Facilitating the Exploration and Visualization of Linked Data

Christian Mader[1], Michael Martin[2](✉), and Claus Stadler[2]

[1] Semantic Web Company, Vienna, Austria
christian.mader@semantic-web.at
[2] University of Leipzig, Leipzig, Germany
{martin,cstadler}@informatik.uni-leipzig.de

Abstract. The creation and the improvement of tools that cover exploratory and visualization tasks for Linked Data were one of the major goals focused in the LOD2 project. Tools that support those tasks are regarded as essential for the Web of Data, since they can act as a user-oriented starting point for data customers. During the project, several development efforts were made, whose results either facilitate the exploration and visualization directly (such as OntoWiki, the Pivot Browser) or can be used to support such tasks. In this chapter we present the three selected solutions *rsine*, *CubeViz* and *Facete*.

1 Introduction

The increasing number of datasets that are available as Linked Data on the Web makes it difficult for dataset curators to review additions, removals or updates of assertions involving resources they authored. Existing approaches like central registries do not scale with the fast-changing nature of the Web, thus being outdated or incomplete. In this chapter we propose a set of approaches that deal with the exploration and visualization of Linked Data. First we present the Resource SubscrIption and Notification sErvice (rsine) in Sect. 2 which enables subscribers to register for notifications whenever changes to RDF datasets occur. Thereby, we outline our approach based on a controlled vocabulary development scenario and integrate it into two exemplary LOD2 stack components to show its applicability. Based on requirements that come from practical experience in thesaurus development at Wolters Kluwer Germany, we describe how rsine can be used to check and avoid introduction of potential thesaurus quality problems.

Secondly, we showcase in Sect. 3 CubeViz, a flexible exploration and visualization platform for statistical data represented adhering to the RDF Data Cube vocabulary. If statistical data is represented according to that vocabulary, CubeViz exhibits a faceted browsing widget allowing to interactively filter observations to be visualized in charts. Based on the selected structural part, CubeViz offers suitable chart types and options for configuring the visualization by users. We present the CubeViz visualization architecture and components, sketch its underlying API and the libraries used to generate the desired output.

S. Auer et al. (Eds.): Linked Open Data, LNCS 8661, pp. 90–107, 2014.
DOI: 10.1007/978-3-319-09846-3_5

By employing advanced introspection, analysis and visualization bootstrapping techniques CubeViz hides the schema complexity of the encoded data in order to support a user-friendly exploration experience.

Lastly, we present Facete in Sect. 4, which is an application tailored for the exploration of SPARQL-accessible spatial data. Facete is built from a set of newly developed, highly modular and re-usable components, which power the following features:

- advanced faceted search with support of inverse properties and nested properties;
- automatic detection of property paths that connect the resources that matches the facet selection with those resources that can be shown on the map; and
- a map component that operates directly on a SPARQL endpoint and automatically adopts its data retrieval strategy based on the amount of available spatial information.

2 Rsine - Getting Notified on Linked Data Changes

With the growing amount of content available on the Web of Data it becomes increasingly difficult for human users to track changes of resources they are interested in. This even holds true for “local” use cases where, e.g., contributors are working on a dataset in a collaborative way, linking and annotating each other’s resources.

For example, contributors who develop controlled vocabularies, typically want to know whenever the meaning of a concept is fundamentally changed. This is because the concept might have been used for indexing documents and the changed meaning impairs search precision. However, with increasing frequency of change and size of the curated information resources, pull-based approaches do not scale anymore.

In this section we introduce the Resource SubscrIption and Notification sErvice (rsine), a framework that notifies subscribers whenever assertions to resources they are interested in are created, updated or removed. It is based on the W3C standards RDF and SPARQL and designed to be used alongside with existing triple storage solutions supporting these technologies. Multiple instances of the framework can communicate dataset changes also among each other. This allows to subscribe for changes of resources that are created or modified in other datasets on the Web that are managed by rsine.

An application of rsine is for instance the support of integrated quality management in controlled vocabulary development. We have shown in our previous work [8,9] that potential quality problems in controlled Web vocabularies can be detected from patterns (“quality issues”) in the underlying RDF graph. We believe that immediate notification of the responsible contributors after such quality issues have been introduced will lead to faster development and higher quality of the created vocabularies.

2.1 Related Work

SparqlPuSH [13] is a subscription/notification framework that allows for "proactive notification of data updates in RDF stores". Users express the resources they are interested in as SPARQL queries which are used by the service to create RSS or Atom feeds. These feeds are published on "hubs" using the PubSubHubbub protocol which handles the dissemination of notifications. Our approach is closely related to SparqlPuSH but is designed to operate on a more general level. In particular, creation and subscription to feeds, as proposed in SparqlPuSH, is only one of the possible options for notifying subscribers in rsine. Furthermore, SparqlPuSH only relies on the extensiveness of the data contained in the underlying RDF store. Thus, it is not possible to, e.g., find out about all resources deleted in a certain period of time. Rsine supports these scenarios by using a standard ontology for storing changeset metadata.

SDShare[1] is a protocol for the distribution of changes to resources that are represented in RDF. A server that exposes data provides four different Atom feeds that provide information about the state of the data and update information. The protocol is designed to support replications of linked data sources and relies on clients actively monitoring the provided feeds. Furthermore, clients only get information about the updated resource URIs and are expected to fetch the actual changes of resources themselves.

In the course of the REWERSE [10] project, a "general framework for evolution and reactivity in the Semantic Web" has been proposed that is based on Event-Condition-Action (ECA) rules. The framework is designed to be independent from the languages used to specify define events, conditions and actions. We stick to this approach but utilize a custom RDF ontology to express the ECA rules. Additionally we decided to use SPARQL for definitions of both events and conditions because of its wide acceptance and our focus on RDF data. This results in a light-weight approach, eliminating the need for custom event matchers and detection engines in favour of SPARQL endpoints and incremental RDF changesets. Actions are represented in our rsine ECA rules by specifying one or multiple notifiers.

2.2 Approach

Figure 1 describes the proposed architecture of the rsine service (Notification Service frame). The service on the left side of the figure is intended to give an overview on the components interacting internally, whereas the notification service on the right side is a second instance of the framework installed on a remote location on the Web.

Our approach uses a *Change Handler* that mediates between the *Managed RDF store* and the rsine service. In our implementation we provide a Change Handler (rsineVad[2]) that can be used for Virtuoso Servers. However, in environments that rely on different RDF storage backends such as openRDF Sesame,

[1] Final Draft:http://www.sdshare.org/spec/sdshare-20120710.html

[2] https://github.com/rsine/rsineVad

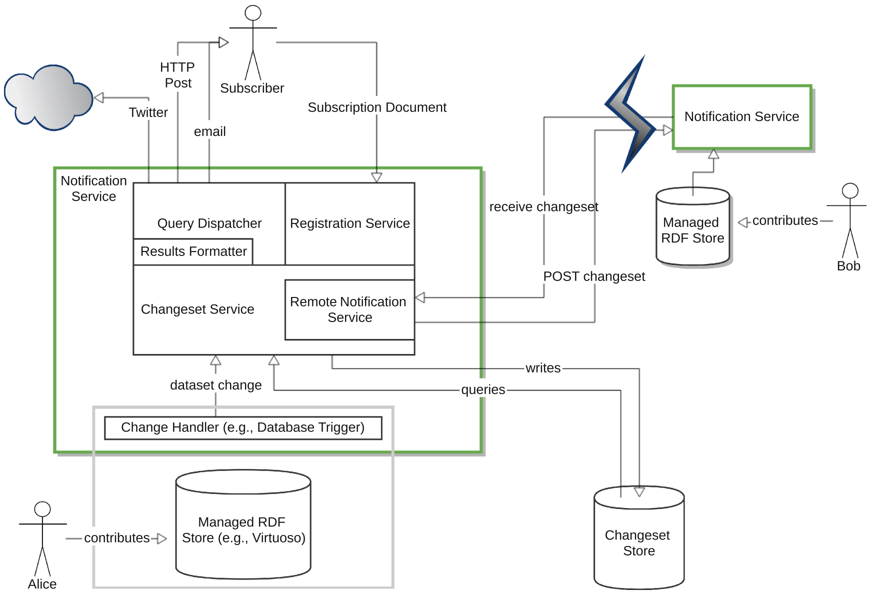


Fig. 1. Conceptual overview

a custom Change Handler that fits to the internals of the used storage solution must be implemented.

The rsine service continuously observes changes to the data held by a *Managed RDF Store* on the triple level, i.e., every time a triple is added, updated or removed the framework is triggered. The triple change events are then passed to the *Changeset Service* which converts the received triple changes to changesets expressed in the Changeset[3] ontology and persists them to an internal *Changeset Store*.

A subscriber who is interested in receiving notifications can subscribe by sending a subscription document to the service that contains SPARQL queries and information about the preferred notification channel (e.g., email, Twitter). The queries from the subscription document select resources the subscriber is interested in and access both the data contained in the *Managed RDF Store* as well as in the *Changeset Store*. The results of these queries, can then be disseminated through the desired channels. Before dissemination, the *Results Formatter* formats the query results into human-readable form by using the template provided in the subscription document.

Rsine can also send ("forward") local dataset changes to remote rsine instances on the Web (small Notification Service box). This feature is useful to get notifications whenever resources in datasets on different servers reference each other. Due to space limitations we refer to deliverable D5.3.1 for a detailed coverage of the workflows for both local and forwarded dataset changes.

[3] http://vocab.org/changeset/schema.html

2.2.1 Subscribing for Notifications

Subscriptions are RDF documents that are sent to the rsine service by HTTP POST. They consist of two mandatory parts: (i) a *query* which specifies the resources the subscriber is interested to get notifications about and (ii) at least one *notifier* that defines the way notification messages should be disseminated. A basic example is provided in Listing 1 (prefixes omitted, for an in-depth coverage we refer to the online documentation[4]).

```
[] a rsine:Subscription;
   rsine:query [
       spin:text"SELECT * WHERE {
               ?cs a cs:ChangeSet .
               ?cs cs:addition ?addition .
               ?addition rdf:subject ?concept .
               ?addition rdf:predicate skos:prefLabel .
               ?addition rdf:object ?newLabel}";];

   rsine:notifier [
       a rsine:emailNotifier;
       foaf:mbox <mailto:me@myoffice.com>].
```

Listing 1. Rsine Subscription.

2.3 Stack Integration

In order to showcase the capabilities of rsine, we integrated it with two exemplary components of the LOD2 stack: The *PoolParty Thesaurus Server* (PPT) and *Pebbles*. PPT is a tool for taxonomy experts to develop thesauri and publish them as Linked Data using SKOS. Pebbles is a Web application that provides a GUI to manage RDF metadata for XML documents. For testing the integration we used the stack installation operated by Wolters Kluwer Germany (WKD).

PPT builds on OpenRDF Sesame infrastructure for persisting RDF data. In order to provide interoperability between PPT and rsine, we implemented a subclass of `RepositoryConnectionListenerAdapter`. It intercepts the triple changes and, before handing them down to the triple store for persistence, announces them to rsine's HTTP interface.

Pebbles uses Virtuoso as storage backend. The only task for integrating rsine with Pebbles was thus to deploy the rsineVad package from the rsine repository to the Virtuoso instance. RsineVad is an extension to Virtuoso that configures database triggers and stored procedures so that all triple changes Pebbles performs to are communicated to rsine.

2.4 Notification Scenarios

WKD specified in total 13 scenarios for notifications that are described in detail in deliverable D7.3. They are divided into scenarios that are important in a thesaurus development process (e.g., to "follow all changes such as deletion,

[4] https://github.com/rsine/rsine#subscriptions

linking or editing of concepts") and scenarios from metadata editing with Pebbles (e.g., "Follow all changes of the document metadata"). We were able to support all but one requirements from the thesaurus development scenario and implemented one metadata editing scenario as a proof-of-concept. Furthermore, we adapted 9 checks for potential controlled vocabulary quality problems from our earlier work[5] and converted them for use with rsine. Among them are, e.g., checks for cyclic hierarchical relations or concepts with conflicting (i.e., identical) preferred labels.

3 CubeViz – Exploration and Visualization of Statistical Linked Data

A vast part of the existing Linked Data Web consists of statistics (cf. LODStats[6] [3]) being represented according to the RDF Data Cube Vocabulary [2]. To hide the inherently complex, multidimensional statistical data structures and to offer a user-friendly exploration the RDF Data Cube Explorer CubeViz[7] has been developed. In this chapter we showcase how large data cubes comprising statistical data from different domains can be analysed, explored and visualized. CubeViz is based on the OntoWiki Framework [7] and consists of the following OntoWiki extensions:

- The *Integrity Analysis Component* (cf. Sect. 3.2) evaluates the existence and the quality of selected RDF graphs according to given integrity constraints.
- The *Facetted Data Selection Component* (cf. Sect. 3.3) is retrieving the structure of the selected Data Cube using SPARQL [5] in order to generate filter forms. Those forms allow to slice the data cube according to user interests.
- The *Chart Visualization Component* (cf. Sect. 3.4) receives all observation as input, that correspond to the given filter conditions, in order to generate a chart visualization.

All components support the comprehensive CubeViz GUI shown in Fig. 2. Before we introduce the three components in more detail, we give a brief introduction of the RDF Data Cube Vocabulary in the next section. We conclude the paper with links to publicly available deployments and a list of some upcoming features planned for the next release. Further information about CubeViz can be obtained in the repository wiki[8] or via a recorded webinar[9] comprising a comprehensive screencast.

3.1 The RDF Data Cube Vocabulary

The *RDF Data Cube vocabulary* is a W3C recommendation for representing statistical data in RDF. The vocabulary is compatible with the Statistical Data and

[5] qSKOS controlled vocabulary quality checker, https://github.com/cmader/qSKOS
[6] http://stats.lod2.eu/rdf_classes?search=Observation
[7] http://aksw.org/Projects/CubeViz
[8] https://github.com/AKSW/cubeviz.ontowiki/wiki
[9] http://www.youtube.com/watch?v=ZQc5lk1ug3M#t=1510

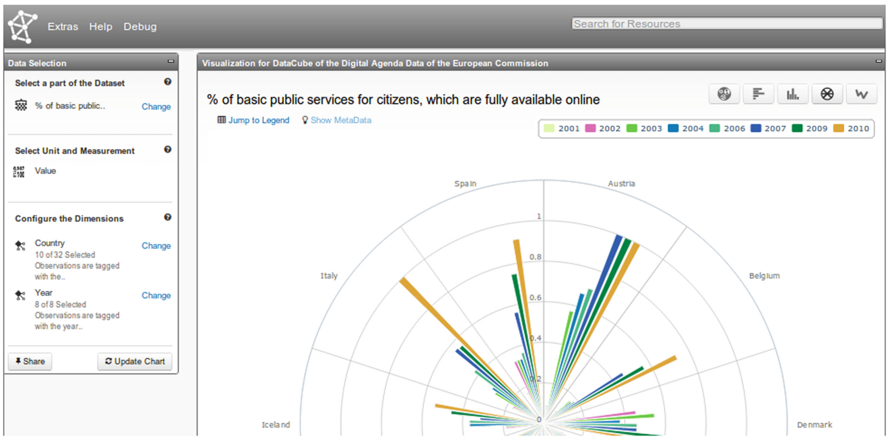


Fig. 2. The CubeViz GUI visualizing a slice of a 2-dimensional RDF DataCube in a combined polar-column chart.

Medadata eXchange XML format (SDMX) [4], which is defined by an initiative chartered in 2001 to support the exchange of statistical data. Sponsoring institutions[10] of SDMX are among others *the Bank for International Settlements*, the *European Central Bank*, *Eurostat*, the *International Monetary Fund*, the *Organisation for Economic Co-operation and Development* (OECD), the *United Nations Statistics Division* and the *World Bank*. Experiences while publishing statistical data on the Web using SDMX were summarized by the *United Nations* in [11] and by the OECD in [12].

The core concept of the Data Cube vocabulary is the class `qb:Observation`[11], that is used to type all statistical observations being part of a Data Cube. Every observation has to follow a specific structure that is defined using the class `qb:DataStructureDefinition` (DSD) and referenced by a dataset resource (DS) of type `qb:DataSet`. Since every observation should refer to one specific DS (which again refers to the corresponding DSD) the structure of the observation is fully specified. DSD components are defined as set of dimensions (`qb:DimensionProperty`), attributes (`qb:AttributeProperty`) and measures (`qb:MeasureProperty`) to encode the semantics of observations. Those component properties are used to link the corresponding elements of dimensions, measure values and units with the respective observation resource. Furthermore, it is possible to materialize groups and slices of observations as well as hierarchical orders of dimension elements using respective concepts.

[10] http://sdmx.org/?page_id=6
[11] Prefix qb:http://purl.org/linked-data/cube#

3.2 Integrity Analysis

As described in the W3C RDF Data Cube Recommendation document data cubes are structurally well-formed if they comply to specific integrity constraints[12]. Those constraints can be used to validate and if necessary to improve the quality of a data cube. For CubeViz, we translated those constraints into SPARQL queries using an `ASK`-clause returning boolean values. The queries were integrated into the Integrity Analysis Component of CubeViz, whose GUI is depicted in Fig. 3. If a query returns false, the corresponding constraint is marked in the GUI in red and can be selected in order to reuse and modify them with a configured query editor. This functionality supports the discovery of potential modelling or conversion flaws.

Additionally, this component is used to introspect the selected RDF model for all included data cubes. If the introspection query (given in Listing **??**) returns a positive result, the Faceted Data Selection and Chart Visualization components are activated.

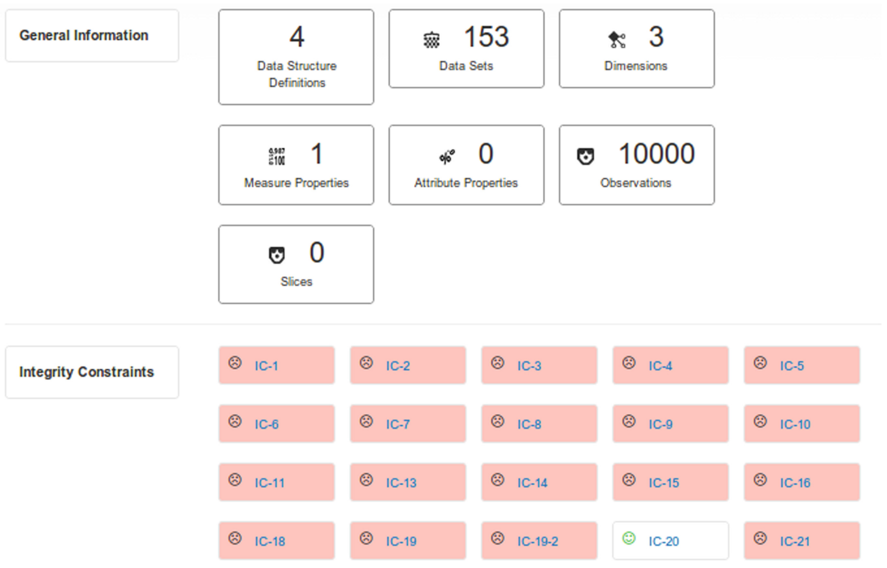
```
PREFIX qb:<http://purl.org/linked-data/cube#>
ASK FROM <http://example.org/> {
  ?observation   a              qb:Observation .
  ?observation   qb:dataSet     ?dataset .
  ?observation   ?dimension     ?dimelement .
  ?observation   ?measure       ?value .
  ?dataset       a              qb:DataSet .
  ?dataset       qb:structure   ?dsd .
  ?dsd           a              qb:DataStructureDefinition .
  ?dsd           qb:component   ?dimspec .
  ?dsd           qb:component   ?measurespec .
  ?dimspec       a              qb:ComponentSpecification .
  ?dimspec       qb:dimension   ?dimension .
  ?measurespec   a              qb:ComponentSpecification .
  ?measurespec   qb:measure     ?measure}
```

Listing 2. Data cube introspection query.

3.3 Faceted Exploration

Given that the introspection was successful, specific structural parts of the identified data cube are queried in order to create a faceted search interface. All components of a DSD have to be integrated into any observation of the respective DS. In order to discover those observations the user has to select values that are referenced by those components. First the user needs to select a DS of a data cube in order to analyse the DSD that is the basis for all further facets. Second the user has to select the measure and attribute property used to identify the representation of values. The last mandatory facet is used to offer the selection of dimensions and its respective elements of interest. CubeViz is processing and visualizing values exactly as they are represented in the data cube and does not support aggregate functions such as `SUM`, `AVG`, `MIN` and `MAX`. As a consequence, users have to select at least one element of each dimension. Furthermore, if materialized slices are aggregated within the selected DS an optional facet will be generated to offer a selection from the retrieved slices.

[12] http://www.w3.org/TR/vocab-data-cube/#wf-rules



General Information

4 Data Structure Definitions

153 Data Sets

3 Dimensions

1 Measure Properties

0 Attribute Properties

10000 Observations

0 Slices

Integrity Constraints

IC-1 IC-2 IC-3 IC-4 IC-5

IC-6 IC-7 IC-8 IC-9 IC-10

IC-11 IC-13 IC-14 IC-15 IC-16

IC-18 IC-19 IC-19-2 IC-20 IC-21

Fig. 3. GUI presenting results of the statistical and integrity analysis.

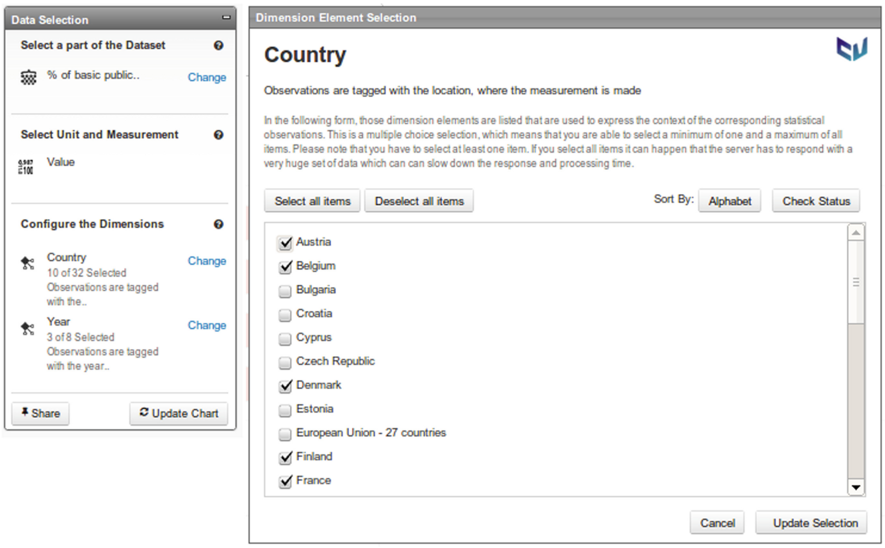
3.3.1 Generation of Dialogues

The detected facets and their generated GUI representations are integrated into a filter form. To select/deselect elements of facets for extracting subsets of the DS, respective interface elements are dynamically created. According to the type of facet (mandatory/optional) a configurable amount of elements (min/max) is selectable. Additionally, the label and textual description of components are retrieved using SPARQL queries and added to the interface. As illustrated in Fig. 4 the selected amount of facet elements is displayed after confirmation. Already discovered RDF resources are cached on the client-side and will be re-used in the Chart Visualization component.

One of the major advantages of faceted exploration is the avoidance of possibly empty result sets. To avoid empty sets of observations after facet selection, the set of selectable elements of all further facets in combination with its respective count of observations is being calculated using respective SPARQL queries. Every selected combination of a component and its respective element is represented by a triple pattern that is conditionally used to retrieve the set of observations and all facet elements.

3.3.2 Initial Pre-selection

To lower the barrier of exploring a data cube from scratch, an initial pre-selection algorithm is started after a positive introspection. As described in Sect. 3.4 it is possible to integrate and configure charts visualizing one or multiple dimensions. The determined maximum amount of dimensions respectively chart axis is used


Data Selection

Select a part of the Dataset

% of basic public.. Change

Select Unit and Measurement

Value

Configure the Dimensions

Country
10 of 32 Selected
Observations are tagged with the.. Change

Year
3 of 8 Selected
Observations are tagged with the year.. Change

Share | Update Chart

Dimension Element Selection

Country

Observations are tagged with the location, where the measurement is made

In the following form, those dimension elements are listed that are used to express the context of the corresponding statistical observations. This is a multiple choice selection, which means that you are able to select a minimum of one and a maximum of all items. Please note that you have to select at least one item. If you select all items it can happen that the server has to respond with a very huge set of data which can can slow down the response and processing time.

Select all items | Deselect all items | Sort By: Alphabet | Check Status

- [✓ Austria]
- [✓ Belgium]
- Bulgaria
- Croatia
- Cyprus
- Czech Republic
- [✓ Denmark]
- Estonia
- European Union - 27 countries
- [✓ Finland]
- [✓ France]

Cancel | Update Selection

Fig. 4. Facets and dialogues.

as input for the pre-selection algorithm. After extracting all obligatory facets exactly one element per facet is pre-selected. According to the number of discovered dimensions and the maximum amount of processable chart axis, dimensions are randomly selected for which more than one element can be selected. To avoid confusing visualizations the amount of pre-selected elements is limited to 10 respectively 30 % of the set of elements. During manual selection these limits are not relevant.

3.4 Chart Visualisation

In order to extract observations according to user interests, filter criteria from the facet selection component are translated into a corresponding SPARQL query. The resulting set of observation resources is serialized in JSON and sent back to the client. On the client-side the result set will be analysed according the amount of disjunctive dimensions and the respective amount of elements in order to select suitable charts. After identifying suitable chart implementations the first one is launched and renders the visualization using the received set of observations. All further suitable charts can be selected using a respective GUI element without querying the observations again.

APIs

CubeViz comprises an abstraction layer to mediate between the retrieval of observations and the APIs used to generate and visualize charts. Currently, charts such as pie, bar, column, line and polar chart are implemented using the

APIs *Data Driven Documents*[13] (D3js) and *HighCharts*[14]. Every chart is implemented using a defined interface and comprises a mechanism to convert the set of observations in combination with the meta data about dimension properties into the chart-specific input format.

Chart Options

Most of the implemented chart visualizations can be adjusted using preconfigured chart options. Hence, it is possible to enable the display of measure values in addition to its graphical representation, to switch axis / dimensions, to switch the value scale between linear and logarithmic or to enable a normal or percentage stacking. Additionally it is possible to combine charts such as a polar chart with a column chart (see Fig. 2).

Element Recognition

On the Linked Data Web, URIs are used to identify resources. In a domain-agnostic tool such as CubeViz, it is not feasible to integrate static mappings between data items and their graphical representations. Most of the chart APIs have a limited amount of pre-defined colors used for colouring dimension elements or select colors completely arbitrarily. In order to display dimension elements in a deterministic manner and to support users to quickly recover selected elements in different charts we integrated a colouring algorithm that uniquely assigns URIs of each dimension element corresponding RGB color codes[15].

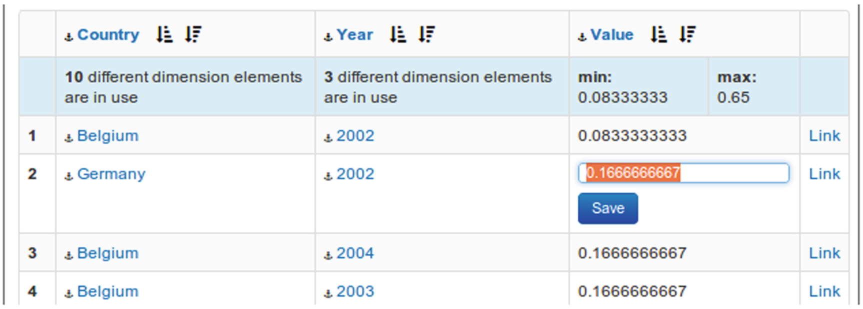
Interactive Legend

Below the generated charts an additional tabular representations of the selected data items is given (cf. Fig. 5). On the one hand they can be used as legend containing additional meta data. On the other hand this view offers support for resolving data inaccuracies with functionality for editing values, that automatically updates the chart representation.

Sharing Views

After exploring, configuring and possible adaption of values users are able to share the created output. Sharing functionality is implemented via a button, which triggers the gathering of all information necessary to reproduce the created output, storing them server-side and returning a shareable link containing an identifying hash code for the particular view configuration. Furthermore, it is possible to export selected data as CSV and RDF in Turtle notation.

[13] http://d3js.org/
[14] http://www.highcharts.com/
[15] http://cold.aksw.org/



	Country	Year	Value		
	10 different dimension elements are in use	**3** different dimension elements are in use	**min:** 0.08333333	**max:** 0.65	
1	Belgium	2002	0.0833333333		Link
2	Germany	2002	0.1666666667 Save		Link
3	Belgium	2004	0.1666666667		Link
4	Belgium	2003	0.1666666667		Link

Fig. 5. Interactive CubeViz legend.

4 Facete - A Generic Spatial Facetted Browser for RDF

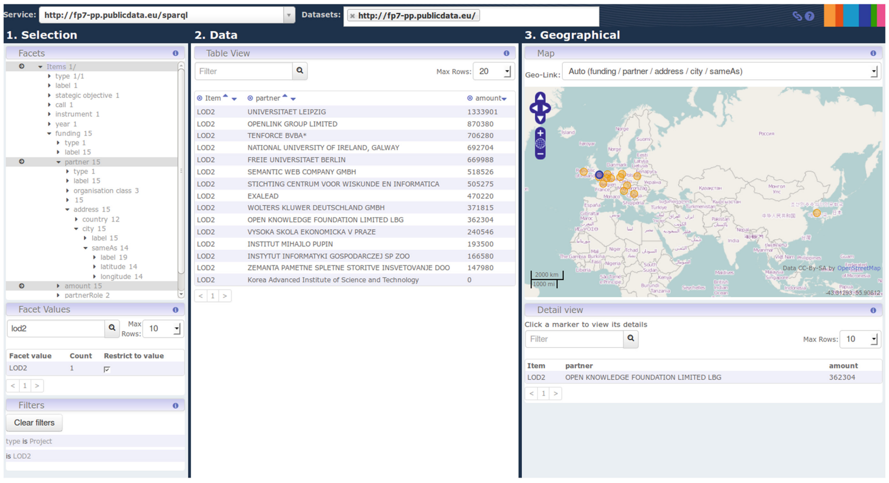
Facete is a web application for the exploration of SPARQL-accessible spatial data, which offers several distinguishing features. First, there is the advanced faceted search component which enables users to filter the data by inverse properties and nested properties. Counts are provided for both facets and facet values. Second, the system will always attempt to detect (possible indirectly) related geometric information for the set of resources matched by the faceted search. For example, if a user filters by the class *Person*, then the system could detect that *birthPlace* and *deathPlace* provide geo-coordinates and appropriate suggestions about what to display on the map would be shown to the user. Third, Facete provides a map display capable of dealing with large amounts of geometric information. Finally, users are able to customize a tabular view for the data related to their facet selection. Information about Facete is available on its project site[16]. All of Facete's user interface components are based on the popular AngularJS[17] framework, and are published as a separate library called *JAvascript Suite for Sparql Access* (Jassa)[18]. In the remainder of this section, we give an overview of Facete's components, which we partly published in [14].

4.1 User Interface

Facete is implemented as a Single Page Application (SPA) whose user interface comprises several UI components, which are depicted in Fig. 6 and explained in the following. In the top area, there are elements that enable the user to select a SPARQL endpoint and chose from one or more of its contained named graphs. The main panel is divided into three columns containing a set of widgets with the following functionality:

1. Selection. The first widget, labeled *Facet*, shows a facet tree that corresponds to the properties of all resources that match the set constraint. If there

[16] http://aksw.org/Projects/Facete
[17] http://angularjs.org/
[18] https://github.com/GeoKnow/Jassa-UI-Angular


Service: http://fp7-pp.publicdata.eu/sparql
Datasets: http://fp7-pp.publicdata.eu/

1. Selection

Facets

```
Items 1/
  type 1/1
  label 1
  stategic objective 1
  call 1
  instrument 1
  year 1
  funding 15
    type 1
    label 15
    partner 15
      type 1
      label 15
      organisation class 3
      15
      address 15
        country 12
        city 15
          label 15
          sameAs 14
            label 19
            latitude 14
            longitude 14
    amount 15
    partnerRole 2
```

Facet Values

lod2 Max Rows: 10

Facet value	Count	Restrict to value
LOD2	1	✓

Filters

Clear filters

type is Project

is LOD2

2. Data

Table View

Filter Max Rows: 20

Item	partner	amount
LOD2	UNIVERSITAET LEIPZIG	1333901
LOD2	OPENLINK GROUP LIMITED	870380
LOD2	TENFORCE BVBA*	706280
LOD2	NATIONAL UNIVERSITY OF IRELAND, GALWAY	692704
LOD2	FREIE UNIVERSITAET BERLIN	669988
LOD2	SEMANTIC WEB COMPANY GMBH	518526
LOD2	STICHTING CENTRUM VOOR WISKUNDE EN INFORMATICA	505275
LOD2	EXALEAD	470220
LOD2	WOLTERS KLUWER DEUTSCHLAND GMBH	371815
LOD2	OPEN KNOWLEDGE FOUNDATION LIMITED LBG	362304
LOD2	VYSOKA SKOLA EKONOMICKA V PRAZE	240546
LOD2	INSTITUT MIHAJLO PUPIN	193500
LOD2	INSTYTUT INFORMATYKI GOSPODARCZEJ SP ZOO	166580
LOD2	ZEMANTA PAMETNE SPLETNE STORITVE INSVETOVANJE DOO	147980
LOD2	Korea Advanced Institute of Science and Technology	0

3. Geographical

Map

Geo-Link: Auto (funding / partner / address / city / sameAs)

Detail view

Click a marker to view its details

Filter Max Rows: 10

Item	partner	amount
LOD2	OPEN KNOWLEDGE FOUNDATION LIMITED LBG	362304

Fig. 6. Graphical user interface of facete

are no constraints, all resources that appear as a subject in the selected graphs are matched. Three actions can be performed for node in the facet tree. A click on the facet's name lists the facet's values in the *Facet Value* widget, where these values can be used for defining constraints. Clicking the caret symbol toggles the display of corresponding child facets. These are the properties of the selected facet's values. Lastly, a facet can be pinned as a column to the *Table View*. Note, that the root of the facet tree is formed by a facet labelled *Items*. This facet's values correspond to the set of resources in subject positions of the selected RDF graphs. The *Facet Values* widget enables a user to paginate through a selected facet's values and optionally filter these values by a search term. Furthermore, clicking the checkbox next to a value creates a constraint. The *Filters* widget lists all active constraints. Individual constraints can be removed by clicking their entry, whereas the *Clear Filters* button purges them all.

2. Data. The middle area contains the *Table View*, which lists a table whose content is based on resources that match the active constraints and the facets that were pinned as columns. Columns can be sorted by clicking the caret icons. Multiple orders are supported by holding the shift key down while clicking.

3. Geographical. The *Geo-Link* drop down menu enables the user to choose from property paths connecting the resources that match the constraints with those that can be shown on the map. By default, the option *automatic* is enabled, which always picks the shortest path among the found ones. The *Map* widget displays markers corresponding to the selected resources and the geo-link. Blue boxes indicate areas that contain too many markers to be shown at once. These boxes disappear when sufficiently zoomed in. Clicking a marker shows its details in the *Detail View*. The *Detail View* shows the excerpt of the *Table View* that corresponds to the selected marker(s).

4.2 Concepts

In this section we briefly outline the key concepts used in Facete, which are related to faceted search, detection of property paths that connect concepts and dealing with large amounts of spatial data.

4.2.1 Faceted Search

Facete's approach to faceted search based on the following concepts.

- A *SPARQL concept* is a pair comprising a SPARQL graph pattern and a variable thereof. As such, it intentionally describes a set of resources. For instance, the pair *({?s a Person}, ?s)* could be used to describe a set of people. SPARQL concepts are a key enabler for indirect faceted search as they can be used to represent virtually any set of resources (within the expressivity of SPARQL), such as the set of facets, the set of child facets, the set of facet values and the set of resources with geometric information.
- *Property Steps* are used to navigate from a set of resources to a related set of resources by following a specific property. A direction attribute determines whether to follow a property in forward or inverse direction. Hence, a destination SPARQL concept can be obtained from a given origin SPARQL concept and a property step.
- A *Property Path* is a sequence of property steps.
- *Constraint Specifications* express constraints via references to property paths. Constraint specifications are internally translated to corresponding SPARQL graph patterns.

4.2.2 Finding Connections between SPARQL Concepts

Depending on how a dataset was modeled, the spatial dimension may not be directly attached to instances of a certain type. In order to visualize the spatial dimension of such objects efficiently and intuitively we need an approach to find connecting property paths between two arbitrary SPARQL concepts efficiently. These paths can become relatively long, and naive path discovery approaches are not feasible. For example, in our RDFized version of the FP7 project funding dataset[19], projects are related to geometric information via paths of length 5.

Our approach is outlined as follows: because we are only interested in the detection of property paths, we pre-compute a *property join summary*. The basic SPARQL query for this purpose is:

```
CONSTRUCT
  { ?p1 :joinsWith ?p2 }
  {
    { SELECT DISTINCT ?p1 ?p2 {
        ?a ?p1 [ ?p2 ?b ]
    } }
  }
```

[19] http://fp7-pp.publicdata.eu/sparql

Conceptually, we could search for arbitrary complex paths, such as ones that include cyclic (same property followed multiple times in the same direction) and zig-zag (forward and backward on the same property traversals. However, for our use cases the restriction to directed acyclic paths leading from a *source concept* to a *target concept* was sufficient: we query the source concept for all of its properties ?*p*, and conceptually add triples *(:source :joinsWith ?p)* to the join summary. Thereby *:source* is a distinguished resource representing the source concept. From a target concept's graph pattern, such as *(?s geo:long ?x ; geo:lat ?y, ?s)*, we can infer that we need to search for properties that according to the join summary are connected to both geo:long and geo:lat. As a consequence, we can query the joinsummary for a set of candidate target properties using:

```
SELECT DISTINCT ?p { ?p :joinsWith geo:long ; joinsWith geo:lat }
```

If the extensions of the source and target concepts have resources in common, this query's result set includes *:source* as a candidate.

We can now search for candidate paths on the join summary that connect *:source* with each of the candidate properties. For each candidate path we then fire an ASK query to check whether the given dataset contains an instance of it. Those paths for which actually data exists, are then listed in Facete's Geo-Link drop down box.

Note, that this approach is independent of any concrete vocabulary.

4.3 Display of Large Amounts of Geometries

Some spatial RDF datasets, such as DBpedia or Freebase, contain significantly more spatial information than can be reasonably retrieved and displayed on a map in a web application considering bandwidth and performance restrictions. Facete handles such cases using a quad tree data structure:

- Based on the users constraints on the facets and the geo-link, a corresponding SPARQL concept, named *geo-concept*, is created. The geo-concept specifies the set of resources to be shown on the map.
- A count of the number of instances matching the geo-concept is requested. If the count is below a configured threshold, all instances are retrieved at once and placed into the root node of the quad tree.
- If this count exceeds the threshold, the extent of the whole map is split recursively into four tiles of equal size. The recursion stops if either a maximum depth is reached, or if the tiles have reached a certain relative size when compared to the map viewport (e.g. when about 4×4 tiles are visible). For each tile, the geo-concept is then modified to only refer to resources within that tiles' bounding box. A tile's resources are only retrieved, if the new count is again below a configured threshold.
- Tiles that still contain too many geometries are rendered as boxes on the map.

An example of such display is shown in Fig. 7, which shows a subset of the approx. 20.000 resources with geo-positions in Germany. For each set of constraints, Facete creates a new quad tree that acts as a cache for the user's current configuration.

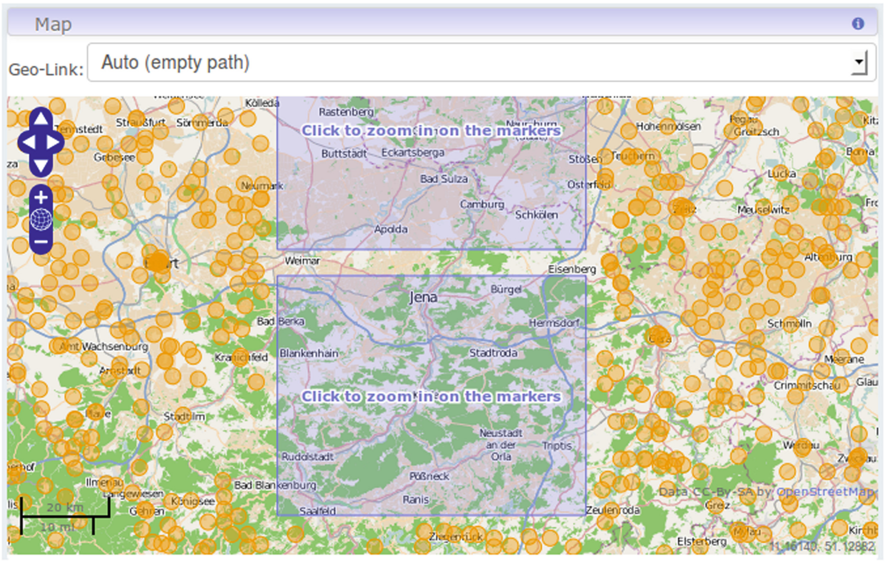

Map
Geo-Link:
Auto (empty path)
Click to zoom in on the markers
Click to zoom in on the markers
Jena
Weimar
20 km
10 mi
Data CC-By-SA by OpenStreetMap

Fig. 7. Display of Freebase instances in Germany.

4.4 Related Work

The *RelFinder* system [6] is capable of finding property paths connecting a pair of *resources*, whereas Facete searches for paths between SPARQL *concepts*. Over the past decade, faceted search has become omnipresent, such as in web shop and content management systems. *Apache Solr*[20] is a popular system that offers extensive faceted search features, however, it does not offer native SPARQL support and thus requires pre-processing of RDF data. *Rhizomer* [1] and the *Virtuoso faceted browser*[21] support changing the focus from one set of resources to a related one (known as pivoting). However, with these systems, the user actually navigates between related list views of resources, whereas in Facete the user pins facets as new columns to the table view.

5 Conclusions and Future Work

In this chapter, we presented three tools as part of the LOD2 project that aim at facilitating the exploration of Linked Open Data. Rsine is a tool that sends notifications about changes in a RDF dataset. CubeViz is a data cube visualisation tool for statistical data. Facete is a faceted browser application with a focus on spartial data.

[20] http://lucene.apache.org/solr/

[21] http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/VirtuosoFacetsWebService

We presented the design, implementation and integration of rsine, a service that notifies subscribers on specific changes in an RDF data set. We integrated the service with two LOD2 stack components and showed its applicability by supporting requirements for notifications in a thesaurus and metadata editing scenario at WKD.

Our immediate next steps are to evaluate the implemented thesaurus notifications in the context of a thesaurus development project at WKD. We are interested in the performance of our approach and the "usefulness" of the notifications, i.e., if and in what way they influence the quality of the created thesauri. We furthermore plan to set up multiple rsine instances in this environment to gain insights about how notifications can help when references between datasets on distinct servers are created, modified or deleted.

We presented the architecture, analysis components and visualization interfaces of CubeViz, a RDF Data Cube browser. In addition to the exploration of locally stored RDF data cubes it is possible to access remotely published ones using a combination of the SPARQL backend and the SPARQL services component. Such a setup was deployed on the European Commission's IT infrastructure as part of the European Data Portal[22].

There are further deployments of CubeViz made online such as LinkedSpending[23], which contain government spendings from all over the world represented and published as Linked Data (more than 2.4 million observations in 247 datasets). Using LinkedSpending, interested users can gather information about greek spending on police in certain regions in 2012 for instance (jump in using the button *Example Visualization 2* on the start page).

CubeViz is publicly available for download[24] and its latest releases can be evaluated using an online demonstrator[25]. CubeViz is under active development and will be further extended with new features such as drill-down functionality, additional interactive and customizable charts, further chart APIs such as the *Google Charts API*[26], aggregate functions and mashup features to compare observations from different domains.

Lastly, we gave an overview about Facete, a tool for browsing Linked Data in a domain-agnostic way with a focus on spatial data. Its major goal is to ease the navigation of RDF data in SPARQL endpoints using advanced faceted search techniques and - in addition - the provision of corresponding visualization widgets. Facete is published under the Apache 2.0, license and its development continues within the GeoKnow project[27]. Further information such as new releases, links to the sources, demos and documentation can be found on the project page[28].

[22] https://open-data.europa.eu/cubeviz/
[23] http://linkedspending.aksw.org/
[24] https://github.com/AKSW/CubeViz/
[25] http://cubeviz.aksw.org/
[26] https://developers.google.com/chart/
[27] http://geoknow.eu
[28] http://aksw.org/Projects/Facete

References

1. Brunetti, J.M., Gil, R., Garcia, R.: Facets and pivoting for flexible and usable linked data exploration. In: Interacting with Linked Data Workshop (ILD) (2012)
2. Cyganiak, R., Reynolds, D., Tennison, J.: The RDF data cube vocabulary. Technical report, W3C, 2013. http://www.w3.org/TR/vocab-data-cube/
3. Auer, S., Demter, J., Martin, M., Lehmann, J.: LODStats – an extensible framework for high-performance dataset analytics. In: ten Teije, A., Völker, J., Handschuh, S., Stuckenschmidt, H., d'Acquin, M., Nikolov, A., Aussenac-Gilles, N., Hernandez, N. (eds.) EKAW 2012. LNCS (LNAI), vol. 7603, pp. 353–362. Springer, Heidelberg (2012)
4. International Organization for Standardization. Statistical data and metadata exchange (SDMX). Technical report, Standard No. ISO/TS 17369:2005 (2005)
5. Harris, S., Seaborne, A.: SPARQL 1.1 Query Language - W3C Recommendation. Technical report, World Wide Web Consortium (W3C) (2013). http://www.w3.org/TR/sparql11-query/
6. Heim, P., Hellmann, S., Lehmann, J., Lohmann, S., Stegemann, T.: RelFinder: revealing relationships in RDF knowledge bases. In: Chua, T.-S., Kompatsiaris, Y., Mérialdo, B., Haas, W., Thallinger, G., Bailer, W. (eds.) SAMT 2009. LNCS, vol. 5887, pp. 182–187. Springer, Heidelberg (2009)
7. Heino, N., Dietzold, S., Martin, M., Auer, S.: Developing semantic web applications with the OntoWiki framework. In: Pellegrini, T., Auer, S., Tochtermann, K., Schaffert, S. (eds.) Networked Knowledge - Networked Media. SCI, vol. 221, pp. 61–77. Springer, Heidelberg (2009)
8. Mader, C., Haslhofer, B., Isaac, A.: Finding quality issues in SKOS vocabularies. In: Zaphiris, P., Buchanan, G., Rasmussen, E., Loizides, F. (eds.) TPDL 2012. LNCS, vol. 7489, pp. 222–233. Springer, Heidelberg (2012)
9. Mader, C., Wartena, C.: Supporting web vocabulary development by automated quality assessment: results of a case study in a teaching context. In: Workshop on Human-Semantic Web Interaction (HSWI14), CEUR Workshop Proceedings, May 2014
10. May, W., Alferes, J.J., Amador, R.: An ontology- and resources-based approach to evolution and reactivity in the semantic web. In: Meersman, R. (ed.) Coop-IS/DOA/ODBASE 2005. LNCS, vol. 3761, pp. 1553–1570. Springer, Heidelberg (2005)
11. United Nations: Guidelines for Statistical Metadata on the Internet. Technical report, Economic Commission for Europe (UNECE) (2000)
12. Management of Statistical Metadata at the OECD (2006)
13. Passant, A., Mendes, P.: sparqlPuSH: Proactive notification of data updates in RDF stores using PubSubHubbub. In: CEUR Workshop Proceedings ISSN 1613–0073, February 2011
14. Stadler, C., Martin, M., Auer, S.: Exploring the web of spatial data with facete. In: Companion proceedings of 23rd International World Wide Web Conference (WWW), pp. 175–178 (2014)