D3.3 – Refined Federated Querying and Aggregation

Summary: TBD

<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>2014-01-14</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>University of Passau</td>
</tr>
<tr>
<td>Revision</td>
<td>1.0</td>
</tr>
<tr>
<td>Authors</td>
<td>Sebastian Bayerl, Emanuel Berndl, Andreas Eisenkolb, Michael Granitzer, Kai Schlegel, Florian Stegmaier</td>
</tr>
</tbody>
</table>

Consortium:

Statement of originality: This document contains original unpublished work except where clearly indicated otherwise.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, 2012
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, Content and Technology (DG CONNECT)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unit Data Value Chain (G3) 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>Infeldgasse 13, 8010 Graz, Austria</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+43 316 873-30800 (phone)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+43 316 873-1030800 (fax)</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, D-94032 Passau</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+49(0)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>Bayerl, Schiegel, Stegmaier</td>
<td>UNI PASSAU</td>
<td>Added table of contents and initial draft.</td>
</tr>
<tr>
<td>2</td>
<td>2014-01-02</td>
<td>Granitzer</td>
<td>UNI PASSAU</td>
<td>Document review</td>
</tr>
<tr>
<td>3</td>
<td>2014-01-04</td>
<td>Bayerl</td>
<td>UNI PASSAU</td>
<td>Improved Section 3</td>
</tr>
<tr>
<td>4</td>
<td>2014-01-10</td>
<td>Berndl</td>
<td>UNI PASSAU</td>
<td>Improved Section 2.5</td>
</tr>
<tr>
<td>5</td>
<td>2014-01-12</td>
<td>Schlegel</td>
<td>UNI PASSAU</td>
<td>Improved Section 2.1 – 2.4</td>
</tr>
<tr>
<td>6</td>
<td>2014-01-14</td>
<td>Stegmaier</td>
<td>UNI PASSAU</td>
<td>Consolidated document</td>
</tr>
</tbody>
</table>
# Table of Contents

1 Introduction - 3

2 Federated querying and caching (Task 3.1) - 5
   2.1 APPLICATION SCENARIO - 5
   2.2 CONCEPTIONAL ARCHITECTURE - 6
   2.3 INDEXING LINKED OPEN DATA - BALLOON OVERFLIGHT - 6
   2.4 TOWARDS AUTOMATIC QUERY FEDERATION - BALLOON FUSION - 8
   2.5 DEPARTURE FOR STRUCTURAL KNOWLEDGE - BALLOON COMMONALITIES - 9
   2.6 TRIPLIFICATION OF RESTFUL APIs - 10

3 Aggregation of Linked Data (Task 3.2) - 13
   3.1 APPLICATION SCENARIO - 13
   3.2 DEFINITION OF DATA CUBES - 13
   3.3 MICROFORMAT FOR ANNOTATION OF HTML TABLES - 15
   3.4 DATA EXTRACTOR - 16
   3.5 CUBE MERGING - 19
   3.5.1 PROBLEM SETTING - 19
   3.5.2 RELATED WORK - 20
   3.5.3 APPROACH - 21
   3.5.4 CHANGELOG - 24
   3.6 RDF DATA CUBE IMPLEMENTATION REPORT - 25

4 Related Work - 27

5 Conclusion - 29

6 References - 30

Appendix A: Example Table Transformation into a Data Cube - 32
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 ★ 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.

Work Package 3 of the CODE project aims towards the ease consumption of Linked Open Data through RESTful services. The developed services and prototypes introduced in this deliverable covered the following two tasks:

- Federated Querying (Task 3.1)
- Linked Data Aggregation (Task 3.2)

Besides the two tasks, Task 3.3 developed a provenance schema to serve as a basement for revenue models and provisioning chains. A complete consideration can be found in the public Deliverable “D3.2 – Provenance Schema”.

The development of the presented prototypes was split in two agile phases: The first phase lasted 12 months and aimed in the establishment of prototypes offering core features. Evaluations and refinement of the prototypes coined the second phase, which took six months in sum. Next, for each tasks the core contributions will be summarized:

**Federated Querying (Task 3.1)**

The outcome of prior investigations of WP3 clearly showed the need for an intelligent discovery of Linked Open Data endpoints since replication of data would be a waste of resources. The idea of establishing a mediator service has been pursued and led to the Balloon prototype in the first phase. This prototype crawled OWL sameAs relations available in reachable SPARQL endpoints registered on the CKAN\(^1\) platform resulting in an amount of approx. 12M relations. Those were clustered with respect to equal concepts and in order to serve as a basis for further calculations. This service is the foundation for intelligent query rewriting. Due to the fact that concepts in a SPARQL query are mostly described by an URI, the Balloon service can be queried with each of those URIs and responds with endpoints storing the same concepts. The initial SPARQL query is then rewritten with this information utilizing the already issued SPARQL 1.1 Query Federation for routing tasks. In the second phase, it was observed that Balloon was even more then a query federation service. A redesign let to a platform enabling Linked Data as a Service with the crawled data as a basement.

Further, mechanisms have been applied to lift REST APIs into the Linked Open Data cloud, in particular the APIs of Mendeley and Mindmeister. We developed a schema for both APIs and are able

---

\(^{1}\) http://www.ckan.org/
to create RDF Dumps of the underlying data. However, to be usable in a SPARQL Query we aim to include REST APIs in the query federation process.

Linked Data Aggregation (Task 3.2)
As already mentioned earlier, statistical data is a key driver in Open Data initiatives. In the first phase of the CODE project, the RDF Cube Vocabulary has been chosen to model statistical data in a semantic web enabled way. Along with that, a first prototype of the Data Extractor for triplifying primary research data has been developed. After its development in the first phase the prototype got a complete makeover in terms of the internal processing stages with respect to the following topics:

- **Data Cube Storage**: Experiments in terms of scalability have shown that the BigData SPARQL server fits the needs for the CODE project best and has therefore replaced the Linked Media Framework. During the workflow, provenance information is tracked and added to the produced data cubes.
- **Input formats**: Besides PDF documents or Excel files, a sophisticated RESTful API has been integrated offering the possibility to transfer large portions of primary research data to the Data Extractor from client applications.
- **Export formats**: To foster a wide applicability of our data ecosystem to the research community two further export formats have been integrated: Comma-Separated Values (CSV) and Attribute-Relation File Format (ARFF). Both formats are well known in the data mining community and lots of tools exist to conduct data analysis.
- **Unified Processing chain**: Irrespective of the input format, a unified processing chain has been established to increase usability of end users.
- **Robustness**: Several expert users have evaluated the prototype within the last six months. The feedback has been integrated and the workflows refined accordingly.

By the help of the Data Extractor, it is possible to create meaningful and large data cubes from a broad application domain.

Based on this result the first phase already introduced methods and prototypes for merging independent data cubes. Merging of data cube has been identified as central point for future developments and key feature to generate new insights in Open Data through unknown data correlations. In order to provide reliable data of high quality, users have to be enabled for merging independently created data cubes. Within the first phase, abstract merging scenarios have been defined and a subset already implemented. Those efforts have been continued in the refinement phase and led to a redesign of the initial user interface as well as the integration of further merging scenarios.
2 Federated querying and caching (Task 3.1)

Task 3.1 of WP3 focuses on the research and development of a query federation component. The query federation component has to be aware to act as a mediator between the Linked Open Data Cloud and a client application, such as the Query Wizard developed by WP4. On basis of the first prototype the feature set of the balloon tool suite were extended comprehensively. Therefore the former balloon prototype has been highly modularised in the refinement phase of CODE. The task of crawling Linked Open Data is now established under the name of balloon Overflight with an extended set of crawled predicates to allow a simplified but global view on structural interlinking. The former balloon service for SPARQL rewriting was renamed to balloon Fusion [Schlegel, 2014] and utilizes the new designed underlying data structure produced by balloon Overflight. As a result of the new supported predicates (see section 2.5), the balloon Commonalities services exploits an extracted structural hierarchy to find shared characteristics of entities and types. To assure a future-proof development the complete balloon tool suite is going to be open source. So everybody can suggest or add features, help bug fixing and reuse the components for other tasks. The balloon tool suite is available on GitHub:

https://github.com/schlegel/balloon

An overview website was created and is reachable via

http://schlegel.github.io/balloon/

Besides balloon, mechanisms were developed to lift generic RESTful APIs into the Linked Data Cloud to enlarge average information entropy.

2.1 Application Scenario

Statistics\(^2\) of the Linked Data Cloud collected by the LOD2 project indicate that it consist of approx. 700 data sets. The numbers further state that only half of the data sets are syntactically valid and only 50 are exposed by a SPARQL endpoint. Regardless the pro-rata small amount of actual accessible Linked Open Data endpoints via a SPARQL interface, the Semantic Web community spots the need for an intelligent query federation service.

As an application scenario for the query federation service, let’s consider the Query Wizard as entry stage for user requests. Within the Query Wizard, the user wants to find more information on an arbitrary input in a Google-like manner. In this case, the following issues are present:

- The user does not know, what particular data or domain is reachable by the Query Wizard.
- Due to the notion of data browsing or knowledge discovery, the user is not aware, what exactly he wants to find or how the data will be structured.

A concrete retrieval scenario could be as follows: The journalist wants to write an article on Michael Jackson and therefore tries to collect information on album releases and so on. Since the journalist is not an expert in information retrieval, nor in Linked Open Data, he is not able to choose an appropriate endpoint as an initial stating point, such as Magnatune\(^3\) or DBPedia\(^4\).

The query federation service improves this situation by acting as a mediator service between a client application and a set of actual Linked Data endpoints as single point of access. Briefly summarized, the query federation service knows all occurrences of similar concepts (described by one or more Linked Data URIs) existent in reachable Linked Data endpoints. On the basis of this information, it

\(^2\) http://stats.lod2.eu/
\(^3\) http://www.dbtune.org/magnatune/
\(^4\) http://www.dbpedia.org/
performs an automatic and intelligent query rewriting whereas routing of the query is managed by utilizing the SPARQL 1.1 Query Federation\(^5\) recommendation. In the following, the components of the query federation service are discussed.

### 2.2 Conceptional architecture

To get a better overview over the developed prototypes of WP3, Figure 1 illustrates an overall end-to-end retrieval workflow consistent with the above application scenario. In general two different workflows have to be differentiated:

i. Continuous crawling process

ii. Single retrieval process

![Figure 1: End-to-end retrieval workflow of developed WP3 prototypes](image)

The first workflow manages the creation of a knowledge base. The central component for it is the balloon Overflight prototype, see Section 2.3. It crawls all sameAs relations and several other interlinking predicates of reachable SPARQL endpoints from the Linked Open Data cloud and stores them in a Neo4j\(^6\) graph database instance for a global but simplified view on the information graph. The second is the actual query federation. Within this process, the query string is analysed and for all occurrences of concept URIs balloon Overflight will be called, whether it has information of equal relations in further Linked Data endpoints. In the positive case, the query is enriched by those similar concepts and reformulated into the SPARQL 1.1 Query Federation syntax. Routing and aggregation of the single query segments will be done inside BigData, which is fully compliant to the W3C standard.

### 2.3 Indexing Linked Open Data - Balloon Overflight

Following the main idea of the Linked Open Data project, data within the Linked Open Data Cloud is highly interlinked, openly available, and can be queried separately on each endpoint. Harmonization at the modelling level is established due to unified description schemes, such as RDF, however, a convenient request considering the global data set is only possible with strong limitations. Given a specific concept, it is hard to find all relevant information, which is distributed over several endpoints. This problem is based on the fact, that a view on all equivalent concepts in the LOD (co-referencing problem) is not given and the existence of uni-directional links instead of bidirectional ones. Let us consider a sameAs relation from a concept X in Freebase\(^7\) to concept Y in DBPedia, there is no backlink from concept Y to concept X hindering efficient navigation or browsing inside the data graph. One would benefit from bi-directional sameAs interlinking to allow an efficient retrieval of all equivalent concepts and associated information about the concepts. A naive solution of this problem would be, to establish a local data replication of all Linked Open Data endpoints for an

---

\(^5\) [http://www.w3.org/TR/sparql11-federated-query/](http://www.w3.org/TR/sparql11-federated-query/)

\(^6\) [http://www.neo4j.org/](http://www.neo4j.org/)

\(^7\) [http://www.freebase.com/](http://www.freebase.com/)
overall view on the data. Due to the already presented statistics, this approach is not feasible due to the amount of data, unnecessary data duplication and related update problems.

To overcome this issue, balloon Overflight aims in the unification of basic and important information of Linked Open Data endpoints to generate a simplified sub-graph. In the first place a differentiation between "content"- and "interlinking"-information (e.g. equal-predicates, hierarchy and type information) is made. Using the interlinking-information, the desired bi-directional view on the Linked Open Data cloud can be created. The result is a service, which returns a list of relevant concepts and endpoints given a start-concept. This serves as a foundation for query federation by querying the relevant endpoints simultaneously to gather live content-information.

Figure 2: Illustration of the continuous crawling process of CODE Balloon

The creation of the knowledge base leading to the bi-directional view is the result of a continuous crawling process visualized in Figure 2:

- **Endpoint Manager**: The endpoint manager is able to query the CKAN\(^8\) open source data portal API\(^9\). With this feature, it is possible to collect metadata information on all currently registered Linked Data endpoints. This information is filtered to generate a set of endpoints reachable by a SPARQL endpoint. RDF dumps or not reachable endpoints will be ignored in this step.

- **Indexing Process**: The indexing process calls each endpoint for its sameAs relations. Those will be stored in the graph database for further processing. From a technical point of view, the crawling of those relations is a hard task due to retrieval limitations (quota) and responsiveness of the endpoints.

- **Graph database**: From the technical point of view we are using a Neo4j database to store the indexed data. The graph structure allows live querying and analysis to get new insights. In our case the graph database allows an easy access to "equal"-sets of a given entity. The traversal of sameAs relationships collects all affected entities rapidly.

- **Public Dumps**: All retrieved triples are stored in dumps, which are available as open download in a zipped way\(^10\). This allows other projects to easily utilize our outcomes.

\(^8\) [http://www_ckan.org/](http://www_ckan.org/)

\(^9\) [http://ckan.org/features/api/](http://ckan.org/features/api/)

\(^10\) [ftp://moldau_dimis_fim.uni-passau.de/data/](ftp://moldau_dimis_fim.uni-passau.de/data/)
In the prior version of balloon we used a MongoDB instance because of heavy read workload. Our experiments showed up, that the equivalence cluster centric approach didn’t scale out as we expected. The extension of the crawled predicates (see section 2.5) amplified this problem and motivated us to modify our database setup. The new graph database approach allows live querying of equivalence cluster and further predicates without upfront calculations.

2.4 Towards Automatic Query Federation – Balloon Fusion

As Figure 1 shows, the balloon Fusion Query Federation component builds upon the balloon Overflight service. To illustrate the already outlined workflow, an example query transformation and rewriting will be sketched. A central focus of the query federation service is to extend the query by further Linked Open Data endpoints without changing the enclosed query semantics.

Let us assume the query given in Listing 1. This query retrieves further information on the Austrian skier named “Thomas Morgenstern” expressed by a Semantic Web URL.

```sparql
SELECT ?p ?o WHERE {
}
```

**Listing 1: Initial SPARQL Query**

The query federation prototype analyses the initial query in order to discover the occurrence of URIs. Those URIs are then forwarded to the balloon Overflight service to receive their equal-set. The response for the above query is shown in Listing 2. Besides status information it inherits the equal-sets containing all semantically identical URIS for the given concept and the input URI as provenance information.

```json
{
  "status" : "200",
  "sameAs" : ["http://dbpedia.org/resource/Thomas_Morgenstern",
               "http://vocabulary.semantic-web.at/AustrianSkiTeam/121",
               "http://rdf.freebase.com/ns/m/08zld9"],
  "query" : "http://vocabulary.semantic-web.at/AustrianSkiTeam/121",
  "response" : "FOUND"
}
```

**Listing 2: Response of balloon Overflight to a Semantic Web URI**

The next phase considers the construction of a SPARQL equivalence block. This means, that all retrieved URIs will be inserted into the query combined with the UNION operator, see Listing 3.

```sparql
UNION
{<http://vocabulary.semantic-web.at/AustrianSkiTeam/121> ?p ?o.}
UNION
{<http://rdf.freebase.com/ns/m/08zld9> ?p ?o.}
UNION
```

http://www.thomasmorgenstern.com/
Listing 3: SPARQL equivalence block

Balloon Overflight can be also queried, which URI belongs to which endpoint in the Linked Open Data cloud. It is possible that a URI is not resolvable and in this case will be eliminated of the equivalence block. In the example this is the case for http://rdf.freebase.com/ns/m/08zld9. For the remaining URIs, endpoints have been deposited and the equivalence block is enlarged with the SERVICE clause defined in the SPARQL 1.1 Query Federation. The complete result of this transformation can be found in Listing 4. It is observable, that this transformation features an exponential growth in the overall query size. Due to this fact techniques for efficient pruning while the selection of URIs as well as algebraic optimization techniques have to be developed in year two of the CODE project.

The current prototype of the query federation is reachable under: http://purl.org/balloon/demo

```
SELECT ?p ?o WHERE {
  {
    {<http://vocabulary.semantic-web.at/AustrianSkiTeam/121> ?p ?o.}
    UNION
    {<http://rdf.freebase.com/ns/m/08zld9> ?p ?o.}
    UNION
  } UNION {
    SERVICE <http://vocabulary.semantic-web.at/PoolParty/sparql/OpenData> {
      {<http://vocabulary.semantic-web.at/AustrianSkiTeam/121> ?p ?o.}
      UNION
      {<http://rdf.freebase.com/ns/m/08zld9> ?p ?o.}
      UNION
    }
  }
  UNION {
    SERVICE <http://vocabulary.semantic-web.at/PoolParty/sparql/OpenData> {
      {<http://vocabulary.semantic-web.at/AustrianSkiTeam/121> ?p ?o.}
      UNION
      {<http://rdf.freebase.com/ns/m/08zld9> ?p ?o.}
      UNION
    }
  }
}
```

Listing 4: Complete Rewritten Query of the CODE Query Federation service

2.5 Departure for structural knowledge – Balloon Commonalities

As things developed the balloon Overflight services was extended to crawl more predicates than only equivalence (e.g. owl:sameAs) relationships. The new predicates now include structural relationships like instances (rdf:type) and class inheritance (rdfs:subClassOf). As a result, the graph database now holds a structural hierarchy, which can be exploited by the balloon Commonalities service. Strictly speaking the graph doesn’t show a clean hierarchy or taxonomy, because characteristic of Linked Open Data show a dense heterarchy without single permanent uppermost types or entities. Instances are likely be related to many types or super types including multiple inheritance. The
crawled dataset contains information for concept bundles. A concept bundle includes all equivalence, type and inferred type (based on subclasses) relationships about a semantic concept. Consequently, a concept bundle would contain all synonyms URIs, types and provenance information like data origin. The main target of this approach is to create a global index of basic structural interlinking information whereas the real “content”-information is not particularly interesting for the index and can be queried efficiently afterwards.

The structural graph is the foundation for understanding the nature of Linked Open Data and can be utilized to accomplish analysis tasks. Balloon Commonalities offers a range of different analysis services:

- **Type Index**: The graph holds a implicit index of different semantic types and can be queried for example to get all encountered instances of data cubes which are spread over many LOD endpoints.

- **Type Matching**: Exploiting the type index, the balloon service can find similar instances to a given concept based on common inheritance (e.g. find different people who are ski jumpers based on the concept of Thomas Morgenstern). This can be especially beneficial to extend available table data or data cubes. On the other side, given multiple concepts the service can also identify which types are important and shared among all participating concepts (canonical types). Using this service for example in a user interface could allow a simplified view on the information, because complex structural knowledge can be condensed to only important relationships.

- **Type Distribution**: Linked Data concepts feature a large number of type relationships. The co-occurrence of different types reveals semantic connections between types. Associate rule mining can provide insights and show up related or unrelated types. This information can be used for disambiguation contexts or type inference.

- **Structural Distance**: The structural graph can be queried for a path between two given concepts. This allows a distance calculation based on sibling & parent relationships. For example Josef Bradl and Thomas Morgenstern have the distance 1 because they have the type AustrianSkiJumpers in common.

- **Further analysis**: Querying further predicates of type instances can offer the possibility of building a kind of fuzzy scheme of Linked Open Data. For instance, almost all instances of the type Person have a predicate birthplace. These predicates play an important role in unstructured data like the semantic web because probable expectations can be used for efficient querying. Another future analysis is the distribution of related information of concept bundles spread over multiple endpoints. In other word: How many different endpoints are involved to define a concept bundle.

2.6 Triplification of RESTful APIs

The introduced statistics of the Linked Data Endpoints landscape clearly indicate that the progress of establishing reliable endpoints is not yet a finished task. Lots of data sources are only available as data dumps and only a fractional amount of endpoints operate within suitable quality of service parameter.

Within the CODE project, it is highly desired to publish the extracted as well as user generated data in a Linked Open Data aware manner. Following this regard, two endpoints have been, one for each of the industrial partners, namely Mindmeister and Mendeley. To overcome the shortcoming of the prior mentioned maintenance issues, an automated service process has been implemented to trigger a periodic update of changed data.

In detail, the following information is store in the according endpoints:
• **Mindmeister endpoint**: As shown in other CODE deliverables, mindmaps support the user to visually manage the extracted and generated data, e.g., produced by the CODE Query Wizard. Further, the encompassed information can be easily changed and shared with others. To follow the idea of Linked Open Data, the Mindmeister API has been extended to dump each public mindmap using RDF.

• **Mendeley endpoint**: The CODE PDF services provided by WP2 are capable of extracting meaningful information of PDF documents (such as title and TOC) whereas named entities are enriched with semantic concepts of the Linked Open Data cloud. The extracted and enriched data is stored in a single JSON-LD file for each PDF document. This information will be aggregated and stored in the triple store accordingly.

![Diagram](image)

**Figure 3: Sequence of an Update for Mindmeister**

Figure 3 shows an example sequence of the integration process for CODE specific Mindmeister data. The entity called “Mindmeister” stands for the API\(^\text{12}\) that is hosted by Mindmeister, “UpdateService” depicts the automated service embedded in the CODE server and “Endpoint” is the bigdata endpoint\(^\text{13}\) where the RDF triples are stored. The Mindmeister specific endpoint can be accessed at: [http://zaire.dimis.fim.uni-passau.de:8080/bigdataMindmeister/](http://zaire.dimis.fim.uni-passau.de:8080/bigdataMindmeister/)

One single iteration of the process contains the following steps:

- **Step 1**: This step is only executed once and can be done in two alterations. Retrieving all IDs of the currently available public mindmaps will result in a full update of the endpoint and its RDF content. A partial update only fetches the IDs for mindmaps that have been altered since the last update. After a full update, the service will continue updating its RDF content once every week with a partial update. A full update can be easily achieved via changing a property.

- **Step 2 and 3**: These two steps are fulfilled for every mindmap who’s ID is contained in the list retrieved by step 1. Step 2 requests the Mindmeister API to respond with the RDF content of the mindmap corresponding to the retrieved ID. This content is then sent as a SPARQL Update query\(^\text{14}\) to the local bigdata endpoint in step 3. The combination of step 2 and 3 is implemented in a multi-threaded way in order to achieve a higher efficiency.

---

\(^{12}\) [http://www.mindmeister.com/developers](http://www.mindmeister.com/developers)


\(^{14}\) [http://www.w3.org/Submission/SPARQL-Update/](http://www.w3.org/Submission/SPARQL-Update/)
The RDF description of a Mindmeister mindmap contains information about every concept used in the map. This spans from labels, descriptions, and the map type to provenance information like the creator, date of creation, and date of last modification. These triples make use of commonly known vocabularies like the following:

- **Dublin Core**[^15]** (dc):** DC is one of the most common approaches to describe resources in an easy-fashioned way. The core set consists of 15 descriptive and domain-independent properties.
- **Friend of a Friend**[^16]** (foaf):** Is a community project to create machine-readable abstractions of people, their relations to each other as well as the resources they are interacting with.
- **Simple Knowledge Organization System**[^17]** (skos):** SKOS is heavily used to create Semantic Web aware classification schemes, thesauri and taxonomies.

The update service for Mendeley runs in a similar way but has two differences. The data is available in the JSON-LD format and is not requested by the service itself. The JOSN-LD files are stored in a Amazon S3 bucket. A periodical task of Mendeley triggers the import service to integrate the recently added JSON-LD files. Besides the parsing of the JSON-LD file and the push mechanism, the remaining architecture follows the same architecture as for Mindmeister. The endpoint is reachable at:

```
http://zaire.dimis.fim.uni-passau.de:8080/bigdataMendeley/
```

Currently, for both endpoints the input routines are up and running. After a short revision phase, both endpoints will be registered at the CKAN platform with their specific information.

[^15]: http://dublincore.org/documents/dces/
[^16]: http://www.foaf-project.org/
[^17]: http://www.w3.org/2004/02/skos/
3 Aggregation of Linked Data (Task 3.2)

Task 3.2 of Work Package 3 is aiming towards aggregation of Linked Data. The main scope is to develop aggregation mechanisms for statistical data in the Linked Open Data cloud. To fulfil this task, in the first phase of the project, two prototypes have been proposed for data modelling of data cubes, data creation, retrieval and merging strategies. Within the refinement phase of CODE, the supported features of the prototypes have been enlarged as well as workflows and user interfaces rearranged.

3.1 Application Scenario

The research community issues the primary research data through two major channels:

- Scientific events, such as conferences or workshops
- Evaluation campaigns

Scientific events are the most common way to promote ones research results. In each domain of science, an abundance of conferences, workshops and so on exist. Most of them arrange the accepted contributions in proceedings, which are further published by digital libraries such as Mendeley\(^{18}\) or ACM\(^{19}\). The logical consequence is the silo-based enclosure of primary research data in proprietary formats like PDF. However, those data silos do not offer the possibility to easily provide analytics or aggregation of multiple observation sources to gather a different or additional view on the underlying data. Furthermore, similar observations from diverse authors stay incomparably and unconnected.

In addition to scientific events, it is quite common, that research communities themself are running evaluation campaigns to foster the progress of the according research fields. In terms of information retrieval, the CLEF Initiative\(^{20}\) is a prominent example. It is an annual evaluation forum with the purpose to promote research, innovation, and development of information access systems. Here, several subtasks are dealing with specific issues of information retrieval. In detail, the CODE project is in contact with the PAN challenge\(^{21}\) that has the mission to uncover plagiarism, authorship, and social software misuse. Within this task, the test bed for Information Retrieval Algorithms (TIRA) [Gollub, 2012] has been developed to enable an easy and automatic test environment for participants of the challenge. The proposed services of this paper will be used to analyse, structure and lift the results of the automatic tests to support easy comparison to results of former years as well as community interaction.

The developed Data Extractor prototype copes with both channels in an abstract way to maximize its coverage and to minimize efforts during the exploration and integration of new data sources. Before diving into the key concepts of the Data Extractor, the data model for the storage of statistical data will be introduced.

3.2 Definition of Data Cubes

When speaking of Online Analytical Processing (OLAP) or Data Warehouses one soon ends up with the term data cube. In the following we use the informal definition of data cubes given by [Chaudhuri, 1997]:

\(^{18}\) http://www.mendeley.com/
\(^{19}\) http://dl.acm.org/
\(^{20}\) http://www.clef-initiative.eu/
\(^{21}\) http://pan.webis.de/
“In a multidimensional data model, there is a set of numeric measures that are the objects of analysis. [...] Each of the numeric measures depends on a set of dimensions, which provide the context for the measure. [...] The dimensions together are assumed to uniquely determine the measure. Thus, the multidimensional data views a measure as a value in the multidimensional space of dimensions. Each dimension is described by a set of attributes.”

The main feature of such a data model is the possibility to perform analytic retrieval on the data sets. Those operations are termed OLAP queries and consist of specific operands, such as roll-up or drill-down [Morfonios, 2008].

Due to the fact that Linked Open Data plays a central role in the CODE project, data cubes should be made available following Semantic Web publication best practises, namely the 5-star deployment scheme. For this kind of publishing, the data has to be available in a Semantic Web language, preferably RDF or OWL. In this regard, the Government Linked Data Working Group of the World Wide Web Consortium (W3C) created the RDF Data Cube Vocabulary for representing multi-dimensional data cubes in RDF. This vocabulary relates to the ISO standard Statistical Data and Metadata eXchange (SDMX). In the following, its basic concepts are described, how it is constructed and how it is extended to suit our needs.

Like a classic OLAP data cube the RDF data cube model is a multi-dimensional dataset with an arbitrary number of dimensions. In addition to the dimensions, attributes and measures are used to describe observations, and all taken together define the dataset. The structure of the data cube is indeed implicitly given by the structure of the observations, but there is an additional explicit definition of the structure. The concept DataStructureDefinition is designed for this purpose. Therefore, a structure definition can be applied to multiple data cubes and enables comprehensive validation or merging of data cubes. This fits our needs, since the import aggregates multiple datasets into the same data cube or extends the datasets by additional rows or columns. Also operations typical for data warehouses and visualizations are easier to realize, if an additional structure definition is available. A structure definition is composed of the definitions of the different components, namely dimensions, attributes and measures. The semantics are as follows and correlate to the informal definition above:

- **Dimension** is used to identify a specific observation, such as year or gender.
- **Attribute** describes, how the observed value must be interpreted or allows the specification of metadata on a syntactic level, e.g., unit of measure or status of the observation.
- **Measure** defines (in contrast to attributes) the semantic of the observations, e.g., life expectancy.

The actual data is therefore modelled as observations. As a consequence, an observation contains information concerning every component, defined in the structure definition and also points to its attached dataset. The dimension and attribute properties are used to denote the actual value of this component, whereas the measure property marks the fact of this observation. This means that every fact can be identified and interpreted.

Further, the vocabulary allows to group observations as slices. Syntactically these selections can be arbitrary, however recommended best practice is to fix one or more dimension values to create uniform slices through the cube. An attachment of external metadata to a Dataset is possible. The

---

22 http://www.5stardata.info/
23 http://www.w3.org/TR/REC-rdf-syntax/
24 http://www.w3.org/2004/OWL/
25 http://www.sdmx.org/
usage of label, comment and creation date are recommended, while the dataset can be marked up with arbitrary metadata like publisher or the subject of the cube. In our use case, provenance information can be stored as external metadata.

To allow for a high-level analytical retrieval on primary research data, the semantic table structure has to be defined. To enable the enriched data for statistical analysis, the dimensions have to be classified by a measure type. This approach distinguishes between nominal, ordinal, interval, and ratio scale types, which are common levels of measurement proposed in [Stevens, 1946]:

- **Nominal** basically refers to categorically discrete enumerations or numbers, which define an unordered set structure, such as name of a product, brands or types.
- **Ordinal** defines a totally ordered set (monotonic increasing order) for example a ranking with a 5-star-schema. It allows statistical functions like median or variance.
- **Interval** additionally considers the difference between two values (affine line). For example: The difference between a temperature (Fahrenheit or Celsius scale) of 100 degrees and 90 degrees is the same difference as between 90 degrees and 80 degrees.
- **Ratio** has all properties of an interval variable and a clear and meaningful definition of zero. Temperature on Kelvin scale is ratio, because all thermal motion ceases at absolute zero K.

Obviously, type definitions of dimensions highly influences the (mathematical) characteristics related to the applicable data aggregation and analytics functionalities. An example transformation of table-based data into the RDF Data Cube Vocabulary can be found in Appendix A.

### 3.3 Microformat for Annotation of HTML Tables

Additionally to the final storage data model of data cubes, an annotation model is needed to collect the information during the triplification process. Since these automatically extracted or manually added annotations are directly stored in a HTML table structure of the Data Extractor, the HTML5 custom data-attributes\(^\text{26}\) can be used. Following this mechanism, a CODE specific microformat has been designed. The set of attributes of the microformat is specified in Table 1. It inherits origin, involved people and the creation / interaction method (e.g., altering table cells) to increase trust of users and traceability. Finally, the attributes are defined with the prefix `data-code-`.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Restriction</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td><code>&lt;table&gt;</code></td>
<td>Defines a human readable label for the extracted dataset.</td>
<td>String</td>
</tr>
<tr>
<td>agent (optional)</td>
<td><code>&lt;table&gt;</code></td>
<td>Defines the person that bears responsibility for the data extraction.</td>
<td>URL: support of Mendeley profile links</td>
</tr>
<tr>
<td>url (optional)</td>
<td><code>&lt;table&gt;</code> or <code>&lt;td&gt;</code></td>
<td>Defines the linked data concept for the value of the cell or table semantic.</td>
<td>URL</td>
</tr>
<tr>
<td>component</td>
<td><code>&lt;td&gt;</code></td>
<td>Defines the structure type of this cell. It can be a dimension or a observation values.</td>
<td>URL</td>
</tr>
<tr>
<td>range (optional)</td>
<td><code>&lt;td&gt;</code></td>
<td>Defines the super or general concept of this cell value e.g. 0.3 could be code:typePercentage.</td>
<td>URL</td>
</tr>
<tr>
<td>type (optional)</td>
<td><code>&lt;td&gt;</code></td>
<td>Qualifies and interpret the cell value.</td>
<td>URL</td>
</tr>
</tbody>
</table>

\(^{26}\)http://www.w3.org/TR/2011/WD-html5-20110525/elements.html
### 3.4 Data Extractor

The current prototype of the Data Extractor aims at the creation of data cubes in the scope of the introduced application scenarios. Figure 4 gives an overview of possible input and output data formats.

The following input formats are divided into sources of primary research data (i-iv) and provenance information (v), which are used to create provenance chains [Bayerl, 2013]:

i. **PDF documents** embed aggregated facts like tables, figures or textual patterns. A single document has a low volume but a high integration effort due to its unstructured nature. The main task is to extract meaningful, encapsulated information. Here, the Data Extractor focuses on tables, which carry the actual evaluation results.

ii. **Excel spreadsheets** mostly originate from evaluation data of research campaigns. The data is mostly generated by automated means. Its volume can be very large, but the schema complexity is low, leading to a minor integration effort.

iii. **RESTful API** enables client applications such as TIRA to transmit primary research data directly to the Data Extractor.

iv. **RDF Data** as input means that it is possible to transfer an existent data cube to the Data Extractor and perform refinements on it.

v. **Provenance data** is being collected throughout all processing steps. If a client application uses the API to inject data portions in the Data Extractor, basic information about the user should be passed on as well in order to establish provenance chains.

---

The presented version of the HTML table microformat demonstrates a draft version and will be continuously developed and modified for future requirements. An up-to-date specification is available at the following link:

In addition to the various input formats, the Data Extractor prototype is able to export primary research data in three data formats:

- RDF Data Cubes
- Comma-Separated Value (CSV)
- Attribute-Relation File Format (ARFF)

The variety of export formats shall increase the applicability of the generated data for researchers of the data mining community. Many data analytics tools exist that are capable of processing CSV and especially ARFF files. Currently, the open source project OLAP4LD[^27] is being integrated into the CODE data cube storage layer. After its finalization, it will serve as the fourth export format for the data, and clients like Saiku[^28] can be used for data analysis.

The processing chain of the Data Extractor is divided in the following six central stages as illustrated in Figure 5. In this example, the water levels of the river Danube in Passau will be lifted into a data cube from an Excel file.

**Stage 1:** Authentication is the entry step of the processing stage. A user starts with this stage, whether he wants to process a PDF document or an Excel spreadsheet. Here a user logs into the Data Extractor by the help of the Mendeley OAuth service[^29] and information such as name and URI-based identifier are collected, cp. Figure 6. In the remaining cases, it is assumed that the desired user information is passed into the system via API calls. The user information is used to track provenance information.

[^27]: https://code.google.com/p/olap4ld/
[^28]: http://analytical-labs.com/
[^29]: http://apidocs.mendeley.com/home/authentication
**Stage 2:** The upload stage actually requires two steps: At first, the actual data is transferred to the server, cp. Figure 1. And then the document (PDF, CSV or Excel) is transformed into a pivot table format - here the HTML table definition is in use. For the extraction of meaningful information from PDF documents [Kern, 2013], the extraction service of WP2 is used, whereas the Apache POI framework is utilized to process Excel files.

![Figure 7: Screenshot of the upload phase (Stage 2)](image)

**Stage 3:** Besides the structural annotation of data, stage 3 serves as an entry point for all data, which are loaded through API calls into the Data Extractor. Here the user is able to edit the content of each cell. To finish this step, a specification of the actual dimensions and the measurements that constitute the data cube has to be made. To guide the user while selecting the appropriate data, an automatic (rule-based) table guessing preselects dimensions and measures. A selection result is shown in Figure 8, where blue columns mark dimensions and green columns the measurements. For each column, a header has been selected that is highlighted in a darker colour. Grey cells are not yet selected for further processing. Further, a complete row or column can be deleted. In this sense, the last action performed on the table can be repeated.

![Figure 8: Screenshot of the selection and refinement phase (Stage 3)](image)

**Stage 4:** The last stage in the processing pipeline is the storage of the data cube. Here, a label as well as a description can be set as well as the provenance link for the data if not

---

30 http://poi.apache.org/
yet provided. Furthermore, the disambiguation service of WP2 [Zwicklbauer, 2013] ensures that for each table header an adequate concept URI of the Linked Open Data cloud is selected, cp. Figure 9. This information is needed to establish links and as a foundation for cube merging. The last step in the triplification process is the conversion of the fully annotated HTML tables to RDF-triples. In this process, the recent W3C recommendations RDF Data Cube for statistical data and the PROV Ontology31 for provenance information are used.

![Figure 9: Screenshot of disambiguation of concepts (Stage 4)](image)

The prototype is hosted at the University of Passau and is reachable by the following link:

http://zaire.dimis.fim.uni-passau.de:8080/code-server/demo/dataextraction

**Note:** In the present phase of the CODE project, data cleansing can be applied to a single table cell in stage 3. If a user wants to perform a global data cleansing on a large document, the recommended best practise is to use OpenRefine32 (formally known as Google refine) as pre-processing.

### 3.5 Cube Merging

The main result of task 3.2 is the establishment of efficient data cube merging. Therefore a cube-merging prototype was implemented. The code is released as open source using the MIT license. The implementation can be found in a github.com repository:

https://github.com/bayerls/bacon

A live instance of the prototype is accessible preferable via the profile page of the CODE Data or directly with the following URL:

http://zaire.dimis.fim.uni-passau.de:8080/cube-merging/select

#### 3.5.1 Problem Setting

The Cube Merging prototype operates on the Data Cubes generated by the Data Extractor. It therefore provides an interaction capability with the Data Cubes. The target is to merge multiple cubes in to a single one. The Data Extractor lifts data from various sources into individual cubes and thereby unifies the structure of the data, but the cubes are still separated. The merging prototype now is able to modify this one to one relationship between data source and the corresponding cube. The result of the process is a homogeneous Data Cube, which relates to a set of input sources with

---

31 http://www.w3.org/TR/prov-o/
32 https://github.com/OpenRefine
arbitrary structure. This is an important feature because further interactions with the cubes like visualization or querying are mostly limited to a single cube. Merging is therefore a necessary intermediate step while working with Data Cubes.

This merging process strongly relates to the Data Integration task in the Data Warehousing process. Data-Integration deals among others with Data-Cleansing and Data-Transformation. The corresponding algorithms can be reused in the context of merging Linked Data but the nature of Linked Data causes additional questions. The semantics of the data must be maintained and the underlying data model changes from table-based to graph-based.

To be able to apply the merging process to cubes using the RDF Data Cube Vocabulary, the following essential problems must be tackled:

1. Modify the structure of the input cubes to fit the structure of the resulting cube.
2. Detect and handle duplicates and therefore maintain global key integrity.

Beside the actual data a cube contains a structure definition similar to a database schema. The first problem makes it necessary not only to adapt the structure of the data, but also the dataset structure definition of the cube to. This relates to a schema transformation, which infers rules for suitable data transformation.

Also the duplicate problem must be handled, which is the second major problem. Similar to a relational database, duplicate data rows and duplicate primary keys in particular violate the unique primary key constraint. Therefore, avoiding duplicate keys is necessary to maintain global key integrity. This is crucial for the RDF Data Cube Vocabulary, because the dimensions must uniquely identify the associated observation. This makes it necessary to discarding duplicate observations or to develop an update strategy.

Beside this essential problems additional questions arose, which led to the following requirements:

1. Selected one cube, sort all other cubes suitable for merging, most adequate first.
2. Allow manual interactions in the structural merging process.

Tackling these additional requirements make the prototype more user friendly and useable. Therefore, suitable cubes can be found more easily. Allowing the user to modify the cubes’ structure, helps to cover a larger set of merging possibilities.

The merging prototype approaches all this problem settings. The in depth description of the actual merging process can be found in section 3.5.3.

3.5.2 Related Work

Publishing multidimensional data is the main field of application for the RDF Data Cube Vocabulary. This multidimensional structure is derived from the OLAP cube, known from Data-Warehousing processes. Therefore, the schema of a Data-Warehouse database relates to the structure of the cube vocabulary. Traditional Data-Warehouses offer sophisticated methods to create, maintain and query these databases in an efficient way. These methods are outlined in [Chaudhuri, 1997] and covered in [Lehner, 2003] in more detail. Relevant methods are for example the Data-Cleaning and Data-Transformation techniques, which are part of the Data-Integration process. Data-Cleaning deals with tasks like, migrating, scrubbing and auditing data. These steps prepare the data for the actual integration process. The Data-Transformation process adapts the data to the schema of the database. These methods can be adopted and reused during the process of merging RDF data cubes. There are tools to interact with these RDF data cubes. The CODE project itself provides for example component for querying and visualization. There is very little work tackling the actual problem of cube integration, when the data is already present in this format. The LATC-Project\(^\text{33}\) proposes to

\(^{33}\) [http://latc-project.eu/](http://latc-project.eu/)
create SPARQL queries with multiple FROM clauses to combine the data from various cubes. This approach actually merges the data, but doesn’t tackle any major Data-Integration problem. This technique might be sufficient for the visualization of homogeneous cubes but also isn’t able to produce a valid merged cube like it can be achieved with the cube merging prototype.

3.5.3 Approach
The merging process of two cubes can be subdivided into a set of succeeding steps. Generally speaking, the merging process takes two cubes as input and delivers a single merged cube as output. The following list outlines the exact steps. It also depicts the order in which they are implemented in the merging prototype. Afterwards, every step of this approach will be described in detail.

1. Select the first cube.
2. Suggest suitable cubes for merging.
3. Select second cube.
4. Merge the cubes.
5. Generate and show observation overlap diagram to visualize the key integrity
6. Allow user to adapt structure:
   a. Add component to the structure.
   b. Provide mapping for unmatched components.
7. Remerge the cubes based on the new structure definition (jump back to step 4).
8. Add metadata.
   a. Add label and description to the merged cube.
   b. Add provenance information.
9. Persist the merged cube.

The entry point for the merging process and step one is the selection of the first cube. Therefore, the Data Extractor prototype provides a button with the label “Merge”, which is depicted in Figure 10. The list of available cubes can be found on the profile page of the Data Extractor prototype.

Figure 10: Entry point to the merging prototype

Assuming a first cube was already selected, the prototype is capable of suggesting suitable candidate cubes for merging in the second step. This is implemented via a ranking function, which compares the dataset structure definitions of two cubes. Depending on the overlap degree of dimensions and respectively the measures a score between zero and one is computed. A score of zero denotes no match, whereby one denotes identical dataset structure definitions. All cubes associated with the user are taken into account for this comparison. The result is presented to the user via an ordered list. An example is shown in Figure 11. The sorting enables the user to quickly find the desired merge partners, without scrolling through a long list only containing labels and descriptions.

34 http://eurostat.linked-statistics.org/
Selecting a second cube for merging is simplified by the preceding step. Selecting a cube from the sorted list and confirming the selection via a click accomplishes step three.

The forth step, merging the two cubes is a more difficult task. It primarily falls back to the problem of identifying common components in the dataset structure definition, which are either dimensions or measures. The assumed precondition is that the components are labelled and disambiguated with URIs of the Linked Data Cloud. Only if the same concept is used in both cubes for a component, it is considered as a part of the resulting cube. This is because this fourth step is fully automated and basically finds the set of common components without any user interaction. This merging step focuses on these common components and for now ignores the remaining component. Afterwards three lists are computed. The first one contains the common components from both cubes and is used to visualize the resulting dataset structure definition. The unmatched components from the first cube is added to the second list and the third list respectively holds the unmatched components from the second cube. These lists are reused in the sixth step, to allow further modifications by the user, what will be explained later in more detail. As far the merging concerned only the dataset structure definition of the cubes. Also the actual data – the observations – must be merged. Merging the data after the user has adapted the structure seems to improve the performance but has a relevant drawback. This violation log is used in the fifth step to generate the observation overlap diagram. The graphical component to help maintain the key integrity is shown in Figure 12. The dimension components of the source observations from both cubes are colour-coded as green and respectively blue, whereby the measure components are grey. The red colour indicates that there are observations with the same dimension values in both cubes. The ratio of overlapping observation is denoted by the percentage.

This diagram helps the user to decide, if he should modify the structure of the resulting cube. There are two cases that can occur, which are valid and can lead to correct results:
1. The overlap is not intended: A structural transformation should be applied.
2. The overlap is intended: The merging process will favour observations from the second cube. This results in an update of the observation if the values in the measure components differ in both source cubes.

In the **sixth step** the lists with unmatched components from the forth step are reused. **Step (6a)** enables the user to extend these lists of unmatched components. Both cubes can have an arbitrary number of unmatched components, which of course can be zero. To point out the benefit of this step a small example will be provided in the following. Assuming the preceding “year: 2014” example an observation overlap can be detected in the previous step. The graphical component now shows a block of red observations. The user now knows that this is not intended, because a distinguishing dimension component is missing. The labels of the cubes tell the user that the data is from different cities but the corresponding dimension components are missing. Therefore, the user can now adapt the structure by adding a new dimension component the both cubes providing the disambiguated URI, which identifies the concept city, and the corresponding values for each cube, e.g. Berlin and Passau. Adding this information separately allows the handling of the situation when one cube already provides an appropriate component. Figure 13 shows these newly added unmatched components on the left side.

![Figure 13: Screenshot of the component for manual merging](image)

Matching this two components will resolve the observation overlap, because the key “Berlin – 2014” can be distinguished from the key “Passau – 2014”.

After the user has matched the components, the prototype can now merge the components and add the additional data the resulting cube. The match and merge action is party of **step (6b)**. Before this, the user has to agree on label, resource and range for the new component. When unequal resource are used for the components, like in this case, the user has decide which one will be used in the merged cube. There is a possibility to extend the matching step by giving the user suggestions. This extension will be discussed after the step explanations.

The **seventh step** just repeats step four and therefore it is possible for the user to have multiple interaction with the structure and see the updated lists and observation overlap diagram.

Adding metadata to the cube is the **eighth step. Step (8a)** again requires user input by providing a new label and description for the merged cube. **Step (8b)** is executed fully automatic. The prototype keeps track about the provenance of the merged cube. Therefore the provenance ontology PROV-O is used. The identifying concepts from the source cubes are linked with the `derivedFrom` property. Also the merging process using this prototype and the identifier of the importing user is linked. The PROV-O concepts `Import und Importer` are used for this.
Finally, in the ninth step, the prototype is capable of persisting valid RDF Data Cubes. Therefore, the persisting service of the Data Extractor is reused. The service is called with the merged cube as payload and will afterwards appear on the profile page in the Data Extractor.

As already mentioned, there are matching of components can be extended. In the trivial case the components already use the same disambiguated resource and therefore can be matched automatically. Also the user is enabled to manually provide a mapping and agree on a resulting resource for the matched components. Using knowledge from the Linked Data Cloud can support this decision by inferring the equality of the components. For example, if a chain of sameAs relationships can be found between them, they are semantically equivalent. This feature can be realized using the balloon service.

The disambiguation service developed in the context of the CODE project can be used to suggest appropriate resources. This is beneficial, when the user provides a label for an unmatched component or while matching two components and also hast to specify the corresponding URI.

3.5.4 Changelog

Since the last reported version, the merging prototype has evolved in many ways. Multiple new features were implemented. This section will give an overview over these improvements in Table 2. Therefore, features of the two versions are compared.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Previous version</th>
<th>Current version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select second cube</td>
<td>Select from unordered list. The list contains cubes with identical dataset structure definition, which are therefore suitable for the merging process.</td>
<td>Second cube is suggested depending on the structure of the first cube. Suggestions are sorted in a list with descending structural equality. Only cubes imported or merged by the user are suggested.</td>
</tr>
<tr>
<td>Metadata</td>
<td>Label and description are added.</td>
<td>Label, description and additional information about provenance like the import process and the importer: source cubes are referenced and hence the merged cube can be found on own profile page.</td>
</tr>
<tr>
<td>Merge process</td>
<td>Merge cubes if the dataset structure definitions have equal components: no structural modifications are possible.</td>
<td>Common and respectively merged components are transferred to the resulting merged cube. Every other component is ignored. Therefore, features to adapt the structure are implemented: add dimensions and merge components.</td>
</tr>
<tr>
<td>Key integrity constraint check</td>
<td>If there is an overlap, the second cube updates the first cube.</td>
<td>Integrated into the merging process. The user is informed about the observation overlap with a graphical component. The user can resolve an observation overlap via structural adaptions.</td>
</tr>
</tbody>
</table>

Table 2: Comparison of the merger prototype versions
3.6 RDF Data Cube Implementation Report

The RDF Data Cube Vocabulary is utilized within the whole project to reach a homogeneous data integration of primary research data as well as to generate an OLAP-aware storage. Besides, this standardized data model also fosters the interaction with consuming peers, such as the envisioned visual analytics component.

As a contribution to the on going standardization of the RDF Data Cube Vocabulary, the Data Cube implementation was evaluated and reported to the W3C Government Linked Data Working Group. Therefore the integrity constrains where checked, which are proposed together with the recommendation of the RDF Data Cube Vocabulary.

On the one hand, the implementation was reported to improve the visibility of the CODE project. Every implementer of the data cube vocabulary is publicly listed after providing the implementation report. This list can be found here:

http://www.w3.org/2011/gld/wiki/Data_Cube_Implementations

On the other hand, passing these implementation checks assures that the vocabulary is correctly used to a certain degree. This ensures the validity of the generated cubes and therefore improves the interoperability with other data cube implementations.

The details of the evaluation can be found in Table 3, how they were reported to the W3C Working Group. If an integrity constraint has failed, a short explanation has been added in the description column.

<table>
<thead>
<tr>
<th>Test case</th>
<th>Result</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IC-1. Unique DataSet</td>
<td>Fail</td>
<td>The observation resource uses a control variable. Identity is guaranteed by the named graphs.</td>
</tr>
<tr>
<td>IC-2. Unique DSD</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>IC-3. DSD includes measure</td>
<td>Fail</td>
<td>:Water_level a rdf:Property , qb:MeasureProperty ;</td>
</tr>
<tr>
<td>IC-4. Dimensions have range</td>
<td>Fail</td>
<td>Range not yet defined in the prototype.</td>
</tr>
<tr>
<td>IC-5. Concept dimensions have code lists</td>
<td>Pass</td>
<td>Code lists are not used.</td>
</tr>
<tr>
<td>IC-6. Only attributes may be optional</td>
<td>Pass</td>
<td>qb:AttributeProperty are not yet used.</td>
</tr>
<tr>
<td>IC-7. Slice Keys must be declared</td>
<td>Pass</td>
<td>Slices are not used.</td>
</tr>
<tr>
<td>IC-8. Slice Keys consistent with DSD</td>
<td>Pass</td>
<td>See IC-7</td>
</tr>
<tr>
<td>IC-9. Unique slice structure</td>
<td>Pass</td>
<td>See IC-7</td>
</tr>
<tr>
<td>IC-10. Slice dimensions complete</td>
<td>Pass</td>
<td>See IC-7</td>
</tr>
<tr>
<td>IC-11. All dimensions required</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>IC-12. No duplicate observation</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>IC-13. Required attributes</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>IC-14. All measures present</td>
<td>Pass</td>
<td></td>
</tr>
</tbody>
</table>
Table 3: Details on the integrity constraints test

Some tests passed although the feature is not implemented or the concept is not used. A valid cube must not use every available concept and therefore, this is a valid design decision. For a better understanding, the following list contains extensive explanations for failed tests:

- **IC-1. Unique DataSet:** Concatenating a prefix with an incremented variable generates the concepts for the individual observations. This guarantees uniqueness for the concepts within a cube but doesn’t guarantee unique concepts for independent ones. The values of the incremented variable are not distinct for every generation process. The current implementation uses named graphs to separate the stored cubes. Therefore, the implementation is able to deal with multiple cubes in a proper way doesn’t fulfill this constraint.

- **IC-3. DSD includes measure:** The W3C Proposed Recommendation proposes to run the normalization algorithm before testing the integrity checks. This normalization algorithm is used to generate homogeneous cubes and therefore adds concepts, which are partially optional or can be inferred using a reasoner. These tests were performed without normalizing the cubes. The generated cubes are valid but do not provide this additional information.

- **IC-4. Dimensions have range:** The current implementation doesn’t use the range predicate and therefore this test fails. As an alternative, similar information is stored with the structure definition. The concepts code:cubeDimensionNominal and code:cubeObservationNumber are examples for this situation.
4 Related Work

There have already been efforts to investigate the structure of the Linked Open Data cloud. In early stages of the Linked Open Data project, Rodriguez investigated in [Rodriguez, 2009] the overall structure of the Web of Data and argues that the distributed storage character has to be enlarged, also in the distributed processing of information fostering federation of data management processes. Ding et al. [Ding, 2010] conducted a deeper analysis, which is similar to ideas of CODE Balloon. Here, the idea of a sameAs-graph is given and applied to an example RDF data portion. CODE Balloon enlarges those ideas by considering the complete Linked Data cloud as well as deriving structural statistics.

One of the first results in terms of federated SPARQL query federation is the DARQ framework [Quilitz, 2008]. It follows a mediator-based architecture and applies well-known distributed query processing phases on SPARQL. The FedX framework [Schwarte, 2011] follows the same approach as DARQ, but issues sophisticated optimization techniques especially for the federated domain, as well as on-demand integration of heterogeneous Linked Data sources. Both approaches perform an algebraic optimization as well as operator reordering, to establish an efficient query federation. Here, a user must know how to distribute the queries or at least the endpoints in scope. Recent work is trying to address the federation issue in a similar manner like the CODE project. The SPLENDID approach [Görlitz, 2011] utilizes the openly available voiD data whereas SchemEX [Konrath, 2012] performs an on-the-fly schema integration for indexing retrieved Linked Data portions.

The Linked Open Data community paid a lot attention to the triplification of legacy data, since it was the logical start towards the “Global Data Space” [Heath, 2011]. This corresponds with annual events organized by the community, such as the Triplification challenge of the iSemantics conference. The outcome is a large set of triplification tools, one for almost any domain. Tools exist for accessing relational databases as linked data endpoints (D2RQ35, Virtuoso RDF View36), for frameworks to extract RDF from XML (Krextor37), and for more generic approaches such as Triplify38, which is a small plugin to reveal the semantic structures encoded in relational databases by making database content available as RDF, JSON or Linked Data. Furthermore, the W3C has issued a recommendation for a mapping language from a relational database to RDF [Das, 2012] and compiled a sophisticated overview of RDF conversion tools39.

There are sophisticated algorithms and techniques to analyse and aggregate relational data. Data Warehousing and OLAP [Chaudhuri, 1997] are useable in effective and efficient ways. OLAP operations like roll up, drill down, slice and dice are used to aggregate values, which are organized in a multidimensional data structure, called the OLAP Cube. Otherwise OLAP, on graph-based data and Linked Data in particular, is not yet solved. Linked Data can be represented in XML namely RDF/XML. Therefore, XML to OLAP cube transformations are related to our approach. Such a concept is described in [Jensen, 2001], which proposes a data integrator that can operate on relational data as well as on data stored in XML documents. RDF vocabularies or concepts are not considered in this approach. There already are efforts to apply OLAP operations to Linked Data [Niemi, 2010], [Kämpgen, 2011]. Both approaches convert the Linked Data into a multidimensional dataset. The authors of [Niemi, 2010] therefore define their own data model with dimensions and measures. They also propose a concept to generate the cube using queries [Niemi, 2001]. Kämpgen in [Kämpgen,

35 http://d2rq.org/
36 http://virtuoso.openlinksw.com/
37 http://trac.kwarc.info/krextor/
38 http://triplify.org/
39 http://www.w3.org/wiki/ConverterToRdf
2011] applies the RDF Data Cube Vocabulary to statistical data. Both approaches are similar to the one presented due to the use of the RDF Data Cube vocabulary. In addition, this work also focuses on a crowd-sourced approach and therefore the extension and merging of multiple cubes gains additional attention. Also linking semantic information to the values makes it necessary to extend the RDF Data Cube Vocabulary. This linking allows aggregation based on concepts and hierarchies retrieved from the Linked Open Data Cloud.
5 Conclusion

In this deliverable, the results of WP3 achieved up to the current date have been reported. The next phase of the CODE project foresees the refinement of the prototypes, a high quality evaluation and the integration into the overall platform. Next, the future work of every prototype discussed in this report will be given:

**CODE Query Federation (balloon Fusion)** integrates the main research goal of Task 3.1. The current prototype will be enlarged in two specific ways: first, pruning strategies will include or exclude data endpoints in order to control the expansion of a SPARQL query with respect to the created query statements. Second, optimization techniques shall improve the generated query structure to guarantee efficient quality of service parameter.

**CODE Balloon** will serve as a foundation for a tool suite aiming at offering public services and tools to take advantage of the semantic web with less effort. Building on the indexed data crawled by balloon Overflight, different services like Query Federation, type mining or LOD analysis can be conducted.

**Data Extractor** will be continuously developed in terms of usability and included features. It is also envisioned to expand the currently exposed API to ease the coupling of client applications such as TIRA. Furthermore, browser extensions and CMS widgets will serve as additional data input and export options to enable blogs for dynamic charting functionalities.

**Cube Merging** is currently restricted in terms of integrated features and covered cases of the analysis. Future work will focus on the algorithmic level and the efficient treatment of data cleansing and missing values.
6 References


Appendix A: Example Table Transformation into a Data Cube

Table 4 shows sample data, which is comparable to the PAN challenge 2009. This data will be used, to illustrate a data cube, which could be generated by the Data Extractor.

<table>
<thead>
<tr>
<th></th>
<th>Jane Doe</th>
<th>Richard Roe</th>
<th>John Doe</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plagdet</td>
<td>0.6957</td>
<td>0.6093</td>
<td>0.6041</td>
</tr>
<tr>
<td>Precision</td>
<td>0.7418</td>
<td>0.5573</td>
<td>0.6727</td>
</tr>
<tr>
<td>Recall</td>
<td>0.6585</td>
<td>0.6967</td>
<td>0.6272</td>
</tr>
<tr>
<td>Granularity</td>
<td>1.0038</td>
<td>1.0228</td>
<td>1.106</td>
</tr>
</tbody>
</table>

In the following, some examples are given in Notation 3 (N3)\(^{40}\), to clarify the transformation steps. At first, an empty dataset is generated and enriched with metadata as shown in Listing 5. In this case, the table caption is used to label the dataset.

```n3
code:datasetpan_2009_1 a qb:DataSet;
   rdfs:label "PAN 2009 − sample data";
```

Listing 5: Dataset definition

Next, iterating over every column generates component definitions. For every column that is annotated as a dimension, a new property is generated and typed as a qb:DimensionProperty. The property also extends a predefined property to introduce additional information into the following data-warehouse processes. Other components are processed accordingly. Listing 6 shows this exemplarily. These components are afterwards pooled in the data structure definition as represented in Listing 7.

```n3
code:dimension_139 a rdf:Property;
   a qb:DimensionProperty;
   rdfs:subPropertyOf code: cubeDimensionNominal.

code:dimension_404 a rdf:Property;
   a qb:DimensionProperty;
   rdfs:subPropertyOf code: cubeDimensionNominal.

code:measure_908 a rdf:Property;
   a qb:MeasureProperty;
   rdfs:subPropertyOf code: cubeObservationNumber.
```

Listing 6: Component definition

Blank nodes are used to provide the possibility to add optional information like dimension ordering, but are not used in the current implementation. Alternatively, it is possible to reuse or augment an existing data structure definition. The different types of dimensions and measures, as introduced in Section 3, are used for these definitions. Observations are now generated by importing the data rows of the table. Listing 8 shows a single observation of the sample dataset.

\(^{40}\) http://www.w3.org/TeamSubmission/n3/
The suitable dataset is referenced and each component, defined by the data structure definition, has an assigned value. The declaration of the dimension values differs from the proposed approach of the RDF Data Cube Vocabulary. Instead of directly denoting a single concept or literal, it is important to be able to specify multiple properties. In addition to the raw literal, an optional disambiguated concept is attached at this point. This extra information at the level of a data structure definition is not adequate, because it can differ for every value of this component.