Generalized Selection via Interactive Query Relaxation

Jeffrey Heer, Maneesh Agrawala, Wesley Willett
Computer Science Division and Berkeley Institute of Design
University of California, Berkeley
Berkeley, CA 94720-1776 USA
{jheer, maneesh, willettw}@cs.berkeley.edu

ABSTRACT

Selection is a fundamental task in interactive applications, typically performed by clicking or lassoing items of interest. However, users may require more nuanced forms of selection. Selecting regions or attributes may be more important than selecting individual items. Selections may be over dynamic items and selections might be more easily created by relaxing simpler selections (e.g., "select all items like this one"). Creating such selections requires that interfaces model the declarative structure of the selection, not just individually selected items. We present direct manipulation techniques that couple declarative selection queries with a *query relaxation* engine that enables users to interactively generalize their selections. We apply our selection techniques in both information visualization and graphics editing applications, enabling generalized selection over both static and dynamic interface objects. A controlled study finds that users create more accurate selection queries when using our generalization techniques.

Author Keywords

Selection, annotation, pointing, reference, information visualization, input techniques, query relaxation

ACM Classification Keywords

H.5.2 Information Interfaces: User Interfaces.

INTRODUCTION

Pointing to an item or region of interest is common in everyday communication because it sets (or *grounds*) the subject of the conversation or action. In the physical world, people coordinate their gestures, gaze, and speech to indicate the objects under discussion [7, 8]. In graphical user interfaces, reference (or selection) remains of critical importance, but is realized through a more limited set of actions, such as clicking or lassoing items of interest. Most interfaces model selections as a simple collection of selected items. While this approach is simple to implement, it makes it difficult for users to specify higher level selection criteria.

CHI 2008, April 5–10, 2008, Florence, Italy.

Figure 1. Map of reported homicides in Los Angeles, 2007. Color indicates cause of death, shape indicates the victim's race (the complete visualization, with legends, is shown in Figure 5).

Consider the visualizations of reported homicides in Los Angeles shown in Figures 1 and 5. Analysts collaborating around these visualizations might refer to regions or attributes of interest [8], such as "East L.A.", "homicides in the month of May", or "all gunshot victims". Similarly, an analyst may point to an item and refer to "all items blue like this one," verbally generalizing a deictic reference based on the properties of the item [3, 7].

One way to express such selections is to use a higher level query language such as SQL. For example, the SQL clause

```
('2007-05-01' ≤ date AND date ≤ '2007-05-31')
```

selects all homicides in the month of May 2007. The query encodes the structure of the selection declaratively, and applying the query results in a set of selected items. Systems such as DEVise [19], VQE [9], and Improvise [27] have recognized that maintaining query structure increases the expressiveness of visualization applications. Each of these systems provides graphical user interfaces for instantiating such general queries visually.

In this paper, we also focus on building a selection interface that represents the selection as a declarative query over the attributes of interface objects or underlying data. Selection queries are modeled in a SQL-like query language and as in earlier systems (e.g., [9, 19, 23]) users create selection queries through direct manipulation. Our system visualizes the structure of the query and highlights the data or interface objects selected by the query. This formulation supports both selection of specific items and selections based on attributes of the data, which may vary over time.

The unique contribution of our work is to couple this query-based approach with generalization mechanisms that allow users to expand their selections based on an initial selection. This approach enables generalized selections such as "select all victims with the same age as this one" over both static

and dynamic data. Such selections are automatically generated by a *query relaxation* engine that analyzes the attributes and network topology of the interface objects or underlying data. Users interactively select generalization criteria through a pop-up dialog (providing choice mediation [20]) or by repeatedly clicking to cycle through a set of alternate selections (providing repetition mediation in a manner similar to [24]).

We begin by reviewing related research on selection and reference. Next, we demonstrate our approach in both a data visualization system and a vector graphics drawing program and describe our system architecture. We then describe a user study of our selection techniques in a data visualization application, finding that subjects used query relaxation to more effectively author selections. Finally, we discuss future work and conclude.

RELATED WORK

Our work on interactive query relaxation draws on research on direct manipulation selection techniques, including brushing, linking, and dynamic queries, as well as query relaxation techniques from the database community. We consider each of these in turn.

Selection Techniques and Reference

Social psychologists have examined the basic prerequisites for communication, including *reference*: indicating items, people, and places to be discussed. Clark and Brennan [3, 7, 8] explain that spatial reference to visible objects and regions takes many forms. Such references may be *general* (e.g., "north by northwest"), *definite* (e.g., named entities), *detailed* (e.g., described by attributes, such as the "blue ball"), or *deictic* (e.g., pointing to an object and saying "that one"). People often apply multiple forms of reference in tandem, across modalities such as speech and gesture. However, graphical interfaces rarely support such fluid and general forms of reference.

Clark [8] further divides deictic reference into *pointing* and *placing*. Pointing involves vectorial reference, such as pointing a finger or directing one's gaze to a specific item. Placing involves referencing a region of space imbued with a shared meaning, such as placing groceries on a counter to indicate items for purchase. To varying degrees, graphical interfaces use both forms of reference. Pointing actions using the mouse cursor are the most common. Placing also occurs, most notably in drag-and-drop, where drop targets have defined semantics. However, systems rarely support interactive specification of new "places." In interfaces, such places may include both spatial regions and abstract spaces defined by data attributes.

In this paper, we describe interface mechanisms that better support these forms of communication. Our selection query and relaxation model enables interactive generalization of *deictic* references and specification of *placing* regions whose contents may change over time.

Dynamic Queries, Brushing, and Linking

Our work is closely related to selection techniques used in information visualization. Dynamic queries [1] typically take the form of widgets, such as range sliders, with which users incrementally filter visualizations. Brushing [2, 4, 21] enables selection through direct manipulation, typically via clicking, lassoing, or "painting" over items of interest.

One class of systems focuses on interactive selection within visualizations [1, 2, 10]. Martin and Ward [21] introduce multi-dimensional brushing, in which users can brush over projected data using 2D selection regions. Their system then considers the min/max values of the brushed points to compute a hypercube enclosing the brushed points in all dimensions. Hypercube construction is a specialized form of query relaxation: the items initially selected are extended to a full hypercube. Hochheiser and Shneiderman's time boxes [14] are dynamic queries that select all time-series that pass through brush regions; our approach generates a similar tool through relaxation of range queries.

Another class of systems uses visual query mechanisms to create visualizations and specify linking relationships for coordinated brushing across visualizations. Snap-Together Visualization [22] implements linking using "primary key actions" that communicate the individual tuples that have been selected. Chen's compound brushing [4] provides a graphical data-flow language enabling user-created brushing operations across visualizations.

Some of these systems explicitly represent the structure of selection queries. Linking in DEVise [19] is specified through chains of linked plots, specified in part with brushing gestures. The system maintains a declarative query structure to perform linking across views. Improvise [27] supports *coordinated queries* authored in an auxiliary tree editor for defining and linking visualizations. Derthick et al's Visual Query Environment (VQE) [9], provides a form-based interface for specifying *intentional* (declarative) queries coupled with brushable visualizations for specifying *extensional* queries (selection of specific items). Olston et al's VIQING [23] provides a direct manipulation interface for specifying queries; users rubber-band a set of visualized tuples to select them and they drag visual canvases on top of one another to join the underlying data. Polaris [25] allows specification of both queries and visualizations by dragging database column names from a list onto "shelves" for visual variables such as position, color, and shape.

Our work follows in the tradition of these systems, enabling users to interactively select visualized data or other interface objects. Similar to DEVise, VQE, and Improvise, our system uses a declarative query model that supports coordination and reuse across visualizations. Like VIQING, our system supports the creation of declarative selection queries through direct manipulation of the visualization itself. Most importantly, our system is unique in using query relaxation to interactively generalize selection queries.

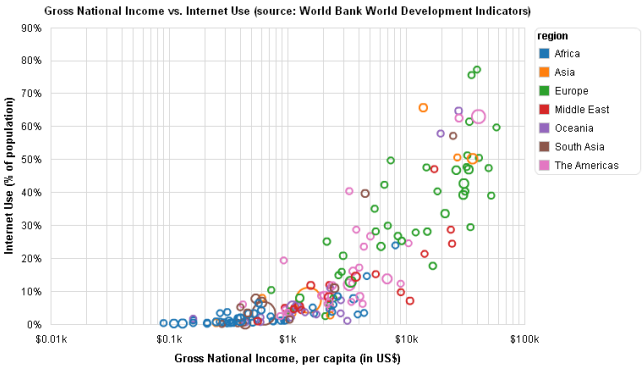
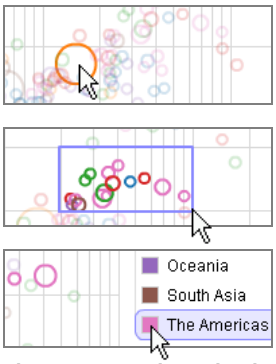


Figure 2. World Development Statistics. The visualization plots income against internet usage for the world's countries (c.f., [12]).



Item Selection by Clicking

`(id = 'China')`

Range Selection by Dragging

`(2000 < gni AND gni < 10000) AND (.1 < internet AND internet < .2)`

Attribute Selection with Legends

`(region = 'The Americas')`

Figure 3. Basic selection operations and resulting query "WHERE" clauses. Images are close-ups from the plot in Figure 2.

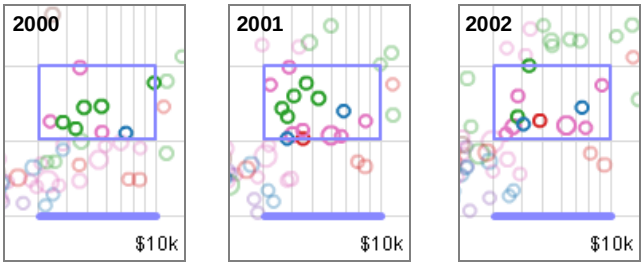


Figure 4. Selection over time-varying data. The selection updates dynamically as animated data points pass through the selection range. The sequence shows views for the years 2000 to 2002.

Query Relaxation

The database community has developed query relaxation with the goal of creating "cooperative" databases that return information beyond that specified by a standard query. Query relaxation expands the query selection criteria to include additional relevant information, often by consulting a semantic model of the data domain. For example, a user seeking to travel from New York to Boston might query for morning flights. If no matches are found, relaxed queries might instead return train routes in the same time frame.

Gaasterland [11] introduces query relaxation techniques in deductive databases, using logic rules to specify legal relaxation constraints. Chu et al provide query relaxation for relational databases [5] and XML documents [6], using type-abstraction hierarchies (hierarchical ontologies) to find semantically similar query results. Hierarchies can be hand-authored or generated by unsupervised clustering [5, 15].

Our work adapts query relaxation techniques to support generalized selection in graphical interfaces. As described in the following sections, our system supports configurable relaxation operations based on the attributes of interface items and relations between them. In most cases our system can produce a variety of relaxations from an initial query.

We provide interaction techniques that enable users to relax selection queries, cycle through the generated selections, and combine relaxed selections as desired. These techniques are modeled after *mediation* interfaces that disambiguate input among multiple alternatives (e.g., [16, 20, 24]). For example, text editors such as Microsoft Word set the cursor position on a single click, select a word on a double click, and select a paragraph on a third click. By cycling through the alternatives users can find the appropriate selection.

EXAMPLE: INFORMATION VISUALIZATION

We have integrated our generalized selection and query relaxation techniques with *flare* (http://flare.prefuse.org), an open-source visualization toolkit for the Adobe Flash Player.

Basic Brushing and Selection

Our selection framework supports common brushing and dynamic query operations. Figure 2 is a scatter plot of development statistics from the World Bank [28], including per capita income, internet usage, and population data for the countries of the world (c.f., Gapminder [12]). As shown in Figure 3, our framework translates selection operations in the visualization into declarative queries over the visualized data. The selection query is in turn used to generate interactive range brushes and highlighting effects.

Users can click an item to select it (Figure 3, top), and optionally hold the shift key to select multiple items. Users can click and drag over the visualization to create a range query (Figure 3, middle). The range is persistent and users can reposition and resize the range as they desire. Users can also drag along the axis labels of the chart to create one-dimensional ranges. Additionally, all legends also function as dynamic query selectors (Figure 3, bottom). Users can select collections of items in discrete legends or select ranges in continuous legends, just as they can in the chart.

Selection Reuse

Because our system maintains the structure of the selection query, it can reapply the selection dynamically over streaming or time-varying data sets. Figure 4 illustrates countries passing in and out of a range selection as the visualization is updated with new data for each year.

Our system can also reapply selection queries across different visualizations of a data set and thereby supports linking across views. Figure 5a shows a visualization of reported homicides in Los Angeles in 2007, collected by the L.A. Times [18]. Color indicates the cause of death and shape indicates the victim's race. The current selection

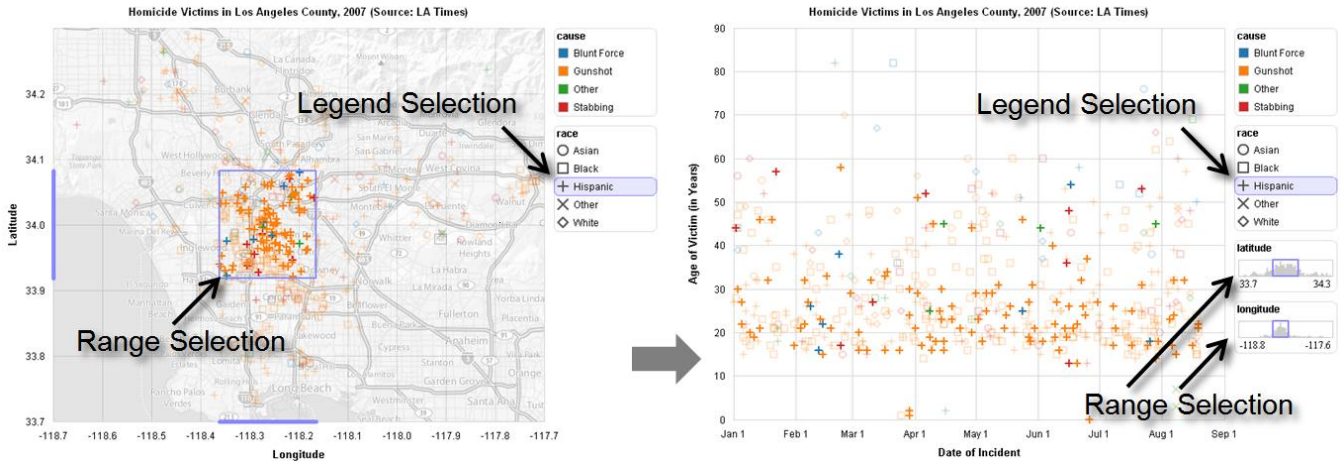


Figure 5. Reported Homicides in Los Angeles County, 2007. (a) Left: Geographic distribution of homicides, including the cause of death (color) and victim's race (shape). A selection highlights Hispanic victims (using a legend selection) in central L.A. (using a range selection). **(b) Right:** The same data plotted using incident date vs. victims' ages. The selection made in the geographic display has been mapped to the scatter plot. Our system extracts the latitude/longitude ranges from the selection query and generates appropriate dynamic query widgets.

highlights Hispanic victims in central L.A. Figure 5b shows the same data plotted as a scatter plot of incident date and victim's age. The selection made in the geographic visualization is preserved across the two views: range criteria for latitude and longitude from the geographic view appear as interactive ranges within query histograms next to the scatter plot. Our system inspects the clauses of the selection query to automatically generate the additional range visualizations and thereby ensure that the complete structure of the selection query is visible.

Data-Aware Annotation

In addition to exploration, selections are important for indicating items for collaboration and presentation [13, 26]. Users can add text annotations as they explore a data set. Our system links the annotation to the data using the current selection query. When collaborators view each others' annotations, the system applies the saved query. Because our system enables reuse of queries across different views, collaborators can view each others' annotations under different visual encodings, potentially providing additional perspectives in subsequent collaborative analysis.

Furthermore, the query structure can be leveraged to rank and filter annotations. For example, when a selection query results in a null result set due to external filtering criteria, it might be omitted from the list of relevant annotations. In addition, the data columns referenced by the query can be compared with the data columns being visualized to form a similarity measure between the selection query and the current view. We apply this measure to sort annotations according to their relevance to the current view.

Query Relaxation: Generalizing to Related Selections

Our system also supports the construction of generalized selections from simpler selections using query relaxation techniques. Users can pick an item or region of interest and generalize the selection to include additional items related to the initial selection (e.g., "select all items like this one").

Consider the date-by-age scatter plot in Figure 5b. Clicking an individual item queries the backing data tuple. Figure 6 depicts the use of repeated clicks to cycle through relaxed queries for the date attribute, expanding the selection to include items in the same day, week, and month. In this case, our query relaxer generates sequential relaxations by traversing a semantic model of time.

A click-and-hold invokes a dialog box, with which the user can choose attributes of interest, such as cause of death, race, and age (Figure 7, left). The relaxed query selects all items that match the attribute values of the initially selected items (Figure 7, right). In this fashion, users construct expanded selections based on attributes of interest.

Query relaxation can also be applied to multiple items or to range queries. Figure 8 shows a range selection that has been relaxed along the 'race' dimension. The resulting query selects all victims whose race matches that of any victim contained within the range bounds. Figure 9 shows a similar relaxation in a time-series visualization. The plot shows aggregated homicide counts over time, grouped by age into 5-year bins. Creating a range query over this visualization selects all individual data points within the range. Relaxing the query along the age dimension selects all time-series that pass through the selection range. Because we retain the query structure, subsequent resizing or repositioning of the range results in dynamic updates to the selection, enabling interactive querying similar to Hochheiser and Shneiderman's TimeSearcher [14].

Alternate Output Modalities

Developers can further extend or customize how selections are presented. Selection queries can be output in a SQL-like syntax to be exported (as in Figure 3) to databases or hand-modified by proficient users. Selection queries can also be mapped into a natural language representation, providing automated captioning for selections and potentially aiding visually impaired users. Using a simple rule-based approach,

our system generates text descriptions of selections. For example, our captioner outputs *"All items from August 1 to August 31"* for Figure 6 (right panel) and *"All items with race equal to 'White'"* for Figure 7.

EXAMPLE: VECTOR GRAPHICS EDITOR

Although our primary motivation for building generalized selection techniques comes from data visualization, our approach is applicable in other visual interfaces. To demonstrate the generalizability of our approach, we have applied our selection techniques in a vector graphics editor, similar to programs such as PowerPoint and Visio. As in the earlier visualization examples, users can select both individual items and ranges, create data-aware annotations (e.g., for design reviews), and generalize selections through query relaxation. The principal difference is that for the vector graphics editor, no translation between visual and data variables is needed, as the data set being queried consists of the graphic objects themselves.

As before, clicking and holding over an item provides a dialog allowing users to generalize their selection to items with matching shapes, colors, and fonts (Figure 10). For example, one can click a text object and generalize by the font type to select all matching text boxes, enabling subsequent batch editing. Thus, our system automatically generates operations similar to the "Select > Same" and "Select > Object" menu commands in Adobe Illustrator.

Moreover, the query relaxer supports additional forms of query relaxation. The drawing editor includes connectors, which link items in an underlying network. This network provides a substrate on which to perform query relaxation. As shown in Figure 11, one click selects an item, two clicks also selects all items one hop away, and three clicks selects the entire connected component. We describe other forms of relaxation over networks in the implementation section.

IMPLEMENTATION

Our generalized selection techniques are implemented in the ActionScript 3 programming language and are intended for use within the Adobe Flash Player. A selection controller enables selection over visual items in the Flash Player scenegraph, using queries over the properties and sub-properties of these objects. In addition to processing input events, the controller coordinates query generation, query visualization, and query relaxation components.

Initialization

The controller takes as input both a container object holding the selectable objects and a *schema mapping* describing the accessible properties of interface objects. If visual variables (e.g., position, color, shape) are determined from backing data, the schema object maintains this mapping, including scale transforms (e.g., ordinal, linear, log scales).

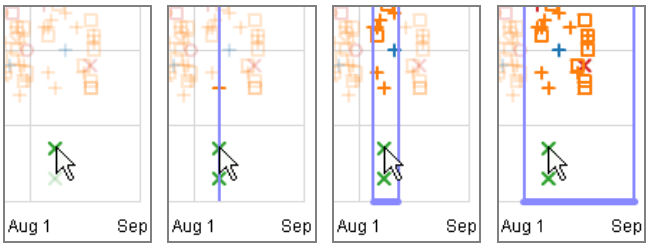


Figure 6. Relaxation of date ranges. One click selects an incident, two clicks selects the day on which the incident occurred, three clicks selects the entire week, four clicks selects the month. The images are cropped close-ups from the scatter plot in Figure 5b.

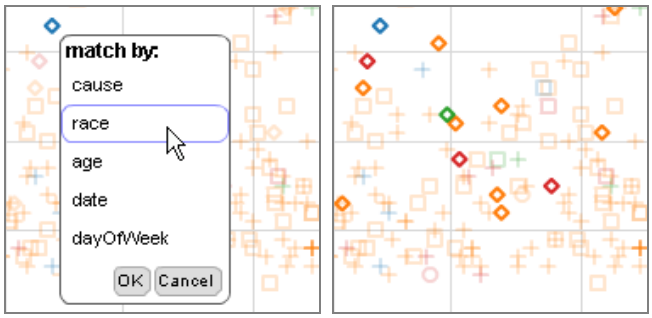


Figure 7. Relaxation by attributes. A click-and-hold action invokes a dialog for relaxing selections using one or more attributes. Above, a user selects all victims whose race matches the initial selection. The images are cropped close-ups from the scatter plot in Figure 5b.

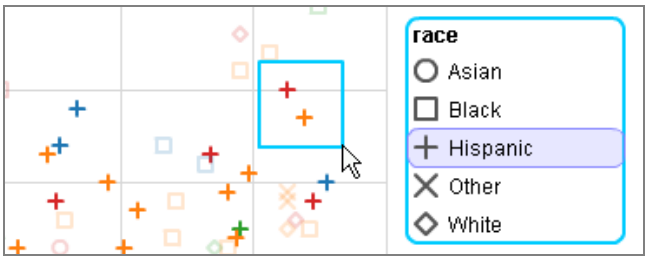


Figure 8. Range selection relaxed along the 'race' attribute. The generalized query selects all victims whose race matches that of any victim within the range bounds. Matching colors for the range selection and legend border indicate the relaxation relation. The image is a cropped close-up of the scatter plot in Figure 5b.

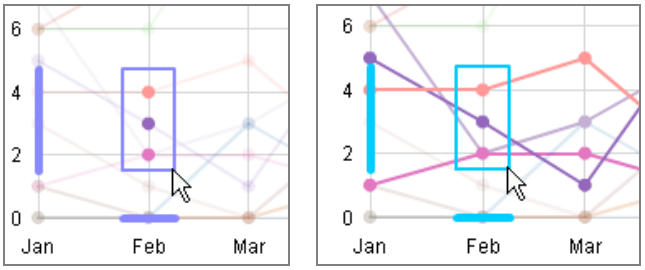


Figure 9. Time-searcher created by query relaxation. A user selects a range and relaxes the selection to create a tool that selects the time-series that pass through the range. Moving or resizing the range updates the relaxed query results. The images are cropped close-ups of a time-series of homicide counts by age group.

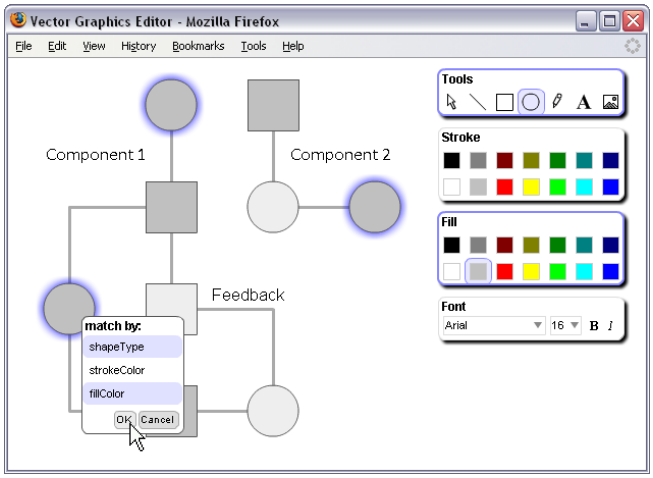


Figure 10. Vector Graphics Editor. Palettes on the right provide drawing operations. Our selection framework has been applied to enable generalized selection: here a user uses attribute relaxation to select all items with a matching shape and fill color.

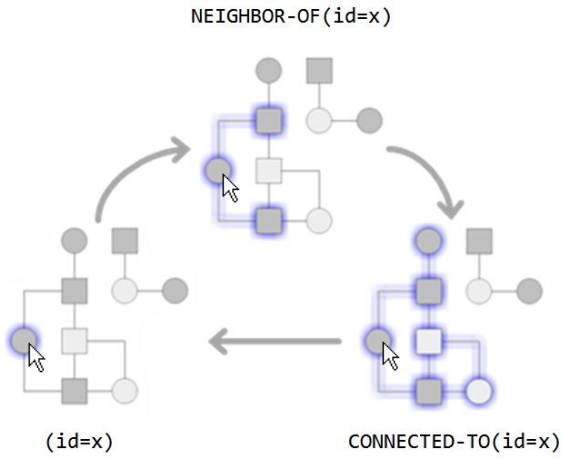


Figure 11. Query Relaxation of Networks. Connectors link visual items in a network. Query relaxation can be performed on the network structure. Here, one click selects an item, two clicks selects connected items, three clicks selects the connected component.

Query Generation

Our query builder converts selection interactions into queries. For example, shift-clicking two items in the geographic plot of Figure 5a generates a query of the form

```
SELECT * FROM data WHERE (id = 10556 OR id = 10548)
```

The query directly selects items via unique IDs (e.g., primary keys). (For simplicity, we show only the "WHERE" clause for the rest of the examples in this section.) Dragging a range creates a query of the form

```
(-118.371 ≤ longitude AND longitude ≤ -118.164)
                        AND
    (33.915 ≤ latitude AND latitude ≤ 34.089)
```

As specified by the schema mapping, our system replaces visual variables such as `x` and `y` with backing data variables such as `latitude` and `longitude`. Similarly, clicking on a legend generates a clause for the corresponding attribute value, e.g., `(cause = 'Gunshot')`.

Selection queries are represented internally as a tree of query operators, including nodes for literal values, variables, comparison operators, and Boolean logic. By default, query clauses generated in the same region of the interface are combined in an `OR` clause and the results are then combined by `AND` clauses. For example, creating two y-axis range selections and clicking the "Stabbing" legend entry in Figure 5b could result in the query clause:

```
((0 ≤ age AND age ≤ 10) OR (30 ≤ age AND age ≤ 40))
                        AND
            (cause = 'Stabbing')
```

Query Visualization

The query visualizer is responsible for visually conveying the structure of the query and indicating the items selected by the query. The query visualizer first traverses the query operator tree to construct an index of the various clause types (e.g., item selections, ranges, attribute selections, and nested queries). Individual query results can be highlighted using any visual highlighting effect, such as fade, blur, glow, and spotlight [17] effects. We use fade and blue transparent overlays as the default highlighting mechanism. For range clauses, the visualizer generates range brush controls, which users can interactively drag or resize (e.g., Figure 1). The query builder updates range clauses in response to these drag and resize actions. For attribute selections, the visualizer highlights each selected attribute in the legend or palette displays (Figure 3). For nested queries, such as the results of query relaxation, the visualizer highlights both the initial selection and the relaxed attributes using matching colors (Figure 8).

Query Relaxation

Query relaxation generalizes the query structure to create expanded selections based on the properties of interface items. We define a relaxation operation based on the semantic structure of the attributes of the underlying data and a policy for generating relaxed queries by traversing this structure. Here we consider three forms of relaxation and their corresponding semantic models.

Relaxation using Semantic Hierarchies

Hierarchies are a common structure for modeling a data domain. For example, we can hierarchically organize time into days, weeks, months, of years, as in the example shown in Figure 6. Similarly, we might hierarchically organize geographic regions into neighborhoods, cities, counties, and states. We can also generate semantic hierarchies in a data-driven fashion. For instance, an analyst might apply hierarchical clustering (c.f., [5]) to analyze her data, and use the resulting cluster trees to describe the data at different levels of abstraction.

To perform relaxation, we traverse these semantic hierarchies. With each relaxation step, the relaxer moves one level higher in the hierarchy and generates a query that selects all values in the current sub-hierarchy. For instance, relaxing date as in Figure 6 results in the query

```
SELECT * FROM data WHERE
  RELAX('date', 1, SELECT * FROM data WHERE id = x )
```

The relaxation is specified as a nested query. The result set of the initial selection is used as input to a relaxation operator. The relaxation operator takes two additional parameters: the name of the semantic structure to use and a parameter specifying a traversal policy. In the example above, the parameter `'date'` indicates that the semantic hierarchy for dates should be used, and the parameter '1' indicates that the query should be relaxed by one level of abstraction, to include all items that occurred on the same day as the initial selection. A level of '2' would relax the selection to all items in the same week. The relaxation operator outputs a new query clause that can be analyzed by the query visualizer. For example, in the example above the relaxation operator returns a comparison clause for the selected day, `(date = '2007-08-05')`.

Our system includes a general software interface for specifying hierarchical ontologies. We also include basic ontologies for common data types such as time (e.g., days, weeks, months, years) and numbers (e.g., relaxing by increasing powers of ten). Application designers can provide their own ontologies for custom data types, whether hand-crafted or data-driven.

Relaxation using Attributes

For some types of data and attributes, semantic structures are not available. When no explicit semantic structure is provided, our system assumes a "flat" hierarchy and relaxes the query to select all items with attributes exactly matching those contained in the initial selection.

The resulting relaxed queries select all items with some subset of attributes matching items contained in the initial selection, as in Figures 7-10. Consider Figure 5b. If the initial selection is a single object `(id = 10556)`, relaxation of the '`race`' attribute results in the query:

```
SELECT * FROM data WHERE
  (race IN SELECT race FROM data WHERE (id = 10556))
```

Because the hierarchy here is "flat," we can forego the relaxation operator. As before, the relaxed query is specified in terms of a nested sub-query. In the example above, the inner query returns the set of '`race`' attributes present in the result set of the initial selection `(id = 10556)`.

If we update the selection clause for the inner query, the result set of the relaxed query also updates. For instance, if we relax a range query rather than an item selection (Figures 8 and 9), we can interactively update the range bounds to refine the inner query, dynamically changing the input to the relaxation. If dynamic updates are not desired, we can collapse the query structure by evaluating the inner query to generate a query without nesting. To generate a "collapsed" query, we evaluate the relaxation clause, replacing it with a static clause such as `(race = 'Asian')`.

Relaxation using Networks

General network (graph) structures can also serve as semantic structures for query relaxation. Our internal query language includes traversal policies for such network structures. We have implemented traversals for selecting neighbors, connected components, ancestors or descendants (for DAG structures), or all items along the shortest-paths between items in the initial selection. Figure 11 depicts relaxations over a network in our vector graphics editor. As before, the relaxed query takes the form of a nested query:

```
SELECT * FROM data WHERE
  NEIGHBOR-OF( SELECT * FROM data WHERE id = x )
```

The formulation of this query is similar to the semantic hierarchy example, except that in this case the semantic structure and traversal policy are implicit for the "`NEIGHBOR-OF`" operator.

Configuration

Application designers can parameterize the query relaxation process by providing semantic structures and traversal policies for data attributes and specifying ordering constraints among attributes. Furthermore, a simple rule engine allows different attributes to be considered by the query relaxer based on the context of the selection. For example, the vector graphics editor contains a rule that enables relaxation of typeface attributes when the initial selection query only contains textbox items.

Query Reuse

One advantage of our framework is that it allows selections to be applied across changes of visual encodings. As shown in Figure 5, new query widgets can be generated as needed to convey the query structure. However, some expressions do not map from one view to another in a straightforward fashion. Consider a pair of 2D range selections, such as two latitude/longitude ranges. These selections result in a selection clause of the form

$$(R1_x \text{ AND } R1_y) \text{ OR } (R2_x \text{ AND } R2_y),$$

where $R1_x$ denotes the `x` component of the first selection, $R1_y$ denotes the `y` component, and similarly for `R2` and the second selection. If we change visual encodings, we might naïvely generate independent query histograms for these ranges: one for `x` and one for `y`, as in Figure 12a. However,

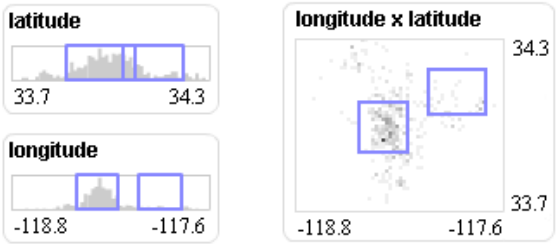


Figure 12. (a) Left: 1D components may incorrectly communicate multiple ranges. **(b) Right:** A scatter plot histogram for 2D ranges.

the default convention is to OR the clauses within a query component and AND the clauses from separate components. Thus, Figure 12a represents a different selection clause than intended, as the range clause groupings have been changed:

$$(R1_x \text{ OR } R2_x) \text{ AND } (R1_y \text{ OR } R2_y).$$

One solution to this problem is to use multivariate query widgets when confronted with multiple 2D ranges. Figure 12b shows a scatter plot histogram that depicts multiple 2D selection ranges and enables interactive refinement.

EVALUATION

To better understand the use of our selection techniques, we conducted a user study of a data visualization application supporting generalized selection. We asked subjects to perform both interpretation and authoring tasks using our selection framework. We were interested in how subjects interpreted visual representations of selection queries and if these interpretations changed in response to interactive use. We were also interested in whether subjects would use only direct clicking and dragging for selection or whether they would also use query relaxation. Finally, we wanted to investigate if the choice of selection mechanism had a subsequent impact on selection accuracy.

Sixteen subjects (11 female, 5 male) aged 18-27 (M = 21.2, SD = 2.43) participated in the study. All subjects were students at our university, studying diverse subjects such as biology, business, engineering, political science, and statistics. Subjects were recruited through the X-Lab (http://xlab.berkeley.edu), a research participation service.

Method

Subjects completed a set of tasks interpreting and authoring selections in a scatter plot visualization of homicides in Los Angeles (Figure 5b). The data set contained 627 data points noting the incident date and victim's age, race, and cause of death. The visualization showed a plot of date vs. age, with race and cause encoded by shape and color, respectively. The study consisted of three phases with 12 tasks each. Subjects required 30-45 minutes to complete the study.

In phase 1, subjects were shown visual selection queries and asked to describe, as completely as possible, the subset of the data highlighted in the view.

In phase 2, subjects first completed an interactive tutorial that provided descriptions of the selection operations and asked subjects to practice each selection before proceeding. For the study, relaxation was performed through repetition mediation only. Subjects could repeatedly click to cycle through relaxations of the cause, race, date, and age attributes. Subjects were then given text descriptions of subsets of the data and asked to make matching selections in the visualization. The provided text descriptions include:

- Victims who were exactly 60 years old.
- Victims killed by Blunt Force between March 1 and April 1.
- Victims killed by the same causes that killed any victims over 80 years old.

While users could complete all tasks by directly specifying the selection via clicking and dragging, they could also complete a subset of these tasks using query relaxation. Thus, subjects could apply relaxation as they saw fit. For example, in the third description above, subjects might either select the matching causes directly or select all victims over 80 and perform query relaxation.

The task in phase 3 was the same as in phase 1; subjects were again asked to interpret pre-defined selections, though the selections were different from those shown in phase 1. Afterwards, subjects completed a short survey.

In each phase, the selection cues were systematically varied to thoroughly cover the query structures expressible with our techniques, including 1D and 2D ranges, category selections, disjunctions within variables, and conjunctions across variables. In addition, multiple selections involved generalizing from a subset of the data. We used the same distribution of query structures in each phase of the study.

Results

We were interested in three primary questions. First, how did subjects interpret selections, and did interpretations change with interactive use? Second, which selection operations did subjects use to create selections, and what effect did they have on subjects' accuracy? Third, what did subjects think of the selection techniques?

Selection Interpretation

To analyze subjects' descriptions of observed selections in phases 1 and 3, we coded each response into one of four categories. *Structure-correct* responses accurately reported the structure of the selection query, including basic clauses and appropriate disjunctions and conjunctions. *Result-set-correct* responses did not report the query structure correctly, but specified criteria which resulted in the same query result set. Note that all *structure-correct* responses also produce correct results. We do not count the *structure-correct* responses in the *result-set-correct* category. *False-conjunctions* correctly identified each query sub-clause but combined then inaccurately. False conjunctions were prevalent when the stimulus involved a range generalization (e.g., selecting all categories contained within a range, as in Figure 8). Finally, we coded all other responses that failed to describe the query structure and results as *incorrect*. Table 1 shows the percentage of responses in each category.

Ideally, selections would be understood by users even if they have not previously used the software. To test if authoring selections changed how subjects interpreted selections, we compared the distributions of coded results from phase 1 (before interaction) and phase 3 (after interaction) across all tasks. We found no significant difference in the distribution of coded response types across study phases ($\chi^2(3, 362) = 2.26$, $p = 0.521$).

However, both phases included two tasks in which the selections were created by relaxing a range query along a

categorical attribute. These selections result in a potentially confusing display: a range brush is visible but selected items exist outside the range (see Figure 8). Unsurprisingly, 96% of all *false-conjunctions* occurred in these cases, in which subjects identified both the range and category criteria but did not understand the relation between the two.

To see if interpretation of such range relaxations was affected by interactive use, we analyzed just the tasks involving relaxation of a range selection and found a significant difference across phases ($\chi^2(3, 60) = 9.22$, $p = 0.027$). The number of *structure-correct* and *result-set-correct* responses increased in phase 3, with a corresponding drop in *false-conjunctions*. However, "correct" cases only accounted for 39% of responses, suggesting that even with exposure our subjects found relaxation of range selections hard to interpret. We also note that this analysis involves a relatively small amount of data, as each subject saw only two such range relaxations per phase.

Selection Authoring

To analyze the selection queries authored in phase 2, we similarly coded the responses into categories. In this case, we used only three categories: *structure-correct*, *result-set-correct*, and *incorrect* queries. Overall, subjects created selections that matched the text descriptions: 62% *structure-correct*, 20% *result-set-correct*, and 18% *incorrect*.

We were also interested in whether or not subjects would use query relaxation. Eleven of 16 users (68%) used multi-click relaxation to respond to a task, over a total of 28 tasks (15%). We hypothesized that subjects would be more accurate using query relaxation, and divided the responses into those that used relaxation and those that did not. We found a significant difference in the distribution of response types between the groups ($\chi^2(2, 192) = 11.45$, $p < 0.003$), with *structure-correct* responses comprising 89% of relaxation-generated responses, compared to 57% of responses made through other means.

We hypothesized that the difference might be due to individual differences— users who apply relaxation may be more advanced and perform better overall. To test this possibility, we divided the responses according to whether or not the subject used relaxation at any point. We found no significant difference between the two groups ($\chi^2(2, 192) = 4.15$, $p > 0.10$), suggesting that the subjects who used query relaxation were not significantly more accurate overall.

These results suggest that subjects may make more accurate selections when using query relaxation in tasks amenable to relaxation. However, we note that the nature of the task likely plays a crucial role in shaping subject performance.

Subject Preferences

At the end of the experiment, subjects were asked to rate the techniques presented within the experiment. Overall, subjects found the selection techniques helpful ($M = 3.75/5$, $SD = 0.45$) and did not find them confusing ($M = 1.75/5$, $SD = 0.77$). We also asked subjects to rate query relaxation. Overall, subjects rated query relaxation favorably ($M = 3.86/5$, $SD = 1.10$). However, the rating distribution was bi-modal, split between those who used query relaxation ($M = 4.36/5$, $SD = 0.50$, $N = 11$) and those who did not ($M = 2.40/5$, $SD = 0.55$, $N = 5$). The difference between groups was significant ($t(14) = 7.04$, $p < 0.001$). Overall, subjects' comments were positive ("it's very useful to find matching characteristics"), but also suggested usability improvements for the visualization application. For example, the fading effect applied to unselected items made it difficult to sequentially select ("shift-click") individual items.

Response Type	Phase 1	Phase 3	Average
structure-correct	73%	78%	75%
result-set-correct	3%	4%	4%
false-conjunctions	11%	7%	9%
incorrect	13%	11%	12%

Table 1. Responses in selection interpretation tasks.

DISCUSSION AND FUTURE WORK

Though the majority of subjects relaxed a range query to successfully complete a task, many had difficulties interpreting relaxed range queries when passively viewed. While most subjects correctly interpreted individual query components, they often did not recognize the generalization relation. This result suggests that simplified selections may be more appropriate for collaboration, as in the use of selection queries to specify annotations. Accordingly, we recommend "collapsing" nested query structures when using selections to communicate with a general audience, by evaluating the nested relaxation query. This limitation suggests future work in designing visual representations. Are there intuitive ways to indicate nested query structures without requiring an additional auxiliary interface?

Another avenue for future work is to further extend the query relaxation mechanisms. Other application domains might suggest new semantic models or traversal policies for relaxation. New relaxation types may be best expressed using additional input gestures, and expert users may want to configure the relaxation engine at runtime.

Finally, a primary motivation for developing our selection techniques is to support web-based collaboration around visualizations (e.g., [13, 26]). We are particularly interested in the use of selection queries for sharing interesting data items and regions during collective data analysis. One potential extension is to adapt our query visualizer and input techniques to support alternate forms of presentation and communicate more nuanced references. Another future direction lies in exploiting selection queries for data mining. Selection queries enable computational analysis of both the items and structure of users' references to data. Mining the various selections made by a collective might help automatically identify data items, attributes, and regions of interest to the community at large.

CONCLUSION

In this paper, we presented a framework that models selections of interface objects as declarative queries in a SQL-like language, capturing both the structure and content of a selection. Users create selection queries through direct manipulation; our system visualizes the structure of the query and highlights the results. Selection queries enable evolving selections over dynamic objects and streaming data. Selections can be reapplied across applications, without loss of structure. Users generalize these selections via interactive query relaxation, expanding their selections according to one or more attributes of interest. Results from our user study suggest that users successfully interpret and author selections using our system and that query relaxation may improve selection accuracy in amenable tasks.

ACKNOWLEDGMENTS

We would like to thank all the participants in our study. We also thank Joe Hellerstein for leading us to related work on query relaxation.

REFERENCES

1. Ahlberg, C. and Shneiderman, B. Visual Information Seeking: Tight Coupling Of Dynamic Query Filters With Starfield Displays. *Proc. ACM CHI '94*, pp. 313-317. 1994.
2. Becker, R.A. and Cleveland, W.S. Brushing Scatterplots. *Technometrics*, 29(2):127-142. 1987.
3. Brennan, S.E. How conversation is shaped by visual and spoken evidence. In Trueswell and Tanenhaus (eds.), *Approaches to studying world-situated language use: Bridging the language-as-product and language-as-action traditions*, pp. 95-129. MIT Press, 2005.
4. Chen, H. Compound Brushing. *Proc. IEEE InfoVis '03*, pp. 181-189. Oct 2003.
5. Chu, W. W., Yang, H., Chiang, K., Minock, M., Chow, G., and Larson, C. CoBase: A Scalable and Extensible Cooperative Information System. *Journal of Intelligence Information Systems*, 6(2):223-259. 1996.
6. Chu, W. W. and Liu, S. CoXML: Cooperative XML Query Answering. In B. Wah (ed.), *The Encyclopedia of Computer Science and Engineering*. John Wiley & Sons Inc, 2007.
7. Clark, H.H. and Brennan, S.E. Grounding in Communication. In L. B. Resnick, R. M. Levine, and S. D. Teasley (eds.), *Perspectives on socially shared cognition*, pp. 127-149. American Psychological Association, 1991.
8. Clark, H. H. Pointing and placing. In S. Kita (ed.), *Pointing. Where language, culture, and cognition meet*, pp. 243-268. Erlbaum, 2003.
9. Derthick, M., Kolojejchick, J. A., and Roth, S. An Interactive Visual Query Environment for Exploring Data. *Proc. ACM UIST '97*, pp. 189-198. Oct 1997.
10. Fishkin, K., and M. Stone. Enhanced Dynamic Queries via Movable Filters. *Proc. ACM CHI '95*, pp. 415-420. May 1995.
11. Gaasterland, T. Cooperative Answering through Controlled Query Relaxation. *IEEE Intelligent Systems*, 12(5):48-59. Sep-Oct 1997.
12. The Gapminder World 2006. http://tools.google.com/gapminder
13. Heer, J., Viégas, F., Wattenberg, M. Voyagers and Voyeurs: Supporting Asynchronous Collaborative Information Visualization. *Proc. ACM CHI '07*, pp. 1029-1038. Apr 2007.
14. Hochheiser, H., Shneiderman, B. Dynamic query tools for time-series data sets: Timebox widgets for interactive exploration. *Information Visualization*, 3:1-18. 2004.
15. Huh, S.-Y., Moon, K.-H., Lee, H. A data abstraction approach for query relaxation. *Information and Software Technology*, 42:407-418. 2000.
16. Igarashi, T., Matsuoka, S., Kawachiya, S., and Tanaka, H. Interactive Beautification: A Technique for Rapid Geometric Design. *Proc. ACM UIST '97*, pp. 105-114. 1997.
17. Khan, A., Matejka, J., Fitzmaurice, G., Kurtenbach, G. Spotlight: directing users' attention on large displays. Proc. *ACM CHI '05*, pp. 791-798. Apr 2005.
18. Los Angeles Times Homicide Map, 2007. http://www.latimes.com/homicidemap/
19. Livny, M., R. Ramakrishnan, K. Beyer, G. Chen, D. Donjerkovic, S. Lawande, J. Myllymaki, and K. Wenger. DEVise: Integrated Querying and Visual Exploration of Large Datasets. *Proc. ACM SIGMOD '97*, pp. 310-312. May 1997.
20. Mankoff, J., Hudson, S.E., and Abowd, G.D. Interaction techniques for ambiguity resolution in recognition-based interfaces. *Proc. ACM UIST '00*, pp. 11-20. 2000.
21. Martin, A. R., Ward M. O. High Dimensional Brushing for Interactive Exploration of Multivariate Data. *Proc. IEEE Visualization '95*, pp. 271-278. Nov 1995.
22. North, C., Shneiderman, B. Snap-Together Visualization: A User Interface for Coordinating Visualizations via Relational Schemata. *Proc. Advanced Visual Interfaces '00*, pp. 128-135. May 2000.
23. Olston, C., Stonebraker, M., Aiken, A., Hellerstein, J. M. VIQING: Visual Interactive Querying. *Proc. IEEE Visual Languages '98*, pp. 162-169. Sep 1998.
24. Saund, E., Fleet, D., Larner, D., Mahoney, J. Perceptually-supported image editing of text and graphics. *Proc. ACM UIST '03*, pp. 183-192. 2003.
25. Stolte, C., Tang, D., and Hanrahan, P. Polaris: A System for Query, Analysis and Visualization of Multi-dimensional Relational Databases. *IEEE Trans. on Visualization and Computer Graphics*, 8(1):52-65. Jan 2002.
26. Viégas, F., Wattenberg, M., van Ham, F., Kriss, J., and McKeon, M. Many-Eyes: A Site for Visualization at Internet-Scale. *Proc. IEEE InfoVis '07*, pp. 1121-1128. Oct 2007.
27. Weaver, C. Building Highly-Coordinated Visualizations In Improvise. *Proc. IEEE InfoVis '04*, pp.159-156. 2004.
28. World Bank World Development Indicators, 2007. http://devdata.worldbank.org/data-query/