Using Aggregation and Dynamic Queries for Exploring Large Data Sets

Jade Goldstein and Steven F. Roth

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3891
Tel: (412) 268-2145
Email: Jade.Goldstein@cs.cmu.edu, Steven.Roth@cs.cmu.edu

To appear in *Proceedings CHI'94*: Human Factors in Computing Systems. April 24-28, 1994. Boston, Massachusetts.

ABSTRACT

When working with large data sets, users perform three primary types of activities: data manipulation, data analysis, and data visualization. The data manipulation process involves the selection and transformation of data prior to viewing. This paper addresses user goals for this process and the interactive interface mechanisms that support them. We consider three classes of data manipulation goals: controlling the scope (selecting the desired portion of the data), selecting the focus of attention (concentrating on the attributes of data that are relevant to current analysis), and choosing the level of detail (creating and decomposing aggregates of data). We use this classification to evaluate the functionality of existing data exploration interface techniques. Based on these results, we have expanded an interface mechanism called the Aggregate Manipulator (AM) and combined it with Dynamic Query (DQ) to provide complete coverage of the data manipulation goals. We use real estate sales data to demonstrate how the AM and DQ synergistically function in our interface.

KEYWORDS: Interactive Techniques, Data Exploration, Data Visualization, Large Data Sets, Graphics Presentation, Intelligent Interfaces

1. INTRODUCTION

Many applications involving large quantities of data require mechanisms by which people can easily search, access, manipulate, view, communicate, extract information from, and discover relationships in their data. This may involve an iterative process in which users select some data, view it in a chart, map, table or other presentation appropriate to their goals; and based on the results, refine the selected data and repeat the process. In such cases that involve repeated human interaction, it is imperative to have user interface mechanisms which maximize both usability and functionality.

To address these needs, we are examining and building general data exploration tools. Our focus is on the class of data that consists of objects (e.g., house) and their corresponding attributes (e.g., sale price and number of bedrooms). This type of data is usually visualized in charts, maps, and network diagrams. Such data differs from scientific data in that it is not sampled and is usually not a measurement in a coordinate space. Accordingly, the interface mechanisms we are considering need to support this kind of data and its corresponding visualizations. To address this issue, we classified the types of interactive data exploration tasks (goals) that users would perform: *data manipulation* goals involve selecting portions of data and transforming data into new forms, *data analysis* goals involve obtaining statistics on portions of the data, and *data visualization* goals involve requirements and specifications for viewing the data through appropriate visualizations. Naturally, these categories are inter-dependent, e.g., selecting portions of data (a data manipulation process) can occur through a particular visualization. However, the categorization provides a useful framework through which to understand the tasks that users perform.

This paper concentrates on the data manipulation aspect of interactive data exploration. By examining the tasks users perform, we categorize user goals for data manipulation into three types: controlling scope, selecting focus of attention, and choosing level of detail (Section 2). This framework allows us to discuss and evaluate current data exploration systems according to how they address the particulars of these three categories (Section 3).

Based on our analysis, we find that Dynamic Query (DQ) [1] and the Aggregate Manipulator (AM)[1] complement each other's functionality and apparent usability. DQ enables selecting data based on value ranges of the attributes. The AM allows the user to create and decompose aggregates, which are groupings of data, and see their derived properties. In order for these mechanisms to be useful in a general purpose system, we extended DQ to handle nominal data attributes as well as quantitative attributes and both mechanisms to work on more than just pre-defined cases (Section 4). These extensions make the new versions of the AM and DQ both comprehensive and flexible. We then discuss our integration of these new versions in a data exploration interface (Section 5) and illustrate the integration of these data manipulation techniques in the real estate domain by providing example analyses to show how the AM and DQ work synergistically for exploring large data sets (Section 6). Since the AM and DQ provide complete coverage of data manipulation goals, combining them provides a highly useful tool for data manipulation functionality in exploring large data sets.

[1]Preliminary versions of the AM as well as the idea of distinguishing scope, focus of attention, and level of detail were developed in collaboration with Maya Design Group Inc., Pittsburgh, PA. We further elaborated the component operations for the work reported here.

2. FRAMEWORK OF DATA MANIPULATION GOALS

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 performed an extensive empirical analysis of the processes that scientists do in data analysis [8]. 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 and have identified three types of exploration goals needed: controlling scope, selecting focus of attention, and choosing level of detail.

Controlling *scope* has to do with restricting the amount of data one wishes to view. One way to control scope is to select a subset of values of a data attribute. For example, a user may wish to look at cities with a population over 2 million. This corresponds to selecting city data objects using the population attribute. Similarly, we can control scope using multiple data attributes (e.g. selecting cities with population over 2 million and average rainfall greater than 20 inches). In order to select data based on *quantitative attributes* (i.e., numeric attributes like population), users can select a range of values. For *nominal attributes* (i.e., non-numeric, unordered elements, such as state), users can select individual elements in the set (e.g., Florida) or select other types of attribute-value groupings such as pre-defined ones (e.g., states in the Eastern US). This type of pre-defined information could be stored in the system as attributes of other objects (i.e., there would be a city object with a state attribute and a state object with a region-of-US attribute). This enables a user to select cities partitioned by region-of-US. Users need the capability to define such groupings and add them to the system explicitly.

An alternative way of controlling scope is to disjunctively join subsets of the data, which possibly have overlapping elements. For example, a user may wish to look at 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 population and region-of-US attributes are used in this second set to control the number of cities considered. Note that the two sets (set 1 and set 2) can have overlapping data. For example, New York is in both sets.

The second class of goals addresses *focus of attention,* which involves choosing the attributes of data one wishes to both view in displays and manipulate through level of detail operations. For example, a database of cars may consist of various attributes (car-model, year, company, cost, miles-per-gallon, repair-rating, safety-rating), but a user may wish to just focus on the relationship between cost and safety-rating. Another type of focus operation is the creation of *derived attributes*, attributes which do not occur in the original data and are defined by the user. For example, we can create an attribute called manufacturing-location (with values of American, European and Asian) by assigning a value to each car based on its manufacturer. We can accomplish this by partitioning the data into three groups. This can be expressed visually by coloring the cars on a display based on their manufacturer [3] with American cars in shades of blue, European cars in shades of yellow/orange, and Asian cars in shades of red. Another method is to manually select all American cars and create a group (aggregate), and then form a group for European cars and one for Asian cars. If the user wants to reuse this derived attribute later, the user would need to have the system store this information, analogous to the previously mentioned attribute region-of-US. The coloring technique is a visualization technique referred to as *brushing* [3] or *painting* [5] and the group creation technique is provided by the aggregate manipulator (Section 3). Another way to create derived attributes is to transform existing attributes by some filter [4], 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*: grouping data in some manner meaningful to the user, or by *decomposition*: partitioning data by the values of attributes, i.e., breaking a larger group into smaller groups. First we give an example of the process of aggregation and then we will discuss decomposition. 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. Users can also create ad-hoc groupings. For example, in the house-sale data, the user could aggregate the dates of the year into quarters beginning in November. On any resultant aggregate, users might want to do data analysis operations, which involve examining *derived properties* of the data. These include defining *summary statistics*, which are statistics that can be computed on the values of attributes (e.g., sum or mean). In the case of the "quarter" aggregates that we created for the house-sale data, we could request the total number-houses-sold or mean (daily) total-sale-price for each quarter. These quarter aggregates represent a coarser grain level of detail than individual months.

Homogeneous and heterogeneous decomposition involve reducing a data group into smaller groups based on the same or different attributes of the included data objects. *Homogeneous decomposition* is the process of using the same attribute to repeatedly partition a group by choosing more and more narrow ranges of the attribute's values. *Heterogeneous decomposition* refers to the use of different attributes to decompose sets for successive partitions. For example, consider a real estate sales database with the following attributes: house-selling-price, neighborhood, number-of-bedrooms. Figure 1 illustrates heterogeneous and homogeneous decomposition for houses whose neighborhood attribute is Shadyside or Squirrel Hill. We can partition this group of houses by the price attribute in the ranges, 0-50K, 51-100K, 101-150K and over 150K. Using homogeneous decomposition, we can further divide the 51-100K house partition into the groups, 51-75K and 76-100K. Using heterogeneous decomposition, we can partition the 101-150K aggregate created by the price attribute, by the attribute number-of-bedrooms.

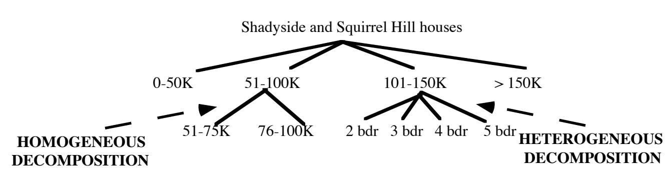

Figure 1: Aggregation: homogeneous & heterogeneous decomposition.

Homogeneous and heterogeneous decomposition are two ways to form a *hierarchy*, which structures data by imposing meaningful groupings. When we perform these operations, we need a way to specify how partitions are formed. We have identified four classes for 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. Psychologists may decide to break the year into holiday days and non-holiday days (Figure 2a).
- element frequency divisions (computed by the system): Each division has the same number of elements, hereafter referred to as *equi-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 each time interval and each time interval can have a different length. (Figure 2b).
- set interval divisions (computed by the system): Each division has the same length interval, *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 2c).

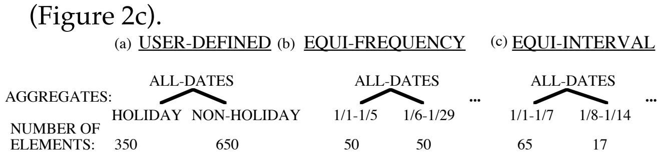

Figure 2: Types of decomposition: (a) user-defined, (b) equi-frequency, (c) equi-interval.

- system-provided statistical methods: The system can use clustering statistics or other methods to partition the data into groups.

Aggregates are groups formed as a result of decomposition or aggregation. Aggregates have *data characterizations* which are derived from the data characterizations of their elements. This characterization describes the application-independent properties of data that are the basis for graphic design [7], e.g., a nominal data type. When forming an aggregate, the system uses the characteristics of the individual data objects to infer the aggregate data characterization, which is similar to computing summary statistics for each of its attributes. For example, if the system is grouping data which includes the attribute costs, a representative data value for the aggregate object could be average or total. This representative value is used when displaying the aggregate. Values in our current system include counts, averages, totals, and ranges. Which values can be selected depends on the data characterizations of the individual elements. For example, an average or total does not make sense for nominal data.

3. EXISTING DATA EXPLORATION TECHNIQUES

There are several existing techniques for exploring large data sets. As mentioned, these methods fall in three categories: (1) data manipulation techniques, (2) data analysis techniques, and (3) data visualization techniques. Data manipulation techniques are methods of selecting, grouping and transforming the data. In this section, we will examine user interface mechanisms for data manipulation techniques in terms of how they address the data manipulation operations of scope, focus of attention and level of detail (Table 1). We will not address query languages, such as SQL, because our focus is on interface tools that assist the user in manipulating their data.

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 corresponds to a data attribute. This technique works best for quantitative data, but can also work in the case of nominal data, for which the user selects elements from an exhaustive list. DQ has previously been implemented for pre-defined attributes. The advantages of this technique are that slider bars are easy to manipulate, and one can see the changing effects on the visualization of the data. One disadvantage is that there is no easy way to represent disjunction, the combination of two sets of attributes. Some simple cases of disjunction may be represented by having multiple ranges available for manipulation on one slider bar.

	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".

Webs performs level of detail operations through an aggregation mechanism called the "aggregate manipulator" (AM). It was designed for level of detail operations, that of aggregation (grouping) and decomposition (partitioning groups). However, using these methods, the user can also perform scope operations by controlling the amount of data and focus of attention operations by partitioning through ad hoc selection of attributes. The AM also provides a mechanism to display summary statistics for attributes of aggregates (a data analysis operation). The AM will be discussed further in sections 5 and 6.

Iconographer [4] has several mechanisms to handle scope and focus of attention, but only weakly supports level of detail operations. Iconographer uses the model of directed graphs that are programmed visually by the user. The Object Filter handles the data filtering. The directed graph methodology allows for both scope operations. The Attribute Builder allows one to create derived attributes. by visually programming a graph of functions that will transform existing attributes. The Switchboard mechanism allows the user to select attributes and link them to display techniques. It does not appear that these mechanisms would allow rapid data manipulation for scope and level of detail operations.

Powerplay [2] allows the user to "drill-down" across pre-defined hierarchical structures called data dimensions. For example, for car sales data, total sales can be broken down by year, then by quarter, then by month. Sales can also be broken down by geographic region, and then by individual states. Powerplay allows both homogenous and heterogeneous decomposition. Powerplay also allows the user to select attributes to display and to choose certain types of display graphs. Displays or tables are created based on the current level of detail. Powerplay does not allow the capability to perform any scope operations other than this drill-down capability. More importantly, since Powerplay only allows decomposition of pre-defined hierarchical structures, it does not allow decomposition by ad hoc attribute selections.

Excel has the ability to perform level of detail operations through an "outline" mechanism. Excel allows the user to create groupings (aggregates) of data sets rather than constraining the user to pre-defined groupings. However, this is cumbersome in that Excel has no knowledge of underlying data types, data attributes, data sets, data objects, and the membership of objects in sets. Consequently, Excel has few mechanisms for controlling the scope easily, other than manually selecting cells. Users must individually link cells to other cells to create a hierarchical outline indented by levels. Once this structure is defined the user can collapse and expand levels, but only within this structure. Changing the decomposition requires changing the spreadsheet model. Excel provides the means of doing summary statistics by attaching formulas to cells, which are manually linked by the user to the cells needed for performing the operation. Since Excel allows the user to select portions of any spreadsheet for display and also to create new attributes from operations on the data, Excel can perform all three focus of attention operations.

4. SELECTING INTERFACE MECHANISMS

Table 1 shows us that to have complete coverage of our desired data exploration operations we could select just the aggregate manipulator. However, the AM does not filter data or select 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) are 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 operation as well as other functions such as displaying summary statistics. Examples of the interactions between DQ and the AM will be given in Section 6.

In order to be able to use both the AM and DQ in our system, we needed to extend them to function for any type 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 the data characterization rather than built-in mechanisms for the particular application. For DQ, we needed to be able to create a slider on demand and to have a method to select elements of nominal data rather than just ranges of quantitative data. For nominal data we use a scrolling list of elements (see Figure 6) 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

In our system, we have selected two data exploration mechanisms: Dynamic Query (DQ) and the Aggregate Manipulator (AM) due to their coverage of the data manipulation operations and their apparent ease of use in an interactive interface. Figure 3 summarizes the main functionality and data flow of the AM and DQ. Decoupling the display 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.

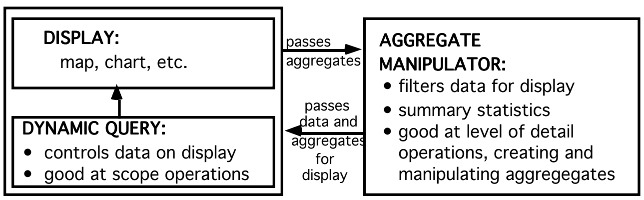

Figure 3: Data Flow for the AM & DQ.

The AM, DQ, and display comprise three of the five workspaces in our interface (see Figure 6). 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 display the elements of the aggregates created by the AM. The dynamic queries are always connected to the current display. Changing the sliders changes the portion of the data that is displayed. There are command buttons (under the aggregate manipulator and display area) which allow the user to create an aggregate, display 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) or by dragging a bounding box around them. Selected icons can be composed into a new aggregate, by the "Create Agg" command. The user gives the aggregate a name, and the representation of the individual data objects disappear and are grouped into the representation of the aggregate data object, which we call an *aggregate gateway*. If the user selects "Dsply->AM", the aggregate is also displayed in the AM with the name specified by the user. The user can decompose an existing aggregate into its components on the display by double clicking on the aggregate gateway object. The number of objects displayed reflects the bounds of the existing dynamic query sliders. This decomposition operation does not affect the AM. If the user wishes to display an aggregate from the AM, the user can select the aggregate(s) and use the "AM->Dsply" command or clear the display first using the "Clear Display" command. An aggregate can be cleared from the AM by selecting the aggregate and using the "Clear Agg" command.

The last two workspaces consist of the attribute area and the data detail area. The data detail area (lower right corner) is used for the display of aggregates or individual data objects. Attributes are displayed in this area by pressing on the "Show Agg" command and selecting an attribute via the resultant pop-up menu. The attribute area lists the attributes available in the database.

6 REAL ESTATE APPLICATION

We have implemented this design for a real estate sales domain, in which the display area currently only consists of a map display. Ultimately, we intend to integrate this system with SAGE [6], which will provide a variety of graphics for the display area. In this section we will discuss the real estate domain and the tasks one might perform with this data set. We will show that we need a mechanism such as the AM to allow the user to disjunctively combine sets and that the combination of DQ and the AM is better than a single mechanism only.

Our data consists of attributes similar to that of an actual real estate listing, but has additional information, e.g., selling price, because the data consists of houses that have been sold. There are 27 attributes (Figure 4) 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 into agents. There are many possible user-defined partition options, such as the partitions in Figures 1, 5 and 6.

• 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 4: 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 solely DQ requires a lot of work for the user. For both these cases, we show how using the combination of the AM and DQ involves less work to achieve the user's goals.

Consider the following scenario. Jennifer is new to the Pittsburgh area and has the following goals for a house:

- in the price range $100,000 to $150,000.
- close to Carnegie Mellon University (CMU).
- a lot size larger than 5000 sq. ft. in the neighborhoods Shadyside or Squirrel Hill, *or* a lot size larger than 8000 sq. ft, in the neighborhood of Point Breeze, which is further away from CMU.

Since Jennifer is new to this area, she would like to see how many houses which sold in recent years match these criteria. In particular, Jennifer would like to see a map display of both sets of houses (Shadyside-Squirrel Hill and Pt. Breeze) that meet her criteria. DQ is not capable of displaying two sets on the map (in one set the lot size must be larger than 5000 sq. ft. and in the other set they must be larger than 8000 sq. ft.). However, the AM is capable of handling this scenario.

Using just the AM, Jennifer would divide the initial aggregate group of AllHouses by selecting a user-defined partition consisting of one group for the attribute selling price. Using the provided slider, she would specify the range $100-$150,000 and give the group a name, “100-150K”. This new group is indented from the initial group AllHouses in the *outliner* portion of the AM (Figure 5a). She would then select the user-defined partition for neighborhood and break this cost range into the three neighborhoods: Shadyside, Squirrel Hill, and Point Breeze. She would then select Shadyside and Squirrel Hill and create a new aggregate, “SqHill-Shady” (composed aggregates are flushed one indentation to the left of the leftmost aggregate in the outliner of the AM table, see Figure 5b). She then partitions SqHill-Shady by the attribute lot-size, specifying one user-defined grouping of data larger than 5000 sq. ft., which she names “>5000”. She does the same for the group Point Breeze specifying larger than 8000 sq. ft. She then composes these two groups (“>5000” and “>8000”) into a new aggregate, named InterestHouses (Figure 5b). From Figure 5b, it is apparent that with several decompositions, recalling how an aggregate was created or what it represents could be confusing. Moreover, creating the user-defined partitions through the AM involves more work than using DQ.

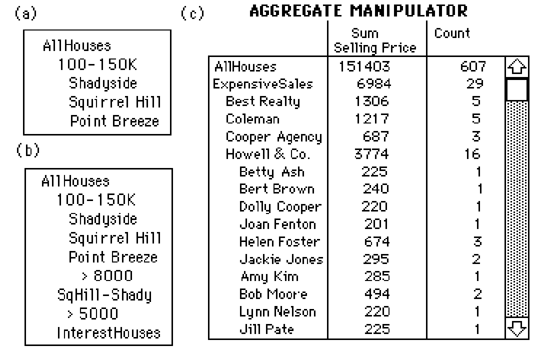

Figure 5: The functionality of the Aggregate Manipulator. (a) The outliner of the AM showing decomposition of houses by partition 100-150K followed by a neighborhood partition of three neighborhoods. (b) The outliner showing creation of the aggregate InterestHouses, from the aggregates >5000 of SqHill-Shady and >8000 of PointBreeze. (c) 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.

It is therefore advantageous for Jennifer to use the combination of the AM and DQ to cover this case (Figure 6). The initial state of the system has the aggregate “AllHouses” in the AM and all houses displayed on the map. First, Jennifer uses the mouse to *press* the “Create Query” button to display a pop-up menu of all attributes. She selects the attribute Selling Price and a dynamic query slider is created. She then uses the same procedure for Lot Size and Neighborhood. After she selects the appropriate ranges or values for these queries, the map displays houses in Shadyside and Squirrel Hill, with a price between 100-150K and lot size between 5000-20000 sq. ft. Jennifer then selects all the data on the map and uses the “Create Agg” command to create a new aggregate “SqHill-Shady”. She then selects this aggregate and uses the command “Dsply->AM” to transfer the aggregate to the AM. To create the second set, she uses “Clear Dsply” to clear the map display, selects the aggregate AllHouses from the AM, uses the command “AM->Dsply” to put the aggregate gateway object on the map and then double clicks on the aggregate gateway to expand the aggregate into the individual house data. She then uses the existing DQ sliders to create a new range for lot size, 8000-20000 and to select the neighborhood Point Breeze. Note that if Jennifer wanted to see 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. After adjusting the sliders, Jennifer creates an aggregate of the whole map, "PtBreezeBigLot" and displays it in the AM. She then selects the two aggregates PtBreezeBigLot and SqHill-Shady and creates a new aggregate "InterestHouses" and displays the result on the map through the "AM->Dsply" command. She then presses on the map aggregate to display a pop-up menu of operations that involve this aggregate. She chooses "Set Query Values" and the dynamic queries are set to reflect the house data in the aggregate "InterestHouses". She then double clicks on the aggregate to see the individual house data (the results of these operations are shown in the map display and dynamic query area of figure 6).

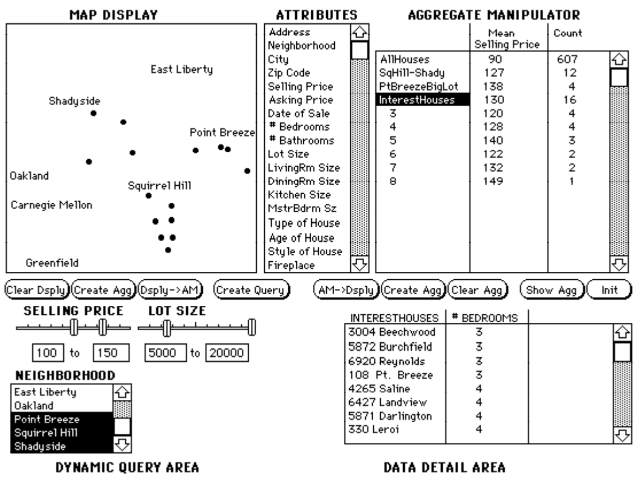

Figure 6: The workspaces of the interface: display, attributes, aggregate manipulator, data detail area and dynamic query area.

Figure 6 also shows summary statistics and data for the aggregate InterestHouses. Jennifer obtains these values by pressing on the top column header area of the AM table, selecting the summary statistic "mean" from the pop-up menu, and then pressing on the lower header area of the table to obtain a pop-up menu of the possible attributes that can be used with "mean" and selecting "selling price". Note that this procedure could be done in reverse, choosing "selling price" and then selecting from a list of possible summary statistic options. This is possible because the system has built-in knowledge of the data types of the attributes and "knows" what summary statistics are possible for each data type. For the second column, Jennifer picks the summary statistic "count", which does not require a corresponding attribute. To partition InterestHouses, she presses on "InterestHouses" 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 (refer to the AM in Figure 6). 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 mean selling price and count). To get the display in the Data Detail Area, she selects the aggregate InterestHouses, presses the command "Show Agg" and then chooses the attribute "# Bedrooms".

The second scenario involves another situation in which we hypothesize 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 figures 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 ranges and value. Thus, first John uses DQ to select ranges for the price, the neighborhood and the date range. From this he creates an aggregate ExpensiveSales (Figure 5c). He then partitions this aggregate by the attribute company and selects two summary statistics: "sum" 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, but only by one house.

7. CONCLUSION AND FUTURE RESEARCH

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 framework for classifying user goals, that of scope, focus of attention, and level of detail. Of current techniques that cover level of detail functionality, the aggregate manipulator seemed to perform the operations in the most straightforward manner. However the aggregate manipulation is somewhat unwieldy for selecting ranges of attributes, and so we integrated into our system the technique of dynamic query, whose strength is these types of scope operations.

While we have demonstrated that a combination of these tools appropriately applied can enable people to efficiently solve questions that are typical in data exploration, an important area of future research will be whether users are able to recognize the most effective ways to use the many possible combinations of aggregate manipulation and dynamic query provided in the interface. Our current work is to perform user studies to assess this and related questions. In addition, we plan to explore how these concepts need to be revised for relational data which does not fall into the object-attribute paradigm, as well as analyze the goals users have for data visualization, provide a painting mechanism which supports coordination of attributes across multiple displays, and integrate this system with SAGE, our automatic presentation system, to provide more types of visualizations.

ACKNOWLEDGMENTS

We would like to thank Pete Lucas, Jeff Senn, Joe Ballay and Carolanne Fisher of Maya Design Group Inc. for their contributions to this work. Thanks also to John Kolojejchick, Joe Mattis, Carolyn Dunmire, Mei Chuah, Octavio Juarez, Francesmary Modugno, Edoardo Biagioni, Bob Doorenbos and Kathyrn Porsche for their useful comments in discussions and/or on versions of this paper. Funding for this project was provided by the Advanced Research Projects Agency and the Army Research Office.

REFERENCES

1. Ahlberg, C., Williamson, C. and Shneiderman, B. Dynamic Queries for Information Exploration: An Implementation and Evaluation. In *Proceedings of CHI'92 Human Factors in Computing Systems* (May 3-7, 1992, Monterey, CA), ACM/SIGCHI, pp. 619-626.
2. Barr, R. Using Graphs to Explore Databases and Create Reports. In *SIGCHI Bulletin* (July 1990), p. 24-27.
3. Cleveland, W.S. and McGill, M.E. *Dynamic Graphics for Statistics*. Wadsworth, Inc., Belmont, CA 1988.
4. 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.
5. 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.
6. Roth, S.F., Kolojejchick, J., Mattis, J., and Goldstein, J. Interactive Graphic Design Using Automatic Presentation Knowledge. In *Proceedings of CHI'94 Human Factors in Computing Systems* (April 24-28, 1994, Boston, MA), ACM/SIGCHI.
7. Roth, S.F. and Mattis, J.A. Data Characterization for Intelligent Graphics Presentation. In *Proceedings of the CHI'90 Human Factors in Computing Systems* (April 1-5, 1990, Seattle, WA), ACM/SIGCHI, pp. 193-200.
8. 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.