Commercially Empowered
Linked Open Data Ecosystems in Research

D 3.4 Evaluation Results

Summary: This deliverable introduces the results produced by WP3 for Task 3.4 “Refinement & Evaluation”. The evaluation results and statistics conducted in this task are available for the prototypes of the Balloon (indexing and discovery of Linked Open Data) and Bacon (management and aggregation of data cubes) project and shows statistics for the MindMeister endpoint, which is covering semantically enriched mind maps produced within the project. The report builds on the previous deliverable D3.3 – Refined Federated Querying and Aggregation.

Project Acronym	CODE
Grant Agreement number	296150
Project Title	Commercially Empowered Linked Open Data Ecosystems in Research
Date	2014-04-30
Nature	R (Report)
Dissemination level	PU (Public)
WP Lead Partner	University of Passau
Revision	1.0/final
Authors	Sebastian Bayerl, Emanuel Berndl, Michael Granitzer, Kai Schlegel, Florian Stegmaier

Consortium:

SEVENTH FRAMEWORK PROGRAMME

Project Officer & Project Coordinators

Project Officer	Stefano Bertolo	European Commission Directorate-General Communications Networks, Content and Technology (DG CONNECT) Unit Data Value Chain (G3) 10, Rue Robert Stümper 2920 Luxembourg stefano.bertolo@ec.europa.eu
Project Coordinator	Stefanie Lindstaedt	Know-Center GmbH Inffeldgasse 13, 8010 Graz, Austria +43 316 873-30800 (phone) +43 316 873-1030800 (fax) slind@know-center.at
Scientific Coordinator	Michael Granitzer	University of Passau Innstrasse 33a, D-94032 Passau +49(0)851-509-3305 michael.granitzer@uni-passau.de

Document Revision History

Revision	Date	Author	Organization	Description
1	23.05.2014	S. Bayerl	UNI PASSAU	Added evaluation of Bacon
2	28.05.2014	K. Schlegel	UNI PASSAU	Added evaluation of Balloon
3	28.05.2014	E. Berndl	UNI PASSAU	Added description of Mindmeister EP
4	30.04.2014	F. Stegmaier	UNI PASSAU	Finalized document
final				

Table of Contents

1. Introduction

Today's vision of a common Web of Data is mostly achieved and coined by the Linked Open Data movement. The first wave of this movement transformed silo-based portions of data into a plethora of open accessible and interlinked data sets. The community itself provided guidelines (e.g., 5 Star Open Data) as well as open source tools to foster interactions with the Web of data. Harmonization between those data sets has been established at the modelling level with unified description schemes characterizing a formal syntax and common data semantic. Without doubt, Linked Open Data is the de-facto standard to publish and interlink distributed data sets in the Web commonly exposed in SPARQL endpoint. As a side effect, the Linked Open Data movement fosters open data activities, such as open government data. Major administrative authorities already publish their statistical data in a Linked Data aware format. Here, one has to mention the Eurostat database, a leading provider of high quality statistics on Europe driven by the European Commission. In this regard, one way to create an added value is to apply data warehousing techniques to analyse and aggregate statistical data. Inside data warehouses, the central data structure is the data cube model, which is a multi-dimensional dataset with an arbitrary number of dimensions. In addition to the dimensions, attributes and measures are used to describe certain observations.

This deliverable describes the outcome of Task 3.4 "Refinement & Evaluation" of WP3. In the following, the prototypes will be briefly presented and the related evaluations and statistics are discussed. The following prototypes are covered:

- ***Bacon*** presenting an open source framework that enables interactive and crowed-sourced Data Integration on Linked Data (Linked Data Integration), utilizing die RDF Data Cube Vocabulary and the semantic properties of Linked Open Data.
- ***Balloon Overflight*** builds a Linked Open Data index of available structural information. The aggregation of this data in one place enables balloon Fusion and balloon Commonalities to ease the access to Linked Data and discovery new knowledge.
- ***MindMeister endpoint*** is a periodically updated triple store storing semantically enriched public mind maps.

The reported evaluation builds on the previous deliverable D3.3 – Refined Federated Querying and Aggregation, where details of the prototypes are covered.

2. Bacon

Bacon is an open source framework and enables interactive and crowed-sourced Data Integration on Linked Data (Linked Data Integration), utilizing the RDF Data Cube Vocabulary and the semantic properties of Linked Open Data. At first, Section 12.1 will describe the specific usage of the RDF Data Cube Vocabulary. Section 12.2 will give a short outline, describing the general workflow and the realized features. Subsequently, the evaluation of bacon will be presented in section 12.3.

2.1 The RDF Data Cube Vocabulary

The primary use of the RDF Data Cube Vocabulary is the sharing and publishing of multi-dimensional. This vocabulary provides a structure to describe numerical facts, similar to the OLAP cubes, known from Data-Warehousing systems. Using this W3C Recommendation brings advantages besides providing an open standardized data format.

- A common public data format strengthens internal and external interoperability. The vocabulary can function as an interchange format for multiple internal services and allows the import of external data sets.
- The RDF-based nature facilitates extensibility, what makes it easy to augment the vocabulary with project specific concepts without violating the validity of a cube in matter of the RDF Data Cube Vocabulary specification.

This implementation utilizes a subset of the available concepts from the vocabulary. These concepts and their relationships will be explained here to get a better grasp of the usage. A complete graphical overview can be found in Appendix A.

The final goal is to store multiple observations. A single observation is the numerical fact together with a set of values, which act as a unique key for the fact. In this context, facts are called measures and the values are called dimensions.

Basically, a valid data cube consists of a dataset structure definition and a set of observations, which are compliant to the structure. Figure 1 shows such a structure definition of a cube. Measures and dimensions are attached as components of the structure. For simplicity, the multi-dimensional model is reduced to a minimal amount of components in this example. In this case, the measure *Value* and a single dimension *Product* are present.

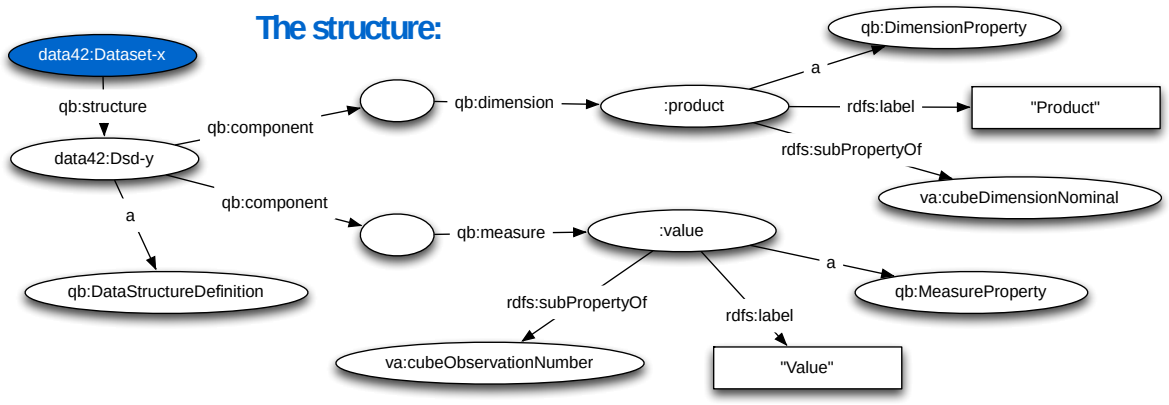

Figure 1: The structure definition of a cube

Now every observation of this dataset must reuse these component concepts to define the observed facts. Figure 2 depicts a single observation for this cube. Differing to the cube vocabulary, the concept Entity is introduced. Therefore, it is possible not only to provide the dimension value as a label, but also to name an optional disambiguated resource for the label. Notice the reuse of the properties *:product* and*:value*, which are defined by the dataset structure definition.

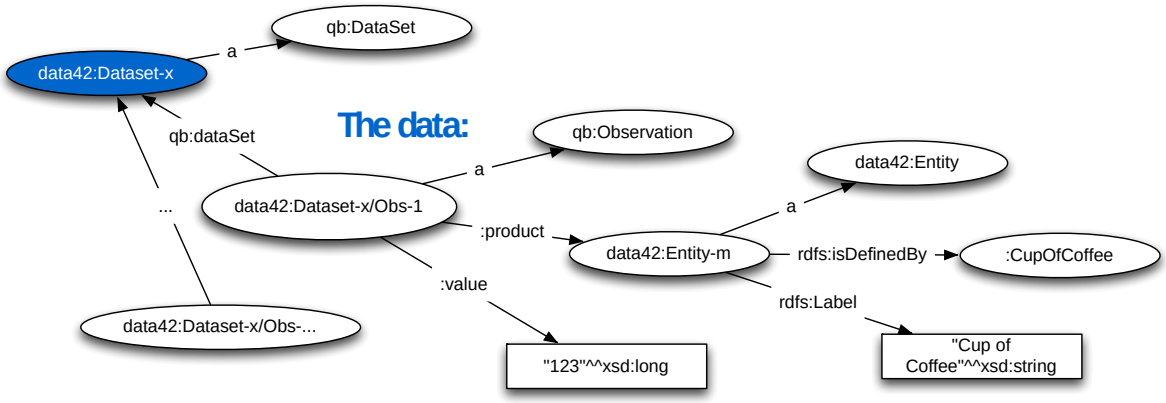

Figure 2: An instance of an observation

So far structure and data of a cube are handled. To add project specific information, additional properties are listed in Figure 3. The RDF Data Cube Vocabulary does not necessarily mention the properties but they can be subsumed as metadata.

- **format:** Denotes the current version of the used vocabulary. The version is independent from the stage of development of the actual RDF Data Cube Vocabulary.
- **label:** A short description, or the name of the cube.
- **comment:** A description of the cube.
- **date:** A timestamp denoting the time of creation.
- **source:** The data source. For example the SPARQL query or a URL of a file.
- **relation:** The source application. For example "Bacon"
- **wasDerivedFrom:** If the cube is the result of a merging process the source cubes are attached via the derived from properties.
- **wasGeneratedBy:** This construct is used to identify the user, who is responsible for the merging process.

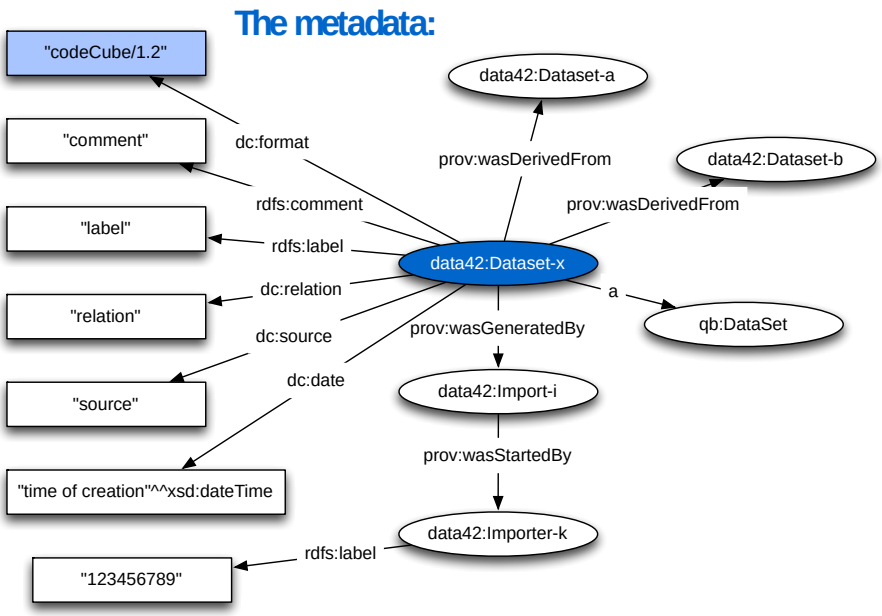

Figure 3: The metadata stored with every cube

The vocabulary was extended to fit the needs of all prototypes (e.g. usage of the Entity concept). A full explanation of the adaptions can be found in the deliverable [Granitzer, 2014].

2.2 Workflow overview

The core contribution of bacon is the Data Integration feature for cubes that use the RDF Data Cube Vocabulary. To put this into practice, workflows have been developed to discover and merge cubes. The main workflow is composed as followed: After discovering similar cubes, which use the RDF Data Cube Vocabulary, the user can modify the structure of the cubes before they are merged into a single cube.

The RDF Data Cube Vocabulary is used as the underlying data format. Diverging from the W3C Recommendation, some new concepts are introduced to carry out the needs of Data Integration. For example, the PROV-O vocabulary is used to map the provenance information of the data and the integrating user.

Given a large set of cubes, it is difficult for the user to find suitable cubes for merging in the local repository. Therefore, a ranking function was developed to sort the available cubes based on their dataset structure definitions. This feature can also be used to discover unknown cubes in the remote LOD Cloud as a possible crawling pre-step.

There are some assumptions to enable consistent merging. A component is part of the target schema if it occurs also in both source cubes. Every other component will not be carried over to the target cube. The observations are handled accordingly. Only the values for the relevant components are carried over to the target cube. Similar to composite primary keys in relational databases, the value combination of the dimensions must be unique to identify the measures. Naturally, this constraint can be violated during the merging process. bacon resolves this by performing an update of the values. These constraints allow a headless and fully automated merging process.

A graphical user interface was developed to enable the user to influence the merging process. Hereby, the structure of the source cubes can be modifies. For example, new dimensions with default values can be added, and unmatched components can be merged. Also there is a component that

informs the user about key integrity violations, which will lead to a value update. External services like balloon or the disambiguation service are integrated to support the user.

A more detailed and technical explanation can be found in the deliverable [Bayerl, 2014].

2.3 Evaluation

This section will evaluate the features of bacon. Therefore, different use cases are conducted to check functionality and performance of the components. The following itemization summarizes the test cases.

1. Use all provided functionality to merge three datasets via the GUI. The ranked list is used to find the merge partners. Add dimensions and merge them before the final cube is persisted. (See section 2.3.1)
2. Evaluate the performance of the ranking functions. (See section **Fehler! Verweisquelle konnte nicht gefunden werden.**)
3. Headless merge: Merge 25 cubes without user interaction. (See section 2.3.3)

The evaluation is performed on a MacBook Pro Intel Core i7 with 2,8GHz. Bacon and the attached triple store (bigdata) both run on a tomcat 7 in a Java 7 virtual machine, whereby some evaluations are triggered via a JUnit test. The JUnit test and the tomcat have assigned two gigabytes of heap size in each case. Bacon sends frequent queries to the triple store and therefore, it is deployed locally to minimize the impact of the network latency.

2.3.1 Manual approach

To ensure the functioning of the merger backend and the graphical user interface, a series of steps were developed, which cover every feature. If every step can be performed without failure and the resulting cube is valid, a functioning of the basic features can be inferred. The use case consists of the following steps:

- Access bacon with the first cube preselected.
- Find the second cube using the ranked list of suggestions (Similarity must be 1.0 with both ranking functions).
- Select the second cube and proceed to initially merge them.
- Every component must be merged and shown in the resulting structure.
- The key integrity diagram must show an observation overlap due to the missing dimension.
- Add a dimension to the first cube with an explicit value.
- Add a dimension to the second cube with an explicit value.
- The newly added dimensions must be shown as unmatched components.
- Merge the unmatched dimensions.
- The merged dimensions now must be part of the resulting structure and the observation overlap must be resolved.
- Store the merged cube with a new label and description.

In order to evaluate this approach, we used Eurostat data. Eurostat[1] is the statistical office of the European Union and provides datasets with statistical data at European level. The CODE Data Extractor is used to lift some of these datasets into the cube format. In particular two cubes about

[1] http://epp.eurostat.ec.europa.eu/

the quantity of specific fruit trees in the European countries are selected. They have an equal dataset structure definition containing the dimensions *location* and *year* together with the measure *quantity*. As expected an observation overlap was indicated due to the missing dimension *fruit*. The corresponding information was only available via the labels of the cubes. Manually adding the dimension with the values orange and apple resolved this conflict. The merge process was passed successfully, what indicates that bacon meets the requirements.

2.3.2 Performance evaluation: ranking function

The triple store maintains an index for faster querying and the total number of triples influences its performance. To lower this influence the full set of data cubes is imported before the performance tests are started. In total, there are 3750 cubes, whereby the different test sets are distinguished by five unique provenance IDs. Therefore the total set can be divided into subsets with 250, 500, 750, 1000 and 1250 cubes. All of these cubes have the same dataset structure definition, which consists of 20 dimensions and ten measures. Here, the structure-based ranking functions show always worst-case performance, because no early structural distinction can be found. To evaluate the performance of the ranking functions, one cube of a set is compared with every cube of this set. Every test is repeated ten times and three unmeasured runs are executed in the beginning to reduce undesired influences of the operating system. Figure 4 shows the performances of the two ranking functions.

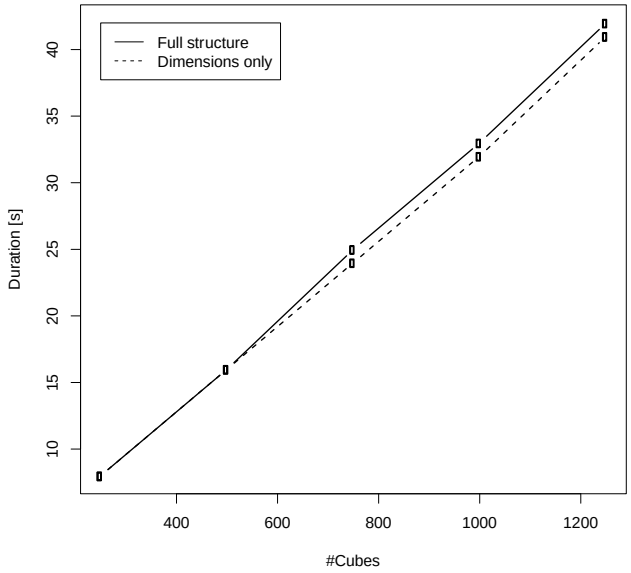

Figure 4: Performance of the ranking functions (no cache)

Figure 4 depicts a linear dependency between time and the number of considered cubes. Further research showed that this runtime is not mainly influences by the computation of the scores, but by querying the triple store. Implementing a simple cache mechanism showed a drastic performance improvement, reducing the runtime from seconds to milliseconds. Figure 5 and Figure 6 show the performance of the ranking function if the dataset structure definitions are hold in the main memory during the evaluation. Using very simple caching mechanisms for parts of the cube is unproblematic, because there are no possibilities to alter an existing cube at the moment, what could invalidate the cache.

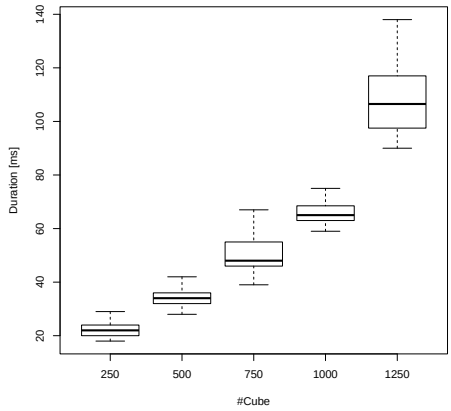

Figure 5: Consider Full structure (with cache)

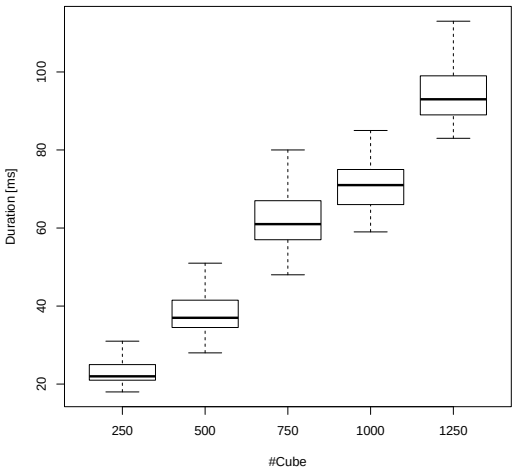

Figure 6: Consider only dimensions (with cache)

2.3.3 Headless merge

The evaluation of the headless mode was performed with 25 real world cubes. The real world data was again downloaded from Eurostat and contains information about fishery landings in the European Union over multiple years. Every cube contains the data for a single country, whereby every cube has three dimensions (*year*, *country* and *type of fish*) and one measure (*total landings in Euro*). Every single cube contains multiple thousands of observations whereby the total sum of observations is 41,776. An identical structure definition in every cube enables the headless mode to work properly, whereby the distinguishing dimension *country* prevents observations from overlapping.

Figure 7 depicts the number of observations in the resulting cube after every merge iteration. This information is relevant to be able to interpret Figure 8, which depicts the accumulative time consumption for the merge iterations. For example, after 270 seconds, all 25 cubes are merged and the final cube contains 41776 observations. The figures show, that the number of observations per cube influences the merging process and that the time for merging doesn't scale linearly with the

number of observations. This is due to the key integrity check, which is necessary to detect observations updates. Here again, the performance might be again highly dependant on the performance of the triple store, similar to the findings in section 2.3.2. Not storing every intermediate cube in the triple store, together with a caching mechanism is a possible approach to improve the performance. Hence, this is a functional test it hast been shown that bacon is able to merge multiple cubes without human interaction and persist a valid merged cube.

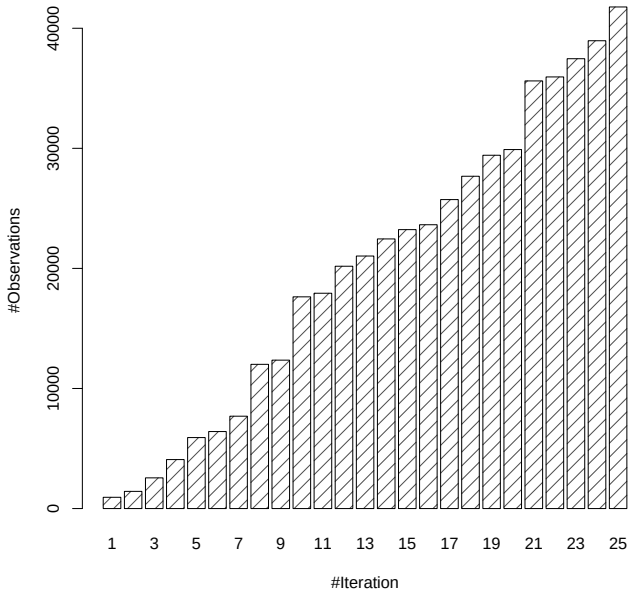

Figure 7: Number of Observations in the merged cube

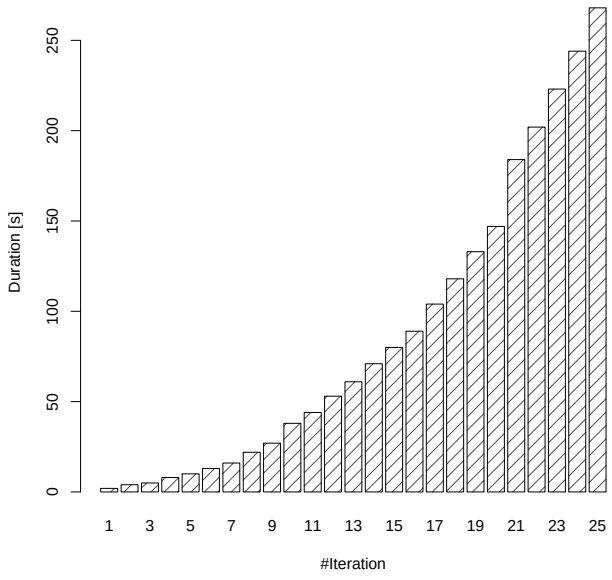

Figure 8: Cumulative time consumption for the merging process

3. Balloon Overflight

Balloon Overflight builds a Linked Open Data index of available structural information. The aggregation of this data in one place enables balloon Fusion and balloon Commonalities to ease the access to Linked Data and discovery new knowledge. Section 12.1 will introduce balloon Overflight in more detail and the conducted experiments are presented in section 2.2.

2.1 Overview

Working with Linked Open Data, you can realize that the fundamental idea of the web of data exists. Many different endpoints publish huge amount of machine-readable raw data. The web of data enables us to link related information in a structured way at the level of single statements. However, one of the biggest challenges, due to the distributed character of the web of data, is the discovery of and access to valuable information.

The CODE project addresses this topic with the balloon tool suite. The main objective of the balloon tool suite is the discovery of existing knowledge and inference knowledge based on commonalities of the globally described dataset. On the one hand balloon Fusion offers a simple service for SPARQL rewriting based on co-reference information (synonyms) combined with an automatic endpoint discovery for an intelligent query federation. On the other hand, balloon Commonalities tries to exploit extracted structures of Linked Data to find shared characteristics of entities and types to accomplish tasks, which range from simple type index up to data mining and knowledge inference.

Balloon Overflight forms the foundation for both mentioned balloon services. It aims in the unification of basic and important "schema-interlinking"-information (e.g. equal-predicates, hierarchy and type information) of Linked Data endpoints to generate a simplified, but global, sub-graph. The creation of this knowledge base leading to the bird's-eye view is the result of a continuous SPARQL crawling process. All datahub.io registered SPARQL endpoints are regularly queried for triples, which contain equivalence, instance or inheritance predicates and than accumulated in a graph database for further use. All crawled data is additionally stored in dumps, which are available as open download in a zipped way at ftp://moldau.dimis.fim.uni-passau.de/data/.

2.2 Experiments

As already reported, the first prototype (mid 2013) of balloon Overflight was applied on all SPARQL endpoints, which are registered on the datahub.io platform. At that moment, we identified 237 Linked Open Data endpoints exposing a SPARQL endpoint and therefore serve as candidates for our indexing service. Unfortunately, only a small subset of endpoints was exploitable without manifesting issues. Eventually 112 endpoints were successfully queried for co-reference information resulting in an amount of approx. 12M relations.

On basis of this first prototype the features of balloon Overflight were extended and the crawling process improved. The current implementation considers inheritance (rdfs:subclass) and instance information (rdf:type) besides the already mentioned co-reference information. The experiment of crawling Linked Open Data endpoints was repeated to determine improvements.

As first remark, the data basis changed dramatically. The Linking Open Data Cloud group of datahub.io now only contains 142 SPARQL endpoints instead of 237 due to quality issues like insufficient interlinking or missing meta-data information. This time, 87 endpoints responded with information which leads to an enhanced responds rate of ~61% (former ~47%). Besides this, the quantity of retrieved co-reference increased as well. In contrast to 12M relations from the first experiments, this time we encountered almost 52M co-reference relations despite the fact that

fewer endpoints were available (see Table 1). It showed up, that DBpedia[2] is the main contributor with approx. 32M co-reference relations, which results in ~63% of the total co-reference information. Considering all endpoints, which contributed co-reference information, the average count of equivalence relations is about 1.15M statements, which is one order of magnitude greater than the first results.

	total	avg	max
EQUIVALENCE	51.684.679	1.148.548	32.317.588
INHERITANCE	8.482.498	114.628	2.216.828
INSTANCE	133.341.501	1.646.191	32.150.237

Table 1: Relationships indexed by balloon Overview

Furthermore, in this iteration of the experiment, inheritance and instance information were likewise indexed by balloon Overflight. As a result over 133M *rdf:type* statements were collected from the distributed Linked Open Data endpoints. Again, DBpedia is a main contributor with over 32M statements but closely followed by the medical domain with over 25M statements from bio2rdf-pubmed[3] and 21M statements from ChEMBL[4](a medicinal chemistry database). As expected, the number of collected inheritance information is much lower, because inheritance information is on the schema level (concern classes instead of concept instances). Balloon Overview was able to gather about 8M *rdf:subclass* information which are now available for structural analysis and inference.

Some general remarks are that the overall quality of SPARQL endpoints performance improved slightly. The bio2rdf-pubmed, for example, endpoint returns a maximum pagination size of 10k statements, which took previously over 1.5 minutes in average and now only took 43 seconds in average. Additionally the pagination size of many endpoints increased as well. As an example, DBpedia raised the pagination size from 1k to 50k, which enhanced the entire query time significantly. The experiments highlight, that the endpoint maintainer make efforts to enhance the accessibility for Linked Data.

[2] http://datahub.io/dataset/dbpedia

[3] http://datahub.io/dataset/bio2rdf-pubmed

[4] http://datahub.io/dataset/farmbio-chembl

4. CODE Mindmeister SPARQL

Using mindmaps is a well known and commonly used mechanism to structure your thoughts and ideas. Starting from a central point, a tree-like structure combined with named nodes and possibilities of annotations offers the visual arrangement of processes, components, business-plans, TODO-lists and so forth.

This data is the central point of the endpoint that has been established in the course of the CODE project. Combined with an RDF storage and management endpoint (we used virtuoso in this case) the knowledge contained in every public mindmap will be stored in a machine-readable fashion. Consecutively, a crowd-sourced taxonomy was created which offers access of the extracted data over various interfaces, for example SPARQL or a free-text search.

In terms of vocabulary, different common standards come into operation. SKOS and PURL, as well as RDF/RDFS are utilized in combination with some own entities and relations. The RDF data of a mindmap itself is divided into a description of a concept for every node. It is enriched with its title, its point of time for creation and its last modification, its creator, and its predecessor. Following this routine, the whole tree-like structure of a mindmap can be remodelled as RDF content. This little excerpt shows an example of a single node:

```
@prefix maps:<http://www.mindmeister.com/maps/show/>
@prefix users:  <http://www.mindmeister.com/users/channel/>
@prefix rdfs:<http://www.w3.org/2000/01/rdf-schema#>
@prefix skos:<http://www.w3.org/2004/02/skos/core#>
@prefix purl:<http://purl.org/dc/terms/>
@prefix xml: <http://www.w3.org/2001/XMLSchema#>

maps:388245497 a skos:Concept;
   rdfs:label "Chapter 1: Introduction";
   purl:created "2014-03-17T08:28:04Z"^^xml:dateTime;
   purl:creator users:2328257;
   purl:modified "2014-03-17T08:28:04Z"^^xml:dateTime;
   skos:broader maps:388244360 .
```

The last line using the relation <http://www.w3.org/2004/02/skos/core#broader> links this node to its predecessor with the given specific ID. Notes or references between nodes of the mindmap can also be modelled via specific relations, linking to the other node via its ID or containing a text content respectively. Images can be applied to nodes, too. They are specified being of the type <http://purl.org/dc/terms/image> and then being referenced from the associated node.

By storing knowledge according to this fashion, different possibilities in terms of querying and exploring the data emerge. For example, you can now search all the given mindmaps for the same keywords or topics (using the free text search offered by the endpoint technology).

The most recent iteration of the whole update process (which was done on April 24. 2014) crawled a total of 66,994 mindmaps containing 32577456 nodes, resulting in an average of about 486 concepts per map. Taking about 40 minutes of time, this was a complete run that requests all mindmaps that are currently available. After that, every week only new and altered mindmaps will be crawled and updated at the endpoint.

5. Conclusion

In this report we have presented the evaluation of the three prototypes developed in the CODE Project for Linked Open Data Discovery and management of statistical data.

First, our interactive and crowed-sourced Data Integration on Linked Data (Linked Data Integration) framework bacon shows reasonable performance for conducting large scale cube merging on open data sets. However, usability of workflows still require polishment in order to enable usage by IT Laymans.

Second, the evaluation of Balloon, our Linked Open Data index of available structural information, showed high performance for crawling SPARQL endpoints and associated services around it. With over 190M triples we have very complete structural information of the SPARQL endpoints in the Linked Open Data cloud. Further, the structural information allows us providing common functions like endpoint discovery, SPARQL Query rewriting or calculating structural similarity measures for data mining algorithms.

Third, we presented the CODE MindMeister endpoint as a source of over 66,000 queryable thesauri. The endpoint opens mind maps for being re-used in the Web of Data.

6. References

[Bayerl, 2014] S. Bayerl, E. Berndl, M. Granitzer, K. Schlegel, and F. Stegmaier, *"D3.3 – Refined Federated Querying and Aggregation,"* Deliverable of the CODE Project, 2014.

[Granitzer, 2014] M. Granitzer, B. Mutlu, V. Sabol, and F. Stegmaier, *"D7.3 – Standardization Proposal,"* Deliverable of the CODE Project, 2014.

7. Appendix A: Example Table Transformation into a Data Cube