlands 1991.

11. Hutchins, E.L., Hollan, J.D., and Norman, D.A. Direct Manpulation Interfaces, in *User Centered System Design ,* D.A Norman and S.W. Draper eds., 1986, pp. 87-124.

12. Mackinlay, J.D. Automating the Design of Graphical Presentations of Relational Information, in ACM Transactions on Graphics, 5, 2 (April 1986), ACM Press, pp. 110-141.

13. McDonald, J.A. and Stuetzle, W. Painting multiple views of complex objects, in Proceedings of the. ECOOP/OOPSLA'90 European Conference on Object Oriented Programming (Oct. 21-25, 1990), ACM Press, pp. 245-257.

14. Marks, J.W. *Automating the Design of Network Diagrams,* Ph.D. Dissertation, Harvard University, 1991.

15. Maya Design. *The Webs Data Exploration Tool*, Maya Design Technical Report, Maya Design, Pittsburgh, PA 1993

16. Roth, S.F. and Mattis, J.A. Data Characterization for Intelligent Graphics Presentation, in Proceedings of the CHI'90 Conference (Seattle, April 1990), ACM Press, pp. 193-200.

17. Roth, S.F. and Mattis, J.A. Automating the Presentation of Information, in Proceedings of the Conference on Artificial Intelligence Applications (Miami Beach, Feb. 1991), IEEE Press, pp. 90-97.

18. Roth, S.F., Mattis, J.A., and Mesnard, X.A. Graphics and Natural Language as Components of Automatic Explanation, in *Architectures for Intelligent Interfaces: Elements and Prototypes*. Addison-Wesley, Reading, Mass., 1991.

19. Roth, S.F., Kolojejchick, J., Mattis, J. and Goldstein, J. Interactive Graphic Design Using Automatic Presentation Knowledge, in Proceedings of the CHI'94 Conference (Boston, April 1994), ACM Press, pp. 112-117.

20. Springmeyer, R.R., Blattner, M.M., and Max., N.L. Developing a Broader Basis for Scientific Data Analysis Interfaces. In *Proceedings of Visualization '92* (October 19-23, 1992, Boston, MA), pp. 235-242.

21. Turo, D. and Johnson, B. Improving the Viusalization of Hierarchies with Treemaps: Design Issues and Experimentation. In *Proceedings of Visualization '92* (October 19-23, 1992, Boston, MA), pp. 124-131.