A Flexible Architecture for Filter/Flow-Based Visual Querying

Florian Haag · Steffen Lohmann · Thomas Ertl

Based on the various extensions to the Filter/Flow concept, we have defined a unified architecture for Filter/Flow-based visual querying systems that can be used in user experiments. This poster describes the core components and provides an overview of possible research questions that can be investigated by taking advantage of the extension points in the described flexible architecture.

The Filter/Flow Concept

With a growing amount of multivariate data, finding and identifying a particular piece of information requires increasingly complex search queries. The Filter/Flow concept is a powerful visual metaphor for representing such complex Boolean filter expressions. It was originally presented by Young and Shneiderman [7] and has been adapted and extended in various ways since (e.g., [2, 4, 5, 6]).

Filter/Flow graphs can be used to express compound Boolean expressions:

\[ a \land (b \land (c \lor d) \lor e) \land f \]

In other words, sequential chains are interpreted as conjunctions and parallel paths are interpreted as disjunctions. Negated terms are shown by using inverted colors. The edges of the graph can be seen as a metaphor for flowing water. The water represents the data that gets filtered by the filter nodes. Hence, if there is at least one path that connects the initial edge with the final edge, the complete filter expression evaluates to true.

Visual models can be reused for several front-ends if the front-ends use the same graphics toolkit. For instance, this allows experiments on desktop computers and on Microsoft PixelSense tabletop displays when using a WPF-based visual model.

Due to the exchangeable data source, a wide variety of scenarios with different groups of expert users can be considered in user studies. Virtually all data sources that accept some form of filter expression are suitable, as an expression factory that outputs the appropriate query format can be implemented.

By exchanging the expression generator implementation, various methods of optimizing query expressions can be tested. Moreover, it is possible to investigate differently prioritized filters if the expression generator omits filters below a threshold priority.

The expression factory is chosen based on the allowable query string formats of the current data source. It has to provide query string representations of different kinds of Filter/Flow graph/expressions tree structures. Current implementations:

- SPARQL for accessing RDF triple stores such as DBpedia [1]
- SQL for relational databases
- ...