D 4.3: Refined Semantic Enrichment, Semantic Integration and Visual Analytics Interfaces

Summary: Task 4.3 in Phase II of the CODE project is concerned with ensuring the usability of the developed visual analytics interfaces. The task includes the enhancement of interfaces and interaction paradigms for analyzing, navigating, browsing and visualizing Linked Data. Formative evaluations of the components and user interfaces have been conducted through continuous usability testing with friendly users. This deliverable reports the outcome of task 4.3 in Phase II, which ends at month 20 of the project. The final outcome of task 4.3, including in-depth user evaluations, will be reported in Deliverable 4.4 in month 24.

Keywords: Semantic Enrichment, Semantic Integration, Coordination Framework, RDF, OWL, Visualizations, Visual Analytics, Linked Data, Web

<table>
<thead>
<tr>
<th>Project Acronym</th>
<th>CODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grant Agreement number</td>
<td>296150</td>
</tr>
<tr>
<td>Project Title</td>
<td>Commercially Empowered Linked Open Data Ecosystems in Research</td>
</tr>
<tr>
<td>Date</td>
<td>2014-01-02</td>
</tr>
<tr>
<td>Nature</td>
<td>P (Prototype)</td>
</tr>
<tr>
<td>Dissemination level</td>
<td>PU (Public)</td>
</tr>
<tr>
<td>WP Lead Partner</td>
<td>Know-Center</td>
</tr>
<tr>
<td>Revision</td>
<td>4 (Final)</td>
</tr>
<tr>
<td>Authors</td>
<td>Patrick Hoefler, Belgin Mutlu, Vedran Sabol</td>
</tr>
</tbody>
</table>

Statement of originality: This document contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

This document reflects only the author's views and the European Community is not liable for any use that might be made of the information contained herein. © CODE Consortium, 2013
Project Officer & Project Coordinators

<table>
<thead>
<tr>
<th>Role</th>
<th>Name</th>
<th>Organization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Officer</td>
<td>Stefano Bertolo</td>
<td>European Commission</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Directorate-General Communications Networks,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Content and Technology (DG CONNECT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unit Data Value Chain (G3)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10, Rue Robert Stümper</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2920 Luxembourg</td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="mailto:stefano.bertolo@ec.europa.eu">stefano.bertolo@ec.europa.eu</a></td>
</tr>
<tr>
<td>Project Coordinator</td>
<td>Stefanie Lindstaedt</td>
<td>Know-Center GmbH</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Inffeldgasse 13, 8010 Graz, Austria</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+43 316 873 30800</td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="mailto:slind@know-center.at">slind@know-center.at</a></td>
</tr>
<tr>
<td>Scientific Coordinator</td>
<td>Michael Granitzer</td>
<td>University of Passau</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Innstrasse 33a, 94032 Passau, Germany</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+49 851 509 3305</td>
</tr>
<tr>
<td></td>
<td></td>
<td><a href="mailto:michael.granitzer@uni-passau.de">michael.granitzer@uni-passau.de</a></td>
</tr>
</tbody>
</table>

Document Revision History

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Author</th>
<th>Organization</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2013-12-16</td>
<td>Patrick Hoefler,</td>
<td>Know-Center</td>
<td>Draft</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Belgin Mutlu</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2013-12-19</td>
<td>Patrick Hoefler</td>
<td>Know-Center</td>
<td>Release Candidate</td>
</tr>
<tr>
<td>3</td>
<td>2013-12-20</td>
<td>Patrick Hoefler,</td>
<td>Know-Center</td>
<td>Final changes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Belgin Mutlu</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2014-01-02</td>
<td>Vedran Sabol</td>
<td>Know-Center</td>
<td>Proof reading</td>
</tr>
</tbody>
</table>
# Table of Contents

1. Introduction .................................................................................................................. 5
2. CODE Query Wizard ..................................................................................................... 7
   2.1 Overview .................................................................................................................. 7
   2.2 Technical Details ..................................................................................................... 8
   2.3 Interfaces to Other CODE Components ................................................................. 11
      2.3.1 CODE Q&A Portal ......................................................................................... 11
      2.3.2 CODE Data Extractor (WP3) .......................................................................... 11
      2.3.3 MindMeister Integration .................................................................................. 11
      2.3.4 Mendeley Integration ....................................................................................... 11
3. CODE Visualization Wizard ............................................................................................ 13
   3.1 Overview .................................................................................................................. 13
   3.2 Describing Visualizations ......................................................................................... 14
   3.3 Suggesting Visualizations and Mappings ................................................................. 14
   3.4 Interactive Visual Analysis ....................................................................................... 18
      3.4.1 Mouse-Over ..................................................................................................... 18
      3.4.2 Brushing .......................................................................................................... 21
      3.4.3 Aggregation ...................................................................................................... 23
      3.4.4 Filtering ............................................................................................................ 23
      3.4.5 Static Images of Visualizations ........................................................................ 28
   3.5 MindMeister Integration ........................................................................................... 28
   3.6 Mendeley Integration ............................................................................................... 29
3.7 Conceptual Architecture ............................................................................................. 29
   3.7.1 Backend ............................................................................................................. 30
   3.7.2 Process View ....................................................................................................... 31
4. Evaluation ....................................................................................................................... 33
5. References ....................................................................................................................... 34
Table of Figures

Figure 1: Overview of WP4 of the CODE project .......................................................... 6
Figure 2: Current results page of the CODE Query Wizard ............................................. 7
Figure 3: Simple interface for aggregating an RDF Data Cube via the CODE Query Wizard .............................................................................................................................................. 8
Figure 4: Current entry page of the CODE Query Wizard .................................................. 9
Figure 5: Overview of the completed SPARQL queries necessary to display the Linked Data......................................................................................................................................................... 10
Figure 6: Export of the displayed Linked Data as JSON-LD .............................................. 10
Figure 7: MindMeister mind map created from Linked Data via the CODE Query Wizard (search was “Toy Story”) .................................................................................................................... 12
Figure 8: Five suggested charts, shown as enabled buttons, for a RDF Data Cube with three components .......................................................................................................................... 15
Figure 9: Mapping combinations changed for scatter plot .................................................. 16
Figure 10: The automatically generated parallel coordinates visualization of a 9-dimensional RDF Data Cube (PAN 2012 challenge results) ..................................................................................... 17
Figure 11: Scatterplot matrix for a 3-dimensional data set (PAN 2012 challenge results) .......................................................................................................................................................... 17
Figure 12: Interactive Pie and Bar Chart (PAN 2009 challenge results) ................................ 18
Figure 13: Interactive Geographic Map (funding per country) ............................................. 19
Figure 14: Interactive Scatter Plot / Bubble Chart .................................................................. 19
Figure 15: Interactive Line Chart ......................................................................................... 20
Figure 16: Interactive Grouped Bar Chart ............................................................................. 20
Figure 17: Brushing in multiple coordinated views with jQuery table as the base chart (funding per country) ........................................................................................................................ 21
Figure 18: Brushing in multiple coordinated views with Parallel Coordinates as the base chart ......................................................................................................................................................... 22
Figure 19: Base cube (left), aggregated cube, grouped by year, displaying the max value (right) ............................................................................................................................................. 23
Figure 20: Filtering of measures ......................................................................................... 24
Figure 21: Filtering a cube using the URI filter .................................................................... 25
Figure 22: Filtering a cube using the numeric filter ............................................................. 26
Figure 23: Filtering a cube using the date filter .................................................................... 27
Figure 24: Preview image for Streamgraph generated with PhantomJS .............................. 28
Figure 25: MindMeister mind map created from Linked Data via the Vis Wizard .............. 29
Figure 26: Using Mendeley Desktop to visualize tables extracted from a research paper ..................................................................................................................................................... 29
Figure 27: Conceptional architecture overview of the CODE Visualization Wizard .......... 30
Figure 28: Routines for matching vocabularies: chart suggestion (left) and suggestion of valid mapping variants (right) ............................................................................................................. 32
1. Introduction

Linked Data has become an essential part of the Semantic Web. A lot of Linked Data is already available in the Linked Open Data cloud, which keeps growing due to an influx of new data from research and open government activities. However, it is still quite difficult to access this wealth of semantically enriched data directly without having in-depth knowledge about SPARQL and related semantic technologies.

WP4 of CODE therefore deals with the development of web-based visual analytics user interfaces for Linked Data. This means that it should be possible for people who don’t know anything about RDF and SPARQL to access information that is already available but difficult to access in the Linked Open Data Cloud.

WP4 consists of two major components:

- The CODE Query Wizard helps people select relevant data from the Linked Data Cloud – to be more exact, from publicly available SPARQL endpoints. The resulting data is presented in an easy-to-use web-based interface that looks and feels similar to current spreadsheet applications. For advanced data manipulation, e.g. aggregations, the data is converted to the RDF Data Cube format. The prototype is publicly available at http://code.know-center.tugraz.at/search

- Once the users have selected the relevant data, the CODE Visualization Wizard (or Vis Wizard for short) supports them in creating visualizations to make the data easier to understand. Multiple visualizations can be combined to form a visual analytics interface. In order to facilitate this step, not only the data but also the visualizations have semantic descriptions in the form of OWL ontologies. The prototype is publicly available at http://code.know-center.tugraz.at/vis

At the time of writing, the Query Wizard and the Vis Wizard are being integrated into the CODE Question & Answer Portal and will become crucial components of this data marketplace.

The following figure combines a conceptual overview of how the CODE Query Wizard and the CODE Visualization Wizard interact.
Figure 1: Overview of WP4 of the CODE project
2. CODE Query Wizard¹

2.1 Overview

The Linked Open Data cloud provides an impressive wealth of semantically enriched, openly available Linked Data. However, this Linked Data is basically only accessible for experts in semantic technologies who know how to write SPARQL queries. And even for those who know how to use SPARQL, it can be quite laborious at times, especially while trying to explore an unknown SPARQL endpoint.

Therefore, the goal of the CODE Query Wizard [Hoefler et al. 2013, Hoefler 2013] is to provide an easy-to-use interface for accessing Linked Data. It should be suitable for non-expert users without any prior knowledge of SPARQL or other semantic technologies.

The working hypothesis for the CODE Query Wizard is: There are not that many people who speak SPARQL and are familiar with graph structures. On the other hand, many people know spreadsheet applications. Therefore, the idea is to develop a web-based tool that brings the graph structure of Linked Data into tabular form and provides easy-to-use interaction possibilities for filtering and exploring Linked Data by using metaphors and techniques the users already know.

Figure 2: Current results page of the CODE Query Wizard

¹ http://code.know-center.tugraz.at/search
Our prototype also supports the current working draft of W3C's RDF Data Cube vocabulary\(^2\), which provides a semantic framework for expressing statistical datasets as Linked Data. Datasets that comply with the RDF Data Cube standard can easily be displayed, filtered, and explored using the CODE Query Wizard. Hence, aggregated data from WP3 can be enriched, extended, and refined through the Query Wizard.

### 2.2 Technical Details

The CODE Query Wizard is a purely web-based system. It uses Python 2.7 on the back end and HTML5, CSS3, and JavaScript — compiled from CoffeeScript — on the front end.

Unfortunately, full-text search is sorely lacking from the current SPARQL specification\(^3\), which is why certain SPARQL endpoints have come up with workaround solutions. Therefore, only Virtuoso, OWLIM, and bigdata are currently supported as SPARQL endpoints by the CODE Query Wizard. However, since one of our prototype's design philosophies is to use Semantic Web standards such as SPARQL wherever possible, support for other suitable endpoints can be added with minimal effort.

Currently, the CODE Query Wizard provides a list of 37 searchable SPARQL endpoints. Unfortunately only few of them provide reliable and satisfactory performance. We are, however, in the process of providing a simple API so that the CODE Query Wizard can be used to access any publicly reachable and technologically supported SPARQL endpoint.

The CODE Query Wizard also makes use of certain SPARQL 1.1 features, especially the aggregation functions. The \texttt{COUNT()} function is used for displaying the number of results for a given query. In order to let users calculate simple aggregate functions for a given dataset, the functions \texttt{AVG()}, \texttt{MAX()}, \texttt{MIN()} and \texttt{SUM()} are needed (see Figure 5). Therefore it is of vital importance that the used SPARQL endpoint supports the SPARQL 1.1 feature set.

\[\text{http://www.w3.org/TR/vocab-data-cube/}\]

\[\text{http://www.w3.org/TR/sparql11-overview/}\]
The CODE Query Wizard offers two entry points: Users can either initiate a keyword search, or they can select any available dataset, represented as an RDF Data Cube (see Figure 4). In both cases, the results are presented in tabular form, similar to what the users are used from spreadsheet applications. They can choose which columns (i.e. RDF predicates) they are interested in, and they can set filters to narrow down the displayed data.

For expert users, it is also possible to display the generated SPARQL queries (see Figure 5) and export the displayed data in the JSON-LD format (see Figure 6).
**Figure 5:** Overview of the completed SPARQL queries necessary to display the Linked Data

**Figure 6:** Export of the displayed Linked Data as JSON-LD
2.3 Interfaces to Other CODE Components

The CODE Query Wizard currently features 4 interfaces to other CODE components:

- CODE Q&A Portal
- CODE Data Extractor (WP3)
- MindMeister Integration
- Mendeley Integration

2.3.1 CODE Q&A Portal

The CODE Question & Answer Portal is the central data marketplace and integration hub of the CODE project. The CODE Query Wizard provides an important end-user facing tool for working with existing datasets or creating new datasets from Linked Data already available in public SPARQL endpoints. Users can integrate data curated with the Query Wizard into answers on the Q&A Portal and thereby provide context for the data, making it even more valuable.

2.3.2 CODE Data Extractor (WP3)

The CODE Data Extractor is developed within WP3. It takes the tables generated by the CODE Query Wizard and turns them into RDF Data Cubes. For the end users, this happens in the background without any interaction. However, there is also an "Expert Mode" available, which offers more flexibility in case of problems with the data.

For more information about the Data Extractor, please refer to D3.3 Refined Federated Querying and Aggregation Service.

2.3.3 MindMeister Integration

The MindMeister mind mapping service is used to turn the results table into a nice mind map (see Figure 7). This feature is especially useful for presenting a quick overview over a certain topic with only a handful of results.

2.3.4 Mendeley Integration

It is possible to log in to the CODE Query Wizard using an existing Mendeley account. This feature is important for two reasons:

1. The Mendeley User ID acts as a central identifying mechanism used by all CODE components. This facilitates the simple integration of all components, especially in the context of the CODE Q&A Portal.
2. Another important topic covered by the CODE project is provenance: Which user has created or modified a certain dataset, and what was the original source? Provenance is mainly covered within WP3, but the CODE Query Wizard provides information about the sources of the data and the users that interact with it.
Figure 7: MindMeister mind map created from Linked Data via the CODE Query Wizard (search was “Toy Story”)
3. CODE Visualization Wizard

To support users in the visual analysis of Linked Data, the CODE project provides another wizard, called the CODE Visualization Wizard (or Vis Wizard for short).

3.1 Overview

The Vis Wizard [Mutlu et al. 2014] relies on a prior extraction and organization of the data using the RDF Data Cube vocabulary. Once the data is represented in a RDF Data Cube, the Vis Wizard automatically suggests suitable and feasible visualizations for the RDF Data Cube. Also, based on the content of the Cube, the Vis Wizard defines meaningful mapping combinations of multi-dimensional data onto visualizations.

In order to automate the process of generating and proposing visualizations, it is necessary to integrate the visualization aspect into the Semantic Web domain. Therefore we represent the complete knowledge about the visualizations in an OWL ontology.

Data can be passed to the Vis Wizard in two ways:

1. CODE Query Wizard via the CODE Data Extractor
2. Public SPARQL endpoint containing RDF Data Cubes

In the first case, the CODE Data Extractor receives generic RDF data from the CODE Query Wizard and turns it into an RDF Data Cube. In the second case, the data is already stored as RDF Data Cubes.

The RDF Data Cube vocabulary is developed by the W3C to represent statistical data (e.g. the research results from tables in a publication) in a common format. The data is represented as a collection of so-called observations, each consisting of a set of dimension and measures. The dimensions identify the observation, and the measures are related to concrete values. More details about RDF Data Cube can be found in our previous deliverable D4.1: Semantic Descriptions for Visual Analytics Interfaces.

Furthermore, the user is supported by the Vis Wizard to organize, manage, refine and inspect the visualized data with the following methods:

1. Mouse-over inspections
2. Brushing in multiple coordinated views
3. Filtering
4. Aggregation

Data manipulations techniques like aggregation and filtering, and interactive techniques such as coordinated brushing and mouse-over, allow the user to refine the visual representation of the data providing powerful visual analysis functionality.

---

3.2 Describing Visualizations

To support the process of mapping visualizations to data described in RDF Data Cube vocabulary, we developed a Visual Analytics vocabulary8 (or VA vocabulary for short) that describes visualizations semantically as an OWL ontology. Our semantic description strictly focuses on describing the visual encoding process; hence we represent visualizations in terms of their visual channels (e.g. axes, colors, item sizes etc.).

The visual channels are characterized in our ontology with following attributes:

1. **Data type**: Defines a set of primitive data types a visual channel can support
2. **Occurrence**: Defines the cardinality of a visual channel
3. **Persistence**: Defines whether a visual channel is mandatory part of the concrete visualization or not

More details about the Visual Analytics vocabulary can be found in our previous deliverable D4.1- Semantic Descriptions for Visual Analytics Interfaces.

3.3 Suggesting Visualizations and Mappings

The mapping between the RDF Data Cube vocabulary and the VA vocabulary is a relation from dimensions and measures in the former to the corresponding visual channels of a visualization in the latter. This relation is valid only if the data types of the cube components and visual channels are compatible.

**Data type compatibility**: Data type compatibility in our context means having exactly the same primitive data types, both conforming to the XSD data type definitions 9.

Beyond data type compatibility, a valid mapping needs to account for structural compatibility, since visualizations from VA vocabulary may have fixed or varying number of visual channels.

**Structural compatibility**: The instantiation of dimensions and measures in the RDF Data Cube is unbounded. That is, we can define observations with arbitrary dimensions and measures. Therefore the possible instantiation patterns (i.e. in the format `dimensions: measure`) for each RDF Data Cube are:

1. 1:1
2. 1:n
3. m:1
4. m:n

In order to find a valid mapping, we have to find in the VA vocabulary visualizations with the same instantiation patterns. The attributes of visual channels (persistence, occurrence, data type) form the basis to prove both types of the compatibility.

For a given RDF Data Cube, the mapping algorithm [Mutlu et al. 2013] returns either a list of visualization candidates together with concrete mappings, or nothing. For example, if the

---

8 http://code.know-center.tugraz.at/static/ontology/visual-analytics.owl
9 http://www.w3.org/TR/2001/REC-xmlschema-2-20010502
dataset contains two dimensions with string as data type and one measure with integer or decimal as data type, the algorithms of the CODE Visualization Wizard will suggest only the charts (see Figure 8) having three visual channels with data types:

1. string
2. string
3. decimal

![Five suggested charts, shown as enabled buttons, for a RDF Data Cube with three components](image)

The user can choose from the suggested charts by clicking on the corresponding button, the visualization will be automatically generated. For a given data set multiple visualizations can be generated and viewed side by side. Supported visualizations currently include:

1. Pie chart
2. Line chart
3. Bar chart
4. Grouped bar chart
5. Bubble chart / scatterplot
6. Stream graph
7. Geovisualisation
8. Scatterplot matrix
9. Parallel coordinates
10. Table view

Whenever there are more dimensions or measures with the same data types that correspond to the same visual channel, there are various mapping variations for a single visualization. In this case, the Wizard creates a candidate table including all possible combinations. For each such combination, a specific visualization is automatically generated. Once the visualization
is selected the user is free to modify the mapping combination. For example, in the scatter plot in Figure 9, the left images shows countries assigned to the x-axis and years assigned to colors, while the right images shows countries assigned to colors and the years assigned to the x-axis.

The following pseudo-code shows the mapping algorithm:

```
Data: RDF Data Cube
Result: set(mapping suggestions)
get visualisation candidates;
get observation components;
while visualisation candidates exist do
    instantiate visual channels;
    generate all combinations for datatypes of visual channels;
    while datatype combinations exist do
        map combination to instantiated visual channels;
        pack mapping configuration;
        if ((occurrence matches) and (persistence matches)) and (type matches) then
            add to mapping suggestion set;
        else
            throw invalid mapping configuration;
```

Algorithm 1: Simplified algorithm for determining feasible mapping suggestions

The `occurrence` attribute (see the VA vocabulary) identifies whether a visual channel can be instantiated only once (e.g. bar chart x-axis and y-axis) or multiple times (e.g. parallel coordinates x-axis). There are two different values for this attribute: `one` and `many`. The occurrence `many` is used for visualizing high-dimensional RDF Data Cubes. In contrast, the occurrence `one` defines a fixed cardinality. Another important role of the occurrence is that it reveals the structure of the concrete visualization.

The visualizations with the occurrence `many` are Parallel Coordinates (see Figure 10), Scatterplot Matrix (see Figure 11) and Table View.
Commercially Empowered Linked Open Data Ecosystems in Research

D 4.3: Refined Semantic Enrichment, Semantic Integration and Visual Analytics Interfaces

Figure 10: The automatically generated parallel coordinates visualization of a 9-dimensional RDF Data Cube (PAN 2012 challenge results)

Figure 11: Scatterplot matrix for a 3-dimensional data set (PAN 2012 challenge results)
3.4 Interactive Visual Analysis

The visual analysis starts with the suggestion of suitable visualizations whereby the user selects a visualization to generate a mapping.

All charts that are integrated in the Vis Wizard are interactive. Through interaction with the visualizations a better understanding of the represented information can be achieved. Interactivity supports users to explore and find so desired or relevant information in research knowledge in a short time.

Currently, two interactive operations with visualizations are supported:

1. Mouse over inspection
2. Brushing over multiple coordinated views

3.4.1 Mouse-Over

Mouse over inspection highlights a visual element, such as a bar or a line, and displays additional information (typically in a tool tip). In the following, we present the visualizations with mouse over functionality.
Figure 13: Interactive Geographic Map (funding per country)

Figure 14: Interactive Scatter Plot / Bubble Chart
Commercially Empowered Linked Open Data Ecosystems in Research

D 4.3: Refined Semantic Enrichment, Semantic Integration and Visual Analytics Interfaces

Figure 15: Interactive Line Chart

Figure 16: Interactive Grouped Bar Chart
3.4.2 Brushing

The idea behind brushing is to select a subset of data in one visualization (the “base chart”) and see the distribution of the same data in one or more other visualizations (see Figure 17 and Figure 18). All integrated visualizations in the Vis Wizard support coordinated brushing over multiple views. Visually, the data elements selected by the brush retain their original color, while data elements not selected by the brush are shown in gray.

Figure 17: Brushing in multiple coordinated views with jQuery table as the base chart (funding per country)
Figure 18: Brushing in multiple coordinated views with Parallel Coordinates as the base chart
3.4.3 Aggregation

The aim of the aggregate function is to group the values of multiple rows together to form a single value. The following aggregation functions are currently included in the Vis Wizard: Average, Count, Maximum, Minimum, and Sum.

Since the numbers of the RDF Data Cube components will be changed after the aggregation, the mapping algorithm determines new visualizations and mapping suggestions for the aggregated RDF Data Cube. After the aggregation, the original and the aggregate cube are displayed together (see Figure 19).

3.4.4 Filtering

The Vis Wizard supports two different filter functions:

- Filtering of measures
- Filtering of the data range

Filtering of measures

Once a measure is removed the base RDF Data Cube has a measure less. Hence the mapping algorithm suggests appropriate mapping and visualizations for this “new” RDF Data Cube (see Figure 20).
Figure 20: Filtering of measures
Filtering of the data range (URI, numeric, date)

The Vis Wizard supports the following filtering functions:

- Filtering by URI
- Filtering by numeric entities
- Filtering by date

We use the CODE Query Wizard to set these filters functions.

Filtering by URI

![Figure 21: Filtering a cube using the URI filter](image)
Consider the following example: We have a RDF Data Cube, whose content is the funding amounts to participants (countries) in FP7-ICT projects in the EU. In case the user is only interested in the amounts for Austria, the user only has to set the URI filter in the Query Wizard on Austria (see Figure 21).

**Filtering numeric entities**

In case we have numeric data, we can use the numeric filter in Query Wizard to select a particular data interval. Only the selected data interval from the RDF Data Cube is visualized, whereby either the dimensions or the measures will be removed or changed (see Figure 22).

![Figure 22: Filtering a cube using the numeric filter](image-url)
Filtering date entities

In case we have date information, we can use the date filter in the Query Wizard to select a particular date interval (see Figure 23).
3.4.5 Static Images of Visualizations

Generating a static image of a specific interactive visualization can be useful for integrating the image in other applications. We use PhantomJS\(^\text{10}\) to generate these images (see Figure 24).

![Streamgraph generated with PhantomJS](http://phantomjs.org)

**Figure 24:** Preview image for Streamgraph generated with PhantomJS

### 3.5 MindMeister Integration

[![MindMeister Mind Map](http://phantomjs.org)]

\(^{10}\) [http://phantomjs.org](http://phantomjs.org)
The MindMeister mind mapping service is used to turn the generated visualization into a nice mind map. Each visualization contains a static preview image and a link back to the interactive version of the chart in the Vis Wizard (see Figure 25).

### 3.6 Mendeley Integration

Mendeley has integrated the Vis Wizard into a special CODE edition of their Mendeley Desktop to automatically visualize statistics after their extraction by the CODE Table Extractor (see Figure 26).

![Figure 26: Using Mendeley Desktop to visualize tables extracted from a research paper](image)

### 3.7 Conceptual Architecture

In the following, the conceptual architecture of the CODE Visualization Wizard is described from the developer's and from the process point of view (see Figure 27).
3.7.1 Backend

The backend consists of the following components:

- **Data Endpoint**: Provides generic RDF data from Linked Open Data Cloud as well as data represent as RDF Data Cubes.

- **RDF Visualization Model**: For the description of the visualizations and their visual channels as well as for the description of the mapping from RDF Data Cube observations into the visualizations, a new Visual Analytic vocabulary has been defined (RDF Chart Model). In our previous deliverable D4.1: Semantic Descriptions for Visual Analytics Interfaces\(^\text{11}\) we provide a detailed description of the VA vocabulary.

- **RDF Parser**: Parses all semantic descriptions as well as the actual data structured as RDF Data Cubes.

- **Mapping Proposal**: The Visualization Wizard suggests mapping options based on data and on the visualization description.

- **Generators**: Generate the source code of a visualization required for the client. Therefore, the content of the RDF Data Cube will be mapped to the visual channels of the suitable visualization. The generated visualization will be displayed in the browser.

3.7.2 Process View

The Visualization Wizard is implemented as a service-oriented architecture (concretely in our case it follows the REST architectural style). Thus the individual tasks like automatic visualization suggestion, mapping definition and visualization generation can be realized as independent services. The services, which have been provided by the server and used by the client, are:

- **getPreview Service**: This service is responsible for delivering corresponding visualization suggestions for a RDF Data Cube. The essential functionality for the selection of the visualizations for any Cube is the extraction and analysis of the dimensions and measures. In this context, the server identifies the primitive data types (string, integer, decimal etc.) of the dimensions and measures, and compares these with the data type(s) of the visual channels of every available visualization to pick out the correct one. Afterwards, the mapping candidates and appropriate visualization suggestions will be sent to the client.
  
  The essential parts of this service are the routines `getSupportedChart` and `getPossibleVisualizationVariants`, which are responsible for the classification of the supported visualizations. These routines implement the two-phase classification algorithm (see Figure 29). In the first phase, only the visualizations are filtered (see the left part of the Figure 29) based on the structural and data type compatibility. In the second part the mapping combinations for the selected visualizations will be computed (see the right part of the Figure 29).

- **getVisualization Service**: This service is used in order to generate the selected visualization. The client builds an array with visual channels and corresponding RDF Data Cube components. The Server responds either with the complete visualization or with the corresponding data (as JSON) to generate the visualization.

- **getDimensions Service**: This service will be used to extract the dimensions of a RDF Data Cube.

- **getMeasure Service**: This service will be used to extract the measures of a RDF Data Cube.
Figure 28: Routines for matching vocabularies: chart suggestion (left) and suggestion of valid mapping variants (right)
4. Evaluation

During the development of the CODE Query Wizard and the CODE Vis Wizard, we followed the "release early, release often" principle. This means that as soon as a feature was completed and ready for testing, it immediately rolled out to our staging server and, if no major problems were found, a short time later (usually within hours, sometimes days) it was publicly available at our production server. This also means that the CODE Query Wizard and CODE Vis Wizard have been under permanent scrutiny of fellow researchers from the CODE project as well as other interested colleagues for several months now. They regularly provided valuable feedback on stability and usability issues as well as helpful feature requests.

Through countless iterative improvements, the CODE Query Wizard and the CODE Vis Wizard are now believed to work well enough to be useful for a more general public.

Additionally, the CODE Query Wizard and the CODE Vis Wizard were used in a workshop setting with around 20 students of the Semantic Technologies course at Graz University of Technologies. There, both tools proved helpful in evaluating the quality of the Linked Open Data the students had previously created in the workshop.

Starting in January, the CODE Query Wizard as well as the CODE Vis Wizard will undergo additional in-depth evaluations. The outcome of these final summative evaluations will be reported in Deliverable 4.4.
5. References


