Open City Data Pipeline

Stefan Bischof
Benedikt Kämpgen
Andreas Harth
Axel Polleres
Patrik Schneider

Arbeitspapiere zum Tätigkeitsfeld
Informationsverarbeitung, Informationswirtschaft und Prozessmanagement
Working Papers on Information Systems, Information Business and Operations

Nr./No. 01/2017
ISSN: 2518-6809
URL: http://epub.wu.ac.at/view/p_series/S1/

Herausgeber / Editor:
Department für Informationsverarbeitung und Prozessmanagement
Wirtschaftsuniversität Wien · Welthandelsplatz 1 · 1020 Wien
Department of Information Systems and Operations · Vienna University of Economics and Business · Welthandelsplatz 1 · 1020 Vienna

Open City Data Pipeline

Stefan Bischof[a,*], Benedikt Kämpgen[b], Andreas Harth[c], Axel Polleres[d], Patrik Schneider[a]

[a]*Siemens AG Österreich, Siemensstrasse 90, 1210 Vienna, Austria*
[b]*FZI Research Center for Information Technology, Karlsruhe, Germany*
[c]*Karlsruhe Institute of Technology, Karlsruhe, Germany*
[d]*Vienna University of Economics and Business, Vienna, Austria*

Abstract

Statistical data about cities, regions and at country level is collected for various purposes and from various institutions. Yet, while access to high quality and recent such data is crucial both for decision makers as well as for the public, all to often such collections of data remain isolated and not re-usable, let alone properly integrated. In this paper we present the Open City Data Pipeline, a focused attempt to collect, integrate, and enrich statistical data collected at city level worldwide, and republish this data in a reusable manner as Linked Data. The main feature of the Open City Data Pipeline are: (i) we integrate and cleanse data from several sources in a modular and extensible, always up-to-date fashion; (ii) we use both Machine Learning techniques as well as ontological reasoning over equational background knowledge to enrich the data by imputing missing values, (iii) we assess the estimated accuracy of such imputations per indicator. Additionally, (iv) we make the integrated and enriched data available both in a we browser interface and as machine-readable Linked Data, using standard vocabularies such as QB and PROV, and linking to e.g. DBpedia.

Lastly, in an exhaustive evaluation of our approach, we compare our enrichment and cleansing techniques to a preliminary version of the Open City Data Pipeline presented at ISWC2015: firstly, we demonstrate that the combination of equational knowledge and standard machine learning techniques significantly helps to improve the quality of our missing value imputations; secondly, we arguable show that the more data we integrate, the more reliable our predictions become. Hence, over time, the Open City Data Pipeline shall provide a sustainable effort to serve Linked Data about cities in increasing quality.

Keywords: open data, data cleaning, data integration

1. Introduction

The public sector collects large amounts of statistical data. For example, the United Nations Statistics Division[1] provides regularly updated statistics about the economy, demographics and social indicators, environment and energy, and gender on a global level. The statistical office of the European Commission, Eurostat,[2] provides statistical data mainly about EU member countries. Some of the data in Eurostat has been aggregated from the statistical offices of the member countries of the EU. Even several larger cities provide data in on their own open data portals, e.g., Amsterdam, Berlin, London, or Vienna.[3] Increasingly, such data can be downloaded free of charge and used under liberal licenses.

Such open data can benefit public administrations, citizens and enterprises. The public administration can use the data to support decision making and back policy decisions in a transparent manner. Citizens can be better informed about government decisions, as publicly available data can help to raise awareness and underpin public discussions. Finally, companies can develop new business models and offer tailored solutions to their customers based on open data. As an example for making use of such data, consider Siemens' Green City Index (GCI) [1], which assesses and compares the environmental performance of cities. The CGI is helpful in particular for public awareness, but also demonstrates the potential for investments in more environmentally friendly technologies.

Inspired by the concrete use case of the GGI, our focus in the present paper is on collecting and integrating quantitative indicators about cities, including basic statistics such as demographics but also socio-economic and environmental information.

Even though there are many relevant data sources which publish such quantitative indicators as open data, it is still cumbersome to use data from multiple sources in combination. Obstacles inhibiting comparability and raising the entry barrier for working with open data include the following.

Heterogeneity: different indicator specifications, different languages, formats, and units, as well as All this data is published in different formats such as CSV, JSON, XML, proprietary format such as XLS, just as plain HTML tables, or even worse within PDF files – and so far to a much lesser degree only as RDF or even as Linked Data [2]. Also, the specifications of the individual data fields – (i) how indic-

*Corresponding author
Email addresses: bischof.stefan@siemens.com (Stefan Bischof), kaempgen@fzi.de (Benedikt Kämpgen), harth@kit.edu (Andreas Harth), axel.polleres@wu.ac.at (Axel Polleres), patrik@kr.tuwien.ac.at (Patrik Schneider)

[1] http://unstats.un.org/unsd/
[2] http://ec.europa.eu/eurostat/
[3] http://data.amsterdam.nl/, http://daten.berlin.de/, http://data.london.gov.uk/, and http://data.wien.gv.at/

ators are defined and (ii) how they have been collected – are often implicit in textual descriptions only and have to be processed manually for understanding whether seemingly identical indicators published by different sources are indeed comparable.

Missing values: Data sources like Eurostat Urban Audit cover many cities and indicators. However, for reasons such as cities providing values on a voluntary basis, the published datasets show a large ratio of missing values. The impact of missing values is aggravated when combining different data sets, due to either covering different cities or using different, non-overlapping sets of indicators.

Updates and changes: Studies like the GCI are typically outdated soon after publication since reusing or analysing the evolution of their underlying data is difficult. To improve this situation, we need regularly updated, integrated data stores which provide a consolidated, up-to-date view on data from relevant sources.

We present the Open City Data Pipeline to integrate and enrich statistical data about cities in a uniform, coherent and re-usable manner. In the Open City Data Pipeline, we

1. collect and integrate data from multiple data sources that publish numerical data about cities in a modular and extensible way,
2. enrich the integrated data with missing values,
3. run the process in an automated manner and provide access to the resulting data in a web-based user interface and as Linked Data.

The collection and integration is based on modelling numerical data as multidimensional datasets (data cubes); we use *Statistical Linked Data* [3] as a standardised format to publish both the data and the metadata of cubes in wrappers. Statistical Linked Data means using Linked Data and the RDF Data Cube (QB) vocabulary [4]. We link all cities through their canonical DBpedia identifiers to the LOD cloud.[4] We use a rule-based crawler to access all relevant data [5].

For the enrichment, our assumption – inspired also by works that suspect the existence of quantitative models behind the working, growth, and scaling of cities [6] – is that most indicators in such a scoped domain as cities have their own structure and dependencies, from which we can build statistical prediction models.[5] To this end, we construct a sparse matrix of the available numerical values for different indicators different sources (x-axis) per city-year pair(s) (y-axis). In order to deal with the sparsity of this raw data matrix, we first apply some cleansing steps, and then constructs principal components (PCs) of the raw indicators from a "completed" data sets where missing values for raw indicators are replaced with "neutral" values [7]). We then use these principal components as predictors for different standard regression methods (namely, linear regression, K-nearest neighbour, and random forest) in order to again approximate/impute the missing values for each raw indicator. Here, we chose the best fitting (by evaluating the estimated root mean square error rate (RMSE)) regression method per indicator to build our overall hybrid prediction model. We further improve our predictions by cross-validating and adapting them with respect to existing equational knowledge [8, 9]. That is, apart from learnt models we use ontological background knowledge in the form of equations to improve our predictions.

We automate the tasks so that we can run the collection and integration pipeline once a day. The integrated and enriched dataset is then prepared for direct consumption in a website with JavaScript-based SPARQL queries and published in a reusable manner as Statistical Linked Data.

1.1. What's New?

A first version of the Open City Data Pipeline has been published at ISWC 2015 [10]. For the present paper we completely re-engineered the architecture and improved the enrichment techniques. The architecture is now modular and extensible since we use standard Linked Data principles for interoperability between the various separate pipeline components. The enrichment technique now combines statistical Machine Learning methods for imputation with logic-based equational methods for inference of new values. Moreover, the integrated data not only is enriched with these imputed/inferred values but also with provenance information. In particular, the following additional contributions are presented beyond our earlier, preliminary results:

1. We have refined our wrappers for both Eurostat and UN Data, and now integrate two more years of data; this allows us to compare our prediction results and indeed our earlier conjecture is confirmed that the more data we collect the more accurate models we can train. Moreover, all relevant numeric data (containing city-level indicator measurements) now is published using standard Linked Data principles, so that Linked Data consumption tools such as crawlers can be re-used and regularly run to consider updated data sources.
2. Whereas earlier we have transformed data for integration into RDF using our own, proprietary ontology, the Open City Data Pipeline represents all collected, integrated and predicted indicator values using the standardised RDF Data Cube vocabulary (QB) [4] which has proven to be a suitable format for flexible publication and consumption of numerical data [3, 11, 12] and for generating a unified view, the global cube [9].
3. We present a streamlined architecture, where our current system updates data regularly and automatically based on a rule-based crawler for all relevant data [5] that is easily maintainable and extensible.
4. We have combined two of our previously presented methods for enrichment of numerical data, namely (i) statistical

[4] http://lod-cloud.net/

[5] We refer to "predicting" instead of "imputing" values when we mean finding suitable approximation models to predict indicators values for cities and temporal contexts where they are not (yet) available. These predictions may (not) be confirmed, if additional data becomes available.

Machine Learning methods as presented in [10][6] and (ii) rule-based reasoning over what we earlier called "attribute equations" [8] and "correspondences" [9]; we include a formalisation of this approach that encodes and infers QB observations from such equations, which we call *QB equations* (this combined approach is unique and novel as such).

5. We demonstrate that our extended approach achieves significant improvements in terms of prediction accuracy for missing values:
   - We compare RMSEs with respect to statistical Machine Learning methods applied to principal components presented before [10], arguing that our conjecture that the more data we collect, the better the RMSEs get holds, and that with the same accuracy thresholds we can impute more missing values.
   - Extending the approach, by combining statistical Machine Learning methods with QB equations, we can show further improvements in terms of again reduced RMSEs.

1.2. Paper Structure

The remainder of the paper is organised as follows. Section 2 introduces the necessary preliminaries in terms of Statistical Linked Data and other technical background, such as an overview of the used machine learning methods for missing value imputation. Section 3 gives an overview of the City Data Pipeline architecture, including a description of data sources and a description of how the resulting data set is made available in a re-usable and sustainable manner via a web interface, a Linked Data interface and a public SPARQL endpoint. Section 4 describes the data gathering as well as the main challenges in this context. Section 5 explains the missing data prediction process in more detail. Section 6 refines this process by introducing and applying QB equations. Both the basic value imputation mechanism and the refinement by QB equations are evaluated in Section 7. Section 8 puts our approach in the context of related work. Section 9 gives conclusions, provides lessons learnt and summaries directions for future research.

2. Preliminaries

In the following we introduce the concepts and technologies that form the basis of the work presented in this paper. We start with introducing Statistical Linked Data followed by a description of provenance annotations to track the origin of data. We then briefly survey machine learning methods and introduce the necessary background for equational background knowledge.

[6]Note that, as opposed to [10] herein we rely on learning based on using principal components (PCs) as predictors only. As we have already shown in [10] the naive attempt to learn predictions based on complete subsets of raw indicator data proved to be insufficient for sparse data like the Eurostat/Urban Audit and UN datasets.

2.1. Statistical Linked Data

Statistical Linked Data refers to numerical data, such as about quantitative indicators, properly modelled and published as Linked Data using the RDF Data Cube Vocabulary. In the following, we first introduce the notion of multidimensional data, then Linked Data and finally the RDF Data Cube Vocabulary.

Numerical data (statistical data) is often represented in a multidimensional data model. The multidimensional data model has roots in On-Line Analytical Processing (OLAP) and encompasses datasets (cubes), dimensions, measures and observations. The relevant numbers (or values) in datasets (cubes) are described in relation to independent, mostly categorical attributes (so-called dimensions) and few dependent, mostly numeric attributes (so-called measures) [13]. Data is often represented in terms of different temporal (e.g. annual, monthly, quarterly) and spatial (e.g. city, region, country) dimensions, where across these dimensions statistical observations for different properties (typically, numerical indicators) are stored and aggregated. Stated in the *metadata* of a cube, every cube not only has a predefined set of dimensions and measures but also every dimension has a predefined set of possible dimension values, also called *members*).

All observations in a dataset (cube) can be represented in tabular form. As example, consider see Table 1 illustrating a dataset about "population" from Eurostat.[7] The dataset has the following dimensions: geography (on the granularity of cities), time (on the granularity of years) and indicator; the measure refers to the value, the measured indicator. Members for the city dimension would be Karlsruhe and Vienna, for the year dimension 2009 and 2014, and for the indicator dimension Population. Finally, there is the numerical value itself: the dependent measure value (the observation) is the actual population.

Table 1: Example dataset/cube (Eurostat's urb_cpop1 dataset) with population values.

City	Year	Indicator	Value
Karlsruhe	2012	Population	291 995
Karlsruhe	2013	Population	296 033
Karlsruhe	2014	Population	299 103
Vienna	2009	Population	1 687 271
Vienna	2013	Population	1 741 246
...	...	...	...

Linked Data refers to data published according to the Linked Data principles [14], a set of best practices widely-adopted within the Semantic Web community. Slightly adapted to our concrete use case of publishing statistical data, we can recap these principles as follows:

1. URIs are used to identify things such as datasets, dimensions, statistical indicators, cities and countries.
2. HTTP URIs are used to allow for looking up descriptions of things.

[7]http://appsso.eurostat.ec.europa.eu/nui/show.do?dataset=urb_cpop1&lang=en

3. HTTP URIs are resolvable and provide useful, machine-readable information in RDF, using well-known vocabularies.
4. Useful information links to other things and dereferenceable datasets by reusing HTTP URIs such as URIs used by others for things such as dimensions, indicators, cities and countries.

The RDF Data Cube Vocabulary (QB) [4] is a widely-used source of URIs to describe numeric data using a multidimensional data model. QB, as a W3C recommendation has established itself as the standard for aggregating and (re-)publishing statistical observations on the web, with off-the-shelf tools to process and visualise QB data, which is why we chose to deploy it also for our work. Figure 1 provides an overview of QB with the most important classes and properties. For readability reasons, we describe URIs with namespaces,[8] slightly abusing the W3C CURIE syntax for expressing compact URIs.

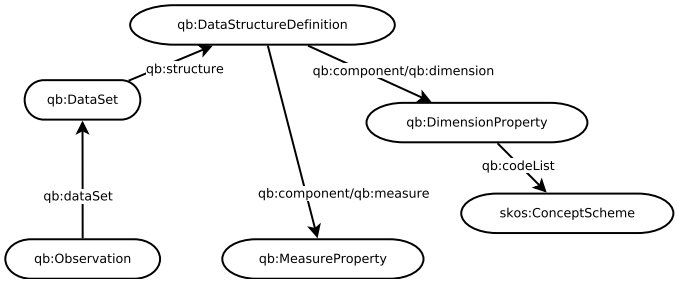

Figure 1: Illustration of most important classes of the RDF Data Cube Vocabulary with properties (or property chains) between instances of concepts; adapted from "Outline of the vocabulary" in the QB specification.

QB allows to describe datasets/cubes (instances of `qb:DataSet`) with observations (instances of `qb:Observation`). In the remainder of the paper we use the terms (statistical) dataset, QB dataset and cube synonymously. Every dataset has a certain structure (instance of `qb:DataStructureDefinition`, short *DSD*) that – using a chain of properties `qb:component` before `qb:measure` or `qb:dimension` – defines measures (instances of `qb:MeasureProperty`) and dimensions (`qb:DimensionProperty`). Attributes (`qb:AttributeProperty`) allow to – per default optionally – add information to observations that help to qualify and interpret the observed value(s). A `qb:DataSet` provides all necessary information about a cube. The `qb:DataSet` URI gives the name of the relation in the tabular representation defined by the cube (see Table 1 for an example). The `qb:DataStructureDefinition` – the metadata of the cube/dataset – defines the independent and dependent attributes of the relation as well as their possible attribute values.

The `qb:Observation` instances describe the entities in the relation.

The following triples describe an example observation as a *blank node*, an instance with an only locally known name, of 1741246 inhabitants of Vienna in 2013 in the population dataset of Eurostat[9]:

```
_:obs1 a qb:Observation ;
  qb:dataSet eurostat:id/urb_cpop1#ds ;
  estatwrap:cities eurostat-cities:Vienna ;
  estatwrap:indic_ur eurostat-indic_ur:Population ;
  dcterms:date "2013" ;
  sdmx-measure:obsValue "1741246" .
```

A HTTP GET request on `eurostat:id/urb_cpop1#ds` returns the RDF describing the `qb:DataSet` instance. The following SPARQL query returns the number of observations in the population dataset[10]:

```
SELECT count(?obs)
WHERE {
    ?obs qb:dataSet eurostat-pjan:ds.
}
```

The QB specification defines the notion of "well-formed qubes".[11] QB further specifies SPARQL ASK queries as *QB integrity constraints* that when applied to an RDF graph return `true` if the graph contains one or more data cubes that are not *well-formed* according to the specification. For instance, a well-formed cube satisfies the following constraints. Every observation in the dataset has a value for each of the measures and dimensions. The values of the measures are functionally dependent on the values of the dimensions and for every possible combination of dimension values, only one fact can be contained in the dataset. Dimension values only can come from a specific list as specified in the "metadata"/data structure definition (e.g., instances of `skos:Concept` in a `skos:ConceptScheme` linked from the dimension via `qb:codeList`).

When generating and publishing QB datasets, we ensure that these constraints are fulfilled. For instance, when we later generate new observations via predictions and computations we also generate new datasets containing these values. Also, when integrating observations from several datasets into a unified view (the global cube) we will ensure integrity by introducing a new dimension.

When using QB, we can rely on an existing toolchain: similarly as multidimensional datasets in OLAP systems, datasets represented in QB can be queried using common OLAP operations using SPARQL [3, 9, 15]:

- Projection: To filter for certain measures from a dataset.
- Dice: To filter for certain dimension values from a dataset, e.g., only values from "2010".
- Slice: To aggregate over one dimension from a dataset, e.g., to not consider different genders.
- Roll-Up: To aggregate one dimension to a higher level of abstraction, from cities to countries.
- Drill-Across: To integrate two datasets into one dataset to combine their values. Different from the other operations, drill-across has as input not one but two datasets.

[8] Use `http://prefix.cc/` to look up prefix definition.

[9] If not stated otherwise, we use (abbreviated) Turtle or N3 notation.

[10] We assume that the population data is accessible in the default graph of the SPARQL processor.

[11] `https://www.w3.org/TR/vocab-data-cube/#wf`

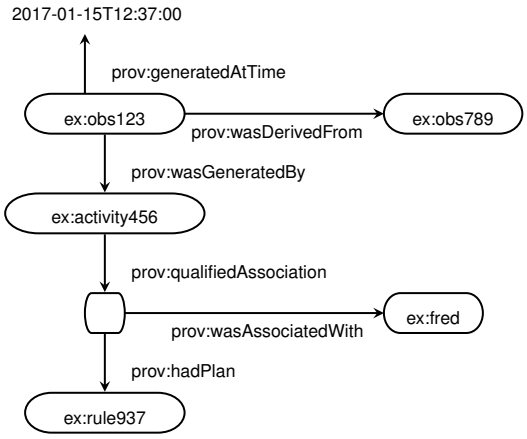

Figure 2: PROV example.

2.2. Provenance Annotations

Apart from storing observations alone, in order to make data traceable and allow users to judge the trustworthiness of data, it is import to record the provenance of data points. There exist several approaches to address this issue for RDF. A lightweight approach is to use different Dublin Core properties to refer from a dataset to its publisher and attach some metadata. For example the property `dc:publisher` is defined to refer to an entity (like a person or organisation) which publishes some resource.

A more flexible approach is PROV [16], which provides an ontology (among other documents) to annotate all kinds of resources with provenance information and allows tracking of provenance of resource representations. On a high level PROV distinguishes between entities, agents, and activities. A `prov:Entity` can be all kinds of things, digital or not, which are created or modified. *Activities* are the processes which create or modify entities. An `prov:Agent` is something or someone who is responsible for an activity (and indirectly also for an entity). A `prov:Activity` is something that happens and, in our case, generates new observations from other observations. PROV also defines *plans* which can be understood as any kind of predefined workflow which a `prov:Activity` might follow to create or modify an entity. Additionally PROV also allows to tag certain activities with *time*, for example a timestamp when an entity was created.

We will rely on PROV annotations to store the data source of observations or, when imputing missing values, to annotate the method that was used to process and generate the enriched observations. Figure 2 shows a PROV example (all other triples removed) of two observations, where a QB observation `ex:obs123` was derived from another observation `ex:obs789` via an activity `ex:activity456` on the 15th of January 2017 at 12:37. This derivation was executed according to the rule `ex:rule937` with an agent `ex:timbl` being responsible. This use of the PROV vocabulary models tracking of source observations, a timestamp, the conversion rule and the responsible agent (which could be a person or software component).

UNSD Dataset

	Population	Hotel Beds
Berlin 2010	3 000 000	
Prag 2010		70 000
Vienna 2010	1 500 000	100 000

Eurostat Dataset

	CO2 Em.	Living Area
Bern 2010	0.34	36
London 2010		30
Paris 2010	0.5	

Combined Dataset

	Population	Hotel Beds	CO2 Em.	Living Area
Berlin 2010	3 000 000			
Prag 2010		70 000		
Vienna 2010	1 500 000	100 000		
Bern 2010			0.34	36
London 2010				30
Paris 2010			0.5	

Figure 3: Combining different Datasets

2.3. Missing Values

After integrating the different datasets, we discovered a large number of missing values in our data sets. We identified two reasons for that:

- As shown in Table 2 and 3, we can observe a large ratio of missing values due to incomplete data published by the data providers;
- More severely, when we combine the different datasets even more missing values are introduced, since there is a fair amount of disjoint cities and indicators between the datasets (see Figure 3).

2.4. Machine Learning Methods

In our attempt to impute missing values for certain indicators and cities, our assumption is that every such indicator has its own distribution (e.g., normal, Poisson) and relationship to other indicators. Hence, we aim to evaluate different regression methods and choose the best fitting model to predict the missing values. We measure the prediction accuracy by comparing the *normalised root mean squared error* in % (RMSE%) [17] of every regression method:

$$RMSE\% = \left(\frac{\sqrt{\frac{\sum_{t=1}^{n}(y_t - y'_t)^2}{n}}}{y_{max} - y_{min}} \right) \times 100$$

where n is the amount of predictions, y_t is the observed (actual) value on t, y'_t is the predicted value on t, and y_{max} (resp. y_{min}) the maximum (resp. minimum) value of the observed values. Further, we use for measuring the quality of prediction calculated by QB equations the *root mean squared error* (RMSE), which is defined as:

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n}(y_t - y'_t)^2}{n}}$$

where n is the amount of predictions, y_t is the observed (actual) value on t, y'_t is the predicted value on t.

In the field of Data Mining [17, 18] (DM) various regression methods for prediction were developed. We focus on "standard" DM methods, since there methos are straightforward to apply and show a robust behaviour. For instance, we use the following methods:

- *K-Nearest-Neighbour Regression* (KNN) that is a widespread DM technique based on using a distance function to partition the instance space.
- *Multiple Linear Regression* (MLR) that has the goal to find a linear relationship between a target and several predictor variables.
- *Random Forest Decision Trees* (RFD) involves the top-down segmentation of the data into multiple smaller regions represented by a tree with decision and leaf nodes.

All these methods, have a common caveat when applied to our raw data: they need complete training data, or respectively, can only be applied to complete subsets of the data. As we will see in Section 4 below though, our data sources contain partially very sparse data, such that training regression methods based on complete subsets only, is often insufficient. To this end, a common and robust method is to first perform a Principal Component Analysis (PCA) to reduce the number of dimensions of the data set and use the new compressed dimensions, called *principal components* (PCs) as predictors for the above-mentioned standard regression methods. As stated in [18], the PCA is a common technique for finding patterns in data of high dimensions, but it can also be used for sparse data. That is, in a first step all the missing values are *imputed* with *neutral* values for the PCA, where the neutral values are created according to the *regularised iterative PCA algorithm* described in [7]. The PCs generated this way shall reflect the characteristics of the whole data set and can be used as predictors.

2.5. Equational Background Knowledge and Inference

Apart from the above-mentioned regression methods, the common assumption in the Semantic Web is that missing information can be *inferred* deductively by applying ontological reasoning over suitably formalised background knowledge.

In most cases, such ontological background knowledge is formalised in terms of taxonomic axioms about class and property hierarchies, in terms of RDFS and OWL ontologies, parts of which (e.g. reasoning about subproperties and subclasses, or entity consolidation using `owl:sameAs` inferences) are well-known to be covered by rule-based inferences. We refer to the respective standard or textbook articles for the necessary background, e.g. cf. [19].

Another kind of often neglected knowledge is equational knowledge in the form of equations defining functional dependencies among certain attributes of a resource. For example, if we know that city1 has 10000 inhabitants (*population*) on an area of 10 square kilometers (*area_km*2), we can derive the population density from the equation

$$populationDensity = population/area\_km2$$

Note, for simplicity reasons, in this example, we do not use QB modelling. The multidimensional data model of QB allows to make explicit different dimension – dimension value combinations, e.g., `_:obs cd:unit "km2".` and `_:obs dcterms:date "2010".` which is important for interpreting the semantics of values and for integration purposes [20].

In [8] we have defined an approach that allows to store and process such so called attribute equations in RDF in order to enable ontological derivations, i.e., –simplifying– given triples

```
:city1 :population 10000 ; :area_km2 10 .
```

the approach can infer

```
:city1 :populationDensity 1000 .
```

Equations are in principle not directed, thus attribute equations would also infer the population, given the area and the population density.

Similar to OWL and RDFS-based reasoning, the approach works on either rule-based forward-chaining inference, or, respectively in terms of query rewriting, where equations such as the one above are interpreted – roughly– as rules of the form

```
{
   ?c :population ?p .
   ?c :area_km ?a .
   ?pd = ?p / ?a .
} =>
{  ?c :populationDensity ?pd }.
```

For further details about the equations supportable in such a rule-based approach we refer to [8]. At this point, let us just emphasise that, in case no computation up to a fixpoint is needed, rule-based inference as the one sketched above can be also realised with simple SPARQL CONSTRUCT or INSERT queries (or, in off-the-shelf SPARQL engines by iteratively applying such queries), such as:

```
INSERT {  ?c :populationDensity ?pd  }
WHERE {
  ?c :population ?p .
  ?c :area_km ?a .
  BIND(?p / ?a AS ?pd)
}
```

As again we will see in Section 4 below, various published statistical indicators for cities (population density being one simple example) are indeed computed indicators, with published equational computation rules available along with the datasets. Herein, we aim to exploit such equations and directly infer new QB observations. To this end, we will extend and adapt the serialisation of equations and inference rules derived from equations from [8] to comply with the QB vocabulary, cf. Section 6.

3. Overview and System Architecture

The OCDP workflow is illustrated in Figure 4 and consists of several steps. Data is provided as Statistical Linked Data via wrappers. A crawler component collects data from different sources and stores it into the triple store. The data is stored in a SPARQL endpoint. Then, the data is integrated into the global cube. After missing values are imputed with statistical Machine Learning methods, QB equations are applied to infer additional values. Finally, the resulting data is made accessible.

The architecture of the OCDP system consist of several components. Figure 5 gives a high level overview of the architecture with a triple store being the central part. The data quality improvement workflow uses various methods to improve data quality and enrich the data.

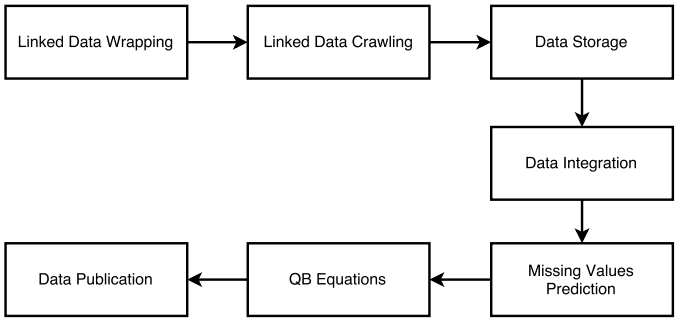

Figure 4: Open City Data Pipeline workflow

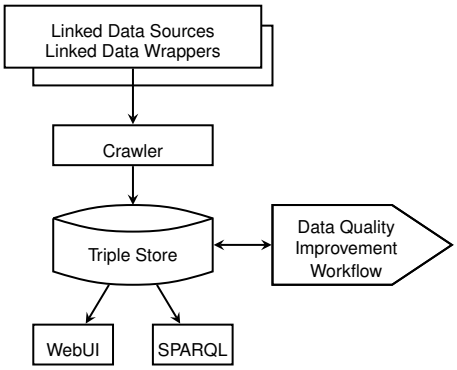

Figure 5: Open City Data Pipeline architecture

We start with surveying data sources that serve as input to the pipeline in Section 3.1. We introduce the different components, their inputs, outputs, and interfaces in Section 3.2 and explain how we make the resulting data available in Section 3.3.

3.1. Data Sources

Many interesting statistical data sources are nowadays available. Many indicators in these data sources are provided on a country level and only a subset of indicators are available on the city level. We have identified the following providers of statistical data concerning cities:

- DBpedia[12];
- Eurostat with Urban Audit;
- United Nations Statistics Division statistics;
- U.S. Census Bureau statistics;
- Carbon Disclosure Project[13];
- individual city data portals.

In particular, we use DBpedia, United Nations statistics, and Eurostat data sources, which are integrated and enriched by the OCDP. The data sources contain data ranging from the years 1990 to 2016, but most of the data concerns the years after 2000. Further, not every indicator is covered over all years, where the highest coverage of indicators is between 2004 and 2015 (see Tables 2 and 3). Most European cities are contained in the Eurostat datasets, but we also include the capital cities and cities with a population over 100 000 from the United Nations Demographic Yearbook (UNDY).[14]

The previous OCDP of ISWC 2015 [10] contains data from 1990 to 2013 with 638 934 values from the Eurostat data source and 69 772 values from the U.N. data source. Due to some reorganisation in the Eurostat and U.N. datasets, Eurostat contains now 506 854 values and the U.N. provides 40 532 values. Regarding indicators, we now have 209 instead of 215 Eurostat and 64 instead of 154 U.N. indicators. The reason for the drop in indicators is due to the fact that the U.N publishes fewer datasets. The same effect can be seen for the cities, where we have 966 instead of 943 Eurostat and 3 381 instead of 4 319 U.N cities. Due to the smaller size of the datasets (see Tables 2 and 3), we now have an improved missing values ratio of 81.7% (before 86.3%) for Eurostat, resp. 94.4% (before 99.5%) for the U.N. dataset.

We now describe each of the data sources in detail.

DBpedia. DBpedia, initially released in 2007, is an effort to extract structured data from Wikipedia and publish the data as Linked Data [21]. We include the URIs, latitude/longitude, and labels of a city in our dataset. For cities, DBpedia provides various basic indicators such as demographic and geographic information (e.g., population, latitude/longitude, elevation) but without a year. While we only integrated textual data, we plan to add other indicators like weather data and the population of a city in the future using the DBpedia Wayback Machine [22] to obtain the values of different points in time.

Eurostat. Eurostat[15] offers various datasets concerning E.U. statistics. The data collection is conducted by the national statistical institutes and Eurostat itself. In particular interesting is the Urban Audit (UA) collection, which started as an initiative to assess the quality of life in European cities. UA aims to provide an extensive look at the cities under investigation, since it is a policy tool to the European Commission: "The projects' ultimate goal is to contribute towards the improvement of the quality of urban life" [23]. Currently, data collection takes place every three years (last survey in 2015) and is published via Eurostat Urban Audit. All data is provided on a voluntary basis which leads to varying data availability and missing values in the collected datasets. At the city level, Urban Audit contains over 200 indicators divided into the categories Demography, Social Aspects, Economic Aspects, and Civic Involvement. Currently, we extract the datasets that include the following topics:

- Population by structure, age groups, sex, citizenship, and country of birth
- Fertility and mortality

[12] http://wiki.dbpedia.org/
[13] https://www.cdp.net
[14] http://unstats.un.org/unsd/demographic/products/dyb/dyb2012.htm
[15] http://ec.europa.eu/eurostat

Table 2: Values of the Eurostat Dataset

Year(s)	Cities	Indicators	Available	Missing	Missing Ratio (%)
1990	131	88	1 799	9 641	84.27
2000	433	163	6 420	63 996	90.88
2005	598	168	20 460	79 836	79.60
2010	869	193	56 528	110 996	66.26
2015	310	69	2 030	19 291	90.48
2004–2016	879	207	437 565	1 331 250	75.26
All (1990–2016)	966	209	506 854	2 257 171	81.66

Table 3: Values of the United Nations Dataset

Year(s)	Cities	Indicators	Available	Missing	Missing Ratio (%)
1990	5	3	8	7	46.67
2000	1 078	61	3 861	61 836	94.12
2005	777	61	2 110	45 226	95.54
2010	1 525	64	5 866	91 670	93.99
2015	216	3	568	77	11.94
2004–2016	2 095	64	28 849	511 759	94.66
All (1990–2016)	3 381	64	40 532	685 548	94.42

- Living conditions and education
- Culture and tourism
- Labour market, economy, and finance
- Transport, environment, and crime.

United Nations Statistics Division (UNSD). The UNSD offers data on a wide range of topics such as education, environment, health, technology and tourism. The focus of the UNSD is usually on the country level, but there are some datasets on cities available as well. Our main source is the UNSD Demographic and Social Statistics, which is based on the data collected annually (since 1948) by questionnaires to national statistical offices.[16] Currently we use the datasets on the city level that include the following topics:

- Population by age distribution, sex, and housing
- Households by different criteria (e.g., type of housing)
- Occupants of housing units / dwellings by broad types (e.g., size, lighting, etc.)
- Occupied housing units by different criteria (e.g., walls, waste, etc.)

The full UNSD Demographic and Social Statistics data has over 650 indicators, wherein we kept a set of 64 course-grained indicators and drop the most fine-grained indicator level. For example, we keep *housing units total* but drop *housing units 1 room*. We prefer more coarse-grained indicators to avoid large groups of similar indicators which are highly correlated. However, for future work, we plan to introduce the fine-grained indicators and relate them to their coarse-grained parents using hierarchies on the indicator dimension.

[16] http://unstats.un.org/unsd/demographic/

Prospective Data Sources. At the point of writing, the data sources are strongly focused on European cities and demographic data. Hence, we aim to integrate further national and international data sources, in particular the U.S. Census Bureau statistics and the Carbon Disclosure Project.

U.S. Census Bureau. The U.S. Census Bureau [24] offers two groups of tabular datasets concerning U.S. statistics: *Table C-1 to C-6* of [24] cover the topics Area and Population, Crime and Civilian Labor Force for cities larger than 20 000 inhabitants; *Table D-1 to D-6* of [24] cover Population, Education, Income and Poverty for locations with 100 000 inhabitants and more.

Contrary to the UNSD or Eurostat datasets, the USCCDB has a low ratio of missing values ranging from 0% to 5% for a total of 1267 cities. The data includes 21 indicators, e.g., population, crime, and unemployment rate.

Carbon Disclosure Project (CDP). The Carbon Disclosure Project (CDP) is an organisation based in the U.K. aiming at "[...] using the power of measurement and information disclosure to improve the management of environmental risk".[17] The *CDP cities* project has data collected on more than 200 cities worldwide. CDP cities offers a reporting platform for city governments using an online questionnaire covering climate-related areas like Emissions, Governance, Climate risks, Opportunities, and Strategies.

Individual city open data portals. Many cities operate dedicated open data portals. The data from these individual city open data portals (e.g., New York, Vienna) could be added and integrated. This is surely a large effort on its own, as we would require a unified interface to many different data portals. Either we would have to write wrappers for every cities' portal, or standardisation efforts on how cities publish data would have to succeed.

3.2. Pipeline Components

We now give an overview of each of the components of the OCDP system.

Statistical linked data wrappers. None of the mentioned data sources publishes statistical data as statistical linked data. Thus we use a set of statistical linked data wrappers which publish the data from these sources according to the principles listed in Section 2. Such a wrapper consumes data from the original source, either in real-time or in batch mode, from the original format, e.g., CSV, and converts the data to statistical linked data and eventually provides it to the consumer. Section 4 also explains the statistical linked data wrappers.

Linked data crawler. To collect the data from all the different sources in one place the linked data crawler starts with a seed list of URIs and crawls relevant connected linked data. The resulting RDF data is collected in one big RDF file and eventually loaded into the triple store. Section 4 extensively explains the linked data crawler in detail.

[17] https://www.cdp.net/en-US/Pages/About-Us.aspx

Triple store. We use a standard Virtuoso 7 triple store as a central component to store data at different processing stages. For data loading we use the Virtuoso SQL console which allows faster data loading. For all other data access we rely on Virtuosos SPARQL 1.1 interface which allows not only to query for data but with SPARQL Update also to insert new triples.

Data quality improvement workflow. In an iterative approach we improve data quality of the crawled raw data. In this configurable workflow we use several different components consecutively. Each workflow component first reads input data (=observations) from the triple store via SPARQL queries, processes the data accordingly and inserts new triples into the triple store either via SPARQL Insert queries or the Virtuoso bulk loader facility (the first option is more flexible – it allows the execution of the workflow on a different machine – the second usually allows faster data loading).

The workflow currently uses three different components: a component to materialise the global cube, a statistical component, and a rule execution component.

The first component materialises the global cube in a separate named graph. This materialisation effectively resolves different types of heterogeneity found in the raw data: (i) different URIs for members, (ii) different URIs for dimensions, (iii) different DSDs (although the DSDs must be compatible to some extent for the integration to make sense). Eventually the global cube provides a unified view over many datasets from several sources. This component is implemented with SPARQL Update queries and supplied background knowledge for the integration. Section 4 details this process of linking statistical data and the materialisation.

The second component for missing value prediction extracts the whole global cube generated by the materialisation as one big dataset. Then it uses different machine learning methods to train models for missing value prediction. This component is implemented as a set of R scripts which extract the data with SPARQL queries. We then train and evaluate the models for each of the indicators. If the selected model delivers predictions in a satisfactory quality we apply the model and get estimates for the indicators. Finally the component exports the statistical data together with error estimates to one RDF file which is then loaded into the triple store with the Virtuoso bulk load feature and added to the global cube. Section 5 explains the details of this components in detail.

The third and last component uses equations from different sources to infer even more data. To this end we introduce QB equations. These QB equations provide an RDF representation format for equational knowledge and a semantics as well as a forward chaining implementation to infer new values. QB equations are implemented in a naive rule engine which directly executes SPARQL Insert queries on the triple store. Section 6 introduces the concept of QB equations with syntax, semantics and implementation.

3.3. Data Publication

Eventually after the data is crawled and loaded into the triple store, improved and enriched by our workflow, the resulting global cube is available for consumption.

We provide a SPARQL endpoint[18] based on Virtuoso, where the global cube is stored in a named graph.[19] The prefix names used in the examples above are already set in Virtuoso, thus no prefix declarations are necessary for SPARQL queries.

We also provide a simple user interface[20] to query values for a selected indicator and city in the global cube. Queries are directly executed on the triple store during loading of the website using a JavaScript library called Spark; thus one can have a look at the SPARQL queries in the source code. We show all predicted values for transparency reasons. We simply order by the error value, i.e., the most trustworthy value per year is always shown first.

4. Data Collection, Cleaning, and Integration

We now explain our data collection, cleaning and integration approach. The approach is modular and extensible in the sense that every new data source can be prepared for consideration separately and independently from other sources in a well-defined manner. The approach allows always up-to-date data since it can be re-run at any time and is limited only by the available amount of storage space and runtime to download from the web as well as store and query all relevant data.

The approach consists of the following components:

- Linked Data wrappers that re-publish numerical data from various data sources as Statistical Linked Data in a well-defined manner (Section 4.1);
- a rule-based Linked Data crawler which is maintainable and efficient in collecting all relevant data (Section 4.2);
- semi-automatically generated links between statistical data from different sources (Section 4.3);
- the definition and materialisation of a unified view over all relevant statistical data, a so-called global cube which includes provenance information (Section 4.4).

4.1. Linked Data Wrappers

Every statistical data source that could be analysed in the OCDP is assumed to be published either as Linked Data (e.g., DBpedia), Statistical Linked Data (e.g., Eurostat), or as collection of datasets in tabular form (e.g., UNSD or U.S. Census). Wrappers are a flexible way to make these data sources available such that (a) *on-the-fly* retrieval from the original source, (b) transformation to RDF, and (c) re-publication as Statistical Linked Data is feasible.

Every data source has an individual dataset, which has their own specialities and structure. Hence the wrapper needs to publish all the necessary information as resources accessible via resolvable URIs. The table of content provides all available datasets as a list of `qb:DataSet` triples and a data structure definition

[18] http://citydata.wu.ac.at/ocdp/sparql
[19] http://citydata.wu.ac.at/qb-materialised-global-cube
[20] http://kalmar32.fzi.de/indicator-city-query.php

(as qb:DataStructureDefinition), which includes the available dimensions (as qb:dimension) and concept schemes (as skos:ConceptScheme). The statistical data is published itself as individual datasets that include all available observations (as qb:Observation).

We use the following wrappers that provide access to the underlying data source via a Linked Data interface:

- *Eurostat Wrapper:* The Eurostat datasets as described in Section 3 are available as Linked Data via the Eurostat Linked Data Wrapper (Estatwrap).[21] The Estatwrap provides a table of contents[22] from which all relevant datasets can be selected. For instance, Estatwrap offers under the eurostat namespace the Eurostat GDP Growth Dataset (eurostat:id/tsieb020#ds).
- *UNSD Wrapper:* Another relevant data source are the UN Data datasets that are available as Statistical Linked Data from the UN Data - Linked Data Wrapper (UN-wrap).[23] For instance, UN-wrap offers under the undata namespace the dataset *Occupied housing units by type of housing unit for selected cities*.

Before integration, locations have varying names in different data sources (e.g., Wien vs. Vienna), a Uniform Resource Identifier (URI) for every city is essential for the integration and enables to link the cities back to DBpedia and other LOD datasets. We choose to have a one-to-one (functional) mapping of every city from our namespace to the English DBpedia resource, which in our republished data is encoded by owl:sameAs relations. We identify the matching DBpedia URIs for multilingual city names and apply basic *entity recognition*, similar to Paulheim et al. [25], with three steps using the city's names from Eurostat and UNYB:

- Accessing the DBpedia resource directly and following possible redirects;
- Using the Geonames API [24] to identify the resource;
- For the remaining cities, we manually looked up the URL on DBpedia.

4.2. Linked Data Crawler

Based on the basic notion of Linked Data, RDF may be stored in a distributed manner. All necessary information about a dataset can be found by resolving URIs of entities related to the dataset. Related entities are all instances of QB-defined concepts that can be reached from the dataset URI via QB-defined properties. For instance, from the URI of a qb:DataSet instance, the instance of qb:DataStructureDefinition can be reached via qb:structure. Similarly, instances of qb:ComponentProperty (dimensions/measures) and skos:Concept (members) can be reached via links.

Once all numeric data is available as Linked Data, we need to make sure to collect all relevant data and metadata starting from a list of initial URIs. First, a seed list of URIs needs to be generated to start the collection from. One example of a "registry" or "seed list" of dataset URIs is provided by the PlanetData wiki.[25] A seed list of such datasets is published as RDF and considered as input to the crawling.[26]

Then, Linked Data crawlers deploy crawling strategies for RDF data where they resolve the URIs in the seed list to collect further RDF and in turn resolve a specific (sub-)set of contained URIs. An example Linked Data crawler is *LDSpider*[26], that uses a depth-first or breadth-first crawling strategy for RDF data. Linked Data crawlers typically follow links without considering the type.

A more direct approach of loading relevant data, a *directed crawling strategy*, starts with resolving and loading the URIs of qb:DataSets interesting to the user, then in turn resolves and loads instances of QB concepts in the order they can be reached from the dataset URI.

To describe how to collect Linked Data, we use the Linked Data-Fu language [5] in which rule-based link traversal can be specified. For instance, to retrieve data from all qb:DataSets, we define the following rule:

```
{
  ?ds rdf:type qb:DataSet.
} =>
{
  [] http:mthd httpm:GET .
     http:requestURI ?ds .
} .
```

The head of a rule corresponds to an update function of an internal graph representation in that it describes an HTTP method that is to be applied to a resource. In our example, the head of a rule applies a HTTP GET method to the resource ?ds. The body of a rule corresponds to the pre-condition in terms of triple patterns that have to hold in the internal graph representation. In our example, ?ds is defined as an instance of qb:DataSet.

Similarly, we retrieve instances of qb:DataStructureDefinition, qb:ComponentSpecification, qb:DimensionProperty, qb:AttributeProperty, qb:MeasureProperty, qb:Slice, qb:SliceKey, and qb:ObservationGroup. Also, we crawl the list of possible dimension values (based on qb:codeList) as well as each single dimension value. The only instances we do not resolve are observations since these are usually either modelled as anonymous blank nodes without own identifier or provided together with other relevant information with the instance of qb:DataSet or qb:Slice.

Crawling may include further information, e.g., rdfs:seeAlso links from relevant entities and information encoded in the Vocabulary of Interlinked Datasets (VoiD).[27] For instance, VoiD descriptions may state that the relevant data can be retrieved from a certain SPARQL endpoint. Assuming that the number of related instances of QB concepts starting from a QB dataset

[21] http://estatwrap.ontologycentral.com/
[22] http://estatwrap.ontologycentral.com/table_of_contents.html (also available in RDF)
[23] http://citydata.wu.ac.at/Linked-UNData/
[24] http://api.geonames.org/
[25] http://wiki.planet-data.eu/web/Datasets
[26] http://kalmar32.fzi.de/triples/ocdp_seed_datasets.nt
[27] http://rdfs.org/ns/void#

is limited and that links such as rdfs:seeAlso for further information are not crawled without restriction (e.g., only from instances of QB concepts), the directed crawling strategy should terminate after finite steps.

Besides all the relevant data and metadata of qb:DataSets, we collect the following further information:

- The City Data Ontology[28] (CDP ontology) that contains lists of common statistical indicators about cities.
- The QB Equations Ontology[29] that contains the vocabulary to describe QB equations and is further detailed in Section 6.
- The Eurostat QB equations[30] that contains a set of QB equations generated from formulas published by Eurostat as further detailed in Section 6.
- Background information[31] that links indicators of Estatwrap to the CDP ontology as further described in Section 4.3.
- Background information providing additional owl:equivalentProperty links[32] between common dimensions not already provided by the wrappers such as between the different indicator dimension URIs estatwrap:indic_ur, cd:hasIndicator and eurostat:indic_na.
- The Global Cube Dataset[33] that defines the common URIs for dimensions, measure and dimension values and that is described further in Section 4.4.

Besides explicit information available in the RDF sources, we also materialise implicit information to 1) make querying over the triple store easier and 2) automatically evaluate relevant QB and OWL semantics. We execute the QB normalisation algorithm[34] in case the datasets are abbreviated. Also, we execute entailment rules[35] for OWL and RDFS. However, we only enable those normalisation and entailment rules that we expect to be evaluated quickly and to provide sufficient benefit for querying.

The crawling is implemented as a Linked Data-Fu program[36] and executed once a night using the Linked Data-Fu interpreter [5] with Version 0.9.9. The crawled data is then made available for loading into a triple store.[37] The large RDF file resulting from the crawl is loaded once a night into a OpenLink Virtuoso triple store (v07) using the standard RDF bulk loading feature.[38] Afterwards data gets preprocessed further, as described in the following sections.

4.3. Linking Statistical Data

In our scenario integration means building a unified view that allows to query over several datasets from the web as if they would reside in a single database. To allow this we need equivalence mappings and joining operations to build and query the unified view over Statistical Linked Data.

Take as an example we want to query all values of the indicator "population" of the area "Vienna", in the year "2010" simultaneously over two datasets: The two datasets may use different identifiers for the same dimensions, e.g., eurostat:geo and sdmx-dimension:refArea and dimension values, e.g., eurostat:dic/geo#AT13 and dbpedia:Vienna.

Therefore, we define equivalence classes of objects and define a common representation that we use in queries:

- For the time dimension, we use dcterms:date that is commonly used in many Statistical Linked Data.
- For the time dimension values, we use single years represented as String values such as "2015".
- For the geo dimension, we use sdmx-dimension:refArea that is recommended by the QB standard and link to it from other representations such as eurostat:geo, eurostat:cities and eurostat:metroreg.
- For the geo dimension values, we use instances of dbpedia:City like dbpedia:Vienna and link to these instances from other representations such as eurostat:dic/geo#AT13.
- For the indicator dimension, we use cd:hasIndicator and link to it from other representations such as eurostat:indic_na and eurostat:indic_ur.
- For the indicator dimension values, we use instances of cd:Indicator such as cd:unemployment_rate and link it from other representations such as the indicator from Eurostat eurostat:dic/indic_ur#EC1020I. For the indicator dimension values, we defined the CDP ontology as the main hub of indicator URIs to link to since no dataset with common indicators existed, yet.
- For all the other dimensions such as sex, age, unit, we used the one that is either recommended by QB or mostly used, e.g., sdmx-dimension:sex, eurostat:unit, and sdmx-dimension:age.

The following RDF snippet contains example links between two dimensions and two dimension values:

```
eurostat:geo owl:equivalentProperty sdmx-dimension:refArea.
eurostat:dic/geo#AT13 owl:sameAs dbpedia:Vienna .
```

[28] http://citydata.wu.ac.at/ontology.ttl
[29] http://citydata.wu.ac.at/ocdp/qb-equations
[30] http://citydata.wu.ac.at/ocdp/eurostat-equations
[31] http://kalmar32.fzi.de/triples/indicator-eurostat-links.nt
[32] http://kalmar32.fzi.de/triples/dimension-property-links.nt
[33] http://kalmar32.fzi.de/triples/global-cube.ttl#global-cube-ds
[34] https://www.w3.org/TR/vocab-data-cube/#normalize-algorithm
[35] http://semanticweb.org/OWLLD/
[36] http://kalmar32.fzi.de/ecdp.ldf
[37] See http://kalmar32.fzi.de/ for a collection of information about the crawling process as well as links to the resulting RDF file
[38] See http://citydata.wu.ac.at/ocdp/import for a collection of information about the loading process.

Since most publishers follow the practice of using an unspecific measure sdmx-measure:obsValue and a dimension indicating the measured variable, e.g., estatwrap:indic_na, and since cubes with multiple measures can be transformed to this form by introducing a new measure dimension, for the remainder of this paper we assume data cubes to have only one general measure, sdmx-measure:obsValue.

If we want to pose one query over the two datasets, we 1) specifically write the query to consider possibly different identifiers (i.e., need to know all identifiers) or 2) assume existing links and reasoning. Then, if we query for values for the canonical identifiers (as for any other identifier in the equivalence class), we also get the values for the other identifiers.

We now describe how we generated these links to map data from different sources to the common vocabulary:

- For the dimension properties, we set the links – if not available by the original data source – manually and published them as RDF to be crawled.
- For the geo dimension URIs: 1) UN-wrap already links to DBpedia URIs. 2) Estatwrap links were created based on the labels to "guess" the DBpedia city URIs.
- Links from Estatwrap indicators to the CDP ontology we created semi-automatically from an Excel sheet provided by Eurostat from which we generated the city data indicator URI and published as RDF for crawling.[39]

4.4. Materialisation of the Global Cube

As the foundation to efficiently query Statistical Linked Data – and in turn enrich as described in Section 5 and Section 6 – we define and materialise a unified view of all crawled datasets about cities, the *global cube* [9]. Table 4 gives an overview of how datasets published as Statistical Linked Data contribute to the global cube. The table shows in the rows datasets and in the columns all their dimensions. The cells give example values for a dimension, "-" if the dimension is not and "..." if the dimension may be used. For readability reasons, we describe URIs with namespaces,[40] slightly abusing the W3C CURIE syntax for expressing compact URIs. Relative URIs such as :DE and :00 are defined by the data source in the respective context.

Note that datasets may use different identifiers for dimensions and dimension URIs which however can be linked to canonical URIs. Remember that we use URIs as unique identifiers for datasets, dimensions, and dimension values from different data sources. Also, remember we assume data cubes to have only one general measure, sdmx-measure:obsValue.

Definition 1 (Global Cube). Based on the definition of operations such as Projection, Dice, Slice, Roll-Up and Drill-Across in earlier work, we define the global cube [9] as follows. Given the set of all available cubes {:ds1, . . . , :dsn} with dimension = {:D1, . . . , :Dn} the set f all dimensions of these available cubes, we define the global cube globalcube:global-cube-ds with dimension(globalcube:global-cube-ds) = dimension. The global cube is defined in terms of the available cubes; for cube :dsi with dimension(:dsi) = {:D1, . . . , :Dj}, the following holds (as N3 rule [5]):

```
{
        ?obs qb:dataSet :dsi.
        ?obs :D1 ?d1.
        ...
        ?obs :Di ?dn.
        ?obs sdmx-measure:obsValue ?value.
} => {
        _:obs1 qb:dataSet globalcube:global-cube-ds ;
               dcterms:publisher :dsi ;
               :D1 ?d1.
               ...
               :Dj ?dj ;
               :Dn-j+1 globalcube:ALL-value ;
               ...
               :Dn globalcube:ALL-value ;
               sdmx-measure:obsValue ?value .
}
```

We denote with globalcube:ALL-value the ALL member [15] aggregating over all possible values in the dimension. Thus, dimensions not used in available cubes are regarded as sliced with respect to the global cube. An OLAP query Q over the global cube with S sliced dimensions then can be answered by:

$$Q(\text{globalcube}) = \text{Drill-Across}_{ds \in DataCube, dimension \setminus S \subseteq dimension(ds)} Q(ds)$$

Since Drill-Across is commutative, the query result over the global cube does not depend on the order of Drill-Across operations.

When using this definition to materialise the global cube, we solve several issues: When integrating values from several datasets with the same dimension values, we need to make sure that the global cube does not to violate the QB specification integrity constraint IC-12, that says "No two qb:Observations in the same qb:DataSet may have the same value for all dimensions.". Therefore, we add another dimension to the global cube, dcterms:publisher, providing as contextual information (provenance) the dataset URI that has provided these numbers.

We require the global cube to explicitly contain global-cube:ALL-value ALL member values [15] when datasets do not serve all dimensions, as visible for several dimensions in Table 4.

In theory, the global cube may have as many dimensions as all relevant datasets exhibit. In our scenario of the Open City Data Pipeline, the relevant datasets all exhibit the three dimensions for dcterms:date (short time or year), sdmx-dimension:refArea (geo, city) and cd:hasIndicator (indicator) as well as possibly dimensions such as sdmx-dimension:sex (sex, gender), estatwrap:unit (unit) and sdmx-dimension:age (age).

We have published the metadata of the global cube as Statistical Linked Data.[41] Besides our general measure (sdmx-measure:obsValue) the qb:DataStructureDefinition of the global cube uses above mentioned dimensions. Also, we have defined instances of qb:AttributeProperty for cd:estimatedRMSE (for describing the error), cd:preferredObservation

[39] http://kalmar32.fzi.de/triples/indicator-eurostat-links.nt
[40] Use http://prefix.cc/ to look up prefix definition.
[41] http://kalmar32.fzi.de/triples/global-cube.ttl#global-cube-ds

Table 4: Overview of data cubes contributing to the global cube in our scenario (in rows) with their dimensions (in columns) and dimension members (in cells).

Dataset \Dimension	cd: hasIndicator	sdmx-dimension: refArea	dcterms: date	sdmx-dimension: sex	estat-wrap: unit	sdmx-dimension: age	...
eurostat:id/ nama_10r_3popgdp#ds (Eurostat Population)	cd: population...	dbpedia:Vienna...	2001...	-	:THS...	-	-
eurostat:id/ lfst_r_lfp2act#ds (Eurostat Economically Active Population)	cd: economically_ active_ population_ total...	dbpedia:Vienna...	2001...	:F...	:THS...	:Y18...	-
eurostat:id/ lfst_r_lfu2ltu#ds (Eurostat Long-term Unemployment)	cd: persons_ unemployed_ total...	dbpedia:Vienna...	2001...	-	:THS...	-	-
eurostat:id/ urb_cpop1#ds (Eurostat Urban Audit Population)	cd:population_ female...	dbpedia:Vienna...	2001...	-	-	-	-
undata:240 (UN Data Population)	cd:population_ female...	dbpedia:Vienna...	2001...	-	:COUNT...	-	-
...	...	...	...	...	...	...	...

(for linking to more reliable values), `prov:wasGeneratedBy` (for describing provenance information) and `prov:generatedAtTime` (for the time of generation) that help to interpret and evaluate the trustworthiness of values.

We implemented the materialisation of the global cube as a single SPARQL query (see AppendixA).

1. The query covers all combinations (necessary dimensions are year, city, indicator; possibly optional dimensions are sex, unit, age);
2. Considers all dimension properties equivalent to the canonical properties for time, indicator, city, sex, unit and age;
3. Sets the new `dcterms:publisher` source dimension with the dataset URI;
4. Sets all not set dimensions to the special-type `globalcube:ALL-value`. For that, uses a pattern to assign a variable $?x$ a default value d (which could be an RDF term or a variable) if an optional pattern did not match (y and z are arbitrary RDF terms or variables): `OPTIONAL { ?x_1 y z } BIND(COALESCE(?x_1, d) AS ?x);`
5. Via a filter makes sure that only datasets are included that do not show other dimensions;
6. Uses for the global cube the canonical dimension URIs; and
7. Loads all triples to a new graph for more efficient querying inside this graph.

Note, the query needs to be executed only once since the rules defining the global cube (Definition 1) are not recursive.

We analyse the complexity of the materialisation SPARQL query as follows: The query could be split up into separate queries for the possible dimension combinations (e.g., datasets that only exhibit indicator, geo, time; datasets that exhibit indicator, geo, time, sex; and so on). The well-known CUBE operator [13] has complexity $O(2^N)$ because of 2^N group bys (with N the number of dimensions). Similar to this CUBE operator, the split up query would have complexity $O(2^N)$ because we need to look for datasets with the selected number of dimensions in order to fill all the other dimensions with the default `ALL` value.

Now, the query could also be specified more by making explicit the dimension URIs. Then, all possible equivalent dimension combinations have to be tried in order to materialise the global cube from all possible relevant datasets. This materialisation approach requires for every combination now $O(M^N)$ because it needs M^N unions with N the number of Dimension and M the minimum number of equivalent dimensions. Therefore, the total complexity of the global cube materialisation query can be estimated by: $O(2^2N)$.

The query works in practise for up to 100k observations. When approaching 1m observations the global cube materialisation query could not be evaluated anymore due to issues with the SPARQL endpoint (query timed out). Thus, we resorted to a different approach involving several steps, effectively modularising the different tasks into several queries:

1. One SPARQL query to tag all the relevant from the crawled datasets (these were all that exhibited dimensions year, city, indicator and optionally sex, unit, age using a similar `FILTER NOT EXISTS` pattern),[42]
2. one SPARQL query to ask for all the schemas (`qb:DataStructureDefinition`) of the cubes, looking up the used dimensions for year, city, indicator, sex, unit, and age,[43]
3. one SPARQL query for each of the three dimension combinations to materialise the global cube,[44] and
4. two SPARQL queries to perform the entity consolidation for indicators and cities.[45]

We attach provenance information for each value using the PROV vocabulary. We can use the provenance information in

[42] http://citydata.wu.ac.at/ocdp/loadqueries/pre-canonicalise4.rq

[43] http://citydata.wu.ac.at/ocdp/loadqueries/pre-canonicalise6.rq

[44] http://citydata.wu.ac.at/ocdp/loadqueries/canonicalise1.rq, http://citydata.wu.ac.at/ocdp/loadqueries/canonicalise2.rq and http://citydata.wu.ac.at/ocdp/loadqueries/canonicalise3.rq

[45] http://citydata.wu.ac.at/ocdp/loadqueries/post-canonicalise1.rq and http://citydata.wu.ac.at/ocdp/loadqueries/post-canonicalise2.rq

queries to the global cube. For instance, to query the materialised global cube for all the values of the indicator "population" in the city "Wien" over all years, for all sources, including provenance information, see the following SPARQL query:

```
SELECT DISTINCT ?obs ?year (xsd:decimal(?snvalue) AS ?value
    ) ?source ?estError ?prefObs ?genBy ?genWhen
FROM <http://citydata.wu.ac.at/qb-materialised-global-cube>
WHERE {
?obs qb:dataSet globalcube:global-cube-ds.
?obs dcterms:publisher ?source.
?obs dcterms:date ?year.
?obs sdmx-dimension:refArea <http://dbpedia.org/resource/
    Wien>.
?obs cd:hasIndicator <http://citydata.wu.ac.at/ns#
    population>.
?obs sdmx-dimension:sex globalcube:ALL-value.
?obs estatwrap:unit ?unit.
?obs sdmx-dimension:age globalcube:ALL-value.
?obs cd:estimatedRMSE ?estError.
?obs sdmx-measure:obsValue ?snvalue.

OPTIONAL { ?obs cd:preferredObservation ?prefObs
OPTIONAL { ?obs prov:wasGeneratedBy ?genBy
OPTIONAL { ?obs prov:generatedAtTime ?genWhen
} ORDER BY ?year ?estError
```

In this query, we query optional provenance information based on the QB dimension `dcterms:publisher` (the `qb:DataSet` publishing the source) and the QB attributes `cd:estimatedRMSE` (estimated RMSE), `cd:preferredObservations` (the – possibly reflexive – preferred/more trustworthy observation), `prov:wasGeneratedBy` (provenance activity) and `prov:generatedAtTime` (the time of generation).

For instance, for the population of Vienna in 2005, we not only get an answer by two separate trustworthy sources, Eurostat (1.633M) and UN-Data (1.626M), but also from a prediction with estimated error of 0.27 (296K).

To only query the most trustworthy values, we can add a `FILTER NOT EXIST` pattern as follows, making sure no other value with the same dimension value combinations and a lower error estimation is available:

```
FILTER NOT EXISTS {
?obsa dcterms:date ?year.
?obsa sdmx-dimension:refArea <http://dbpedia.org/resource/
    Wien>.
?obsa cd:hasIndicator <http://citydata.wu.ac.at/ns#
    population>.
?obsa sdmx-dimension:sex globalcube:ALL-value.
?obsa estatwrap:unit ?unit.
?obsa sdmx-dimension:age globalcube:ALL-value.
?obsa sdmx-measure:obsValue ?valuea .
# either get error value or use default value
OPTIONAL { ?obsa cd:estimatedRMSE ?errorap }
BIND(COALESCE(?errorap, 0.0) AS ?errora)

FILTER(?errora < ?estError) }
```

For querying existing values, it is necessary to decide which dimensions and dimension values are relevant or need to be aggregated over (i.e., set to `globalcube:ALL-value`).

Setting dimensions to `globalcube:ALL-value` can also be done during the generation of new values so that it may be necessary to define intermediary QB equations (e.g., computing the unit "Thousand" to "ALL" by multiplying the value by 1000).

Now, given this approach of materialising and querying the global cube, new values can be generated from existing ones inside the global cube.

For both the imputation and inference methods described in the following sections, SPARQL queries are generated that query the global cube for the necessary existing values to generate new values. The newly imputed and inferred values also include provenance information.

5. Predicting Missing Values

As discussed in the motivation, the filling-in of missing values is a central requirement for the OCDP, as we discovered a large number of missing values in our datasets (see Table 2 and 3).

Base Methods. Our assumption is that every indicator has its own distribution (e.g., normal, Poisson) and relationship to other indicators. Hence, we aim to evaluate different regression methods and choose the best fitting model to predict the missing values. We measure the prediction accuracy by comparing the *normalized root mean squared error* in % (RMSE%) [17] of every regression method. In the field of Data Mining [17, 18] (DM) various regression methods for prediction were developed. We chose the following three "standard" methods for our evaluation due to their robustness and general performance. The chosen methods *K-Nearest Neighbour Regression*, *Multiple Linear Regression*, and *Random Forest Decision Trees* are straightforward to apply and show a robust behavior.

K-Nearest-Neighbour Regression (KNN), models denoted as M_{KNN}, is a wide-spread DM technique based on using a distance function to partition the instance space. As stated in [18], the algorithm is simple, easily understandable and reasonably scalable. KNN can be used in variants for clustering as well as regression.

Multiple Linear Regression (MLR), models denoted as M_{MLR}, has the goal to find a linear relationship between a target and several predictor variables. The linear relationship can be expressed as a regression line through the data points. The most common approach is *ordinary least squares* to measure and minimize the cumulated distances [18].

Random Forest Decision Trees (RFD), models denoted as M_{RFD}, involve the top-down segmentation of the data into multiple smaller regions represented by a tree with decision and leaf nodes. Each segmentation is based on splitting rules, which are tested on a predictor. Decision nodes have branches for each value of the tested attribute and leaf nodes represent decision on the numerical target. A random forest is generated by a large number of trees, which are built according to a random selection of attributes at each node. We use the algorithm introduced by Breiman [27].

5.1. Preprocessing

The preprocessing starts with the extraction of the base data from the global cube. We use the following SPARQL queries with the fixed period of *2004–2016* and the selection of the top hierarchy for the dimensions sex and age:

```
SELECT DISTINCT ?city ?indicator ?year ?value
FROM <http://citydata.wu.ac.at/qb-materialised-global-cube>
WHERE {

?obs dcterms:date ?year.
?obs sdmx-dimension:refArea ?city.
```

```
?obs cd:hasIndicator ?indicator.
?obs sdmx-measure:obsValue ?value.
?obs sdmx-dimension:sex kalmar:ALL-value.
?obs sdmx-dimension:age kalmar:ALL-value.

OPTIONAL { ?obs cd:preferredObservation ?po }.
FILTER(xsd:integer(?year) >= 2004 and xsd:integer(?rmse) =
    0)
FILTER( !bound( ?po ) )
} ORDER BY ?indicator ?city ?year
```

The SPARQL query "flattens" the multidimensional data to an input dataset as a matrix with tuples of the form:

$$\langle City, Indicator, Year, Value \rangle.$$

Based on the initial matrix, we perform the preprocessing as follows:

- Removing boolean and nominal columns, as well as all weather related data and sub-indicators in the U.N. data set, e.g., *occupants of housing units with 2 rooms*;
- Merging the dimensions year and city, resulting in: $\langle City\ Year, Indicator, Value \rangle$;
- Transposing the initial matrix by moving the indicators into the columns, resulting in tuples of the form: $\langle City\ Year, Indicator_1 Value, \ldots, Indicator_n Value \rangle$;
- Deleting columns and rows which have a missing values ratio larger than 95%.

Our initial data set from Eurostat and UNSD contains 2 974 cities with 207 indicators. By merging city and year and transposing the matrix we create 8 545 city/year rows. And after deleting the cities/indicators with a missing values ratio larger than 95%, we have the final matrix of 7 905 rows (city/year) with 146 columns (indicators).

5.2. Approach 1 - Building Complete Subsets

In the first approach (A1), we try to build models for a target indicator by directly using the available indicators as predictors. For this, we are using the correlation matrix of the data to find indicators which are suitable predictors. Subsequently, we build a complete subset from our data, i.e., we first perform a projection on our data table, keeping only the predictors and the specific target as columns. More detailed, our approach has the following steps on the initial data set, the matrix A_1 and a fixed number of predictors n (we test this approach on different $n's$):

1. Select the target indicator I_T;
2. Calculate the correlation matrix A_C of A_1 between I_T and the remaining indicators;
3. Create the submatrix A_2 of A_1 with I_T and the n "best" indicators (called the predictors). The predictors are selected according to the highest absolute correlation coefficients in A_C;
4. Create the complete matrix A_3 by deleting all rows in A_2 with missing values;

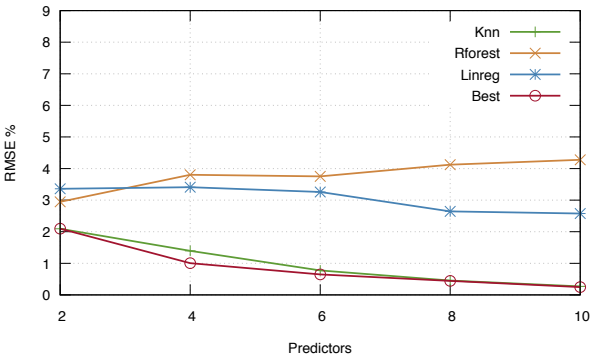

Figure 6: Prediction results Approach 1

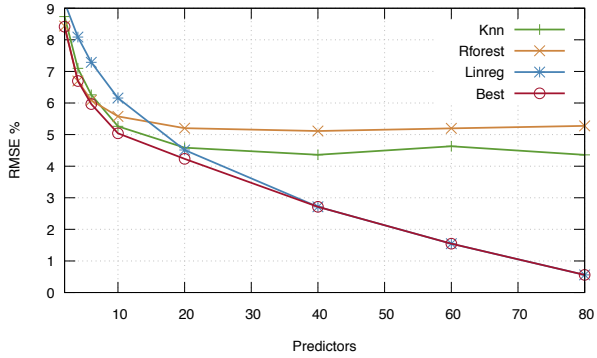

Figure 7: Prediction results Approach 2

5. Apply *stratified tenfold cross-validation* (see [17]) on A_3 to obtain ten training- and test sets. Then, train the models M_{KNN}, M_{MLR}, and M_{RFD} using the training sets. Finally, calculate the mean of the ten RMSE% based on the test set for each model and choose the best performing model M_{Best} accordingly;
6. Use the method for M_{Best} to build a new model on A_2 for predicting the missing values of I_T.

Evaluation. The performance of the regression methods were evaluated for two to ten predictors. Two regression methods have their best RMSE% with ten indicators: 0.27% for KNN and 2.57% for MLR. Whereas RFD has the best RMSE% of 4.12% with eight indicators. Figure 6 gives an overview of the results. By picking the best performing regression for every indicator (red line) the median RMSE% can be reduced only slightly. For ten predictors the median RMSE% improves to 0.25% over KNN with 0.27%. Depending on n, we fill-in between 122 056 for ten and 296 069 values for two predictors. For a single city and ten predictors, the number of predicted values range from 7 to 1 770. The limited number of filled-in values is due to the restriction of using the complete matrix for the regression methods.

5.3. Approach 2 - Principal Component Regression

In the second approach (A2), we omit the direct use of indicators as predictors. Instead, we first perform a Principal

Component Analysis (PCA) to reduce the number of dimensions of the data set and use the new compressed dimensions, called *principal components* (PCs) as predictors. As stated in [18], the PCA is a common technique for finding patterns in data of high dimensions. Parts of the evaluation is similar to Approach 1, but we have an additional step where we *impute* all the missing values with *neutral* values for the PCA. The neutral values are created according to the *regularized iterative PCA algorithm* described in [7]. This step is needed to perform the PCA on the entire data set. The following steps are evaluated having an initial data set A_1 as a matrix and a predefined number of predictors n (we test this approach also on different $n's$):

1. Select the target indicator I_T;
2. Impute the missing values in A_1 using the regularized iterative PCA algorithm resulting in matrix A_2 and remove the column with I_T;
3. Perform the PCA on the A_2 resulting in a matrix A_3 of a maximum of 80 PCs;
4. Append the column of I_T to A_3 creating A_4 and calculate the correlation matrix A_C of A_4 between I_T and the PCs;
5. Create the submatrix A_5 of A_4 on the selection of the PCs with the highest absolute correlation coefficients and limit them by n;
6. Create submatrix A_6 of A_5 for validation by deleting rows with miss. values for I_T;
7. Apply stratified tenfold cross-validation on A_6 with the Step 5 from Approach 1, which results in the best performing model M_{Best};
8. Use the method for M_{Best} to build a new model on A_5 (not A_6) for predicting the missing values of I_T.

5.4. Evaluation and Publishing

Figure 7 shows the results for an median RMSE% with an increasing number of predictors and compares the performance of KNN, RFD, MLR, and the selection of best method. Clearly, for 80 predictors MLR performs best with a median RMSE% of 0.56%, where KNN (resp. RFD) has a median RMSE% of 4.36% (resp. 5.27%). MLR is the only method that improves steady up to 80 predictors. KNN provides good results for a lower number of predictors, but starts flattening with 20 predictors. Contrary to MLR, KNN and MLR have to be adjusted according to number of predictors, hence optimizing the number of clusters for KNN could improve the result. The red line in Figure 7 shows the median RMSE% with the best regression method chosen. Up to 60 predictors, the overall results improves by selecting the best performing method (for each indicator). The best median RMSE% of 0.55% is reached with 80 predictors, where MLR is predominant and only 3 out of 145 indicators are predicted by KNN. Compared to the result of the ISWC experiments, the median RMSE% improved from 1.36% to 0.55%, which mainly related to the lower sparsity of the datasets.

We only publish the predicted values created by Approach 2 (see a discussion in Chapter 7) and apply a threshold of RMSE% of 20% as a cut off. This leads to 5 indicators (e.g. *price of a m^3 of domestic water in EUR*) being dropped for the execution with 80 predictors. Following our strategy of using statistical linked data wrappers, we publish the predicted values using the *Missing Values Wrapper*,[46] which provides table of content, structure definition, and datasets, where a new dataset is created for each prediction execution with different parameter (i.e., the number of predictors).

5.5. Workflow and Provenance

The full prediction workflow is shown in Figure 8 and is based on all the observed values ignoring old predicted values in the global cube. *Data cleansing and transposing* is written in Python, but all other steps such as *PCA*, *model building*, and *model evaluation* are developed in R [28] using its "standard" packages. All the scripts and their description are available on the website of the *Missing Values Wrapper*. We conducted an evaluation of the execution time on our Ubuntu Linux server with 2 cores, 2.6 GHz, and 16 GB of RAM. A single prediction run requires approx. 6min for each indicator (2min for each method) resulting in a total time of approx. 12 hours for all indicators.

One can see that the workflow branches after four steps, where we distinguish two cases. In the case of no previous executions, we perform the full prediction steps as described in the previous section. In the case of previous executions, we already have provenance information available in our triple store, which describes the last execution and the related model provenance data (for each indicator). The model provenance includes for each indicator the number of predictors, prediction method, method parameter (i.e., the number of clusters in the KNN), the number of PCs, and the RMSE%. Note, that initially we aimed to store the entire model, however due to Approach 2, we would need to adjust the PCs for each indicator with the updated values, which had the effect of recalculating the full workflow. However, we still save evaluation time since, only one model (and not 3) has to be rebuild and evaluated. Summarizing, we keep provenance for our predictions on three levels:

- For each execution, we publish the RMSE%, creation date, and the creation agent;
- For each indicator, we publish the above mentioned model provenance data;
- For each predicted value published as a `qb:Observation`, we publish the overal RMSE% and the absolute RMSE. Further, we point to better observations (published with an lower RMSE%), which might occur if another approach as QB Equations improve the predicted values.

For describing the model provenance, we use the *MEX vocabulary*, which is compared to other vocabularies (i.e., DMOP [29]) lightweight and designed for exchanging machine learning metadata [30]. We use the *MEX Algorithm* layer to describe our prediction method and its parameter and the *MEX Performance* layer to describe the RMSE%. Further, we describe each execution using attributes from *MEX Execution*.

[46] http://citydata.ai.wu.ac.at/MV-Predictions/

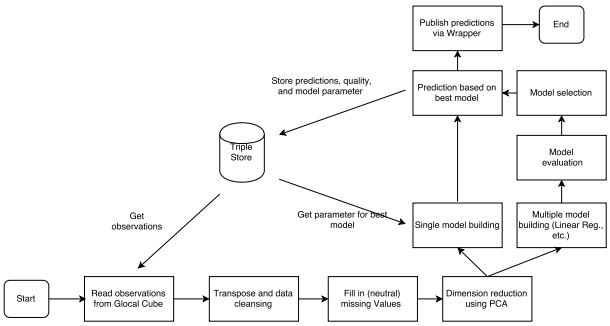

Figure 8: Prediction Workflow

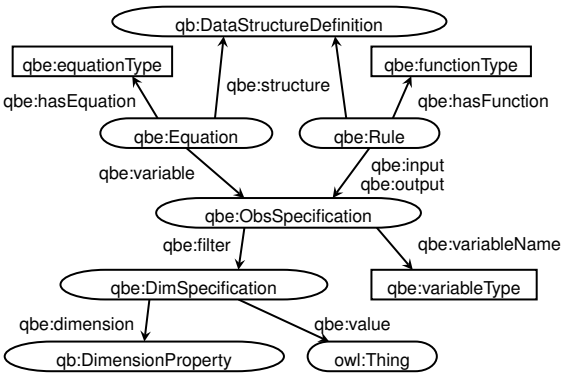

Figure 9: QB equations ontology

6. QB Equations

Usually ontological reasoners are limited to infer new knowledge in the domain of instances. Reasoning in the domains of literal values such as numbers is out of scope for such reasoners. In fact ontological reasoners can not use equational knowledge, which is knowledge given as equations which relates number values, for inferring new numerical knowledge, although such knowledge is already published in some form, such as the QUDT ontology [31].

For statistical data, especially statistical linked data, such knowledge can be interesting to fill in the gaps of missing values which usually occur and give the user of an RDF data warehouse a better user experience. Examples for useful equational knowledge include unit conversion, indicator definitions, or linear regression models. Currently there exists no framework to exploit this knowledge to infer new statistical data.

With QB equations we introduce a framework to exploit equational knowledge to infer new numerical data. Similarly QB rules can express unidirectional relationships, using non-invertible functions, e.g., rounding of values, specific forms of aggregation, or linear regression models. QB equations include an RDF syntax for representing equational knowledge and a formal semantics for reasoning. We propose an implementation strategy used in the OCDP allowing to encode relationships between numerical observations and to derive new numerical values based on these relationships.

Example 6.1. We can relate some observation value given in kilometres to the same observation given in miles with an equation: $d\,\mathrm{km} = d\,\mathrm{mi} \times 1.609344$. This equation is a high level specification of the *relationship* between observation values.

When encoded in RDF we call these relationships *QB equations* or *QB rules*. QB equations specify relationships between observations which can be reformulated into different "directions" while QB rules are valid only in one direction.

The approach of QB equations presented in the following is a combination and extension of two approaches which we earlier called "RDF attribute equations" [8] and "complex correspondences" [9].

6.1. Syntax

We express QB equations in an RDF syntax. Since – to the best of our knowledge – no vocabulary exists for this purpose so far we have to introduce a new vocabulary expressing QB equations and QB rules.

Each QB equation is identified by an IRI and consists of two parts: (i) a representation of a mathematical equation using (arithmetic) functions and variables, and (ii) a mapping of observations to variables using observation specifications. Additionally each equation is related to a QB DSD with the property `qbe:structure` – inspired by the `qb:structure` property.

Figure 9 gives an overview of the QB equations ontology showing all the introduced classes, properties, and datatypes as well as reuse of the QB ontology.

Representation of the equation or function. One possibility to represent equations in RDF would be building an operator tree in RDF, similar to, e.g., the RDF-reification of OWL axioms. Such an encoding could use RDF data structures such as RDF lists or might be derived from other representations such as MathML, which defines an XML representation of mathematical concepts. While such a representation can be interesting for some use cases, we refrained from such a reified representation, as it is difficult to handle with standard tools such as SPARQL queries and the added complexity is not necessary.

Rather we define the core of the QB equations, which is the equation itself, as a literal that directly reuses SPARQL's arithmetic expression syntax,[47] i.e., we use a datatype literal with the datatype `qbe:equationType`, the lexical space of which is defined by the following grammar rule (in the syntax and referring to non-terminal symbols of the SPARQL grammar):

equationType ::= Var '=' NumericExpression

This choice enables standard SPARQL parsers or other standard libraries for mathematical expressions for processing these equations, and – as we will see – straightforward implementation of the application of equations by SPARQL engines. An example for an equation for a simple unit conversion is

```
"?mile = ?km * 0.6214"^^qbe:equationType.
```

The property `qbe:hasEquation` relates an instance of an equation `qbe:Equation` to such an equation literal.

[47] cf. http://www.w3.org/TR/sparql11-query/#rNumerivExpression

The lexical space of datatype `qbe:variableType` is – analogous to `qbe:equationType` – defined by the SPARQL grammar non-terminal 'Var'.

Observation specification. The second part maps observations to the variables used in the equation. Usually observations are specified by giving values for all of the dimensions. This approach would be too constraining and might lead to multiple representations of essentially the same equation. Instead an observation specification only needs values for some of the dimensions. In an example of unit conversions, one would only specify the value of the unit dimension because the equation should be applicable to any kind of observation given in that unit, regardless of the values of the other dimensions. Intuitively the values of all other unspecified dimensions must be the same among all specified observations.

Example 6.2. The following example shows the complete definition of the equation for unit conversion between kilometres and miles. The QB equation defines a variable `?km` which binds to all observations which have set the dimension `estatwrap:unit` set to `qudt-unit:Kilometer`. The second variable `?mile` is defined analogously. Eventually the QB equation gives the equation relating the variables as a `qbe:equationType`-typed literal.

```
ex:unit-km-mile a qbe:Equation ;
  qbe:variable [ a qbe:ObsSpecification ;
    qbe:filter [ a qbe:DimSpecification ;
      qb:dimension estatwrap:unit ;
      qbe:value qudt-unit:Kilometer ] ;
    qbe:variablename "?km"^^qbe:variableType ] ;
  qbe:variable [ a qbe:ObsSpecification ;
    qbe:filter [ a qbe:DimSpecification ;
      qb:dimension estatwrap:unit ;
      qbe:value qudt-unit:MileInternational ] ;
    qbe:variablename "?mile"^^qbe:variableType ] ;
  qbe:hasEquation "?mile = ?km * 0.6214"^^qbe:equationType.
```

QB equations can be evaluated in multiple directions, effectively creating a function to compute a value for each of the variables from all the other variables. In the example above we can infer observations given in miles from observations given in kilometres and vice versa. Obviously this works only for invertible functions including the usual arithmetic operators: addition, subtraction, multiplication, and division.[48] In fact, we can reuse the definition of simple equations from [8], which guarantee this property:

Definition 2 (from [8]). Let $\{x_1,\ldots,x_n\}$ be a set of variables. A *simple equation* E is an algebraic equation of the form $x_1 = f(x_2,\ldots,x_n)$ such that $f(x_2,\ldots,x_n)$ is an arithmetic expression over numerical constants and variables $x_2,\ldots,x_n$ where f uses the elementary algebraic operators '+', '−', '∗','/' and contains each x_i exactly once.

Equations of this form can be easily transformed into an equivalent form $x_i = f'(x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n)$ for each appearing variable x_i, $2 \leq i \leq n$. For instance, in our example the equation can likewise be used to convert miles to kilometres:

```
"?km = ?mile / 0.6214"^^qbe:equationType.
```

[48] while we have to take care of division by zero, for details cf. [8]

These equivalent transformations can be easily computed by standard mathematical libraries (which we will use in our implementation, cf. Section 6.3 below). A central piece of this transformation is a function *solve* with two parameters: the equation as string and the name of the target variable to solve for. The *solve* function algebraically solves an equation for a variable and returns a function. For example `solve("a=b/c",c)` would return the function `"b/a"` whereas `solve("a=b/c",b)` would return `"a*c"`. The function *solve* is implemented in every computer algebra system – for example in Maxima with roots going back to the 1960s. That is, we could write

$$f'(x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n) = solve(x_1 = f(x_2,\ldots,x_n),x_i)$$

Analogously to `qbe:equationType` we define a datatype `qbe:functionType` describing an arithmetic function or expression a whose lexical space is again defined via a SPARQL grammar non-terminal rule:

functionType ::= NumericExpression

Following the example for `equationType` a *function* for converting kilometres into miles is:

```
"?km * 0.6214"^^qbe:functionType.
```

QB rules (or functions) are similar to equations but can be evaluated only in one direction. Thus QB rules specify not variables but one or more input variables and exactly one output variable. These variables are specified in the same way as the variables in QB equations, while the output variable does not need a `qbe:variableName`.

Example 6.3. A QB rule to compute a population in thousands approximation, using the function `round` to demonstrate a non-invertible function.

```
ex:qbrule1 a qbe:Rule ;
  qbe:input [
    qbe:filter [
      qbe:dimension ex:indic ;
      qbe:value un:population ] ;
    qbe:variablename "?population"^^qbe:variableType ;
  ] ;
  qbe:output [
    qbe:filter [
      qbe:dimension ex:indic ;
      qbe:value ex:approx-pop-k ] ;
  ] ;
  qbe:function "round(?population/1000)"^^qbe:functionType.
```

6.2. Semantics

We define the semantics of QB equations by using a rule language. In fact, as indicated in Section 2.5 above, SPARQL INSERT queries can be seen as rules over RDF triple stores where the pattern of the INSERT clause is the rule head and the graph pattern in the WHERE clause is the rule body. We note that this "idea" is not new and straightforwardly implementing the same concept as interpreting CONSTRUCT statements as rules, introduced e.g. in [32], where we defined a formal semantics for such rules based on the Answer Set Semantics for non-monotonic Datalog programs (ASP) with external (built-in) predicates and aggregates [33]; builtin-predicates are introduced

Listing 1: Algorithm to convert QB equations to QB rules

```
R := {}
for each (?e, ?eq) where { ?e rdf:type qbe:Equation.
                           ?e qbe:hasEquation ?eq }:
   for each (?v, ?vname) where { ?e qbe:variable ?v .
                                 ?v qbe:variableName ?vname }
       ?f := solve(?eq, ?vname)
       add r(?e, ?vname, ?f) to R
return R
```

Listing 2: Algorithm to create a QB rule

```
def r(?e, ?outvar, ?f):
rule := empty graph
?rulename := sk(?e, ?vname)
rule := { ?rulename a qbe:Rule .
          ?rulename qbe:hasFunction ?f .
          ?rulename prov:wasDerivedFrom ?e .
          ?rulename qbe:output ?outvar . }
for each (?v, ?vname) where  { ?e qbe:variable ?v .
                               ?v qbe:variableName ?vname }
    if ?vname != ?outvar:
        add { ?rulename qbe:input ?v } to rule
return rule
```

in SPARQL1.1 through expressions and assignment (BIND ... AS), and non-monotonicity in SPARQL is introduced by features such as OPTIONAL and NOT EXISTS.

We explain the semantics of QB equations in three steps: (i) normalisation of QB equations to QB rules, (ii) conversion of QB rules to SPARQL INSERT queries, (iii) a procedure to evaluate a program (a set of SPARQL INSERT queries) until a fixpoint is reached. Note here, that in the general case rules with the expressive power of ASP with external predicates do not have a unique, finite fixpoint, but we will define/discuss how we can guarantee termination in our case.

6.2.1. Normalisation

A QB equation in n variables can be viewed as a meta rule representing n rules: as discussed before, for each variable x_i a rule to compute x_i from all the other variables in the equation e can be generated by resolving the equation to $x_i = solve(e, x_i)$. To simplify the semantics specification we thus first normalise each QB equation to n QB rules and then in the next step give the semantics for QB rules.

That is, the QB rules generated in the normalisation, have x_i as the output variable, and the other $(n-1)$ variables as input variables with $f'(x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n)$ being the function to compute the output.

Listing 1 shows the main algorithm of the conversion. The function `r` in Listing 2 takes three parameters: the original QB equation IRI, the name of the output variable, and the function. The function `sk(...)`, with a variable number of parameters is a Skolem function deterministically returning a unique IRI for each unique parameter combination.

Eventually, after applying Listing 1 we could replace all triple encoded QB equations in an RDF graph with the triple-encoded QB rules in R , i.e. we view these representations equivalent. The remainder of our semantics only deals with rules.

Example 6.4. The QB equation of Example 6.2 results in two QB rules, one for each of the two variables. The QB rule to compute an observation in miles from an observation given in kilometres is shown below. Instead of variables we now have `input` and `output` and the equation was replaced by a function in the input variable `?km`.

```
ex:unit-km-mile-km a qbe:Rule ;
 prov:wasDerivedFrom ex:unit-km-mile ;
 qbe:input [ a qbe:ObsSpecification ;
  qbe:filter [ a qbe:DimSpecification ;
   qb:dimension estatwrap:unit ;
   qbe:value qudt-unit:Kilometer ] ;
  qbe:variablename "?km" ] ;
  qbe:output [ a qbe:ObsSpecification ;
   qbe:filter [ a qbe:DimSpecification ;
    qb:dimension estatwrap:unit ;
    qbe:value qudt-unit:MileInternational ] ;
   qbe:variablename "?mile" ] ;
 qbe:hasFunction "?km * 0.6214"^^qbe:equationType .
```

The second QB rule accordingly represents the function to compute an observation in kilometres from an observation given in miles, with the QB function corresponding to the right-hand side of Section 6.1 above.

6.2.2. Rule Conversion

In this step QB rules are converted to SPARQL INSERT queries. The query has to implement several tasks: retrieve the input observations, compute the output observations, generate several URIs, propagate error and provenance and ensure termination when evaluated repeatedly.

Compared to other rule languages SPARQL queries provide very complex features, but, as shown earlier [34, 35], can be compiled to –essentially– non-recursive Datalog with negation, wherefore INSERT queries, read as rules, have the same expressivity.

Without loss of generality, we make the following assumptions (which could be easily checked in a pre-processing step, e.g., with a SPARQL ASK query assuring that there is a single measure value per observation):

- there is always only a single measure per observation
- the measure predicate that holds the measure value is fixed to `sdmx-measure:obsValue`

On a high level, the INSERT queries corresponding to QB rules have the following structure:

```
INSERT {
  output observation
    - according to QB vocabulary
    - with PROV annotations to describe the generation by
        QBE rules
    - error estimation  }
WHERE {
  one pattern for each input observation
    - with all dimensions specified in the DSD and error
        estimate

  BINDs for IRI creation for
    - ID for the newly generated observation
    - prov:Activity

  further BINDs to
   - assign current time to variable for PROV annotation
   - compute measure value of target observationn
   - estimate error of target observation

  Termination condition  }
```

Output observation. The output observation is set in the head with the fixed dimensions from the DSD and fixed dimension values if specified in the observation specification of the QB rule. The other dimension values are taken from the input variables. The rule head for Example 6.4 would look like the following query fragment, incorporating the PROV annotations:

```
?obs qb:dataSet globalcube:global-cube-ds ;
  cd:hasIndicator ?indicator ;
  dcterms:publisher ?source ;
  dcterms:date ?year ;
  sdmx-dimension:refArea ?city ;
  sdmx-dimension:sex ?sex ;
  estatwrap:unit qudt-unit:MileInternational;
  sdmx-dimension:age ?age ;
  sdmx-measure:obsValue ?value ;
  prov:wasDerivedFrom ?km_obs ;
  prov:wasGeneratedBy ?activity ;
  prov:generatedAtTime ?now ;
  cd:estimatedRMSE ?error .
```

It is important to note that the SPARQL INSERT application is *idempotent*, i.e., repeated applications of a generated SPARQL INSERT query will not add any more triples after the first application. Idempotence would be lost if blank nodes are used in the head, because they would create a fresh blank node for every application of a SPARQL query, even if the SPARQL query returns only a single result. Furthermore we have to ensure that all values generated by the query are completely determined by the variable bindings of the WHERE clause.

Provenance propagation. For every new derived observation we record the provenance, i.e., each derived observation has a link to each input observation `?obsin1, . . . , ?obsinN` and to the rule or equation used for the computation `?equation`. Firstly this provenance information provides transparency: We know precisely how a derived observation was computed. Secondly we can use the provenance information during the derivation process to ensure termination. Furthermore we record the time of the rule application and the agent, which could be the script or person responsible for the query creation.

```
?obs a prov:Entity ;
  prov:wasDerivedFrom ?obsin1, ... ?obsinN ;
  prov:wasGeneratedBy [
    a prov:activity ;
    prov:qualifiedAssociation [
      a prov:Association ;
      prov:wasAssociatedWith  ex:fred ;
      prov:hadPlan ex:unit-km-mile-km ] ] ;
  prov:generatedAtTime "2017-01-15T12:37:00" .
```

The preliminaries in Section 2.2 give an example of a part of the derivation tree generated by this rule head fragment.

Input observations. For each input observation one set of triple patterns which asks for one observation is generated for the SPARQL WHERE clause. For each `qbe:DimSpecification` a dimension value is fixed. For all the other dimensions a fixed variable is used in all input observations. In the example below, again generated from Example 6.4 the query contains for all dimension values variables, except for `estatwrap:unit` which is fixed to `qudt-unit:Kilometer` as specified by the QB rule input dimension specification. Furthermore the observation value and the error estimated are retrieved.

```
?km_obs qb:dataSet globalcube:global-cube-ds;
  dcterms:publisher ?km_src ;
  dcterms:date ?year;
  sdmx-dimension:refArea ?city ;
  cd:hasIndicator ?indicator ;
  sdmx-dimension:sex ?sex ;
  estatwrap:unit qudt-unit:Kilometer;
  sdmx-dimension:age ?age ;
  sdmx-measure:obsValue ?km_value ;
  cd:estimatedRMSE ?km_error .
```

Value creation with `BIND`. Several SPARQL variables used for the output observation need to be computed using variables from the input observations. Most importantly the output measure value has to be created using the function of the QB rule.

```
BIND(?km * 0.6214 AS ?value)
```

Several IRIs have to be generated for the rule head. We use a Skolem function to generate these IRIs. The inputs of this Skolem function are the IRI of the QB rule `rule`, the input variables `var1`, . . . `varN` and a string `"_static_"` to differentiate the different variables in the head. We implement this Skolem function with string concatenation and a hash function.

```
BIND(IRI(CONCAT(STR(rule), "_static_", MDA(CONCAT(STR(?var1
    ), ..., STR(?varN))))) AS ?targetvar)
```

We have to generate two URIs: observation, and PROV activity.

```
BIND(IRI(CONCAT("http://example.com/unit-km-mile-km", "
    _obs_", SHA1(CONCAT(STR(?km_obs))))) AS ?obs)
BIND(IRI(CONCAT("http://example.com/unit-km-mile-km", "
    _activity_", SHA1(CONCAT(STR(?km_obs))))) AS ?obs)
```

Furthermore we bind the current time to a variable to use in the provenance part of the head.

```
BIND(NOW() as ?now)
```

Error propagation. Values computed based on values with an associated error also need an error estimate. The procedure to estimate an error of the new value is called *error propagatione* [36, 37]. In our use case we do not promise precise statistical error quantifications, but just want to propagate an upper bound of the error estimations of the inputs to the computed output value. We chose a error propagation function which is simple to implement in standard SPARQL. To this end, we incorporate a relatively naive error propagation function which however can be adapted to more accurate estimations if necessary in the future [36, 37].

We proceed herein as follows. The error values we have from our predictions are given as RMSE, i.e., the root-mean-square-error, which intuitively characterises how far off in absolute numbers the actual value is on average from our prediction. To compute a conservative estimate of how these errors "add up" when used in computations, we proceed as follows. Depending on the function f used for computing the computed output value, the n variables $x_1, \ldots x_n$ and their associated indicators $ind_1, \ldots ind_n$, we denote by $r_1, \ldots r_n$ the estimated RMSEs for these indicators, i.e. $r_i = RMSE(ind_i)$.

In Table 5 we define the propagated estimated RMSE (per) of a computed observation recursively over the operator tree of the function term $expr = f(x_1, \ldots x_n)$. Intuitively, we assume here the following: if the real values x'_i for indicators ind_i lie exactly r_i away –i.e., exactly the estimated RMSE above ($x'_i =$

expr	*per*(*expr*)
const	0
x_i	r_i
$a+b$	$per(a)+per(b)$
$a-b$	$per(a)+per(b)$
a/b	$(\|a\|+per(a))/(\|b\|-per(b))-a/b$
$a*b$	$(\|a\|+per(a))*(\|b\|+per(b))-a*b$

Table 5: Computing the propagated estimated RMSE (*per*) for a given expression (*expr*)

$x_i + r_i$) or below ($x'_i = x_i - r_i$) – from the predicted value x_i, we intend to estimate how much off would a value computed from these predicted values *maximally* be; here, *const* denotes a constant value, and a, b are sub-expressions. Furthermore we assume that the RMSE r_i is always less than the observed value x_i.

If now, for an equation $x_f = f(x_1, \ldots x_n)$, the propagated estimated RMSE $per(f(x_1, \ldots x_n))$ is smaller than the so far estimated RMSE r_f for indicator ind_f then we assume it potentially pays off to replace the predicted value so far with the newly computed value by the rule corresponding to the equation.

To cater for rounding errors during the computation we add a small ε to the error estimate. In some sense this ε punishes each rule application and thus enables quicker termination later. Eventually the following BIND expression will be generated to compute the propagated error as defined by *per* and assign it to the corresponding variable used in the head of the rule (a more interesting example for the function `?nationals/?population`).

```
BIND((ABS(?nationals)+?nationals_error)/(ABS(?populatio)-?
     population_error) + 0.1 AS ?error)
```

Termination. So far we introduced triple patterns and `BIND` expressions into the rule body. As remarked above the `BIND` expressions implement Skolem functions and thus avoid duplicating the same output observations over and over again (our SPARQL INSERT queries are idempotent). We now give two different termination conditions which can be used separately or together to ensure termination of the QB rules program.

To ensure termination of the whole SPARQL INSERT rule program we use a similar termination condition as in earlier work [8], we block the repeated application of the same rule to derive a particular observation. With the PROV annotations in fact we create an equation dependency graph. Given an observation o, a SPARQL path expression o `prov:wasDerivedFrom+` o' returns all the observations o' transitively used in the computation of o. Furthermore the SPARQL path expression o `prov:wasDerivedFrom*/prov:wasGeneratedBy/prov:qualifiedAssociation/prov:hadPlan` r gives all the rules r transitively used during the computation of o. So, in order to ensure termination, we define that a QB rule r is only *applicable* to materialise an observation o if r does not occur in the result of that path expression.

In the SPARQL INSERT query can we implement this condition by adding one of the following patterns for each input observation `?i` where `r` is the URI of the rule (or equation) itself.

```
FILTER NOT EXISTS {
?i prov:wasDerivedFrom*/prov:wasGeneratedBy/prov:
     qualifiedAssociation/prov:hadPlan/prov:wasDerivedFrom?
     r }
```

Thus as a worst case the evaluation will be terminated by this condition after applying each rule n times, where n is the number of QB rules in the system, because after applying each rule once for the derivation of a single observation no rule can be applicable anymore. An example of such a worst case would be a chain of QB rules where $r_i = r_{i+1}$ and $0 < i < n$ and a single given observation for r_0.

Another termination condition is based on the error propagation described above. Intuitively the condition ensures that an observation o from a computation is only materialised if no observation o' exists that (i) shares the same dimension values and (ii) has a lower error estimate.

```
FILTER NOT EXISTS {
  ?obsa qb:dataSet globalcube:global-cube-ds ;
       dcterms:date ?year;
       sdmx-dimension:refArea ?city ;
       cd:hasIndicator ?indicator ;
       sdmx-dimension:sex ?sex ;
       estatwrap:unit qudt-unit:Kilometer;
       sdmx-dimension:age ?age ;
       cd:estimatedRMSE ?errora .

  FILTER(?errora <= ?error) }
```

In the error propagation we use a function to rather overestimate the error. While we materialise only better observations (i.e., observations with a lower error estimate) the error estimates are tending to increase with each rule application. Thus the rule program will terminate. Together the two termination conditions can lead to faster termination.

However, this would need to ensure that the error propagation "converges" in the sense that application of rules does not decrease error rates. Our simple method for error propagation would – in connection with cyclic rule application – not guarantee this as demonstrated by the following, simple example:

Example 6.5. For two indicators i and j let two equations be $i = j/2$ and $j = i/2$. Essentially, this boils down to the equation $i = i/4$ where – in each application – we would derive smaller error estimate.

While this example is quite obviously incoherent (in the sense of rules being cyclic in terms of the rule dependency graph defined in [8][Definition 12]), we still see that with cyclic application of rules the convergence of error rates cannot be ensured in general. In practice such incoherent systems of equations are hardly useful, however the first termination condition would still serve its purpose.

6.3. Implementation

As described in Section 6 we compile QB equations into a semantically equivalent set of rules. Usually there are two strategies for query answering in rule based knowledge bases: forward or backward chaining. For the OCDP we decided to implement a forward chaining approach to enrich the global data cube with the newly inferred observations. Forward chaining approaches materialise as much as possible thus allowing faster query evaluation times. On the other hand forward chaining

approaches require more memory or disk space and updates lead to re-materialisation. In our case the space requirements are manageable and updates are not frequent.

Our forward chaining implementation approach relies on the iterative application of SPARQL INSERT queries (which implement the rules). Overall, QB equations infer and persist new observations in three steps: (Normalisation) convert all QB equations to QB rules, (Rule conversion) for each QB rule we create a SPARQL query, and (Query evaluation) iteratively evaluate the constructed SPARQL INSERT queries until a fixpoint.

Normalisation. As described in the semantics above in this first step we convert QB equations to QB rules. The algorithm in Listing 1 already outlines our implementation. We implemented the algorithm using Python 2.7 and the libraries `rdflib` for RDF/SPARQL processing and `sympy` providing the *solve* function to algebraically solve an equation for a variable. The Python script reads the QB equations from an RDF file containing the QB equations,[49] converts them to QB rules and publishes them again as Linked Data[50] and in the triple store.

Create SPARQL queries. We create a SPARQL INSERT query for each QB rule. The listing in AppendixB gives as a complete example of the SPARQL INSERT query resulting from converting one of the QB rules. Due to a serious performance hit of SPARQL INSERT queries applied on the Virtuoso triple store in a preliminary evaluation, we used SPARQL CONSTRUCT queries instead, and load the resulting triples into the triple store afterwards. The conversion from QB rules to SPARQL CONSTRUCT queries is analogous to the algorithm described in Section 6.2.2 above to convert a QB rule to a SPARQL INSERT query. To cater for a bug in Virtuoso[51] we have to check each division for division-by-0. In the implementation we use an IF-expression to test each divisor for 0 and in this case return a value which will lead to an empty result (a static string `"err"`). This applies to the observation measure computation and to the error propagation computation.

Evaluate queries in rule engine. In principle the rule engine naively evaluates SPARQL INSERT queries, or respectively, CONSTRUCTs + re-loads, over and over again until a fixpoint is reached.

Apart from the termination conditions described in Section 6.2.2 we ensure that the repeated application of a rule on the same observations does not create any new triples by using Skolem constants instead of blank nodes (see also discussion on idempotency above). Thus, in order to check whether a fixpoint has been reached, it is enough to check in each iteration simply if the overall number of triples has changed or not. So, for a naive implementation we could simply use a SPARQL query to count the number of triples in the named graph.

However, unfortunately, in our experiments, such a simple implementation solely based on "onboard" means of the SPARQL engines turned out to be infeasible due to performance reasons. Thus, for the time being, we resorted to just evaluating one iteration of all generated rules, in order to evaluate our conjecture that rules improve our prediction results.

Eventually, we may need to resort to (offline) using a native rule engine. Indeed, in practical applications such rule/datalog engines have shown to perform better than recursive views implemented directly on top of databases in the past for instance for computing RDFS closure, cf. [38]. For the moment, we leave this to future work and resort, as mentioned, to a fixed number of iterations of rule applications.

7. Evaluation

The OCDP runs distributed over several systems. The data wrappers are distributed over several machines and collected by the crawler on `http://kalmar32.fzi.de/`. The crawl file is then regularly downloaded and inserted on the OCDP server, which is a virtual machine with 2 assigned cores with 2.6GHz and 16GB main memory running 64bit Ubuntu Linux. The OCDP server then performs the global cube materialisation. The missing value prediction is performed asynchronously on a workstation. Eventually the QB equations are evaluated again by the OCDP server.

7.1. Crawling and Global Cube Materialisation

We require a considerable amount of memory for Linked Data-Fu (our machine is a virtual machine with Intel(R) Xeon(R) CPU E5520 @ 2.27GHz and 8GB RAM), thus we split the program into two separate programs, one for crawling Eurostat and another one for UN-Data. The crawling takes 60 minutes for Eurostat and 30 minutes for UN-Data and results in a merged file with > 7 M triples and > 1 M observations.

The queries given in the implementation of Section 4.4 behave in practise as follows:

1. The SPARQL query to tag all the relevant datasets tags 32 datasets,
2. the SPARQL query to ask for all the schemas returns returns three variants,
3. the SPARQL queries for these three dimension combinations to materialise the global cube need altogether 8 minutes, and
4. the two SPARQL queries to perform the entity consolidation for indicators and cities and take 78 seconds.

From the crawl we materialise 936k observations (7M triples) into the global cube in 15 minutes.

7.2. Predicting Missing Values

In Chapter 5, we evaluated two approaches for filling-in the missing values. Approach 1 uses indicators directly as predictors, but requires complete subsets of the matrices In Approach 2, we perform PCA to reduce the number of dimensions, which allows us to impute all the missing values with neutral values for the PCA. Since, we will only apply a single approach, we have two quality measurements to evaluate them:

[49] `http://citydata.wu.ac.at/ocdp/eurostat-equations.rdf`
[50] `http://citydata.wu.ac.at/ocdp/eurostat-rules.rdf`
[51] `https://github.com/openlink/virtuoso-opensource/issues/317`

1. It is important to build models which are able to predict many (preferably all) missing values.
2. Second, the prediction accuracy of the models is essential, so that the Open City Data Pipeline can fulfil its purpose of publishing high-quality, accurate data and predictions.

Prediction accuracy is slightly higher with 0.25% in Approach 1 than in Approach 2 with 0.55%, which we relate to the smaller size of the dataset. However in Approach 1, we fill-in only 296 069 values with 2 predictors (median RMSE% of 2.09%). Note that with more predictors we filled-in even less missing values. This is about 42% of the 699 390 values in Approach 2.

Summarizing, despite the good RMSE%, of Approach 1, a major drawback is the reduced number of predictions, hence we will apply Approach 2 as the input for the following steps and for publishing the filled-in missing value.

7.3. QB Equations

Section 6.3 described the implementation of QB equations in the OCDP. In this section we give some results about the behaviour of the QB equations part of the OCDP system[52] and some evaluation of the QB evaluation themselves and how they improve the results of the whole OCDP.

Normalisation. The normalisation to generate QB rules from QB equations took 19 seconds to normalise 61 QB equations from Eurostat into 274 QB rules.[53]

AppendixC contains a complete example QB equation from Eurostat and one of the normalised QB rules.

Create SPARQL queries. First we filter out 104 QB rules for which at least one input variable matches no existing observation. Such rules can never deliver any results and evaluating them is useless. Note that due to another bug in the Virtuoso cost estimation we could not evaluate the 34 QB rules containing seven or more input variables. Eventually we created 129 SPARQL CONSTRUCT queries in five seconds.

AppendixC shows a complete example of a SPARQL CONSTRUCT query together with the corresponding QB rule.

Evaluate queries in rule engine. This one iteration of evaluating all generated 129 SPARQL CONSTRUCTs took 24 minutes and inserted 1M observations (21M triples) into the global cube.

Observation evaluation. As we can demonstrate in even such an incomplete materialization of equations allows us to predict a significant amount of new or improved observations, that could not be predicted with equal accuracy with solely the methods described in Section 5:

Out of 1M newly generated observations from rules, 32k are fresh observations, i.e., observations with a dimension-instance combination where no previous predicted values existed.

The SPARQL queries could generate 180k observations (for 29 indicators) containing propagated error estimates which are better than the error estimates of the corresponding prediction observations. These 180k observations thus improve the estimations of the missing value prediction.

Additionally, we verified the computed values against observed values from the crawled sources where possible. That is, we computed the RMSE per indicator (we have comparable data for 6 of these 29 indicators) comparing the computed values through equations against those values where an original source observation existed. In all these cases, the average error was lower than the best RMSE computed through the regression methods from Section 7.2, which shows that our combined approach effectively improves the missing value predictions.

We could show that extending the approach of using statistical missing value prediction methods with equational knowledge can improve the quality of missing value prediction.

8. Related Work

We begin the review of related work with approaches related to the handling of numerical values in databases. We only review a small selection of the large related work in the area of databases, because our focus is on data integration using web technologies. As such, we use technologies such as RDF [39], RDFS, OWL and SPARQL 1.1 (both query language [40] and update language [41]) to represent, query and integrate multidimensional data. We assume the reader is familiar with these standards. After we cover RDF data pipelines for data access and integration, we survey vocabularies for modelling and representing multidimensional data in RDF, followed by methods for representing equations. Finally, we deal with work on imputation from the field of statistics, which is concerned with substituting missing values in numerical data.

8.1. Numerical Data in Databases

Siegel et al. [42] introduce the notion of semantic values – numeric values accompanied by metadata for interpreting the value, e.g., the unit – and propose conversion functions to facilitate the exchange of distributed datasets by heterogeneous information systems.

Diamantini et al. [43] suggest to uniquely define indicators (measures) as formulas, aggregation functions, semantics (mathematical meaning) of the formula, and recursive references to other indicators. They use mathematical standards for describing the semantics of operations (MathML, OpenMath) and use Prolog to reason about indicators, e.g., for equality or consistency of indicators. In contrast, we focus on heterogeneities occurring in terms of dimensions and members, and allow conversions and combinations.

As a basis for sharing and integration of numerical data, XML is often used [44]. XML standards such as XCube fulfil requirements for sharing of data cubes [45] such as the conceptual model of data cubes, the distinction of data (observations) and metadata (dimensions, measures), a network-transportable data format, support for linking and inclusion concepts, extensibility, conversion capability and OLAP query functionality. Other

[52] for more details see http://citydata.wu.ac.at/ocdp/import

[53] the Eurostat indicator definition for the population change over 1 year is not expressible in QB equations

advantages include that XML allows to define a schema (XML Schema), there are data modification and query languages for XML such as XSLT and XQuery, and there are widely-used XML schemas for representing specific information, e.g., XBRL for financial reports, SDMX for statistics, DDI[54] for research studies. Another XML-based exchange standard for ETL transformations and data warehouse metadata is the Common Warehouse Metamodel (CWM)[55] by the Object Management Group (OMG).

However, the integration of data across different standards is still an open issue. CWM – but also other interfaces and protocols to share multidimensional datasets such as XML for Analysis and OLE DB – lack a formal definition making it more difficult to use such formalism as a basis for integration [46]. XML schemas are concerned with defining a syntactically valid XML document representing some specific type of information. Yet, XML schemas do not describe domain models; without formal domain models, it is difficult to derive semantic relationships between elements from different XML schemas [47]. Often, the domain model for an XML schema is represented in a semi-formal way using UML documents and free text. In contrast, schemas described as an OWL or RDFS ontology such as QB have a formal domain model based on logics.

Conceptually, we distinguish the global-as-view (GAV, also known as source-based integration) approach of data integration where the global schema is represented in terms of the data sources and the local-as-view (LAV) approach that requires sources to be defined as views over the global schema [48–50]. We use the GAV approach and define the global cube in terms of single data cubes using the drill-across operation. With GAV, queries over the global schema can easily be translated to queries over the data sources [49]. The advantage of LAV is that the global schema does not need to change with the addition of new data sources. The advantage of GAV is that queries over the global schema can easily be translated to queries over the data sources.

8.2. RDF Data Pipelines

Within the Semantic Web community there is extensive work around triplification and building data pipelines and Linked Data wrappers for publicly available data sources on the web, where for instance the LOD2 project has created and promoted a whole stack of tools to support the lifecycle of Linked Data, i.e. creating maintainable and sustainable mappings/wrappers of existing data sources to RDF and Linked Data, a good overview is provided in the book chapter by Auer et al. [51, 52]. All this work could likewise be viewed as an application of the classical ETL (Extract-Transform-Load) [53] methodology extended to work on the web, based on open standards and Linked Data principles [14]. Our work is not much different in this respect, with the difference that we apply a tailored architecture for a set of selected sources around a focused topic (city data), where we believe that a bespoke combination of rule-based reasoning methods in combination with statistical machine learning can provide added value in terms of data enrichment. This is a key difference to the above-mentioned methods that rather focus on entity linkage and object consolidation in terms of semantic enrichment. However, this focused approach is also different from generic methods for reasoning over Linked Data on the web (cf. e.g. [54] and references therein for an overview), solely based on OWL and RDFs which (except very basic application of `owl:sameAs` (for consolidating different city identifiers across sources) and `rdfs:subPropertyOf` reasoning (for combining overlapping base indicators occurring within different sources).

Other work tries to automatically derive new from existing data. Ambite and Kapoor [55] present Shim Services providing operations for accessing remote data, integrating heterogeneous data, and deriving new data. Workflows of operations are automatically created based on semantic descriptions of operators. Subsumption reasoning is included to match inputs of services to outputs of other services. To avoid the infinite execution of operations, a limit is defined to the depth of nested operations of the same type. In contrast to the automatically constructed workflows, our pipeline consists of a fixed set of processing steps. Instead of "shim services" that act as standalone components accessible via the network, we base the computation on local formulas and use a vocabulary to represent the formulas.

8.3. Data Modelling and Representation

We have chosen the Data Cube Vocabulary (QB) [4] as basis for representing statistical data (both for the individual data points and the "metadata", the structure definitions). QB is one among several vocabularies available to publish raw or aggregated multidimensional datasets. For instance, there are various OWL ontologies available for representing multidimensional datasets [56]. Also, several light-weight ontologies have been proposed, such as SCOVO [57] and SCOVOLink [20]. Other vocabularies for statistical data are the DDI RDF Vocabularies,[56] several vocabularies inspired by the Data Documentation Initiative, and the StatDCAT application profile[57] (StatDCAT-AP) to express in a structured way the metadata of statistical datasets which are currently published by the different agencies in the European Union. In comparison to these approaches, we see the following reasons for choosing QB: QB is widely-adopted, which is an important factor for data integration use cases, as sources already represented in QB can be integrated more easily than sources in other representations. Further, QB promises flexibility and efficiency benefits in various use cases [58], and exhibits the necessary complexity to allow efficient and flexible integration and analysis of statistical datasets.

8.4. Modelling of Equations

Modelling the actual numerical data and the structure of that data captures only a part of the knowledge around statistical data

[54] http://www.ddialliance.org/
[55] http://www.omg.org/spec/CWM/
[56] http://www.ddialliance.org/Specification/RDF
[57] https://www.europeandataportal.eu/de/content/statdcat-ap-wg-virtual-meeting

that can be represented in a machine-interpretable manner. Equations in particular are a rich source of knowledge in statistical data. Lange [59] gives an extensive overview of representations of mathematical knowledge for the Semantic Web. We first cover representation of equations for layout purposes, and then cover representations that permit the interpretation of the formulas by machines.

Non-RDF based representations of mathematical knowledge include MathML [60] and OpenMath [61] and use XML for serialisation and focus more on a semantic representation of mathematical entities and are not directly useful for reasoning. Although GeoSPARQL [62] uses MathML in XMLLiterals and OpenMath provides preliminary integration into RDF,[58] these representations are hard to reuse for RDF tools and still are not suitable for an RDF QB setup.

The OWL ontologies QUDT [31], OM [63], and SWEET [64] provide means to describe units and to some extent model conversion between these units, but do not specify a concrete machinery to perform these conversions. Our approach is orthogonal to these efforts in that it provides not only a modelling tool for unit conversions but more general equations and also gives a semantics to automatically infer new values.

Semantic Web rule languages and systems often implement numerical functions – for example RIF uses numerical functions from XPath.[59] Other examples for rule languages and systems include SWRL and Apache Jena rules. Converting equations to rules naively can lead to a set of recursive rules which often lead to non-termination even for one equation alone (cf. [8]).

To add reasoning over numbers Description Logics were extended with *concrete domains* (cf. [65]). A concrete domain is a domain separate from the usual instance domain of the model based semantics. Examples for concrete domains include different sets of numbers or strings. A specific concrete domain extension defines predicates over the concrete domain, e.g., *greater than* for numbers, or *substring* for strings. Often also a limited set of functions (for computation) can be supplied. Racer [66] implements concrete domains with numbers. But computed values are only used during reasoning and are not available to the user afterwards. OWL Equations [67], a concrete domain extension carried over to OWL, allows comparing numerical values – even computed values; still the same limitations apply.

8.5. Missing Value Imputation

Several books provide information on handling missing values from the perspective of statistics as well as from social sciences, e.g. cf. [68, 69]. Within the Semantic Web community, a main focus on value completion has been in the prediction of generic relations, and mainly object relations (i.e. link-prediction) on the object level rather than on numerical values, cf. [70] for an excellent survey on such methods. The usage of numerical values is a a rather recent topic in this respect. Along these lines, but complementary to the present work, Neumaier et al. [71] (as well as similar works referenced therein) have discussed methods to assign bags of numerical values to property-class pairs in knowledge graphs like DBpedia (tailored to finding out relations such that for instance a certain set of numbers could possibly be "population numbers of cities in France"), but not specifically to complete/impute missing values. Our method rather uses fairly standard, robust, and well-known methods (KNN, linear regression, and random forrest) for numerical missing value imputation based on principle components [7]. This could be certainly refined to more tailored methods in the future, for instance using time-series analysis; indeed our predicted values, while resonably realistic in the value ranges often show some non-realistic "jumps" when raw data for a certain indicator is available over a certain sequence of years, but missing for only a few years in between. Since the missing value imputation component in our architecture is modularly extensible with new/refined methods, such refinements could be added as future work.

9. Conclusions

In this paper we have presented the *Open City Data Pipeline*, an extensible platform for collecting, integrating, and enriching open city data from several data providers including Eurostat and UNSD. We have developed several components including wrappers, a data crawler, an ontology-based integration platform, and a missing value prediction module, which relies on both statistical regression methods as well as ontological inference over equational background knowledge: since we deal with very sparse datasets, the prediction (or, as it is often referred to, imputation) of missing values is a crucial component. For this, we have developed two approaches, one based on basic regression methods, and one based on exploiting known equations.

As for the former, we both predict target indicators directly from other indicators, if available, and if the available data is too sparse rely on predictors from components calculated by Principal Components Analysis (PCA). We applied for both approaches three basic regression methods and selected the best performing one.

As for the latter, we have shown that the predictions computed this way can be further improved by exploiting equations, where we have estimated and verified the assumption that this combination improves prediction accuracy overall, in terms of the number of filled-in values and estimated errors.

The created prediction values are fed back into our triple store and are accessible via our SPARQL endpoint or Web UI. Here, we additionally publish provenance information including the used prediction methods and equations along with estimated prediction accuracy.

9.1. Lessons Learnt & Future work

In the wrapper component, integrating cities and indicators for a new dataset (often CSV tables) is still a slow manual process and needs custom scripting. Particularly, entity recognition for cities can only partially be automated and needs manual adaption for each wrapper. Also, mapping indicators is still a

[58] http://www.openmath.org/cd/contrib/cd/rdf.xhtml

[59] https://www.w3.org/TR/2007/REC-xpath-functions-20070123/#op.numeric

largely manual process, where in the future, we plan to also apply instance based mapping learning techniques used in ontology matching (cf. [72]). We emphasize here, that in fact such approaches could rely on and extend similar regression techniques as we used for imputing missing values.

As for our enrichment approach (combining missing value prediction techniques and equations), we have improved over the past 1 1/2 years significantly [10], not only refining our methods, but also by proving the conjecture that the more data we collect, the better the predictions actually get: by applying the PCA-based prediction approach, using *standard* regression techniques without customization, we reach a good quality for predictions (overall RMSE% of 0.55%) and are able to fill large gaps of the missing values, which can be further improved by the combination with equational knowledge.

The republication of our enriched dataset as Statistical Linked Data [3], using the standard QB format shall allow to combine and integrate our dataset with other datasets of the Global Cube [9].

Our future work includes extensions of the presented datasets, methods, and the system itself. Regarding the datasets, we already mention several potential additional data sources that could be included, e.g., the Carbon Disclosure Project,[60] or the World Council for City Data (WCCD), formerly Global City Indicator Framework (GCIF) which cover a wider range of cities worldwide, but aren't yet providing Open Data. Apart from that, we could include sources like DBpedia or Wikidata in a more timely fashion, e.g. recording changes in values, through projects such as the DBpedia wayback machine [22], to collect also historical data from DBpedia.

Compared to [10] we have completely refurbished our crawling framework and architecture to automatically and regularly update the integrated sources dynamically. We therefore expect new lessons learnt from more regular updates; e.g.; Eurostat only since very recently updates its datasets monthly, instead of annually only,[61] which we expect to benefit from.

We also plan to connect our plattform to the Linked Geo Data Knowledge Base [73] including OpenStreetMap (OSM) data. Based on such data, new indicators could be directly calculated, e.g., the size of public green space by aggregating all the parks. Some preliminary works on integrating indicators extracted from OSM with our Open City Data Pipeline have been presented in [74].

As we integrate more sources and datasets, another future direction we should pursue is to revisit cross-dataset predictions of missing values in more detail,[62] i.e., how predictions can be made from one data source to another. This is particularly important, as typically different data sources have only a handfull (if any) of overlapping indicators.

We further aim to extend our basket of base regression methods with other well established methods. Promising candidates are Support Vector Machines [75], Neural Networks, and Bayesian Generalized Linear Models [76].

Moreover, we plan to publish more details on the best regression method per indicator as part of our ontology: so far, we only indicate the method and estimated RMSE%, whereas further details such as used parameters and regression models would be needed to reproduce and optimize our predictions. Ontologies such as [29] could serve as a starting point here.

Furthermore, we are in the process of improving the user interface to make the Web application easier to use. For this we investigate several libraries for more advanced information visualization.

Last, but not least, in many components of our pipeline we deliberately relied on off-the-shelf SPARQL engines. This design choice should hopefully allow us to benefit from further improvements of these engines: for instance, we have experienced scalability limitations in the scalability of update queries for implementing inferences through QB equations or also for materializing provenance annotations in the Global Cube. We shall explore optimizations, test other SPARQL stores, or combine our approach with e.g. custom rule engines, in the future, to improve performance of our data pipeline.

[60] https://www.cdp.net/de

[61] cf. Section 8.1 at http://ec.europa.eu/eurostat/cache/metadata/de/urb_esms.htm: "From 2017 new data will be published the first day of every month."

[62] Some preliminary work along these lines have been presented in [10].

References

[1] Economist Intelligence Unit (Ed.), The Green City Index, Siemens AG, 2012.

[2] S. Neumaier, J. Umbrich, A. Polleres, Automated quality assessment of metadata across open data portals, ACM Journal of Data and Information Quality (JDIQ) 8 (1) (2016) 2.

[3] B. Kämpgen, S. O'Riain, A. Harth, Interacting with Statistical Linked Data via OLAP Operations, in: 9th Extended Semantic Web Conference (ESWC) Satellite Events, 2012.

[4] R. Cyganiak, D. Reynolds, J. Tennison, The RDF data cube vocabulary, W3C Recommendation, W3C (Jan. 2014).

[5] S. Stadtmüller, S. Speiser, A. Harth, R. Studer, Data-Fu : A Language and an Interpreter for Interaction with Read / Write Linked Data, in: 22nd International Conference on World Wide Web (WWW), 2013.

[6] L. M. A. Bettencourt, J. Lobo, D. Helbing, C. Kühnert, G. B. West, Growth, innovation, scaling, and the pace of life in cities, Proc. of the National Academy of Sciences of the United States of America 104 (17) (2007) 7301–7306.

[7] S. T. Roweis, EM algorithms for PCA and SPCA, in: Advances in Neural Information Processing Systems 10 (NIPS 1997), 1997, pp. 626–632.

[8] S. Bischof, A. Polleres, RDFS with Attribute Equations via SPARQL Rewriting, in: Proc. of ESWC 2013, Springer, 2013, pp. 335–350.

[9] B. Kämpgen, S. Stadtmüller, A. Harth, Querying the Global Cube: Integration of Multidimensional Datasets from the Web, in: 19th International Conference on Knowledge Engineering and Knowledge Management (EKAW), 2014.

[10] S. Bischof, C. Martin, A. Polleres, P. Schneider, Collecting, integrating, enriching and republishing open city data as linked data, in: The Semantic Web - ISWC 2015 - 14th International Semantic Web Conference, Bethlehem, PA, USA, October 11-15, 2015, Proceedings, Part II, 2015, pp. 57–75.

[11] B. Kämpgen, T. Weller, S. O'Riain, C. Weber, A. Harth, Accepting the XBRL Challenge with Linked Data for Financial Data Integration, in: 11th Extended Semantic Web Conference (ESWC), 2014.

[12] B. Kämpgen, Flexible integration and efficient analysis of multidimensional datasets from the web, Ph.D. thesis, KIT, Fakultät für Wirtschaftswissenschaften (Feb. 2015).

[13] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, H. Pirahesh, Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals, Data Mining and Knowledge Discovery 1 (1).

[14] T. Berners-Lee, Linked Data, W3C Design Issues, from http://www.w3.org/DesignIssues/LinkedData.html; retr. 2010/10/27 (July 2006).
[15] B. Kämpgen, A. Harth, No Size Fits All – Running the Star Schema Benchmark with SPARQL and RDF Aggregate Views, in: 10th Extended Semantic Web Conference (ESWC), 2013.
[16] T. Lebo, D. McGuinness, S. Sahoo, PROV-O: The PROV ontology, W3C recommendation, W3C, http://www.w3.org/TR/2013/REC-prov-o-20130430/ (Apr. 2013).
[17] I. H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, 3rd Edition, Morgan Kaufmann Publishers Inc., 2011.
[18] J. Han, Data Mining: Concepts and Techniques, 3rd Edition, Morgan Kaufmann Publishers Inc., 2012.
[19] A. Polleres, A. Hogan, R. Delbru, J. Umbrich, RDFS & OWL reasoning for linked data, in: Reasoning Web. Semantic Technologies for Intelligent Data Access (Reasoning Web 2013), Vol. 8067 of LNCS, Springer, Mannheim, Germany, 2013, pp. 91–149.
[20] D. Vrandečić, C. Lange, M. Hausenblas, J. Bao, L. Ding, Semantics of Governmental Statistics Data, in: Web Science Conference (WebSci), 2010.
[21] C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, S. Hellmann, DBpedia - A crystallization point for the web of data, J. Web Sem. 7 (3) (2009) 154–165. doi:10.1016/j.websem.2009.07.002.
[22] J. D. Fernández, P. Schneider, J. Umbrich, The dbpedia wayback machine, in: Proceedings of the 11th International Conference on Semantic Systems, (SEMANTICS 2015), Vienna, Austria, 2015, pp. 192–195. doi:10.1145/2814864.2814889.
[23] Office for Official Publications of the European Communities, Urban Audit. Methodological Handbook (2004).
[24] U.S. Census Bureau, County and City Data Book 2007, https://www.census.gov/compendia/databooks/ (2007).
[25] H. Paulheim, J. Fürnkranz, Unsupervised generation of data mining features from linked open data, in: Proc. of WIMS 2012, ACM, 2012, p. 31.
[26] R. Isele, J. Umbrich, C. Bizer, A. Harth, LDspider: An Open-Source Crawling Framework for the Web of Linked Data, in: 9th International Semantic Web Conference (ISWC) Posters and Demos, 2010.
[27] L. B. Statistics, L. Breiman, Random forests, in: Machine Learning, 2001, pp. 5–32.
[28] R Development Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, 2009.
[29] C. M. Keet, A. Lawrynowicz, C. d'Amato, A. Kalousis, P. Nguyen, R. Palma, R. Stevens, M. Hilario, The data mining OPtimization ontology, Web Semantics: Science, Services and Agents on the World Wide Web 32 (0) (2015) 43 – 53.
[30] D. Esteves, D. Moussallem, C. B. Neto, T. Soru, R. Usbeck, M. Ackermann, J. Lehmann, MEX vocabulary: a lightweight interchange format for machine learning experiments, in: Proceedings of the 11th International Conference on Semantic Systems, SEMANTICS 2015, Vienna, Austria, September 15-17, 2015, 2015, pp. 169–176.
[31] P. J. K. Ralph Hodgson, Qudt - quantities, units, dimensions and data types in owl and xml (2011).
URL http://www.qudt.org/
[32] A. Polleres, F. Scharffe, R. Schindlauer, SPARQL++ for mapping between RDF vocabularies, in: OTM 2007, Part I : Proceedings of the 6th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE 2007), Vol. 4803 of LNCS, Springer, Vilamoura, Algarve, Portugal, 2007, pp. 878–896.
[33] T. Eiter, G. Ianni, R. Schindlauer, H. Tompits, A uniform integration of higher-order reasoning and external evaluations in answer-set programming, in: Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05), Edinburgh, Scotland, UK,, 2005, pp. 90–96.
[34] A. Polleres, J. Wallner, On the relation between SPARQL1.1 and answer set programming, Journal of Applied Non-Classical Logics (JANCL) 23 (1–2) (2013) 159–212, special issue on Equilibrium Logic and Answer Set Programming.
[35] R. Angles, C. Gutierrez, The expressive power of sparql, in: International Semantic Web Conference (ISWC 2008), Vol. 5318 of LNCS, Springer, Karlsruhe, Germany, 2008, pp. 114–129.
[36] P. R. Bevington, D. K. Robinson, Data Reduction and Error Analysis for the Physical Sciences, 3rd Edition, McGraw-Hill, Boston, 2003.
[37] H. H. Ku, Notes on the use of propagation of error formulas, Journal of Research of the National Bureau of Standards 70 (4).
[38] G. Ianni, A. Martello, C. Panetta, G. Terracina, Efficiently querying RDF(S) ontologies with answer set programming, J. Log. Comput. 19 (4) (2009) 671–695.
[39] G. Schreiber, Y. Raimond, F. Manola, E. Miller, B. McBride, Rdf 1.1 primer, W3C working group note, W3C (Jun. 2014).
URL https://www.w3.org/TR/rdf11-primer/
[40] S. Harris, A. Seaborne, Sparql 1.1 query language, W3C recommendation, W3C (Mar. 2013).
URL http://www.w3.org/TR/sparql11-query/
[41] P. Gearon, A. Passant, A. Polleres, Sparql 1.1 update, W3C recommendation, W3C (Mar. 2013).
URL http://www.w3.org/TR/sparql11-update/
[42] M. Siegel, E. Sciore, A. Rosenthal, Using Semantic Values to Facilitate Interoperability Among Heterogeneous Information Systems, ACM Transactions on Database Systems (TODS) 19 (2).
[43] C. Diamantini, D. Potena, E. Storti, A Logic-Based Formalization of KPIs for Virtual Enterprises, in: Advanced Information Systems Engineering Workshops, 2013.
[44] J. M. Pérez, R. B. Llavori, M. J. Aramburu, T. B. Pedersen, Integrating data warehouses with web data: A survey, IEEE Trans. Knowl. Data Eng. 20 (7) (2008) 940–955. doi:10.1109/TKDE.2007.190746.
[45] W. Hümmer, A. Bauer, G. Harde, XCube – XML for Data Warehouses, in: 6th ACM International Workshop on Data Warehousing and OLAP (DOLAP), 2003.
URL http://portal.acm.org/citation.cfm?id=956060.956067
[46] P. Vassiliadis, T. Sellis, A Survey of Logical Models for OLAP Databases, ACM SIGMOD Record 28 (4).
[47] M. Klein, D. Fensel, F. van Harmelen, I. Horrocks, The relation between ontologies and schema-languages, LinkÃűping Electronic Articles in Computer and Information Science 6 (4).
[48] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, R. Rosati, Data Integration in Data Warehousing, International Journal of Cooperative Information Systems 10 (3).
[49] A. Cal, D. Calvanese, G. D. Giacomo, M. Lenzerini, Data Integration Under Integrity Constraints, Information Systems 29 (2).
[50] M. R. Genesereth, Data Integration: The Relational Logic Approach, Morgan & Claypool Publishers, San Rafael, California (USA), 2010. doi:10.2200/S00226ED1V01Y200911AIM008.
[51] A. N. Ngomo, S. Auer, J. Lehmann, A. Zaveri, Introduction to linked data and its lifecycle on the web, in: Reasoning Web. Reasoning on the Web in the Big Data Era - 10th International Summer School 2014, Athens, Greece, September 8-13, 2014. Proceedings, 2014, pp. 1–99.
[52] S. Auer, L. Bühmann, C. Dirschl, O. Erling, M. Hausenblas, R. Isele, J. Lehmann, M. Martin, P. N. Mendes, B. V. Nuffelen, C. Stadler, S. Tramp, H. Williams, Managing the life-cycle of linked data with the LOD2 stack, in: The Semantic Web - ISWC 2012 - 11th International Semantic Web Conference, Boston, MA, USA, November 11-15, 2012, Proceedings, Part II, 2012, pp. 1–16.
[53] M. Golfarelli, S. Rizzi, Data Warehouse Design: Modern Principles and Methodologies, McGraw-Hill, 2009.
[54] A. Hogan, Exploiting RDFS and OWL for Integrating Heterogeneous, Large-Scale, Linked Data Corpora, Ph.D. thesis, Digital Enterprise Research Institute, National University of Ireland, Galway, available from http://aidanhogan.com/docs/thesis/ (2011).
[55] J. L. Ambite, D. Kapoor, Automatically Composing Data Workflows with Relational Descriptions and Shim Services, in: International Semantic Web Conference (ISWC), 2007.
[56] M. Niinimäki, T. Niemi, An ETL process for OLAP using RDF/OWL ontologies, J. Data Semantics 13 (2009) 97–119. doi:10.1007/978-3-642-03098-7_4.
[57] M. Hausenblas, W. Halb, Y. Raimond, L. Feigenbaum, D. Ayers, SCOVO: Using Statistics on the Web of Data, in: 6th European Semantic Web Conference (ESWC), 2009.
URL http://dblp.uni-trier.de/db/conf/esws/eswc2009.html#HausenblasHRFA09
[58] B. Kämpgen, R. Cyganiak, Use Cases and Lessons for the Data Cube Vocabulary, Working Group Note – http://www.w3.org/TR/2013/NOTE-vocab-data-cube-use-cases-20130801/, W3C, USA (Aug 2013).
[59] C. Lange, Ontologies and languages for representing mathematical know-

ledge on the semantic web, Semantic Web 4 (2) (2013) 119–158.
[60] R. R. Miner, D. Carlisle, P. D. F. Ion, Mathematical markup language (MathML) version 3.0 2nd edition, W3C recommendation, W3C, http://www.w3.org/TR/2014/REC-MathML3-20140410/ (Apr. 2014).
[61] S. Buswell, O. Caprotti, D. P. Carlisle, M. C. Dewar, M. Gaetano, M. Kohlhase, The open math standard, Tech. rep., version 2.0. Technical report, The Open Math Society, 2004. http://www.openmath.org/standard/om20 (2004).
[62] M. Perry, J. Herring, Ogc geosparql-a geographic query language for rdf data, OGC Implementation Standard. Sept.
[63] H. Rijgersberg, M. van Assem, J. Top, Ontology of units of measure and related concepts, Semantic Web 4 (1) (2013) 3–13.
[64] R. G. Raskin, M. J. Pan, Knowledge representation in the semantic web for earth and environmental terminology (sweet), Computers & geosciences 31 (9) (2005) 1119–1125.
[65] F. Baader, The description logic handbook: Theory, implementation and applications, Cambridge university press, 2003.
[66] V. Haarslev, R. Möller, Description of the racer system and its applications., Description Logics 49.
[67] B. Parsia, U. Sattler, Owl 2 web ontology language data range extension: Linear equations, W3C working group note, W3C, https://www.w3.org/TR/owl2-dr-linear/ (Dec. 2012).
[68] E. R. Buhi, P. Goodson, T. B. Neilands, Out of sight, not out of mind: strategies for handling missing data, American journal of health behavior 32 (1) (2008) 83–92.
[69] F. S. Switzer, P. L. Roth, Coping with missing data, in: Handbook of Research Methods in Industrial and Organizational Psychology, Blackwell Publishing Ltd, 2008, pp. 310–323.
[70] H. Paulheim, Knowledge graph refinement: A survey of approaches and evaluation methods, Semantic Web 8 (3) (2017) 489–508.
[71] S. Neumaier, J. Umbrich, J. Parreira, A. Polleres, Multi-level semantic labelling of numerical values, in: Proceedings of the 15th International Semantic Web Conference (ISWC 2016) - Part I, Vol. 9981 of LNCS, Springer, Kobe, Japan, 2016, pp. 428–445.
[72] J. Euzenat, P. Shvaiko, Ontology matching, 2nd Edition, Springer, 2013.
[73] C. Stadler, J. Lehmann, K. Höffner, S. Auer, LinkedGeoData: A core for a web of spatial open data, Semantic Web 3 (4) (2012) 333–354.
[74] M. Posada-Sánchez, S. Bischof, A. Polleres, Extracting geo-semantics about cities from openstreetmap, in: Proceedings of the Posters and Demos Track of the 12th International Conference on Semantic Systems (SEMANTiCS2016), Leipzig, Germany, 2016.
[75] V. Sanchez, Advanced support vector machines and kernel methods, Neurocomputing 55 (1–2) (2003) 5–20.
[76] M. West, P. J. Harrison, H. S. Migon, Dynamic generalized linear models and bayesian forecasting, Journal of the American Statistical Association 80 (389) (1985) 73–83.

AppendixA. Global Cube Materialisation

```
INSERT INTO GRAPH <http://citydata.wu.ac.at/qb-materialised
    -global-cube> {
?obs qb:dataSet globalcube:global-cube-ds.
?obs dcterms:publisher ?ds.
?obs dcterms:date ?year.
?obs sdmx-dimension:refArea ?city.
?obs cd:hasIndicator ?indicator.
?obs sdmx-dimension:sex ?sex.
?obs <http://ontologycentral.com/2009/01/eurostat/ns#unit>
    ?unit.
?obs sdmx-dimension:age ?age.
?obs sdmx-measure:obsValue ?value.
}
WHERE { GRAPH <http://citydata.wu.ac.at/qb> {
# Considering all dimension property URIs
?yearDimension1 owl:equivalentProperty <http://purl.org/dc/
    terms/date>.
?cityDimension1 owl:equivalentProperty <http://purl.org/
    linked-data/sdmx/2009/dimension#refArea>.
?indicatorDimension1 owl:equivalentProperty <http://
    citydata.wu.ac.at/ns#hasIndicator>.
# in our global cube for OCDP we assume date, refArea,
    indicator as compulsory; sex, unit, age as optional
OPTIONAL { ?sexDimension2 owl:equivalentProperty sdmx-
    dimension:sex. }
bind(coalesce(?sexDimension2, sdmx-dimension:sex) as ?
    sexDimension1)
OPTIONAL { ?unitDimension2 owl:equivalentProperty <http://
    ontologycentral.com/2009/01/eurostat/ns#unit>. }
BIND(coalesce(?unitDimension2, <http://ontologycentral.com
    /2009/01/eurostat/ns#unit>) AS ?unitDimension1)
OPTIONAL { ?ageDimension2 owl:equivalentProperty sdmx-
    dimension:age. }
BIND(coalesce(?ageDimension2, sdmx-dimension:age) AS ?
    ageDimension1)

?obs qb:dataSet ?ds.
?ds qb:structure ?dsd.

# Querying the observations
?obs ?yearDimension1 ?year.
?obs ?cityDimension1 ?city1.
?obs ?indicatorDimension1 ?indicator1.
OPTIONAL { ?obs ?sexDimension1 ?sex1. }
BIND(coalesce(?sex1, globalcube:ALL-value) AS ?sex)
OPTIONAL { ?obs ?unitDimension1 ?unit1. }
BIND(coalesce(?unit1, globalcube:ALL-value) AS ?unit)
OPTIONAL { ?obs ?ageDimension1 ?age1. }
BIND(coalesce(?age1, globalcube:ALL-value) AS ?age)
?obs sdmx-measure:obsValue ?value.

?city1 owl:sameAs ?city .
?city a <http://dbpedia.org/ontology/City>.

?indicator1 owl:sameAs ?indicator.
?indicator a <http://citydata.wu.ac.at/ns#Indicator>.

# Only those datasets that do not show other dimensions
FILTER NOT EXISTS { ?dsd qb:component ?acomp. ?acomp qb:
    dimension ?otherdim.
FILTER(?otherdim != dcterms:date
&& ?otherdim != estatwrap:cities && ?otherdim != sdmx-
    dimension:refArea && ?otherdim != estatwrap:geo && ?
    otherdim != estatwrap:metroreg
&& ?otherdim != estatwrap:indic_ur && ?otherdim != cd:
    hasIndicator
&& ?otherdim != estatwrap:unit && ?otherdim != cd:unit
&& ?otherdim != estatwrap:sex && ?otherdim != sdmx-
    dimension:sex
&& ?otherdim != estatwrap:age && ?otherdim != sdmx-
    dimension:age
)}}}
```

AppendixB. SPARQL INSERT query for QB rule

The following SPARQL INSERT query is the result of converting the QB rule from Example 6.4. We assume the referenced dataset `ex:ds` contains the dimensions `sdmx-dimension:refArea`, `ex:indic`, and, as used by the QB rule `ex:unitDimension`.

```
INSERT {
  ?obs qb:dataSet ex:ds ;
    sdmx-dimension:refArea ?city ;
    dcterms:date ?year ;
    ex:indic ?indic ;
    ex:unitDimension qudt-unit:MileInternational ;
    sdmx-measure:obsValue ?value ;
    a prov:Entity ;
    prov:wasDerivedFrom ?km_obs ;
    prov:wasGeneratedBy ?activity ;
    prov:generatedAtTime ?now ;
    ex:error ?error .

  ?activity a prov:activity ;
  prov:qualifiedAssociation [
    a prov:Association ;
    prov:wasAssociatedWith   ex:fred ;
    prov:hadPlan ex:unit-km-mile-km ] .
}
WHERE {
  ?km_obs qb:dataSet ex:ds ;
    sdmx-dimension:refArea ?city ;
    dcterms:date ?year ;
    ex:indic ?indic ;
    sdmx-e:unitMeasure qudt-unit:Kilometer ;
    sdmx-measure:obsValue ?km ;
    s ex:error ?km_error_1 .

  BIND(IRI(CONCAT("http://example.com/unit-km-mile-km", "
       _obs_", MD5(CONCAT(STR(?km_obs))))) AS ?obs)
  BIND(IRI(CONCAT("http://example.com/unit-km-mile-km", "
       _activity_", MD5(CONCAT(STR(?km_obs))))) AS ?activity
       )
  BIND(NOW() as ?now)
  ## computation and variable assignment
  BIND( ?km * 0.6214 AS ?value)
  ## error propagation
  BIND(?km_error + 0.1 as ?error)
  ## 1st termination condition:
  ## there exists no better observation with the same
       dimension values
  FILTER NOT EXISTS {
    ?obsa a qb:Observation ;
      sdmx-dimension:refArea ?city ;
      ex:indic ?indic ;
      ex:unitDimension qudt-unit:MileInternational ;
      sdmx-measure:obsValue ?valuea .
    # either get error value or use default value
    OPTIONAL { ?obsa ex:error ?errorap }
    BIND(COALESCE(?errorap, 0.0) AS ?errora)

    FILTER(?errora < ?error) }
  ## 2nd termination condition:
  ## the same equation was not used for the computation of
       any source observation

  FILTER NOT EXISTS { ?km_obs prov:wasDerivedFrom*/prov:
       wasGeneratedBy/prov:qualifiedAssociation/prov:hadPlan
       /prov:wasDerivedFrom? ex:unit-km-mile-km . }

}
```

AppendixC. Complete example of QB equation steps

An example for a QB equation is the Eurostat indicator definition for EU foreigners as proportion of the population. The next listing gives the QB equation.[63]

```
<qbe:Equation rdf:about="http://citydata.wu.ac.at/ocdp/
     eurostat-equations#
     eu_foreigners_as_a_proportion_of_population">
  <qbe:hasEquation rdf:datatype="http://citydata.wu.ac.at/
       qb-equations#equationType"> ?
       eu_foreigners_as_a_proportion_of_population = ?
       eu_foreigners / ?population </qbe:hasEquation>
  <qbe:structure rdf:resource="http://kalmar32.fzi.de/
       triples/global-cube.ttl#global-cube-dsd"/>
  <qbe:variable>
    <rdf:Description>
      <qbe:filter>
        <rdf:Description>
          <qbe:dimension rdf:resource="http://citydata.wu.
               ac.at/ns#hasIndicator"/>
          <qbe:value rdf:resource="http://citydata.wu.ac.at
               /ns#eu_foreigners"/>
        </rdf:Description>
      </qbe:filter>
      <qbe:variableName rdf:datatype="http://citydata.wu.ac
           .at/qb-equations#variableType">?eu_foreigners</
           qbe:variableName>
    </rdf:Description>
  </qbe:variable>
  <qbe:variable>
    <rdf:Description>
      <qbe:filter>
        <rdf:Description>
          <qbe:dimension rdf:resource="http://citydata.wu.
               ac.at/ns#hasIndicator"/>
          <qbe:value rdf:resource="http://citydata.wu.ac.at
               /ns#population"/>
        </rdf:Description>
      </qbe:filter>
      <qbe:variableName rdf:datatype="http://citydata.wu.ac
           .at/qb-equations#variableType">?population</
           qbe:variableName>
    </rdf:Description>
  </qbe:variable>
  <qbe:variable>
    <rdf:Description>
      <qbe:filter>
        <rdf:Description>
          <qbe:dimension rdf:resource="http://citydata.wu.
               ac.at/ns#hasIndicator"/>
          <qbe:value rdf:resource="http://citydata.wu.ac.at
               /ns#
               eu_foreigners_as_a_proportion_of_population"/
               >
        </rdf:Description>
      </qbe:filter>
      <qbe:variableName rdf:datatype="http://citydata.wu.ac
           .at/qb-equations#variableType">?
           eu_foreigners_as_a_proportion_of_population</
           qbe:variableName>
    </rdf:Description>
  </qbe:variable>
  <prov:wasDerivedFrom rdf:resource="http://ec.europa.eu/
       eurostat/cache/metadata/Annexes/urb_esms_an2.xlsx"/>
</qbe:Equation>
```

In the first step this equation is normalised to three QB rules. We give one example here:

```
<qbe:Rule rdf:about="http://citydata.wu.ac.at/ocdp/eurostat
     -rules#e0299b9f18a91815ba4f9cbd73960b112">
  <qbe:hasFunction rdf:datatype="http://citydata.wu.ac.at/
       qb-equations#functionType">?eu_foreigners/?population
       </qbe:hasFunction>
  <qbe:input>
    <rdf:Description>
      <qbe:filter rdf:nodeID="
           b1df1135590c30324cd362c1efd286125"/>
      <qbe:variableName rdf:datatype="http://citydata.wu.ac
           .at/qb-equations#variableType">?population</qbe:
           variableName>
    </rdf:Description>
  </qbe:input>
  <qbe:input>
    <rdf:Description>
      <qbe:filter rdf:nodeID="
           bba05d10b662474c6c0a474a8146c3fff"/>
      <qbe:variableName rdf:datatype="http://citydata.wu.ac
           .at/qb-equations#variableType">?eu_foreigners</
           qbe:variableName>
    </rdf:Description>
  </qbe:input>
  <qbe:output>
    <rdf:Description>
      <qbe:filter rdf:nodeID="
           b0299b9f18a91815ba4f9cbd73960b112"/>
```

[63] http://citydata.wu.ac.at/ocdp/eurostat-equations#eu_foreigners_as_a_proportion_of_population

```
    </rdf:Description>
  </qbe:output>
  <qbe:structure rdf:resource="http://kalmar32.fzi.de/
       triples/global-cube.ttl#global-cube-dsd"/>
  <prov:wasDerivedFrom rdf:resource="http://citydata.wu.ac.
       at/ocdp/eurostat-equations#
       eu_foreigners_as_a_proportion_of_population"/>
</qbe:Rule>
<rdf:Description rdf:nodeID="
     b1df1135590c30324cd362c1efd286125">
  <qbe:dimension rdf:resource="http://citydata.wu.ac.at/ns#
       hasIndicator"/>
  <qbe:value rdf:resource="http://citydata.wu.ac.at/ns#
       population"/>
</rdf:Description>
<rdf:Description rdf:nodeID="
     bba05d10b662474c6c0a474a8146c3fff">
  <qbe:dimension rdf:resource="http://citydata.wu.ac.at/ns#
       hasIndicator"/>
  <qbe:value rdf:resource="http://citydata.wu.ac.at/ns#
       eu_foreigners"/>
</rdf:Description>
<rdf:Description rdf:nodeID="
     b0299b9f18a91815ba4f9cbd73960b112">
  <qbe:dimension rdf:resource="http://citydata.wu.ac.at/ns#
       hasIndicator"/>
  <qbe:value rdf:resource="http://citydata.wu.ac.at/ns#
       eu_foreigners_as_a_proportion_of_population"/>
</rdf:Description>
```

Next the QB rules are converted to SPARQL CONSTRUCT queries. We give here the SPARQL query for the example QB rule above.

```
CONSTRUCT {
  ?obs qb:dataSet globalcube:global-cube-ds ;
       <http://citydata.wu.ac.at/ns#hasIndicator> <http://
            citydata.wu.ac.at/ns#
            eu_foreigners_as_a_proportion_of_population> ;
       dcterms:publisher ?source ;
       dcterms:date ?year ;
       sdmx-dimension:refArea ?city ;
       sdmx-dimension:sex ?sex ;
       estatwrap:unit ?unit ;
       sdmx-dimension:age ?age ;
       sdmx-measure:obsValue ?value ;
       prov:wasDerivedFrom ?population_obs, ?
            eu_foreigners_obs ;
       prov:wasGeneratedBy ?activity ;
       prov:generatedAtTime ?now ;
       cd:estimatedRMSE ?error .

  ?activity a prov:activity ;
  prov:qualifiedAssociation [
       a prov:Association ;
       prov:agent   cd:import.sh ;
       prov:hadPlan <http://citydata.wu.ac.at/ocdp/eurostat-
            rules#e0299b9f18a91815ba4f9cbd73960b112> ] .
}
WHERE { { SELECT DISTINCT * WHERE {
  ?population_obs qb:dataSet globalcube:global-cube-ds;
       <http://citydata.wu.ac.at/ns#hasIndicator> <http://
            citydata.wu.ac.at/ns#population> ;
       dcterms:date ?year;
       sdmx-dimension:refArea ?city ;
       sdmx-dimension:sex ?sex ;
       estatwrap:unit ?unit ;
       sdmx-dimension:age ?age ;
       sdmx-measure:obsValue ?population ;
       cd:estimatedRMSE ?population_error .

  ?eu_foreigners_obs qb:dataSet globalcube:global-cube-ds;
       <http://citydata.wu.ac.at/ns#hasIndicator> <http://
            citydata.wu.ac.at/ns#eu_foreigners> ;
       dcterms:date ?year;
       sdmx-dimension:refArea ?city ;
       sdmx-dimension:sex ?sex ;
       estatwrap:unit ?unit ;
       sdmx-dimension:age ?age ;
       sdmx-measure:obsValue ?eu_foreigners ;
       cd:estimatedRMSE ?eu_foreigners_error .

  BIND(CONCAT(REPLACE("http://citydata.wu.ac.at/ocdp/
       eurostat-rules#e0299b9f18a91815ba4f9cbd73960b112", "
       e0299b9f18a91815ba4f9cbd73960b112", MD5(CONCAT("http
       ://citydata.wu.ac.at/ocdp/eurostat-rules#
       e0299b9f18a91815ba4f9cbd73960b112",STR(?
       population_obs), STR(?eu_foreigners_obs))))) AS ?
       skolem)
  BIND(IRI(CONCAT(?skolem, "_source")) AS ?source)
  BIND(IRI(CONCAT(?skolem, "_obs")) AS ?obs)
  BIND(IRI(CONCAT(?skolem, "_activity")) AS ?activity)
  BIND(NOW() as ?now)
  ## computation and variable assignment
  BIND(?eu_foreigners*1.0/IF(?population != 0, ?population,
       "err") AS ?value)
  ## error propagation
  BIND((ABS(?eu_foreigners)+?eu_foreigners_error)*1.0/IF((
       ABS(?population)-?population_error) != 0.0, (ABS(?
       population)-?population_error), "err")-?eu_foreigners
       *1.0/IF(?population != 0, ?population, "err") + 0.1
       as ?error)
  FILTER(?error > 0.0)
  ## 1st termination condition:
  ## there exists no better observation with the same
       dimension values
  # FILTER NOT EXISTS {
  #     ?obsa qb:dataSet globalcube:global-cube-ds ;
  #         dcterms:date ?year;
  #         sdmx-dimension:refArea ?city ;
  #         <http://citydata.wu.ac.at/ns#hasIndicator> <
       http://citydata.wu.ac.at/ns#
       eu_foreigners_as_a_proportion_of_population> ;
  #         sdmx-dimension:sex ?sex ;
  #         estatwrap:unit ?unit ;
  #         sdmx-dimension:age ?age ;
  #         cd:estimatedRMSE ?errora .
  #     FILTER(?errora < ?error) }
  ## 2nd termination condition:
  ## the same equation was not used for the computation of
       any source observation

  #FILTER NOT EXISTS { ?population_obs prov:wasDerivedFrom
       */prov:wasGeneratedBy/prov:qualifiedAssociation/prov:
       hadPlan/prov:wasDerivedFrom? <http://citydata.wu.ac.
       at/ocdp/eurostat-rules#
       e0299b9f18a91815ba4f9cbd73960b112> . }

  #FILTER NOT EXISTS { ?eu_foreigners_obs prov:
       wasDerivedFrom*/prov:wasGeneratedBy/prov:
       qualifiedAssociation/prov:hadPlan/prov:wasDerivedFrom
       ? <http://citydata.wu.ac.at/ocdp/eurostat-rules#
       e0299b9f18a91815ba4f9cbd73960b112> . }
  }}}
```