BENEDIKT KÄMPGEN

Flexible Integration and Efficient Analysis of Multidimensional Datasets from the Web

KIT Scientific Publishing

Benedikt Kämpgen

FLEXIBLE INTEGRATION AND EFFICIENT ANALYSIS OF MULTIDIMENSIONAL DATASETS FROM THE WEB

Flexible Integration and Efficient Analysis of Multidimensional Datasets from the Web

by
Benedikt Kämpgen

Scientific Publishing

Dissertation, Karlsruher Institut für Technologie (KIT)
Fakultät für Wirtschaftswissenschaften, 2015

Tag der mündlichen Prüfung: 25. Februar 2015
Referenten: Prof. Dr. Rudi Studer, Prof. Dr. Axel Polleres

Impressum

Scientific Publishing

Karlsruher Institut für Technologie (KIT)
KIT Scientific Publishing
Straße am Forum 2
D-76131 Karlsruhe

www.ksp.kit.edu

Print on Demand 2015

ISBN 978-3-7315-0379-8
DOI 10.5445/KSP/1000047013

Flexible Integration and Efficient Analysis of Multidimensional Datasets from the Web

Zur Erlangung des akademischen Grades eines
Doktors der Ingenieurwissenschaften
(Dr.-Ing.)

bei der Fakultät für Wirtschaftswissenschaften
des Karlsruher Instituts für Technologie (KIT)

genehmigte
DISSERTATION

von

Dipl.-Inform. Benedikt Kämpgen

Tag der mündlichen Prüfung:	25. Februar 2015
Referent:	Prof. Dr. Rudi Studer
Korreferent:	Prof. Dr. Axel Polleres
Karlsruhe	August 2, 2015

Abstract

Numeric data such as statistics and from sensors are increasingly published on the Web and – if brought together for comparisons and calculations – can answer important questions in science, industry, and politics. For instance, natural scientists compare rainfall values from sensors with hydrological estimations documented in a semantic wiki; financial analysts evaluate companies based on comparisons between KPIs from balance sheets filed with the SEC and daily stock market values from Yahoo! Finance; and citizens want to explore the GDP per Capita of different countries, independent from information sources such as Eurostat, the IMF, and the World Bank.

However, the integration of datasets for analysis is difficult. First, heterogeneities remain a problem since publishers use different dimensions to describe numeric data, several identifiers for common entities, as well as differing levels of detail, units, and formulas. Second, aggregation and filtering operations, a varying selectivity of queries, and chains of joins over a growing number of possibly large datasets together with background information from the Web render analytical queries more complex than typical data analysis settings.

The broad acceptance of the RDF Data Cube Vocabulary (QB) for publishing multidimensional datasets and of Online Analytical Processing (OLAP) interfaces for intuitive and interactive knowledge discovery call for a uniform view – the Global Cube – over available numeric data for exploratory analysis. This work presents four complementary contributions to query the global cube:

1. A mapping between the common model of data cubes and QB to use existing OLAP engines for efficient queries over datasets from the Web.
2. An algorithm to evaluate OLAP operations over data cubes using SPARQL queries over QB datasets for more flexible data integration based on RDF stores.
3. A method to optimise analytical query processing via materialised aggregate views in RDF, including an evaluation with a realistic benchmark.
4. A method to declaratively describe complex relationships between datasets for flexibly increasing the number of answers from the global cube.

The contributions are applied to three scenarios in the areas of water resources management, company performance analysis, and Open Government Data exploration.

Acknowledgements

During these years working on my thesis at AIFB I always had the certainty of being able to talk to and to get help from several people. Without listing everybody by name, I wish to thank these important persons in my life.

I am very grateful to my supervisor Rudi Studer for always having an open ear and nourishing a working atmosphere that is inspiring and challenging without pressure;

to Andreas Harth for teaching me a lot about scientific and technical work, and for endless fruitful and constructive discussions;

to Basil Ell for his friendliness and openness about work, life, and everything;

to Roland Stühmer and Andreas Kämpgen for providing fresh views on my thesis;

to Axel Polleres for his interest and advise;

to Denny Vrandečić, Markus Krötzsch and many more for creating and maintaining the best knowledge management tool ever, Semantic MediaWiki;

to Daniel Herzig, Günter Ladwig, Andreas Wagner, Martin Junghans, Elena Simperl, Achim Rettinger, and all my other former and current colleagues at the Rudiverse for encouragement and support as well as diversity and diversion;

to Andreas Kleinberg, Alikali Fofana, Mark Menninger, Thomas Knäulein, Felix Steeger, and all my other friends, who stayed in touch;

to David Riepl, Yury Katkov, Karsten Hoffmeyer, Daniel Sommer, Seán O'Riain, and all other people I worked with on papers, events and projects for being fun and help at the same time;

to the BMBF, the DFG, and the European Union for making projects possible such as SMART, ACTIVE, PlanetData, SFB, Software Campus, and LD-Cubes;

to my family Christiane, Eckhart, Florian, Andreas, Gisela, Christel, Anni, Mathias, and several more for their encouragement and for always being there for me;

and in particular, to my wife Stefanie for her self-sacrificing support and love.

Thank you very much.

Contents

List of Figures

List of Tables

List of Listings

1 Introduction

For many important questions in science, industry, and politics, numeric data such as statistics and from sensors are available from the Web that – if integrated – provide relevant information for the answers.

For instance, natural scientists compare rainfall values from sensors with hydrological estimations documented in a semantic wiki. Financial analysts evaluate companies based on comparisons between Key Performance Indicators (KPI) from quarterly balance sheets and daily stock market values from a finance platform. Also, citizens want to explore the GDP per Capita of different countries, independent from sources such as Eurostat, the International Monetary Fund (IMF), and the World Bank.

However, the integration of datasets for analysis is difficult due to heterogeneities and the size of potentially interesting data. The work presents methods to increase flexibility and efficiency when using numeric datasets from the Web. In the remainder of this introduction chapter, the integration and analysis of multidimensional datasets from the Web is exemplified. Challenges are described, the overall research problem and approach are stated. Four research questions give rise to four contributions presented in later chapters. Descriptions of how research has been conducted, of previous publications supporting the contributions, and of how the thesis is organised then follow.

1.1 Numeric Data from the Web

Take as a concrete example the task to confirm or oppose Okun's law that proposes a relationship between employment and a country's Gross Domestic Product (GDP): "real GDP growth and change in unemployment are negatively correlated" [BJL14].

The relevant numbers (or values) are described by relations with several independent, mostly categorical attributes (so-called dimensions) and few dependent, mostly numeric attributes (so-called measures) [GCB+97]. See Table 1.1 for Employment Growth numbers from a relation `employmentgrowthdataset(Time, Geo, Sex, Employment Growth)`, available on the Web from Eurostat[1] with attributes for the dimensions and measures.

[1] http://appsso.eurostat.ec.europa.eu/nui/show.do?dataset=tps00180&lang=en, last accessed 2015-06-13.

Table 1.1: Example multidimensional dataset with employment growth values.

Time	Geo	Sex	Employment Growth
2006	AT	F	2.0
2007	AT	F	1.8
2006	AT	M	1.5
2007	AT	M	1.8
2006	UK	F	1.1
2007	UK	F	0.5
...	...	...	...

Every entity in such a multidimensional dataset is a numeric fact and contains independent dimension values for the period in time (Time), the country (Geo), the gender of the group of inhabitants (Sex) as well as a dependent measure value (Employment Growth). To confirm or oppose Okun's law, one needs to integrate this dataset with a dataset describing the real GDP growth rate and to compute the correlation. In many cases, however, a suitable dataset is not available for the specific task. Thus, different datasets published at different locations on the Web need to be integrated for analytical queries.

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. The present work investigates methods such as equivalence mappings and joining operations to build and query the unified view over a large set of multidimensional datasets from the Web.

Analysis is often done in an ad-hoc fashion of "overview first, zoom and filter, then details-on-demand" [Shn96] and allows the Extract-Visualize-Analyse loop [GCB+97] in an interactive search for interesting patterns in datasets. Therefore, operations to select (filter) and to sum-up (aggregation) numeric values need to be executed efficiently. This work presents methods such as materialisation and reuse of existing OLAP systems to allow domain experts – people with knowledge in a specific domain such as hydrology but without background in computer science – for exploratory analysis of integrated datasets.

Often, data analysis is done for selecting, understanding, and pre-processing of datasets. On top of analytical query results, automatic analysis methods can be applied, e.g., multi-criteria decision analyses and time series analyses. For example, to assess Okun's law, one can use existing mathematical methods to evaluate whether the Real GDP Growth Rate and the Employment Growth are correlated (e.g., using the Pearson Correlation Coefficient).

Similar tasks are needed in various domains. Three examples are given in the following: Natural scientists are interested in assessing planning alternatives in water resources

management according to indicators such as for quantifying the efforts in treating waste water. Necessary information come from datasets such as the monthly volume rate of water flow at a specific spring from sensors[2] and estimated in scientific analyses documented in a semantic wiki[3]. For instance, scientists would filter for a set of springs and aggregate to average annual discharge values as a basis for strategic decisions.

Also, business analysts are interested in a holistic view on companies such as Rayonier Inc. Analysts compare key performance indicators (KPI) such as the total asset in XBRL balance sheets from the SEC[4] with opening stock quotes from Yahoo! Finance[5]. Also, background information about companies such as from Wikipedia[6] and Freebase[7] are relevant. For instance, analysts would filter for companies with more than 1,000 employees and compare their average annual revenues and stock values.

As another example, given that organisations such as Eurostat, the International Monetary Fund (IMF), and the World Bank make available government data, citizens can get confirmed statistics such as the Gross Domestic Product (GDP) per Capita of European countries by several organisations[8]. For instance, citizens would filter for specific indicators such as the GDP Growth Rate, an Unemployment Fear metric, and the GDP per Capita and aggregate from a regional to a national level.

The following section shows that making use of these datasets on the Web is not an easy task for natural scientists, business analysts, and citizens. From such difficulties, an overall research problem can then be formulated.

1.2 Overall Research Problem and Approach

The integration of multidimensional datasets from the Web for analytical queries is difficult due to heterogeneity and size, as described in the following.

[2] Retrievable from http://www2.ufz.de/smarthydro/smartquery?location=AM0528&analysis_object=Q, last accessed on 2014-06-23.
[3] Retrievable from http://dropedia.iwrm-smart2.org/index.php/Shorea_Spring_10-year_average_discharge, last accessed on 2014-06-23.
[4] Retrievable from http://www.sec.gov/Archives/edgar/data/52827/000119312510238973/0001193125-10-238973-xbrl.zip, last accessed on 2014-10-16.
[5] Retrievable from http://ichart.yahoo.com/table.csv?s=RYN&a=00&b=01&c=2010&d=11&e=31&f=2010&g=d&ignore=.csv, last accessed on 2014-10-16.
[6] http://en.wikipedia.org/wiki/Rayonier, last accessed on 2014-10-17.
[7] http://www.freebase.com/m/089vkv, last accessed on 2014-10-17.
[8] Retrievable from http://appsso.eurostat.ec.europa.eu/nui/show.do?dataset=nama_aux_gph&lang=en, last accessed on 2014-10-16.

1.2.1 Heterogeneity of Datasets

The Web fosters free exchange of information on the basis of generic standards such as HTTP and URIs. Even if using similar protocols and publishing mechanisms, the same information may be represented differently for reasons such as varying technical environments, instantiations of data models, policies, technical background of staff, as well as priorities in making complete, high-quality, and self-descriptive information available. Consequently, multidimensional datasets found on the Web are heterogeneous and difficult to integrate, as exemplified in the following.

Data is published in different formats and models: For instance, sensor data from a water spring may be described in XML whereas research literature consists of semi-structured text. Balance sheets are described in a specific XML schema (XBRL) whereas stock market values come in CSV and background information in HTML. European statistics may be published in an XML profile (SDMX) but also using other common formats such as PC-Axis, Excel, SAS, Stata, SPSS, and the Google Dataset Publishing Language that each only can be consumed by specific tools.

Publishers use different identifiers for the same entities to contextualise their datasets: For instance, sensor data may be described with a "Geo" information versus other data with a "Location" information. Similarly, different values for the same entity are used such as "UK" and "United Kingdom". Also, there are no unique company identifiers across different reporting sources, and relationships between companies are obscure.

Numeric values carry a long information trail about the origin, the collection, and the context needed for understanding: For instance in the natural sciences, single values or entire datasets may be generated from different analyses by researchers and contain micro-data directly from a sensor or estimated data from a simulation. Values and datasets may also be produced by independently developed and executed data transformation workflows such as a conversion to RDF after a lookup on a CSV file and a web crawl from an RDFa-embedding website. Besides such provenance information, values and datasets may exhibit varying quality such as with respect to reliability and accuracy.

Datasets provide information on different levels of detail: For instance, a Regional GDP dataset[9] may first need to be aggregated from a regional level (NUTS 2) with regions such as "Shropshire and Staffordshire" to a country level (NUTS 0) with countries such as the "UK" to be compared with a dataset that serves the population only on a country level.

Every domain or community uses specific contexts to communicate the meaning of values: For instance, units of measurements such as "Million Euro" versus "Euro",

[9] http://estatwrap.ontologycentral.com/id/tgs00004, last accessed on 2014-10-18.

different currencies, and varying definitions of units cannot be directly compared. If the context is only informally communicated and implicitly known, there is the risk of misunderstandings and misinterpretations. For instance, it makes a difference whether the "Mean Discharge" of water from a spring is measured on a monthly or yearly basis.

Indicators may be compound and calculation procedures may only be implicitly described, hidden in code or in textual descriptions: For instance, natural science groups differently simulate, compute, or estimate indicators such as the "Waste Water Recharge Ratio". Similarly, relationships between financial concepts are only textually described and semantics are limited, e.g., between "SalesRevenueNet" and "Revenues" in the U.S. Generally Accepted Accounting Principles (US-GAAP). Financial ratios are computed to assess the performance of companies. Platforms return different values for important governmental statistics such as the GDP Per Capita. Also, within one institution, the same governmental statistic can be available several times; for instance, the GDP Per Capita may be stated in one dataset or calculated from several datasets describing components of the Nominal GDP and the number of inhabitants.

1.2.2 Large Size and Number of Datasets

An Open Data trend can be observed in different domains. More and more platforms such as FigShare[10], DataCite[11] and Pangaea[12] help scientists to not only publish their analysis results but also the raw (or also pre-processed) data for citations, reproduction, and further analysis. As another example, the G8 Open Data Charter encourages the publication of governmental statistics for transparency and innovation[13]. The increasing size and number of datasets lead to various data management problems.

Single datasets are large: For instance, natural scientists may have access to sensor data from decades. Also, public bodies are collecting millions of values within few years as a by-product from their daily tasks that are made available through Open Data platforms such as data.gov.uk.

The number of datasets is large: Financial analysts have access to quarterly and yearly balance sheets for more than 8,000 companies since 2009 from the U.S. Securities and Exchange Commission (SEC) as well as daily stock market values from Yahoo! Finance. Eurostat in total makes available more than 5,000 datasets with European statistics[14]. The World Bank publishes more than 8,000 datasets[15].

[10] http://figshare.com/, last accessed 2014-11-23.
[11] http://www.datacite.org/, last accessed 2014-11-23.
[12] http://pangaea.de/, last accessed 2014-11-23.
[13] https://www.gov.uk/government/publications/open-data-charter/g8-open-data-charter-and-technical-annex, last accessed on 2014-10-17.
[14] Depending on the modelling, according to http://eurostat.linked-statistics.org/, accessed 2014-10-17.
[15] Depending on the modelling, according to http://worldbank.270a.info/, last accessed on 2014-10-17.

Efficient query execution over an increasing size and number of datasets also is a problem since analytical queries involve a lot of values and require complex operations:

Possibly complex filter criteria with varying selectivity may be evaluated: For instance, natural scientists may be interested in certain water springs only. Financial analysts may want to specifically ask for companies with more than 1,000 employees. And citizens may be interested in values of a certain country such as the UK and a certain time period such as 2010. Complexity increases if several filter conditions are combined and are related to external background information.

Collections of values are aggregated with possibly complex aggregation functions or formulas: For example, queries may require a reduction of dimensionality (slice, e.g., average total assets over the last ten years), and an aggregation to a higher level (roll-up, e.g., from values of single companies such as Mastercard to average values of an industry such as "Service Business, NEC" from the Standard Industrial Classification, SIC).

Also, indicators may need to be computed from more granular figures, for instance, the "Waste Water Recharge Ratio" in the natural sciences, financial ratios of KPIs in company assessments, and the GDP Per Capita from the Nominal GDP and the population in Open Government Data analysis.

1.2.3 Overall Research Problem

Motivated by the challenges of heterogeneities between and large sizes and numbers of datasets from the Web, the overall research problem investigated in this thesis is:

Overall Research Problem: *How can we flexibly integrate and efficiently analyse multidimensional datasets from the Web?*

This research investigates approaches to integrate and analyse heterogeneous datasets in large numbers and sizes from the Web. Flexibility is improved if less manual work is needed to consider new data sources in an ad-hoc fashion, if interoperability is increased to reuse existing approaches and systems, and if solutions are applicable to different domains and heterogeneity challenges. Efficiency is improved if query processing time decreases, including the time for domain experts to identify, extract, transform, and load relevant data.

Traditional integration and analysis approaches are visualisation techniques, spreadsheets, relational database management systems, and knowledge discovery tools. They do not focus on the integration of datasets from the Web but knowledge discovery over a single pre-processed, multidimensional dataset of limited size. Current research either focusses on building semantic or high-performance integration and analysis systems. Semantic integration and analysis systems [Har10, HHU+11, KD08] consider

the heterogeneity of datasets but their interfaces and backends focus on metadata instead of large amounts of multidimensional datasets from the Web. Examples of high-performance systems are OLAP and Data Warehousing systems [VTBL13], NoSQL approaches [CHH+13], and real-time integration systems [DCSW09]; they allow efficient analytical query processing but for data integration of multidimensional datasets from the Web those approaches require continuous manual work such as for designing, implementing, and maintaining extract-transform-load (ETL) pipelines.

1.2.4 Overall Approach

This work builds on two assumptions. First, data providers increasingly publish numeric datasets as Statistical Linked Data using Semantic Web technologies. Second, domain experts are capable and willing to do exploratory knowledge discovery using Online Analytical Processing (OLAP). In the following, reasons for the two assumptions are given.

Statistical Linked Data: Statistical Linked Data is referred to as multidimensional datasets – together with related background information – published using best practices (Linked Data) and using a widely-adopted and standardised vocabulary for multidimensional datasets (RDF Data Cube Vocabulary). Statistical Linked Data already provide an abundance of available datasets and background information[16].

This thesis assumes that data providers increasingly publish multidimensional datasets as Statistical Linked Data [KC13]. Reasons are for example: Statistical Linked Data

- are easy to publish and sufficiently generic for a wide range of different use cases;
- fulfil the need of organisations in- and outside of the public sector to publish collected and aggregated numeric data in a standardised, machine-readable way on the Web;
- simplify the development of applications such as data ETL pipelines consuming datasets on the Web through standard access mechanisms (HTTP) and the use of a schema-flexible, graph-based data model (RDF);
- and support automatic processing and integration via self-descriptive datasets and well-defined ontologies based on logics.

[16] For an overview of datasets see http://wiki.planet-data.eu/web/Datasets, last accessed on 2014-10-17.

Online Analytical Processing (OLAP): Most interaction paradigms over Web data such as follow-your-nose browsers, faceted-search interfaces, and query builders [Har10, HHU+11, KD08] do not allow users to explore large amounts of numerical data. OLAP uses a conceptual model of Data Cubes close to the way of thinking of analysts [MLT09] and provides operations to view statistics from different angles and granularities, to filter for specific entities, and to compare aggregated measures [CCS93, CD97, PMT08].

This thesis assumes that domain experts are capable and willing to do exploratory knowledge discovery using Online Analytical Processing (OLAP) for several reasons. For example, OLAP provides

- intuitive and exploratory user interfaces over multidimensional datasets with pivot tables via OLAP clients;
- a formal definition of filter and aggregation operations over data cubes that can be optimised for efficient query processing over datasets stored in a data warehouse;
- and existing implementations of clients and engines that can be reused.

Given the two assumptions, hypotheses can be formulated that lead to research questions as described in the next section.

1.3 Research Questions and Contributions

This section describes four research questions and four contributions.

Figure 1.1 illustrates the overall research problem investigated in the present work. There are heterogeneous multidimensional datasets published in large numbers and sizes on the Web and available as Statistical Linked Data (right side of figure). Domain experts use familiar OLAP tools to issue queries over a unified view – the so-called global cube – over all multidimensional datasets from the Web (left side of figure).

From this setting, hypotheses can be formulated leading to one of four research questions illustrated in Figure 1.1 (middle of figure, numbered from 1 to 4) and described in the following.

One hypothesis states that the functionalities of OLAP engines can be utilised. Since widely-used for decision making in industry, many OLAP engines are developed. Each of the existing OLAP engines translate analytical queries from domain experts to queries over datasets stored in database management systems. Engines provide performance optimisation such as caching and capabilities for data integration. Thus, the following research question is formulated:
Research Question 1. *How can we use existing OLAP engines for efficient query processing over heterogeneous multidimensional datasets from the Web?*

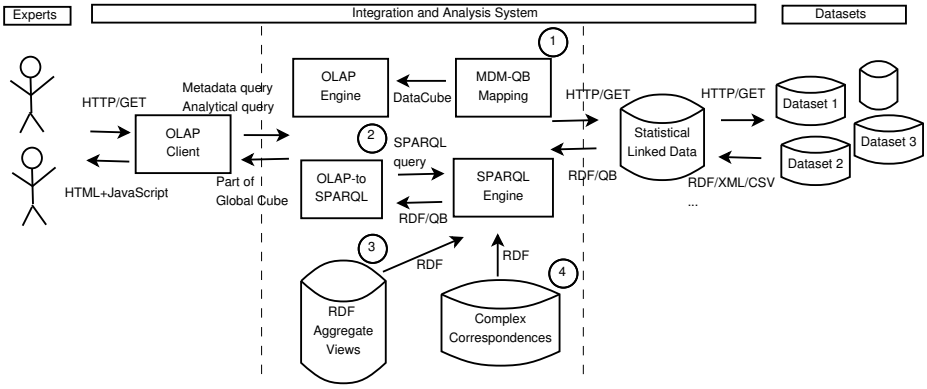


Figure 1.1: Illustration of overall research problem (left and right sides of figure) and single research questions each answered by an artifact as a contribution in the present work (middle of figure, numbered from 1 to 4).

This question cannot be easily answered for several reasons: Related work from research and industry focusses on functionality of single engines and on automatic multidimensional modelling but not on interoperability between OLAP systems. There are no standardised formats for sharing of data between OLAP engines. There is a semantic gap between the conceptual and logical level of data models for multidimensional datasets. Also, querying of multiple datasets from the Web leads to difficulties such as how to identify and load necessary data as well as how to bring measures from different datasets together.

Another hypothesis states that flexibility can be increased if existing Linked Data query engines are used for query processing. Due to the flexible schema of RDF, such engines are assumed to require less up-front effort to prepare both numeric data and arbitrary background information for analysis than relational database management systems. Example background information possibly useful to consider in an analysis include the number of employees of a company and the language spoken in a country. SPARQL 1.1, the most current version of the RDF query language, is assumed sufficiently expressive for analytical queries and allows to query for arbitrary background information to be displayed in analysis interfaces. The respective research question is:
Research Question 2. *How can we use existing SPARQL engines for query processing and flexible integration of multidimensional datasets from the Web?*

However, this question is not easy to answer. Related work concentrates on analytical query processing using relational database management systems but not on queries over schema-flexible RDF data from the Web. OLAP queries as issued by domain experts cannot directly be translated to SPARQL queries over Statistical Linked Data. For example, meaningful aggregation functions and multi-level hierarchies need to be

considered. Also, it is unknown how measures from several datasets with no explicit overlaps can be brought together.

Even if analytical operations such as filter, aggregation, and integration can be issued over statistics as schema-flexible RDF data using SPARQL, existing engines are assumed to be less efficient than relational database management systems and to require optimisations such as materialisation. It is assumed that an empirical argument can be given in favour of creating a specialised OLAP engine over Statistical Linked Data. Thus, the following research question is formulated:
Research Question 3. *How can we optimise analytical query processing over multidimensional datasets from the Web using aggregate views?*

However, related work mainly provides answers to views over RDF data when metadata but no numeric data is queried. Furthermore, it is unknown how to select, compute, and store materialised views for aggregated statistics in RDF. So far, few benchmarks are available to argue for the case of RDF in analytics scenarios.

The power of RDF lies not only in the flexible schema, but also in the self-descriptive data with well-specified semantics. Standard Semantic Web ontologies allow to describe implicit relationships between datasets as a basis for automatic integration. One hypothesis of the present work states that multidimensional datasets from the Web can be integrated to one unified view, the global cube. Furthermore, if heterogeneities between datasets are reduced, it is expected that the same statistic can be confirmed by different datasets and sources. Thus, the last research question to be addressed is:
Research Question 4. *How can we increase the number of answers from the global cube in case of complex relationships between multidimensional datasets from the Web?*

However, related work concentrates on relational settings with few sources that are not distributed on the Web. Research results on semi-automatic matching of ontologies cannot be applied to the integration of numeric datasets. Also, relationships between different datasets are often buried in informal descriptions, and the routines to resolve semantic conflicts are provided in code or external background information. If aligned, datasets and data sources may contain contradicting information and lead to a global cube too large in size for efficient analytical queries.

For each of the four research questions the work provides a contribution in the form of a viable artifact according to the guidelines of design science [HMPR04] (the research approach will be described in the next section). As illustrated in Figure 1.1 (middle of figure, numbered from 1 to 4) this work presents the following contributions:

1. The MDM-QB Mapping (Chapter 5) describes how datasets from Statistical Linked Data relate to the common conceptual model of data cubes (Multidimensional Data Model, MDM) used in existing OLAP engines for integration and query processing.

2. The OLAP-to-SPARQL Algorithm (Chapter 6) describes how to transform analytical OLAP queries to SPARQL queries over Statistical Linked Data, including multi-cube queries over several datasets.

3. RDF Aggregate Views (Chapter 7) suggest how to model and store pre-aggregated numeric values for more efficient query execution. Performance is compared with the traditional relational approach.

4. Complex Correspondences (Chapter 8) propose a syntax and semantics for conversion and merging relationships between datasets and increase the number of possible answers from the global cube.

The next section explains the scientific method used to answer the research questions.

1.4 Design Science Method

According to Simon [Sim96, p. 111], an important group of artificial things can be created with a computer. The intellectual activity of an engineer in building such *artifacts* using a computer is comparable with the intellectual activity of a physicians in creating a diagnosis and prescribing a remedy for a sick patient. In contrast, *Design Science* is concerned with how such artificial things created with a computer ought to be, with devising artifacts to attain goals.

The four research questions of this thesis are investigated along design science research guidelines [HMPR04] as described in the following:

Guideline 1: Design as an Artifact. This thesis designs artifacts to allow domain experts to flexibly integrate and efficiently analyse multidimensional datasets from the Web. Artifacts from this thesis are a mapping between a logical and a conceptual data model (MDM-QB Mapping), an algorithm for translating between query languages (OLAP-to-SPARQL), a data model for pre-aggregated numeric values (RDF Aggregate Views), and a syntax and semantics for complex relationships between datasets (Complex Correspondences).

Guideline 2: Problem Relevance. In this thesis, the challenges of integration and analysis of datasets from the Web for domain experts are illustrated by three concrete scenarios from different domains (natural sciences, financial data analysis, politics). For each scenario, user requirements are derived that only can be fulfilled if challenges in heterogeneities and amount of numeric datasets are solved. The four research questions investigate flexible and efficient solutions.

Guideline 3: Design Evaluation. Every artifact is separately evaluated in experiments on the capability to solve challenges of heterogeneity as well as size and number of

datasets from the Web. For that, the artifacts are implemented and executed in controlled environments with given datasets and workloads. In an overall evaluation, the applicability of the artifacts is shown along three detailed scenarios from different domains.

Guideline 4: Research Contributions. This work investigates the advantages and difficulties of applying Semantic Web standards – primarily used for sharing of metadata – for flexible integration and efficient analysis of numeric datasets from the Web. Four complementary artifacts are presented that can be used individually or combined to increase flexibility in building and efficiency in querying the global cube.

Guideline 5: Research Rigor. Artifacts in the present work are described such that they can be re-implemented with a computer, and such that experiments can be repeated to reproduce the results. The MDM-QB Mapping is described using set notation and a well-specified query language (SPARQL). The OLAP-to-SPARQL Algorithm is described in pseudo-code and implemented in an Open Source Java software. The definition of RDF Aggregate Views is based on Semantic Web standards, is implemented, and compared with the relational pendant. The syntax and semantics of Complex Correspondences are described using relations and the programming language Datalog.

In performance experiments, the queries, the data, as well as the software and hardware environments are described. Implementations are published as Open Source, datasets as Open Data.

Guideline 6: Design as a Search Process. Assuming datasets published as Linked Data, and analysis conducted by domain experts via OLAP, there are different approaches to solve challenges of heterogeneity and size of data. Since flexibility and efficiency are contradictory goals, a compromise needs to be found for every concrete scenario. The four contributions presented in this work investigate different directions for achieving a good compromise. The MDM-QB Mapping provides the foundations for metadata, the OLAP-to-SPARQL Algorithm for analytical queries over Statistical Linked Data. Whereas RDF Aggregate Views increase efficiency but require management effort, Complex Correspondences increase flexibility but are costly to evaluate. Three scenarios from different domains can benefit from these contributions.

Guideline 7: Communication of Research. The problem of integration and analysis of datasets from the Web is illustrated for both technology- and management-oriented readers in three scenarios. The four artifacts presented in this work are formally described and evaluated for people with respective background in data modelling and integration as well as query processing and optimisation. The artifacts are separately applied and discussed along three scenarios to convince practitioners of further implementations and deployments.

1.5 Previous Publications

The overall research problem (Chapter 1) has been discussed in a doctoral consortium paper [Kä11]:

> Benedikt Kämpgen. DC Proposal: Online Analytical Processing of Statistical Linked Data. In *11th International Semantic Web Conference (ISWC) Doctoral Consortium*, 2011.

The contributions of this work have been peer-reviewed and published as follows.

The MDM-QB Mapping (Chapter 5) has been published in a research paper [KH11]:

> Benedikt Kämpgen and Andreas Harth. Transforming Statistical Linked Data for Use in OLAP Systems. In *7th International Conference on Semantic Systems (I-SEMANTICS)*, 2011.

The OLAP-to-SPARQL Algorithm (Chapter 6), including a formal description of the MDM-QB Mapping (Chapter 5), has been published as a revised selected paper from the ESWC Workshop on Interacting with Linked Data [KOH12]:

> Benedikt Kämpgen and Seán O'Riain and Andreas Harth. Interacting with Statistical Linked Data via OLAP Operations. In *9th Extended Semantic Web Conference (ESWC) Satellite Events*, 2012.

RDF Aggregate Views (Chapter 7), including a thorough evaluation and extension of the OLAP-to-SPARQL Algorithm (Chapter 6) for query processing over large datasets with multi-level hierarchies, have been published as a research paper [KH13]:

> Benedikt Kämpgen and Andreas Harth. No Size Fits All – Running the Star Schema Benchmark with SPARQL and RDF Aggregate Views. In *10th Extended Semantic Web Conference (ESWC)*, 2013.

Complex Correspondences (Chapter 8), including a formal definition of the global cube and an extension to the OLAP-to-SPARQL Algorithm (Chapter 6) for evaluating the Drill-Across operation, have been published as a research paper [KSH14]:

> Benedikt Kämpgen and Steffen Stadtmüller and Andreas Harth. Querying the Global Cube: Integration of Multidimensional Datasets from the Web. In *19th International Conference on Knowledge Engineering and Knowledge Management (EKAW)*, 2014.

The contributions of the thesis have been implemented in an Open Source Java software, OLAP4LD[17]. Benefits of using the RDF Data Cube Vocabulary for representing statistics have been identified in a W3C Working Group Note about use cases and lessons of the RDF Data Cube Vocabulary [KC13].

Applications of OLAP4LD in three scenarios (Chapter 9) have been peer-reviewed and published as follows.

The SMART Knowledge Base (Section 9.2) has been discussed in Deliverable 203 of the SMART research project [KRH+13] and published in a workshop co-located with the ESWC [KRK14]:

> Benedikt Kämpgen and David Riepl and Jochen Klinger. SMART Research using Linked Data – Sharing Research Data for Integrated Water Resources Management in the Lower Jordan Valley. In *ESWC Workshop on Semantic Publishing (SePublica)*, 2014.

The Financial Information Observation System (Section 9.3) has been presented and nominated as best In-Use paper at the ESWC [KWO+14]:

> Benedikt Kämpgen and Tobias Weller and Seán O'Riain and Craig Weber and Andreas Harth. Accepting the XBRL Challenge with Linked Data for Financial Data Integration. In *11th Extended Semantic Web Conference (ESWC)*, 2014.

The Linked Data Cubes Explorer (Section 9.4) has been published as a demo paper at the ESWC 2014 [KH14]:

> Benedikt Kämpgen and Andreas Harth. OLAP4LD – A Framework for Building Analysis Applications over Governmental Statistics. In *11th Extended Semantic Web Conference (ESWC) Satellite Events*, 2014.

[17] http://www.linked-data-cubes.org/index.php/OLAP4LD, last accessed on 2015-05-02.

1.6 Organisation of this Thesis

The remainder of the present work is structured as follows:

The next chapter (Chapter 2) introduces three scenarios from different domains (natural sciences, finance, politics) to illustrate the overall research problem of this thesis. Each of the scenarios exhibits specific user requirements. From the specific user requirements and concrete examples of heterogeneity and size of numeric data in the scenarios, general requirements are derived that can only be solved with flexible and efficient approaches.

Chapter 3 gives definitions needed for the understanding of this work. Statistical Linked Data, the common conceptual model of data cubes, as well as analytical queries over data cubes are formally introduced.

Chapter 4 gives an overview of the state of the art. The limitations of traditional integration and analysis systems as well as existing implementations of high-performance and semantic approaches are described.

Four core chapters (Chapter 5 to Chapter 8) each investigate one of the research questions. Assuming the overall setting that publishers of datasets use Statistical Linked Data and that domain experts are familiar with OLAP, each core chapter presents an approach to increase flexibility and efficiency in querying a unified view of all available datasets, the global cube:

- MDM-QB Mapping (Chapter 5): allows for efficient dataset integration and query execution using existing OLAP engines.
- OLAP-to-SPARQL Algorithm (Chapter 6): allows for flexible dataset integration and query execution using existing SPARQL engines.
- RDF Aggregate Views (Chapter 7): allows for analytical query optimisation over RDF using materialised aggregate views.
- Complex Correspondences (Chapter 8): allows for describing complex mappings between datasets to increase the number of answers from the global cube.

Chapter 9 describes in an overall evaluation how the contributions of the present work can be applied in the three scenarios to fulfil the user requirements. The chapter also discusses how the contributions fulfil the general requirements to data integration and analysis.

Chapter 10 gives a summary of the results and describes open questions.

Further guide to the reader. The Index contains important terms used throughout the thesis, including possible abbreviations and acronyms. If a term is defined that is relevant for later parts of the thesis, the term is made *italic* and added to the index. A

relevant term previously defined in an earlier part of the thesis and extended in a chapter is made *italic* and also added to the index.

For several parts of the work, additional information is made openly available on the Web. The Appendix A gives short descriptions and links of the available information.

For readability reasons, in the remainder of the thesis “we” refers to the author.

2 Scenarios

In this chapter, we present three scenarios from different domains motivating the integration and analysis of multidimensional datasets from the Web.

The SMART scenario (Section 2.1) is about domain experts from the natural sciences that investigate water scarcity in the Middle East. The XBRL scenario (Section 2.2) involves business analysts that perform company assessments based on financial data. The Open Government Data (OGD) scenario (Section 2.3) is concerned with citizens and politicians interested in statistical indicators about European countries. For each of the scenarios, specific user requirements are described.

We then derive domain-independent, general requirements (Section 2.4) of integration and analysis over datasets from the Web. In later core chapters we investigate approaches to increase flexibility and efficiency in dataset integration and analysis. In a later overall evaluation chapter (Chapter 9), we describe how the specific user requirements and the general requirements are solved by our contributions.

2.1 Sharing Research Data for Water Resources Management (SMART)

Numerous regions of the world face immense pressure and competition on their natural freshwater resources[1]. As a critical example, in the region of the Lower Jordan River in the Middle East, a steadily increasing population has access only to constantly decreasing natural freshwater resources.

Decision makers such as politicians have to choose among many alternative strategies of how to improve the situation in the Lower Jordan valley.

Different from other water resources management methods, *Integrated Water Resources Management* (IWRM) considers social, economical and ecological objectives simultaneously when deciding on long-term strategies in a study area [AdAB00]. Scientists and decision makers from Israel, Jordan, Palestine and Germany in the SMART project try to establish IWRM approaches for Sustainable Management of Available Water

[1] http://politics.slashdot.org/story/13/02/13/1731237/nasa-huge-freshwater-loss-in-the-middle-east, last accessed on 2014-09-23.

Resources with Innovative Technologies (SMART) for countries bordering the Lower Jordan[2].

For instance, the Jordanian Water Strategy [Rie13, see Table 3.14] has as social objective to "decrease the leakages from sewer pipes in As Salt, Fuheis and Mahis", as economical objective to "improve meter reading and billing accuracy to reduce administrative losses", and as ecological objective the "implementation of Protection Zones for the springs Azraq, Baqourria, Hazzir and Shorea".

Table 2.1 illustrates how in IWRM the best possible water management strategy in an area is selected. In this *decision matrix* a set of alternative water management strategies to improve the situation in the Lower Jordan valley (IWRM scenarios: reference, business-as-usual, and full implementation) are compared regarding multiple, partly conflicting evaluation criteria (indicators: e.g., the "Waste Water Recharge Ratio") covering social, economical and ecological objectives. For each IWRM scenario and indicator a value is estimated from analyses and simulations; the higher a value, the better the performance of an IWRM scenario.

Table 2.1: Example decision matrix to select best water management strategy in an area; normalised indicator values for reference IWRM scenario, business-as-usual (BAU) scenario, and full implementation (FI) scenario in Wadi Shueib in 2025 [Rie13, average values from Table 3.28]; the higher a numeric value in the table, the better the performance of an IWRM scenario regarding one specific indicator.

Objective	**Indicator**	**IWRM Scenario**		
		Reference	BAU	FI
Social	Waste Water Treatment Ratio	0.00	0.08	0.23
Social	Waste Water Recharge Ratio	0.30	0.26	0.87
Social	Available Groundwater	0.04	0.12	0.41
Social	Available Surface Water	0.28	0.42	0.21
Social	Available Reclaimed Water	0.02	0.09	0.73
Economical	Municipal Shortage	0.17	0.20	0.75
Economical	Agriculture Shortage	0.38	0.56	0.60
Economical	Municipal Supply Requirement	0.21	0.15	0.30
Economical	Unit Cost	0.38	0.28	0.05
Ecological	Environmental Water Stress	0.08	0.05	0.01

In the following, we describe the typical steps in an IWRM decision process, including how a decision matrix is created and how the final decision is made.

Problem formulation. After initialising a decision process, decision makers define social economical, and ecological IWRM objectives that are to be optimised in a specific region, e.g., to "increase volume of captured and treated waste water".

[2] http://www.iwrm-smart2.org/, last accessed on 2014-06-16.

Domain modelling. Domain experts from social, economical and ecological sciences define indicators to evaluate the grade of reaching an IWRM objective within a water strategy, e.g., "Waste Water Recharge Ratio", the ratio of untreated waste water to natural groundwater recharge. Also, IWRM scenarios are selected. Every scenario is a description of a development pathway towards a future state of a study area at a defined planning horizon such as 2025. IWRM scenarios should be consistent and plausible (internal factors, e.g., climate change) and propose implementable actions (external factors, e.g., building a new well).

Model execution. Based on the assumptions in the domain model, experts create analyses to estimate indicator values for IWRM scenarios.

For instance, the untreated waste water is estimated from inflow volumes of waste water treatment plants and pumped sewage of cesspits; the natural groundwater recharge is estimated from the average volume of generated waste water by one inhabitant per month times the total population in a catchment (an area with certain water resources and consumers).

Various data sources such as publications (e.g., the official Water Strategy of Jordan), encyclopaedias (e.g., BMBF Water Glossary, Agricultural and Farm Systems documentation of the Food and Agriculture Organization) and basic indicator values (e.g., sensor records) are relevant. Also, in IWRM analyses, indicators are often calculated or estimated based on sensor data and the simulation of complex domain models.

Multi-criteria decision analysis (MCDA). Finally, the decision makers fill a decision matrix such as shown in Table 2.1 with values provided by the domain experts. Determining the preferable IWRM scenario from a decision matrix is a multi-criteria decision analysis problem (MCDA). The typical MCDA approach is to ask decision makers to assign weights to criteria (indicators) and to rank the decision alternatives (IWRM scenarios) with respect to the values and weights. Possible methods to compute this rank include the Analytical Hierarchy Process (AHP) [Saa80]. The highest ranked IWRM scenario is selected.

However, IWRM processes are collaborative and knowledge intensive: Data in the IWRM domain is complex, often having many dimensions or leaving a long provenance trail from sensors over analyses to reports; is available from distributed sources such as research publications, dataset catalogues, and official documents; is heterogeneous since coming from social, economical and ecological domains; and may contain unstructured information such as maps and free text.

To handle this information complexity, applied IWRM projects and case studies usually use multi-thematic information systems to share data between their interdisciplinary modelling tools. Although these systems are capable of providing raw data on the one hand and highly aggregated model outputs on the other hand, they fail to support collaborating scientists. As a consequence, IWRM researchers often only collaborate informally and within small groups using email and spreadsheets. Assumptions or

research data between such groups are rarely shared or aligned, so that research results are not comparable.

Therefore, in the SMART project, information technology is used to make more transparent the decision process for third-parties and to define operational guidelines for scientists contributing to IWRM processes.

First, the web-based Knowledge Management System *Dropedia*[3] is based on wiki technology and aims at fostering collaboration among stakeholders in the SMART project. Dropedia is open for reading by the entire IWRM community; all SMART project members also have write access. In Dropedia, decision makers can explicitly establish decision processes and transparently decide upon and share "IWRM objectives", "IWRM indicators" and "IWRM scenarios". Scientists can share and discuss information about assumptions and preliminary research data.

Second, to archive and share climate sensor data, the SMART project provides stakeholders Web-based access to an Oracle database (*SMART-DB*). SMART-DB contains large amounts of measurements, e.g., daily precipitation data of 70 climate stations since 1980 and borehole characteristics for more than 3000 wells in Jordan. Additional sensor data can be imported by project members in a Web form.

To implement and simulate domain models, scientists create or import indicator values from documents or SMART-DB to the *Water Evaluation And Planning* software (WEAP). To solve a decision process, decision makers create Excel sheets with IWRM decision matrices and import them to a multi-criteria analysis tool *EWRE-AHP* that is based on the Analytical Hierarchy Process (AHP) [Saa80] and computes the most promising strategy.

The *SMART scenario* leads to the following user requirements: Domain experts need to be able to identify, integrate and re-use research data from expert analyses such as the calculation or estimation of a single value, a literature study, and complex domain assumptions (**SMART User Requirement**). However, there are challenges:

Heterogeneity of Datasets. There is limited interoperability between Dropedia, WEAP, SMART-DB, and EWRE-AHP.

Information is represented according to different data models. For instance, sensor data from SMART-DB is available from a Web-API in HTML or XML whereas research descriptions from Dropedia mostly are semi-structured text.

[3] Dropedia has been presented in a dissertation in the natural sciences [Rie13]; high-level information about Dropedia can be found at http://semantic-mediawiki.org/wiki/Dropedia last accessed 2014-11-04; throughout the SMART project, Dropedia was reachable at http://dropedia.iwrm-smart2.org/, last accessed 2014-08-27.

Also, data sources and tools use different identifiers. For instance, the Web-API refers to a "location" dimension whereas Dropedia uses a dimension "geo". Similarly, "analysis objects" describing indicator values use abbreviations such as "Q" for "Water Discharge".

The relational database of SMART-DB does not store all information necessary for understanding; datasets may have been generated from different analyses by researchers and may contain micro-data directly from a sensor or estimated data from a simulation. Another important information for interpretation that easily is taken for granted and not explicitly shared are scientific units such as volume rate of water flow in m^3/h.

Large Size and Number of Datasets. To use information across tools, domain experts often need additional efforts such as manual copy-pasting of tables with tools such as Microsoft Excel. However, processing large amounts of sensor data from the SMART-DB is error-prone and costly with spreadsheets.

For instance, natural scientists need to select sensor data from certain springs and during certain time frames for their analyses. Also, scientists are interested in average values of annual water discharge of springs and the number of climate values available in the SMART-DB.

The estimation and simulation of indicator values such as the "Waste Water Recharge Ratio" from datasets in Dropedia and SMART-DB requires a lot of manual effort by scientists.

In a later overall evaluation chapter (Chapter 9), we show how to overcome these challenges to fulfil the SMART user requirement.

2.2 Integrating Finance Data for Company Performance Analysis (XBRL)

Analysts play a crucial role in the functioning of equity markets. Besides the actual analysis, e.g., comparing *key performance indicators* (KPIs) such as the Gross Profit Margin between companies, analysts spend a disproportionate amount of time with data curation, i.e., identifying, gathering and preparing data [DdAF+10] and pursue to minimise time spent on tedious data pre-processing tasks. The *Extensible Business Reporting Language* (XBRL)[4] is an XML format for financial information that is more amenable to automatic processing than traditional financial information representations such as PDF, HTML and text documents. Still, XBRL does not solve the problem of data integration – e.g., of company background information, balance sheets, stock quotes – for a holistic view on companies [OCH12]:

[4] http://www.xbrl.org/Specification/XBRL-2.1/REC-2003-12-31/XBRL-2.1-REC-2003-12-31+corrected-errata-2013-02-20.html, last accessed on 2014-10-20.

- XBRL uses an XML syntax that is difficult to understand and process, e.g., due to an extension with link bases for referencing across documents [CTG12].
- Automatically deriving information from XBRL is difficult since formal semantics are limited [WTB11, Spi10]. Relationships between financial concepts, such as "SalesRevenueNet" and "Revenues" in the United States Generally Accepted Accounting Principles (US-GAAP), are only textually described.
- Financial information from different XBRL documents often cannot be compared since accounting and regulatory organisations across countries and branches do not align their taxonomies of financial concepts; new versions, e.g., of US-GAAP, lack backward compatibility; and XBRL allows publishers to define their own concepts.
- Gathering information about a company is difficult since there are no unique company identifiers across different reporting sources[5], and relationships between companies are obscure.
- There is no globally accepted schema for finance data and other finance-related Open Data such as stock quotes (e.g., CSV) and background information (e.g., HTML) are published using different data models.

The financial data analysis scenario introduced in this section is inspired by the XBRL Challenge organised by XBRL US. In this challenge, solutions are sought to provide benefit around financial data analysis using XBRL from the *U.S. Securities and Exchange Commission* (SEC) that since 2009 requires more than 8,000 U.S. companies traded on the stock market to file financial statement information such as quarterly and yearly balance sheets in the XBRL format to the SEC Edgar Database[6].

A quarterly balance sheet for example would disclose that Rayonier Inc. had a *sales revenue net* of 377,515,000 USD from 2010-07-01 to 2010-09-30[7].

In our scenario, an investor wants to assess companies based on XBRL and other available financial data sources. The investor would find several analyses useful:

- *Background information analysis*, e.g., looking at company information from different sources such as the address, the founding date and the industry.
- *Multi-company KPI analysis*, e.g., comparing KPIs for several companies over time such as the stock market price within the same industry.

[5] http://sunlightfoundation.com/sixdegrees/, last accessed 2015-01-31.
[6] http://sec.gov/rules/final/2009/33-9002.pdf, last accessed 2014-11-07.
[7] Retrievable from http://www.sec.gov/Archives/edgar/data/52827/000119312510238973/0001193125-10-238973-xbrl.zip, last accessed on 2014-10-16.

- *Cross-data-sources KPI analysis*, e.g., comparing values from heterogeneous datasets such as the Earnings per Share from yearly balance sheets with prices per share from electronic stock quotes as well as Total Assets published using the US-GAAP version 2009 and version 2011.

The *XBRL scenario* leads to the following user requirements: to answer above analysis queries, business analysts need to integrate data across sources (**XBRL User Requirement 1**); to understand and trust data, analysts need to be able to explore data in a fashion of "overview first, zoom-in, details on demand" [Shn96] (**XBRL User Requirement 2**); finally, since analysts are non-tech-savy, have little time and cannot use complex query languages, they need to create their own analyses with Excel-like functionality (**XBRL User Requirement 3**). However, there are challenges:

Heterogeneity of Datasets. As mentioned above, XBRL does not solve the problem of heterogeneities between financial information; yearly and quarterly balance sheets from the SEC Edgar Database with varying taxonomy versions of US-GAAP, company and industry background information from Wikipedia/DBpedia, and daily stock quotes from the Yahoo! Finance Web API are all published with different schemas.

When extracting data, additional contextual information about values and datasets (provenance) is important for understanding, since datasets may be produced by independently developed and executed data transformation workflows such as when created by a conversion to RDF after a lookup on a CSV file and crawled from an RDFa-embedding website [FKaGO+12]. Also, a lot of information is only implicitly described, e.g., in footnotes attached to single financial facts. Also, different currencies need to be considered.

Large Size and Number of Datasets. Efficient query processing is difficult due to large numbers and sizes of datasets; for instance, the SEC would publish balance sheets for over 8,000 US companies; assuming any company publishes four quarterly and one yearly balance sheet since 2009, there are more than 200,000 XBRL instance documents available for analysis. Also, new stock market values are published at least once a day for any traded company.

Query processing. Those challenges arise from complicated queries. For instance, analysts might filter for companies from cities with more than 100,000 inhabitants or filter for companies that have a CEO who is younger than 20 years old. Such metadata may for instance come from background information such as Wikipedia.

Also, queries may require a reduction of dimensionality (e.g., average total assets over the last ten years), and a roll-up from company to an industry classification such as the *Standard Industrial Classification* (SIC).

In a later overall evaluation chapter (Chapter 9), we show how to overcome these challenges to fulfil the XBRL user requirements.

2.3 Exploring Governmental Statistics from the Web (OGD)

According to the G8 Open Data Charter and Technical Annex[8] statistics in *Open Government Data* (OGD) provide information of high value for improving transparency and encourage innovative re-use of data.

Driven by efforts such as the Share-PSI 2.0 project on Shared Standards for Open Data and Public Sector Information[9] and the Government Linked Data Working Group (GLD)[10] of the *World Wide Web Consortium* (W3C), more and more governmental statistics are made available on the Web in machine-readable formats such as the *Statistical Data and Metadata Exchange* format (SDMX).

For instance, *Eurostat*[11] publishes on behalf of the European Commission more than 5,000 datasets with indicators about European countries. Example datasets include the *Eurostat GDP Growth Dataset* with the growth rate of the gross domestic product of all European countries per year[12].

Another example is the *Cumulated German General Social Survey* (ALLBUS) conducted by the *Leibniz-Institute for the Social Sciences* (GESIS) that on behalf of the German federal state North Rhine-Westphalia provides information about "attitudes, behaviour and social structure in Germany"[13]; among others, we can retrieve the *Unemployment Fear Survey Dataset* where German employees were asked about their fear of becoming unemployed in the last few years.

Given the Open Data policy of governments and intergovernmental organisations, citizens can find important statistical indicators in several datasets on the Web. For example, one can find the Gross Domestic Product of countries per year from Eurostat, the World Bank and the International Monetary Fund. Integrating such multidimensional datasets will allow for more complete answers and detailed comparisons of indicators. For example, the GDP of a country from one and the population from another dataset enable analysts to compute the GDP per Capita and to cross-check these derived values with values from other publishers.

Even if available on the Web in machine-readable formats, integration is difficult [TC05] since datasets

[8] https://www.gov.uk/government/publications/open-data-charter/g8-open-data-charter-and-technical-annex, last accessed 2014-11-23.
[9] http://www.w3.org/2013/share-psi/, last accessed on 2014-06-08.
[10] http://www.w3.org/2011/gld/, last accessed on 2014-06-08.
[11] http://ec.europa.eu/eurostat, last accessed on 2015-06-13
[12] http://appsso.eurostat.ec.europa.eu/nui/show.do?dataset=tec00115&lang=en, last visited 2015-06-13.
[13] http://www.gesis.org/en/allbus, last accessed 2014-10-29.

- describe their indicators with different dimensions, e.g., "geo-location", "time" or "gender",
- use different names for the same dimensions, e.g., "geo" and "location" or dimension values, e.g., "DE" and "Germany",
- provide different levels of detail, e.g., regional or national level,
- contextualise values differently, e.g., units of measurement such as "Million Euro" and "Euro", and data collection characteristics such as "sampling",
- and publish datasets derived from other datasets, e.g., "GDP per Capita" computed via "Nominal GDP" divided by "Population".

For the *OGD scenario* we can derive the following user requirements: In front-ends such as Microsoft Excel, pivot tables have proved intuitive to build and easy to understand when exploring numeric datasets. Citizens need to explore any of the governmental statistics published on the Web in pivot tables (**OGD User Requirement 1**); and citizens want to have confirmed important statistical indicators by as many datasets from the Web as possible (**OGD User Requirement 2**).

However, there are heterogeneity and scalability challenges:

Heterogeneity of Datasets. As mentioned above, even if published in a machine-readable and widely-used format, heterogeneities remain a problem. For instance, Eurostat datasets exhibit varying number of dimensions such as geo, time, gender and age for the population dataset, which makes combined visualisations, such as in two-dimensional line diagrams, and comparisons difficult.

Eurostat statistics are published using an XML format (SDMX); other statistics such as from the World Bank are published differently, e.g., using HTML web tables, Excel, CSV, and JSON[14]. Also, proprietary formats only understandable by certain tools are in use, for instance, PC-Axis, SAS, Stata, SPSS, and the Google Dataset Publishing Language. One cannot easily ensure that all relevant data is found and properly modelled.

As an example for datasets with different levels of detail, the Regional GDP dataset from Eurostat[15] first needs to be aggregated from a regional level (so-called NUTS 2) with regions such as "Shropshire and Staffordshire" to a country level (NUTS 0) with countries such as the "UK" to be combined with the population per country.

Similarly, datasets may have different data quality such as with respect to reliability and accuracy. For instance, Eurostat attaches an "estimated" attribute to values with lower reliability. Also, units are relevant when values are described in "Million Euro" or "Euro" as well as complexer units such as "Euro per Inhabitant".

[14] http://data.worldbank.org/, last accessed on 2014-06-03

[15] http://estatwrap.ontologycentral.com/id/tgs00004, last accessed on 2014-10-18.

Large Size and Number of Datasets. Efficient query processing is difficult due to the amount of data as described in the following:

There is a large number of datasets available, e.g., 5,000 from Eurostat[16] and 8,301 from the World Bank[17]. The size and number of datasets not only depends on the original publisher of numeric data but also on the modelling of third parties re-publishing the data[18]. For instance, OECD Linked Data publishes 123 datasets with 26,243 observations on average. The International Monetary Fund (IMF) contains 4 datasets each with 806,992 facts on average. As another example of a large dataset, the Open Data platform data.gov.uk publishes in their Combined On-line Information System (COINS) five financial datasets from across the public sector in the UK with in total 4.9 million rows of data[19].

Similarly, efficient query processing is difficult since citizens may issue complex queries. On the one hand, citizens may be interested in values of a certain country such as the UK and a certain time period such as 2010. On the other hand, aggregated overall values may be needed as an overview[20].

Also, citizens interested in important statistical indicators may find several answers. For instance, for the GDP Per Capita for the UK in 2010, Wolfram Alpha returns 29,520[21] whereas Eurostat returns 27,800 Euros per Inhabitant[22]. Similarly, within the same institution, indicators may be available in different ways; for instance, the GDP Per Capita can be computed from the nominal GDP divided by the population.

In a later overall evaluation chapter (Chapter 9), we show how to overcome these challenges to fulfil the OGD user requirements.

2.4 Requirements Analysis

For each of the scenarios we have derived specific *user requirements*. User requirements only can be fulfilled if challenges due to heterogeneities between and large number and sizes of datasets from the Web are solved.

[16] Depending on the modelling, according to http://eurostat.linked-statistics.org/, last accessed on 2014-10-17.

[17] Based on a SPARQL query for datasets over http://worldbank.270a.info/.html, last accessed on 2014-06-06.

[18] For instance, the following estimations are based on SPARQL queries for datasets over http://oecd.270a.info/.html and http://imf.270a.info/.html, last accessed on 2014-06-06.

[19] http://data.gov.uk/resources/coins, last accessed on 2014-10-17.

[20] As an example, see use case on "Publishing hierarchically structured data from StatsWales and Open Data Communities" [KC13]

[21] http://www.wolframalpha.com/input/?i=+gdp+per+capita+in+uk+in+2010+in+eur, changing over time, last accessed 2014-11-23.

[22] http://appsso.eurostat.ec.europa.eu/nui/show.do?dataset=nama_aux_gph&lang=en, last accessed 2014-11-23.

In this section, we derive *general requirements* from these specific data integration and query processing challenges. Table 2.2 shows an overview of general requirements; we indicate with "X" that a general requirement is supported by a scenario. Every requirement is supported by at least two scenarios.

Table 2.2: Overview of general requirements derived from the SMART, XBRL, and OGD scenarios; with "X" we indicate requirements supported by a scenario.

General Requirement	**Scenario**		
	SMART	XBRL	OGD
Flexibility			
R1: Integrate different data formats and models	X	X	X
R2: Consider different identifiers for the same entities	X	X	X
R3: Store provenance trail of datasets	X	X	X
R4: Integrate datasets of different levels of detail	X	X	X
R5: Consider datasets with different units of measurements	X	X	X
R6: Merge datasets for compound measures	X	X	X
Efficiency			
R7: Scale to large datasets	X		X
R8: Scale to large number of datasets		X	X
R9: Efficiently filter for values with varying selectivity	X	X	X
R10: Efficiently execute aggregation functions and formulas	X	X	X

General Requirement 1 to Requirement 6 (R1 – R6) are concerned with the flexibility of approaches to cope with heterogeneity of multidimensional datasets from the Web.

R1: Integrate different data formats and models. All three scenarios require the integration of different data formats and models. Examples include Web APIs, XML profiles, and semi-structured descriptions of publications in a semantic wiki.

R2: Consider different identifiers for the same entities. All three scenarios give examples of identical things referred to by different identifiers. Examples include identifiers for locations, companies, and countries.

R3: Store provenance trail of datasets. All scenarios would benefit from the capability to attach provenance information to single numeric values and entire datasets. Provenance information for instance describes tools used in natural science analyses, data transformation pipelines generating financial facts, and the reliability of statistical indicators.

R4: Integrate datasets of different levels of detail. All three scenarios give examples of datasets with different levels of detail that need to be compared. The hierarchy

from single days, over months, to years is a common example. For instance, micro sensor data capture the climate on a specific day, whereas many estimations only are given on a yearly basis. Other examples include a point in time versus a period in time as distinguished for financial facts and regional- versus country-level statistical indicators[23].

R5: Consider datasets with different units of measurements. All three scenarios require the consideration of different units of measurements. In SMART, there are scientific units; in XBRL, there are currencies; in OGD, there are units such as "Euro per Inhabitant". Not only units of measurements but also other contextual information such as whether an indicator is measured on a monthly or yearly basis are important.

R6: Merge datasets for compound measures. In all three scenarios, the computation of compound indicators and formulas is useful. Indicators such as the "Waste Water Recharge Ratio" in the natural sciences, financial ratios such as the "Earnings per Share", and statistical indicators such as the GDP per Capita all require measures from different datasets and the calculation of formulas.

General Requirement 7 to Requirement 10 (R7 – R10) are concerned with the efficiency of approaches to cope with size and number of multidimensional datasets from the Web as well as complex analytical operations.

R7: Scale to large datasets. The SMART and OGD scenarios give examples of large datasets. For instance, the relational database in SMART-DB contains datasets with daily precipitation data of 70 climate stations since 1980 and borehole characteristics for more than 3,000 wells in Jordan. Large Open Government datasets are published by the International Monetary Fund (IMF), and the Combined On-line Information System (COINS). Although not large in comparison to data warehouses with Gigabytes of data in industry, datasets of 100MB and larger take more than 100sec to download from the Web[24].

R8: Scale to large number of datasets. Whereas every balance sheet or daily stock market table in the XBRL scenario is small, with 200,000 balance sheets and daily stock market values, the number of datasets is large. In the SMART project, the number of available datasets interesting to scientists is limited but the OGD scenario contains more than 5,000 datasets from Eurostat.

A number of more than 1,000 datasets certainly creates a bottleneck in the analysis since assuming 500ms per HTTP look-up on the respective URI of each dataset, only resolving all URIs – without downloading – takes more than 8min.

[23] For more examples see the Master thesis by Dominik Siegele [Sie12], co-supervised by the author, available at http://www.aifb.kit.edu/images/4/4b/Masterarbeit_Dominik_Siegele.pdf, last accessed 2014-12-17.

[24] Based on 6.7Mbps global average connection speed, estimated at http://techcrunch.com/2012/08/09/akamai-global-average-broadband-speeds-up-by-25-u-s-up-29-to-6-7-mbps/, last accessed 2014-11-16.

The modelling of datasets determines the size or the number of datasets. For instance, in the SMART scenario, we re-publish one dataset per location (6,517 locations[25]) and per indicator (132 indicators[26]) instead of publishing all data in one single dataset.

R9: Efficiently filter for values with varying selectivity. All three scenarios give examples of different filter criteria applied to datasets in queries. In SMART, natural scientists are interested in sensor data from a certain spring as well as from all springs in a certain region. Financial analysts of XBRL may want to specifically ask for companies with more than 1,000 employees. And citizens may be interested in values of a certain country and time period such as the UK in 2010.

R10: Efficiently execute aggregation functions and formulas. All three scenarios give examples of aggregations along multi-level hierarchies and the execution of formulas. If climate sensor data is aggregated to an average per year, many single observations are aggregated. Similarly, if a financial ratio is computed for an industry, values for each single company in the industry are aggregated.

Also, if a formula such as for converting "Million Euro" to "Euro" in the OGD scenario is executed over a large or many small datasets, the formula has to be executed many times.

Each of the research questions in the forthcoming chapters investigate approaches to increase flexibility in data integration and efficiency in query processing. Afterwards, in Chapter 9 we discuss how well the contributions fulfil the general requirements.

[25] According to Linked Data wrapper of SMART-DB, http://smartdbwrap.appspot.com/locationlist.html, last accessed on 2014-10-22.

[26] According to Linked Data wrapper of SMART-DB, http://smartdbwrap.appspot.com/analysisobjectlist.html, last accessed on 2014-10-22.

3 Basic Definitions

Whereas the next chapter (Chapter 4), describes related work with similar research goals, this chapter defines common data structures, data models, query languages, concepts, and methods required for understanding the contributions in this thesis.

In Section 3.1, we introduce a common data model of data cubes that we use as a basic conceptualisation for the integration and analysis of multidimensional datasets. In Section 3.3, we define Statistical Linked Data that we use as a main data source for and representation formalism of multidimensional datasets from the Web. In Section 3.2, we define analytical queries as nested set of operations from an algebra over data cubes.

3.1 Multidimensional Data Model

In this section, we first explain the difficulties of modelling numeric data. Second, we introduce a formal definition of a common data model to describe multidimensional datasets conceptually. Third, we describe the star schema, a common way to represent and store multidimensional datasets in a relational database for efficient analytical queries.

3.1.1 Modelling Numeric Data

Modelling *numeric data* is challenging for the following reasons:

- Numeric data can be aggregated (also called statistics, or macro data) or raw such as from sensors (also called micro data).
- Representing statistics requires more complex modelling as discussed by Martin Fowler [Fow96]: Recording a statistic simply as an attribute to an object – for instance a person weighs 185 – fails to represent important concepts such as the measurement period and unit.

- Instead, a statistic is modelled as a first-class object, an observation. The object describes an observation of a value, e.g., a numeric value (e.g., 185) in the case of a measurement and a categorical value (e.g., "blood group A") in the case of a categorical observation.
- To allow correct interpretation of the value, the observation needs to be further described by "dimensions" such as the specific phenomenon, e.g., "weight", the time the observation was made, e.g., "January 2013" or a location where the observation was made, e.g., "New York".
- To further improve interpretation of the value, attributes such as presentational information, e.g., a series title "Combined Online Information System (COINS) Results 2013" or critical information for understanding the data, e.g., the unit of measure "pounds" are given to observations.
- Given additional background information, e.g., arithmetical and comparative operations, humans and machines can appropriately visualise such observations or perform conversions between different quantities.

Therefore, we model numeric data as *multidimensional datasets*[1] as per Definition 1.
Definition 1 (Multidimensional Dataset)**.** *We define a multidimensional dataset as a relation with n independent attributes (mostly categorical, dimensions) and a certain number of dependent attributes (mostly numeric, measures) [GCB+97]. The measure attributes are functionally dependent on the dimension attributes. Therefore, for every set of values for the dimension attributes there is only one value for each measure attribute.*

An example relation with three attributes[2] denoting dimensions and one attribute denoting a measure was given in Chapter 1 `employmentgrowthdataset(Time, Geo, Sex, Employment Growth)` in Table 1.1.

However, one relation cannot capture all metadata relevant for the integration and analysis of multidimensional datasets. Therefore, in the next section, we introduce an elaborate conceptual model.

3.1.1.1 Data Cubes, Dimensions, Measures

In the following, we define a conceptual model for multidimensional datasets, a common *multidimensional data model* (MDM). The MDM serves an intuitive, semantically rich and platform independent abstraction of a multidimensional dataset [PMT08].

A formally defined abstraction of multidimensional datasets allows 1) to define logical representations of multidimensional datasets for storage, e.g., in a relational database

[1] If not stated otherwise, we also use the term *dataset* for denoting multidimensional datasets.
[2] We use attribute and "variable" interchangeably.

and 2) to define operations to filter, aggregate, and integrate datasets within an analysis.

Although there is no standard MDM [PJD01, LMTS06, GGV12], all models have certain *Multidimensional Elements* in common. We intuitively define an MDM as per Definition 2, illustrate a common MDM in Figure 3.1, and then formally define each single element.

Definition 2 (Multidimensional Data Model)**.** *A multidimensional data model treats data as n-dimensional* data cubes*. The independent attributes of a data cube are called* dimensions*, the dependent attributes* measures*. The possible values of dimensions are referred to as* members*. Members are grouped along* hierarchies *of one or more* levels*. The higher the level in the hierarchy, the less granular the members.* Facts *are the single data points in a cube. Facts have a value for every dimension and measure of the cube.*

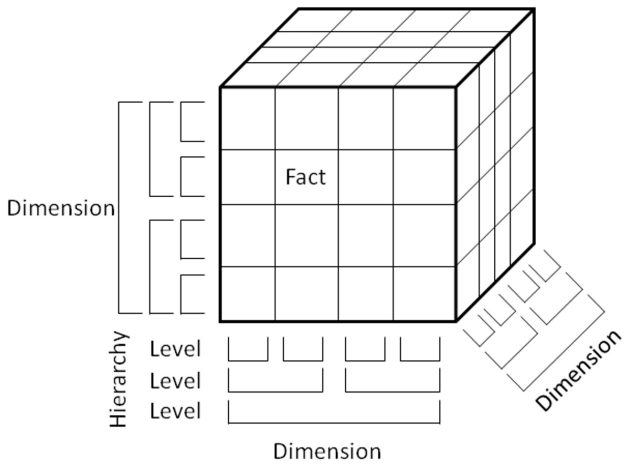


Figure 3.1: Illustration of a common multidimensional data model.

We use simple set notation as syntax and set theory as semantics [HKR09, p. 363ff] to formally define for each multidimensional element in an MDM the set of all possible instances as follows:

Member defines the set of members as $Member = \{(nameM) \in String\}$. Let $\mathcal{V} = 2^{Member}$, $V \in \mathcal{V}$. There is a special-type member ALL.

Level defines the set of levels as $Level = \{(nameL, V, depth) \in String \times \mathcal{V} \times Integer\}$. Let $\mathcal{L} = 2^{Level}$, $L \in \mathcal{L}$, $ROLLUPMEMBER \subseteq Member \times Member$, $ROLLUPLEVEL \subseteq Level \times Level$, depth indicates the distance from the (implicit) *ALL* level (having depth 0) within a $rolluplevel(L)$ relation, $rollupmember(L) = \{(v_1, v_2) \in$

$V_1 \times V_2 |\ l_1 = (\text{"}l_1\text{"}, V_1, depth_1), l_2 = (\text{"}l_2\text{"}, V_2, depth_2) \in L$, v_1 is more specific than v_2 $\}$, $rolluplevel(L) = \{(l_1, l_2) \in L \times L |\ l_1$ is more specific than l_2 $\}$.

There is a special-type level ALL.

Given $rollupmember(L)$ and $v \in V$, the parent (sometimes parents) of a member is given by $rollupmember(L)(v)$. Given $rolluplevel(L)$ and $l \in L$, the next higher level is given by $rolluplevel(L)(l)$. Rollupmember and rolluplevel are not transitive.

Hierarchy defines the set of hierarchies as $Hierarchy = \{(nameH, L, rolluplevel(L), rollupmember(L)) \in String \times \mathcal{L} \times ROLLUPLEVEL \times ROLLUPMEMBER\}$.

Let $\mathcal{H} = 2^{Hierarchy}$.

The *rolluplevel* relation defines an ordering of levels by which the depth of a level can be computed. The *ALL* level has depth 0.

The lowest levels define the maximum depth of a hierarchy. Let *maxdepth* be the maximum depth.

$rollupmember(L)$ is a rollup relation with a partial order on the members in L. Hereby, members in a lower level roll up to members of the next higher level.

Dimension defines the set of dimensions as $Dimension = \{(nameD, H) \in String \times \mathcal{H}\}$. Let $\mathcal{D} = 2^{Dimension}$. Let $dom : Dimension \rightarrow \mathcal{V}$ return all members of a dimension.

Measure defines the set of measures as $Measure = \{(nameMS, calc) \in String \times CalculationExpression\}$

To reduce conceptual differences between dimensions and measures, and to simplify querying, every measure implicitly is a member of a special-type level "Measures" in a hierarchy "Measures" of a dimension "Measures".

Let $\mathcal{M} = 2^{Measure}$. Let $dom : Measure \rightarrow 2^{String}$ return all possible values of a measure.

Similar as defined by Köppen et al. [KSS12], a *calculation expression* defines how a measure value is computed and consists of *aggregation functions* such as SUM and possibly algebraic functions between measures such as `GDP per Capita = Nominal GDP / Population`. The latter allows to define *compound measures* built by combining several measures [DPS14].

The standard aggregation functions for numeric values are SUM, AVG, COUNT, and for nominal values COUNT. Every aggregation function has as input a set of numeric values and output a numeric value. For instance, SUM is defined as follows: $SUM : 2^{Decimal} \rightarrow Decimal$.

Fact defines the set of possible statistical facts as $Fact = \{(nameF, C, E) \in String \times 2^{Dimension \times Member} \times 2^{Measure \times String}\}$, $c \in C = (\{(d_1, c_1), \ldots, (d_{|D|}, c_{|D|})\}, \{(m_1, t_1), \ldots, (m_j, t_j)\})$, $j \leq |M|$ with $c_i \in Member \cup ALL$ and $t_i \in T$ with T a numeric domain including the special *null* value in case of cube sparsity. Also, $(\forall (d_1, c_1), (d_2, c_2) \in C, c_1 \neq c_2) d_1 \neq d_2$ and $(\forall (m_1, t_1), (m_2, t_2) \in E, t_1 \neq t_2) m_1 \neq m_2$; thus each fact has for each dimension and measure at maximum one member and value, respectively. Let $\mathcal{F} = 2^{Fact}$.

DataCubeSchema defines the set of data cube schemas as $\{(nameCS, D, M) \in String \times \mathcal{D} \times \mathcal{M}\}$.

DataCube defines the set of data cubes as $DataCube = \{(nameDC, cs, F) \in String \times DataCubeSchema \times \mathcal{F} | cs = (cs, D, M)\}$, with $(\forall fact1 = (\text{"}fact1\text{"}, C_1, E_1), fact2 = (\text{"}fact2\text{"}, C_2, E_2) \in F, fact1 \neq fact2) C_1 \neq C_2, \{d | \exists (d, m) \in C_1\} = D, (\forall (d, m) \in C_1)\ m \in dom(d)$.

There are several assumptions to data cubes:

- The measure value is fully dependent on the dimension members, thus, any two facts of a data cube need to have a different member on one of their dimensions.
- Any fact needs to have a member for each dimension mentioned in the schema.
- Each member needs to be contained in a level of a hierarchy of the dimension. A data cube may be sparse and not containing facts for each possible combination of dimension members.
- If not said otherwise, we assume the explicitly given facts of a data cube to be *base facts*. Base Facts have members only on the lowest level of each dimension.
- Implicitly, a data cube contains *aggregate facts*, facts with members on higher levels for dimensions that can be computed by aggregating lower-level facts, e.g., facts describing Male and Female can be aggregated to ALL, meaning the total of Male and Female [GCB+97].
- Similarly according to Gómez et al. [GGV12], we assume that the $rolluplevel(L)$ relation of a hierarchy forms a directed acyclic graph (DAG), i.e. every level has a unique parent level. As a consequence, every hierarchy has a unique bottom level $BOTTOM$ and a unique top level ALL.

Two data cubes may *overlap* if they have *shared dimensions*, i.e., use the same (part of their) dimensions, hierarchies, levels, and members. In the literature shared dimensions are often also referred to as *conformed dimensions* [KR02].

In the following section we give a concrete example of an MDM with multidimensional elements from one concrete multidimensional dataset.

3.1.1.2 Example Multidimensional Data Model

As an example dataset we use "Population on 1 January by age and sex" published by Eurostat[3]. The example MDM contains members, e.g., Germany and single years, including the special-type member *ALL*:

```
Member = {DE, ES, ..., 2004, ..., ALL}
DE = ("DE")
2004 = ("2004")
...
```

The model contains levels such as for single days in which members are grouped. Also, there are special-type ALL levels with the ALL member. Additionally, there is a measure level containing the members for measures:

```
Level = {timeLevelDay, ..., geoLevelNUTS0, ..., sexLevel, ...,
    ageLevel, measuresLevel}
timeLevelDay = ("timeLevelDay", {2004-01-01, 2004-01-02, ...}, 3)
timeLevelMonth = ("timeLevelMonth", {2004-01, 2004-02, ...}, 2)
timeLevelYear = ("timeLevelYear", {2004 , 2005, ..., 2013}, 1)
timeLevelAll = ("timeLevelAll", {ALL}, 0)
geoLevelNUTS0 = ("geoLevelNUTS0", {DE, ES, ...}, 1)
measuresLevel = ("measuresLevel", {populationMeasSUM,
    populationMeasAVG, populationMeasCOUNT})
```

Hierarchies such as for time give an order to levels, e.g., single days to months to years. A hierarchy also defines roll-up relations between levels and members. For instance, there is a hierarchy from day to months to years to the ALL level.

```
Hierarchy = {timeHierarchy, geoHierarchy, sexHierarchy,
    ageHierarchy, measuresHierarchy}
timeHierarchy = ("timeHierarchy", timeL, rolluplevel(timeL),
    rollupmember(timeL)
timeL = {timeLevelDay, timeLevelMonth, timeLevelYear, timeLevelAll}
geoHierarchy = ("geoHierarchy", geoL, rolluplevel(geoL),
    rollupmember(geoL)
geoL = {geoLevelNUTS2, geoLevelNUTS1, geoLevelNUTS0, geoLevelAll}
rolluplevel(timeL) = {(timeLevelDay, timeLevelMonth),
    (timeLevelMonth, timeLevelYear), (timeLevelYear,
    timeLevelAll)...}
rollupmember(timeL) = {(2004-01-01, 2004-01), (2004-01, 2004),
    (2004, ALL)...}
```

The model contains dimensions such as for time, geo, and age.

```
Dimension = {timeDim, geoDim, ageDim, sexDim, measuresDim}
```

Also, measures are defined by the model.

[3] http://appsso.eurostat.ec.europa.eu/nui/show.do?dataset=demo_pjan&lang=en, accessed 2014-06-16.

```
Measure = {populationMeasSUM, populationMeasAVG,
    populationMeasCOUNT}
populationMeasSUM = ("populationMeasSUM", SUM)
```

The values in the dataset can be aggregated by different measures and aggregation functions. As default aggregation functions for numbers we have SUM, AVG and COUNT, for nominal values, COUNT.

Also, the model contains facts that describe numeric values. Facts can refer to micro data that is not aggregated such as `fact1` where AVG and SUM contain the same value since COUNT is 1. Similarly, facts can refer to macro data such as `fact2` where the age dimension is aggregated to the ALL level and member. Macro data typically is not explicitly encoded in a dataset but only implicitly given and can be made explicit by aggregation.

```
Fact = {fact1, fact2, fact3...}
fact1 = (fact1, fact1C, fact1E)
fact1C = {(timeDim, 2013), (geoDim, DE), (sexDim, F), (ageDim, Y18)}
fact1E = {(populationMeasSUM, "390,156"), (populationMeasAVG,
    "390,156"), (populationMeasCOUNT, "1")}
fact2 = (fact2, fact2C, fact2E)
fact2C = {(timeDim, 2013), (geoDim, DE), (sexDim, F), (ageDim, ALL)}
fact2E = {(populationMeasSUM, "41,673,725"), (populationMeasAVG,
    "467,835"), (populationMeasCOUNT, "44")}
```

The model defines a cube schema with time, geo, sex, and age as dimensions and also a dimension for measures. Also, this schema defines the measures.

```
DataCubeSchema = {populationCubeSchema}
populationCubeSchema = (populationCubeSchema, {timeDim, geoDim,
    ageDim, measuresDim}, {populationMeasSUM, populationMeasAVG,
    populationMeasCOUNT})
```

Finally, the model contains the population cube. The cube contains any number of facts and also can be sparse.

```
DataCube = {populationCube}
populationCube = ("populationCube", populationCubeSchema, {fact1,
    fact2, fact3...})
```

3.1.1.3 Summarisability

Summarisability, the possibility to correctly compute aggregate facts with a lower granularity from facts with higher granularity is important for performance optimizations based on pre-aggregation and ensures results understandable to analysts [MLT09].

According to Lenz and Shoshani [LS97] summarisability is given if

- dimension members only belong to one parent member (disjointness),
- members belong to exactly one parent member and every member on a higher level has at least one child member (completeness),
- and aggregation functions are meaningful (meaningful aggregation function).

Disjointness is given in case of *strict* dimension hierarchies [RTZ11] where every member at the child level relates to no more than one member of its parent level.

Completeness is given in case of *symmetric* dimension hierarchies [MZ06] where every higher-level member (parent) has a child member.

Whether an aggregation function is meaningful depends on the dimensions used for aggregation and the semantics of the measure. For the dimensions we distinguish time dimensions from dimensions that do not describe a time aspect. Assuming the standard aggregation functions MIN, MAX, AVG, SUM, and COUNT, only the following aggregation functions are not meaningful [LS97]:

- SUM, if a fact aggregates over a time dimension and uses a measure quantifying the stock of something at a certain point in time (e.g., a bank account balance).
- SUM, if a fact aggregates over any dimension and uses a measure computing a ratio at a certain point in time (e.g., a price per item).

3.1.2 Representation in a Star Schema

According to Vassiliadis [VS99], we can distinguish three different perspectives in modelling data for databases: the conceptual perspective, dealing with the high level, abstract representation of the world, the physical perspective, dealing with the details of the representation and processing of data in the hardware, and the logical perspective, acting as an intermediate between the two aforementioned extremes, trying to balance a storage-independent paradigm and a natural representation of the information in terms of computer-processable concepts.

After having explained modelling of data cubes on the conceptual perspective, we next present the modelling in the logical perspective in a relational database.

The most common form is the *star schema* [CD97]. Advantages of the star schema are: an intuitive transformation from data cubes to a star schema; the possible application in any of the widely-used relational databases; a good compromise between ease of maintenance by normalisation (separation of fact and dimension tables) and efficiency by reducing the number of joins needed for queries (denormalised dimension tables). Also, industry-relevant data such as from accounting and customer relationship management often resemble star schemas [SBC$^+$07] and research has focused on optimising analytical queries over relational data approaches [MKIK07].

In a star schema, the database contains one *fact table* for each data cube and one *dimension table* for each dimension of the data cube.

The fact table has an attribute with a foreign key for each of the dimensions of the data cube. Also, the fact table has an attribute for each measure of the data cube. Typically, those attributes are numeric such as for describing the population. The primary key of the fact table consists of the combination of foreign keys.

Besides a generated primary key, every dimension table has an attribute for each of the levels of the dimension. Tables of shared dimensions are used by several data cubes.

See Figure 3.2 for an example star schema for the population data cube represented as a *UML Class Diagram*. Usually, a UML class diagram is used in object-oriented programming, e.g., using the language Java. Classes define the attributes and methods of any objects constructed from the class. In Figure 3.2, we instead use classes to define the attributes of the fact and dimension tables.

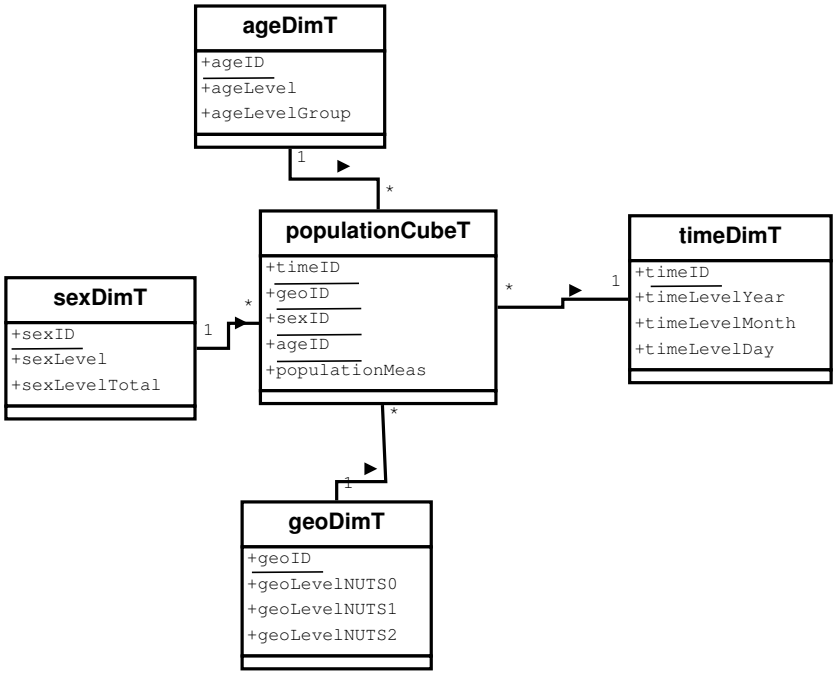


Figure 3.2: Example star schema for Population Data Cube; represented in UML class diagram using classes to define attributes of fact and dimension tables.

Assumptions taken for multidimensional datasets also hold for the relational representation: Most importantly, in the fact table, the attributes for measures are functionally dependent on the values of the dimensions. That means, in a fact table for every unique

combination of dimension members, there is only one row in the fact table serving the measure values.

Also, a fact table usually contains measure values on a specific level of granularity (mostly lowest) of each dimension table. That means, no redundancy is contained; consequently, no aggregate facts are logically and physically stored in the fact table and will be computed by the aggregation of lower level facts.

Aggregate facts are often pre-computed [CD97, ABD+99] to speed up query processing. Typically, aggregate facts are logically and physically stored in *aggregate tables*, new fact tables aggregating the original fact table on one or more selected dimensions. Aggregate tables reuse dimension tables of dimensions that are not aggregated and introduce new *shrunken dimension tables* for aggregated dimensions. Storing aggregate facts in the original fact table of the data cube complicates querying and is error prone since special type *ALL* members and *null* values need to be distinguished in SQL queries. Also, query processing becomes slower due to a larger number of facts that need to be scanned.

According to Mazón et al. [MLT09] a data cube is close to the way of thinking of human analysts and, therefore, it helps users to understand data; also processing (querying) a data cube can be well optimised since the clear structure allows designers to predict decision makers' intentions. In the next section, we will present a way to query data cubes modelled using our MDM.

In our work the star schema is relevant because many existing OLAP engines assume data to follow this schema for analytical queries.

3.2 Online Analytical Processing

We define Online Analytical Processing (OLAP) as per Definition 3 as a way to analyse multidimensional datasets .
Definition 3 (OLAP)**.** *In* Online Analytical Processing *(OLAP), domain experts issue analytical queries over data cubes according to an MDM (Definition 2) using OLAP operations Projection, Slice, Dice, Roll-Up, and Drill-Across from an OLAP algebra.*

OLAP fulfils static and dynamic requirements for data analysis [WB97]: The static aspect of analysis – an object for analysis and a collection of variables – is provided by facts and dimensions. The dynamic aspect of analysis – analytical activities such as forecasting, comparing, ranking, aggregation, and filtering – are provided by OLAP operations.

OLAP [CCS93] also is a useful way to issue analytical queries for the following reasons: 1) OLAP supports the information seeking mantra of "overview first, zoom and filter, then details-on-demand" [PMT08]. 2) Users do not need to learn a complicated

query language since OLAP clients provide intuitive user interactions over so-called pivot tables; for instance, an aggregation can be issued by removing a dimension from a pivot table. 3) There are OLAP systems available for reuse. 4) On top of OLAP, more complex analyses are possible; for instance, OLAP can be used as a way to select and pre-process data to be inserted into machine learning algorithms.

OLAP queries are complex to evaluate; for efficient query execution, data cubes are stored and accessed from data warehouses, e.g., relational databases with star schemas, to be executed efficiently [HRU96] with OLAP systems.

The typical architecture of an OLAP system consists of an *ETL* pipeline that extracts, transforms and loads (ETL) data from the data sources into a data warehouse, e.g., a relational or multidimensional database. OLAP clients allow users to build OLAP queries and display multidimensional results in pivot tables. OLAP engines transform OLAP queries into queries to the data warehouse, and deploy mechanisms for fast data cube computation and selection, under the additional complexity that data in the original data sources as well as the typical query workload may change dynamically [MKIK07, GCB+97].

In the following, we give more details about the OLAP operations and the components of OLAP systems, including a definition of a pivot table and a description of a quasi-standard query language for OLAP queries, MDX.

3.2.1 OLAP Operations

In this section we define analytical queries over multidimensional datasets as used in this thesis.

In the previous section, we have defined multidimensional datasets as data cubes in a multidimensional data model and described how they are typically stored in relational databases using a star schema. Typically, a more abstract query formalism is helpful to allow humans to issue queries directly [BJK+12] or machines to optimise query execution [Cha98].

We distinguish *metadata queries* and analytical *OLAP queries* over data cubes. Whereas metadata queries return multidimensional objects such as the cube schema, the dimensions, and the measures, OLAP queries on a certain data cube return facts, possibly represented as tuples from a relation describing the data in a dataset.

Analytical queries (OLAP queries) substantially differ from transaction processing queries (also known as OLTP queries) [BPZ11, SBC+07]. Therefore, systems to execute analytical queries and systems to execute transaction processing queries fulfil complementary requirements as summarised in Table 3.1 [KSS12, p. 7]:

Table 3.1: Complementary requirements of OLTP systems and OLAP systems [KSS12, p. 7].

Requirement	OLTP system	OLAP system
Data sources	mostly one	many (e.g., all company data)
Data volume	MB to GB	GB to TB to PB
Data characteristic	current, detailed, primary data	historical, summarised, derived, integrated
Query types	read, write, update, delete	read, periodical inserts, no updates and deletes
Query characteristics	simple, short, atomic	complex, ad-hoc, aggregating, ordering, filtering

In the following, we describe differences between an OLAP algebra over data cubes and relational algebra, define an OLAP query as a nested set of operations from the OLAP algebra and describe a common query language, MDX, for issuing OLAP queries.

Whereas a query in *SQL*, the declarative query language for *relational database management systems* (RDBMS), is a nested set of relational algebra operations, an OLAP query (e.g., described in a language such as MDX) is a nested set of OLAP operations from an OLAP algebra.

Both relational and OLAP algebra allows us to form expressions of arbitrary complexity by applying operations to the result of other operations. Algebraic expressions can be represented in an expression tree or a special syntax.

A query asked by a user may have many equivalent expressions. A query optimiser creates an expression that can be evaluated efficiently.

The relational algebra [GMUW08] includes: set operations, operations to remove parts of a relation (selection, projection), operations that combine tuples of two relations (cartesian product, join), renaming operations to change the relation schema such as names of attributes and the name of relations.

OLAP operations include: set operations (union, intersection, difference), operations to remove parts of a data cube (projection, dice, slice, roll-up), operations that combine facts of two cubes (drill-across).

In the following, we show how to issue *OLAP queries* (also called *analytical queries* or ad-hoc decision support queries) over data cubes. We define common OLAP operations on single data cubes [RA07b, PMT08, PGSJ09, RMA+11].

Figure 3.3 illustrates the effect of common OLAP operations, with inputs and outputs. For instance, in the illustration, Projection has a cube and the cube's first measure as input, and has as output the input cube with the second measure removed.

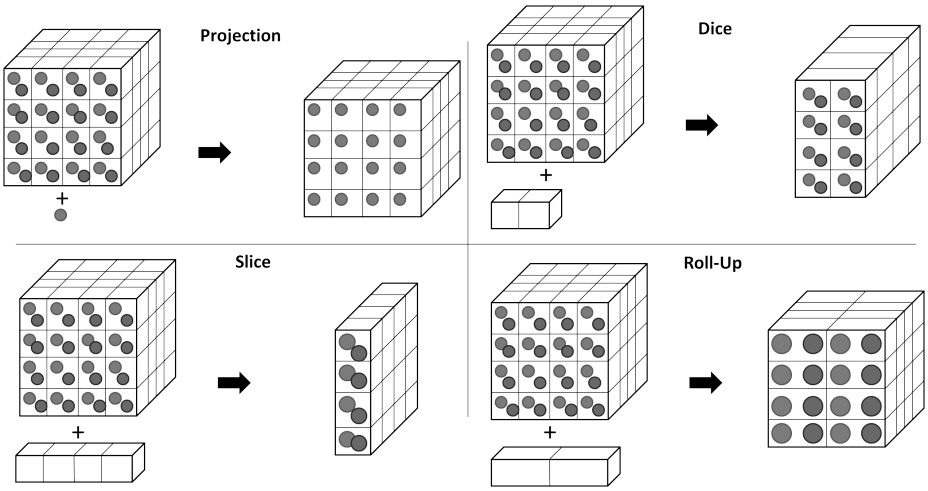


Figure 3.3: Illustration of common single-cube OLAP operations; for each operation, input is a cube and possibly multidimensional elements, output is a modified cube (adapted from [RMA+11]).

Intuitively, *projection* has as input a cube and a set of "projected" measures and has as output the input cube with any measure removed that is not "projected". Accordingly, *slice* allows for removing dimensions, *dice* for filtering of certain facts, and *roll-up* to aggregate facts to a higher level.

In the following, we give a more formal definition for each these common single-cube operations. Also, we define the drill-across operation over multiple data cubes.

We use simple set notation as syntax and set theory as semantics [HKR09, p. 363ff] to define operations. For instance, projection is defined as a function with input an instance of a data cube and a set of measures and with output an instance of a data cube.

Projection is defined as $Projection : DataCube \times 2^{Measure} \rightarrow DataCube$ and selects measures from the input cube; all non-selected measures are removed from the cube. $Projection(c, PM) = c'$ with $c = (\text{"}c\text{"}, (\text{"}cs\text{"}, D, M), F)$, $c' = (\text{"}c'\text{"}, (\text{"}cs'\text{"}, D', M'), F')$, $D' = D$, $M' = M \backslash PM$.

Dice is defined as $Dice : DataCube \times Dimension \times 2^{Member} \rightarrow DataCube$ and allows to filter for facts with certain dimension member combinations (so called *positions*). Since facts are filtered, we assume no change for the schema of the resulting cube.

Slice is defined as $Slice : DataCube \times 2^{Dimension} \rightarrow DataCube$ and removes dimensions from the input cube, i.e., aggregates all facts over those dimensions and

removes those dimensions. $Slice(c, SD) = c'$ with $c = (\text{"}c\text{"}, (\text{"}cs\text{"}, D, M), F)$, $c' = (\text{"}c'\text{"}, (\text{"}cs'\text{"}, D', M'), F')$, $D' = D \backslash SD$, $M' = M$.

Roll-Up is defined as $Roll-Up : DataCube \times Dimension \times Level \rightarrow DataCube$ and allows to create a cube that contains instance data on a higher aggregation level.

The chosen level needs to be set higher than the current level of the data cube since there is no way to compute more granular from less granular values. Setting a lower level, i.e., doing a $Drill-Down$ we do not consider in our algebra. By retaining cubes at lower granularities after a roll-up, a drill-down can be achieved by doing a roll-up on the respective detailed cube. Since there is only a change of data in the cube, we assume no change for the schema of the resulting cube.

Drill-Across is defined as $Drill-Across : DataCube \times DataCube \rightarrow DataCube$ and allows to combine instance data from two cubes into a new cube, as follows: $Drill-Across(c_1, c_2) = c'$ with $c_1 = (\text{"}c_1\text{"}, (\text{"}cs_1\text{"}, D_1, M_1), F_1)$, $c_2 = (\text{"}c_2\text{"}, (\text{"}cs_2\text{"}, D_2, M_2), F_2)$, $c' = (\text{"}c'\text{"}, (\text{"}cs'\text{"}, D', M'))$, $D' = D_1 \cup D_2$, $M' = M_1 \cup M_2$.

Drill-across returns a cube with the union of all dimensions and measures and computes a join of facts on the dimensions. Typically, for drill-across, it is assumed that both input cubes contain the same dimensions and different measures [KR02]. Then, drill-across allows to compare measures from two or more cubes.

Every OLAP operations has as input a Data Cube (two in case of Drill-Across) and returns a Data Cube; thus, operations can be nested, e.g., `Dice(Projection(...))`. An OLAP query is thus a *nested set of OLAP operations*.

Whereas Projection, Dice, Slice, and Roll-Up operate over single data cubes, Drill-Across allows for the combination of multiple data cubes. Other multi-cube operations such as Union and Intersection of two cubes (requires identical dimensions and measures) could be defined accordingly, but were not required by our scenarios.

3.2.1.1 OLAP Operations over Example MDM and Star Schema

In the following we give a concrete example of operations issued over the population dataset modelled using our MDM. To make the semantics of OLAP operations explicit, we explain how to execute the operations using SQL over a star schema.

We assume no redundant facts stored for the data cube, i.e., contained facts all have the same granularity level such as years and all higher level facts can be computed by aggregation. Otherwise, queries would need to consider aggregate facts to ensure summarisability [CD97].

A special operation used in the queries is *group by*. Group by partitions a table into groups according to *grouping values*. Every group must then be aggregated by an aggregation function and given an *aggregate value* [GCB+97].

In the following, we execute OLAP operations over the population data cube defined in Section 3.1. For multi-cube queries (Drill-Across), we also assume a second data cube with the "Regional gross domestic product (million PPS) by NUTS 2 regions" published by Eurostat[4]. Thus, we assume an MDM with `DataCube = {populationCube, gdpCube}`.

With Projection, the grouping values are not changed:

`Projection(populationCube, {populationMeasSUM})`

```
SELECT timeLevelYear geoLevelNUTS0 sexLevel ageLevel
    SUM(populationMeas) as populationMeasSUM
FROM populationCubeT, timeDimT, geoDimT, sexDimT, ageDimT
WHERE
populationCubeT.timeID = timeDimT.timeID
AND populationCubeT.geoID = geoDimT.geoID
AND populationCubeT.sexID = sexDimT.sexID
AND populationCubeT.ageID = ageDimT.ageID
GROUP BY timeLevelYear geoLevelNUTS0 sexLevel ageLevel
```

In the SQL, the fact table (`populationCubeT`) is joined with the dimension tables and the projected measures selected.

Similarly, with Dice, the grouping values are not changed. The level to do the dicing on is determined by the members in the first position, `DE.level = geoLevelNUTS0` in this case:

`Dice(populationCube, geoDim, {DE, ES})})`

```
SELECT timeLevelYear geoLevelNUTS0 sexLevel ageLevel
    SUM(populationMeas) as populationMeasSUM AVG(populationMeas) as
    populationMeasAVG COUNT(populationMeas) as populationMeasCOUNT
FROM populationCubeT, timeDimT, geoDimT, sexDimT, ageDimT
WHERE
populationCubeT.timeID = timeDimT.timeID
AND populationCubeT.geoID = geoDimT.geoID
AND populationCubeT.sexID = sexDimT.sexID
AND populationCubeT.ageID = ageDimT.ageID
AND (geoLevelNUTS0 = "DE" OR geoLevelNUTS0 = "ES")
GROUP BY timeLevelYear geoLevelNUTS0 sexLevel ageLevel
```

Slice removes the grouping value of the respective dimension:

`Slice(populationCube,{ageDim})`

```
SELECT timeLevelYear geoLevelNUTS0 sexLevel SUM(populationMeas) as
    populationMeasSUM AVG(populationMeas) as populationMeasAVG
    COUNT(populationMeas) as populationMeasCOUNT
FROM populationCubeT, timeDimT, geoDimT, sexDimT
WHERE
populationCubeT.timeID = timeDimT.timeID
```

[4] http://appsso.eurostat.ec.europa.eu/nui/show.do?dataset=tgs00004&lang=en, last accessed on 2015-06-13.

```
5  AND populationCubeT.geoID = geoDimT.geoID
6  AND populationCubeT.sexID = sexDimT.sexID
7  GROUP BY timeLevelYear geoLevelNUTS0 sexLevel
```

Roll-Up changes the grouping value of the respective dimension to another level:

`Roll-Up(populationCube, sexDim, sexLevelTotal)`

```
1  SELECT timeLevelYear geoLevelNUTS0 sexLevelTotal ageLevel
       SUM(populationMeas) as populationMeasSUM AVG(populationMeas) as
       populationMeasAVG COUNT(populationMeas) as populationMeasCOUNT
2  FROM populationCubeT, timeDimT, geoDimT, sexDimT, ageDimT
3  WHERE
4  populationCubeT.timeID = timeDimT.timeID
5  AND populationCubeT.geoID = geoDimT.geoID
6  AND populationCubeT.sexID = sexDimT.sexID
7  AND populationCubeT.ageID = ageDimT.ageID
8  GROUP BY timeLevelYear geoLevelNUTS0 sexLevelTotal ageLevel
```

To evaluate Drill-Across, the union of dimensions of the single cubes constitute the grouping values. Drill-Across brings together the measures of the two cubes and allows direct comparison. Here, we query over our population and GDP cubes.

`Drill-Across(populationCube, gdpCube)`

```
1  SELECT timeLevelYear geoLevelNUTS0 sexLevelTotal ageLevel unitLevel
       SUM(populationMeas) as populationMeasSUM AVG(populationMeas) as
       populationMeasAVG COUNT(populationMeas) as populationMeasCOUNT
       SUM(gdpMeas) as gdpMeasSUM AVG(gdpMeas) as gdpMeasAVG
       COUNT(gdpMeas) as gdpMeasCOUNT
2  FROM populationCubeT, gdpCubeT, timeDimT, geoDimT, sexDimT,
       ageDimT, unitDimT
3  WHERE
4  populationCubeT.timeID = timeDimT.timeID
5  AND gdpCubeT.timeID = timeDimT.timeID
6  AND populationCubeT.geoID = geoDimT.geoID
7  AND gdpCubeT.geoID = geoDimT.geoID
8  AND populationCubeT.sexID = sexDimT.sexID
9  AND gdpCubeT.sexID = sexDimT.sexID
10 AND populationCubeT.ageID = ageDimT.ageID
11 AND gdpCubeT.ageID = ageDimT.ageID
12 AND populationCubeT.unitID = unitDimT.unitID
13 AND gdpCubeT.unitID = unitDimT.unitID
14 GROUP BY timeLevelYear geoLevelNUTS0 sexLevelTotal ageLevel
       unitLevel
```

To result in a non-empty data cube, the standard definition of drill-across [KR02] requires that the two input data cubes share all dimensions. However, `gdpCube` does not have the dimensions `sexDim` and `ageDim` and `populationCube` does not have the dimension `unitDim`. Thus, the resulting cube in our example will be empty, i.e., the fact table would be empty (*empty cube*). Also, whereas the GDP is available on a regional level, the population is only available on a country level.

OLAP operations can be nested to create data cubes that share all dimensions, and are on the same level of detail. For instance, the following nested set of OLAP operations can result in a non-empty data cube, given that `populationCube` and `gdpCube` contain information about the same years and countries. We use a function-type syntax to denote the iterative application of OLAP operations:

```
Drill-Across(
        Slice(populationCube, {sexDim, ageDim}),
        Roll-Up(Slice(gdpCube, {unitDim}), geoDim, geoLevelNUTS0)
)
```

The sex and age dimension are removed from the population cube. The unit dimension is removed from the GDP cube and the cube is aggregated to a country level. The resulting cubes share all dimensions and levels and are joined with drill-across.

The respective SQL query is shown in the following:

```
SELECT timeLevelYear geoLevelNUTS0 SUM(populationMeas) as
    populationMeasSUM AVG(populationMeas) as populationMeasAVG
    COUNT(populationMeas) as populationMeasCOUNT SUM(gdpMeas) as
    gdpMeasSUM AVG(gdpMeas) as gdpMeasAVG COUNT(gdpMeas) as
    gdpMeasCOUNT
FROM populationCubeT, gdpCubeT, timeDimT, geoDimT
WHERE
populationCubeT.timeID = timeDimT.timeID
AND gdpCubeT.timeID = timeDimT.timeID
AND populationCubeT.geoID = geoDimT.geoID
AND gdpCubeT.geoID = geoDimT.geoID
GROUP BY timeLevelYear geoLevelNUTS0
```

Next, we describe main components of systems that provide OLAP functionalities, including an OLAP language and result format (pivot tables).

3.2.2 OLAP Systems

OLAP systems feature a client-server architecture with one or more OLAP clients accessing a designated OLAP engine [CCS93, CD97]. An OLAP engine allows exploration of datasets accessible from a Data Warehouse.

A *data warehouse* is concerned with providing efficient access to heterogeneous datasets in a unified schema. The data is modelled and stored for efficient analytical queries, e.g., using the *star schema* [CD97]. *Extract-Transform-Load* (ETL) pipelines extract, transform, and load datasets.

A typical ETL system consists of *wrappers* and *mediators* [CDL$^+$01]. A wrapper accesses a source, extracts the relevant data, and presents such data in a specific format. The mediator collects, cleans, and combines data produced by different wrappers to a unified schema.

OLAP engines (also called *OLAP servers*) translate OLAP queries into a target query language of a data warehouse storing the multidimensional data, e.g., the star schema in a relational database management system. OLAP engines include *Palo OLAP Server*[5] and *Mondrian*[6].

OLAP clients use a common language for querying an MDM and visualise results in pivot tables. OLAP clients include *JPivot*[7], *Palo Client*[8], and *Saiku*[9].

Also, there are programming libraries to build own OLAP servers (e.g., the Open Java API for OLAP *olap4j*[10]) and OLAP clients (e.g., *xmla4js*[11]).

OLAP clients communicate OLAP queries via a common OLAP query language such as MDX. In the following, we introduce MDX as a declarative OLAP query language to display data from a data cube in pivot tables.

Multidimensional Expressions (MDX) is the most widely used OLAP query language, adopted by OLAP engines such as Microsoft SQL Server[12] and Mondrian, programming libraries such as olap4j, and protocols such as *XML for Analysis*[13] (XMLA).

XMLA is an XML API for the communication between OLAP engines and clients over the Web based on the Simple Object Access Protocol (SOAP). XMLA supports MDX as a query language and our MDM. XMLA is widely adopted in industry [PBAP08]. OLAP engines such as Palo OLAP Server and Mondrian provide XMLA interfaces and OLAP clients such as JPivot and Palo Client connect to XMLA interfaces.

In contrast to an SQL query that returns a relational table, an MDX query returns parts of a data cube to be displayed in a pivot table [GCB+97].

We define *pivot tables* as per Definition 4 and as illustrated in Figure 3.4.
Definition 4 (Pivot Table)**.** *Pivot tables display data from a data cube in a compact, two-dimensional, tabular form where both the number of rows and columns are variable depending on the multidimensional dataset represented in the cube [CGLG04]. The metadata of a pivot table describes a queried data cube, lists of member combinations (positions) from a fixed set of levels from different dimensions to be displayed on*

[5] http://en.wikipedia.org/wiki/Palo_%28OLAP_database%29, last accessed on 2014-11-18.
[6] http://mondrian.pentaho.com/, last accessed 2014-11-18.
[7] http://jpivot.sourceforge.net/, last accessed on 2014-06-25.
[8] http://www.palo.net/, last accessed on 2014-06-25.
[9] http://www.analytical-labs.com/, last accessed 2014-11-05.
[10] http://www.olap4j.org/, last accessed 2014-11-05.
[11] https://github.com/rpbouman/xmla4js, last accessed 2014-11-23.
[12] For an introduction to MDX, see
http://msdn.microsoft.com/en-us/library/aa216770%28v=sql.80%29.aspxand the reference at
http://technet.microsoft.com/en-us/library/ms145506.aspx, accessed 2014-10-23.
[13] http://en.wikipedia.org/wiki/XML_for_Analysis, last accessed 2014-11-18.

rows and columns, and member combinations from a fixed set of levels as filter conditions about which facts to summarise in the pivot table. The cells in a pivot table are populated with measure values of facts in the cube.

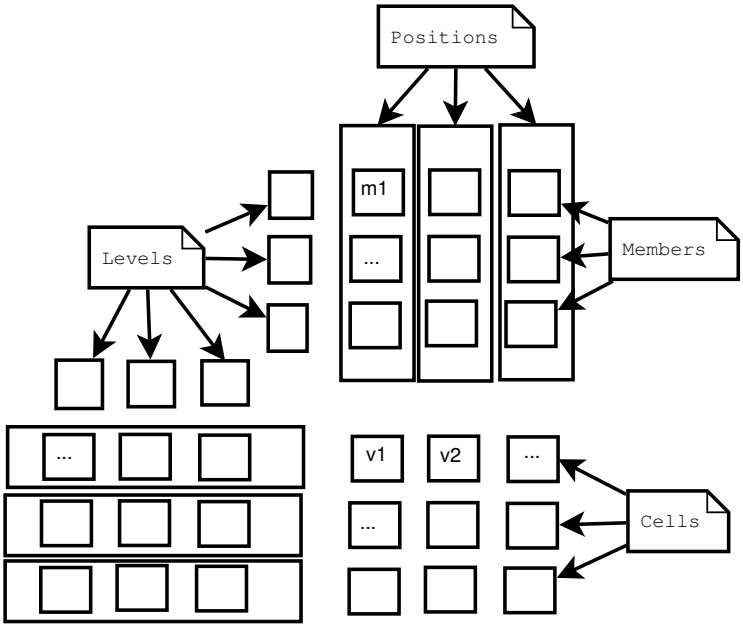


Figure 3.4: Schema of pivot table as generated by a typical MDX query; this pivot table displays six levels of different dimensions on columns and rows.

Pivot tables are provided in spreadsheet programs such as Microsoft Excel and also provide an intuitive interface for issuing OLAP (MDX) queries.

The Basic MDX Query[14] is given in Listing 1.

Listing 1: Basic MDX query.

```
1 SELECT
2 [<column_axis_specification>] ON COLUMNS,
3 [<row_axis_specification>] ON ROWS
4 FROM [<cube_specification>]
5 [WHERE [<slicer_specification>]]
```

In an MDX query, the cube specification in the FROM clause describes the queried data cube, the axis specifications describe the columns and rows, and the slicer specification describes the member combinations as filter conditions. An MDX query requests the summarised values to be displayed in the cells of the pivot table.

[14] http://msdn.microsoft.com/en-us/library/aa216770%28v=sql.80%29.aspx

As a concrete MDX query example, see Listing 2. Here, from a Eurostat dataset "Employment Rate"[15], we create a pivot table with all members of the dimension sex on columns (Female, Male, Total) and all possible member combinations (retrieved with built-in function `CrossJoin`) of dimensions location and time on rows (e.g., position AT, 2010). The built-in function `Members` returns all members of a dimension or level. In the WHERE clause, the measure is selected.

Listing 2: MDX query for Employment Rate example.

```
1 SELECT
2 {Members([sex])} ON COLUMNS,
3 CrossJoin({Members([location])}, {Members([time])}) ON ROWS
4 FROM [Employment Rate]
5 WHERE { [Measures].[obsValue] }
```

Results of MDX queries fill the cells of the pivot table. The possible result of our example query is illustrated in Figure 3.5, showing that 69.6% of women in Austria in 2010 were employed.

		F	M	T
AT	2006	66.4	80.0	73.2
	2007	67.2	81.6	74.4
	2008	68.6	81.7	75.1
	2009	69.4	80.1	74.7
	2010	69.6	80.2	74.9
	2011	69.6	80.8	75.2
	2012	70.3	80.9	75.6
	2013	70.8	80.3	75.5
BE	2006	58.8	74.0	66.5
	2007	60.3	75.0	67.7
	2008	61.3	74.7	68.0
	2009	61.0	73.2	67.1
	2010	61.6	73.5	67.6
	2011	61.5	73.0	67.3
	2012	61.7	72.7	67.2
	2013	62.1	72.3	67.2

Figure 3.5: Example pivot table with Employment Rate values per gender (Female, Male, Total) in columns and per location and time in rows.

Query processing in OLAP engines often follows the *iterator model* [Gra93]. A query represented as a nested set of OLAP operations describes a *logical operator query plan*.

[15] http://appsso.eurostat.ec.europa.eu/nui/show.do?dataset=tsdec420&lang=en, last accessed 2015-06-13.

This means that the single-cube operations projection, dice, slice, and roll-up and the multi-cube operation drill-across are logical query operators.

Query optimisation is concerned with restructuring a logical query plan so that equal results are requested but with fewer expected costs in data processing. Logical operator query plans logically describe in which order operations are executed but do not describe how the operations are executed over concrete physical data.

Therefore, a logical query plan is transformed into a *physical operator query plan*. A physical query plan consists of *iterators*. Whereas logical operators are only concerned with data manipulation, an iterator may also implement operations that do not change the data such as an iterator for loading or indexing of the data [Gra93]. An iterator contains a method `next()` to request the next result; this method will in turn issue the `next()` method of input iterators, and return (possibly modified) results. Results between iterators can be pipelined or transferred via temporary data structures such as tables.

3.3 Statistical Linked Data

We define *Statistical Linked Data* (SLD) as the main data source of multidimensional datasets as per Definition 5.
Definition 5 (Statistical Linked Data)**.** *Statistical Linked Data are RDF data with multidimensional datasets properly modelled and published as Linked Data according to the RDF Data Cube Vocabulary.*

In the following, we explain Linked Data and the RDF Data Cube Vocabulary.

3.3.1 Linked Data

Linked Data (LD) is data published according to the *Linked Data principles*[16], a set of best practices widely-adopted within the Semantic Web community. The main technological building blocks are Web standards HTTP and URI. The Hypertext Transfer Protocol (HTTP) is the basic protocol for transfer of data on the Web. Uniform Resource Identifiers (URIs) allow to identify things based on the Web.

Applied to statistics, Linked Data recommends the following:

1. HTTP URIs are used to identify things such as datasets, dimensions, statistical indicators, countries, and companies.

[16] http://www.w3.org/DesignIssues/LinkedData.html, last accessed on 2014-10-23.

2. HTTP URIs are resolvable and provide useful, machine-readable information based on Semantic Web technologies.
3. Useful information links to other things by reusing HTTP URIs such as used by others for dimensions, indicators, countries, and companies.

More concretely, for resolvable HTTP URIs, Linked Data distinguishes between non-information HTTP URIs that refer to a thing in the real world and information HTTP URIs that provide information in a human- or machine-readable form about a thing. Resolveable URIs identify things and – if looked-up via HTTP – provide useful information. The client application can request different serialisations of the same information, among them human-readable or machine-readable formats. The webserver returns the data in the format or directs to the appropriate information HTTP URIs[17].

In this work, we often abbreviate URIs with well-known prefixes as listed by *prefix.cc*[18]. We slightly abuse the W3C CURIE syntax for expressing compact URIs.

Whereas for the human, mostly HTML is returned, Linked Data recommends to return information represented using the *Resource Description Framework*[19] (RDF). Data formats for RDF include RDF/XML, Turtle, and JSON-LD.

An RDF document contains an RDF graph made of RDF triples as defined in Definition 6.

Definition 6 (RDF Graph with Terms and Triples)**.** *The set of terms in an RDF graph consists of the set of HTTP URIs $\mathcal{I}$, the set of blank nodes $\mathcal{B}$ and the set of literals $\mathcal{L}$. A triple $(s,p,o) \in \mathcal{T} = (\mathcal{I} \cup \mathcal{B}) \times \mathcal{I} \times (\mathcal{I} \cup \mathcal{B} \cup \mathcal{L})$ is called an RDF triple, where s is the subject, p is the predicate and o is the object.*

In the following, we show two example RDF triples in Turtle format describing that `eurostat-pjan:ds`[20] is a dataset and has a certain label:

```
eurostat-pjan:ds rdf:type qb:DataSet.

eurostat-pjan:ds rdfs:label "GDP and main components –
Current prices".
```

SPARQL[21] is the standard query language for RDF data. We can distinguish *SPARQL SELECT queries* that return results in a tabular representation, *SPARQL CONSTRUCT queries* that return results as RDF, and *SPARQL ASK queries* that return a Boolean result whether the query was matched in the underlying dataset or not. SPARQL queries use *graph patterns*, sets of triples that each also can contain variables such as `?dataset`

[17] Two common implementations are "fragment identifiers" and "HTTP Status Code 303 See Other", https://en.wikipedia.org/wiki/HTTPRange-14, last accessed 2014-10-30.
[18] http://prefix.cc/; last visited on 2014-04-03.
[19] http://www.w3.org/TR/ld-glossary/#resource-description-framework-rdf, last accessed on 2014-10-23.
[20] http://estatwrap.ontologycentral.com/id/demo_pjan#ds, last accessed 2014-11-06.
[21] http://www.w3.org/TR/sparql11-overview/, last accessed 2015-01-30.

to denote placeholders to which terms of triples from an RDF graph need to bind to find a match in SPARQL query processing. For instance, only the first triple of the example above would bind to the following graph pattern: `{?dataset rdf:type qb:DataSet. }`. The new version SPARQL 1.1 includes functionalities such as aggregation functions and subqueries.

We refer to *SPARQL engines* as systems that provide SPARQL query capabilities over specific RDF graphs. This includes *triple stores* with SPARQL endpoints such as *Open Virtuoso*[22] and *Sesame*[23] as well as Linked Data query systems such as *qcrumb.com*[24] that directly allow queries over RDF graphs without permanently storing the data.

In Linked Data, *RDF vocabularies* and *ontologies* provide URIs that can be reused in RDF graphs to describe certain domains. Special RDF vocabularies [PHDU13] that most other vocabularies are based on include the *Web Ontology Language* (OWL)[25] and *RDF Schema* (RDFS)[26]. They introduce classes (`rdfs:Class`, `owl:Class`) and properties (`rdf:Property`, `owl:ObjectProperty`, `owl:DatatypeProperty`) with logic-based semantics that allow to infer implicit triples, and that SPARQL engines may consider in query processing. For instance, RDFS allows for defining sub-classes (`rdfs:subClassOf`) and sub-properties (`rdfs:subPropertyOf`); OWL allows for setting two things equivalent (`owl:sameAs`). The following triple states that Germany defined by Eurostat, `eurostat-geo:DE`[27], is equal to Germany defined by Gesis, `allbus-geo:00`[28]:

```
eurostat-geo:DE owl:sameAs allbus-geo:00.
```

In this work, we assume that multidimensional datasets are originally published using Linked Data or are re-published using a *Linked Data wrapper*. Examples of Linked Data Wrappers include *DBpedia*[29] that extracts structured information from *Wikipedia* and publishes the data as Linked Open Data[30].

In the next section, we present a Linked Data vocabulary that we use in the thesis for representing multidimensional datasets.

[22] http://sourceforge.net/projects/virtuoso/files/virtuoso/, last accessed 2014-11-17.
[23] http://sourceforge.net/projects/sesame/files/Sesame%202/, last accessed on 2014-11-17.
[24] http://qcrumb.com/, last accessed on 2014-10-23.
[25] http://www.w3.org/TR/2012/REC-owl2-overview-20121211/, last accessed on 2014-06-29.
[26] http://www.w3.org/TR/rdf-schema/, last accessed on 2014-06-29.
[27] http://estatwrap.ontologycentral.com/dic/geo#DE, last accessed 2014-11-06.
[28] http://lod.gesis.org/lodpilot/ALLBUS/geo.rdf#00, last accessed 2014-11-06.
[29] http://dbpedia.org/About, last accessed 2014-11-11.
[30] Referred to as Linked Open Data since published with an open licence.

3.3.2 The RDF Data Cube Vocabulary

The RDF Data Cube Vocabulary (QB)[31], is a widely-used source of URIs to describe multidimensional datasets. We have contributed to the recommendation of QB by the W3C with a description of use cases and lessons for the vocabulary [KC13]. See Figure 3.6 for an overview of QB with the most important classes and properties.

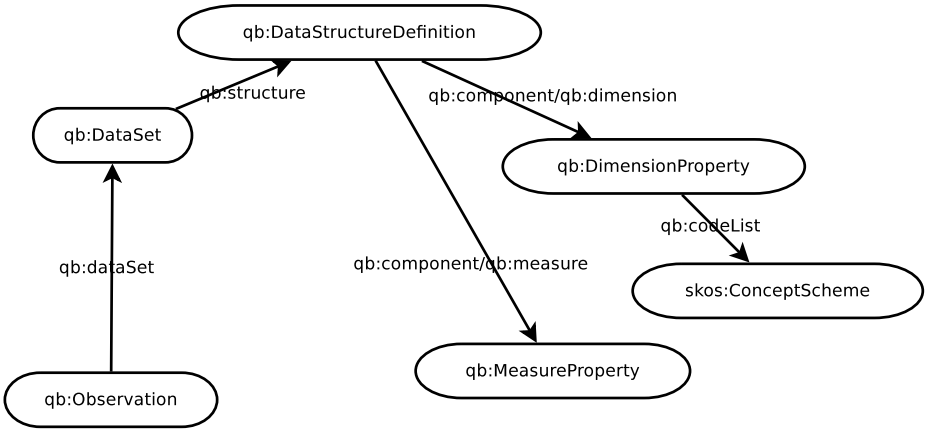


Figure 3.6: Illustration of most important classes of The RDF Data Cube Vocabulary with properties (or property chains) between instances of concepts; adapted from "Outline of the vocabulary" in specification.

QB allows to describe datasets (instances of `qb:DataSet`) with observations (instances of `qb:Observation`). Every dataset has a certain structure (instance of `qb:DataStructureDefinition`) that – using a chain of properties `qb:component` before `qb:measure` or `qb:dimension` – defines measures (instances of `qb:MeasureProperty`) and dimensions (`qb:DimensionProperty`). 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 (e.g., instances of `skos:Concept` in a `skos:ConceptScheme` linked from the dimension via `qb:codeList`). The following triples describe an example observation as a *blank node*, an instance with an only locally known name, of 390,156 inhabitants of 18-year-old female persons in 2013 in Germany in the population dataset of Eurostat:

```
1 _:obs1 a qb:Observation;
2     qb:dataSet eurostat-pjan:ds;
3     estatwrap:age eurostat-age:Y18;
4     estatwrap:sex eurostat-sex:F;
```

[31] http://www.w3.org/TR/vocab-data-cube/, last accessed on 2014-10-23.

```
5       estatwrap:geo eurostat-geo:DE;
6       dcterms:date "2013";
7       sdmx-measure:obsValue "390156".
```

The following SPARQL query returns the number of observations in the population dataset:

```
1  SELECT count(?obs)
2  WHERE {
3  ?obs qb:dataSet eurostat-pjan:ds.
4  }
```

A QB dataset serves all necessary information about a multidimensional dataset. The QB dataset URI gives the name of the relation defined by the multidimensional dataset. The QB data structure definition defines the independent and dependent attributes of the relation. The QB observations describes the entities in the relation.

QB 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.

We have chosen QB among several other vocabularies available to publish raw or aggregated multidimensional datasets. For instance, there are various other OWL ontologies available for representing multidimensional datasets [NN09]. Also, various lightweight ontologies, Linked Data vocabularies, have been presented such as SCOVO [HHR+09] and SCOVOLink [VLH+10].

3.3.2.1 Reasons for Choosing QB for Modelling Multidimensional Datasets

Our reasons for using QB for modelling multidimensional datasets are as follows:

1) In contrast to other vocabularies, QB is widely-adopted, an important factor for data integration use cases. For instance, see the PlanetData wiki for a collection of data sources using QB[32].

There are already several statistical datasets published in QB format, e.g., financial reports from the U.S. Security and Exchange Commission[33] and spendings by the UK district council Lichfield[34].

Also, as a pre-condition for recommendation by the W3C, several implementations of QB have been identified[35]. OLAP2DataCube and CSV2DataCube support publishers to transform common representations of statistics to QB [RPM+13, SMDMM+12].

[32] http://wiki.planet-data.eu/web/Datasets, last accessed on 2014-10-23.
[33] http://edgarwrap.ontologycentral.com/, last accessed 2014-11-11.
[34] http://spending.lichfielddc.gov.uk/download, last accessed 2014-11-23.
[35] http://www.w3.org/2011/gld/wiki/Data_Cube_Implementations, last accessed 2014-11-11.

An earlier version of QB [CFG+10] existed for some time and proved applicable in several deployments. QB is based on the Statistical Data and Metadata Exchange (*SDMX*)[36] which is applied in many contexts. QB has shown flexible to represent different kinds of multidimensional datasets, e.g., sensor data, finance data, and statistics [KC13].

2) QB promises benefits in various use cases [KC13]. Statistics can more easily be disseminated since registries directly get their metadata from the published datasets instead of from separately created and maintained documents. A standard representation such as Linked Data and QB makes it easier to work with the data, to annotate the data and to consume the data with third-party tools. Various data formats such as for Microsoft Excel spreadsheets can be translated to a machine-readable format so that search engines can make better use of the data. Data can automatically be aggregated to higher levels and stored so that queries may be answered by fast look-ups. Publishers may internally only manage the canonical data model using QB and provide the data in other formats on-the-fly. This way, a publisher can meet the needs of many data consumers in a uniform way.

Other benefits include:

- Since the data is self-describing, contains links to related information and the use of well-defined ontologies, Linked Data allows for easier automatic interpretation of data.
- Existing Linked Data tools such as crawlers can be used for extract-transform-load pipelines.
- SPARQL 1.1 allows expressive queries after loading of RDF into a triple store.
- Other information can be added, e.g., licence information; all in a uniform representation using RDF and using different vocabularies.

Etcheverry and Vaisman [EV12a] specifically explain the benefits of having a data structure definition of multidimensional datasets:

- to assist in the integration process of different data sets;
- to provide a self-contained description of the contents of the data set, which allows for example to implement client applications and operators;
- and to verify that the dataset instances match the expected structure.

3) QB fulfils requirements for sharing of data cubes [HBH03].

- The format has to support a conceptual model of data cubes. In Chapter 5, we show a formal mapping between our MDM and QB.

[36] http://sdmx.org/, last accessed 2014-11-23.

- The conceptual distinction between the description of "schema, master or dimension data" and "transaction or fact data" has to be supported. QB distinguishes `qb:DataStructureDefinition` with `qb:DimensionProperty` for the schema and `qb:DataSet` with `qb:Observation` for fact data that in Linked Data also can be provided at different locations.
- The format has to be transportable over a network, primarily over the Internet. As natively supported by Linked Data, datasets are shared using HTTP and URIs.
- To achieve a high level of flexibility and reuse, the format has to support linking and inclusion concepts. Other formats such as XCube [HBH03] at most highlight the development of standardised reference dimensions and allow the linkage of schema, dimension and fact data; beyond that, Linked Data and QB allow reuse and linking of any URI available on the Web.
- The format should be extensible to be able to adapt to different data models or to introduce new concepts. Etcheverry and Vaisman [EV12b, EV12a] show possible extensions of QB.
- The format must be easily convertible to and from various data sources and formats. RDF/XML representations of QB data allow XSLT transformations. In several case studies we have shown that QB can be used to represent heterogeneous data sources and formats, e.g., SDMX, XBRL, sensor data [KC13]. Converting QB data to other RDF schemas, and from there to other data formats, is also possible using SPARQL 1.1 CONSTRUCT or SELECT queries.
- The format must allow Online Analytical Processing (OLAP) and reduce the amount of data to be transferred over the network. Etcheverry and Vaisman [EV12b, EV12a] have shown how to merge locally stored datasets and Web-distributed datasets that are represented with an extension of QB using OLAP operations. In Chapter 6, we show how to translate OLAP operations to SPARQL queries over QB datasets.

4) QB allows to ground the semantics of statistics and still is easy to use. Since QB does not directly support all necessary components of complex multidimensional models, e.g., complex OLAP hierarchies and aggregation functions [EV12a], QB has proven to be a suitable compromise between expressivity and simplicity for publishing statistics as Linked Data.

The Statistical Core Vocabulary (SCOVO)[37] [HHR+09] was a first attempt to cater for the specificities of multidimensional datasets when publishing as Linked Data. The main drawback of SCOVO is that only `svc:dimension` is used and different dimensions and measures are not distinguished; thus, SCOVO provides only limited capabilities to ground the semantics of statistics. Vrandečić et al. [VLH+10] and Cyganiak et al. [CFG+10] recommend to extend SCOVO.

[37] http://purl.org/NET/scovo, last accessed 2014-11-11.

QB intends to be easier to handle and better able to capture the semantics of the statistical Linked Data than SCOVO. Also, datasets using SCOVO vocabulary can be re-expressed in the QB vocabulary. QB intends to be generally suitable to model many different kinds of numeric datasets.

Most importantly, QB distinguishes not only dimension values but also dimension types and therefore fulfils the requirements that Vrandečić et al. [VLH+10] state for grounding statistics to their described entities.

5) QB is an OWL ontology and as such has a formal domain model that helps with automatic data integration. Other works specifically aim at sharing descriptions between systems. The Common Warehouse Metamodel (CWM)[38] is a standard by the Object Management Group (OMG) for data warehousing. CWM is an XML-based exchange standard for ETL transformations and data warehouse metadata.

However, CWM – but also other interfaces and protocols to share multidimensional datasets such as XML for Analysis and OLE DB – lack a solid theoretical background making it more difficult to use such formalism as a basis for integration [VS99].

Sharing multidimensional datasets often is done using XML [PBAP08]. XML allows to define a schema (XML Schema). There are data modification and query languages for XML such as XSLT and XQuery. There are XML schemas for representing specific information, e.g., XBRL for financial reports, SDMX for statistics, DDI[39] for research studies. Financial reports contain financial facts for a company, for a valid period and for a financial concept such as Total Assets. Statistics contain (aggregated) macro data, e.g., about countries, in a specific year, for a specific group of people. In research studies micro data is collected, e.g., answers by a specific type of person, on a date, to a certain question in questionnaires.

The integration of data across different standards is still an open issue. 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 [KFvHH01].

A domain model for instance includes a top-down inheritance of attributes from superclasses to subclasses. Assume employee is a subclass of a class person. Then any employee inherits all attributes that are defined for the class person. Also, the bottom-up inheritance of instances from subclasses to superclasses. Then the class person inherits all instances that are elements of the class employee. Another example of a domain model element are logical axioms such as equivalance axioms.

[38] http://www.omg.org/spec/CWM/, last accessed 2014-11-11.
[39] http://www.ddialliance.org/, last accessed 2014-11-23.

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.

We regard Statistical Linked Data, Linked Data using QB, as the primary source of multidimensional datasets in this work.

4 State of the Art

Chapter 5 to Chapter 8 include descriptions of specific related work related to the single research questions. In this chapter we give an overview of designed and implemented systems that can provide a solution to our overall research problem:

Overall Research Problem: *How can we flexibly integrate and efficiently analyse multidimensional datasets from the Web?*

We argue that most systems are either flexible to integrate heterogeneous datasets or efficient to do so on Web-scale. In Figure 4.1, we categorise integration and analysis approaches over datasets from the Web and identify a gap of combined approaches.

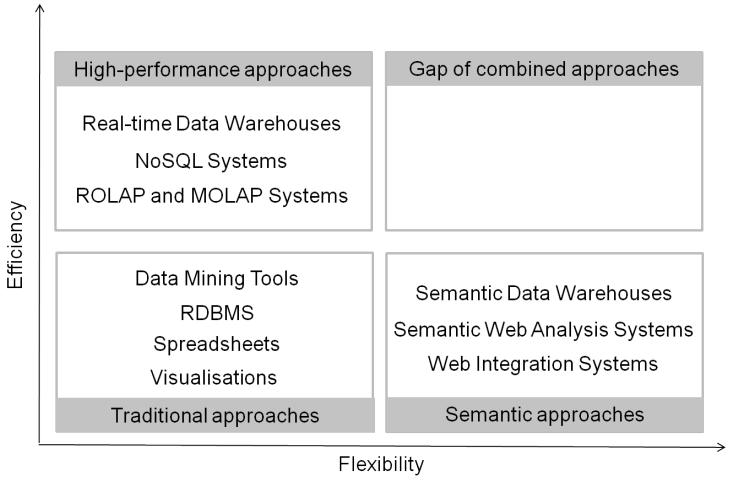


Figure 4.1: Integration and analysis approaches over datasets from the Web, categorised by flexibility (x-axis) and efficiency (y-axis).

Traditional approaches fulfil specific requirements in data pre-processing and analysis and are not designed to integrate heterogeneous datasets of large sizes and numbers from the Web.

Semantic approaches make use of Semantic Web technologies or other schema-flexible methods and have focussed so far on semi-structured metadata instead of large amounts of numeric values from multidimensional datasets.

High-performance approaches focus more on scalability and still require manual effort for designing and maintaining of ETL pipelines to pre-process, integrate, and load heterogeneous data. In the following, we describe example systems for each category.

4.1 Traditional Integration and Analysis Approaches

Traditional approaches are often used to pre-process and analyse multidimensional datasets. With respect to the integration of datasets from the Web, they do not focus on flexibility and efficiency but knowledge discovery over a single pre-processed, multidimensional dataset of limited size.

4.1.1 Visualisations

Effective visualisations for specific domains, e.g., healthcare, exist [KKEM10, p. 32]. Visualising domain-independent multidimensional data, e.g., using parallel coordinates[1] easily becomes confusing to analyse. Therefore, most visualisations are limited to a certain number of dimensions to display at the same time. An example is the *Gapminder* system [KKEM10, p. 32] that allows to visualise up to five dimensional and measurement attributes (x- and y-axis; color, size of circles; movement over time). Usually, such systems require cleaned data.

4.1.2 Spreadsheets

Microsoft Excel with its capability to create Pivot tables is one of the most widely-used spreadsheet programs. Spreadsheets are intuitive to create and work with small datasets. If datasets grow, come from different sources and may dynamically change, the manual effort in maintaining spreadsheets increases. The ease of adapting spreadsheets to one's needs such as adding nested tables, colours and formulas increases the risks of copy-paste errors and inconsistencies. As an example, the Economist spread

[1] http://en.wikipedia.org/wiki/Parallel_coordinates, last accessed 2015-01-30.

false information about countries in an article about their economic vulnerability that incurred from errors in spreadsheets[2].

The *Anzo* tool by Cambridge Semantics[3] allows to connect tabular data in Microsoft Excel spreadsheets to RDF ontologies in a triple store. Their focus lies more on user-friendly data acquisition and less on exploratory analyses of numeric datasets. *Dexter*[4] is a spreadsheet tool based on Datalog and JavaScript for the integration of small tabular datasets on the Web.

4.1.3 Relational Database Management Systems

Transaction processing queries and analytical queries substantially differ and put complementary requirements on backends [BPZ11, SBC+07]. Originally developed for transaction processing tasks, traditional RDBMS are no perfect fit to analytical queries [SBC+07]. Also, few domain experts such as from the natural sciences, are sufficiently familiar with RDBMS and SQL to quickly insert and ad-hoc query data from systems such as *Microsoft SQL Server* and *Oracle MySQL*; domain experts prefer more abstract representations [BJK+12].

4.1.4 Data Mining Tools

Data Mining and Knowledge discovery tools can be used for finding interesting patterns in multidimensional datasets. Example systems include *RapidMiner*, *Weka*, and *R*; they focus on allowing users to run machine learning algorithms such as for classification, recommendation and clustering over multidimensional datasets. Those tools typically assume datasets to be locally available, to be pre-processed, and to fit in memory. Therefore, such tools have difficulties if datasets are available from different data sources, if datasets are heterogeneous, and if the number and size of datasets are considerably large.

Our contributions in the integration and analysis of datasets from the Web can be applied for the data selection, pre-processing, and transformation steps in data mining processes. Thus, data mining tools could be used on top of our approach.

[2] See Economist 2013: "This spreadsheet is different", http://www.economist.com/news/finance-and-economics/21586569-error-apology-and-revision-spreadsheet-different, last accessed 2014-10-24.
[3] http://www.cambridgesemantics.com/products/anzo-express, last accessed 2015-01-31.
[4] http://dexter.stanford.edu/main/, last accessed 2015-01-31.

4.2 Semantic Integration and Analysis Approaches

Semantic approaches are well-suited to automatically integrate and make sense of data published on the Web. They often use Semantic Web concepts or technologies. With respect to exploration and analysis of many large numeric datasets from the Web, they lack efficiency.

4.2.1 Web Integration Systems

Web integration systems aim at providing uniform access to data sources published on the Web. The most common format to increase interoperability between systems and to explicitly share datasets is XML. For instance, the *Google Dataset Explorer* allows to describe datasets using the XML-based Google *Dataset Publishing Language* (DSPL) and provides interactive and intuitive visualisations such as line or bar charts. Such systems may be limited in integrating and using datasets that are published according to domain- or application-dependent specifications such as DSPL for the Google Public Data Explorer, SDMX for European statistics, and XBRL for financial reports. Some XML integration systems use mechanisms to access data over the Web that are not based on light-weight, basic Web standards such as HTTP URI and HTTP GET. Similarly, XML-based solutions do not necessarily use a formal domain model and typically are less efficient than relational or in-memory approaches [PBAP08].

Other systems do not need to rely on data made available in a machine-processable format such as XML. Examples include *Google Base*, *Needlebase*, and *Google Squared* that automatically retrieve (e.g., scrape) semi-structured data from the Web, e.g., from HTML tables and the Deep Web behind HTML forms. Systems such as *PAYGO* [MJC+07] and *RUBIX* [ALS+12] run elaborate algorithms based on machine learning, natural language processing, pattern matching, and probabilistic methods to identify, extract and match datasets; they do not focus on providing interactive exploration of large numeric datasets.

4.2.2 Semantic Web Analysis Systems

Some integration systems such as the *Semantic Web Search Engine* (SWSE) [HHU+11], intend to integrate and make explorable all structured data published on the Semantic Web. They implement algorithms such as entity consolidation [HHU+11] to automatically pre-process and integrate data on Web-scale.

For instance, *Humboldt* [KD08] combines browsing, faceted-search, and query-building capabilities for more powerful Linked Data exploration. The interfaces used by semantic systems such as follow-your-nose browsers, faceted-search interfaces, and query builders [Har10, HHU+11, KD08] over RDF data are designed more for analysing metadata and less for aggregations and complex measures over large amounts of numerical data in an exploratory fashion of overview first and details on demand [DR11].

The *SPARQL Package for R* by Willem Robert van Hage and Tomi Kauppinen[5] is an example tool to analyse numeric datasets from Linked Data with the statistics software R and is suitable for users familiar with R, SPARQL, and RDF.

A common tool for pre-processing Linked Data before an analysis is the RDF extension to OpenRefine, *RDF Refine*[6].

A more user-friendly approach is pursued by the *RapidMiner Linked Open Data Extension*[7] by Heiko Paulheim et al. that provides scripts for accessing of Linked Data from within the Data Mining Tool RapidMiner. Results from such scripts can be pipelined to Machine Learning algorithms in a visual interface.

Explain-a-LOD [Pau12] explains statistics based on correlations with information found in Linked Data. Such work show that Linked Data is a useful additional data source for knowledge discovery tasks. So far, it is unclear how such approaches can scale with additional datasets from Statistical Linked Data; for instance, Explain-a-LOD only uses DBpedia as a data source.

4.2.3 Semantic Data Warehouses

Since ontologies and Semantic Web technologies improve interoperability between information systems, they promise Situational Business Intelligence [ADE+13, LHM09]: allowing users to ask ad-hoc analytical queries to new data sources.

Besides difficulties in reconciling and integrating data sources, Pardillo et al. [PM11] discuss further shortcomings of current data warehouse design approaches which ontologies can contribute to overcome: incompleteness in multidimensional models; unclear characteristics of measures; missing semantic-aware summarisability checks; semantically-traceable models; reasoning on OLAP queries; asserting suitable visualizations; and security constraint validation [PM11].

Semantic Data Warehouses use ontologies, e.g., are based on Semantic Web technologies to formally describe the domain of data to be analysed [NN10, NN09, NB12, NBP+09].

[5] http://linkedscience.org/tools/sparql-package-for-r/, last accessed on 2014-10-26.
[6] http://refine.deri.ie/, last accessed 2014-11-23.
[7] http://dws.informatik.uni-mannheim.de/en/research/rapidminer-lod-extension/, accessed 2014-10-26.

DrillBeyond [ETBL12] is a system that allows users to query a local database automatically enhanced with external data. If a query is posed to the system, parts of the query that the local database cannot fulfil are tried to be taken from Open Data. The system is not designed for efficiency: OLAP-style analyses are not possible and much time is taken for pre-fetching and matching of datasets from Open Data.

Semantic Cockpit [NSL11] uses multidimensional modelling enriched with the explicit representation of and reasoning with external knowledge, such as domain knowledge represented in ontologies, semantics of derived measures and scores, and previous insights represented as so-called judgement rules. It remains unclear how such ontological semantics and rules are to be evaluated efficiently.

The *SODA system* [BJK+12] allows key-word search over data warehouses. Synonyms of multidimensional elements are loaded from DBpedia.

The *Information Workbench* [HSS11] uses Semantic Web technologies and provides typical Business Intelligence functionalities such as ETL of arbitrary data sources and flexible reports. Data is converted to RDF and stored in a centralised SPARQL engine. Follow-your-nose browsing, query builders and faceted search are possible. Since conceptually the data is not modelled as an MDM, OLAP interfaces and typical Data Warehouse optimisations are not directly applicable.

Ontology repositories such as *BioPortal*[8] and data catalogues such as *data.gov.uk* based on the *CKAN*[9] software store large amounts of data. Such work focus on search and reuse of ontologies and datasets instead of integration and analysis thereof.

Triple stores such as *Open Virtuoso* and *Sesame* allow to store RDF data, but neither are designed for OLAP analyses nor – similar to RDBMS – analytical query processing over large amounts of numeric data.

Wolfram Alpha[10] is a web-based front-end to a well-maintained knowledge base. Much manual effort is presumably put in formally representing and integrating common knowledge using the mathematical software Mathematica. Keyword search over the knowledge base and automatic execution of mathematical formulas allow user-friendly access to well-structured information from several integrated data sources. There is little known about the effort needed to integrate additional data sources and in optimising query processing over the knowledge base.

Current Semantic Data Warehouses show the potential of combining integration approaches based on Semantic Web technologies and exploration and analysis approaches from the area of Business Intelligence [ADE+13, LHM09]. More work has to be done to evaluate both flexibility and efficiency of such systems. Two examples of current

[8] http://bioportal.bioontology.org/, last accessed 2014-11-23.
[9] http://ckan.org/, last accessed 2014-11-23.
[10] http://www.wolframalpha.com/, last accessed 2014-11-23.

EU research projects are *CUBIST*[11] and *OpenCube*[12] that at the time of writing of this thesis have not demonstrated own solutions to the problem.

Our contributions in the integration and analysis of datasets from the Web uses Semantic Web standards as a basis, as well as established methods such as entity consolidation.

4.3 High-Performance Integration and Analysis Approaches

High-performance approaches are well-suited to allow efficient analytical queries over structured and semi-structured data sources. They make use of recent developments in software and hardware to efficiently execute low-level data processing operations. With respect to the pre-processing and integration of numeric datasets from the Web, they do not focus on flexibility.

4.3.1 ROLAP and MOLAP Systems

Since transaction processing and analytical queries put complementary requirements to databases, OLAP systems optimise analytical query processing over data extracted, transformed and loaded from heterogeneous data sources into a Data Warehouse.

For instance, systems such as *SAP NetWeaver Business Intelligence* and *Mondrian OLAP Server* rely on RDBMS and star schemas (ROLAP) or multidimensional arrays (MOLAP) and specific optimisations such as materialisation to efficiently execute analytical queries.

An example is *Tableau*, a widely-used OLAP and Data Analysis platform based on the Polaris system [STH02]. Polaris uses an algebra for translating user interactions to queries. The algebra uses “fields” as operands and “concatenate”, “cross”, “nest” as operations on axes and layers of pivot tables or other visualisations. Query processing conceptually is done by several SQL queries for selecting, partitioning and transforming the records to be visualised. In the implementation, an OLAP server and the MDX query language are used. Polaris also allows for roll-up and drill-down along dimension hierarchies. Polaris is more concerned with providing an intuitive and useful graphical user interface than on data integration and query optimisation.

[11] http://www.cubist-project.eu/, last accessed on 2014-10-31.
[12] http://opencube-project.eu/, last accessed on 2014-10-31.

Analytical (OLAP) queries can also be optimised with in-memory [VTBL13], column-oriented [AM08], and multi-core parallel [HS13] storage and processing techniques. *SAP HANA* [BPZ11] and *MonetDB*[13] are example systems.

When using OLAP systems for integrating heterogeneous datasets, most effort is put into design and maintenance of ETL pipelines, e.g., using *SAP ETL* software, *Pentaho Data Integration*[14], and *Kapow Software* [DCSW09]. If data at the sources change, the ETL process as well as the database schema often have to be modified. With tabular schemas that explicitly need to be maintained separately from the data, traditional OLAP systems also are less flexible in storing arbitrary provenance information [FKaGO+12].

4.3.2 NoSQL Systems

As an alternative to traditional relational databases, *NoSQL* (not only SQL) data management systems focus on scalability. NoSQL systems include document databases such as *CouchDB*, key-value stores such as *Cassandra*, and query engines such as *Hive* [CHH+13] based on the programming paradigm *MapReduce* for efficient parallel data processing.

Being able to spread processing tasks and data over a cluster of commodity machines such systems scale well with simple queries and loading of data. NoSQL systems are less suited for analytical queries that involve several joins, touch a lot of data (as in aggregation operations) or contain complex filters [CHH+13].

Abelló et al. [AFR11] have shown that the three most common low-level operations to building Data Cubes – full database scan, index access, and range index scan – can be implemented with MapReduce over a distributed database (Apache HBase). Here, MapReduce is used as an scalable ETL tool to generate data cubes in OLAP engines.

Another group of database systems not focusing on relational data and SQL queries are Graph databases such as *Neo4j*[15]. They allow to store schema-flexible graph data and efficient execution of graph analysis queries such as path traversals. Integration, filtering, and aggregation of numeric data is not the focus of these systems.

NoSQL systems work on semi-structured and schema-flexible data, but do not make use of more formal descriptions of data sources, e.g., based on Semantic Web technologies. Although NoSQL systems also can provide access to unstructured data and languages such as *Pig Latin* allow to write MapReduce programs in a high-level language with User Defined Functions (UDF) in common programming languages, the generation and

[13] https://www.monetdb.org/Home, last accessed 2015-01-23.
[14] http://community.pentaho.com/projects/data-integration/, last accessed 2015-01-23.
[15] http://neo4j.com/, last accessed 2014-11-23.

maintenance of ETL processes remain a costly, mostly manual, and error-prone process, similar as for OLAP systems.

4.3.3 Real-Time Data Warehouses

There are several research efforts in making data warehouses more automated, data-driven, and faster with respect to new or changing data sources. The goals are to link the business processes and objectives with the ETL design process and to consume data from a wider variety of data sources in near real-time [DCSW09].

Active data warehouses pro-actively react to changes in the data by taking appropriate actions such as triggering email alerts or creating reports. The respective logic for example is described using Event-Condition-Action rules [VS09].

Near real-time data warehouses allow fast loading over continuously changing data sources.

RiTE [TPL08] is a main-memory-based middleware system that allows data producers and consumers to coordinate. Based on an intermediary, in-memory store, data producer can make data available immediately via INSERT-like statements and – if user requirements such as for freshness allows – with bulk-load speed.

In-memory databases such as *SAP HANA* [BPZ11] and the Hybrid OLTP&OLAP High Performance DBMS *HyPer*[16] allow high speed of both OLAP and OLTP queries over up-to-date transaction data.

The management of ETL processes can be improved with data transformation models [FKaGO+12]. Such models ground ETL processes with prospective and retrospective provenance descriptions to their domain. For instance, business processes and objectives are uniquely specified to allow for partly automating design, development and maintenance of ETL processes [DCSW09].

Reducing the time to create, execute, and update ETL processes in case of changes to the data sources is difficult due to the wide range of possible data sources and transformations. Only semi-automatic approaches combined with more or less formal descriptions such as using the Business Process Model and Notation (BPMN) and Semantic Web technologies seem applicable [FKaGO+12].

A variety of systems can be used to provide access to continuously changing data sources (without necessarily permanent storage in a data warehouse): *Linked Data-Fu* [SSHS13] presents a rule-based language and engine for reading and writing of data on the Web, *SPADE* [GAW+08] provides an abstract language to execute processing operations over streams of data, and *ETALIS* [AF11] presents a language and engine for processing of complex events. It is unclear how well such solutions allow for the

[16] http://hyper-db.de/, last accessed 2014-12-11.

continuous development of data warehouses, for instance, if dimensions, members, and measures evolve ("slowly changing dimensions") or new facts are added to datasets.

In this thesis, our scenarios do not require the integration of quickly (e.g., several times a day) changing data sources. Instead, we are more concerned with integrating statistical datasets and background information. Our contributions in the integration and analysis of datasets from the Web make use of high-performance approaches such as optimised ROLAP systems.

In summary, traditional, semantic, and high-performance approaches are flexible or efficient, but seldomly show both characteristics. This work intends to fulfil this "gap of combined approaches" illustrated in Figure 4.1. For that, semantic data access and processing methods are combined with optimisation methods such as materialisation from ROLAP and MOLAP systems.

5 Mapping Data Cubes and Statistical Linked Data

In this chapter, we investigate the following research question:
Research Question 1. *How can we use existing OLAP engines for efficient query processing over heterogeneous multidimensional datasets from the Web?*

Figure 5.1 illustrates the contribution given in this chapter. We map between the Multidimensional Data Model (MDM) and the RDF Data Cube Vocabulary for efficient analytical query processing using existing OLAP engines.

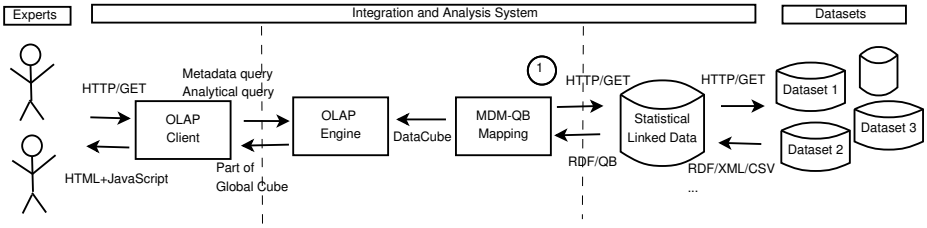


Figure 5.1: Illustration of contribution of the MDM-QB Mapping.

5.1 Introduction

Assuming domain experts familiar with OLAP interfaces and pivot tables, and datasets distributed over servers and heterogeneously published as Statistical Linked Data, flexible integration and efficient query processing approaches are necessary.

For instance, the Open Government Data (OGD) scenario motivates three example queries that citizens may want to pose over governmental statistics. In the following, we describe those three queries that require efficient data integration and query processing over datasets published as Statistical Linked Data:

Unemployment Fear and GDP Growth (UNEMPLOY). The GESIS LOD Pilot Application[1] publishes ALLBUS as Statistical Linked Data. This Linked Data source

[1] http://multiweb.gesis.org/gesis-lod-pilot/, last accessed 2014-11-06.

under the namespaces `allbus`[2] and `gesis`[3] includes the Unemployment Fear Survey dataset (`allbus:ZA4570v590.rdf#ds`).

Eurostat datasets are available as Linked Data from the *Eurostat Linked Data Wrapper* (Estatwrap)[4]. For instance, Estatwrap offers under the `eurostat` namespace[5] the Eurostat GDP Growth Dataset (`eurostat:id/tsieb020#ds`).

Citizens may want to integrate and make comparable the "Unemployment Fear" and the "GDP Growth Rate" over time for Germany to get insights about the correlation between GDP and employees perceived situation.

Number of Death by Illness and of Hospitals (HEALTH). The World Health Organisation (WHO) publishes in its Global Health Observatory Data Repository various datasets on important health topics. Among others there is a dataset reporting about mortality and burden of disease for different countries, i.e., describing the number of people dying from a certain illness in specific countries[6]. On the website *gho.aksw.org*, the *WHO Mortality Dataset* is available as Linked Data reusing QB[7]. Consider WHO Mortality Dataset in conjunction with the number of hospitals as provided by Eurostat in a *Number of Hospitals Dataset*. For all European countries citizens may want to compare the difference between people dying from a cause treated at hospitals and the number of hospitals to identify a possible correlation.

Comparing EU 2020 Indicators (EU2020). Estatwrap also publishes datasets with *EU 2020 indicators*, i.e., datasets containing several Eurostat metrics, such as the employment rate, the gross domestic expenditure on R&D, the energy intensity of the economy, and greenhouse gas emissions. Citizens may want to aggregate such indicators by average for all countries and to show the aggregated numbers per year, so that they can spot trends.

Existing OLAP engines need data modelled as data cubes. Therefore, a mapping between our Multidimensional Data Model (MDM) and Statistical Linked Data is necessary. Then, we can expect the following advantages:

- We can reuse existing OLAP engines such as SAP NetWeaver Business Intelligence[8].
- We can easily switch OLAP engines, as long as engines use a well-defined logical representation for data cubes such as the star schema or multidimensional arrays.

[2] http://lod.gesis.org/lodpilot/ALLBUS/, last accessed 2014-11-06.
[3] http://lod.gesis.org/lodpilot/ALLBUS/vocab.rdf#, last accessed 2014-12-17.
[4] http://estatwrap.ontologycentral.com/table_of_contents.html; last visited on 2014-04-03.
[5] http://estatwrap.ontologycentral.com/, last accessed 2014-11-06.
[6] http://apps.who.int/ghodata/; last visited on 2014-04-03.
[7] http://gho.aksw.org/; last visited on 2014-04-03.
[8] Allows basic exchange of metadata using XML, http://help.sap.com/saphelp_nw70ehp2/helpdata/de/60/2edd3b8f1b127de10000000a114084/content.htm, last accessed 2014-11-05.

- We can make use of built-in performance optimisations such as caching.
- We can make use of data integration capabilities of OLAP engines such as "virtual data cubes" from several single cubes.

There are three issues:

Non-standardised formats for sharing of data cubes: According to Rizzi et al. [RALT06], interoperability between data warehouses is still an open problem. The industry standards such as CWM are not expressive enough to capture the complex semantics such as summarisability represented by conceptual models.

Existing OLAP engines do not have standardised interfaces to load schema and data of cubes and the few formats for serialisation and sharing of data cubes are not widely adopted [HBH03]. Since common OLAP engines require their data in different formats to execute OLAP queries, it is unclear which representation is useful to load multidimensional datasets.

Semantic Gap between the conceptual and logical level of multidimensional data models: The semantic gap refers to the difference between the conceptual level of multidimensional datasets (data cubes and common elements such as dimensions) as well as queries to such datasets and the logical level of representing, storing and accessing the datasets [PMT08]. There are different ways to represent, store and access multidimensional datasets and automatically building and evolving data warehouses has long been a topic of research [RALT06]. It is unclear how to overcome the semantic gap if multidimensional datasets are published as Linked Data.

Linked Data specific problems for preprocessing Statistical Linked Data: OLAP engines are common; however, one does not know how to apply them for flexible integration and querying of Statistical Linked Data.

Also, it is unknown how to ensure – given an analytical query – that all necessary data have been loaded from Linked Data and are properly modelled. Also, data quality may be varying. For example, Eurostat would only give predictions for certain values and large datasets may be truncated[9].

Other issues are how to deal with missing information such as aggregation functions, how to ensure meaningful aggregations, and how to represent commonalities between several datasets to compare values. Similarly, permanent availability is not guaranteed.

Several heterogeneous vocabularies are used in Linked Data. Still, there is no common agreement on how to make important aspects of statistical data self-descriptive, e.g., mathematical aggregation functions [VLH+10].

[9] As is the case for the Eurostat Linked Data Wrapper due to timeouts of the Google App Engine.

As an overall contribution, in this chapter, we show how to transform Statistical Linked Data for use in existing OLAP engines. More specifically, our contribution are as follows:

- Based on the arguments in the basics chapter (Section 3.3.2.1), we select the widely-used RDF Data Cube Vocabulary as a format for sharing of data cubes. To reduce the semantic gap, we present a mapping between a common multidimensional data model of cubes and Linked Data sources using the RDF Data Cube Vocabulary, including a definition of relevant, correctly modelled, and meaningfully aggregated data for a multidimensional dataset (Section 5.2.2).
- We show how to deal with Linked Data specific problems and present an automatic deployment of the common multidimensional data model from Statistical Linked Data to data warehouses, considering implicit *overlaps* between multidimensional datasets for integration (Section 5.2.3).
- We evaluate the mapping with example queries motivated by our Open Government Data scenario; we use the star schema for ROLAP as a typical logical representation and describe a prototypical system reusing a common ROLAP engine for query processing and data integration (Section 5.3).

In Section 5.4 we discuss our results and present lessons learned. In Section 5.5, we describe related work, after which, in Section 5.6, we conclude.

5.2 Approach: MDM-QB Mapping

Our approach consists of two parts: an offline extract-transform-load (ETL) part that creates a data warehouse for an existing OLAP engine based on datasets URIs; and a runtime part where – after population of the data warehouse with relevant data – ad-hoc OLAP queries can be issued to the OLAP engine. We now give an overview of the approach:

1. **Dataset Selection:** The domain expert defines the datasets to be integrated. For that, URIs of datasets refer to specific datasets in Linked Data sources.
2. **Dataset Extraction:** For information found on the Web via the dataset URIs, the multidimensional data model is automatically created.
3. **Data Warehouse Creation:** From the conceptual data model, a logical representation for use in an existing OLAP engine is created.
4. **Query Processing:** An OLAP client issues analytical queries to the OLAP engine which executes such queries over the logical representation.

For that, we define relevant data in Section 5.2.1 and a mapping between the common MDM and Statistical Linked Data in Section 5.2.2. We use an object-oriented representation as well as implicit overlaps between datasets in Linked Data that we present in Section 5.2.3.

5.2.1 Relevant RDF Data Describing Multidimensional Datasets

In this section, we define relevant data as a specific RDF graph to be queried with a SPARQL engine.

Based on the notion that in Linked Data, RDF may be stored in a distributed manner, Definition 7 defines *relevant data for multidimensional datasets*.
Definition 7 (Relevant Data for Multidimensional Datasets)**.** *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 dataset URI, 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.

Then, all relevant data can be found using Linked Data crawlers that – starting from a seed list of URIs – use a depth-first or breadth-first crawling strategy for RDF data. One crawler that we use in this work is *LDSpider*[10] [IUBH10]. 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.

Crawling may include further information, e.g., `rdfs:seeAlso` links from relevant entities and information encoded in the Vocabulary of Interlinked Datasets (VoiD)[11]. 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 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.

[10] http://code.google.com/p/ldspider/, last accessed 2014-11-04.
[11] http://rdfs.org/ns/void#, last accessed on 2014-06-21.

5.2.2 Mapping the Multidimensional Data Model and Statistical Linked Data (MDM-QB Mapping)

In this section, we describe a mapping between Statistical Linked Data and our common multidimensional data model (MDM).

This *MDM-QB Mapping* [KH11, KOH12] will serve as a basis for transforming RDF terms reusing QB to multidimensional elements in an MDM.

We use classes and properties defined by the RDF Data Cube Vocabulary (QB) and other standard vocabularies for publishing statistical Linked Data, e.g., SKOS[12] and XKOS[13]. XKOS is an extension to SKOS allowing for the representation of classification hierarchies; an earlier version was named SKOSCLASS[14].

We describe the mapping using SPARQL graph patterns. Every match over an RDF graph (Definition 6) describes a multidimensional element of the MDM. The bindings (terms and triples) of graph patterns describe the properties of the multidimensional element.

Given an RDF graph from Statistical Linked Data, we define the set of all instances of multidimensional elements of the multidimensional data model described in the graph. For that, we use simple set notation and basic SPARQL graph patterns as syntax and set theory as semantics [HKR09, p. 363ff]. Several graph patterns are separated by "`.`". Given a multidimensional element x, $id(x) \in (\mathcal{I} \cup \mathcal{B} \cup \mathcal{L})$ returns its RDF identifier. For conjunctions we use "`,`" and for disjunctions "`;`".

In the following list, for each multidimensional element, the set of all instances is defined according to the RDF graph. For instance, the first definition states that the set of all data cubes is defined as a set of tuples each with three elements: the name of the cube as a String ($?nameDC$), the data cube schema (cs), and the set of all facts in the cube (F). The name of the cube is given by the URI of the respective instance of `qb:DataSet`. The data cube schema is defined by the URI linked from the dataset URI via `qb:structure`. And the facts are defined by instances linked to the dataset URI via `qb:dataSet`.

DataCube is defined by $DataCube = \{(?nameDC, cs, F) \in String \times DataCubeSchema \times \mathcal{F} | cs = (\text{"}cs\text{"}, D, M)$, `?nameDC a qb:DataSet . ?nameDC qb:structure id(cs)`, $\forall (?obs \in F)$ `?obs qb:dataSet ?nameDC`$\} \cup MultiCube$.

The set of data cubes also includes all pre-defined multi-cubes.

[12] http://www.w3.org/2004/02/skos/, last accessed 2014-11-20.
[13] http://purl.org/linked-data/xkos# and http://rdf-vocabulary.ddialliance.org/xkos, accessed 2014-06-21.
[14] http://www.w3.org/2011/gld/wiki/ISO_Extensions_to_SKOS, last accessed on 2014-06-21.

DataCubeSchema is defined by $\{(?nameCS, D, M) \in (\mathcal{I} \cup \mathcal{B}) \times \mathcal{D} \times \mathcal{M} |$ `?nameCS a qb:DataStructureDefinition`, $D = \{d |$ `?nameCS qb:component ?comp. ?comp qb:dimension id(d)` $\}$, $M = \{m |$ `?nameCS qb:component ?comp. ?comp qb:measure id(m)` $\}\}$.

Fact is defined by $\{(?nameF, C, E) \in (\mathcal{I} \cup \mathcal{B}) \times 2^{Dimension \times Member} \times 2^{Measure \times Literal}$ $|$ `?nameF a qb:Observation`, $C = \{(d,m) |$ `?nameF id(d) id(m)` $\}$, $E = \{(m,t) |$ `?nameF id(m) id(t)` $\}\}$.

Measure is defined by $\{(?nameMS, ?calc) \in (\mathcal{I} \cup \mathcal{B}) \times (\mathcal{I} \cup \mathcal{B} \cup \mathcal{L}) |$ `?comp qb:measure ?nameMS. ?comp qb4o:aggregateFunction ?calc;` $?calc \in \{SUM, AVG, COUNT\}\}$ with $calc : 2^{Fact} \rightarrow \mathcal{L}$.

Dimension is defined by $\{(?nameD, H) \in (\mathcal{I} \cup \mathcal{B}) \times \mathcal{H} |$ `?nameD a qb:DimensionProperty`, $H = \{h |$ `?nameD qb:codeList id(h)` $\} \cup \{h = ?nameD |$ `?nameD rdfs:range ?range. FILTER(?range != skos:Concept)` $\}\} \cup \{(\text{"}Measures\text{"}, Measures)\}$.

Hierarchy is defined by $\{(?nameH, L, rolluplevel(L), rollupmember(L))$ $\in (\mathcal{I} \cup \mathcal{B}) \times \mathcal{L} \times ROLLUPLEVEL \times ROLLUPMEMBER | Hierarchies\_Regular \cup Hierarchies\_Without\_Codelist \cup Hierarchies\_XKOS \cup Hierarchies\_Measure\_Dimension\}$ with

$Hierarchies\_Regular = \{(?nameH, L, rolluplevel(L), rollupmember(L)) |$ `?dim qb:codeList ?nameH`, $L = \{l = ?nameH | (\nexists l)$ `id(l) skos:inScheme ?nameH. id(l) a xkos:ClassificationLevel`$\}\}$, $rolluplevel(L) = \{\}$, $rollupmember(L) = \{\}\}$,

$Hierarchies\_Without\_Codelist = \{(?nameH, L, rolluplevel(L), rollupmember(L))$ $|(\nexists l)$ `?dsd qb:component ?comp. ?comp qb:dimension ?nameH. ?nameH qb:codeList ?codelist`, $L = \{?nameH\}$, $rolluplevel(L) = \{\}$, $rollupmember(L) = \{\}\}$,

$Hierarchies\_XKOS = \{(?nameH, L, rolluplevel(L), rollupmember(L)) |$ `?nameH a skos:ConceptScheme`$), L = \{l |$ `id(l) skos:inScheme ?nameH. id(l) a xkos:ClassificationLevel`$\}$, $rolluplevel(L) = \{(l_1, l_2) \in L \times L |$ `id(`l_1`) xkos:depth ?x. id(`l_2`) xkos:depth ?y. FILTER(?x = ?y+1)`$\}$, $rollupmember(L) = \{(v_1, v_2) \in V_1 \times V_2 | (l_1, V_1), (l_2, V_2) \in L,$ `id(`v_1`) skos:broader id(`v_2`); id(`v_2`) skos:narrower id(`v_1`)`$\}\}$,

$Hierarchies\_Measure\_Dimension = \{(\text{"}Measures\text{"}, measureL, rolluplevel(measureL), rollupmember(measureL)\}$.

Level is defined by $\{(?nameL, V, ?depth) \in (\mathcal{I} \cup \mathcal{B}) \times \mathcal{V} \times \mathcal{L} | Levels\_Degenerated \cup Levels\_topConcept \cup Levels\_XKOS \cup Levels\_Measure\_Dimension\}$ with

$Levels\_Degenerated = \{(?nameL, V, ?depth) | (\nexists l)$ `?dsd qb:component ?comp. ?comp qb:dimension ?nameL. ?nameL rdfs:range`

`?range. FILTER(?range != skos:Concept)`, $V = \{v|$ `?obs qb:dataSet qb:structure ?dsd. ?obs ?nameL id(v),?`$depth = 1\}$,

$Levels\_topConcept = \{(?nameL, V, ?depth)|$ `?dsd qb:component ?comp. ?comp qb:dimension ?dim. ?dim qb:codeList ?nameL.`, $V = \{v|$ `?nameL skos:hasTopConcept id(v),?`$depth = 1\}\}$,

$Levels\_XKOS = \{(?nameL, V, ?depth)|$ `?dsd qb:component ?comp. ?comp qb:dimension ?dim. ?dim qb:codeList ?nameH. ?nameL skos:inScheme ?nameH. ?nameL a xkos:ClassificationLevel,` $V = \{v|$ `id(v) skos:member ?nameL`$\}$, `?nameL xkos:depth ?depth` $\}$,

$Levels\_Measure\_Dimension = \{(?nameL, V, ?depth)|?nameL = Measures, V = Measure, ?depth = 1\}$.

Member is defined by $\{(?nameM) \in (\mathcal{I} \cup \mathcal{L})\}$.

In the following, we explain the mapping along an example.

5.2.2.1 Example Multidimensional Data Model

We use MDM-QB Mapping to represent the MDM as defined from the dataset URIs of the Real GDP Growth Rate from Estatwrap[15] and the Unemployment Fear from ALLBUS[16].

`DataCube = {eurostat:id/tec00115#ds, allbus:ZA4570v590#ds, eurostat:id/tec00115#ds;allbus:ZA4570v590#ds}`

Any cube in an MDM is uniquely identified by an instance of `qb:DataSet` related by the property `qb:structure` to an instance of `qb:DataStructureDefinition`. In our example, also one multi-cube, identified by a semi-colon-separated list of dataset URIs is contained and is later described.

We do not distinguish machine-readable and human-readable names in our mapping. The `rdfs:label` of the dataset can be used as a human-readable name of a data cube. The property `rdfs:comment` can be used for a description of a data cube. Both properties can generally be used to name and describe multidimensional elements.

`DataCubeSchema = {eurostat:dsd/tec00115#dsd,allbus:za4570dsd.rdf#dsd,eurostat:dsd/tec00115#dsd;allbus:za4570dsd.rdf#dsd}`

[15] http://estatwrap.ontologycentral.com/id/tec00115#ds, last accessed on 2014-06-18.
[16] http://lod.gesis.org/lodpilot/ALLBUS/ZA4570v590.rdf#ds, last accessed on 2014-06-18.

The data cube schema of a cube is given by the data structure definition of the QB dataset. Also the multi-cube gets defined a data cube schema.

A Cube contains Facts representing the actual statistical data. In QB, a Fact is an instance of `qb:Observation`. Such facts are connected to their respective cube via the property `qb:dataSet`.

`Fact = {_:tec00115_obs1, _:tec00115_obs2, ..., _:ZA4570v590-_obs1, _:ZA4570v590_obs2,...}`

Since RDF provides a flexible schema, additional information can be attached. In QB, such information can be taken from the resources representing the Member values of Dimensions. Also, a sub-property of `qb:ComponentProperty`, `qb:Attribute-Property`, can be used. All kinds of metadata can be added to interpret an observation, e.g., the unit of measurement, qualitative information such as whether the value has been estimated. Such metadata can both be simply represented as human-readable literal values of data-type string and as URIs reused from existing vocabularies and datasets.

The following shows an example fact for Austria, in 2006, with percentage change from previous period as unit, and different measure values about the GDP growth: an average and sum of 3.7, and a count of 1.

`_:tec00115_obs1 = ("_:tec00115_obs1", {(estatwrap:geo, euro-stat:dic/geo#AT), (dcterms:date, 2006), (estatwrap:unit, eurostat:dic/unit#PCH_PRE)}, {((sdmx-measure:obsValue, euro-stat:id/tec00115#ds, SUM), 3.7), ((sdmx-measure:obsValue, eurostat:id/tec00115#ds, AVG), 3.7),((sdmx-measure:obsVal-ue, eurostat:id/tec00115#ds, COUNT), 1)})`

For each predefined Dimension a Fact has a Member. Members form the possible values of a Dimension. In QB, these Dimension Members can be given explicitly via `qb:codeList` by instances of `skos:ConceptScheme`, or implicitly by the Members used by actual Facts in the data. Additionally, the `rdfs:range` of a `qb:ComponentProperty` can state the type of the Members.

`Member = {eurostat:dic/geo#DE, ..., 2010, ..., PCH_PRE, ...,allbus:geo.rdf#00,..., allbus:variable.rdf#v590_1,...}`

`Measure = {(sdmx-measure:obsValue, eurostat:id/tec00115#ds, SUM),(sdmx-measure:obsValue, eurostat:id/tec00115#ds, AVG), (sdmx-measure:obsValue, eurostat:id/tec00115#ds, COUNT), (sdmx-measure:obsValue, allbus:ZA4570v590#ds, SUM), (sdmx--measure:obsValue, allbus:ZA4570v590#ds, AVG), (sdmx-meas-ure:obsValue, allbus:ZA4570v590#ds, COUNT)}`

In QB, Dimensions and Measures are predefined by the data structure definition of the dataset and represented as instances of `qb:ComponentProperty`.

In QB, for Measures, another subproperty of `qb:ComponentProperty`, `qb:MeasureProperty`, is available. QB does not describe how to model aggregation functions.

We consider the explicit representation of aggregation functions using the QB4OLAP ontology[17].

However, few publishers explicitly represent aggregation functions. If aggregation information is not given via `qb4o:aggregateFunction` – as is the case in our example datasets – we create one Dimension and a Measure for each aggregation function possibly correct, e.g. SUM, AVG, MIN, MAX, COUNT, and DISTINCT-COUNT for numerical measures (for dimensions with range `xsd:decimal`) as well as COUNT and DISTINCT-COUNT for nominal measures (e.g., `xsd:string` and `xsd:date`).

`Dimension = {estatwrap:geo, dcterms:date, estatwrap:unit, gesis:geo, gesis:variable}`

Every dimension has a hierarchy for every code list (possibly `skos:ConceptScheme`, `skos:Collection`, and `qb:HierarchicalCodeList`, each with `rdfs:range skos:Concept`) and for every range that is not `skos:Concept`.

In the latter case, we simply use the dimension name as hierarchy name. The dimension-hierarchy-level combination labelled "Measures" is added manually.

Members of a Dimension are grouped along one or more Hierarchies of one or more Levels of granularity. In QB, Hierarchies of Levels depend on the actual Members of the Dimension. For instance, if we have `xsd:date` as range, we can have the natural hierarchy of year, month, day.

Or, Members of type `dbpedia:Country` might be put into categories such as Federal Countries and Alpine Countries. Also, countries could be classified according to their official language.

`Hierarchy = {estatwrap:geo, dcterms:date, eurostat:dsd/tec00115#cl_unit,allbus:geo.rdf#list,allbus:variable.rdf#list}`

There are different ways to define the levels, rolluplevel and rollupmember for a hierarchy. In case the hierarchy is a `skos:ConceptScheme` levels may be attached as instances of `xkos:ClassificationLevel`. In all other cases, simply the hierarchy itself is the level. In case no code list is defined, the dimension name is used

[17] http://purl.org/qb4olap/cubes and `qb4o:aggregateFunction`, e.g., `qb4o:Sum`, last accessed on 2014-06-18.

as a hierarchy name. The vocabulary XKOS allows to explicitly represent hierarchy levels.

`Level = {estatwrap:geo, dcterms:date, eurostat:dsd/tec001-15#cl_unit, allbus:geo.rdf#list, allbus:variable.rdf#list}`

In case no levels are defined, the hierarchy has exactly one level with the same name as the hierarchy.

`Member = {eurostat:dic/geo#DE, ..., 2010, ..., eurostat:dic/unit#PCH_PRE, ..., allbus:geo.rdf#00, ..., allbus:variable.rdf#v590_1, ...}`

Members can be resources or literal values. For instance, the data structure definition of the GDP Growth Cube contains as `qb:DimensionProperty dcterms:date` with a literal of type `xsd:date` such as "2008". As another example, `estatwrap:geo` has a code list with resources as Members that represent countries.

In the following, we go into more details of three aspects of our mapping: 1) integrity constraints to check correct modelling of cubes, 2) the possible integration of datasets via shared dimensions and multi-cubes as well as 3) meaningful aggregations (summarisability).

5.2.2.2 Ensuring Correct Modelling of Data Cubes

The QB specification defines 21 *QB integrity constraints*[18] (IC-1 to IC-21) as SPARQL ASK queries. By evaluating the queries over RDF graphs with QB datasets, one can check whether datasets are correctly modelled. In the following, we describe the constraints put to the mapped MDM due to the integrity constraints:

- According to IC-1, every fact can only be contained in one data cube.
- According to IC-2, every data cube has exactly one data cube schema.
- According to IC-3, every data cube schema contains at least one measure.
- According to IC-11, every fact has a value for each dimension declared in its data cube schema.
- According to IC-12, no two different facts in a data cube may have the same value for all dimensions.
- According to IC-14, each fact must have a value for every declared measure.
- According to IC-4 and IC-5, every dimension has at least one hierarchy.

[18] http://www.w3.org/TR/vocab-data-cube/#wf-rules, last accessed on 2014-06-20.

- According to IC-19, IC-20, and IC-21, every value of a dimension d on every fact must be in $dom(d)$ which is defined as the union of all members in all hierarchies and levels of the dimension.

Some parts of QB we do not represent in our MDM:

In our MDM, we do not represent instances of `qb:AttributeProperty` (IC-6, IC-13). `qb:AttributeProperty` only allow to give additional information to instances of `qb:Observation`, `qb:DataSet` and `qb:Slice`, but not to single members as sometimes done with *dimension attributes* in multidimensional data models [PJD01]. In Chapter 6, we present the OLAP-to-SPARQL algorithm which allows analytical queries directly over the RDF and therefore allows to consider RDF terms such as the number of inhabitants of a city in queries.

In our MDM, we do not represent instances of `qb:Slice` (IC-7, IC-8, IC-9, IC-10, IC-18). In Chapter 7, we present RDF Aggregate Views, i.e., instances of `qb:Slice` that are used to store pre-aggregated facts of a data cube.

5.2.2.3 Integrating Datasets in Multi-Cubes

Whereas our MDM only considers single data cubes, our mapping considers multi-cubes for querying over several data cubes. For instance, in our scenario we intend to integrate and make comparable metrics about Unemployment Fear and GDP Growth from the respective cubes.

For queries over two data cubes simultaneously, it is possible to define a unified schema, a *multi-cube*. Given a specific set of data cubes, we define a multi-cube as per Definition 8.

Definition 8 (Multi-Cube)**.** *A multi-cube is a* virtual *data cube [SDN00] formed from a specific set of data cubes by taking the union of dimensions and measures of the single data cubes. If a multi-cube is queried, an OLAP engine will automatically compute a join of facts on selected dimensions.*

In our mapping, we define one multi-cube consisting of all cubes in the MDM that share dimensions and members.

For example, the Unemployment Fear Cube and the GDP Growth Cube together make a multi-cube with the URI `eurostat:id/tec00115#ds;allbus:ZA4570v590-#ds` (we denote multi-cubes with a semi-colon-separated list of single dataset URIs) and the following dimensions:

```
Dimension = {estatwrap:geo, dcterms:date, estatwrap:unit,
gesis:geo, gesis:variable}
```

For a query over a multi-cube to return non-empty results, the data cubes defining the multi-cube need to *overlap*, i.e., need to have *shared dimensions*. Dimensions are

shared in multidimensional datasets published reusing QB if they are used by several data cubes.

Shared members are members used by several cubes in shared dimensions. For instance, both cubes use the same time dimension `dcterms:date` with literal values denoting the same time points, e.g., 2004 and 2005.

A more formal definition of shared dimensions and members can be given as follows: A dimension `:dimension` is shared between the data cube `:ds1` and `:ds2` if the following graph patterns hold:

```
:ds1 qb:structure/qb:component/qb:dimension :dimension.
:ds2 qb:structure/qb:component/qb:dimension :dimension.
```

Similarly, hierarchies and levels can be shared using QB.

A member `:mem` is shared if in addition to those graph patterns there are observations in `:ds1` and `:ds2` that exhibit this member `:mem` as values for the sharing dimension.

The equivalence of literal values is identified by string matching and could be extended with canonical representations of literal values, e.g., in case of decimal "1.0" versus integer "1".

However, dimensions and members also may be shared implicitly, i.e., *equivalence statements* such as by `owl:sameAs` and `owl:equivalentProperty` can be given in RDF.

We note that not only equivalence statements such as `owl:equivalentProperty` and `owl:sameAs`, but also other statements indicating shared identifiers, e.g., `rdfs:subPropertyOf`, could be used.

For instance, the Unemployment Fear survey dataset and GDP Growth dataset both have geo dimensions and members denoting Germany that can be linked. If the Unemployment Fear survey dataset and GDP Growth dataset share the dimension for geo and the member representing Germany as well as the time dimension and members, the multi-cube defined by the two datasets exhibits more overlap. See Figure 5.2 for an illustration. Only after considering the equivalence of "geo" and "location" as well as "Germany" and "DE", the number of answers given (No.) can be compared with the GDP Growth Rate (GDP).

To compare metrics from separate cubes – different from Dimensions and Members – we assume that measures denote different metrics, even though they may be described by the same property or linked by `owl:sameAs`. Therefore, measures are not directly corresponding to `qb:MeasureProperty` but are uniquely identified by 1) the cube using the measure, 2) the aggregation function (differently said, the component initialising the measure, possibly defining an ordering and an aggregation function and 3) the measure property. When defining the Drill-Across operation in Chapter 6, we will cover the more general case where data cubes may use the same metric.

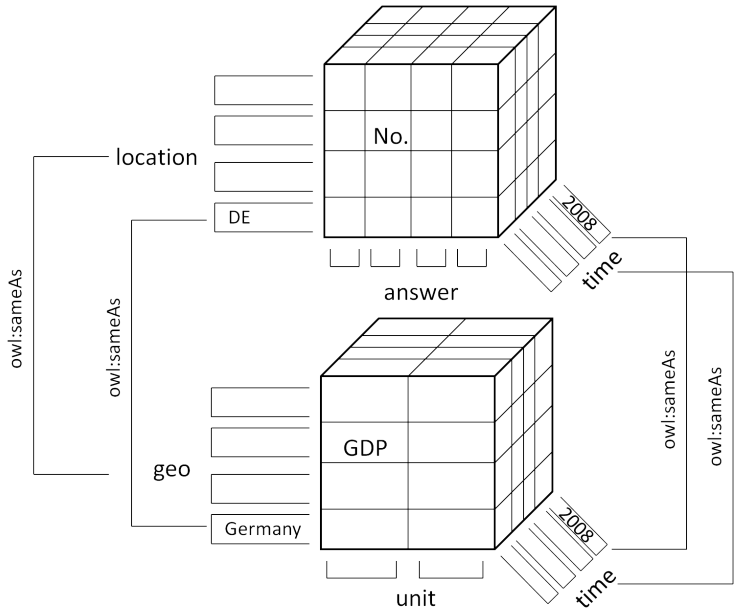


Figure 5.2: Example of a multi-cube consisting of two implicitly overlapping data cubes.

In the following, we explain how we evaluate implicitly shared dimensions and members. *OWL* semantics[19] require all statements involving identifiers of shared dimensions and members to be duplicated for all equivalent terms (we call it *equivalence duplication strategy*); however, according to Hogan et al. [HHU+11], this duplication possibly results in an addition of triples quadratic with respect to the size of groups with equivalent entities. Also, in entity-centric systems such as query engines the duplication would confuse users by presenting duplicate results.

Therefore, we apply *entity consolidation* [HHU+11], the use of canonical values for consolidated entities of shared dimensions and members.

For that, to integrate implicitly shared dimensions and members, we first query for all equivalence statements (e.g., `owl:sameAs`) in the relevant data and implement the reflexive, symmetric and transitive properties of instance equivalence [HHD04].

For that, we compute an equivalence list.

In their algorithm "equivs" Hogan et al. [HHD04] create a list of lists with equivalent URIs (equivalence class). We create an associative array from the URI to an integer

[19] See "Table 4. The Semantics of Equality" at http://www.w3.org/TR/owl2-profiles/, last accessed on 2014-06-29.

as unique identifier for every equivalence class. For every equivalence statement, we check whether the URIs are in different equivalence classes. If so, we add every URI of the second equivalence class to the first and merge the equivalence classes. The unique integer for each equivalence class we use as canonical identifier for entities.

Different from Hogan et al. [HHD04] we do entity consolidation not on the RDF but on the results of SPARQL queries over the RDF. This is equivalent since every single SPARQL result represents a multidimensional element. After replacement of canonical values and removal of duplicate results, for each distinct canonical value, a dimension or member is created.

For instance, from the set of dimensions of the multi-cube the duplicate geo dimension `gesis:geo` is removed:

`Dimension = {estatwrap:geo, dcterms:date, estatwrap:unit, gesis:variable}`

In metadata queries such as for a list of dimensions, canonical values need to be used to uniquely identify the elements. This is no restriction to our approach, since first queries will ask for all multidimensional elements, from which the user would pick. However, while evaluating those metadata queries over Linked Data using SPARQL (according to the MDM-QB Mapping), the canonical values are not used by all datasets and queries would return no results. Therefore, we modify the metadata SPARQL queries to consider all possible URIs in the equivalence class of a canonical value using a FILTER clause and OR conditions over variables for shared dimensions and members. In results from those metadata SPARQL queries, identifiers have to be replaced by canonical values and duplicates have to be removed.

5.2.2.4 Ensuring Summarisability

For summarisability we require from a QB dataset:

- Disjointness: Every member on a lower level has no more than one member on a higher level pointing to it with `skos:narrower`; this means, as defined for the MDM, we assume the *rolluplevel* relation of a hierarchy to form a directed acyclic graph, as well as strictness of the *rollupmember* relations [RTZ11].
- Completeness: Every member on a higher level points to at least one member on a lower level with `skos:narrower`. Therefore, we assume *symmetric* hierarchies [MZ06].
- Meaningful Aggregation Function: A meaningful aggregation function is selected by the analyst.

We define a *lean dataset* in Definition 9 as to correctly model a QB dataset and to fulfil our assumptions for summarisability.

Definition 9 (Lean Dataset)**.** *A lean dataset is modelled correctly according to the Integrity Constraints of the RDF Data Cube Vocabulary. The dimension hierarchies of a lean dataset fulfil disjointness and completeness. Also, a meaningful aggregation function is selected by the analyst or automatically selected. Finally, a lean dataset does not exhibit redundant observations, i.e., either observations are stored only on a higher level such as per years or only on the lowest level of each dimension. Higher-level aggregate facts can then be computed from aggregating the given facts [CD97].*

An aggregation of facts from different granularities would result in incorrect numbers, e.g., a *SUM* over gender *male*, *female*, and *total*.

In summary, the mapping of datasets using QB to an MDM is not as straightforward as the name suggests. Aggregation functions and multi-level hierarchies are common in MDMs, but QB does not have a direct pendant and alternative vocabularies such as `qb4o` and `xkos` need to be used. In particular, the integration of datasets in multi-cubes, and summarisability require specific handling.

5.2.3 A Logical Representation of the Multidimensional Data Model to Populate OLAP Systems

In this section, we describe how our MDM can be deployed on existing OLAP engines for query processing and integration.

For that, we show how to represent the MDM using objects of classes defined in a UML class diagram in Figure 5.3. The class diagram resembles an object-oriented representation of our MDM (Section 3.1). Every multidimensional element is considered an object in the UML class diagram. Other authors also use UML for modelling multidimensional datasets [PMT08].

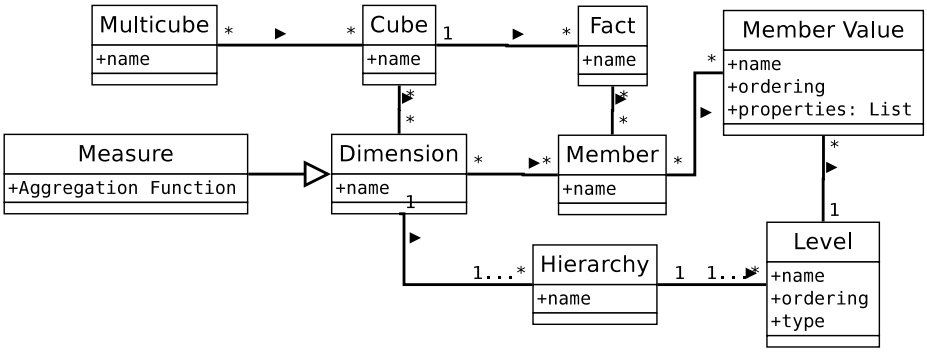


Figure 5.3: UML class diagram of common multidimensional data model.

5.2.3.1 Representation of an Example Multidimensional Data Model

Along an example, we describe how the class diagram resembles our MDM:

The central class is the cube (`DataCube` in our MDM). The multidimensional data model, represented as a class diagram, describes one or more cubes which are thematically related. In the class diagram, `DataCubeSchema` and `DataCube` from our MDM are not distinguished since every cube has an individual, implicitly known schema. As in our MDM, a cube contains facts representing the actual statistical data. For each predefined dimension, a fact will have a member.

As an example fact, 1,126 participants have answered "No Unemployment Fear" according to the Unemployment Fear survey in 1980 in Germany. As another example, in Germany 2004 there was a 1.2% change of the Real GDP Growth on previous years according to Eurostat statistics.

As also possible in our MDM, dimensions can be shared by several cubes in multi-cubes (Multicube). For example, the Unemployment Fear Cube and the GDP Growth Cube share the geo and time Dimensions. Whereas both cubes share the location and time dimension, only the Unemployment Fear Cube has a "variable" Dimension denoting the kind of answer.

A cube defines one or more `Measures`. Every Measure is regarded as a separate Dimension. For instance, the Unemployment Fear Cube has as Measure the number of participants for which a mean (AVG) results in a meaningful aggregation.

Every Dimension has one or more `Hierarchies` which in turn have one or more `Levels` with an `Ordering` and a `Type`, e.g., Boolean, Decimal, Integer, String, Date, Time, and Timestamp. For instance, year would be Integer and job role would be String. For instance, the time dimension may have a typical hierarchy of day, month and year levels; whereas day has type Date, month and year may be of type Integer, also.

The datasets in our scenario do not make hierarchies and levels explicit. Thus, every dimension has exactly one hierarchy and level with the same name as the dimension.

Slightly different from our MDM, members form the possible values of a dimension and *Member Value* represent the actual values of a member on a level of a hierarchy. For instance, the time dimension may have a member "2010-01-01". For any level of its dimension, a member has a *Member Value*. For instance, the member "2010-01-01" may have as day "1", as month "January" and as year "2010".

Similar to dimensions, members can be shared by several dimensions. For example, the Unemployment Fear Cube and the GDP Growth Cube share the geo dimension member for Germany.

5.2.3.2 Automatically Populating a Data Warehouse

In the following, we show how the logical representation (as objects and classes) can automatically be deployed on existing OLAP engines.

For instance, ROLAP engines use the star schema in an RDBMS to represent the MDM. There, we have a Fact table for facts that for each fact contains a row, for each measure and dimension contains a column, and for each dimension joins to a dimension table with the members and values. To transform our logical representation to a star schema the following steps are done:

1. For each Cube, create a Fact table with as many columns as Dimensions and Measures.
2. For each Dimension, create a Dimension table with one column for the primary key and as many additional columns as Levels in Hierarchies. Note, shared dimensions are represented by the same dimension table. The type information from the level is used for typing the column.
3. For each Member in a Dimension, add a new row to the Dimension table.
4. For each Fact, add a new row in the Fact table. For the Members, add a foreign key to join with the primary key of the Dimension table.

When preparing a lean dataset for ROLAP, summarisability is ensured and every single fact can be stored in the fact table.

If a multi-cube is queried, the OLAP engine evaluates drill-across as a join of all fact tables on the shared dimensions to create a new fact table reusing the existing dimension tables [SDN00].

Since conceptually, there is no difference in representing the data in an in-memory multidimensional array or in an RDBMS, we argue that common OLAP systems can be used by our approach.

5.3 Evaluation

We describe the *Cube-to-ROLAP Prototype*, an implementation of the MDM-QB Mapping in an extract-transform-load pipeline that automatically prepares Statistical Linked Data using QB. We evaluate the mapping by using the prototype for our example queries. In the following, we will describe our implementation and then the application to the scenarios.

5.3.1 Implementation

Figure 5.4 shows the architecture of our system consisting of two parts: an ETL pipeline that creates a data warehouse based on datasets given by their URIs; and an OLAP engine, to that any number of OLAP queries can be issued after the ETL process has been finished. Running an experiment with the system includes the following steps, also indicated with numbers in Figure 5.4.

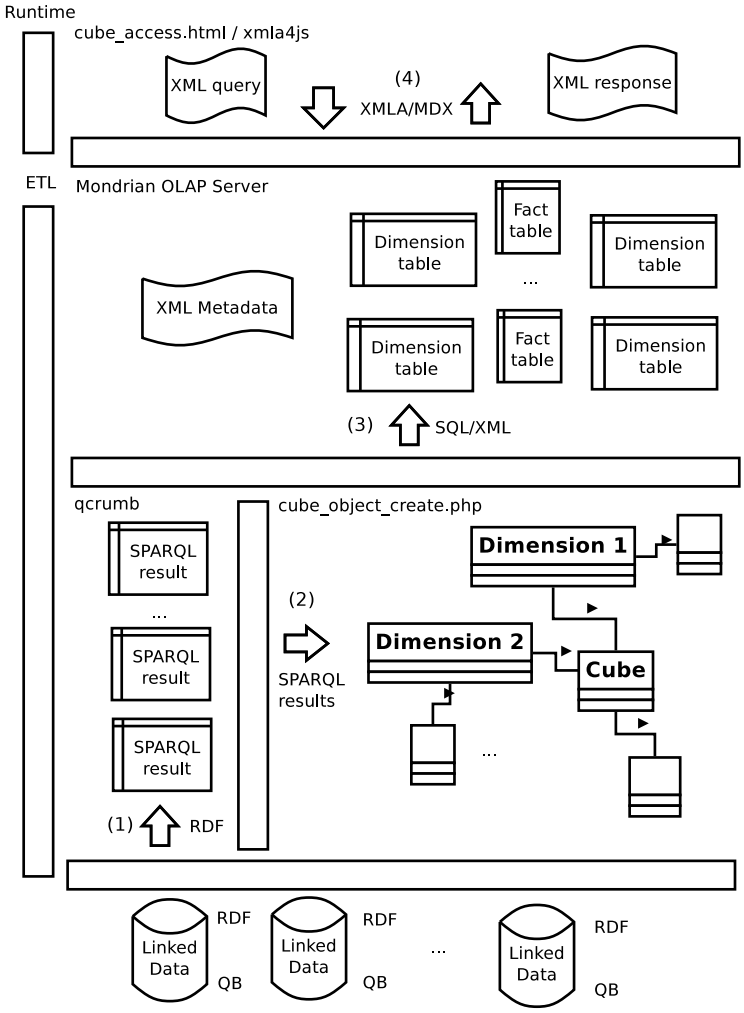


Figure 5.4: Architecture of Cube-to-ROLAP Prototype.

1) Dataset Selection: After the user specifies the dataset URIs, the relevant data about these datasets from Linked Data are retrieved (see bottom of figure).

2) Dataset Extraction: The mapping from Linked Data sources to our logical representation, we have implemented in a web application *cube_object_create.php* written in the programming language PHP 5.3.0. The pipeline has a semi-colon-separated list of dataset URIs as input.

Our system issues SPARQL queries for metadata about multidimensional elements over the relevant data as defined by the dataset URIs.

For extraction and loading of data, our system uses *qcrumb*[20] which allows to specify the location of the files in the FROM clause to issue SPARQL queries to the entire RDF graph as defined by the files' content. SPARQL results are represented in JSON format and parsed to an associative array.

According to our definition of *Relevant Data for a Multidimensional Dataset*, we create the set of location URIs for qcrumb incrementally. First, we only ask for the information URIs of the specified datasets. Then, after the data structure definition is known, its RDF/XML information URI is also added to the FROM CLAUSE of SPARQL queries.

Also, we manually added URIs providing information that is not contained in either dataset or data structure definition URIs but required for the mapping in the specific experiments. For instance, since only the members for Germany of the Employment Fear and GDP Growth datasets were linked but not the geo dimensions, we manually created a file with `owl:sameAs` triples[21] with the respective links and manually added it as a default URI to consider with qcrumb queries. The links between Germany `eurostat:dic/geo#DE` and `allbus:geo.rdf#00` from Eurostat and ALLBUS already are given by allbus[22].

For any queried dataset, we also added the query location URIs of instances of `skos:ConceptScheme` and `qb:DimensionProperty` to the SPARQL FROM clause.

While querying for dimensions and members, `owl:sameAs` links between entities are queried and entity consolidation is done. Hereby we might miss possible equivalence statements since for performance reasons we did not consider data sources specified by the object position of `owl:sameAs` triples nor data sources defined by member URIs. In our scenario, this was sufficient to retrieve all relevant equivalence links.

3) Data Warehouse Creation: As a representative OLAP engine, we use Mondrian, which uses XML for representing multidimensional elements and a star schema on an RDBMS (we use MySQL) as a data warehouse for the data about Dimensions and Facts.

[20] http://qcrumb.com/; last accessed on 2014-06-24.
[21] http://people.aifb.kit.edu/bka/Public/cube_additionalRDF.rdf, last accessed on 2014-06-30.
[22] http://lod.gesis.org/lodpilot/ALLBUS/geo.rdf, last accessed on 2014-06-30.

We have selected Mondrian, since it 1) is a widely-used Open Source OLAP engine (has been downloaded 82,243 times in 2013 according to the code repository SourceForge) and 2) has been used for several other work for evaluation [EV12a, DP08, NN09].

Each multidimensional element is logically represented (serialised) in either XML metadata or relational tables. For instance, from all observations, each represented in an array describing the observation values, we create an SQL INSERT query to populate the Fact table.

Every dimension, we represent as a "shared dimension" in Mondrian. Then, we represent the multi-cube consisting of all single Cubes. Multi-cubes can directly be described as "Virtual Cubes" in Mondrian[23].

Mondrian supports Multidimensional Expressions (MDX) to issue OLAP queries. Any number of MDX queries can be issued to the data in the warehouse.

Listing 3 shows an example multi-cube MDX query[24]. Here, a multi-cube of the two datasets "Unemployment Fear" and "GDP Growth" (separated by a semi-colon) is queried to retrieve the percentage of negative answers by survey participants and the average Real GDP growth rate given for Germany at every available point in time.

Listing 3: Example multi-cube MDX query for employment fear metric and GDP growth rate for Germany over time in UNEMPLOY query.

```
1 SELECT
2 {[Measures].[Percentage_of_Nos],
      [Measures].[obsValue_Unemployment_Fear_sum],
      [Measures].[obsValue_Real_GDP_growth_rate_avg]}
3 ON COLUMNS,
4 {Members([Date])} ON ROWS
5 FROM [Unemployment_Fear;GDP_Growth]
6 WHERE {[Federal_State].[Germany]}
```

4) Query Processing: OLAP clients provide an interface for users to issue queries to the OLAP engine. For the interface between OLAP clients and OLAP engine, we use XMLA. Since Mondrian natively supports XMLA, we can use any OLAP client implementing XMLA. As OLAP client, we created a JavaScript website, *cube_access.html*, based on xmla4js, that allows to connect via XMLA to an OLAP engine, to issue an MDX query and to display the results as a pivot table.

Figure 5.5 shows how our system is controlled from the JavaScript-based website. The website consists of four areas: In the "RDF Cube Dataset" area the user specifies a semi-colon-separated list of dataset URIs. Such URIs, the user typically finds on the respective human-readable websites. For instance, Estatwrap, provides a table of con-

[23] http://mondrian.pentaho.com/documentation/schema.php#Virtual_cubes; last visited on 2014-04-03.
[24] We have simplified the names. Our system uses URIs to identify multidimensional elements.

tents for all datasets[25]. After clicking on "Make accessible dataset...", the system runs the ETL pipeline to create the data warehouse and the "XMLA/A Connection" area shows information about the connected OLAP engine. In "MDX Statement" a query in the MDX language can be inserted and issued with "Execute Statement...". Finally, "Resultset" shows the requested pivot table. For instance, in 1980 more than 90% from 1,214 interviewees answered "No Fear of Unemployment".

Figure 5.5: Screenshot of Web interface of Cube-to-ROLAP Prototype after issuing a query over a multi-cube of two cubes.

5.3.2 Application and Testing

We first describe our setup and then the experiments. We run our experiments on a Microsoft Windows 7 64-bit workstation with Intel(R) Core(TM) i5 CPU, M520, 2.40GHz, and 4 GB RAM. Our system runs Apache web server, Apache Tomcat, and Mondrian OLAP Server (v3.2.1).

[25] http://estatwrap.ontologycentral.com/table_of_contents.html, last accessed 2014-04-03; for a dataset URI "#ds" has to be appended to the provided links.

With our evaluation, we intend to evaluate the following hypothesis: We can automatically transform Statistical Linked Data into a representation that can be used by existing OLAP engines for efficient query processing and integration.

We do an analytical evaluation. We run a performance analysis and describe experiments on a realistic workload motivated by the OGD scenario. We now describe how we run the three queries introduced earlier.

Unemployment Fear and GDP Growth (UNEMPLOY). The Unemployment Fear Survey Dataset published as Linked Data is distributed over several files for the actual data, the data structure definition, and the Member Values. The actual data of the survey dataset consists of 30 instances of `qb:Observation`. In total we are querying 1,547 triples.

The Eurostat GDP Growth Dataset[26] contains 320 observations. At the time of conducting the experiments, QB was not correctly used: Observations in the dataset did not fulfil the data structure definition they follow. We have made our system robust to this error so that such dimensions are ignored. Here, we are querying 18,721 triples. The large number of triples is mainly due to loading the code lists such as for the geo dimension with $13,481$ triples[27].

The ALLBUS and the Eurostat datasets are partly linked. ALLBUS links its values of the geo dimension[28] to values of the geo dimension of Eurostat. What is missing, however, is a link between both dimensions. Both dimensions describe a location where metrics have been taken from. For this experiment, we have created an RDF file containing the missing link and added it to the queried resources[29]. For the time Dimension, both datasets use the same property so that the links are automatically given.

We have run both datasets through our system. In total, the program ran for 273sec. Split up, the SPARQL queries on the datasets took 234sec; it took 27sec to create the MDM; and it took 12sec to serialise the data model for XMLA. Our system creates dimensions for Federal State (geo), Variable, Date, and a cube with a measure for the observation value for each dataset. One Cube contains measures aggregating the survey answers, e.g., by sum, the other Cube contains aggregated measures about the GDP Growth. Both Cubes share the Federal State (geo) and the Date dimensions. A multicube is created consisting of both Cubes.

Afterwards, we have run an MDX query that asks for the percentage of people saying that they had no fear of becoming unemployed. Calculating the percentage is an example of a compound measure. The MDX we have already shown in Listing 3. Listing 4 specifies the percentage calculation. We need this complex measure because the fear

[26] http://estatwrap.ontologycentral.com/id/tsieb020#ds; apparently in Eurostat was replaced by dataset with ID "tec00115" after conducting these experiments.

[27] http://estatwrap.ontologycentral.com/dic/geo, last retrieved on 2014-05-06.

[28] http://lod.gesis.org/lodpilot/ALLBUS/geo.rdf; last visited on 2014-04-03.

[29] http://people.aifb.kit.edu/bka/Public/cube_additionalRDF.rdf; last visited on 2014-04-03.

of unemployment dataset defines a Dimension `gesis:variable` that indicates the type of answer given by survey participants. The possible answers are "no fear", "fear to need to switch job", and "fear to become unemployed". The percentage of "no fear" among all answers is requested, which means to divide the sum of negative answers by the number of all answers given.

Listing 4: Compound measure to query for employment fear metric in UNEMPLOY query.

```
1  WITH MEMBER [Measures].[Percentage of Nos] AS
2  '([Measures].[obsValue Unemployment Fear sum],[variable].[Nein])/
3   ([Measures].[obsValue Unemployment Fear sum],[variable].[All
       variables])'
```

The MDX took 0.073sec to run. The result shows a table with 16 rows and three columns of aggregated measures. Each row indicates a year. The first column contains the percentage, the second column the sum of Unemployment answers, and the third column the GDP growth. Unfortunately, only for 2006 and 2008 both metrics are available, making the table very sparse. Yet, we have successfully integrated both datasets using an OLAP engine.

Number of Death by Illness and of Hospitals (HEALTH). Integrating the mortality metric from the *WHO Mortality Dataset* with the number of hospitals from Eurostat promises useful information. However, the published Statistical Linked Data is not sufficiently self-descriptive to be automatically mapped to our MDM. For instance, different from the vocabulary's guidelines, observations are not linked to a `qb:DataSet` from where an application can find a `qb:DataStructureDefinition` as a description.

Comparing EU 2020 Indicators (EU2020). We integrate four different *EU 2020 Indicator Datasets* from Eurostat as published from Estatwrap (EU2020a): Employment rate by gender, age group 20-64; Gross domestic expenditure on R&D; Greenhouse gas emissions, base year 1990; and Energy intensity of the economy. Altogether these datasets contain $1,247$ observations. Our ETL pipeline finished in 654sec on 24,636 triples that are related to these datasets. Afterwards, we ran a query to retrieve the metrics for all created measures showing their numbers over time and aggregating by average for all countries. To get a better impression about scalability of our system, we have run a fourth experiment (EU2020b) with the same datasets plus another four, e.g., the population at-risk-of-poverty or exclusion. Altogether, these datasets include $2,682$ observations.

In comparison to EU2020a, EU2020b involves twice as many datasets and twice as many observations; the SPARQL queries took 2.5 times as long as with four datasets. Creating the MDM took 3 times and serialising it (XML/SQL) took 2.7 times as long. Also, the same MDX query issued on these eight datasets took 2.9 times as long.

We manually checked that returned numbers resembled the content of integrated datasets.

In correspondence to our architecture, we measured the time for each step in an experiment. The scenarios are Unemployment Fear and GDP Growth (UNEMPLOY), Number of Death by Illness and of Hospitals (HEALTH), and Comparing four and eight EU 2020 Indicators (EU2020a and EU2020b). Table 5.1 summarises the results from our experiments.

In the table, we show the query execution time, including time for preparing the data, in relation to the number of triples in integrated datasets. Also, we investigate the relative time for the steps necessary to access and query the Linked Data sources. The steps correspond to the steps (1–4) in our architecture (Figure 5.4).

Table 5.1 describes the time it takes for Linked Data Wrappers to extract and convert the original data sources to RDF (column "SPARQL"), for transforming retrieved data into the representation of our MDM (column "MDM"), including consolidation, for deploying the representation in the ROLAP engine (column "XML/SQL", and for query processing in the ROLAP engine (column "MDX").

Table 5.1: Performance evaluation for each experiment (Exp.) with number of datasets (# DS), number of triples (# T), and evaluation time for each step corresponding to the system architecture.

Exp.	**# DS**	**# T**	**SPARQL (sec)**	**MDM (sec)**	**XML/ SQL (sec)**	**MDX (sec)**	**Total (sec)**
UNEMPLOY	2	20,268	234	27	12	0.073	273
HEALTH	n/a	n/a	n/a	n/a	n/a	n/a	n/a
EU2020a	4	24,636	580	36	38	0.161	654
EU2020b	8	35,482	1,417	116	105	0.473	1,638

In summary, we successfully integrated and queried several datasets. Since conceptually, there is no difference in representing the data in an in-memory multidimensional array or in an star schema, our evaluation more generally confirms the fact that common OLAP systems can be used. According to the results, the bottleneck lies in retrieving the data from the data sources and to run SPARQL queries on them to fill the star schema. Before analysis, few missing information such as equivalence links and compound measures need to be added by human analysts. Experiments indicate a linear increase of query execution time with increasing number of triples.

5.4 Discussions and Lessons Learned

In this section we discuss the results of the experiments from the previous section and give lessons learned.

We discuss the automatic transformation of Statistical Linked Data to our MDM, the advantages and disadvantages of reusing existing OLAP engines, and possible optimisations of data integration and query processing.

5.4.1 Automatic Transformation of Statistical Linked Data to Common Multidimensional Data Model

We were interested in the following problems:

- Despite Linked Data specific problems for preprocessing Statistical Linked Data, can we automatically gather all relevant data about a multidimensional dataset?
- Although there are no standardised formats for sharing of data cubes, can we automatically create an MDM from Linked Data reusing QB?
- Despite the semantic gap between the conceptual and logical level of multidimensional data models, can we integrate multidimensional datasets from the Web for analytical queries?

In our evaluation, we successfully 1) loaded relevant data using qcrumb.com, 2) populated an MDM from datasets published using QB, and 3) integrated real-world multidimensional datasets.

We executed different types of OLAP queries. In particular, in query UNEMPLOY (as shown in Listing 3), we filter for the Member Germany of the geo Dimension. In query EU2020 a/b we aggregate with AVG the datasets by the geo dimension and thus show how to reduce the dimensionality of a dataset. In all queries, we successfully showed how dimensions and members can be shared to compare different Measures in one pivot table.

The QB vocabulary seems to provide a suitable trade-off between convenience to publish and expressivity to make statistics self-descriptive. For instance, links between statistical datasets can be used to describe overlaps.

However, as apparent from the WHO mortality dataset relevant for the HEALTH query, the vocabulary is not always used correctly. For instance, with no `qb:DataStructureDefinition` given, a fully automatic transformation is not possible. Publishers possibly do not yet recognise the benefits of self-descriptive statistics in applications.

Still, our mapping could successfully be applied to analyse Statistical Linked Data with few manual efforts in pre-processing and querying as we describe in the following:

For instance, we showed that the directed crawling approach for relevant data is applicable as long as the data is sufficiently self-descriptive, provides the necessary links, and is correctly modelled according to the QB integrity constraints.

Since aggregation functions were not used by publishers, we automatically generated measures for each possible aggregation function. Additionally, MDX allowed us to manually create the compound measure that describes the employment fear metric and was not represented in the data.

Multi-level hierarchies are not used by publishers. For summarisability, we assume one hierarchy and level for each dimension, a meaningful aggregation functions, and facts with members on the lowest level (lean dataset).

We automatically created multi-cubes from all relevant datasets for each query and thereby successfully integrated Unemployment Fear and GDP Growth as well as several EU indicator datasets. To pre-process and integrate heterogeneous data to one global dataset, we evaluated OWL semantics with entity consolidation and made use of the possibility in Mondrian to define and query over "shared dimensions" and "virtual" multi-cubes.

We only had to manually add a link between the geo dimensions, since links between Germany members already were existing.

Multi-cubes are important to our goal to allow queries over all available datasets. In this chapter, we have shown how to build multi-cubes using RDBMS and OLAP engines. Also, we used equivalence mappings between datasets in Linked Data. In Chapter 6, we use the drill-across operation to build multi-cubes from data cubes represented in RDF. In Chapter 8, we use the drill-across operation to formally define a generalisation of multi-cubes, the Global Cube.

5.4.2 Advantages and Disadvantages of Reusing Existing OLAP Engines

The approach of using an existing OLAP engine for query processing and integration has the following advantages and disadvantages:

The advantages expected in the introduction were fulfilled:

- We used an existing OLAP engine, Mondrian, for query processing, including queries over several single cubes in multi-cubes and over compound measures.
- Mondrian could be replaced by other OLAP engines, in particular, by other ROLAP engines.
- Performance optimisations can be reused, e.g., Mondrian caches results of earlier queries. As long as the original data source does not change, after the Data Warehouse is populated, any number of queries can be efficiently processed by the OLAP engine.

- No SPARQL 1.1 with aggregation functions are needed, since metadata queries do not require aggregation functions and analytical queries are executed by the OLAP engine.

However, also disadvantages can be identified:

- An OLAP engine is required. Since there is no common interface to share data between existing OLAP engines, the system most probable needs to be adapted for other existing OLAP engines.
- Data queried from the OLAP engine has to comply to the MDM. Other information represented in RDF cannot be queried unless the star schema is modified.
- We were not able to test all common OLAP operations in our experiments. For instance, the vocabulary recommends ways to model Hierarchies and Levels. However, these possibilities are not used by publishers.
- Since the OLAP engine is used for query processing, we cannot further optimise OLAP queries over Statistical Linked Data.
- If data sources change, the entire ETL pipeline must be repeated to propagate the changes. Also Linked Data-specific optimisations are more difficult to organise such as gathering of all relevant data and incremental updates.

Our mapping is implemented as an ETL pipeline to store the RDF in a data warehouse for use by common OLAP systems. Another direction is to have OLAP operations directly on the RDF for which we would need a mapping of query languages used for OLAP and RDF. This possibility is investigated in more detail in the next chapter; in the remainder of this section, we describe first ideas.

SQL and RDBMS are often used as the underlying technology to store and query data cubes (ROLAP). Since RDF in a broader sense represent relations, we believe that SPARQL can be used to run queries over multidimensional datasets. We will not focus on the differences between SQL and SPARQL, here, but try to give an impression of how a basic analytical query with SQL corresponds to a query over RDF. Figure 5.2 illustrates a possible mapping between OLAP and SPARQL. A complete mapping in the form of an algorithm is presented and evaluated in Chapter 6.

Table 5.2: Preliminary mapping of OLAP and RDF query languages.

OLAP	**SPARQL**
Selection	Query for data with certain Dimensions.
Projection	Query for aggregated Measures.
Drill-Down/Roll-Up	Querying more/less fine grained values of Members.
Slice/Dice	Filtering on Facts with certain Members.

One can issue basic OLAP operations on cubes described by our MDM. OLAP allows to select dimensions, hierarchies, and levels to query for data from a cube. OLAP also allows to select Measures to aggregate. For instance, we can query a multi-cube of two datasets to retrieve the percentage of no answers by survey participants and the average Real GDP growth rate given for Germany at every available point in time. Listing 5 shows a SPARQL query that queries for such information from Estatwrap. Here, projection is done to select and aggregate the percentage and the growth rate. However, from the "Unemployment Fear and GDP Growth" implementation, we have seen that the percentage of negative answers forms a compound measure calculated from several metrics on different levels of detail (see Listing 4), which would not be as easy as indicated in this example, and may require SPARQL subqueries.

Listing 5: SPARQL query for employment fear metric and GDP growth rate for Germany over time in UNEMPLOY query.

```
1  SELECT ?time ?geo avg(?nos) avg(?grorate)
2  WHERE {
3  ?s qb:dataset <http://estatwrap.ontologycentral.com/id/tsieb020#ds>
       .
4  ?s dcterms:date ?time .
5  ?s eus:geo ?g .
6  ?g rdfs:label ?geo FILTER(?geo = "Germany") .
7  ?s eus:nos ?nos .
8  ?s eus:growthrate ?grorate
9  }
10 ORDER BY ?geo
```

Drill-down or roll-up lead to more granular or less granular results. For instance, if we drill-down the geopolitical entity from country to city, each country is split up in its cities resulting in more fine grained information regarding the projected metrics.

In SPARQL, one would instead of directly query for the label of the member, use a query pattern that groups the Members and then query for the label of the grouping resource. However, this will lead to slow queries, since different from the star schema representation where every member exactly has one value for each level, in RDF, queries always assume members to have any number of level values.

Slice (and dice) fix dimensions on one member (several members) and denote subsets of the data. In our example, we use this functionality to filter for information about Germany. In SPARQL terms, a slice corresponds to filter patterns.

5.4.3 Possible Optimisations of Data Integration and Query Processing

For every approach presented in this theses, we are interested in the improvements in flexibility and efficiency. In the approach presented in this chapter, we argue for

an increased flexibility since links between dimensions and members can be added for increasing the overlap between published multidimensional datasets. Efficiency is increased by using existing OLAP engines that deploy optimisations such as the efficient star schema on a relational database.

After the MDM has been successfully serialised into a data warehouse, the actual OLAP operations only take an instant. The performance of our mapping is assessed best by the time it takes to create the MDM and to serialise it.

Evaluation times for transforming retrieved data into the representation of our multidimensional data model (including consolidation) and for deploying the representation in the ROLAP engine take similar amount of time.

Our system terminated at every experiment. Only the HEALTH query could not be executed successfully since relevant data has not been properly published as Linked Data.

Since Eurostat datasets share their dimensions and members (use the same URIs), no explicit consolidation has to be done. Only for the UNEMPLOY query `owl:sameAs` links between Dimensions and Members had to be added, and possibly result in a slightly higher evaluation time for representing the MDM (27s versus 12s).

The performance can be improved: for instance, instead of querying for all Dimensions and Members and to then resolve equivalence relationships in a reflexive, symmetric and transitive closure table to find unique Dimensions and Members, entity consolidation [HHU+11] could be done directly on the RDF. Hogan et al. [HHU+11] present two algorithms that, depending on the number of equivalence statements, perform faster or slower. Currently, we only consider `owl:sameAs` links, other reasoning engines would be capable of also considering other forms of equivalent statements, e.g., `owl:equivalentProperty` and `rdfs:subPropertyOf`.

So far we built the entire MDM as described by the Linked Data about the input datasets; instead, modelling could be more directed by user queries as sometimes is done in Web Warehousing [RALT06].

The bottleneck of our system are the SPARQL queries; per run of the ETL pipeline, there are $\#ds + 6 * \#dim + 4$ SPARQL queries issued, with $\#ds$ as the number of datasets and $\#dim$ as the number of dimensions. For example, we do not consider distributed storage of the actual data and its metadata and always issue the queries to the entire set of triples related to the datasets.

Note, in our experiments, SPARQL query times do not only consist of the time for retrieving the RDF files from the servers and executing the queries, but also includes the time that Linked Data wrappers may need to produce RDF from the original data sources. For instance, Estatwrap needs to retrieve SDMX files from the Eurostat website and to transform SDMX to RDF. Thus, possible bottlenecks may depend on the concrete scenario.

Also, we do not distinguish between URIs that provide useful information about metadata such as Cubes and Dimensions and the actual data in the form of Facts. Since the data usually is larger than the metadata, leaving it out for metadata queries may lead to faster execution.

Experiments indicate a linear increase of query execution time with increasing number of triples, but our datasets have not had very large numbers of observations and dimensions.

Once data is serialised for the OLAP engine, we expect queries to be fast as indicated in the results.

At least regarding an increasing number of observations, we do not expect an overly negative impact on the performance of representing the MDM and serialising the MDM in XML and SQL, since single Facts from observations represented in arrays are directly used to create an SQL INSERT query to populate the Fact table.

With increasing number of Dimensions or Measures, our logical representation needs to store more multidimensional elements and needs more memory for the objects but compared to facts, the number over dimensions is not expected to be overly large.

There is the question whether a more direct mapping between Linked Data and the OLAP engines representation would lead to better performance. We assume that in most cases, an in-memory representation of multidimensional elements from which a serialisation for OLAP engines is created, performs best.

For instance, a transformation directly on the RDF files, e.g., using XSLT, 1) would not be applicable to RDF formats other than RDF/XML and 2) since the same RDF can be represented by different XML documents, only would be on a syntactic level, would be more complex, and would be less robust to changes in modelling.

Checking for the right modelling of datasets with integrity constraints also requires additional execution time.

The efficiency of the given approach can be further optimised by 1) more efficient loading such as parallel threads for loading the triple store and 2) by using more scalable data warehouse backends such as an in-memory and column-oriented database.

5.5 Related Work

We distinguish related work about semi-automatic multidimensional modelling from Web data and about logically representing formalised MDM for integration and queries.

5.5.1 Semi-Automatic Multidimensional Modelling from Web Data

We focus on Linked Data using QB. Other work tries to do automatic multidimensional modelling from other data sources.

Modelling from Ontologies: Several work recommends the use of ontologies for Data Warehouse design.

An overview of challenges in Data Warehouse Modelling and Design, e.g., the Semantic Gap, Web Warehousing and effective Conceptual Modelling using Ontologies, is given by Rizzi et al. [RALT06].

There is recent work on creating Multidimensional Models from ontologies [RA07a, NN10, PM11, NB12]; For instance, Romero et al. [RA07a] propose a semi-automatic method to find multidimensional concepts from heterogeneous data sources sharing a common ontology in three steps: 1) identifying facts, 2) identifying dimension keys and 3) identifying dimension hierarchies.

Nebot et al. [NB12] describe how to semi-automatically derive an MDM from Semantic Web data. In their architecture, they describe a Fact Extractor and a Dimensions Extractor that produce data for a Cube Generator. The Cube Generator populates a Data Warehouse and OLAP engine similar to our ETL pipeline. Analysts then can use an OLAP engine to analyse the data. The Dimension Extractor allows to automatically derive useful dimension hierarchies. For that the authors measure the quality of dimension hierarchies. Data sources need to be represented in domain-specific OWL ontologies for which the authors define an OWL ontology for MD schema specification. Users can then map domain-specific OWL ontologies from the Semantic Web to this OWL ontology using DL constructs. The authors use MySQL to store the semantic annotations, and the Business Intelligence (BI) tool of Microsoft SQL Server 2008 to instantiate the MD schema designed by the user, create cubes, and allow OLAP queries. However, the authors only evaluate the semi-automatic multidimensional modelling according to performance and quality, but do not give details about query processing performance; also, no integration of several datasets is done.

Also, methods to help designers with multidimensional modelling to reduce the semantic gap have been developed: Lujanmora et al. [LMTS06] proposes UML for multidimensional modelling.

The work closest related to our approach is by Niinimäki and Niemi [NN09]. They present an own RDF/OWL ontology as a representation for an MDM. Their system follows steps that are similar to ours:

1. They assume raw data available on the Web.

2. The raw data is converted to RDF using an OWL ontology describing the MDM. For that, "ontology maps" are assumed to be available that describe transformations (often based on XSLT) from an XML document to an RDF document.

3. An OLAP cube is constructed by posing queries to the RDF files. Queries can be generated automatically since the RDF is conforming to a known ontology.

4. OLAP cube is deployed to the OLAP engine Mondrian.

5. User can analyse data using a typical OLAP client.

There are two main differences between their [NN09] and our work:

No Linked Data principles for accessing data but XML files with raw data downloadable from the Web. We assume available Linked Data using QB, whereas the authors assume existence of ontology maps to automatically transform XML to domain-specific ontologies reusing an OLAP model. Also in our case, often raw data will first be transformed to QB for which manually transformation scripts need to be created. We represent our MDM – apart from the facts – as an object-oriented data structure whereas they directly use SPARQL query results to deploy the populated OLAP model on the server. For the facts, we similarly create SQL queries directly from the SPARQL results. The main reasons for us to have an intermediate representation of multidimensional elements are 1) that for some elements there is no direct correspondence in QB, e.g., for aggregation functions and 2) that we consolidate shared Dimensions and Members for integration.

As for the evaluation, the conversion of a data source to RDF as well takes most of the time. This is similar to our approach where SPARQL queries also may include the time for wrappers to produce RDF from original data sources. Another bottleneck in their work lies in converting query results to SQL insert/replace statements. In the case of facts, we are doing the same, directly from the SPARQL results.

The work does not consider integration of datasets. Also, the authors do not focus on the problem of semantic heterogeneity in datasources to be integrated in an MDM. They put much focus on their ontology, which directly models an MDM. Our approach is based on a vocabulary that already has been adopted by different parties and we focus on mapping the statistical data to a meaningful MDM.

Etcheverry and Vaisman [EV12a] argue for the use of ontologies available on the Web and consider Statistical Linked Data as a possible data source. They investigate whether it is possible to use web data to enhance local OLAP analysis. They emphasise the importance of situational Business Intelligence that allows ad-hoc analytical queries to Web data without the burden of incorporating data sources and data requirements into existing Data Warehouses. They introduce Open Cubes [EV12b], also called

QB4OLAP[30], a vocabulary specified using RDFS that allows to represent multidimensional datasets. The vocabulary builds on QB and adds aggregation functions and explicit multi-level hierarchies, but is not adopted by publishers. The authors assume that subsets of web cubes contain relevant data with which to enhance local cubes.

Similar to our work, they sketch a mapping from a Web cube to the multidimensional data model that can be deployed on OLAP engines such as Mondrian.

They built a similar pipeline to transform web cubes into a star schema.

Different from our work, the authors do not explain where the Web Cube descriptions can automatically be found on the Web; instead, our approach and experiments use QB datasets with resolvable URIs. Most importantly, Etcheverry and Vaisman [EV12a] do not go into details of integrating datasets.

It remains unclear how dimensions represented using URIs in a Web cube can be aligned with dimensions represented with application-specific identifiers of a local OLAP cube. Also, no descriptions are added how data integration is done in the OLAP engine.

Although the authors introduce explicit aggregation functions for measures, they also add them to operators. In their algorithm, they ask for the aggregation function of each measure. Thus, they also assume meaningful aggregation functions to be pre-defined and – similar to our work – do not further investigate the summarisability problem.

Modelling from Graphs: Several work try to execute OLAP queries over graph data. Inoue et al. [IAK13] try to allow users to analyse general RDF graphs from Linked Open Data using OLAP. For that the authors present a framework with which to extract Linked Open Data and to semi-automatically model the data as data cubes. Their approach does not require the RDF data to comply to a specific vocabulary such as QB. Instead, users are required to manually select facts and measures of a data cube. From this input, the system automatically identifies possible dimensions from cardinalities of relationships between facts and attributes. Also, dimension hierarchies are identified from concept hierarchies.

In Graph OLAP [CYZ+08], a set of graphs, e.g., a graph of conferences with papers of authors is defined by so-called info dimensions, e.g., year, and topology dimensions, e.g., affiliation of authors. In I-OLAP, OLAP operations are executed over info dimensions to result in similarly-structured graphs, e.g., the graph of conferences with papers of authors in a specific year. In contrast, in T-OLAP, OLAP operations change the topology of graphs, e.g., transform of a graph of authors to a graph of institutions affiliated to the authors.

In Graph Cube [ZLXH11], OLAP queries over a multidimensional network, e.g., a social network, result in cuboids. Cuboids are aggregate networks corresponding to a

[30] http://publishing-multidimensional-data.googlecode.com/git/index.html, last visited on 2014-04-06.

specific aggregation of the multidimensional network. Beyond cuboid queries, cross-cuboid queries are defined that allow to ask about the network structure between different cuboids, e.g., the difference between the network of users and the network aggregated to single locations.

Beheshti et al. [BBNA12] define GOLAP, a graph data model for OLAP on RDF graphs and present an extension to SPARQL for OLAP over RDF. This extension also allows to define compound measures.

Although multidimensional modelling from graphs can be applied to QB data [MCG11], we see this line of work as complementary to our work since it addresses the following problem: More and more Linked Open Data comprising multidimensional data is published, yet, few techniques have been investigated of how to make sense of that graph-structured data. For this line of work, multidimensional modelling from graphs is a promising technique.

Instead, our work is more concerned with the following problem: Nowadays, when distributed datasets are to be analysed, often Excel Pivot, ROLAP or MOLAP methods are applied, with a lot of manual effort. We propose to use Linked Data as the common data representation and exchange method to improve data integration capabilities without loosing intuition and efficiency of data analyses.

5.5.2 Representing Formalised Multidimensional Data Models

There are several ways to logically (formally) represent multidimensional datasets so that they can be easily and correctly used for integration and queries. We use an object-oriented representation of the MDM based on our MDM-QB mapping.

Pardillo et al. [PMT08] try to solve the semantic gap with a model-driven architecture and the definition of OLAP operations over an abstract model that is independent of a logical implementation. We instead use our abstract model to populate a data warehouse and for query processing use existing OLAP engines.

Integration of datasets: Kimball [KR02, p. 79] was the first to recommend having shared (conformed) dimensions of coexisting data warehouses that adhere to a standard; common dimensions of data warehouses would be collected and communicated using a so-called *data warehouse bus matrix*. Kimball requires shared dimensions to have identical schema and data. Similarly, we require that members can only be shared if also the respective dimension, hierarchy, and level are shared. Whereas Kimball requires a dimension to be shared up-front during design time, we consider the case where dimensions are mapped on-demand. In those cases, two cubes may originally use the same dimension with different members that only later are brought together in one shared dimension. For instance, whereas one data cube may only include European countries for the geo dimension, another data cube may describe Non-European countries. Based

on shared dimensions and members, we define multi-cubes and allow drill-across over several cubes using OLAP engines.

Whereas in our work we apply equivalence statements for defining shared dimensions and members, other work tries to automatically learn such statements. For instance, RUBIX [ALS+12] maps instances of cell values to instances and column headers to type information from Linked Data for the integration of tabular data. Zapilko and Mathiak [ZM14] find matching object properties from Statistical Linked Data by their links to individuals of semantically similar code lists. The authors also argue that traditional ontology matching approaches are not effective in identifying matching object properties. Ermilov et al. [EAS13] propose the collaborative mapping of tabular data using a semantic wiki where users can map table columns.

Hümmer and Bauer [HBH03] present an XML schema to share Data Cubes and to act as a basis for integration. XCube is an approach to define a standard logical representation of multidimensional datasets to be shared over the Web.

According to XCube, there is not much work on creating an XML standard for sharing MDMs. Previous work is focusing on the metadata or on queries for specific data but not on sharing the entire Data Cubes.

XCube contains formats to share Cubes, Dimensions and Facts. Also, collections of Data Cubes can be represented. However, the multi-cubes mentioned in this work do not align with our definition of a multi-cube which is to refer to integrated Data Cubes consisting of data from several Data Cubes.

Regarding integration, they mention the use of standardized reference dimensions as a promising tool for integrating data from various sources. They also refer to keys that can be given to Dimensions for unique identification. However, they do not further describe the possibility to integrate several Data Cubes published using XCube. Also, they do not consider heterogeneity or semantic conflicts.

In our work, we do not only share the metadata of a Cube and split up queries to single Data Warehouses in a federated way, but collect both metadata and data to create a multi-cube from single cubes based on Shared Dimensions and Members. We show how to store the multi-cube in a Data Warehouse for use by an existing OLAP engine. In later chapters, we extend the notion of multi-cubes to the Global Cube over all available datasets.

Summarisability: Although the use of formal domain models promise benefits in solving summarisability issues, in our mapping between QB and our logical representation of an MDM, we simplify the problem and automatically create measures for all possibly correct aggregations functions such as COUNT and AVG.

Automatically checking summarisability is a complex problem and often requires background information such as dimension constraints [HGM05]. For instance, Niemi and

Niinimäki [NN10] automatically conclude summarisability from OLAP queries by describing measure units, measure types, dimension types, and aggregation functions in an OLAP ontology. Similar to our work, they assume complete and disjoint hierarchies and the OLAP ontology populated with the respective information.

Similarly, Prat et al. [PMA12] define an OWL-DL ontology that also allows to automatically derive summarisability. Again, the respective domain knowledge has to be (manually) formalised using that ontology.

QB also is an OWL-DL ontology which however does not cover the possibility to explicitly describe measure types as required to infer summarisability [LS97]. One reason for that is that QB tries to be simple to use; although with `qb4o:aggregateFunction` there is a way to directly add correct aggregation functions to measures, this extension of QB is not used by publishers.

5.6 Conclusions

Along example queries motivated by the Open Government Data scenario, in this chapter, we have presented an approach, the MDM-QB Mapping, to map statistics published on the Web to our MDM that can be automatically deployed on an existing OLAP engine for efficient analytical query processing.

The mapping between the RDF Data Cube Vocabulary and our MDM includes a definition of relevant data possibly distributed over several sources, requirements for meaningful aggregations, and the possibility to integrate several datasets into multi-cubes.

Reusing existing OLAP engines allows efficient query processing, also over multi-cubes. Flexibility is increased, since dimensions and members also can be implicitly shared via equivalent statements. We expect that the flexibility can be further increased by reusing existing SPARQL engines for query processing, which we investigate in the next chapter.

Also, we define relevant RDF data describing multidimensional datasets and lean datasets that ensure summarisability.

We have implemented the mapping in a system and evaluated it with the example queries; results indicate that an automatic and scalable transformation into an MDM and efficient query execution over Statistical Linked Data are possible.

Published datasets do not use all features, e.g., no multi-level hierarchies. Also some datasets are not self-descriptive enough to be automatically analysed. We believe that more datasets – also including multi-level hierarchies – are published and in Chapter 6 and Chapter 7 investigate query processing over multi-level hierarchies and larger datasets using existing SPARQL engines.

In future work, it would be interesting to investigate whether summarisable aggregation functions of measures and compound measures also can automatically be identified from ontological information [NN10, PM11] in Statistical Linked Data.

For instance, mathematical functions could be explicitly stated with the statistical data, for which there are various ways [VLH+10].

In case datasets exhibit modelling errors – as in our real-world examples – a more robust interpretation or an automatic repair are needed. Also, it would be interesting to evaluate the possibility to relax the limitations of lean datasets; for instance, the population dataset of Eurostat contains both facts for single genders and total values.

Current Statistical Linked Data is published in a batch mode. In the future, with statistical data generated and published in near real-time, relevant data about multidimensional datasets for a query dynamically change and require the consideration of update-time intervals [HKS+13] and published deltas of datasets. This issue also holds if a federated query approach such as over SPARQL endpoints is used; SPARQL endpoints often are populated in a batch mode, provide limited querying capabilities (e.g., max query execution time, max results, limited to SPARQL 1.0). Also, SPARQL endpoints do not provide continuous "streaming" data and need a polling-strategy in case query results are to be cached for faster repeated queries.

6 Executing OLAP Operations Using SPARQL

In this chapter, we investigate the following research question:
Research Question 2. *How can we use existing SPARQL engines for query processing and flexible integration of multidimensional datasets from the Web?*

Figure 6.1 illustrates the contribution given in this chapter. We present an *OLAP-to-SPARQL Algorithm* that translates OLAP to SPARQL queries over Statistical Linked Data for more flexible query processing using SPARQL engines in comparison to using an existing OLAP engine with a fixed star schema.

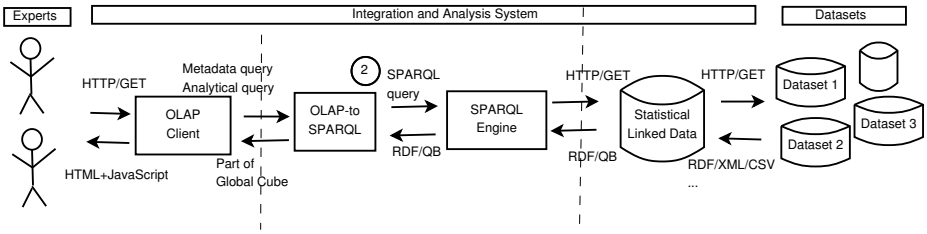


Figure 6.1: Illustration of contribution of the OLAP-to-SPARQL Algorithm.

6.1 Introduction

The fact that in the approach presented in the previous chapter OLAP queries are executed not on the RDF directly but by a traditional OLAP engine after automatically populating a data warehouse results in the following drawbacks:

1. Although our ETL pipeline showed capable of automatically creating a quasi-standard logical representation (ROLAP) for data warehouses, our approach requires an OLAP engine to execute OLAP queries. Also, our logical representation does not allow to store other information than previously defined.
2. Since the OLAP engine is used in a black-box fashion we cannot further optimise OLAP queries over Linked Data sources.

3. For the same reason, if Linked Data sources are updated, e.g., if a single new statistic is added, the entire ETL process is repeated to have the changes propagated.
4. Integration of additional data sources is difficult, since the MDM has a fixed schema that does not allow to easily attach background information.

To reduce the processing effort of managing the OLAP engine and to have OLAP operations directly over the RDF, we would need a mapping of query languages used for OLAP and RDF. We need to investigate the following problems:

- SPARQL is an expressive query language on RDF and with the new version SPARQL 1.1 supports aggregate and subqueries; yet, it is unclear whether SPARQL expressivity is sufficient for analytical queries and whether the evaluation of analytical queries with SPARQL engines is sufficiently efficient.
- Since RDF is a more flexible data model than the relational data model, the performance of SPARQL engines over QB may be different from the performance of OLAP engines such as RDBMS over a Star Schema.

For example, in the XBRL scenario, a business analyst may want to assess companies. The *Edgar Linked Data Wrapper* (Edgarwrap)[1] provides access to XBRL filings from the SEC as Linked Data reusing QB. For instance, one can find the quarterly balance sheet of Rayonier Inc.[2] disclosing a sales revenue net of 377,515,000 USD from 2010-07-01 to 2010-09-30.

Assuming we have access to a data cube `edgar:SecCubeGrossProfitMargin` with financial facts from Edgarwrap. All facts are fully dependent on the following dimensions: the disclosing company (Issuer), the date a disclosure started (Dtstart) and ended (Dtend) to be valid, and additional meta information (Segment). Also, every fact discloses Cost Of Goods Sold (`edgar:CostOfGoodsSold`) and Sales Revenue Net (`edgar:Sales`) as measures with unit USD. Then, an analyst may ask the following analytical query.

Cost of Goods Sold for selected companies (COST). In the XBRL scenario, a business analyst may want to compare the number of disclosures of Cost of Goods Sold for certain companies. He requests a pivot table with issuers *RAYONIER INC* and *WEYERHAEUSER CO* on the columns, and the possible periods for which disclosures are valid on the rows, and in the cells showing the number of disclosed cost of goods sold, or – if only one – the actual number. Figure 6.2 shows the requested pivot table.

The analyst may want 1) to have access to background information to increase the trust in presented data, e.g., data transformations executed over the data [FKaGO+12] and

[1] http://edgarwrap.ontologycentral.com/, last accessed 2014-12-10.
[2] http://edgarwrap.ontologycentral.com/archive/52827/0001193125-10-238973#ds, last accessed 2014-11-24.

Columns (issuer) Rows (dtstart, dtend)		RAYONIER INC	WEYERHAEUSER CO
2009-01-01	2009-3-31	*1,100,335 USD*	*0 values*
2009-04-01	2009-06-30	*2 values*	*2,300,800 USD*
...	...	...	...

Filters: CostOfGoodsSold

Figure 6.2: Pivot table requested in our COST query.

2) to consider background information in queries, e.g., filter for companies from cities with more than 100,000 inhabitants.

As another example, in the OGD scenario, a citizen may want to compare values from Estatwrap datasets. For example, two example queries were introduced in Chapter 5 and require the integration of several data cubes:

Unemployment Fear and GDP Growth (UNEMPLOY): Citizens want to compare the indicator about unemployment fear from dataset `allbus:ZA4570v590.rdf#ds` with the "GDP Growth" from dataset `eurostat:id/tec00115#ds` over time for Germany to get insights about the relation between GDP and employees' perceived situation.

Comparing EU 2020 - Indicators (EU2020): Here, citizens want to aggregate and compare important metrics about European countries by average for all countries and to show the aggregated numbers per year, so that trends of important indicators for European countries become visible.

In this chapter, we are looking for a way to execute OLAP queries over one or more datasets directly using a SPARQL engine.

More concretely, we 1) simplify the ETL pipeline to crawling all relevant RDF 2) allow flexible integration of background information, and 3) generate and query multi-cubes using the drill-across operation. We approach the following problem as illustrated in Figure 6.3:

On the front-end (left of figure), given a common OLAP client capable of running OLAP operations on data cubes. At the backend, a SPARQL engine such as a triple store with data crawled from Linked Data and accessible via a SPARQL endpoint. The problem is to design a mediator [CDL+01] (combined OLAP engine and Triple Store in middle of figure) to collect, clean, and combine data from different Linked

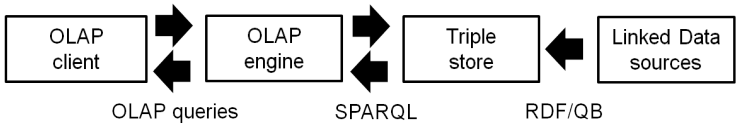


Figure 6.3: Data flow for OLAP queries over Statistical Linked Data using SPARQL engine.

Data sources (e.g., wrappers), so as to meet an information need of the integration and analysis system specified as an OLAP query.

As an overall contribution, in this chapter, we show how to map OLAP queries to SPARQL queries. More specifically, our contribution are as follows:

- We show how to transform an OLAP query to a SPARQL query which generates all required facts from the data cube (Section 6.2.1). For that, we define OLAP queries over Statistical Linked Data as nested sets of OLAP operations.
- We extend this definition for queries over several data cubes with the drill-across operation to generate multi-cubes (Section 6.2.2). The evaluation of drill-across considers implicitly shared dimensions and members and is compared with results from the previous chapter using an OLAP engine.

The result is an OLAP-to-SPARQL algorithm that we evaluate in Section 6.3 both with the COST query motivated by the XBRL scenario and with the UNEMPLOY and EU2020 queries introduced in Chapter 5 motivated by the OGD scenario.

In Section 6.4, we discuss the results and describe some lessons learned, after which, in Section 6.5 we describe related work. In Section 6.6, we conclude.

6.2 Approach: OLAP-to-SPARQL Algorithm

In this section, we present the *OLAP-to-SPARQL Algorithm* to execute OLAP queries using SPARQL over Statistical Linked Data, as previously illustrated in Figure 6.3.

Given a data cube (cs, F) with data cube schema $cs = (?x, D, M)$ as well as facts on the lowest level of each dimension $(?obs, C, E) \in F$, we define a *materialised data cube* as per Definition 10.

Definition 10 (Materialised Data Cube)**.** *Adopting the concept of Gray et al. [GCB+97], we can fully materialise a data cube represented in QB and compute all facts* $(nameF, C, E) \in String \times 2^{Dimension \times Member} \times 2^{Measure \times String}$ *as follows: We extract a relational table containing attributes for measures* $M = \{M_1, ..., M_{|M|}\}$ *and dimensions* $D = \{D_1, \ldots, D_N\}$*. The relation contains an entity with measure values for each possible combination of dimension members from the data cube. We compute* 2^N *aggregations of each*

measure value by the measure's aggregation function over all possible grouping values*, i.e., subsets of dimensions* $gv \in 2^D$*, with a standard* GROUP BY*. Then, we merge all aggregation results with a standard* UNION*, substituting the special* ALL *value in the aggregation columns for each aggregated dimensions* $ad \in 2^{D \setminus gv}$.

Subqueries and aggregation functions in SPARQL 1.1 make it easily possible to apply the relational concept of Gray et al. [GCB+97] to a data cube represented as Linked Data reusing QB.

According to the definition, if the number of possible members of a dimension is $card(D_i)$, then $\prod(card(D_i)+1)$ is the number of facts in the materialised data cube. Also, the number of required aggregation queries (subqueries) growth exponentially with the number of dimensions. The definition shows that the potential size of a data cube with many dimensions from Statistical Linked Data is too large to pre-compute and store. The SPARQL query would have an exponential number of subqueries and would take a long time to execute. Also, the query would compute all possible facts from a data cube, although OLAP queries may require only a small subset. Instead, it is our goal to compute on-demand all requested facts from a data cube.

Therefore, in the following, we first show how to evaluate single-cube OLAP queries and compute the requested facts directly without subqueries and without fully materialising the cube (Section 6.2.1). Then, we show how to evaluate multi-cube OLAP queries (Section 6.2.2).

6.2.1 A Mediator for Evaluating Single-Cube OLAP Queries Using SPARQL

In this section, we describe an approach to translate OLAP queries over single data cubes to one SPARQL query over Statistical Linked Data. For that, we define a data structure with which to represent OLAP queries. For query processing, we explain how to translate a nested set of OLAP operations to a subcube query and how a subcube query can be evaluated with SPARQL over RDF reusing QB.

In this case, multiple datasets can be queried together if they are described in multi-cubes using the same `qb:DataStructureDefinition`. Similar as for the mapping of QB datasets to an MDM and preparation for an OLAP engine in Chapter 5, we assume for OLAP-to-SPARQL a *lean dataset* and do not need to consider redundant observations for summarisability.

For interpreting a set of OLAP operations and evaluating the OLAP query with SPARQL on QB, we use and slightly adapt the syntax and semantics of *subcube queries* [LHG04].

Given a data cube $c = (nameDC, cs, F) \in DataCube$, with $cs = (?x, D, M) \in DataCube\text{-}Schema$, $F \in 2^{Fact}$, $D = \{D_1, D_2, \ldots D_N\} \subseteq Dimension$ an ordered list of dimensions

with a set of levels $L_i = \{l_1, l_2, \ldots\} \subseteq Level$, including the special-type *ALL* level; with each level l_i having $memberno(l_i)$ members; $M \subseteq Measure$ an ordered list of measures. Then we represent an OLAP query over this cube as a subcube query as per Definition 11.

Definition 11 (Subcube Query)**.** *A tuple $Q = (c, SlicesRollups, Dices, Projections)$, with $SlicesRollups \subseteq Level$ a level for each dimension (for roll-ups), including the special-type level ALL (for slices), with $Dices \in 2^{\mathcal{V}}$ member combinations, as positions (for dice) and with $Projections \subseteq M$ a set of selected measures from the data cube (for projection). For simplicity reasons we assume the same, fixed, ordering of dimensions in the cube and in the subcube query tuple. A subcube query returns a set of facts from a materialised data cube as per Definition 10.*

As example, we use the data cube *SecCubeGrossProfitMargin* motivated by our XBRL scenario. Given an MDM with one data cube that only exhibits hierarchies with one level, we can use an abbreviated syntax of a subcube query $Q = (q_1, ..., q_N, m)$, with $q_i \in \{?, ALL, x\}$, $m \in 2^M$ to illustrate the OLAP-to-SPARQL Algorithm [KOH12]. The subcube query tuple contains for each dimension a tuple element q_i, $Dimension(q_i) = D_i \in D$, $Hierarchy(q_i) = H_i \in Hierarchy$, $Level(q_i) = L_i \in Level$. The element "?" marks a dimension as *inquired* and refers to the bottom level $BOTTOM$; the *ALL* marks a dimensions as *aggregated* and refers to the unique top level *ALL*; and the $x \in \mathcal{V}$ aggregates over a *fixed* set of members of a dimension. Also, for any queried measure the subcube query tuple contains an $m_i \in m$. For each dimension a granularity in the form of a hierarchy and level is specified.

As examples, we describe three distinguishable subcube queries:

- *Full-cube query* $(?, ?, ?, ..., M)$ returns the facts on the highest granularity, i.e., the lowest level of each dimension, inside a data cube.
- *Point query* $(a_1, ..., a_{|D|}, M)$ with $a_i \in \mathcal{V}, |a_i| = 1, Dimension(a_i) = D_i$ returns one specific fact from a data cube.
- *Fully-aggregated query* $(ALL, \ldots, ALL, M)$ returns one single fact aggregated over all facts (over all dimensions, with no grouping value).

As a pre-processing step before interpretation using SPARQL, in the following, we describe how any nested set of OLAP operations can be translated to one specific subcube query. For that, we show how to evaluate each OLAP operation in terms of subcube queries.

Given a data cube as input to an OLAP operation, we use the full-cube query tuple $(?, ?, ?, ..., M)$ as an initial subcube query. Then, we translate any nested set of OLAP operations without Drill-Across over this data cube by going through the OLAP operations and modifying in turn the subcube query tuple as defined for each operation as follows:

Projection. We evaluate $Projection(c, PM) = c', (c, PM) \in DataCube \times 2^{Measure}$ with PM the selected measures, and M the measures of cube c by removing every non-selected measure $nsm \in M \backslash PM$ from the subcube query tuple.

Dice. We evaluate $Dice(c, dd, DM) = c', (c, dd, DM) \in DataCube \times Dimension \times 2^{Member} \rightarrow DataCube$ with dd the diced dimension and DM the set of selected members as follows:

In a subcube query, dimensions can either be *inquired* or *aggregated*. Any *fixed* dimension also is aggregated.

Therefore, we retrieve all diced members for the respective dimension from DM.

If the dimension is *aggregated*, we set the tuple element of the dimension to the specific set of members[3]; if the diced dimension is *inquired*, we do not change the subcube query since fixing the dimension would also aggregate over the dimension (according to the subcube query definition).

Slice. We evaluate $Slice(c, SD) = c', (c, SD) \in DataCube \times 2^{Dimension}$ with SD the sliced dimensions by setting the value of every sliced dimension $d \in SD$ to ALL.

Roll-Up. We evaluate $Roll-Up(c, rd, rl) = c', (c, rd, rl) \in DataCube \times Dimension \times Level$ with rd the rolled-up dimension, rl the selected level by setting the respective level rl of dimension rd.

As an example, consider the OLAP query described by the MDX query in Listing 6 on our `edgar:SecCubeGrossProfitMargin` cube for the cost of goods sold (`edgar:CostOfGoodsSold`) for each member of the dimension `edgar:issuer`[4] and each date until when each disclosure is valid (`edgar:dtend`), aggregating over the valid start dates and over two specific segments (`edgar:segment`).

Listing 6: Example MDX query for cost of goods sold of specific segments of all companies from single cube.

```
1  SELECT
2  Members(edgar:issuer) ON COLUMNS,
3  Members(edgar:dtend) ON ROWS
4  FROM [edgar:SecCubeGrossProfitMargin]
5  WHERE {(edgar:CostOfGoodsSold,
       edgar:segmentAHealthCareInsuranceCompany),
       (edgar:CostOfGoodsSold,
       edgar:segmentAResidentialRealEstateDeveloper)}
```

[3] For instance, we can search the nested set of OLAP operations for any slice of the dimension.

[4] The MDX-built-in `Members` function returns all members of a dimension or level; since in our example only single-level hierarchies are used, it is not necessary to specify the level such as `edgar:issuerRootLevel` and `edgar:dtendRootLevel`.

A purely syntactic translation from MDX to SPARQL is not possible since additional metadata about multidimensional elements is needed to interpret the query. For instance, the aggregation function of the measure is not described in the query, MDX does not distinguish between dimensions and measures, and built-in functions such as `Members` need specific treatment depending on whether the input is a dimension, hierarchy, or level.

In the following we describe the process from parsing MDX to a nested set of OLAP operations, over translating the set of OLAP operations to a intermediary representation (subcube query), to translating a subcube query to a SPARQL query.

The MDX string can be parsed and translated to the metadata (queried data cube, positions to display on rows and columns, positions to filter facts) of a pivot table.

The metadata in turn can be translated to a nested set of OLAP operations from the OLAP algebra (Section 3.2.1) as follows:

Every data cube defined by the dataset URI in the FROM clause is queried; for the chosen measures or if no measure is chosen the first measure, `Projection` removes not selected measures; for every possible member combination on axes, `Dice` removes filtered dimension members; `Slice` removes every dimension not mentioned in either column or row axis (i.e., aggregates over with aggregation function of measures); and for any higher level selected on columns or rows axes, `Roll-Up` aggregates dimensions to higher levels.

The nested set of OLAP operations from our example is described in Listing 7. In all our queries, we use prefixes to make URIs more readable. We use a function-like syntax to describe that cubes are modified along OLAP operations parametrised with elements from the MDM to eventually result in a cube to be displayed to the user in a pivot table.

Listing 7: Example nested set of OLAP operations for single-cube query for cost of goods sold of specific segments of all companies.

```
1  Slice(
2      Dice(
3          Projection(
4                  edgar:SecCubeGrossProfitMargin,
5                  {edgar:CostOfGoodsSold}),
6              edgar:segment,
7              {edgar:segmentAHealthCareInsuranceCompany,
8              edgar:segmentAResidentialRealEstateDeveloper}),
9          {edgar:dtstart, edgar:segment})
```

The logical OLAP query plan is then transformed to a subcube query with dimensions Issuer, Dtstart, Dtend, Segment, e.g., by a Visitor pattern:

```
(?, ALL, ?, {edgar:segmentAHealthCareInsuranceCompany,
edgar:segmentAResidentialRealEstateDeveloper},
    {edgar:CostOfGoodsSold})
```

Next, we describe a simplified *OLAP-to-SPARQL Algorithm* of how to evaluate such an OLAP query using a SPARQL query on QB. In this approach, since QB provides different ways to specify OLAP hierarchies, and since dimensions in our scenario are flat, we simplify the problem of translating a subcube query to SPARQL and assume a queried data cube with only one hierarchy and level per dimension; a Roll-Up has a similar effect to a Slice operation and can be added to our method by a group-by not on the members of the lowest level of a dimension but on members of a higher level, specified via their *ROLLUPMEMBER* and *ROLLUPLEVEL* relations. Afterwards, the complete *OLAP-to-SPARQL Algorithm* considering multi-level hierarchies will be easier to understand.

An OLAP query $Q = (q_1, \ldots, q_N, m)$, $m \in 2^M$ over $c \in DataCube$ represented using abbreviated syntax of a subcube query tuple as per Definition 11 can be translated into a SPARQL query using the following steps:

1. We initialise the SPARQL query using the URI of the data cube. We query for all instance data from the data cube, i.e., observations linking to datasets which link to the data structure definition.

2. For each *inquired* dimension, we query for the observations showing property-value pairs. We use OPTIONAL graph patterns for dimensions for cases of cube sparsity, and require all members – including literal members using `skos:Concept` and `skos:notation`[5] – to be explicitly added to a level.

   To display inquired dimensions in the result and correctly aggregating the measures, we *GROUP BY* each *inquired* dimension and leave out all *aggregated* (or *fixed*) dimension variables.

3. For each *fixed* dimension, we filter for those observations that exhibit for each dimension one of the listed members.

4. For each *selected measure*, we incorporate it in the SPARQL query by selecting additional variables for each measure and by aggregating the variables using the aggregation function of that measure. We use OPTIONAL graph patterns for cases of cube sparsity.

We transform our example from above to the SPARQL query in Listing 8. *UDF* represents the aggregation function necessary in our COST query; if the input set of literals only contains one literal, *UDF* returns the literal itself, otherwise *UDF* returns a literal

[5] See https://groups.google.com/d/msg/publishing-statistical-data/mQOynmcMXJQ/95w3JQwfUK4J for a discussion of this approach, last accessed 2014-11-12.

describing the number of values. *UDF* is an algebraic aggregation function in that it can be computed by distributive functions *COUNT* and *SUM* [GCB+97].

Listing 8: Example SPARQL query for single-cube query for cost of goods sold of specific segments of all companies.

```
select ?dimMem0 ?dimMem1 UDF(?measureValues0) where {
?obs qb:dataSet ?ds.
?ds qb:structure edgar:SecCubeGrossProfitMargin.

?dimMem0 skos:member edgar:issuerRootLevel.
OPTIONAL { ?obs edgar:issuer ?dimMem0. }

?values1 skos:member edgar:dtendRootLevel.
?values1 skos:notation ?dimMem1.
OPTIONAL { ?obs edgar:dtend ?dimMem1. }

?obs edgar:segment ?slicerMem0.
Filter(?slicerMem0 = edgar:segmentAHealthCareInsuranceCompany
OR ?slicerMem0 = edgar:segmentAResidentialRealEstateDeveloper)

OPTIONAL { ?obs egar:CostOfGoodsSold ?measureValue0. }
} group by ?dimMem0 ?dimMem1
```

Given an MDM with *Member*, *Level*, *Hierarchy*, *Dimension*, *Measure*, *DataCube-Schema*, *Fact*, and *DataCube* as multidimensional elements, we define OLAP Engine $\subseteq$ OLAP Query $\times$ Target Query, with OLAP Query as per Definition 11, Target Query a query in a target query language such as SQL and SPARQL.

In the following pseudocode algorithm we now present the complete *OLAP-to-SPARQL Algorithm* [KH13] for an *OLAP engine* that transforms a subcube query as per Definition 11 into a SPARQL query. The algorithm separately creates the WHERE, SELECT and GROUP BY clause. In the pseudocode we allow direct access to parts of a multidimensional element, e.g., *dimension.Hierarchies* returns the hierarchies of a dimension. Also, we disregard translating multidimensional elements to URI representations and variables, more efficient filters, complex measures, and ordering:

Listing 9: Pseudocode of OLAP-to-SPARQL Algorithm.

```
Algorithm 1: OLAP-to-SPARQL
Input: Subcube query (cube, SlicesRollups, Dices, Projections)
Output: SPARQL query string
begin
  // Evaluating cube
  whereClause = "?obs qb:dataSet " + cube.nameDC.

  // Evaluating SlicesRollups
  for level ∈ SlicesRollups AND level != ALL {
    levelHeight = level.hierarchy.maxdepth - level.depth
    dimension = level.hierarchy.dimension
    dimVar = makeUriToParameter(dimension)
    hashMap.put(dimension, levelHeight)
    for i = 0 to levelHeight - 1 {
      rollUpsPath += dimVar + i + ". " + dimVar + i + " skos:narrower "
```

```
16      }
17      whereClause += "?obs " + dimension.nameD + rollUpsPath + dimVar +
            levelHeight + ". "
18      whereClause += dimVar + levelHeight + " skos:member " + level.nameL
19      selectClause, groupByClause += " "+ dimVar + levelHeight
20    }
21    // Evaluating Dices
22    for member ∈ Dices.positions.get(0) {
23      dicesLevelHeight = member.level.hierarchy.maxdepth - member.level.depth
24      slicesRollupsLevelHeight = hashMap.get(dimension)
25      // If necessary, add new patterns
26      if (dicesLevelHeight > slicesRollupsLevelHeight) {
27        dimension = member.level.hierarchy.dimension
28        dimVar = makeUriToParameter(dimension)
29        for i = slicesRollupsLevelHeight to dicesLevelHeight - 1 {
30          dicesPath += dimVar + i + ". " + dimVar + i + " skos:narrower "
31        }
32        whereClause += "?obs " + dimension.nameD + dicesPath + dimVar +
              dicesLevelHeight + ". "
33      }
34    }
35    whereClause += " Filter("
36    for position ∈ Dices {
37      for member ∈ position {
38        dimVar = makeUriToParameter(member.level.hierarchy.dimension)
39        memberFilterAnd += "AND " + dimVar + dicesLevelHeight + " = " +
              member.nameM
40      }
41      memberFilterOr += "OR " + memberFilterAnd
42    }
43    whereClause += memberFilterOr + ") "
44
45    // Evaluating Projections
46    for measure ∈ Projections {
47      if (measure.iscompound) {
48        selectClause += measure.calc.aggregation + "(" + measure.calc.algebraic
              + ") "
49      } else {
50            measVar = makeUriToParameter(measure)
51            selectClause += measure.calc.aggregation + "(" + measVar + ") "
52        whereClause += " ?obs " + measure.uri + " " + measVar + " ."
53          }
54    }
55    return selectClause + whereClause + groupByClause
```

We query for all observations of the cube (line 6). Then, for each level of non-sliced dimensions, we create a property path starting with the variable `?obs` for the fact and ending with a dimension variable at the respective level (line 9 to 19). The level height determines the length of the property path and is computed from the highest depth in a hierarchy (maxdepth) minus the depth of the level (10). Each level height we store in a map in order to later check whether graph patterns need to be added for dices (13). Then, we add the variables to the select and group by clause (19). Now, we add graph patterns for dices (22 to 43). *Dices* is a set of positions with each position describing a possible combination of members for each diced dimension (22). Diced dimensions and levels are fixed for each position; therefore, we only use the first position for adding graph patterns (22). We assume furthermore that measures are only contained in *Projections* but not in *SlicesRollups* and *Dices*. We only need to add graph patterns

if the height of the diced level is larger than the level mentioned for the same dimension in *SlicesRollups* (26). Then, from the positions in *Dices*, we filter for one (OR, 36) of all possible combinations (AND, 37) of members for each diced dimension. Finally, for each measure in *Projections*, if it is compound, we use the respective aggregation and algebraic function of the measure to compute the compound (48), otherwise, we add a variable with the aggregation function to the SELECT clause and graph patterns to the WHERE clause (51,52).

In the next section, we extend OLAP-to-SPARQL for the drill-across operation and multi-cube OLAP queries.

6.2.2 Drill-Across with Shared Dimension Mappings in Linked Data

Integration of datasets can be done by describing multi-cubes [SDN00], i.e., cubes with shared dimensions and members from several single cubes. Query engines then allow to query over a multi-cube, however, there is little known about how query engines organise the integration internally. Also, there is no standard of how to define multi-cubes.

In this section, we show how to create and query *multi-cubes* in Statistical Linked Data with the *drill-across* operation.

We describe the data of a cube `ds ∈ DataCube` as a relation `ds(D1, D2, ..., Dn, M)` with `dimension(ds)` the set of dimensions used by a cube and `M` the unspecific measure `sdmx-measure:obsValue`. For each dimension and measure, the relation contains an attribute. As entities, the relation contains all possible dimension-member combinations on a specific level of detail, possibly with M an empty value such as “null” or “”. We use Functional Datalog [Gen10, See Chapter Basic Concepts] for describing rules about relations.

We extend the approach of direct querying of single data cubes (with the projection, dice, slice, and roll-up operations) with the integration of several data cubes through drill-across.

Definition 12 defines drill-across, the basic operation for integrating cubes. *Drill-across* brings together measures from several cubes, has as input two data cubes, computes the join of facts on their dimensions and members, and returns a new data cube with the union of all dimensions and the union of measures [SDN00, ASS03].

Definition 12 (Drill-Across)**.** *Given two data cubes* `ds1(D11, D12, ..., D1n, M1)` *and* `ds2(D21, D22, ..., D2n, M2)`*, we define* $Drill-Across : DataCube \times DataCube \rightarrow DataCube$ *[GGV12] with* `Drill-Across(ds1, ds2)=ds3` *as follows: If* `dimension(ds1) != dimension(ds2)` *then* `ds3(D31, ...,`

D3n, M3), with dimension(ds3) = dimension(ds1)∪dimension(ds2) *empty, i.e., its relation contains no tuples (*empty cube*); else then* D1i = D2i, 1 <= i <= n *and the following rule holds:* ds3(D1, ..., Dn, M) :- ds1(D1, ..., Dn, M1), ds2(D1, ..., Dn, M2), M = f(M1, M2), *with* f(M1, M2) *defined as follows: If* (M1 != null AND M2 == null) *then* M1; *else if* (M1 == null AND M2 != null) *then* M2; *else if* (M1 == M2) *then* M1; *else "Integrity Constraint Violation".*

As stated in other work [SDN00, KR02], drill-across requires as input two data cubes with all dimensions shared and for the resulting cube computes an OUTER JOIN of facts on the dimensions, i.e., measures can have empty values such as "null" or "". Such strict definition is cleaner than other relaxed ones [ASS03, GGV12] where facts from one cube may be joined with several facts from another cube; in our case this aggregation can be achieved by preceding Slice and Roll-Up operations.

Existing work assumes different measures of input cubes so that drill-across brings together measures for comparisons. However, in Statistical Linked Data, datasets often exhibit same measures such as sdmx-measure:obsValue. Therefore, we consider the more general case where the same measure may be used by the input data cubes; in case two facts from the two cubes have identical dimension-member combinations and different values for the measures, the resulting cube violates the constraint to not have different measure values for the same dimension-member combination (IC-12 in QB specification; can also be denoted by ''integrity constraint violation'' :- ds(D1, ..., Dn, M1), ds(D1, ..., Dn, M2), M1 != M2). Use-case-specific conflict resolution is then possible.

In an MDX query, Drill-Across can be issued by a special-character-separated[6] list of dataset URIs mentioned in the FROM clause.

For instance, Listing 10 shows an example MDX query issuing a Drill-Across, for the Unemplyoment Fear and GDP Growth (UNEMPLOY) query (adapted from our multi-cube MDX query in Listing 3 in the previous chapter).

Listing 10: MDX query for employment fear metric and GDP growth rate for Germany over time in UNEMPLOY query, including drill-across over two cubes.

```
1 SELECT
2 {[sdmx-measure:obsValue eurostat:id/tec00115#ds qb4o:avg],
      [sdmx-measure:obsValue allbus:ZA4570v590.rdf#ds qb4o:avg]} ON
      COLUMNS,
3 CrossJoin(Members([dcterms:date]), Members([estatwrap:geo])) ON ROWS
4 FROM [eurostat:id/tec00115#ds;allbus:ZA4570v590.rdf#ds]
5 WHERE {[eurostat:dic/geo#DE]}
```

[6] In our implementation, we used a comma, for better readability in this section, we use "+" or ";".

Here, a multi-cube is created via Drill-Across over the data cubes Unemployment Fear and GDP Growth.

Similar as for the MDX-query without issuing Drill-Across in the previous section (Listing 6), the MDX string can be parsed and translated to metadata (columns, rows, filters) of a pivot table. The metadata in turn can be translated to a nested set of OLAP operations.

Whereas in the simple case all operations are executed over a single cube, we interpret a Drill-Across query in MDX as follows: Every dataset URI in the FROM clause describes a queried data cube; all single-cube operations in the MDX query are separately issued over the data cubes. The resulting data cubes are integrated with a nested set of Drill-Across operations.

Similar as for the simple case, we use a function-like syntax to describe that cubes are modified along OLAP operations parametrised with elements from the MDM to eventually result in a cube to be displayed to the user in a pivot table.

In our example, the unit and variable dimensions are sliced and the average of measures over all years for Germany requested. We can use a nested set of analytical operations as in Listing 11 to describe the query in terms of the two available cubes.

Listing 11: Nested set of analytical operations for employment fear metric and GDP growth rate for Germany over time in UNEMPLOY query, including drill-across over two cubes.

```
1  Drill-Across(
2          Slice(
3                  Dice(
4                          Projection(
5                                  eurostat:id/tec00115#ds,
6                  {sdmx-measure:obsValue qb4o:avg}),
7              estatwrap:geo,
8              {eurostat:dic/geo#DE}),
9              {estatwrap:unit, gesis:variable}),
10         Slice(
11                 Dice(
12                         Projection(
13                                 allbus:ZA4570v590.rdf#ds,
14                 {sdmx-measure:obsValue qb4o:avg}),
15                 estatwrap:geo,
16                 {eurostat:dic/geo#DE}),
17             {estatwrap:unit, gesis:variable})
18 )
```

This query only returns results if Unemployment Fear and GDP Growth data cubes exhibit a dimension `estatwrap:geo` and a member `eurostat:dic/geo#DE`. However, GDP Growth and Unemployment Fear cubes use different geo dimensions (`estatwrap:geo` and `gesis:geo`) and different members representing Germany (`eurostat:dic/geo#DE` and `allbus:geo.rdf#00`) that only implicitly may

be mapped, e.g., via `owl:sameAs`. In the following, we describe how to evaluate Drill-Across using SPARQL, also in case of implicitly shared dimensions and measures.

To execute the analytical query given in Listing 11, we need to evaluate the query plan of a nested set of OLAP operations over QB datasets.

As shown in Section 6.2.1, every sub-query-plan of OLAP operations not including the Drill-Across operation we can translate to a subcube query. Using the OLAP-to-SPARQL algorithm in Listing 9, we can translate the subcube query to a SPARQL query.

Similarly, if several of such sub-query-plans are brought together using a nested set of Drill-Across operations, we can evaluate the Drill-Across operation using SPARQL.

Given two subcube queries $Q_1 = (c_1, SlicesRollups_1, Dices_1, Projections_1)$ and $Q_2 = (c_2, SlicesRollups_2, Dices_2, Projections_2)$, we evaluate Drill-Across over the two resulting data cubes as follows:

First, we check whether the resulting data cubes from the subcube queries share all dimensions: for every level $l_1 \in SlicesRollups_1, l_1 \neq ALL$ there is a level in $l_2 \in SlicesRollups_2, l_2 \neq ALL$ with $l_1 = l_2$. Be D_{new} the set of dimensions of these common levels and $M_{new} = Projections_1 \cup Projections_2$. If not, we return an *empty* cube without tuples.

If the input cubes share all their dimensions, we can evaluate Drill-Across over the RDF describing the two input cubes using SPARQL. See Listing 12 for an example SPARQL query for our previous OLAP query in Listing 11.

Listing 12: SPARQL query for employment fear metric and GDP growth rate for Germany over time in UNEMPLOY query, including drill-across over two cubes.

```
1  select ?geo0 ?date0 f(avg(?obsValue1), avg(?obsValue2))
2  where {
3  OPTIONAL { ?obs1 qb:dataSet eurostat:id/tec00115#ds;
4      estatwrap:geo ?geo0;
5      dcterms:date ?date0;
6      sdmx-measure:obsValue ?obsValue1 .
7  FILTER (?geo0 = eurostat:dic/geo#DE) }
8  OPTIONAL { ?obs2 qb:dataSet allbus:ZA4570v590.rdf#ds;
9      estatwrap:geo ?geo0;
10     dcterms:date ?date0;
11     sdmx-measure:obsValue ?obsValue2 .
12 FILTER (?geo0 = eurostat:dic/geo#DE)
13 }} group by ?geo0 ?date0
```

For each of the two input cubes, we query for observations linked via `qb:dataSet` to the respective QB dataset URI (line 3 and 8); the observations from both datasets we join on the values of their dimension properties (4,5 and 9,10) and bind the values of

their measures to separate variables (6,11) and combine them with `f(M1, M2)` with `f` resolving possible integrity constraint violations (1)[7]. Various optimisations such as materialisation [SDN00] are possible but not the topic of this chapter. The integration of more than two data cubes is possible by chaining Drill-Across operations.

Drill-Across requires data cubes to share dimensions and members.

For instance, since the datasets for GDP Growth and for Unemployment Fear both use `dcterms:date` and literal values for years such as `2006`, drill-across over the time dimension can directly be done. However, the GDP Growth and Unemployment Fear cubes use different geo dimensions, `estatwrap:geo` and `gesis:geo`, as well as different members representing Germany, `eurostat:dic/geo#DE` and `allbus:geo.rdf#00`.

To allow the implicit definition of shared dimensions and members, we assume that the standard OWL semantics hold. OWL axioms can either be loaded from existing Linked Data or manually added to the system.

After stating `eurostat:dic/geo#DE owl:sameAs allbus:geo.rdf#00` as well as `estatwrap:geo owl:sameAs gesis:geo`, the query from Listing 11 will bring together GDP Growth and Unemployment Fear for Germany.

6.3 Evaluation

In this section, we first demonstrate in a small experiment the correctness and applicability of our OLAP-to-SPARQL algorithm for our COST query; then to evaluate the Drill-Across extension to our algorithm and to investigate possible performance bottlenecks, we repeat our experiments for UNEMPLOY and EU2020 (Chapter 5).

6.3.1 Evaluating the OLAP-to-SPARQL Algorithm

For this experiment, we used a Linked Data crawler (LDSpider) to collect relevant data for multidimensional datasets (Definition 7) about balance sheets of financial companies such as *RAYONIER INC* from *Edgarwrap* and loaded the data into an Open Virtuoso triple store with a SPARQL endpoint. Using SPARQL INSERT queries, we defined the multi-cube `edgar:SecCubeGrossProfitMargin` with financial facts that disclose Cost Of Goods Sold (`edgar:CostOfGoodsSold`) or Sales Revenue Net (`edgar:Sales`). In total, we loaded around 148,426 triples into the SPARQL engine.

[7] In this example, no conflict resolution would be done but the values of the two cubes displayed to the user, e.g., using the concat function.

The data cube `edgar:SecCubeGrossProfitMargin` contained 17,448 disclosures that either disclose Cost Of Goods Sold or Sales Revenue Net. The values of the measures fully depend on one of 625 different issuers (dimension *edgar:issuer*), the date a disclosure started (27 members of dimension *edgar:dtstart*) and ended (20 members of *edgar:dtend*) to be valid, and additional information (21,227 members of *edgar:segment*). The two measures (*edgar:CostOfGoodsSold* and *edgar:Sales*) have the unit USD and an aggregation function that returns the number of disclosures, or – if only one – the actual number. If fully materialised according to Definition 10, the cube contains $626 \cdot 28 \cdot 21 \cdot 21,228 = 7,813,772,064$ facts. To compute all of its facts, $2^4 = 16$ SPARQL subqueries would be needed.

The OLAP-to-SPARQL algorithm for this experiment we implemented in a Java program[8]. The MDM-QB Mapping, we implemented with SPARQL queries for every metadata query method over the pre-filled Open Virtuoso triple store.

We manually created the MDX query shown in Listing 13 for our XBRL scenario and sent it to our program, the resulting facts are visualised using a pivot table. Multidimensional elements are described in the MDX query using their unique URIs[9].

Listing 13: MDX query for cost of goods sold of specific companies over periods of time in single-cube COST query.

```
1 SELECT
2 {edgar:cik1417907idConcept , edgar:cik106535idConcept }} ON COLUMNS,
3 CrossJoin(Members(edgar:dtstart), Members(edgar:dtend)) ON ROWS
4 FROM [edgar:SecCubeGrossProfitMargin]
5 WHERE {edgar:CostOfGoodsSold}
```

The MDX-built-in `Members` functions over each dimension select in our one-level hierarchies the members on the lowest level of the dimensions[10]. The MDX-built-in `CrossJoin` function returns the cross product of two sets, in our case all possible combinations of start and end dates. The MDX query can be represented as a nested set of OLAP operations as illustrated in Listing 14.

This nested set of OLAP operations can be represented as an OLAP subcube query (Definition 11) with dimensions Issuer, Dtstart, Dtend, Segment: `(?, ?, ?, ALL, {CostOfGoodsSold})`.

Listing 14: Nested set of OLAP operations for cost of goods sold of specific companies over periods of time in single-cube COST query.

[8] See `executeOlapQuery(Cube cube, List<Level> slicesrollups, List<Position> dices, List<Measure> projections)` at https://github.com/bkaempgen/olap4ld/blob/master/OLAP4LD-trunk/src/org/olap4j/driver/olap4ld/linkeddata/OpenVirtuosoEngine.java, accessed on 2015-05-02.

[9] URIs need to be translated to an MDX-compliant format that does not use reserved MDX-specific characters, which is why we use a prefixed notation of URIs.

[10] Therefore, we do not need to ask for the members on the root levels `edgar:dtstartRootLevel` and `edgar:dtendRootLevel`.

```
Slice(
Projection(
edgar:SecCubeGrossProfitMargin,
{edgar:CostOfGoodsSold}),
{edgar:segment})
```

Based on the OLAP-to-SPARQL algorithm, the subcube query can be translated to the SPARQL query in Listing 15.

Listing 15: SPARQL query for cost of goods sold of specific companies over periods of time in single-cube COST query.

```
select ?dimMem0 ?dimMem1 ?dimMem2
    count(xsd:decimal(?measureValue0))
    sum(xsd:decimal(?measureValue0))
where {
?obs qb:dataSet ?ds.
?ds qb:structure edgar:SecCubeGrossProfitMargin.

?dimMem0 skos:member edgar:issuerRootLevel.
OPTIONAL {?obs edgar:issuer ?dimMem0. }

?values1 skos:member edgar:dtstartRootLevel.
?values1 skos:notation ?dimMem1.
OPTIONAL {?obs edgar:dtstart ?dimMem1. }

?values2 skos:member edgar:dtendRootLevel.
?values2 skos:notation ?dimMem2.
OPTIONAL {?obs edgar:dtend ?dimMem2. }

OPTIONAL {?obs edgar:CostOfGoodsSold ?measureValue0. }
} group by ?dimMem0 ?dimMem1 ?dimMem2
```

`UDF`, our default aggregation function is algebraic, therefore, we had to compute the `SUM` and `COUNT` for the measure which would then be used by the compound UDF measure.

In this experiment, we used the simplified OLAP-to-SPARQL Algorithm designed for QB datasets with single-level hierarchies, before the standardisation of QB, and before the definition of *QB Integrity Constraints* for well-formed cubes.

According to QB integrity constraints IC-11 (“all dimensions required”) and IC-14 (“all measures present”) all dimensions and measures in an observation are required and the OPTIONAL clauses (introduced for cases of cube sparsity) can be removed.

Also, a well-formed cube does not need to follow our proposal to explicitly represent levels using `xkos:ClassificationLevel`. Since we assume a lean dataset with all observations with members on the lowest level of each dimension, summarisability for the OLAP-to-SPARQL algorithm is ensured, and the explicit selection of specific levels can be removed in cases of single-level hierarchies.

The complete OLAP-to-SPARQL algorithm would thus result in a SPARQL query with fewer graph patterns and can be expected to run faster than this SPARQL query used in our experiment.

The result from the SPARQL query is the requested subset of all possible facts from a data cube. The pivot table determines what dimensions to display on its columns and rows.

We run the SPARQL query after a reboot of the triple store. The query took 18sec and returned 58 tuples to be filled into the requested pivot table. The number of $7,813,772,064$ potential tuples in the cube does not have a strong influence on the query since the cube is very sparse, for instance, the triple store contains observations only for a fraction of segment members.

6.3.2 Evaluating Drill-Across

In this section, to evaluate the Drill-Across extension to our OLAP-to-SPARQL algorithm and to investigate possible performance bottlenecks, we repeat our experiments from Chapter 5.

We implemented all operations, including the Drill-Across operation, in a Java program[11]. The program uses a directed crawling strategy to load and validate all data cubes into a Sesame Repository (v2.7.10) as embedded triple store. Due to lack of space, in the further descriptions we assume all available data cubes loaded[12].

Drill-Across is implemented as a nested loop join directly over the results of the OLAP-to-SPARQL Algorithm. We evaluate OWL semantics with the equivalence duplication strategy and repeatedly execute SPARQL INSERT queries implementing entailment rules of equality[13] to materialise implicit triples from equivalence statements.

Setup: Just as for the experiments in Chapter 5, we created MDX queries for the scenario UNEMPLOY as well as MDX queries over four and eight EU2020 Indicator datasets (EU2020a and EU2020b) that surely overlap, e.g., the energy dependence, productivity, and intensity. Also, we manually created a file with `owl:sameAs` triples[14] to denote the `:geo` dimensions as shared and manually added the URI as a default data source to our directed crawling strategy. Since both dimensions also exhibit different hierarchies and levels, we manually added an `owl:sameAs` link between

[11] See iterators at https://github.com/bkaempgen/olap4ld/tree/master/OLAP4LD-trunk/src/org/olap4j/driver/olap4ld/linkeddata, last accessed 2015-05-02.

[12] Additional information can be found in the respective section of the evaluation website of the published paper [KSH14]: http://www.linked-data-cubes.org/index.php/Global_Cube_Evaluation_EKAW14, last accessed 2014-11-11.

[13] http://semanticweb.org/OWLLD/#Rules, last accessed 2014-11-18.

[14] http://people.aifb.kit.edu/bka/Public/cube_additionalRDF.rdf, last accessed on 2014-06-21.

`ramon:NUTSRegion` and `allbus:geo.rdf#list`. The links between Germany `eurostat:dic/geo#DE` and `allbus:geo.rdf#00` from Eurostat and Gesis are already given when resolving `allbus:geo.rdf#00`[15].

For *EU2020a/b*, we selected four and eight EU 2020 Indicator datasets that surely overlap, e.g., the energy dependence, productivity, and intensity.

The two queries from our scenario we executed five times on an Ubuntu 12.04 workstation with Intel(R) Core(TM) i5 CPU, M520, 2.40GHz, 8 GB RAM, 64-bit on a JVM (v6) with 512M initial and 1524M maximum memory allocation.

Results: Table 6.1 gives an overview of experiment results. We compare query results from previous experiments executing our MDM-QB Mapping for Drill-Across with the OLAP engine Mondrian over MySQL (UNEMPLOY_1, EU2020a_1, EU2020b_1, see Chapter 5) and from executing our OLAP-to-SPARQL algorithm including Drill-Across (UNEMPLOY_2, EU2020a_2, EU2020b_2). The experiments are comparable since we used the same machine.

We successfully integrated the GDP Growth from Eurostat[16] and the Unemployment Fear from ALLBUS and found overlaps for Germany in 2004 and 2006. Mappings between implicitly shared dimensions and members were considered. Also, for the EU2020 a/b queries, we successfully integrated four and eight datasets.

Table 6.1: For every experiment, number of integrated datasets #DS, triples #T, observations #O, look-ups #LU, and average elapsed query times in sec for loading and validating datasets (L&V), executing (MD) a certain number of metadata queries (#MD), generating the logical query plan (LQP), generating the physical query plan (PQP), executing the physical query plan (EQP), and total elapsed query time (T).

Experiment	#DS	#T	#O	#LU	L&V	MD	#MD	LQP	PQP	EQP	T
UNEMPLOY_1	2	20,268	350	22	273	-	-	-	-	0.073	273
UNEMPLOY_2	2	3,897	362	12	11	5	41	3	3	0.036	22
EU2020a_1	4	24,636	1,247	26	654	-	-	-	-	0.161	654
EU2020a_2	4	19,714	2,212	12	18	15	67	3	3	0.094	39
EU2020b_1	8	35,482	2,682	34	1,638	-	-	-	-	0.473	1,638
EU2020b_2	8	38,069	3,992	20	47	40	103	6	10	0.151	103

From the total elapsed query times (T), we see that our OLAP-to-SPARQL approach (UNEMPLOY_2 with 22s, EU2020a_2 with 39s, and EU2020b_2 with 103s) is 10 to 17 times more efficient than the MDM-QB approach with Mondrian OLAP engine (UNEMPLOY_2 with 273s, EU2020a_2 with 654s, and EU2020b_2 with 1,638s). Also,

[15] http://lod.gesis.org/lodpilot/ALLBUS/geo.rdf, last accessed on 2014-06-30.

[16] Apparently, dataset http://estatwrap.ontologycentral.com/id/tsieb020#ds in Eurostat was replaced by dataset tec00115 after conducting these experiments.

we see that for both approaches an increase from 4 to 8 datasets leads to 2.5 times larger total elapsed query times.

Total elapsed query times mainly is improved since loading and validating datasets takes much less time than with our previous implementation (Chapter 5) which results from three aspects: 1) from a switch from qcrumb.com to Sesame as a SPARQL engine. In comparison with downloading data to Sesame, qcrumb.com took more time for downloading required data before answering a SPARQL query; 2) from downloading fewer triples by using the directed crawling strategy with fewer hard-coded URIs to download; and 3) from the time included for populating the data warehouse. In both implementations, L&V includes the time for reasoning.

Disregarding the benefit of a more efficient loading process, we are interested in elapsed query times for query processing. Once the data is loaded in the data warehouse, query processing with the ROLAP engine in our previous implementation is very fast. For our new approach with a SPARQL engine, most time is spent in executing several queries for multidimensional elements (MD) and in generating the logical and physical query plans (LQP+PQP). Our experiments with 2, 4, and 8 datasets indicate that MD, LQP, PQP and EQP increase linearly. LQP includes the time for interpreting the query language (MDX) and building a nested set of OLAP operations. PQP includes the time to run the OLAP-to-SPARQL Algorithm. Executing the SPARQL queries and Drill-Across operations (EQP) only takes a fraction and is similar to processing time in the OLAP engine (Chapter 5).

6.4 Discussions and Lessons Learned

In our small experiment, we showed the applicability of our mapping between OLAP and SPARQL queries for single-level hierarchies and without considering Drill-Across.

Our approach focuses on OLAP queries that can be composed by common OLAP operations and can be represented as a subcube query. Our mapping allows translating of a subcube query to a SPARQL query to be run on the RDF without storage of intermediate results, so that updates to the RDF are propagated directly to OLAP clients.

We correctly aggregate data on one specific granularity, defined by the mentioned inquired and fixed dimensions. Dimensions that are not mentioned will be automatically handled as having an *ALL* value [GCB+97], representing all possible values of the dimension. The aggregation results in correct calculations, since we assume a lean dataset that only contains facts with members on the lowest level of each dimension.

The SPARQL query created by our approach shows sufficiently fast in our small experiment (and will be even faster with a reduced set of graph patterns in SPARQL

queries created by the most current version of the algorithm) but may not scale for larger datasets for the following reasons:

- data from the data cube is queried on demand, and no materialisation is done. Every OLAP query is evaluated using a SPARQL query without caching and reusing of previous results.

- only the `<slicer_specification>` is currently considered for the `Dice` operation in our translation from MDX to subcube queries; dimensions in the `<column_axis_specification>` and `<row_axis_specification>` are translated as inquired dimensions and all possible member combinations calculated, even though only specific combinations might be required, as in the case of the two issuers *RAYONIER INC* and *WEYERHAEUSER CO*.

- OLAP clients and pivot tables require multidimensional data, i.e., data cubes containing facts linking to specific members of dimensions, but our SPARQL query returns relational tuples. Using the unique identifiers of dimensions, members, and measures, query result tuples need to be joined with the multidimensional data points as required by the OLAP client for filling the pivot table. Since no materialization is done, only few extra space is required for a hashmap to fill the pivot table with the SPARQL result set.

Etcheverry and Vaisman [EV12a] present another possible approach; to use SPARQL CONSTRUCT queries to first materialise data cubes resulting from OLAP operations as RDF. Populating the pivot table could then be done by simple SPARQL SELECT queries on this resulting multidimensional view. However, the authors have not evaluated the applicability and performance of this approach to answer OLAP queries.

We believe[17] that there are more possibilities (e.g., based on heuristics, probabilistic metrics or machine learning algorithms) to find meaningful hierarchies in statistical Linked Data. In our work, we assume hierarchies to be strict and symmetric in order to ensure summarisability.

Repeating the experiments from Chapter 5 for evaluating the MDM-QB Mapping, we successfully evaluated the OLAP-to-SPARQL extension for Drill-Across. Our implementation was slightly different from the evaluation of the MDM-QB Mapping (Chapter 5):

- we used a directed crawling for all relevant data that lead to faster loading;

- we executed integrity constraints to check for the right modelling of QB datasets;

[17] See Master thesis by Siegele [Sie12], co-supervised by the author, available at http://www.aifb.kit.edu/images/4/4b/Masterarbeit_Dominik_Siegele.pdf, last accessed 2014-12-17.

- and we used the equivalence duplication strategy instead of entity consolidation since it can be implemented directly with a SPARQL engine without preprocessing of data;

We showed how to generate and query multi-cubes using the drill-across operation over Statistical Linked Data. In Chapter 8 we will use the drill-across operation to generalise the notion of multi-cubes to the Global Cube for integrating available datasets on the Web.

6.5 Related Work

As related work we distinguish approaches to OLAP query processing, in particular Drill-Across query processing, and approaches of more expressive queries over enriched data cubes.

6.5.1 Analytical Query Processing

Providing uniform access to distributed data sources over the Web can be provided either in a federated or a Data Warehouse architecture [GMP+12, PBAP08]. Though potentially faster since query processing is distributed on several machines and less data is transferred over the network, federated query processing requires considerably more logic from data sources than a Data Warehouse architecture. Although Linked Data publishers sometimes also provide SPARQL endpoints and SPARQL 1.1 now supports querying simultaneously over several endpoints, our experiences show that publishing resolvable URIs providing RDF is simpler, more reliable, and more widely-used. Therefore, in this work, we assume data from relevant URIs loaded into a Data Warehouse for efficient integration and querying.

Query processing often is done by the conventional pull-based iterator model with pipelining of temporary tables [Gra93] on top of which query optimisers are implemented [KBZ86].

OLAP query processing in general has long been a topic of research [VS99]. Similar as for the modelling of data on different abstraction levels, we can distinguish three query processing levels [dACCG+13]: The Conceptual Level of OLAP algebras over data cubes that is independent from a logical representation; the Logical Level of queries over a specific logical representation of data cubes, e.g., SQL over Star Schemas in relational databases; and the Physical Level that is concerned with efficient execution of low-level executions such as index lookup or sorting over the data stored given a specific hardware and software.

On the conceptual level, related work defines OLAP algebras and tries to bridge the semantic gap between the conceptual model and logical representations [RA07b, GGV12, PMT08, CRB+06, AGS97, Vas98, PJD01]. Though sometimes lower-level operations are introduced [AGS97, Vas98], the most common operations from an OLAP algebra are projection, dice, slice and roll-up.

On the logical level, query processing mainly depends on the type of data structure on which to perform the computations and in which to store the results. Data structures can roughly be grouped into ROLAP using relational tables and star (or similar such as snowflake) schemas and MOLAP using multidimensional arrays for directly storing and querying of data cubes [VS99, CD97, GCB+97].

The CUBE operator [GCB+97] computes all possible facts in a data cube that can be navigated via roll-up of a multidimensional dataset. Dimension hierarchies and drill-across are not considered.

The execution of OLAP operations mainly is concerned with the computation of the data cube and with storing parts of the results of that computation to efficiently return the results, to require few disk or memory space, and to remain easy to update if data sources change [MKIK07].

In this chapter, we have mainly discussed the conceptual (OLAP operations) and logical (SPARQL queries) levels of querying Statistical Linked Data. Our approach of evaluating analytical queries using SPARQL over RDF is less concerned about the physical level and relies on the efficiency of SPARQL engines such as triple stores. In the next chapter, we present an approach to optimise analytical query processing with triple stores purely on the logical level.

Other work recognise the reduced initial processing and update costs of using triple stores and other dedicated query engines for query processing [NBP+09] but do not present approaches for executing OLAP queries directly over such Semantic Data Warehouses.

Approaches regarding OLAP query processing using SPARQL over RDF seem to have concentrated so far on multidimensional modelling from ontologies [NBP+09, DP08, NN09, NB12, EV12a].

Etcheverry and Vaisman [EV12a] present an algorithm to translate OLAP operations such as Roll-Up and Slice to SPARQL CONSTRUCT queries. Since results are cubes, one can nest operations. Different from our work, OLAP operations are executed for preprocessing of cubes from the Web that are then exported to a data warehouse for query processing. Thus, their approach is a mixture of our approach of using existing OLAP engines and of using existing SPARQL engines for query processing. However, the authors leave out the important Drill-Across operator.

In our work we use the graph-based RDF data model for querying and storing of multidimensional data reusing QB. Both schema information and actual data is accessed using Linked Data principles and managed using SPARQL on a triple store.

Linked Data Query processing refers to complementary research about the specificities of queries over Linked Data sources. We can distinguish the bottom-up approach where data sources are defined by resolvable URIs in the given SPARQL query [HBF09] from the top-down approach where index structures summarising the content of data sources are available to the query system [HHK+10].

Our work is the first to evaluate drill-across over Statistical Linked Data. According to Kimball [KR02] drill-across is the act of requesting data from two or more fact tables sharing all their dimensions. Kimball recommends to use multipass SQL to query each data cube separately and then to join the query results based on common dimension attributes. We have adopted this definition, as have several authors [RA07b, SDN00, KR02].

Other authors [ASS03, GGV12] allow drill-across over cubes whose dimensions may not be fully shared but for instance have different granularity, e.g., monthly versus yearly numbers, and for which one dimension can be defined as the association of several ones for which a mapping is needed, e.g., latitude/longitude versus point geometry. Riazati et al. [RTZ08] further relax restrictions on drill-across operations with a "loss ratio" to quantify the dissimilarity between dimension attributes and to combine heterogeneous pivot tables. Different from these approaches, we keep a strict Drill-Across definition but also cover the more general case in Statistical Linked Data where the same measures are used by input cubes.

Compound measures such as the unemployment fear metric "Percentage of Nos" for the UNEMPLOY query requires Drill-Across over measures of several data cubes. Composite Subset Measures [CRB+06] are one way to model and evaluate such more complex queries.

Although there may be more efficient querying approaches such as special indexing, materialised views, and caching, to the best of our knowledge, this is the first work on executing OLAP queries over data cubes represented as RDF using SPARQL.

6.5.2 More Expressive Queries over Enriched Data Cubes

Several related work consider enrichments of Data Cubes that could also be applied to our approach to make use of additional information provided by RDF data sources.

Entity-centric object databases [PGSJ09] allow the enrichment of data cubes with external data for extended OLAP queries, however, have so far not been applied to Linked Data. The authors define an extended OLAP query language SumQL++ that allows to query over object databases and multidimensional databases simultaneously. The

SumQL++ query is translated to several queries to the respective object databases and a (MDX or SQL) query to the multidimensional database. The results are then integrated and returned to the user. Different from their work, we directly execute multidimensional queries using SPARQL over RDF.

The work [PGSJ09], also fits well with the idea of Linked Data extended OLAP querying since it argues for typed links between multidimensional databases and object-oriented databases (including equivalence links).

Summarisability is ensured by requiring a complete hierarchy and by not allowing grouping by data from external data sources.

The extension of members of the MDM with attributes from external data is made possible. For instance, the members of the dimension "hospital" are "enriched" – sometimes called "decorated" – with the name of the city and the mayor.

Also Yin and Pedersen [YP04] motivate the problem of integrating data from the Web in OLAP systems. The authors present a federated approach to enrich data cubes with virtual dimensions built from external data, which means that XML data can also be used in filter operations. To issue such queries, the authors define an XML-extended Multidimensional SQL which supports referencing external XML data.

For instance, Members of a level "Nation" in a Data Cube are linked to nations in an XML document providing additional information such as the population which then can be used to filter for certain nations. For that, a user needs to define links between dimension values such as from the level nation to nodes in an XML document.

The authors [YP04] also describe the problem of query optimisation when having logical operators for enrichment. Retrieving XML values will become the bottleneck for processing, i.e., if external XML data is large, then, in comparison, the loading takes much more time than other tasks. In those cases, it will be best to execute some of the operators on the single datasets in order to do it in parallel and to avoid transmitting large amounts of data between operators. Thus, the authors plan to push-down operators to single datasets.

Different from our work, they store and query data cubes in the traditional way using SQL but enrich dimensions with data retrieved in XML from the Web. Instead, we represent and query both data cubes and metadata using RDF and SPARQL.

Diamantini and Potena [DP08] enrich an MDM with a domain ontology represented using OWL as well as a mathematical ontology represented in an XML standards for mathemtical descriptions (MathML). Their goal is to provide analysts with useful background information and to possibly allow novel types of analyses, e.g., drill-down into single compound measures. A formal description of formulas of a measure also may help to derive new formulas with the help of symbolic mathematical reasoners.

In other work, the authors [DP10, DP11] describe products and relationships among products as well as explain the meaning of financial and economic concepts with a domain ontology. The mathematical ontology describes formulas for indicators. The work is about further grounding the semantics of multidimensional datasets and about extending the logical representation of an MDM. The authors use Mondrian for deploying the extended MDM and for allowing users to issue analytical queries directly to an OLAP engine.

Mazón et al. [MTSP07] also motivate the use of external data to enhance Data Cubes; they propose the use of semantic relations such as hypernymy ("is-a-kind-of", generalization, e.g., cake is kind of baked goods) and meronymy ("is-a-part-of", aggregation, e.g., wheel is a part of car) between concepts provided by WordNet to enrich Dimension Hierarchies.

If the multidimensional data is represented and directly queried in RDF using SPARQL, linking of provenance information can be done easily since the W3C-recommended *PROV Ontology*[18] can be reused. Similarly, data transformations executed over numeric data can be represented in RDF, referred by multidimensional datasets, and visualised in a browsable visualisation interface to foster trust in the data [FKaGO+12].

Not only the multidimensional data, but also analyses over data cubes can be represented in RDF and used for recommending promising steps in new analyses to business users [FTC11].

Although we so far only translate the common analytical operations, our approach could be extended with operations allowing filtering over enrichments (decorations) described in RDF. Also, OLAP client interfaces can be extended to display additional information related to analysed data cubes.

6.6 Conclusions

We presented an approach to map OLAP queries to SPARQL queries over lean QB datasets. For that, more concretely, we defined projection, slice, dice, and roll-up operations on single data cubes as well as drill-across over multiple data cubes in RDF reusing QB, interpret OLAP queries as nested sets of OLAP operations, and evaluate such operations using SPARQL. Both metadata and OLAP queries are directly evaluated by a SPARQL engine; therefore, if the RDF is modified or updated, changes are propagated directly to OLAP clients.

We evaluated the OLAP-to-SPARQL algorithm. First, we implemented a simplified version for a small experiment over one cube with single-level dimension hierarchies.

[18] http://www.w3.org/TR/prov-o/, last accessed 2015-01-30.

Second, we repeated the experiments from the OLAP engines chapter with an implementation of the complete OLAP-to-SPARQL algorithm extended with the Drill-Across operation for multi-cube queries.

Experiments indicate that we can interpret any analytical (MDX) query as a nested set of OLAP operations; that the set can be transformed to a subcube query which in turn can be transformed to one single SPARQL query.

Our approach to evaluate OLAP queries using SPARQL engines allows OLAP clients 1) to display background information about multidimensional elements, e.g., formulas of compound measures [DP08] and 2) to issue more expressive queries over data cubes enriched with background information, e.g., to filter for hospitals of cities with a certain mayor [PGSJ09].

Flexibility is increased, since links between implicitly shared dimensions and members can be added based on logics and any background information can be attached using the flexible RDF data model.

Although we so far only translate the common analytical operations, our approach could be extended with operations allowing filtering over enrichments described in RDF. Also, OLAP client interfaces can be extended to display additional information related to analysed data cubes.

Our OLAP-to-SPARQL mapping may not result in the most efficient SPARQL query and requires additional efforts in populating requested pivot tables but correctly calculates required tuples from data cubes without inefficient full materialisation and without the need for explicitly introducing the non-relational *ALL* member and using subqueries.

Current datasets available in Statistical Linked Data do not exhibit multi-level hierarchies. Also, the size of existing datasets is limited. In the next chapter, we investigate the benefit of materialisation for query optimisation over a SPARQL engine and use a benchmark with a realistic dataset of arbitrary size and multi-level hierarchies.

In future work, it would be interesting to automatically find and to allow queries over non-strict and non-symmetric hierarchies in Statistical Linked Data[19]. Also, an operation for adding compound measures to a cube would be a useful extension.

Instead of loading all relevant data into a SPARQL engine (in a data warehousing fashion), one could investigate at least two other options: The first option is to only load the data structure definitions, and then to load the actual numeric facts only if requested by a query; however, currently most observations are directly provided with the URI of the respective dataset so that a costly filter would need to be executed over the triples.

[19] For examples, see Master thesis by Siegele [Sie12], co-supervised by the author, available at http://www.aifb.kit.edu/images/4/4b/Masterarbeit_Dominik_Siegele.pdf, last accessed 2014-12-17.

Other options are federated Linked Data query processing techniques [MVC+12] and traversal-based Linked Data query processing [LT10]; SPARQL queries could still be generated by the OLAP-to-SPARQL Algorithm but the query is directly executed over the sources (a similar approach was followed by using qcrumb.com in the previous chapter); such an approach may be more flexible in adding new data sources (e.g., SPARQL endpoints) and allows access to the most current data without permanently storing possibly unnecessary data; however, a data warehousing approach is typically more reliable and more efficient than a federated query processing approach. For instance, in a data warehouse, we can optimise query processing with materialisation as investigated in the next chapter.

7 Query Optimisation using Materialised RDF Aggregate Views

In this chapter, we investigate the following research question:
Research Question 3. *How can we optimise analytical query processing over multidimensional datasets from the Web using aggregate views?*

Figure 7.1 illustrates the contribution given in this chapter. We extend the approach presented in Chapter 6 on the OLAP-to-SPARQL Algorithm and pre-aggregate and store certain values from a Data Cube as *RDF Aggregate Views* for more efficient query processing in the SPARQL engine in comparison to the case where all aggregation are computed on-demand.

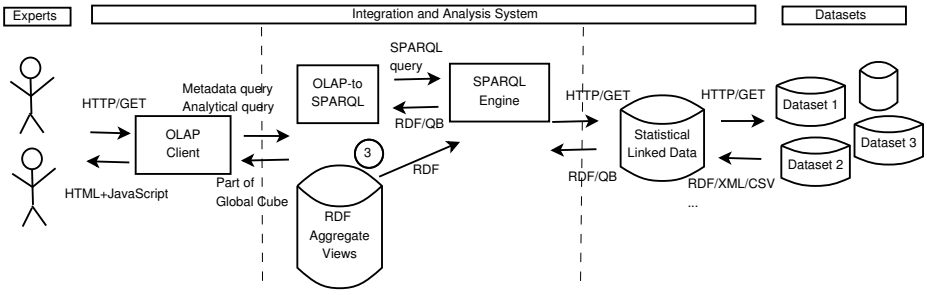


Figure 7.1: Illustration of RDF Aggregate Views; pre-aggregated values are stored in the SPARQL engine; look-up of aggregated values is expected to be faster than on-demand computation.

7.1 Introduction

We can automatically collect Statistical Linked Data reusing the RDF Data Cube Vocabulary (QB) and use existing OLAP engines for efficient query processing. Instead of storing the data in a relational database and using a ROLAP engine we can also use

an RDF store and transform OLAP into SPARQL queries. This allows for more flexible data integration.

Yet, there is little work on evaluating and optimising analytical queries on RDF data [Erl10, Erl12]. We expect that, similar to general-purpose relational databases, a "one size fits all" [SBC+07] SPARQL engine will not scale for analytical queries. In this chapter, we investigate this hypothesis.

Similarly, Nebot et al. [NBP+09] argue for a system that only at query time extracts data from the sources, executes validation and integrity constraint checks, and evaluates the query. The advantage is that only the data that is relevant to the query are extracted, transformed and validated. The authors expect that a triple store can be used for intermediary storage and processing, but also that a dedicated query engine performs better in this dynamic scenario.

In this chapter we give an empirical argument in favour of creating a specialised OLAP engine for analytical queries over Statistical Linked Data. For that, we optimise query processing using existing SPARQL engines with materialised aggregate views. More specifically, contributions of this chapter are centred around four analytical query approaches listed in Table 7.1.

Table 7.1: Overview of analytical query approaches investigated to support an empirical argument in favour of a specialised OLAP engine over Statistical Linked Data.

	No Materialisation	Materialisation
Relational data / SQL	RDBMS / ROLAP	ROLAP-M
Graph data / SPARQL	OLAP4LD-SSB/-QB (Chapter 6)	OLAP4LD-QB-M

The four approaches are investigated as follows:

- We compare the performance of traditional relational approaches (RDBMS / ROLAP) and of using a SPARQL engine and an RDF representation closely resembling the tabular structure (OLAP4LD-SSB). We compare those approaches with our approach using the OLAP-to-SPARQL Algorithm described in the previous chapter (Chapter 6) reusing a standard vocabulary for describing statistics (OLAP4LD-QB). We use a credible benchmark with multi-level dimension hierarchies.
- We measure the performance gain of the common ROLAP query optimisation approach to define aggregate views, i.e., certain parts of the data cube after aggregation, and to materialise those views in tables since they do not fit in memory [MKIK07, GM95] (ROLAP-M). We apply precomputation to our approach, represent materialised views in RDF (OLAP4LD-QB-M) and evaluate their performance gain.

In Section 7.2, we describe realistic data and queries of an OLAP benchmark and present our optimisation approach of using RDF Aggregate Views. In Section 7.3, we compare the approach with RDF Aggregate Views, the pure OLAP-to-SPARQL Algorithm approach (Chapter 6), and a ROLAP system. In Section 7.4, we discuss the results, after which we describe related work in Section 7.5 and conclude in Section 7.6.

7.2 Approach: RDF Aggregate Views

Before applying materialised aggregate views to RDF, we introduce realistic data and queries from the *Star Schema Benchmark* (SSB) [OOC09]. In Section 7.3, we will use the data and queries from the benchmark for a performance evaluation.

7.2.1 Star Schema Benchmark Data and Queries

SSB describes a data cube of lineorders. Any lineorder (fact) has a value (member) for six dimensions: the time of ordering (*dates*), the served *customer*, the product *part*, the *supplier*, the ordered *quantity* and granted *discount*. Depending on the member for each dimension, a lineorder exhibits a value for several measures. Every measure has a calculation expression over facts, composed of an aggregation function with which to aggregate a measure over several facts and possibly an algebraic function with which to compute the measure from other measures (compound measure).

For instance, *sum_profit* is computed with aggregation function *SUM* and algebraic function *lo_revenue* minus *lo_supplycost* over aggregated facts; *sum_revenue* is computed with *SUM* and *lo_extendedprice* multiplied by *lo_discount*.

Dimensions exhibit hierarchies of levels that group members and relate them to higher-level members, e.g., dates can be grouped starting from the lowest *dateLevel* over *yearmonthLevel* to *yearLevel*. Since a week can be spread over two months or years, there is a separate hierarchy where dates can be grouped by *weeknuminyear*, e.g., "199322". Customers and suppliers can be grouped into cities, nations, and regions and parts into brands, categories and manufacturers. Any hierarchy implicitly has a special-type *ALL* member, which groups all members into one special-type *ALL* level.

In the following bullet point list, we describe the SSB data cube on Scale 1 (with 6,000,000 lineorders) as Statistical Linked Data and how the RDF data can be mapped to our MDM using the MDM-QB Mapping (Chapter 5).

Member All 3,094 dates, 30,280 customer, 201,030 part and 2,280 supplier members from each level are represented as URIs. Any member, e.g., `rdfh:category-MFGR-35`, links to members on the next lower level via `skos:narrower`,

e.g., rdfh:brand1MFGR-3527. 51 quantity and 11 discount members we encode as RDF Literal values. Also, we define URIs representing the special-type *ALL* member for each dimension, e.g., customer rdfh:lo_custkeyAllAll. Those *ALL* members are needed later for representing materialised aggregate views.

Table 7.2: Example pivot table showing the revenue (in USD) for product brands from product category MFGR#12 and of suppliers from AMERICA.

Year\Brand	MFGR#121	MFGR#1210	...	MFGR#129
1992	667,692,830	568,030,008	...	614,832,897
...	...	...	...	...
1998	381,464,693	335,711,347	...	319,373,807

Filter: partCategory = "categoryMFGR#12"
AND supplierRegion = "AMERICA"

Level Every level is represented as a URI, e.g., rdfh:lo_orderdateDateLevel, has a xkos:depth within its hierarchy and links to a set of members via skos:member.

Hierarchy Each dimension has one (or two for dates) hierarchies. Every hierarchy is represented as a URI, e.g., rdfh:lo_orderdateCodeList. Levels with a depth link to the hierarchy via skos:inScheme.

Dimension Every dimension such as dates is represented as an object property, e.g., rdfh:lo_orderdate and defines its hierarchy via qb:codeList. The simple dimensions rdfh:lo_quantity, rdfh:lo_discount are represented as datatype properties.

Measures Every measure such as the sum of revenues is represented as a datatype property, e.g., rdfh:lo_revenue. The component specification of a measure defines the calculation expression, e.g.,"sum(rdfh:lo_revenue - rdfh:-lo_supplycost)", via qb4o:hasAggregateFunction, as proposed by Etcheverry and Vaismann [EV12b]. Since there is no recommended way to represent more complex functions, for formulas, we use String Literals using measure URIs as variables.

DataCubeSchema The data cube schema of the SSB data cube is represented as an instance rdfh-inst:dsd of qb:DataStructureDefinition and defines the dimensions and measures of the data cube.

Fact Every possible lineorder can be represented as a qb:Observation. Any observation links for each dimension property to the URI of a member or a Lit-

eral value (quantity, discount), and for each measure property to a Literal value. Whereas base facts with each dimension on the lowest level are given by the SSB dataset, aggregated facts on higher levels of dimensions of the cube need to be computed.

DataCube The SSB data cube is identified by the dataset `rdfh-inst:ds`. The dataset defines the schema `rdfh-inst:dsd` and has attached via *qb:dataSet* all base facts.

In the following, we explain how our OLAP-to-SPARQL Algorithm (Chapter 6) can be applied to SSB.

SSB provides a workload of 13 queries on the data cube. Each query is originally provided in SQL. For instance, *Q2.1* computes per year the revenues (in USD) for product brands from product category MFGR#12 and of suppliers from AMERICA. Results from this query usually are shown in pivot tables such as in Table 7.2.

All queries of SSB can be formalised as subcube queries over single data cubes with multi-level hierarchies as per Definition 11, e.g., *Q2.1* as follows with abbreviated names: (rdfh-inst:ds, {yearLevel, ALL, brand1Level, ALL, ALL, ALL}, {categoryLevel = categoryMFGR-12, s_regionLevel = s_regionAMERICA}, {lo_revenue}). *Q2.1* slices dimensions customer, supplier, discount, quantity, rolls up dates to years and part to product brands, dices for a specific product part category and supplier region and projects the revenues.

Listing 16 shows the relevant parts of the SPARQL query for *Q2.1* created by our *OLAP-to-SPARQL Algorithm* (Chapter 6).

Listing 16: SPARQL query for Q2.1 in SSB benchmark.

```
1  SELECT ?rdfh_lo_orderdate ?rdfh_lo_partkey1 sum(?rdfh_lo_revenue)
       as ?lo_revenue
2  WHERE {
3  ?obs qb:dataSet rdfh-inst:ds; rdfh:lo_orderdate ?rdfh_lo_orderdate0.
4  ?rdfh_lo_orderdate1 skos:narrower ?rdfh_lo_orderdate0.
5  ?rdfh_lo_orderdate2 skos:narrower ?rdfh_lo_orderdate1.
6  ?rdfh_lo_orderdate skos:narrower ?rdfh_lo_orderdate2.
7  rdfh:lo_orderdateYearLevel skos:member ?rdfh_lo_orderdate.
8  ?obs rdfh:lo_partkey ?rdfh_lo_partkey0.
9  ?rdfh_lo_partkey1 skos:narrower ?rdfh_lo_partkey0.
10 ?rdfh_lo_partkey skos:narrower ?rdfh_lo_partkey1.
11 rdfh:lo_partkeyCategoryLevel skos:member ?rdfh_lo_partkey.
12 ?obs rdfh:lo_suppkey ?rdfh_lo_suppkey0.
13 ?rdfh_lo_suppkey1 skos:narrower ?rdfh_lo_suppkey0.
14 ?rdfh_lo_suppkey2 skos:narrower ?rdfh_lo_suppkey1.
15 ?rdfh_lo_suppkey skos:narrower ?rdfh_lo_suppkey2.
16 rdfh:lo_suppkeyRegionLevel skos:member ?rdfh_lo_suppkey.
17 ?obs rdfh:lo_revenue ?rdfh_lo_revenue.
18 FILTER(?rdfh_lo_partkey = rdfh:lo_partkeyCategoryMFGR-12 AND
       ?rdfh_lo_suppkey = rdfh:lo_suppkeyRegionAMERICA ).
```

```
19  } GROUP BY ?rdfh_lo_orderdate ?rdfh_lo_partkey1 ORDER BY
        ?rdfh_lo_orderdate ?rdfh_lo_partkey1
```

Here, `Dices`, {categoryLevel = categoryMFGR-12, s_regionLevel = s_regionAMERICA}, is translated into one position with one member for part category level and one member for supplier region level. The SPARQL query queries for all facts within the data cube (line 3), adds *skos:narrower* paths up to yearLevel, categoryLevel and s_regionLevel (3 to 16), selects lo_revenue as measure (17), filters for members of part category and of supplier region (18) and groups by yearLevel and brand1Level (19). We assume all RDF data stored in a default graph.

More information about the benchmark we provide on a benchmark website [KH12].

In the next section, we apply materialisation to optimise query processing.

7.2.2 RDF Aggregate Views

We apply a common optimisation technique to the *OLAP engine* implementing our OLAP-to-SPARQL Algorithm (Chapter 6): data cube materialisation, i.e., pre-aggregating of certain facts from the entire data cube and storing them for reuse [ABD+99].

For that, we define *aggregate views* over our MDM (in Definition 13) and select, compute and represent views using RDF and SPARQL.

Just as Harinarayan et al. [HRU96], we assume that the cost of answering an OLAP query is proportional to the number of facts that need to be scanned, e.g., for validating a filter or calculating an aggregation. So far, any OLAP query to the SSB data cube needs to scan the 6,000,000 base facts.

Intuitively, we 1) specify views as certain kinds of queries (slice queries) over a data cube; 2) we pre-compute the facts from these slices and store them as RDF; and 3) we use views not by query rewriting [Gen10, see Query Folding Chapter] but by running a query over the pre-computed facts in RDF. The expectation is that the number of facts in those slices are smaller than in the full data cube and that fewer expensive aggregations need to be computed to answer a query.

Definition 13 (Aggregate View)**.** *We define an aggregate view in a data cube* `c` *as a specific subcube query* `Q = (c, SlicesRollups, Dices, Projections)` *with* `SlicesRollups` $\subseteq L_1 \times L_2 \times \ldots \times L_d$, `Dices` *the empty set, and* `Projections` *a set of measures.*

Thus, any fact within an aggregate view gives a value for each of its measures for a certain combination of level members. A view may be sparse and not contain facts for each possible combination of members. The maximum number of facts within a view is given by $\prod memberno(l_i), l_i \in L_i$, with `memberno()` the number of members in a level. The number of views in the data cube is given by $\prod |L_i|$. The facts from

an aggregate view can be generated by executing the OLAP query using an OLAP engine.

The SSB data cube contains $6 * 5 * 5 * 5 * 2 * 2 = 3,000$ views with dates having six levels since the two hierarchies of dates contain the same lowest and *ALL* level. The advantage of aggregate views as per Definition 13 is that the entire set of views of a data cube with multi-level hierarchies can be represented as a *data cube lattice* [HRU96]; Figure 7.2 shows an illustration of the lattice of the SSB cube. Any view is represented by the level of each dimension, omitting any *ALL* levels. The single view on the lowest level corresponds to the OLAP query that contains all base facts, i.e., the view returns all non-aggregated facts from the SSB dataset. The view contains maximum $2,555 * 30,000 * 200,000 * 2,000 * 51 * 11 \geq 1.7 * 10^{19}$ facts, however, SSB provides a sparse data cube with $6,000,000$ facts. From this lowest view one can reach higher views via *roll-up* operations on dimensions, e.g., the next higher view on the right side rolls up to the *ALL* level of quantity. The single view on the highest level in Figure 7.2 corresponds to the OLAP query that returns one single fact grouping by the special-type level *ALL* with the single member *ALL* for each dimension.

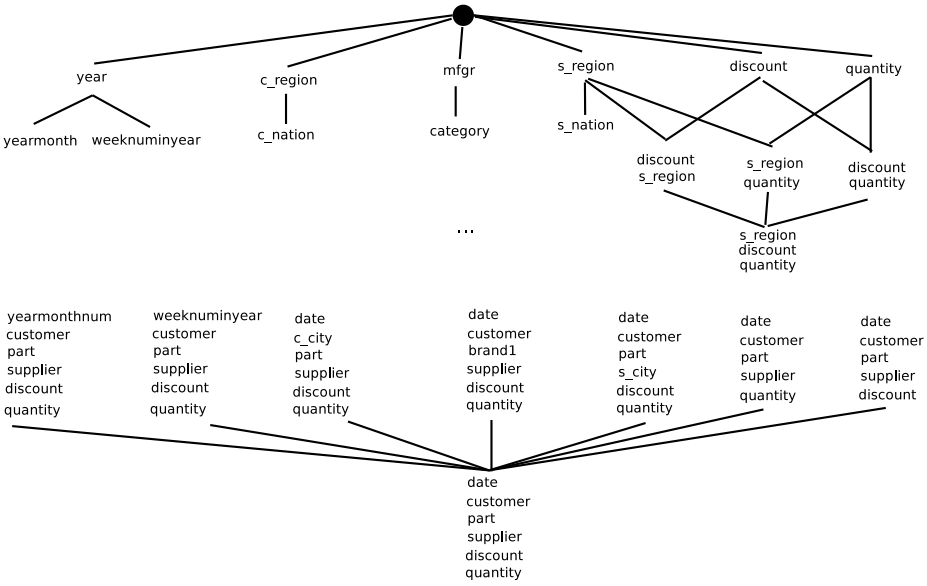


Figure 7.2: Illustration of data cube lattice of SSB data cube.

The higher the view on a path in the lattice, the fewer facts it contains, since higher levels group lower-level members into groups of fewer members.

The type of aggregation function determines whether facts can be further aggregated from pre-aggregated facts [GCB+97]. For distributive aggregation functions such as

COUNT and SUM as well as algebraic aggregation functions such as AVG this is the case, whereas for holistic aggregation functions such as MEDIAN, aggregation always has to be done starting from the most granular level of detail.

Since SSB only uses distributive aggregation functions such as SUM and algebraic formulas over measures[1], e.g., `SUM(sum_profit)` and `sum_profit = rdfh:lo_revenue - rdfh:lo_supplycost`, we do not run into summarisability problems [GCB+97] and a view or query can be computed from any view on a lower level that can be reached via a *roll-up* path; for instance, the view grouping by quantity on the right upper corner of Figure 7.2 can be computed from the view grouping discount and quantity, s_region and quantity, their collective child view grouping by s_region, discount, and quantity, the lowest level view with all the base facts, as well as from any other reachable lower level view not displayed.

Summing up for each dimension the number of members on the lowest level, the numbers of members on each level per hierarchy, and the special-type member *ALL*, we can calculate the maximum number of facts in the entire data cube: $3,095 * 30,281 * 201,031 * 2,281 * 52 * 12 > 2.6 * 10^{19}$. Since computing all views materialises the entire data cube and thus 1) takes too much time and 2) requires too much hard disk space, we are concerned with deciding which views to materialise.

We now describe how we select and compute views as well as how we use views in query processing. Intuitively, for each query in our workload, we 1) select the *closest view*, i.e., the smallest view from which to compute the results; 2) we compute the view using SPARQL CONSTRUCT queries and store the results in RDF as a slice of the original SSB lineorder cube; and 3) in query processing, we automatically select the right slice to answer a query using the metadata of the slice. In most scenarios, we would not know the workload to optimise beforehand; since our goal is to evaluate the ideal performance gain of materialised aggregate views, we assume the exact workload to be known.

We define for a given OLAP query `Q = (c, SlicesRollups, Dices, Projections)` as per Definition 11 a single *closest view* in the lattice from which we can create the results by only scanning the facts in the view [HRU96]: We create a view `(c, SlicesRollups', Dices', Projections')` on the same cube that contains in `SlicesRollups'` for each dimension the lowest level mentioned in `SlicesRollups` and `Dices`, contains an empty set for `Dices'` and `M' = M`. The following term describes the *closest* view for *Q2.1*, the other views are translated, accordingly: `({yearLevel, ALL, brand1Level, s_regionLevel, ALL, ALL},∅,{lo_revenue})`. The view contains maximum 35,000 facts and as such is considerably smaller than the SSB dataset with 6,000,000 facts. *Q2.2* and *Q2.3* can use the same view as *Q2.1* and *Q3.3* can use the same view as *Q3.2*, resulting in less

[1] Without changing the results, the function `SUM(rdfh:lo_extendedprice * rdfh:lo_discount)` is not a holistic function as we erroneously stated in previous work [KH13].

time and less space for creating the views. Though some views can contain as many facts as there are base facts in the data cube, they often do not due to sparsity, e.g., *Q4.3* contains 4,178,699 actual from 8,750,000 possible facts. For views may still be large, in ROLAP, views typically are stored in *aggregate tables*.

Similarly, we represent views as *RDF Aggregate Views* reusing QB and store the triples together with the other data in the same triple store. See Figure 7.3 for an illustration of this approach for *Q2.1* which we explain in the following:

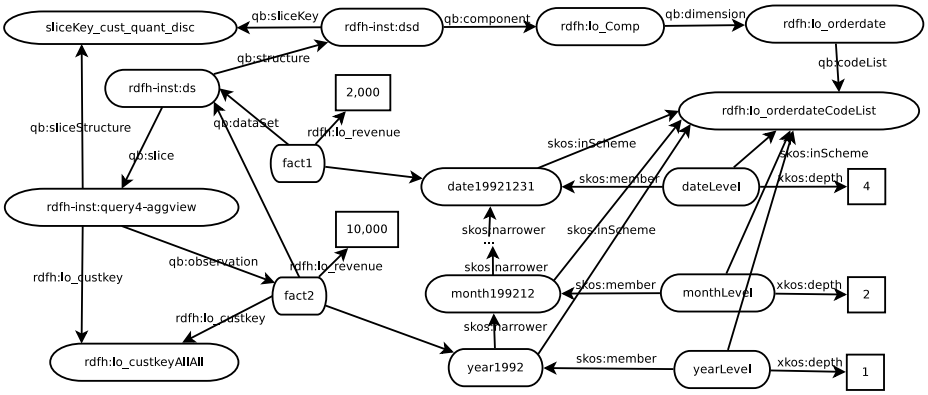


Figure 7.3: RDF graph illustrating RDF Aggregate View rolling up to year level with observation instance.

In ROLAP, additional metadata describe the table used for storing each aggregate view. With RDF Aggregate Views, we assume all metadata about aggregate views to be represented as RDF. Therefore, we have the problem of linking the original dataset to its materialised aggregate views. For dimensions on the *ALL* level, aggregate views only contain facts that fix those dimensions to the *ALL* member, e.g., *Q2.1* fixes customer. Therefore, we can represent *RDF aggregate views* as instances of `qb:Slice`.

See `rdfh-inst:query4-aggview` as an example RDF aggregate view in Figure 7.3. A `qb:SliceKey` describes the structure of a slice, i.e., the sliced dimensions (sliced dimensions in slice key not illustrated in Figure 7.3). A slice explicitly states to what member a sliced dimension is fixed, e.g., `rdfh:lo_custkey` to *ALL*. In addition to base facts, e.g., *fact1* with member *date19921231* in the date level, facts are created that aggregate on the specific levels of the view. For instance, the view of Q2.1 contains via `qb:observation` a *fact2* that rolls-up to the higher-level-member *year1992* in the year level of the dates hierarchy. The higher-level-member is connected to the lower-level-member in a `skos:narrower` path. Also, *fact2* rolls-up to the special-type *ALL* member of customer (`rdfh:lo_custkeyAllAll`). The datatype property `xkos:depth` states for each level the depth of a level starting with 0 from the (implicit) *ALL* level.

After materialisation of RDF Aggregate Views, a dataset is not a lean dataset anymore since the dataset explicitly contains facts on different levels of granularity. Therefore, when issuing queries over the entire data cube, one needs to consider summarisability and to avoid aggregating over facts several times.

Different from our assumption in the relational representation (star schema) to have for each view a new aggregate table, QB does not consider different graphs. Therefore, resulting triples are stored in the default graph.

Listing 17 shows the relevant parts of a SPARQL INSERT query on the SSB data that populates the RDF Aggregate View for *Q2.1*.

Listing 17: SPARQL INSERT query to generate RDF Aggregate View for Q2.1 in SSB benchmark.

```
1  INSERT {
2  rdfh-inst:query4-aggview qb:observation _:obs.
3  _:obs rdfh:lo_orderdate ?d_year; rdfh:lo_custkey
       rdfh:lo_custkeyAllAll; rdfh:lo_partkey ?p_brand1;
       rdfh:lo_suppkey ?s_region; rdfh:lo_quantity
       rdfh:lo_quantityAllAll; rdfh:lo_discount
       rdfh:lo_discountAllAll; rdfh:lo_revenue ?lo_revenue.}
4  WHERE {{
5  SELECT ?d_year ?p_brand1 ?s_region sum(?rdfh_lo_revenue) as
       ?lo_revenue WHERE {
6  ?obs qb:dataSet rdfh-inst:ds.
7  ?obs rdfh:lo_orderdate ?d_date.
8  ?d_yearmonthnum skos:narrower ?d_date.
9  ?d_yearmonth skos:narrower ?d_yearmonthnum.
10 ?d_year skos:narrower ?d_yearmonth.
11 rdfh:lo_orderdateYearLevel skos:member ?d_year.
12 ?obs rdfh:lo_partkey ?p_part.
13 ?p_brand1 skos:narrower ?p_part.
14 rdfh:lo_partkeyBrand1Level skos:member ?p_brand1.
15 ?obs rdfh:lo_suppkey ?s_supplier.
16 ?s_city skos:narrower ?s_supplier.
17 ?s_nation skos:narrower ?s_city.
18 ?s_region skos:narrower ?s_nation.
19 rdfh:lo_suppkeyRegionLevel skos:member ?s_region.
20 ?obs rdfh:lo_revenue ?rdfh_lo_revenue.
21 } GROUP BY ?d_year ?p_brand1 ?s_region
22 }}
```

Here, we first create a SELECT query using our OLAP-to-SPARQL algorithm on the OLAP query (line 5), then this SELECT query is made a subquery of an INSERT query. Observations roll-up to members of specific levels and fix sliced dimensions (3). Resulting triples are stored in the default graph. We can adapt our OLAP-to-SPARQL Algorithm to consider RDF Aggregate Views and not only the base facts from the SSB dataset to execute an OLAP query. Listing 18 shows the SPARQL query for *Q2.1*.

Listing 18: SPARQL SELECT query for Q2.1 in SSB benchmark, considering the RDF Aggregate View.

```
1  SELECT ?d_year ?p_brand1 sum(?rdfh_lo_revenue) as ?lo_revenue
2  WHERE {
3  rdfh-inst:ds qb:slice ?slice.
4  ?slice qb:observation ?obs;
5          rdfh:lo_custkey rdfh:lo_custkeyAllAll;
6          rdfh:lo_quantity rdfh:lo_quantityAllAll;
7          rdfh:lo_discount rdfh:lo_discountAllAll.
8  ?obs rdfh:lo_orderdate ?d_year.
9  rdfh:lo_orderdateYearLevel skos:member ?d_year.
10 ?obs rdfh:lo_partkey ?p_brand1.
11 ?p_category skos:narrower ?p_brand1.
12 rdfh:lo_partkeyCategoryLevel skos:member ?p_category.
13 ?obs rdfh:lo_suppkey ?s_region.
14 rdfh:lo_suppkeyRegionLevel skos:member ?s_region.
15 ?obs rdfh:lo_revenue ?rdfh_lo_revenue.
16 FILTER(?p_category = rdfh:lo_partkeyCategoryMFGR-12 AND ?s_region =
       rdfh:lo_suppkeyRegionAMERICA ).
17 } GROUP BY ?d_year ?p_brand1 ORDER BY ?d_year ?p_brand1
```

Here, we query for observations from slices of *rdfh-inst:ds* that fix customer, quantity and discount to the *ALL* member, as indicated in `SlicesRollups` of *Q2.1* (lines 3 to 7). In comparison to the OLAP SPARQL query of *Q2.1* without views (Listing 16), we have a reduced set of graph patterns for rolled-up dimensions (*skos:narrower* paths) (8 to 14).

With aggregate facts together with base facts stored in the data cube, the dataset is not lean and we have to distinguish between facts from views slicing the same dimensions but rolling-up to different levels. Therefore, we require for each dimension and member the correct level (9, 12, 14). Finally, we add filters on diced dimensions (16).

Different from aggregate tables in ROLAP where the respective aggregate table has to be known, in QB slices are stored in the same graph, aggregate facts are stored in the same dataset, and the query first needs to select the right slice.

7.3 Evaluation

We give an overview of tested approaches and the reasons for their selection, then explain the design of the tests. More information about the benchmark and experiments, including required and generated data, we provide on a benchmark website [KH12].

7.3.1 Design of Experiments

Table 7.3 shows an overview of tested approaches, including their main characteristics such as the data format, metadata, query language, backend, time for pre-processing, and size in terms of rows or triples.

Table 7.3: Overview of approaches tested with Star Schema Benchmark (SSB), including their main characteristics.

Name	Data Format	Metadata	Q. Lang.	Engine/Database	Pre-proc. (s)	Rows / Triples
RDBMS	Relational	-	SQL	MySQL	22	6,234,555
ROLAP-M	Relational	XML	SQL	MySQL, Mondrian	4,507	14,975,472
OLAP4LD-SSB	Graph-based	-	SPARQL	Open Virtuoso	5,352	108,021,078
OLAP4LD-QB	Graph-based	RDF/QB	SPARQL	Open Virtuoso	5,744	116,832,479
OLAP4LD-QB-M	Graph-based	RDF/QB	SPARQL	Open Virtuoso	26,032	190,060,632

RDBMS and ROLAP-M represent the traditional approaches with a widely-used Open-Source relational database (MySQL 5.1 v5.1.63) and SQL. ROLAP-M uses aggregate tables for optimising queries. The other tests represent graph-based approaches with a widely-used Open-Source triple store (Open Virtuoso v06.01.3127) and SPARQL 1.1 for aggregate and sub-queries. Whereas OLAP4LD-SSB represents SSB data without using a standard vocabulary, OLAP4LD-QB uses QB which allows us to materialise parts of the data cube as RDF Aggregate Views in OLAP4LD-QB-M.

We use the Star Schema Benchmark [OOC09], since SSB 1) refines the decision support benchmark TPC-H by deriving a pure star schema from the schema layout to evaluate analytical query engines [BPZ11], and 2) can be regarded as a realistic data source since statistics published as Linked Data are typically highly structured [Erl10, DKSU11]. We run each approach on a Debian Linux 6.0.6, 2x Intel(R) Xeon(R) CPU E5-2670 @ 2.60GHz with 16 cores, 128GB RAM and 900GB RAID10 on 15k SAS disks for local data storage. We assume unlimited amount of space but configure the databases to only use an amount of memory clearly below 100% of the space the data files surmount to (400MB for relational approaches, $<$ 650MB for graph-based approaches), since storing all multidimensional data in main memory is often too costly. For each approach we 1) translate the SSB data cube at Scale 1 with 6,000,000 lineorders into the respective data format for storage in the database, 2) simulate an OLAP engine translating the SSB OLAP queries into the respective query language of the database 3) before each test, shut-down all other applications not needed and run the test once to populate the disk cache (warm-up), and 4) document the elapsed query time of each query in turn. We do not consider data refreshes. For running the SSB benchmark and collecting the data about elapsed query times, we used the Business Intelligence Benchmark (BIBM)[2]. BIBM also ensured identical results for the approaches through qualification files. We now describe for each approach how we stored SSB data in the database and translated SSB OLAP queries to the database query language.

[2]http://sourceforge.net/projects/bibm/, last accessed 2014-09-27.

7.3.2 Description of Tested Approaches

RDBMS. We created a schema file for dimension and fact tables and populated the database with an SSB data generator. We set up column data types as recommended by SSB and primary keys for dimension tables in a standard star schema fashion[3].

Loading of 6,234,555 rows of data took 22s. The SQL queries of SSB could be reused with minor MySQL-syntax-specific modifications. We switched off query cache so that MySQL after a warm-up would not read all queries from cache. Note, we have compared those SQL queries with SQL queries created by the widely-used Open-Source ROLAP engine Mondrian (v3.4.1). Mondrian stores data cube metadata in XML and would for example deliberately query for more data than requested by the query to cache the results for later use; however, SSB minimises overlap between queries, e.g., *Q1.1* uses discounts between 1 and 3, *Q2.1* between 4 and 6. Since the performance gain of using Mondrian-created SQL queries instead of the original SSB SQL queries showed small, we only include a Mondrian test in the benchmark website (ROLAP).

ROLAP-M. We created aggregate tables without indices and without keys for (shrunken) dimension tables for the *closest* view to each query using SQL INSERT queries on the original tables from *RDBMS*. We did not use shrunken dimension tables, since no higher levels of aggregated tables were required. Also, we did not store aggregate facts in existing fact tables of data cubes but used new tables. Both decisions ensure most efficient queries.

Pre-processing time included 22s for preparing approach *RDBMS* with 6,234,555 rows and 4,485s for creating the aggregate tables with in total 8,740,917 additional rows. For each OLAP query we created an SQL query using the *closest* aggregate table. Similarly, Mondrian would choose the aggregate table with the smallest number of rows and create an SQL query with comparable performance.

OLAP4LD-SSB. With BIBM we translated the SSB tabular data into RDF/TTL files using a vocabulary that strongly resembles the SSB tabular structure: A lineorder row is represented as a URI which links for each dimension via an object property, e.g., `rdfh:lo_orderdate`, to a URI representing a row from the respective dimension table, e.g., `rdfh:lo_orderdate19931201`. From this URI, datatype properties link to Literal values for members, e.g., month "199312". Quantity and discount are directly given using datatype properties from a lineorder. Each measure is attached to the lineorder URI using a datatype property. Translation took 48sec, bulk loading of $108,021,078$ triples 5,304sec. For each SSB OLAP query, we tried to build the most efficient SPARQL-pendant to the original SSB SQL queries, e.g., reducing the number of joins.

[3] Different from as stated in the paper, we have not used indices for each foreign key in the fact table. Therefore, without affecting the overall results, query performance may have even be better without switching to a more scalable RDBMS.

Table 7.4: Overview of SSB queries and their performance-relevant features.

Feature	Q1.1	Q1.2	Q1.3	Q2.1	Q2.2	Q2.3	Q3.1	Q3.2	Q3.3	Q3.4	Q4.1	Q4.2	Q4.3
Filter factor	.019	.00065	.000075	.008	.0016	.0002	.034	.0014	.000055	.00000076	.016	.0046	.000091
View factor	.00064	.0073	.0032	.0058	.0058	.0058	.0007	.0728	.0728	.5522	.0007	.0036	.6964
RDBMS joins	1	1	1	3	3	3	3	3	3	3	4	4	4
SSB joins	5	5	6	8	7	7	9	9	7	8	10	12	12
QB joins	8	6	6	15	13	14	16	16	16	12	22	22	22
QB-M joins	9	9	9	12	11	11	13	13	11	12	13	14	14

OLAP4LD-QB. We created RDF metadata for the SSB data cube as a basis for our OLAP-to-SPARQL Algorithm and via a small script over the RDF file storing the SSB data cube for adding links from each lineorder of *OLAP4LD-SSB* to `rdfh-inst:ds`. Using SPARQL INSERT queries for each dimension, we grouped dimension members into levels of hierarchies, and added them to the triple store. Creating the *OLAP4LD-SSB* data and adding links took 48sec and 38sec, the INSERT queries 14sec; compressing and bulk loading of $116,832,479$ triples took 60sec and 5,584sec. Simulating our OLAP-to-SPARQL algorithm, we manually translated the SSB queries to SPARQL.

OLAP4LD-QB-M. For each SSB query, we created a *closest* RDF Aggregate View using a SPARQL INSERT query. Setting up *OLAP4LD-QB* took 5,744sec, the SPARQL INSERT queries 20,288sec for another $73,228,153$ triples. We created SPARQL queries that use the *closest* views.

7.4 Discussions and Lessons Learned

In this section, we evaluate 1) the scalability of our OLAP-to-SPARQL Algorithm for single-cube queries and 2) the performance gain of RDF Aggregate Views. Table 7.4 lists performance-relevant SSB query features.

Filter factor measures the ratio of fact instances that are filtered and aggregated. Filter factors are computed by multiplying the filter factors of each dice, e.g., for *Q2.1* the filter factor is $1/25$ for part times and $1/5$ for supplier.

View factor measures the ratio of fact instances that are contained in a view in relation to the 6M base facts. For example, from the filter factor and view factor, we see that query flight 4 (Q4.1, Q4.2, Q4.3) iteratively drills-down to more granular levels (up to 4,178,699 facts) but filters for fewer and more specific lineorders.

With *RDBMS joins* we describe the number of joins between tables in the SQL representation of a query. Note, ROLAP-M does not need joins. With *SSB, QB and QB-M joins* we state the number of graph pattern joins, pairs of graph patterns mentioning the same variable.

Table 7.5: SSB evaluation results with single and total elapsed query time (s).

Approach	Q1.1	Q1.2	Q1.3	Q2.1	Q2.2	Q2.3	Q3.1	Q3.2	Q3.3	Q3.4	Q4.1	Q4.2	Q4.3	Total
RDBMS	1.6	1.1	1.1	16.1	15.7	15.4	10.4	7.8	7.6	3.1	11.0	5.3	5.0	101
ROLAP-M	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.6	0.1	0.1	0.8	2
OLAP4LD-SSB	22.5	0.8	0.2	16.1	0.9	0.2	28.5	2.1	1.0	0.4	N/A	36.8	9.6	119
OLAP4LD-QB	46.1	1.3	0.2	55.0	49.4	31.1	145.7	12.5	1.8	87.2	175.3	544.5	24.9	1,175
OLAP4LD-QB-M	19.9	10.2	10.2	366.3	356.7	356.3	468.5	467.6	4.6	4.6	0.1	0.4	55.4	2,121

Table 7.5 lists the elapsed query times (s) which we now discuss. *ROLAP-M* overall is 50 times faster than *RDBMS* for not requiring any joins and a reduced number of facts to scan for aggregation. Whereas RDBMS has to first scan 6M rows and then to aggregate, ROLAP-M only has to scan the view and to aggregate from there. Affirmatively, the views of Q3.4 and Q4.3 with very low selectivity show smaller benefits. However, preparing ROLAP-M takes 200 times longer than RDBMS.

Comparing *OLAP4LD-SSB* and RDBMS, we see that the SPARQL engine is as fast as the relational alternative for some of the queries (e.g., Q1.2, Q2.1), slower for other queries (e.g., Q1.1, Q3.1, Q4.2), and even faster for others (Q2.2, Q3.3). Over all queries, OLAP4LD-SSB is only slightly worse, however, Q4.1 for no known reason does not successfully complete.

Differences can be explained by the number of joins; for instance, whereas RDBMS requires for Q3.1 and Q4.2 three and four joins, OLAP4LD-SSB requires nine and twelve joins, respectively.

If the number of joins is dless divergent, differences can be explained by the filter factor and the fact that after filtering facts still need to be aggregated. In general, the smaller the filter factor, the better the graph database seems in comparison to the relational database, for instance Q2.2, Q2.3 and Q3.3. For low-selective queries, the SPARQL engine performs worse, e.g., Q1.1, Q3.1, Q4.2. This aligns with our expectations that a SPARQL engine is more optimised for high-selectivity metadata queries without aggregations. OLAP4LD-SSB requires 243 times as much time for loading.

OLAP4LD-QB reusing QB requires up to twice as many joins than OLAP4LD-SSB (Q2.3), since hierarchies are explicitly represented through *skos:narrower* paths from higher-level to lower-level members, and consequently is 10 times slower. Both approaches require similar pre-processing time. Yet, only OLAP4LD-QB can represent hierarchies and be optimised using RDF Aggregate Views.

Although *OLAP4LD-QB-M* overall leads to 1.8 times slower queries and performs considerably worse for query flights 2 and 3 (Q2/3), it succeeds in optimising query flight 4 (Q4.1, Q4.2, Q4.3). Similar to ROLAP-M, the performance gain of RDF Aggregate Views can be explained by a reduced number of joins such as for Q4.1 and Q4.2. However, for query flights 2 and 3 (Q2.1 to Q2.3, and Q3.1 to Q3.3), OLAP4LD-QB-M

performs worse, since RDF aggregate views – different from ROLAP-M with separately created aggregate tables – are stored in the same graph and do not reduce the number of facts scanned for a query. Thus, whereas OLAP4LD-QB needs to scan for 6M facts, OLAP4LD-QB-M also needs to scan over facts from the aggregate views, in total 14.98M facts. Also, we first need to select the right view before querying the facts from the view. Queries need to compensate for the increased effort in first selecting the view, then scanning more facts by a reduced number of joins and aggregation operations, in which Q2.1 to Q3.2 apparently do not succeed.

Therefore, to a much wider extend than for aggregate tables in the relational scenario, the usefulness of RDF Aggregate Views depends on the dataset and the queries. In case of regular queries that aggregate a dataset to a high level, pre-materialisation is worth the additional time for the computation of views and the additional data in the store. For instance, the RDF Aggregate View for Q4.1 aggregates all dimensions to a high level, requires 901s for pre-computation, adds at maximum 4,375 observations to the triple store, and reduces average elapsed query time from 175.3s to 0.1s for Q4.1.

7.5 Related Work

In this section, we describe related work on 1) evaluating and optimising analytical query execution on RDF data, and 2) materialising aggregate views over RDF data.

7.5.1 Analytical Query Optimisation and Evaluation

In general, query processing over Data Cubes requires specific optimisations, e.g., Harinarayan et al. [HRU96] present the lattice framework to model dependencies among Data Cube views and state that we can assume the cost of answering a query to be proportional to the number of rows examined.

The computation and selection of views to materialise for more efficient query processing is a common analytical query optimisation method [MKIK07]. The simplest method is Algorithm 2^D [GCB+97] that computes each group-by directly from the original fact table and then takes the union of all partial results for the data cube. The simplest selection methods is the Greedy Algorithm [HRU96] that given a limit k of the number of views that can be stored next to the most detailed view (the original dataset) and the storage cost (size) of each view, with a greedy heuristic in turn selects k views to materialise. In each step the choice of a view u given the previous choices of other views S depends on the benefit $B(u,S)$ that quantifies the reduction in costs for computing u and every other view in the cube. The cost to compute a view u_1 from u_2 is given by the storage cost (size) of u_2.

For data cube computation, other common optimisation methods include executing OLAP queries using join indexes and index-only tables or to place tuples that aggregate together (i.e., tuples with the same values in the grouping attributes) in adjacent in-memory positions, so that all group-bys can be computed with as few full data scans as possible. For data cube selection, other methods include data compression and materialised views that aim to balance the trade-off between the amount of resources consumed by the data cube (such resources usually include storage space and time for incremental maintenance) and query response time. The problem of selecting the optimum subset of a cube is NP-complete, hence, all methods search for near-optimum solutions using appropriate heuristics [MKIK07, SBC$^+$07].

In our approach using the OLAP-to-SPARQL Algorithm we have chosen QB datasets as a logical representation, SPARQL as a query language for computation, and materialised *closest* views from the data cube lattice that promise the largest performance gain. We compare analytical queries on RDF with common alternatives with realistic data and queries, according to Erling [Erl10] a prerequisite for successful RDF use cases and targeted optimisations [Erl12]. Most notably, the *Berlin SPARQL Benchmark BI Use Case* allows quantifying analytical query capabilities of RDF stores, but, so far, no work compares the RDF performance with the industry-standard of relational star schemas.

Although it has been criticised that many benchmarks do not resemble RDF datasets publishing metadata, e.g., DBpedia/Wikipedia [DKSU11], we argue that multidimensional datasets published in RDF exhibit a high degree of structure and therefore benchmarks such as the Star Schema Benchmark can be used to evaluate systems for analytical queries.

Whereas most convenient in terms of loading and querying of data, RDF stores and SPARQL engines are not the fastest solution for analytical queries over Linked Data.

DipLODocus [WPWCM11] uses a hybrid storage model for RDF data where parts of the data are stored in a highly structured form into compact lists of literal values. However, the hybrid storage puts a higher overhead in loading of new datasets than common SPARQL engines. Still, separating data (observations) and metadata (data structure definition) seems a good idea; for example, one could store the observations (with a fixed structure) in a relational database for efficient query processing and all relevant metadata (including background information about entities mentioned in the observations) in a triple store for flexible enrichments with additional information.

Also using the Star Schema Benchmark, Abadi et al. [AM08] have shown that column-oriented databases are faster than row-stores since they only need to access attributes from a cube that are needed in the query. Other possible optimisation methods are in-memory execution such as in SAP HANA [VTBL13], MapReduce [KUBM12] as well as parallel computing approaches [HS13].

Also Hagedorn and Sattler [HS13] point out that MapReduce may help with efficiently scanning large datasets but requires higher effort for programming of query processing tasks such as joins. However, CameLOD, the system that Hagedorn and Sattler propose, requires large memory and multi-core processors.

SQL and SPARQL engines equally benefit from such optimisation approaches of low-level data-processing tasks. We focus on the inherent differences between the relational and RDF representation of multidimensional datasets; therefore, we have concentrated on the widely-used Data Warehouse optimisation method of data cube selection and computation and compared the performance gain in a common SQL database versus in a SPARQL engine.

7.5.2 Views over RDF Datasets

Recent work [KH11, EV12b, EV12a] discusses approaches to represent multidimensional datasets in RDF, however, no approach deals with the computation and selection of data cube slices and dices in RDF, in particular, considering the special-type *ALL* members and levels for uniquely identifying all possible facts of a data cube.

Etcheverry and Vaisman [EV12c] give an overview of approaches to specify and use views over RDF datasets, not only for query answering using views as in our case, but also for data integration and to apply security policies. Mentioned approaches SPARQL++, Networked Graphs and vSPARQL are using sub-queries or extensions to the SPARQL query language to define views. Such work is broader and would not provide a more efficient way to execute analytical queries; we define aggregate views as specific parts of a data cube and materialise them as slices represented in RDF.

Several authors also discuss materialised views over RDF data. Castillo and Leser [CL10] have presented work on automatically selecting and materialising views from a workload of SPARQL queries. In the evaluation, they use a dataset with 10M triples and disregard queries that exhibit a high selectivity. Also, Goasdoué et al. [GKLM11] have discussed the creation and selection of RDF views. Their evaluation is done on a 35M triple dataset. In contrast to these approaches, our approach considers more complex views based on aggregation functions and hierarchies, materialises views as RDF reusing QB in a triple store and evaluates the applicability for high- and low-selectivity queries on a $>$ 100M triple dataset.

Although we have evaluated RDF Aggregate Views only on a single cube, the RDF representation is sufficiently generic to materialise parts of a *multi-cube*. The main difference is that the view would not only pre-compute slices and roll-ups (aggregations), but also drill-across (joins between two datasets). Shukla et al. [SDN00] present specific algorithms and cost models for the selection of materialised views for more efficient query processing over multi-cubes.

7.6 Conclusions

The results in this chapter can clarify two hypotheses related to our research question on how to optimise analytical query processing:

Hypothesis 1: SPARQL engines storing statistics as schema-flexible RDF data are even less suited for analytical queries than relational databases and require optimisations such as materialisation.

A comparison of the relational and the RDF representation of multidimensional datasets indicate that the number of joins needed in queries strongly affect the performance of queries. Similarly, the differences between performance gain in the relational and RDF representation can mostly be explained by the number of joins as well as the number of items to be scanned; whereas in the relational representation we use separate aggregate tables, we store materialised RDF Aggregate Views in the same graph as the original dataset.

For both SPARQL engines and relational databases the query complexity – the filter factor and considered dimensions – make a difference; low-selectivity queries take longer which can be explained by the higher effort for the aggregation function summarising the results.

Although with MySQL and Open Virtuoso (v6) we have not used high-end SQL or SPARQL engines, our experiments give useful insights about how two widely-used Open Source databases from the relational and the RDF world compare.

First, analytical SPARQL query processing is not per se more difficult than analytical SQL query processing. Query performance strongly depends on how the data is structured. If the structure is very similar to the relational representation (as in the OLAP4LD-SSB case), the SPARQL engine is similarly fast.

However, to make the data self-descriptive the structure may be more complicated, for instance, if dimension hierarchies are explicitly represented. As a result, mixed star-shaped and path queries require more joins and query processing in the SPARQL engine becomes much slower. Thus, there is an efficiency versus flexibility trade-off.

Our experiments were repeated at the database vendor OpenLink with the column-oriented database MonetDB as well as newer commercial versions of the SQL and SPARQL Virtuoso database[4]. The experiments confirm that whereas an efficient query plan over relational data in a star schema is easy to find, it is difficult for the SPARQL engine since the schema is much more flexible (also called schema-less) than in the relational representation and since every join logically requires a self-join on a quads table (including the graph). For instance, in a SPARQL query involving the nation of a

[4]http://www.openlinksw.com/dataspace/oerling/weblog/Orri%20Erling%27s%20Blog/1732, last accessed on 2014-09-27

supplier, whereas in the relational representation, the supplier is represented by one row with exactly one nation, in the RDF representation, it is unknown that suppliers never have more than one nation. According to the experiments of OpenLink, switching from a strict star schema on a relational database to a flexible schema on a SPARQL engine, thus leads to an increase in query processing time by at least 2.7[5].

In summary, we state that a SPARQL engine is less suited for analytical scenarios where more complex data such as hierarchies and semi-structured background information is queried. Instead, an RDBMS more efficiently executes analytical queries, but requires a pre-defined schema that is more difficult to extend with semi-structured background information.

Hypothesis 2: By using materialisation, we can speed up performance of SPARQL engines so that they compete with relational databases.

Per definition, datasets contain facts on a specific granularity. Any higher level of aggregation first needs to be computed. By also storing pre-aggregated facts with the dataset, no aggregation but only lookups of values need to be done, for which we expected the SPARQL engine to do well.

However, experiments show that not only the performance gain is much less in comparison to the one achieved by materialisation by aggregate tables. Also, the performance gain in comparison to no-materialisation is limited.

The limited performance gain can be explained by two aspects: First, our aggregate tables do not contain foreign keys to Shrunken Dimension Tables. This way, queries on aggregate tables do not require joins and are very fast, but on the other hand cannot be used for further aggregation. Second, different from the relational world where data and metadata are strictly separated (e.g., the schema of a table or the table name), RDF allows to represent and query data and metadata in the same way. In the relational database, aggregate tables are separated from the dataset table and their names known. Instead, RDF Aggregate Views are stored in the same graph and provide self-descriptive links to the original dataset. Consequently, the right view within the graph is automatically selected in the SPARQL query which further complicates the execution.

Rewriting the SPARQL query may have been possible. For instance, we could have split the query in two, one for selecting the view and another for querying the data in the view. Or, we could have used SPARQL subqueries. However, written in the declarative language SPARQL, we expected the query to be sufficiently optimised by the SPARQL engine.

[5] http://www.openlinksw.com/dataspace/vdb/weblog/vdb%27s%20BLOG%20%5B136%5D/1735, last accessed on 2014-09-27.

Thus, we cannot generally confirm our hypothesis; if views are stored in the same graph and data is made self-descriptive, performance of queries is only optimised if the number of joins is considerably reduced.

As a summary, we can give an empirical argument in favour of creating a specialised OLAP engine for analytical queries on RDF. Although a triple store has shown almost as fast as a relational database, OLAP scenarios such as motivated by the Star Schema Benchmark used in our evaluation require results in seconds rather than minutes. Materialised views with aggregate tables overall reach 50 times faster queries. Queries by our OLAP-to-SPARQL approach on data reusing the RDF Data Cube Vocabulary (QB) overall are 10 times slower than queries on data without reusing QB, for a large number of joins are required for rolling-up on dimensions; yet, only QB metadata allows to explicitly represent dimension hierarchies and to materialise parts of the data cube. RDF Aggregate Views show the capability to optimise query execution, yet, overall still take six times longer for preprocessing and not nearly reach the performance gain of aggregate tables in ROLAP. The reason seems that the reduced number of joins for queries on RDF Aggregate Views often cannot compensate for the increased number of facts that are stored in the triple store and need to be scanned for query execution. We conclude that the query optimisation problem intensifies in many OLAP scenarios on Statistical Linked Data and that OLAP-to-SPARQL engines are promising that deploy specific optimisation methods such as management of RDF Aggregate Views [MG14] and cardinality estimations on star-shaped RDF data [NM11].

In future work, one could investigate the effect of schema-flexible background information to analytical queries. We expect that not only the size, but also the semi-structuredness of background information has a negative impact on the performance.

In particular, strategies to manage views over Statistical Linked Data are interesting. In Chapter 5 and Chapter 6, we used multi-cubes and the drill-across operation to query over several datasets simultaneously. In the next chapter, we formalise a unified view over multidimensional datasets available on the Web, the global cube. It would be interesting to investigate the applicability of RDF Aggregate Views to optimise queries over multi-cubes and the global cube.

Finally, a comparative analysis of materialisation and other optimisation strategies such as column-oriented, in-memory, and parallelised storage is possible future work.

8 Building the Global Cube with Complex Dataset Relationships

In this chapter, we investigate the following research question:
Research Question 4. *How can we increase the number of answers from the global cube in case of complex relationships between multidimensional datasets from the Web?*

Figure 8.1 illustrates the contribution given in this chapter. We present complex relationships (correspondences) between statistical datasets as a way to more flexibly solve semantic conflicts and to increase the number of answers in queries over the global cube.

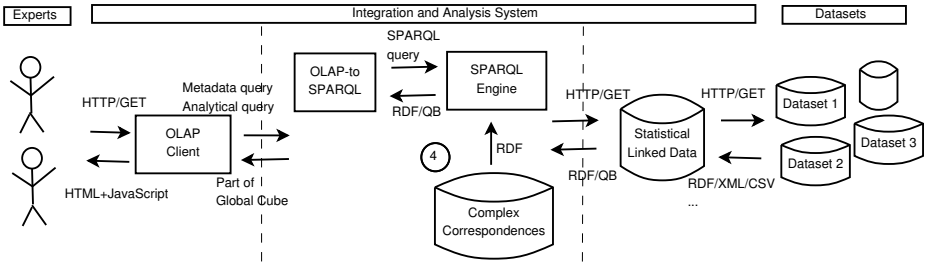


Figure 8.1: Illustration of contribution of Complex Correspondences between statistical datasets; conversion and merging relationships between datasets are declaratively described and increase the size of the global cube.

8.1 Introduction

Towards providing uniform access to governmental statistics, many datasets are also made available – directly or by third-parties – as Statistical Linked Data. Also, analytical operations over QB datasets have been defined [EV12a] and indicators from two datasets can be compared in visualisations [CAR13].

However, defining a unified schema over such datasets is still challenging since related work has so far concentrated on relational settings [CDL$^+$01] with few sources that

are centrally integrated. Also, relationships between different datasets are often buried in informal descriptions, and the routines to resolve semantic conflicts are provided in code or external background information [SSR94].

Let us assume that all datasets described in our Open Government Data scenario are available as Statistical Linked Data and that we can apply the MDM-QB Mapping (Chapter 5) and query over them using the OLAP-to-SPARQL Algorithm (Chapter 6).

Table 8.1 gives an overview of such datasets. The table shows in the rows all 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.

Table 8.1: Overview of data cubes available as Statistical Linked Data in the OGD scenario (in rows) with their dimensions (in columns) and dimension members (in cells).

Cube \Dimension	estat-wrap-:geo	estat-wrap-:unit	dcterms:date	gesis:geo	gesis:variable	estat-wrap-:indic_na	estat-wrap-:sex	estat-wrap-:age
eurostat:id/tec00115#ds (GDP Growth)	:DE...	:PCH_PRE...	2001...	-	-	-	-	-
allbus:ZA4570v590.rdf#ds (Unemploy. Fear)	-	-	2004...	:00...	:v590_1...	-	-	-
eurostat:id/tsdcc310#ds... (EU 2020 Indicator)	:DE...	...	2001...	-	-	...	...	...
eurostat:id/nama_aux_gph#ds (GDP Per Capita)	:DE...	:EUR_HAB...	2001...	-	-	:NGDPH...	-	-
eurostat:id/nama_gdp_c#ds (GDP Components)	:DE...	:MIO_EUR...	2001...	-	-	:B1G, :D21_M_D31	-	-
eurostat:id/demo_pjan#ds (Population)	:DE...	-	2001...	-	-	-	:F...	:Y18...

We use URIs as unique identifiers for datasets, dimensions, and dimension values from different data sources such as Eurostat (`eurostat`) and ALLBUS (`allbus`). For readability reasons, we describe URIs with namespaces[1], 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.

Since most publishers – also of the available data cubes in the OGD scenario – follow the practice of using an unspecific measure `sdmx-measure:obsValue` and a dimension indicating the measured variable, e.g., `estatwrap:indic_na` and `gesis:variable`, 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`.

[1] Use http://prefix.cc/ to look up prefix definition; last accessed 2014-11-18.

Every multidimensional element is published by a data source identified by the namespace. The `eurostat` namespace[2] makes available thousands of data cubes with indicators about European countries from Eurostat. Data cubes include the GDP Growth cube with the growth rate of the gross domestic product of all European countries per year, with the unit "percentage change on previous period" (`:PCH_PRE`) and for the geo dimension denoting Germany as `:DE`. Also, Eurostat provides citizens with EU 2020 Indicators, e.g., the energy dependence, productivity, and intensity as well as the greenhouse gas emission. Table 8.1 gives an example of one of those datasets; every EU 2020 Indicator cube exhibits the geo and time dimension and can contain other dimensions from the same data source. The GDP Components cube provides granular values from which other indicators can be computed. For instance, the Nominal GDP data cube (GDP at market prices) can be computed from adding the "total gross value added" (`estatwrap:indic_na` of `:B1G`) and "taxes less subsidies on products" (`:D21_M_D31`); similarly, the Nominal GDP divided by the Population should result in the GDP Per Capita.

The `allbus` namespace[3] provides the Unemployment Fear data cube. The measure describes the number of answers for a given question. The `allbus:variable` dimension denotes the type of answers given "no fear", "yes, of becoming unemployed", "yes, of having to change employer"[4]. The `gesis:geo` dimension describes the participants' country, e.g., Germany is denoted via `:00`.

Given a unified view over available data cubes in Statistical Linked Data, citizens may want to pose the following query:

GDP per Capita from Different Sources (GDP_CAP): Citizens request the GDP per Capita for all countries and years. To increase their trust in Open Government Data, citizens would like to have confirmed those values by as many data sources as possible. Data sources include cubes from different institutions but also cubes derived from other cubes. A useful solution would allow efficient querying over a large – and flexibly to extend – number of datasets that provide the GDP per Capita. For that, we present the following contributions:

- Based on a formal definition of the global cube, we describe how to derive previously unknown values in the global cube using conversion [SSR94] and merging correspondences [CDL$^+$01] (Section 8.2).
- We investigate the applicability of our approach for integrating government statistics (Section 8.3) and analyse the complexity of materialising the global cube (Section 8.4).

We describe related work in Section 8.5, and conclude with Section 8.6.

[2] http://estatwrap.ontologycentral.com/, last accessed 2014-11-06.
[3] http://lod.gesis.org/lodpilot/ALLBUS/, last accessed 2014-11-06.
[4] `allbus:variable.rdf#v590_1` to `allbus:variable.rdf#v590_3`

8.2 Approach: Global Cube and Conversion and Merging Correspondences

In the previous chapters, we used multi-cubes to query over pre-selected data cubes simultaneously and defined the drill-across operation as a way to create multi-cubes. Our overall goal of this thesis is a unified schema to query over all multidimensional datasets available and relevant for a specific use case. This unified schema we call global cube.

We describe the data of a cube `ds∈DataCube` as a relation `ds(D1, D2, ..., Dn, M)` with `dimension(ds)` the set of dimensions used by a cube and `M` the unspecific measure `sdmx-measure:obsValue`. The relation contains all possible dimension-member combinations on a specific level of detail, possibly with M an empty value such as "null" or "" We use functional datalog for describing rules about relations.

Based on drill-across defined in Chapter 6, Definition 14 defines the *global cube*.
Definition 14 (Global Cube)**.** *Given the set of all available cubes* {`ds1, ..., dsn`} *with* `dimension` *the set of all dimensions of these available data cubes, we define the global cube* `globalcube(D1, ..., Dn, M)` *with* `dimension(global-cube) = dimension`. *The global cube is defined in terms of the available cubes; for cube* `dsi(Di1, ..., Dik, Mi)`, *the following holds:* `globalcube(all, ..., Di1, ..., Dik, all ..., Mi) :- dsi(Di1, ..., Dik, Mi)`. *We denote with* `all` *the ALL member [KH13] 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(globalcube) =` $Drill\text{-}Across_{ds \in DataCube, dimension \setminus S \subseteq dimension(ds)}$ `Q(ds)`. *Since Drill-Across is commutative, the query result over the global cube does not depend on the order of Drill-Across operations.*

Similar as for drill-across, we require the global cube not to violate IC-12 in QB specification: ``''integrity constraint violation'' :- globalcube(D1, ..., Dn, M1), globalcube(D1, ..., Dn, M2), M1 != M2.``

Whereas multi-cubes (Definition 8, Chapter 5) are "virtual" data cubes over a pre-selected and limited set of overlapping data cubes, the global cube is formally defined over all multidimensional datasets available and relevant for a specific use case such as Open Government Data.

For instance, our UNEMPLOY query for Unemployment Fear and GDP Growth (introduced in Chapter 5) can be directly issued over the global cube as the following nested set of OLAP operations:

```
Slice(
    Dice(
        Projection(
            globalcube,
            {sdmx-measure:obsValue qb4o:avg}),
        estatwrap:geo,
        { eurostat:dic/geo#DE }),
    {estatwrap:unit, gesis:variable...})
```

Here, all dimensions such as the unit and variable are sliced and the average of measures over all years for Germany requested. Assuming the global cube is defined from the Unemployment Fear and GDP Growth data cubes, we execute the slice and dice operations over each cube separately, and join the results using drill-across. The rewritten OLAP query is then equal to the nested set of analytical operations for the UNEMPLOY query (Listing 11).

In Chapter 6, we presented one possibility to reduce heterogeneities between data cubes to increase the number of answers returned for queries over the global cube: Slicing of dimensions and equivalence mappings between shared dimensions and members. In the following, we present another possibility: converting and merging of data cubes.

A *conversion correspondence* according to Definition 15 describes relationships between two cubes in terms of their dimension-member combinations (`inputmember, outputmember` $\in 2^{Dimension \times Member}$), i.e., how facts with certain members on dimensions in an `inputcube` can be converted to facts with other members on such dimensions in an `outputcube`. The actual conversion is described using a conversion function `f` $\in$ `Function` that describes how the value of the measure of the `outputcube` can be computed by the value of the measure of the `inputcube` and that may be implemented in any programming language [SSR94].

Definition 15 (Conversion Correspondence)**.** *We define a conversion correspondence adapted from correspondences over relational data [CDL+01] and conversion functions [SSR94] as follows:* `ConversionCorrespondence = {(inputmembers, outputmembers, f)` $\in 2^{Dimension \times Member} \times 2^{Dimension \times Member} \times$ `Function }` *with* `Function: String` $\rightarrow$ `String`*. Given two data cubes* `ds1(D11, ..., D1n, M1)` *and* `ds2(D21, ..., D2n, M2)` *with* `D1i = D2i, 1 <= i <= n`*. A conversion correspondence* `cc` *between the cubes,* `cc(ds1) = ds2`*, holds if the following rule holds:* `ds2(D21, ..., D2n, M2) :- ds1(D11, ..., D1n, M1), inputmember` $\in$ `inputmembers hold for ds1, outputmember` $\in$ `outputmembers hold for ds2,` $(\forall$ `D1i` $\in$ `dimension(ds1)\inputmembers) D2i = D1i, M2 = f(M1).`

We define a *convert-cube* operation with `Convert-Cube: DataCube` $\times$ `ConversionCorrespondence` $\rightarrow$ `DataCube` to denote the application of a conversion correspondence to an input data cube to result in a new derived cube with the same

structure as the input cube: `Convert-Cube(ds1, cc) = ds2 <=> cc(ds1) = ds2`.

For instance, the relationship between the member "Million Euro" and "Euro" in Eurostat can be described with the following correspondence: `MIO2EUR = ({(estatwrap:unit, eurostat:dic/unit#MIO_EUR)}, {(estatwrap:unit, eurostat:dic/unit#EUR)}, "` $f(x) = 1{,}000{,}000 \cdot x$ `")` The application of `MIO2EUR` over the GDP Components data cube is denoted by `Convert-Cube(estatwrap:id/nama_gdp_c#ds, MIO2EUR)` and returns a new data cube containing values with unit "Euro". To allow the consecutive application of conversion correspondences to a data cube in a nested set of convert-cube operations, each convert-cube operation we evaluate using a SPARQL 1.1 CONSTRUCT query generating the RDF of the derived cube to which in turn another convert-cube operation can be applied. Listing 19 shows the SPARQL query for our example.

Listing 19: SPARQL CONSTRUCT query to evaluate `MIO2EUR` over GDP Components data cube.

```
1  CONSTRUCT {
2  ds12c44:ds qb:structure ?dsd .
3  _:outputobs qb:dataSet ds12c44:ds;
4      estatwrap:unit eurostat:dic/unit#EUR;
5      gesis:geo ?gesisgeo;
6      estatwrap:geo ?estatwrapgeo;
7      estatwrap:indic_na ?estatwrapindicna;
8      dcterms:date ?dctermsdate;
9      sdmx-measure:obsValue ?outputvalue1 .
10   } where { {
11 select ?dsd ((1000000 * ?inputvalue1) as ?outputvalue1)
       ?estatwrapgeo ?gesisgeo ?dctermsdate ?estatwrapindicna
       ?inputvalue1
12 where {
13 estatwrap:id/nama_gdp_c#ds qb:structure ?dsd .
14 ?inputobs qb:dataSet estatwrap:id/nama_gdp_c#ds;
15     estatwrap:unit eurostat:dic/unit#MIO_EUR;
16     gesis:geo ?gesisgeo;
17     estatwrap:geo ?estatwrapgeo;
18     estatwrap:indic_na ?estatwrapindicna;
19     dcterms:date ?dctermsdate;
20     sdmx-measure:obsValue ?inputvalue1 .
21   } } }
```

The SPARQL CONSTRUCT query can be divided by graph patterns in the body (line 13 to 20 in Listing 19) that provide bindings for graph patterns in the head (line 2 to 9) that in turn define the constructed triples. Since in our implementation no functions are possible in graph patterns (see `1000000 * ?inputvalue1`), we surround body graph patterns with a SPARQL SELECT query. The query generates for every fact in the input cube with unit `eurostat:dic/unit#MIO_EUR` a new fact with unit `eurostat:dic/unit#EUR` in an output cube with the same structure (dimensions

and measures); the value of the (generic) measure is $1,000,000$ times the value of the input cube's measure. Along this example, we explain how a cube and a conversion correspondence as input to a convert-cube operation can be translated to the respective SPARQL CONSTRUCT query. The body graph patterns are created in the following steps:

1. **Dataset Triples:** We bind the data structure definition and observations from the dataset URI of the input cube (line 13 and 14).
2. **Inputmembers Triples:** For each dimension-member combination in `inputmembers`, we add a respective graph pattern (15).
3. **Dimensions Triples:** For each dimension from the input cube which is not contained in `inputmembers`, we bind from the observation the value for the dimension URI to a variable (e.g., `?gesisgeo`) derived from the dimension URI to refer back to it in the head graph patterns later (16 to 19). Since the data cubes share their geo dimensions, there are graph patterns for `gesis:geo` and `estatwrap:geo`.
4. **Measures Triples:** For each measure in `inputcube`, we bind from the observation the value to a variable that is unique per measure for referral in other parts of the rule (20).
5. **Function Triples:** For each measure in `inputcube`, we bind a variable for the derived dataset's measure with an expression for function `f` with the input variable of `f` replaced by the respective measure variable (11).

Similarly, we create the graph patterns in the head:

1. **Dataset Triples:** We create a URI for the derived output dataset from a combination of the input dataset URI and the name of the conversion correspondence (in our example `ds12c44:ds` (line 3) comes from an internally used function `createuri(estatwrap:id/nama_gdp_c#ds, MIO2EUR)`); we add the data structure definition of the input dataset to the output dataset (line 1); we add new observations to the output dataset using a blank node[5] (line 2)
2. **Outputmembers Triples:** For each dimension-member combination in `outputmembers`, we add a respective graph pattern (line 4).
3. **Dimension Triples:** For each dimension from the input cube which is not contained in `outputmembers`, we add to the new observation the dimension values of the observation in the body (line 5 to 8).
4. **Measure Triples:** For each measure in `inputcube`, we assign to the respective measure in the derived observation the variable describing the converted value from the body (9).

[5] We could also use a function over `?inputobs` to create a new URI for each derived observation.

The Dimension Triples make sure that the derived data cube has the same dimensions as the input cube and copy all dimension values not touched by the conversion correspondence. For that, contrary to the so-called *open-world assumption* (OWA)[6] in Linked Data, we have to assume all dimensions stated by the data structure definition of the input dataset to be known.

Otherwise, according to the open-world assumption one could not be sure that somewhere on the Web additional dimensions are given to a dataset. However, all dimensions have to be known to complete a conversion. A closed world assumption for the data structure definitions of data cubes is particularly important in the case of multi-cube operations such as drill-across and the extension to convert-cube, merge-cubes. Otherwise, with possibly additional dimensions on which to check equality, two observations from separate cubes could never be joined. A closed-world behaviour could also be achieved with scoped negation in rules [PFH06].

The SPARQL query is evaluated over the RDF representing the data cube to generate the derived cube. To answer a query over the global cube, we need to take into account all derived data cubes, including those derived by nested convert-cube operations. Given an OLAP query with nested convert-cube operations, any convert-cube operation is evaluated using one evaluation of the respective SPARQL CONSTRUCT query over the input data cube's RDF. Iteratively, the RDF of the input data cube may first need to be derived by the evaluation of another convert-cube operation. In the next section, we will describe an analysis of the number of derived data cubes in the global cube.

We can extend conversion correspondences to merging correspondences to combine values from two cubes. A *merging correspondence* according to Definition 16 describes how facts with certain members on dimensions in two data cubes can be merged to facts in a third data cube with members on such dimensions and with the same structure as the first input cube.

Definition 16 (Merging Correspondence)**.** *We define* `MergingCorrespondence` $= \{$ `(inputmembers1, inputmembers2, outputmembers, f)` $\in 2^{Dimension \times Member} \times 2^{Dimension \times Member} \times 2^{Dimension \times Member} \times$ `Function` $\}$ *with* `Function: String × String → String`*. Given three data cubes* `ds1(D11, D12 ..., D1n, M1)`*,* `ds2(D21, ..., D2n, M2)`*, and* `ds3(D31, ..., D3n, M3)`*. A merging correspondence* `mc` *between the three cubes,* `mc(ds1, ds2) = ds3` *holds if the following rule holds:* `ds3(D31, ..., D3n, M3) :- ds1(D11, ..., D1n, M1), ds2(D21, ..., D2n, M2), inputmember1 ∈ inputmembers1 hold for ds1, inputmember2 ∈ inputmembers2 hold for ds2, outputmember ∈ outputmembers hold for ds3, (∀ D2i ∈ dimension(ds2) \ inputmembers2) D2i = D1i, (∀ D1i ∈ dimension(ds1) \ inputmembers1) D3i = D1i, M3 = f(M1, M2).`

[6] See explanation at http://www.w3.org/TR/owl2-primer/, last accessed 2014-12-11.

We define a *merge-cubes* operation with `Merge-Cubes: DataCube × DataCube × MergingCorrespondence → DataCube` to denote the application of a merging correspondence to two input data cubes to result in a derived cube.

The following example computes the Nominal Gross Domestic Product (NGDP) from the sum of two GDP component indicators:

`COMPUTE_GDP = ({(estatwrap:indic_na, eurostat:dic/indic_na#-B1G)},{(estatwrap:indic_na,eurostat:dic/indic_na#D21_M_D31)}, {(estatwrap:indic_na,eurostat:dic/indic_na#NGDP)},"`$f(x_1,x_2) = x_1+x_2$`").`

And the following example computes the GDP per Capita in Euro per Inhabitant from the Nominal GDP and the Population:

`COMP_GDP_CAP = ({(estatwrap:indic_na,eurostat:dic/indic_na#-NGDP),(estatwrap:unit, eurostat:dic/unit#EUR)} ,{(estatwrap-:sex, eurostat:dic/sex#T), (estatwrap:age, eurostat:dic/age-#TOTAL)}, {(estatwrap:indic_na,eurostat:dic/indic_na#NGDPH), (estatwrap:unit, eurostat:dic/unit#EUR_HAB)},"`$f(x_1,x_2) = x_1/x_2$`").`

Here, from the second input cube only facts are selected that contain measures for all genders and age groups, assuming they describe the population. The algorithm to evaluate convert-cube using a SPARQL CONSTRUCT query can be extended to the merge-cubes operation.

In the next section, we evaluate the applicability of conversion and merging correspondences to integrate governmental statistics.

8.3 Evaluation

Figure 8.2 illustrates how a client issues the *GDP_CAP* query (using the query language MDX) to the designed integration system over a global cube defined by data cubes GDP Per Capita, GDP Components and Population. The integration engine 1) loads and validates available data cubes defined by QB datasets (`Base-Cube`), 2) translates the query over the global cube to a logical operator query plan over the available as well as all derived data cubes, 3) transforms the logical to a physical operator query plan with iterators that 4) are then executed. Here, `MIO2EUR` converts "Million Euro" to "Euro", `COMP_GDP` computes the Nominal GDP, and `COMP_GDP_CAP` computes the GDP Per Capita which in the global cube is brought together with values from the GDP Per Capita data cube.

We implemented all operations, including the drill-across, convert-cube, and merge-cubes operations, in a Java program as logical operators and physical iterators. The

program uses a directed crawling strategy to load and validate all data cubes into a Sesame Repository (v2.7.10) as embedded triple store. Due to lack of space, in the further descriptions we assume all available data cubes loaded[7].

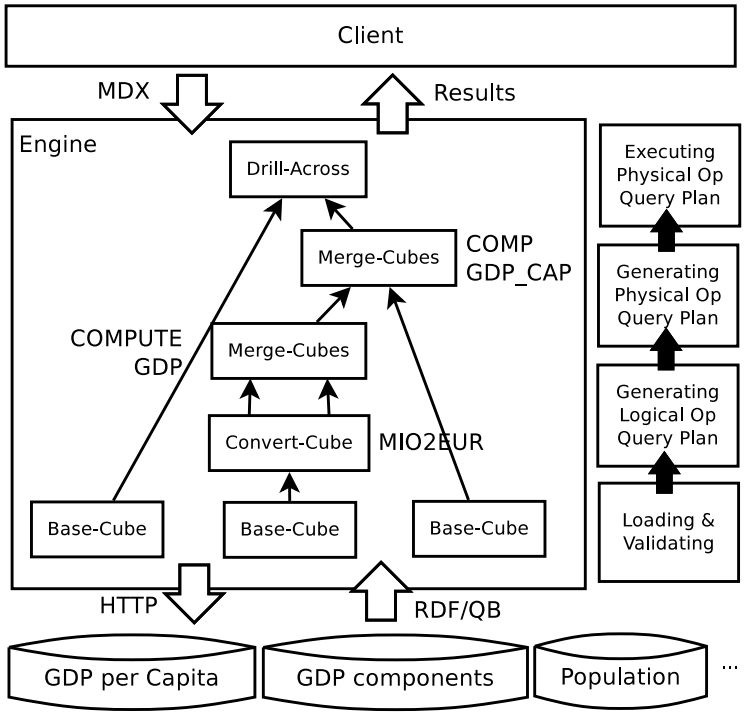


Figure 8.2: Overview of integration system using conversion and merging correspondences.

Given a query and a set of available datasets, the logical query plan is automatically generated per Definition 14 of the global cube, and by automatically applying mappings between shared dimensions and members as well as conversion and merging correspondences.

Drill-Across is implemented as a nested loop join directly over the results of the OLAP-to-SPARQL Algorithm. We evaluate OWL semantics with the equivalence duplication strategy and repeatedly execute SPARQL INSERT queries implementing entailment rules of equality[8] to materialise implicit triples from equivalence statements.

[7] Additional information can be found on the evaluation website of the respective paper [KSH14], http://www.linked-data-cubes.org/index.php/Global_Cube_Evaluation_EKAW14, last accessed 2014-31-10.
[8] http://semanticweb.org/OWLLD/#Rules, last accessed 2014-11-18.

Every SPARQL query defined by a convert-cube and merge-cubes operation is evaluated once and the result is loaded in the triple store for usage by consecutive operations.

Setup: We created a ConversionCorrespondence for `MIO2EUR` and MergingCorrespondences for `COMPUTE_GDP`, and `COMP_GDP_CAP`.

To show the applicability of the convert-cube and merge-cubes operations, we execute the logical query plan as illustrated in Figure 8.2 for comparing GDP per Capita from different datasets. In this case, we assume that we know that only two (derived) data cubes serve the requested values: the GDP Per Capita data cube and the GDP per Capita as derived from the Nominal GDP and the number of inhabitants. Consequently, no time is spent for metadata queries and generating the logical query plan. In the next section, we will give an estimation of the number of derived datasets to query from the global cube in case the logical query plan is not given.

The *GDP_CAP* query we executed five times on an Ubuntu 12.04 workstation with Intel(R) Core(TM) i5 CPU, M520, 2.40GHz, 8 GB RAM, 64-bit on a JVM (v6) with 512M initial and 1524M maximum memory allocation.

Results: We successfully executed the GDP_CAP query, the resulting cube allows us to compare the computed and the given GDP per Capita from Eurostat. By the small divergence between the computed and the given values, we presume that the computations are correct; for instance, for UK in 2010, the Nominal GDP per Capita is directly given as $27,800$ and computed as $27,704$ Euro per Inhabitant.

On average, the query takes 246sec. We load 1,015,044 triples that do not only come from 10 lookups to the three integrated datasets, but also from loading derived datasets to the embedded triple store. Similarly, in total, the engine loads or creates 126,351 observations. A long time of 119sec for the 10 look-ups, loading, and validating results from the fact that integrated datasets per se are larger than the datasets we have loaded in previous experiments, e.g., the Population data cube is described by more than 22MB of RDF/XML.

Generating the physical operator query plan in 16sec on average is fast, but executing the query plan in 111sec on average takes as long as loading and validating. This is because different from the pipelining strategy of the Drill-Across iterator to directly process the results of a previous iterator, for convert-cube and merge-cubes the physical query plan involves materialising data cubes as derived cubes and storing them in the embedded triple store for the next iterator. From 111sec needed for processing the physical query plan, on average 91sec (82%) was spent on generating and storing the derived data cubes.

Although we only integrated three datasets and only computed three derived data cubes, with more than 4min, the elapsed query time takes too long for an exploratory analysis. Nevertheless, our experiment shows that we can materialise and query any derived data

cube. Since in many cases datasets will not often change (e.g., one new GDP per Capita value per year), an offline computation of all derived data cubes, i.e., *building* the global cube, is possible. The time will mainly depend on the number of derived data cubes. Therefore, in the next section, we give an estimation.

8.4 Analysis of the Global Cube

In this section, we want to estimate the difficulty to materialise the global cube for efficient query processing. The size of the global cube mainly depends on the number of derived data cubes.

Given a set of data cubes and conversion and merging correspondences, an algorithm to generate all derived data cubes to answer a query over the global cube may not terminate since correspondences can be infinitely nested.

If we forbid that the same correspondence is applied repeatedly, we can give an upper bound estimation of the number of (derived) cubes based on ds datasets and mc merging and conversion correspondences: $noderivedds(ds,mc)$.

For that, we define a recursive function $noc(dp,ds,mc)$ that distinguishes the depth dp of a nested set of correspondence applications with $noc(0,ds,mc)=ds$, $noc(dp,ds,mc)=mc^2*noc(dp-1,ds,mc-1)^2+2*mc^2*\sum_{0<=i<dp-1} noc(dp-1,ds,mc-1)*noc(i,ds,mc-1)$. In the recursion, since the merge-cubes operation is not commutative, we need to consider the ordering of inputs to the merging, i.e., the consecutive application of merging correspondences in the left, right and both inputs. Then, $noderivedds(ds,mc)=noc(0,ds,mc)+noc(1,ds,mc)+\ldots+noc(mc,ds,mc)$.

As an example, for our GDP_CAP query we assume the GDP Per Capita, the GDP Components, and the Population as data cubes, `MIO2EUR` as conversion, and `COMP_GDP` and `COMP_GDP_CAP` as merging correspondences. The maximum number of (derived) cubes is given by: $noderivedds(3,3)=noc(0,3,3)+noc(1,3,3)+noc(2,3,3)+noc(3,3,3)=3+81+13,608+3,003,480=3,017,172$.

However, most derived data cubes are empty, e.g., `MIO2EUR(GDP Per Capita)` since GDP Per Capita does not contain values with unit "Million Euro". Therefore, if we require in a nested application of correspondences that `outputmembers` of the first correspondence fit `inputmembers` of the second correspondence, the number of derived datasets in many cases is reduced as we show in the following for our example.

To generate all possible derived data cubes in our GDP_CAP query, we use a functional Datalog program (using XSB Prolog notation), where we define the datasets using relation `dataset(Ds)`, dimensions using `dimension(Ds, Dim)` and dimension-member combinations using `dimensionmember(Ds, Dim, Mem)`; also, we de-

fine every conversion and merging correspondence using four rules for 1) generating the dataset, 2) copying over the dimensions, 3) copying over the dimension-members, and 4) setting the new dimension member. For instance, rule 1) for `MIO2EUR` is as follows:

```
dataset(mio2eur(X)) :- dataset(X), dimension(X,unit), ( \+
    dimensionmember(X,unit,Z); dimensionmember(X,unit,mioeur) ).
```

Here, MIO2EUR only is applied over data cubes that have a dimension unit and that either have Million Euro or no specific unit specified.

Executing the program in XSB Prolog, we get a small number of 54 derived datasets, including a computation of the GDP per Capita via `comp_gdp_per_cap(comp_gdp-(mio2eur(gdpcomponents),mio2eur(gdpcomponents)),population)`, to evaluate using SPARQL. The execution takes milliseconds on commodity hardware, since the program only contains 14 atoms and 12 rules; the facts do not need to be represented in the program since 1) the actual conversion and merging is done separately and 2) we find matches between datasets to convert or merge only by looking at the definition of correspondences, i.e., output- and inputmembers.

Lemma 1 allows to have cycles in the definition of input-/outputmembers, e.g., conversion correspondences `MIO2EUR` and `EUR2MIO`.

Lemma 1. *We can design an algorithm that after a finite number of steps terminates with the application of conversion and merging correspondences when no new facts can be added to the global cube.*

Sketch of Proof According to Definition 12 of Drill-Across and Definition 14 of the global cube, an "Integrity Constraint Violation" is returned for a measure if two input cubes during the computation of the global cube contain different measure values for identical dimension-member combinations.

Only different dimension-member combinations can provide new facts, all other data cubes either provide facts with the same measure value or an "Integrity Constraint Violation". If dimension-member combinations are limited, so are derived cubes that provide new facts to the global cube.

The number of dimension-member combinations is limited: Considering ds datasets as special combinations and the order of combining combinations, cc conversion and mc merging correspondences provide at max $(ds+cc+mc)!$ combinations of dimension-member combinations. □

In our example, this would result in $(3+1+2)! = 720$ possible combinations.

Assuming one derived data cube per combination and three derived datasets per four minutes as estimated by our experiment in the previous section, the materialisation of

all derived data cubes in this example will require 16h. Considering that our example with three datasets, one conversion, and two merging correspondences is rather small, we can expect that building the global cube will easily become a difficult problem.

8.5 Related Work

We distinguish approaches for 1) automatically integrating data warehouses and datasets, and 2) for overcoming heterogeneities.

8.5.1 Data Warehouse Integration

Many integration systems build on XML. Perez et al. [PBAP08] distinguish tree integration architectures: the mediator-based federated, the XML-database federated, and wrapper-based architecture. The mediator-based federated architecture requires per integrated data warehouse a mediator that translates queries over the global to the local schema. Maintaining the mediators has shown error prone and costly. The XML-database federated architecture requires less effort in maintaining mediators but has the disadvantage that XML databases do not compute aggregations as efficiently as relational database management systems [PBAP08]. The wrapper-based architecture (distributed data warehouse architecture) uses a Collection Server to store data from distributed sources wrapped to XML. Our approach is a mixture: We rely on wrappers [CDL+01] such as Estatwrap publishing original data sources as Statistical Linked Data; the main integration work is done by mediators [CDL+01] such as our implemented Java program that crawl the necessary data, store the data in a triple store, evaluate declarative relationships between data sources to build the global cube, and execute SPARQL queries. Relationships can be provided directly with Linked Data, e.g., equivalence mappings, or additionally by experts, e.g., Complex Correspondences.

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 [CDL+01, CCGL02, Gen10]. 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 [CCGL02].

For finding relationships between multidimensional datasets, common ontology matching approaches are less suitable [ZM14]. Torlone [Tor08] automatically matches heterogeneous dimensions by checking requirements of shared dimensions such as coherence and soundness; similar to our work, they use joins and materialisation approaches.

We integrate data cubes from the Web and focus on more complex mappings that explicitly need to be given by experts.

8.5.2 Resolving Semantic Conflicts

Different from materialised aggregate views for performance optimisation, in this chapter, we have defined the global cube as an integrated view over available datasets in Statistical Linked Data.

Tseng and Chen [TC05] define a classification scheme of the semantic conflicts between local cubes:

- Cube-to-cube conflicts (same conceptual modelling but different logical modelling)
- Dimension-to-dimension conflicts (dimension schema conflicts, dimension member conflicts, naming conflicts)
- Measure-to-measure conflicts (measure naming conflicts, inconsistent measures, measure scaling conflicts)

Their definitions of conflicts is difficult to understand since the authors do not properly define terms for relationships between multidimensional elements, e.g., ”semantically-related”, ”mismatched” and ”semantic discrepancies”. Instead, our work uses well-defined RDF vocabularies based on OWL and RDFS to describe equivalence relationships between shared Dimensions and Members.

Tseng and Chen distinguish the ”global schema approach”, the ”federated approach” and the ”multi-database query language approach”. Their approach is a ”global schema approach”; they extend the idea of XCube to integrate multiple data cubes. Semantic conflicts are resolved using XQuery/XSLT. In their solution, the cubes are adapted to one new global schema with dimensions having canonical identifiers. However, XQuery/XSLT scripts need to be created manually. Mapping from MDX directly to XQuery/XSLT is open work.

Different from this approach to overcome heterogeneities, we allow solving of semantic conflicts with abstract conversion and merging relationships.

Bischof and Polleres [BP13] introduce “attribute equations” that are closely related to our work. Attribute equations describe arithmetic relationships between datatype properties. The work includes an RDF representations of such equations using a new property `definedByEquation`. The authors consider variants of equations for all possible directions to convert one or more values of datatype properties to the value of another datatype property. They adapt an existing query rewriting algorithm to reformulate a SPARQL query with certain datatype properties according to attribute equations.

For that, UNION graph patterns are added for each possible way to compute the value of a datatype property from other datatype property values. To avoid infinite expansion of equations when transforming the SPARQL query, the algorithm ensures that equations are not applied recursively. Similar to our work, Bischof and Polleres [BP13] have a notion of semantic conflict. In their definition, an RDF graph is not coherent if attribute equations lead to different values for the same datatype property of an instance. Our definition leads to an integrity constraint violation when the global cube contains different values for the same dimension-member combination. Whereas attribute equations require very specific datatype properties such as `populationRateMale`, complex correspondences from the present work describe relationships between multidimensional datasets; therefore, every dimension of a dataset can be considered when converting and merging values. Since materialising derived data cubes is costly, a query rewriting approach may be promising but – due to the structure of observations – would be more complex than for attribute equations.

Other work tries to automatically derive new from existing data. Ambite and Kapoor [AK07] presents 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. Bressan and Goh [BG97] present a context mediation network based on Datalog and constraints. The authors give examples of possible query optimisations based on semantic descriptions. For instance, a data source may not be processed if requested data is not expected to be contained in the source. Whereas in such work, data sources provide data as relational tables, we deal with a higher abstraction as data cubes, and with the specificities of Linked Data sources with resolvable URIs. Also, we demonstrate the problem of efficiently integrating available datasets in a global cube.

Wilkinson and Simitsis [WS11] propose flows of hypercube operators as a conceptual model from which ETL processes can be generated. The Linked Data-Fu language [SSHS13] uses N3 rules for describing complex data processing interactions on the Web. A rule engine could possibly improve our query processing approach, e.g., by bulk-loading, crawling and query processing in parallel threads and if backtracking from a query is supported. However, N3 does not support functions such as needed in our conversion and merging correspondences. Also, we provide an abstraction layer specific to multidimensional datasets published as Linked Data. Etcheverry and Vaisman [EV12a] map analytical operations to SPARQL over RDF but do not define multicube operations and mappings.

Siegel et al. [SSR94] 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. Calvanese et al. [CDL$^+$01] describe a rule-based approach to automatically

find the matching between two relational schemas. We extend their approaches to data cubes published as Linked Data.

Diamantini et al. [DPS13] 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.

8.6 Conclusions

As the number of statistical datasets published as Linked Data is growing, citizens and analysts can benefit from methods to integrate national indicators, despite heterogeneities of data sources. In this chapter, we have shown that we can provide a unified view, the global cube, over multidimensional datasets available as Statistical Linked Data. Furthermore, we can increase the number of answers from the global cube with conversion and merging mappings between datasets.

Complex Correspondences describe rules of how data cubes can be derived from other data cubes. Complex correspondences are evaluated using two new types of OLAP operations: convert-cube and merge-cubes. We use rules to define both the semantics of correspondences and the semantics of the operations. For query processing, all derived data cubes can be materialised to build the global cube. Depending on the number of correspondences and the evaluation method of rules, the number of derived data cubes is very large.

We conducted an experiment of querying the GDP Per Capita motivated by our Open Government Data scenario. The experiment used a manually generated query plan and showed that even with few derived data cubes, query processing takes a lot of time. Offline computation of derived data cubes from data cubes that are seldom updated may be a possibility. The experiment showed that complex correspondences can reduce heterogeneities between datasets. Important statistical indicators can be confirmed by several datasets for increased trust in the data.

There is several interesting future work.

Many datasets do not sufficiently describe their meaning for integrating with other data sources. For instance, the name of a dataset still often contains important information about the meaning of the content, such as is the case for the population dataset that would distinguish between different genders and age groups but would not express the

unit ”Euro per Inhabitant” or the indicator ”population”. When integrating underspecified datasets, their facts will appear to describe the same things with different values; in this case, the global cube cannot be built since integrity constraints are violated. Besides helping publishers, one possibility would be to allow for generating new dimensions and members in datasets. Then, correspondences can be added by domain experts to uniquely define the meaning of datasets. For instance, a new dimension such as “variable” with value “population” could be added or a specific measure “population” could be given.

Currently, correspondences are manually created by the application designer. It is an open question how to more easily generate and possibly maintain correspondences and to feed those relationships to the system. For that, an RDF representation to be shared in Statistical Linked Data such as presented for attribute equations [BP13] may be useful. Also, publishers may provide correspondences. For instance, XBRL allows “Calculation Linkbases” that describe simple part-of relationships between financial concepts and can automatically be extracted.

In future work, it would be interesting to represent and execute all OLAP operations with (possibly recursive) rules. For instance Mohapatra and Genesereth [MG12] show how aggregation functions can be added to Datalog. Then, rule engines could be used to fully materialise the global cube for efficient lookups.

Finally, querying over the global cube in case of many datasets and many correspondences will require specific performance optimisations, e.g., specific cost models of aggregate views over multi-cube structures [SDN00].

In summary, complex mappings between multidimensional datasets are a promising step towards reliable and reusable numeric values. For instance, in our experiments, we noticed a difference between the GDP Per Capita for the UK in 2010 in Euro per inhabitant of $27,800$ in Eurostat[9] and of $29,520$ in Wolfram Alpha[10]. There are often differences between numeric values from different institutions, e.g., due to different dates of exchange rates and different data collection characteristics such as “sampling”. Complex correspondences introduce one way to identify and to make aware of such differences.

[9] http://appsso.eurostat.ec.europa.eu/nui/show.do?dataset=nama_aux_gph&lang=en, last accessed 2015-06-13.

[10] http://www.wolframalpha.com/input/?i=+gdp+per+capita+in+uk+in+2010+in+eur, changing over time, last accessed 2014-11-23.

9 Application and Discussion of Contributions

In this overall evaluation chapter, we apply and discuss the contributions of this thesis along our three scenarios. For that, in Section 9.1, we first give an overview of our proposed solution and describe the design and implementation of OLAP4LD, an OLAP engine to build flexible and efficient integration and analysis applications over Statistical Linked Data. Then, we describe applications of OLAP4LD to our scenarios. In Section 9.2, we present the SMART Knowledge Base (SKB) improving our SMART scenario. In Section 9.3, we present the Financial Information Observation System (FIOS) fulfilling the user requirements of our XBRL scenario. In Section 9.4, we present the Linked Data Cubes Explorer (LDCX) allowing exploration of governmental statistics in our OGD scenario.

For each of the approaches, we explain the modelling and mapping of data cubes, as well as the query processing and query optimisation over those data cubes.

9.1 Overview

We developed an *OLAP Engine for Statistical Linked Data* (OLAP4LD) to apply our contributions. OLAP4LD is used in our scenarios to fulfil the user requirements.

In this section, first, we describe the general architecture of integration and analysis applications based on OLAP4LD (Section 9.1.1). Second, we explain the implementation of OLAP4LD (Section 9.1.2).

9.1.1 Design of OLAP4LD

OLAP4LD is a framework for building analysis applications over statistics published as Linked Data. Our approach makes use of OLAP clients and the Multidimensional Expression Language (MDX) for query building and of Statistical Linked Data for the representation and extraction of multidimensional datasets from the Web.

Figure 9.1 illustrates the main components of an analysis application based on OLAP-4LD. Any OLAP client can provide a GUI for the domain expert (top of figure) to issue metadata and analytical queries. Metadata queries return multidimensional elements such as cubes, dimensions, and measures which are used to form analytical queries. Analytical queries are described in MDX and return data from the global cube to be displayed in rows, columns, and cells of a pivot table. OLAP4LD translates queries to the global cube to queries over Statistical Linked Data (bottom of figure).

OLAP4LD application developers can make use of a common abstraction of datasets as data cubes according to our Multidimensional Data Model (MDM), the quasi-standard analytical query language MDX, and existing OLAP clients.

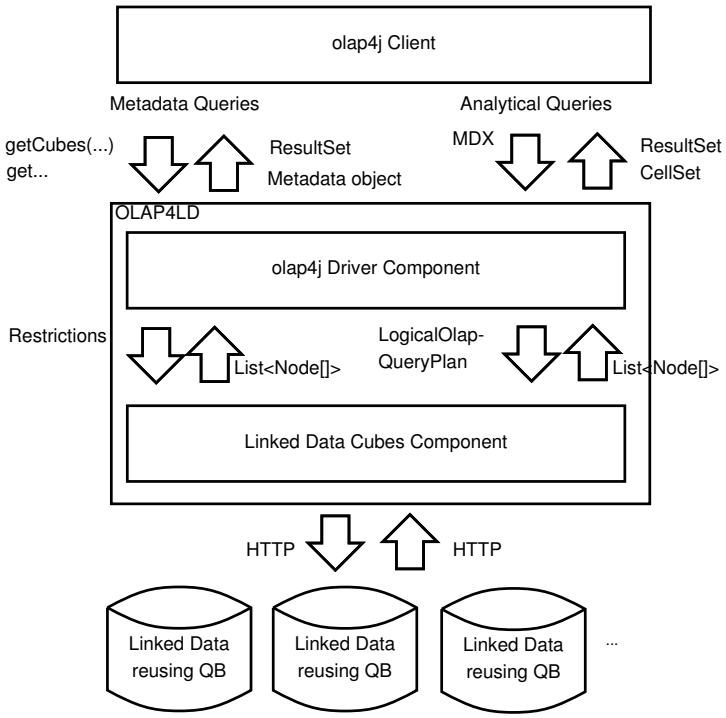


Figure 9.1: Architecture of OLAP4LD applications.

OLAP4LD consists of two components. The *Driver Component* translates queries from an OLAP client to queries more suitable for processing over Statistical Linked Data in the *Linked Data Cubes Component*. For instance, analytical queries in MDX are interpreted as nested set of OLAP operations (so-called logical OLAP operator query plans, `LogicalOlapQueryPlan`). Vice-versa, the driver component translates results from Statistical Linked Data to representations understandable by the client (re-

sults from SPARQL queries, `Node[]`, to tabular or multidimensional results, `ResultSet` or `CellSet`).

The available data cubes forming the global cube are input to the system as a set of QB dataset URIs. For instance as a special-character-separated list in a metadata query for cubes or in the FROM clause of an MDX query.

Metadata and analytical queries can be executed in different ways over Statistical Linked Data. Query processing can be done with an existing OLAP engine over relations or in-memory, and directly with an RDF store. Aggregated values from the data cube may be computed on demand or views selected and maintained. Similarly, data preprocessing and integration can be done differently, e.g., a database may be pre-filled with all relevant data in advance or populated dynamically. The quality of data can be ensured in a pre-processing step or on the filled database. Also, there are various ways in which information can be provided, e.g., packed in data dumps or queryable from several SPARQL endpoints; using different vocabularies (e.g., DCAT to describe data catalogues, VoiD to describe metadata of datasets) and various RDF representations (e.g., RDFa, Turtle). Last but not least, mappings of heterogeneous datasets can be available from different sources, e.g., directly in Linked Data and from built-in (ontology) matching algorithms.

Therefore, for executing metadata and analytical queries, an OLAP4LD application has to implement a *Linked Data Cubes Engine*. Developers can build new or extend existing engines and concentrate on the challenges of query execution and integration over Statistical Linked Data.

Linked Data Cubes Engines apply the contributions of this thesis:

Based on the MDM-QB Mapping (Chapter 5), relevant data is accessed by a SPARQL engine according to the directed crawling strategy, and metadata queries are executed. For that, any instance of `qb:DataSet` is mapped to a data cube. Similarly, other resources represented in QB are mapped to multidimensional elements.

Based on the OLAP-to-SPARQL Algorithm (Chapter 6), analytical queries in the form of a nested set of OLAP operations are translated to SPARQL queries. Also, equivalence mappings are evaluated according to OWL semantics.

RDF Aggregate Views (Chapter 7) can be used to optimise query execution and Complex Correspondences (Chapter 8) to describe complex mappings between data cubes and to increase the number of answers from the global cube.

9.1.2 Implementation of OLAP4LD

We have published an implementation of OLAP4LD as an Open Source software for building analysis applications over statistics published as Linked Data.

The OLAP4LD source code as well as example data is hosted on GitHub[1]. In the following, we describe the implementation specific aspects of OLAP4LD as illustrated in Figure 9.1. A more detailed documentation about OLAP4LD can be found on the project website of the "Linked Data Cubes" project[2].

OLAP4LD implements the Open Java API for OLAP *olap4j*[3], a programming library and standard interface between OLAP front-ends and backends; therefore, any OLAP clients supporting olap4j such a Saiku and JPivot can be used as a GUI to OLAP4LD.

9.1.2.1 Metadata Query Execution in OLAP4LD

In the following, we first describe the details of Metadata Queries over OLAP4LD, i.e., queries for metadata elements from our common Multidimensional Data Model (MDM).

Clients access metadata from OLAP4LD with methods such as `getCubes(...)` and `getMeasures(...)`. Every olap4j driver exposes metadata in two different ways[4]: metadata objects (e.g., Cube) with methods to query for further metadata objects from a Java object (e.g., `getHierarchies(...)`) and methods of classes implementing the `OlapDatabaseMetaData` interface returning JDBC schema result sets (e.g., `getCubes(...)`). Objects of `ResultSet` contain all metadata about a multidimensional element, e.g., the name of a cube[5].

The Linked Data Cubes Component of OLAP4LD exposes metadata as follows: as SPARQL SELECT query results parsed by the NxParser library[6] to lists of arrays of RDF terms (`List<Node[]>`) from an instance of a `LinkedDataCubesEngine` interface. The metadata methods are adopted from the `OlapDatabaseMetaData` interface of olap4j; also, the schema of returned result sets, i.e., the name and types of columns, is adopted from the olap4j result set schema[7].

The metadata methods of OLAP4LD have as input an object of class `Restrictions`. Restrictions are adopted from the metadata signature of olap4j metadata methods and define filter criteria over metadata queries. For instance, by setting the `cubeName` of the `getCubes(...)` metadata query, one can query for a specific cube.

[1] https://github.com/bkaempgen/olap4ld, last accessed on 2015-05-02.
[2] http://www.linked-data-cubes.org/index.php/OLAP4LD, last accessed on 2014-11-05.
[3] http://www.olap4j.org/, last accessed 2014-11-05.
[4] http://www.olap4j.org/olap4j_fs.html#Metadata, last accessed on 2014-11-05.
[5] For the columns returned by each rowset of a specific type of multidimensional element see the olap4j specification http://www.olap4j.org/olap4j_fs.html# The_OlapDatabaseMetaData_interface_and_methods_which_return_schema_rowsets, last accessed 2014-11-05.
[6] http://code.google.com/p/nxparser/, parses RDF and SPARQL results in the Nx format, last accessed 2014-12-15.
[7] http://www.linked-data-cubes.org/index.php/Olap4ld_Metadata, last accessed 2014-11-05.

Therefore, the olap4j Driver Component has to translate olap4j metadata queries to metadata queries on an interface `LinkedDataCubesEngine`. Vice-versa, the olap4j Driver Component translates `List<Node[]>` to `ResultSet` or Metadata objects understandable by the client.

See the following list for common queries for *multidimensional elements* from datasets, adopted from olap4j. For instance, the first metadata query type, `getCubes(...)`, has as input a string for the name of a data cube (`nameDC`) – possibly containing wildcards – and as output a set of Data Cubes whose names match the string.

GetCubes is defined as $GetCubes : nameDC \in String \rightarrow 2^{DataCube}$

GetMeasures is defined as $GetMeasures : nameDC \in String \times nameMS \in String \rightarrow 2^{Measure}$

GetDimensions is defined as $GetDimensions : nameDC \in String \times nameD \in String \rightarrow 2^{Dimension}$

GetHierarchies is defined as $GetHierarchies : nameDC \in String \times nameD \times nameH \in String \rightarrow 2^{Hierarchy}$

GetLevels is defined as $GetLevels : nameDC \in String \times nameD \in String \times nameH \in String \times nameL \in String \rightarrow 2^{Level}$

GetMembers is defined as $GetMembers : nameDC \in String \times nameD \in String \times nameH \in String \times nameL \in String \times nameM \in String \rightarrow 2^{Member}$

We have implemented two example Linked Data Cubes Engines. One, the *EmbeddedSesameEngine*, uses an embedded Sesame triple store for query processing. The other, the *OpenVirtuosoEngine*, can access the SPARQL endpoint of a pre-filled Open Virtuoso triple stores for query processing.

Our Linked Data Cubes Engines implement our MDM-QB Mapping using SPARQL templates. For instance, see Listing 20 for the SPARQL query template for data cubes[8].

Listing 20: Example SPARQL query template to query for metadata of data cubes, used by OLAP4LD.

```
1 SELECT DISTINCT ?CATALOG_NAME ?SCHEMA_NAME ?CUBE_NAME ?CUBE_TYPE
      ?CUBE_CAPTION ?DESCRIPTION
2 {{{STANDARDFROM}}}
3 WHERE {
4 ?CUBE_NAME a qb:DataSet.
5 OPTIONAL {?CUBE_NAME rdfs:label ?CUBE_CAPTION FILTER (
      lang(?CUBE_CAPTION) = "en")}.
6 OPTIONAL {?CUBE_NAME rdfs:comment ?DESCRIPTION FILTER (
      lang(?DESCRIPTION) = "en" )}
7 BIND('{{{TABLE_CAT}}}' as ?CATALOG_NAME).
```

[8] Also available at https://github.com/bkaempgen/olap4ld/blob/master/OLAP4LD-trunk/query_templates/sesame_getCubes_regular.txt, last accessed on 2014-11-13.

```
8  BIND('{{{TABLE_SCHEM}}}' as ?SCHEMA_NAME).
9  BIND('CUBE' as ?CUBE_TYPE).
10 {{{FILTERS}}}
11 } ORDER BY ?CATALOG_NAME ?SCHEMA_NAME ?CUBE_NAME ?CUBE_TYPE
       ?CUBE_CAPTION ?DESCRIPTION
```

The olap4j-specifically-required properties for cubes such as a catalog, schema, cube type are hard-coded. For additional information about cubes such as a human-readable caption and a description pendants from common Linked Data vocabularies, e.g., `rdfs:label` and `rdfs:comment` are selected. The template is parameterised with {{{`FILTERS`}}} that allow to lookup cubes with certain properties.

For each multidimensional element, there may be several SPARQL templates for different ways of modelling, e.g., there are different ways to model hierarchies and measures can either define their own aggregation functions or SUM, AVG, and COUNT are used by default.

Where our MDM requires elements not represented in QB, the SPARQL 1.1 BIND function is used to generate such elements using SPARQL. For instance, our MDM uses a special-type Measure Dimension that contains as members all the measures.

9.1.2.2 Analytical Query Execution in OLAP4LD

After having described metadata queries, we describe in more detail the implementation of analytical queries[9].

Analytical queries in olap4j are called Statements and use the MDX query language. MDX statements contain identifiers for multidimensional elements. While validating an MDX query for correct syntax and parsing, the olap4j Driver Component issues metadata queries to retrieve the multidimensional elements mentioned in the query. From the retrieved information, the driver component derives the metadata for a pivot table schema to be populated with summarised values.

More concretely, the olap4j Driver Component initiates a query optimiser by translating the MDX query to a nested set of OLAP operations and sends this logical OLAP operator query plan[10] to the Linked Data Cubes Engine for execution. Vice-versa, the olap4j Driver Component translates the results from the Linked Data Cubes Engine as instances of `List<Node[]>` to tabular structures (ResultSet or CellSet).

The Linked Data Cubes Component receives analytical queries as logical OLAP operator query plans (data structure `LogicalOlapQueryPlan`).

[9] Also documented at http://www.linked-data-cubes.org/index.php/Olap4ld_-_Olap4j_Driver_Component, last accessed 2014-11-07.

[10] http://www.linked-data-cubes.org/index.php/Olap4ld_Query_Optimizer, last accessed on 2014-11-05.

For query execution, a Linked Data Cubes Engine such as EmbeddedSesameEngine implements every operation from our OLAP algebra as an operator that can be executed according to the iterator model; a logical OLAP operator query plan is transformed into a physical OLAP operator query plan (optimisations to the logical query plan can be applied if needed). The physical query plan consists of iterators and is executed to return the answers of the OLAP query. For instance, the OLAP-to-SPARQL Algorithm (Chapter 6) is implemented as an iterator; the OLAP-to-SPARQL iterator assumes data available from an input iterator via a SPARQL engine. Other iterators implement the directed crawling strategy and execute the Convert-Cube and Merge-Cubes operations (Chapter 8).

9.2 SMART Approach

In the SMART Scenario (Section 2.1), innovative tools such as the knowledge management system *Dropedia* and the Web-based sensor database *SMART-DB* are used to improve decision making for Integrated Water Resources Management (IWRM) in the Lower Jordan Valley.

In this scenario the following user requirement exists: Domain experts need to be able to identify, integrate and re-use research data from expert analyses such as the calculation or estimation of a single value, a literature study, and a complex model application.

In the following section, we present our approach, the SMART Knowledge Base (SKB), with the following contributions:

- We formalise the IWRM knowledge and decision support domain in an OWL ontology, reusing the RDF Data Cube Vocabulary for numeric values from sensors as well as research analyses.
- We use a semantic wiki and Linked Data wrapper to allow easy sharing of research data via forms and application-specific data imports.
- We use Linked Data to represent, extract, integrate, and load community-created research and sensor data into a knowledge base for browsing and expressive queries.
- We present consumption tools on top of the knowledge base that allow scientists to explore the numeric data. Applications also make use of formal OWL semantics in Linked Data.

After explaining SKB in detail in the following section, we apply the approach to our scenario in Section 9.2.2. Then, we reflect on the solution (Section 9.2.3), describe related work (Section 9.2.4) and conclude (Section 9.2.5).

9.2.1 SMART Knowledge Base (SKB)

As illustrated in Figure 9.2, the components of the *SMART Knowledge Base* (SKB) roughly can be divided into IWRM relevant data sources and data consumption tools. To execute necessary steps in IWRM decision processes, decision makers and domain experts utilise consumption tools. Consumption tools access data via a SMART triple store that is filled in advance or on-demand with data from the data sources.

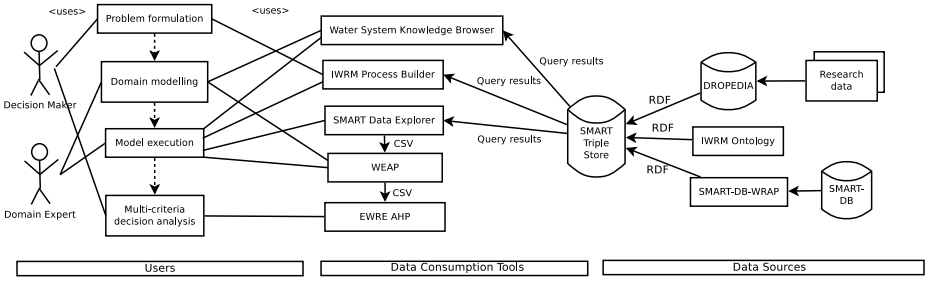


Figure 9.2: Diagram illustrating scenario and architecture of SMART Knowledge Base.

Main *IWRM relevant data sources* are an IWRM ontology, the knowledge management system Dropedia and the SMART-DB for climate sensor data. *Data consumption tools* using data from the data sources include the water system knowledge browser, the IWRM process builder, and the SMART Data Explorer.

In the following, we explain the modelling and mapping of data cubes, as well as the query processing and query optimisation over those data cubes in SKB.

Modelling of data cubes. The IWRM domain is described in an ontology that we describe in the following: See Figure 9.3 for an illustration of the IWRM ontology as graph with concepts and common properties between instances of those concepts.

The illustration only contains the most important concepts and properties of the IWRM ontology. The IWRM ontology is implemented in Dropedia, the collaborative knowledge management system of SMART. As a semantic wiki, Dropedia publishes and supports browsing of the ontology under the `iwrm` namespace[11].

The IWRM ontology models decision processes (`iwrm:Process`) around catchments (`iwrm:Catchment`, areas with certain water resources and consumers). An IWRM process has social, economical, and ecological objectives (`iwrm:Objective`).

Most importantly, the IWRM ontology models data cubes of type `iwrm:Analysis` with facts of type `iwrm:Observation`. For that, the RDF Data Cube Vocabulary is used. Every data cube has as dimensions the location (`iwrm:has_location`),

[11] http://dropedia.iwrm-smart2.org/index.php/Special:URIResolver/, last accessed 2014-08-27.

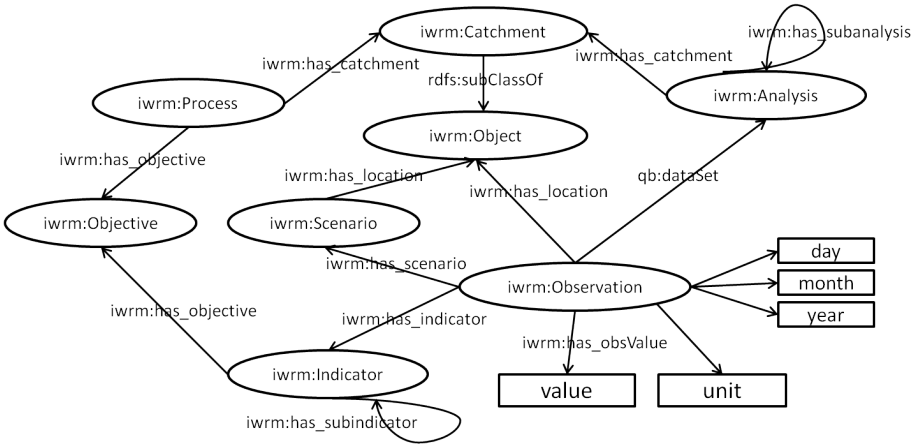


Figure 9.3: Illustration of IWRM ontology as graph with concepts and common properties between instances of concepts.

the valid time (consisting of a hierarchy of day, month, and year), the unit, the IWRM scenario, and the indicator. Every fact in a data cube contains a value for each of the dimensions as well as a numeric value for the measure `iwrm:has_obsValue`.

The ontology is capable of storing micro- and macro-data. For instance, an amount of 125 in m^3/h of monthly discharged water at Shorea Spring on October 1st 1983 in a Status Quo IWRM scenario can be represented. Also, an estimated average annual value of 143.8 in m^3/h of monthly dicharged water at Shorea Spring for 2010 can be represented.

Thus, every analysis data cube quantifies for a certain catchment (via `iwrm:has_catchment`), how well a certain objective is reached by an IWRM scenario according to an indicator (via `iwrm:has_objective`.

If populated, the links between individuals in the ontology can be browsed in a follow-your-nose fashion similar to common Web browsing. Also, the link identifiers provide the schema for querying the data using SPARQL.

In the following, we describe how the IWRM ontology can be populated, i.e., how concrete instances of concepts in the ontology can be created.

Dropedia is used to populate the IWRM ontology with semi-structured information about IWRM processes, objectives, catchments and other locations, IWRM scenarios, indicators, and units. Also, IWRM analyses such as from research publications and experiments can be described and structured information in the form of estimated and simulated observations added. For every observation in an analysis, a value for the location, the valid time, the unit, the IWRM scenario, and the indicator can be given.

Dropedia is based on the Open Source semantic wiki software, Semantic MediaWiki[12], which combines ease of use and collaboration functionalities of the well-known MediaWiki software with flexible support to capture and use structured information via Semantic Web technologies. Users can fill in forms for important IWRM concepts such as catchments, scenarios, and indicators. The data is then automatically made available as Linked Data, e.g., about `dropedia:Wadi_Shueib` under the `dropedia` namespace[13]. Structured information can be queried directly within Dropedia and visualised, e.g., as tables.

Also, forms are available to add single numeric facts to analyses; for that "subobjects"[14] are used. Subobjects allow to instantiate and describe new objects (instances) within wiki pages without creating a separate wiki page for each object.

We manually import[15] ontologies such as the RDF Data Cube Vocabulary to Dropedia so that URIs are reused in the RDF export and less integration work has to be done when querying over numeric values in Dropedia and other data sources such as SMART-DB.

SMART-DB is used to automatically populate the IWRM ontology with sensor data. Sensor data are inserted to SMART-DB via application-specific input forms that SMART partner use if they want to share sensor data with other partners.

We create a wrapper that publishes SMART-DB data as Statistical Linked Data. SMART-DB-WRAP re-publishes sensor data from SMART-DB in the `smart-db` namespace[16]. For that, SMART-DB-WRAP is based on a Google App Engine and on-the-fly translates data from SMART-DB into an RDF representation using the IWRM ontology. For that, a Web interface, HYDROSMART, serving sensor data from SMART-DB as XML was developed by SMART project partner "Helmholtz-Zentrum für Umweltforschung GmbH" (UFZ).

Besides a list of locations, e.g., "AM0530" ("Baqqouria Spring" in Dropedia) and a list of indicators, e.g., "Q" ("Mean Discharge"), SMART-DB-WRAP publishes for each location and indicator a data cube with respective observations. Such data cubes are modelled according to analyses in the IWRM ontology and have as dimensions the location, the valid time, the unit, the IWRM scenario, and the indicator.

Table 9.1 shows example objects and their URIs identifying those entities in Linked Data (abusing CURIE syntax).

[12] http://semantic-mediawiki.org/, last accessed on 2014-10-22.
[13] http://dropedia.iwrm-smart2.org/index.php/Special:URIResolver/, last accessed 2014-08-27.
[14] http://semantic-mediawiki.org/wiki/Help:Adding_subobjects, last accessed 2014-11-17.
[15] https://semantic-mediawiki.org/wiki/Help:Import_vocabulary, last accessed 2014-11-04.
[16] http://smartdbwrap.appspot.com/, last accessed 2014-08-27.
[17] http://www2.ufz.de/smarthydro/smartquery?location_data=AM0528, last accessed 2014-11-04.

Table 9.1: Example objects and their URIs from certain data sources in SMART.

Object	URI
Wadi Shueib as referred to in Dropedia	`dropedia:Wadi_Shueib`
Shorea Spring as referred to in Dropedia	`dropedia:Shorea_Spring`
Shorea Spring as referred to in SMART-DB[17]	`smart-db:/id/location/AM0528`
Average Discharge in Dropedia	`dropedia:Annual_Average_Discharge`
Average Discharge from SMART-DB	`smart-db:/id/analysisobject/Q`
Dataset of locations from SMART-DB	`smart-db:/id/location/ds`
Dataset of indicators from SMART-DB	`smart-db:/id/analysisobject/ds`
Dataset of Mean Discharge for Shorea Spring from SMART-DB	`smart-db:/id/locationdataset/AM0528/Q`

All instances of `iwrm:Observation` from datasets in Dropedia and SMART-DB are integrated in one single multi-cube `dropedia:SMART-DB-DSD` with location, IWRM scenario, date, indicator (analysis object) and unit as dimensions and AVG and COUNT of `smart:obsValue` as measures.

Mapping of data cubes. There is the challenge that SMART-DB and Dropedia use different identifiers for the same objects, e.g., "AM0530" vs. "Baqqouria Spring" and "Q" vs. "Mean Discharge". Integration allows queries over both data sources simultaneously considering identical elements. For every location described in Dropedia, we insert the respective identifier from SMART-DB via forms. From an Excel sheet provided by project partner UFZ, we manually copy identifiers from SMART-DB. Templates in Dropedia then create equivalence mappings – `owl:sameAs` links – between location URIs in Dropedia and location URIs in SMART-DB.

Query processing over data cubes. For query processing, we automatically and regularly fill a triple store with up-to-date relevant data for multidimensional datasets from the data sources using *LDSpider*. LDSpider starts with a seed list of locations in Dropedia. Via above mentioned `owl:sameAs` links, LDSpider reaches the same locations in SMART-DB. Crawled data is then inserted in the triple store. We selected the Open Virtuoso triple store that not only provides a SPARQL 1.1 endpoint for expressive queries (e.g., aggregations), but also is able to evaluate our equivalence mappings (`owl:sameAs` links). Then, queries for one identifier also returns answers for every other possible identifier; thus, Dropedia and SMART-DB are integrated. The goal is not to permanently duplicate information from SMART-DB and other data sources, but to provide unified access and analysis capabilities over selected data.

The following *data consumption tools* access data from the triple store.

The **Water System Knowledge Browser** is implemented as a set of pages in Dropedia with which users get overviews of the knowledge base and can visit catchments, water resources, demand sites and many other aspects of the water domain in the Lower Jordan Valley. The SPARK extension[18] allows for embedding SPARQL queries to the

[18] http://www.mediawiki.org/wiki/Extension:Spark, last accessed 2014-11-04.

triple store in Dropedia pages; SPARQL queries are issued upon a visit to a respective page and results are shown in tables and diagrams using JavaScript, e.g., the number of analyses for a specific catchment.

The **IWRM Process Builder** allows to find existing IWRM decision processes in the Lower Jordan Valley on regional and local scales; in every IWRM decision process, decision makers define objectives for an IWRM problem, scientists create a domain model, e.g., define locations, indicators and IWRM scenarios, and further investigate the model in analyses.

The **SMART Data Explorer** provides an exploratory interface to analyse numeric data from the `dropedia:SMART-DB-DSD` multi-cube. The SMART Data Explorer uses the OLAP client Saiku[19] as a front-end embedded into a Dropedia page. Saiku allows domain experts to issue OLAP operations such as slice and dice in an intuitive interface. The SMART Data Explorer uses OLAP4LD as a backend. OLAP4LD translates OLAP queries from Saiku to SPARQL queries to the SMART triple store.

For that, we use our Linked Data Cubes Engine for the Open Virtuoso triple store (`OpenVirtuosoEngine`[20]). The engine implements the MDM-QB Mapping using SPARQL templates and the OLAP-to-SPARQL Algorithm. Excel or CSV exports of indicator values from Saiku can be imported to other tools such as WEAP.

Query optimisation. No specific query optimisation methods are applied.

In the next section, we apply the SMART Knowledge Base to a concrete IWRM decision process.

9.2.2 An IWRM Process for Wadi Shueib in Jordan

In this section, we analyse the possible benefit of using the SMART approach for an IWRM process in Wadi Shueib, Jordan [Rie13]. From a SMART Knowledge Base start page[21], data sources and consumption tools can be visited. Also, information about the implementation and case study are given.

The SMART Triple Store is based on Open Virtuoso version 06.01.3127 and runs on an AMD Athlon(tm) 64 Processor 3000+ with 2G memory with Ubuntu Linux. Crawling on average takes less than 60min. At the time of writing, 6 IWRM processes, 7 objectives, 22 indicators, 22 IWRM scenarios and 27 analyses are described in Dropedia.

[19] http://www.analytical-labs.com/, last accessed 2014-11-05.

[20] https://github.com/bkaempgen/olap4ld/blob/master/OLAP4LD-trunk/src/org/olap4j/driver/olap4ld/linkeddata/OpenVirtuosoEngine.java, last accessed on 2014-10-22.

[21] http://dropedia.iwrm-smart2.org/index.php/SMART_Knowledge_Base, last accessed 2014-08-27.

Problem formulation: In the case study [Rie13], the "Wadi Shueib IWRM Decision Process" (`dropedia:Wadi_Shueib_IWRM_Analysis`) is motivated by the National Water Strategy of Jordan[22]. The water strategy defines objectives, e.g., `dropedia:Increase_volume_of_captured_and_treated_wastewater`.

Domain modelling: Basic provenance information such as the analysis area (`dropedia:Wadi_Shueib`) and the authors of the process are given. The analysis area already is further described, e.g., by the exact geo-spatial and political origin, important buildings within the area, and synonymous area names. The domain expert selects indicators from discussions with other domain experts. For instance, to evaluate the increased volume of captured and treated waste water, the "Municipal Waste Water Treatment Ratio" relates the assumed volume of total waste water produced with the amount of municipal waste water treated in centralised and decentralised treatment facilities.

Also, the domain expert defines or reuses IWRM scenarios, e.g., "Wadi Shueib Business as Usual (BAU)", the water strategy implementation according to the current plans of the Jordanian national water strategy. This scenario includes the reduction of physical and administrative supply network losses and a sewer rehabilitation and connection program in As-Salt. Whereas in the BAU scenario, further implementation of the water strategy either is regarded as not feasible until 2025 or is hampered by slow political decision making, the "Full Implementation (FI)" scenario assumes that all obstacles are overcome and the full range of stated implementation approaches is realised.

Model execution: Based on the assumptions in the domain model, the domain expert selects suitable analyses or creates own analyses leading to the computation of required indicators for the given planning IWRM scenarios. To find expert analyses the domain expert can use the keyword search functionality of Dropedia. Also, analyses are linked to knowledge objects and can be browsed in the `dropedia:Water_System_Knowledge_Browser` and the `dropedia:IWRM_Process_Builder`. SMART-DB identifies Shorea Spring with "AM0528" and provides for example water discharge numbers from 1973 to 2006; see Figure 9.4 for a screenshot. The particularly high discharge value of around $687 m^3/h$ in 1992 indicates an incorrect sensor record in the SMART-DB.

In addition, based on data records from SMART-DB of Shorea Spring from 1995 to 2005 the domain expert computes an overall average annual discharge as an estimation for future years; see Figure 9.5 for a screenshot of the values documented in Dropedia.

The domain expert uses the SMART Data Explorer to explore, compare and analyse numeric assumptions from other analyses. For instance, the decision maker can ask for the average annual discharge for Shorea Spring over time, see Figure 9.6. A pivot

[22] http://www.joriew.eu/uploads/private/joriew_org_jordan_national_water_strategy.pdf, last accessed 2014-11-17.

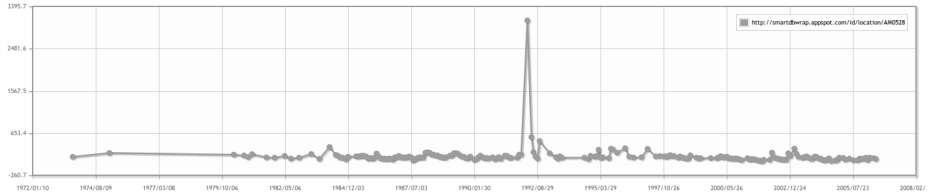


Figure 9.4: Screenshot of line chart from Dropedia with overview of water discharge records in m^3/h of Shorea Spring over years from SMART-DB.

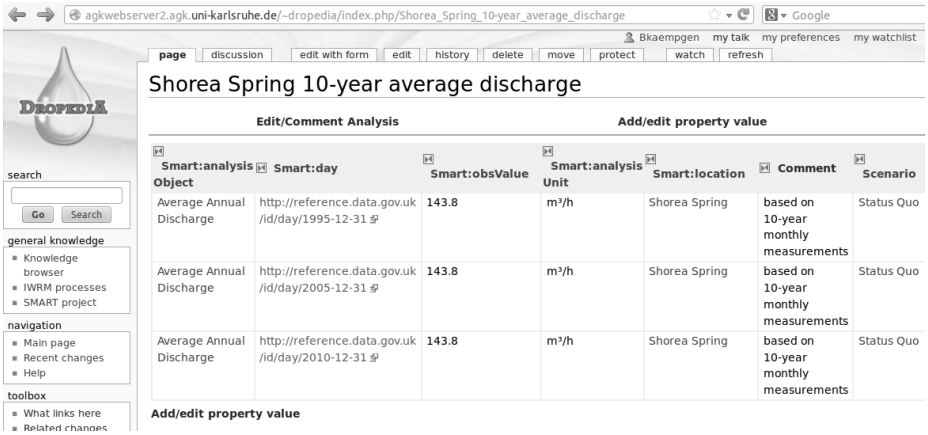
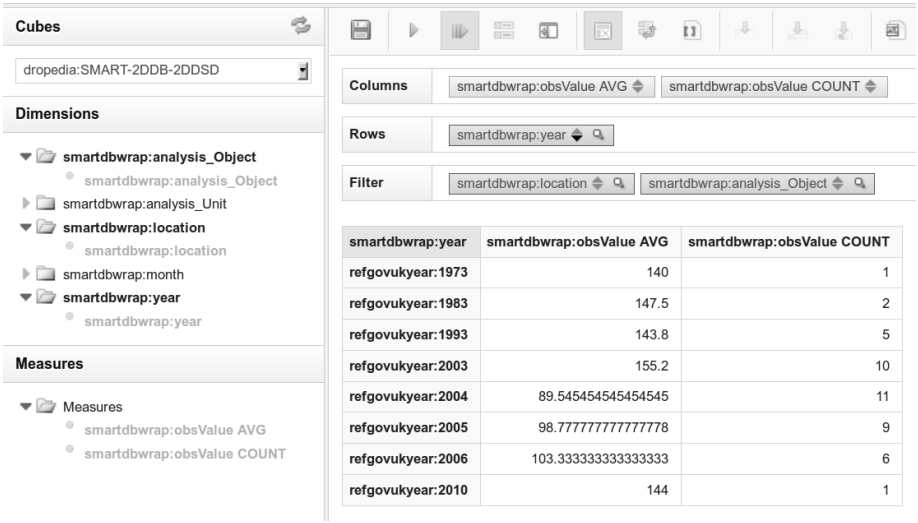


Figure 9.5: Description of expert analysis about Shorea Spring 10-year average discharge in Dropedia; discharge values are estimated to remain constant based on measured value for earlier years.

table is queried with the average value and the number of values (on columns) for specific years (on rows; first ten year steps, then yearly). Besides filtering for specific years, values are filtered for location Shorea Spring and analysis object water discharge (filter); we see that for some years there are more values than for others.

Also, we see that Dropedia and SMART-DB data are integrated: On the one hand values for 1973 and 1983, for instance, are taken from SMART-DB. Such values we saw in the overview of actual sensor values from SMART-DB in Figure 9.4. On the other hand, the values for 2005 and 2010, for instance, have their origin in Dropedia. Such values are documented in Dropedia as displayed in Figure 9.5 (the value for 2005 is contained in an aggregation). Having values represented as observations is not sufficient for integration in this case since both “Shorea Spring” and “Mean Discharge” are identified differently in SMART-DB and Dropedia. For that, equivalence mappings are manually added and automatically resolved by the SMART Data Explorer.

Loading time for Saiku may take several minutes due to queries with a large number of results to be displayed in the interface.



smartdbwrap:year	smartdbwrap:obsValue AVG	smartdbwrap:obsValue COUNT
refgovukyear:1973	140	1
refgovukyear:1983	147.5	2
refgovukyear:1993	143.8	5
refgovukyear:2003	155.2	10
refgovukyear:2004	89.545454545454545	11
refgovukyear:2005	98.777777777777778	9
refgovukyear:2006	103.333333333333333	6
refgovukyear:2010	144	1

Figure 9.6: Pivot table showing average and count of water discharge records in Shorea Spring for specific years from both Dropedia and SMART-DB.

Multi-criteria decision analysis: A tabular overview of all estimated indicator values for IWRM scenarios can be given on the analysis page. Provenance information about any single observation can be browsed by clicking on the value in the table. The decision maker exports such data directly from Dropedia or with the SMART Data Explorer. Table 2.1 from the SMART Scenario section illustrated the decision matrix for the Wadi Shueib Process comparing different IWRM scenarios projected to 2025. Indicator values are normalised between 0 and 1; a higher score means a better performance. The FI-alternative shows the best performance for most of the indicators; only regarding surface water, cost effectiveness (unit cost of supply in Jordanian dinar per cubic meter), and environmental water stress (through fluctuations) other scenarios are evaluated higher. Given weights for indicators, the multi-criteria decision analysis tool EWRE-AHP will recommend one IWRM scenario for Wadi Shueib over the other.

9.2.3 Discussions and Lessons Learned

In this section, we discuss the results from applying the SMART approach to the Wadi Shueib IWRM process.

Although only few tasks from the IWRM process for Wadi Shueib were repeated using the SMART Knowledge Base (SKB) and although the problem and IWRM scenario

definitions as well as the modelling and simulation tasks can be arbitrarily complex, SKB has shown potential to support researchers and decision makers in sharing relevant IWRM information (**SMART User Requirement**) as follows:

The IWRM ontology allows to represent both numeric values from sensors as well as research analyses. Users can share numeric values with the entire IWRM community for citations via forms and database import GUIs. Values also can be discussed with respect to objectives, indicators, and scenarios in the IWRM Process Builder. Applications such as the SMART Data Explorer can be developed on top of published information for innovative usages; interoperability between Dropedia and the SMART-DB was demonstrated in visualisations showing data from both sources.

The SKB approach benefits from Semantic Web concepts through semantics, e.g., equivalence statements using `owl:sameAs` and extensibility, e.g., new data sources can easily be added to the SMART Knowledge Base by adding new links that LDSpider would follow. If data loaded to the triple store reuse the same vocabularies such as our IWRM ontology or the RDF Data Cube Vocabulary, data may show up in existing visualisations without additional effort.

In particular, we see potential regarding the ongoing "Open Data" trend. More and more institutions such as FigShare, DataCite and Pangaea help scientists to not only publish their analysis results but also the raw (or pre-processed) data for citations, reproduction and further analysis. If also published as Linked Data – for instance using a wrapper approach as for SMART-DB-WRAP – interoperability of the data contained in such silos can be improved.

Two main areas of possible improvements were identified:

Usability and Training. It proved a challenge to make scientists and decision makers share research-relevant data. There may be many reasons for this behaviour, yet, it is clear that stakeholders are especially reluctant if interfaces to data sharing platforms are not familiar and if there is no clear personal benefit. The flexibility of a semantic wiki and an embedded OLAP client do not reach the usability of commercial products, in particular, Microsoft Excel and widely-used E-Mail clients. SKB intends to provide benefits directly to research data providers, e.g., by visualisations and integration with SMART-DB data; yet, most benefit will be achieved if there is a culture of two-way sharing and reusing of research data.

To support this aim, regular tutorials with specific cross-group analysis objectives seem necessary. SMART members will get familiar with and continuously insert new information to SKB. Also the benefits of operational guidelines and a way to make transparent SMART research results for IWRM will become clearer.

Complex Modelling. An initial working hypothesis stated that Semantic Web ontologies allow semi-automated IWRM analysis. The water simulation tool WEAP provides a complex model of inter-dependent indicators in a system of water resources, demand

sites and operational network elements connected by flow vectors of various types. WEAP provides algorithms to compute indicators. However, ontologies such as OWL, RDFS and Linked Data vocabularies have difficulties to represent and do reasoning over such mathematical relationships [VLH+10].

Although we were able to represent and share measurements of environmental indicators using the RDF Data Cube Vocabulary, estimating indicators still requires mostly manual effort, e.g., copy-paste from publications and spread sheet-processing in Excel. A more automatic computation of indicators will require to formalise relationships between (collections of) measurements. For instance, from assumed volumes of waste water produced in single municipalities, a total waste water discharge assumption for an area could automatically be aggregated and be re-used for a Municipal Waste Water Treatment Ratio computation. Conversion Correspondences (Chapter 8) may be a possible solution to represent and use such relationships between datasets; however, so far we do not know from where to retrieve such relationships

9.2.4 Related Work

The German-Vietnamese water-related information system for the Mekong Delta (WISDOM) project provides a web-based information system [GWK+10]. The system is based on PostgreSQL for geographical data management. Services are provided via representational state transfer (REST) and as such identify and allow access to resources similar to the Linked Data principles. However, it is not clear whether this setup of using REST services also improves interoperability between systems for efficient integration of WISDOM data sources with third-party data sources. Different from the WISDOM information system, the SKB concept focuses on data integration and making available data for third-party usage.

In research and industry, wikis are widely perceived as potent knowledge management instruments. And also in hydrology, some organizations have recently started initiatives of which probably the most visible examples are the UNDP-initiated WaterWiki[23] and the IWAWaterwiki[24]. Different from Dropedia, such platforms do not generate self-descriptive RDF to build applications on top of structured information.

The CUAHSI Water Data Center[25] provides data services to communities that require access to various sources of water data to perform research. Their software stack provides tools to publish hydrologic datasets with web services as well as a metadata catalog to discover and client tools to analyse published datasets. They allow tagging of

[23] http://waterwiki.net, last retrieved on 2014-03-14
[24] http://www.iwawaterwiki.org, last retrieved on 2014-03-14
[25] http://wdc.cuahsi.org/, last retrieved on 2014-04-16

variables with the CUAHSI HIS Ontology describing concepts such as chemical, biological and physical variables. They do not use RDF and as such are limited to data complying to a fixed relational model for observation data.

The Semantic Ecology and Environmental Portal[26] integrates water data from different authoritative sources using Linked Data to enable pollution detection and monitoring. Their interface is able to display both geo-spatial and measurement data, but does not support collaboration on analyses as possible in Dropedia.

Wiljes and Cimiano [WC12] use Linked Data to publish research results in the natural sciences. To make scientists less reluctant to share research data, a scientific data curator helps with the Linked Data publication process. In our work, we first allow users to make research data available as Linked Data without any Linked Data specificities (through Dropedia and SMART-DB). Second, we have a stronger publishing argument since we describe possible applications (Dropedia/SPARK, SMART Data Explorer) on top of published data.

9.2.5 Conclusions

We have presented the SMART Knowledge Base (SKB) approach for Integrated Water Resources Management (IWRM) in the Lower Jordan Valley.

SKB formalises the IWRM decision process using an OWL ontology, integrates research data from a semantic wiki with climate sensor records from a relational database, and allows exploring and analysing IWRM data using browsing and OLAP.

We have applied the knowledge base in an IWRM decision process for the Wadi Shueib region in Jordan. IWRM processes can be arbitrarily complex, but for simple tasks SKB showed capable to improve interoperability between systems to help researchers and decision makers in sharing and consuming IWRM-relevant information, including numeric values such as from sensors and intermediary research.

Lessons learned promise to easily connect further data sources available on the Web using Linked Data, but demand a more systematic training of potential users for quantifiable improvements in the domain, and a more formal representation of indicators and IWRM scenarios for semi-automatic IWRM analysis.

9.3 XBRL Approach

After previously having introduced the XBRL scenario (Section 2.2), we present an approach that based on Linked Data fulfils the following user requirements:

[26] http://tw.rpi.edu/web/project/SemantEco, last retrieved on 2014-04-16

- User Requirement 1: Business analysts need to integrate data across sources.
- User Requirement 2: Analysts need to analyse data in a fashion of "overview first, zoom-in, details on demand".
- User Requirement 3: Analysts need to create their own analyses with Excel-like functionality.

Literature has proposed the use of Semantic Web technologies, but has not evaluated the benefit in financial case studies [WTB11, GG10, BRLD10].

We present a solution that uses Linked Data to identify, gather and integrate financial data and we apply interfaces for analysing such integrated financial Linked Data in a realistic scenario. In the next section, we present this Financial Information Observation System (FIOS) with the following contributions (Section 9.3.1):

1. For standardised data access, FIOS models XBRL and non-XBRL as Linked Data using the RDF Data Cube Vocabulary and other standard vocabularies.
2. FIOS integrates financial data using entity consolidation for background information, multi-company KPI, and cross-data-sources KPI analysis.
3. For intuitive and exploratory analyses, FIOS provides SPARQL templates with visualisations, a Linked Data browser and a self-serve OLAP interface on top of a triple store.

For evaluation, we describe a case study implementing and applying FIOS for financial analysis (Section 9.3.2) and derive lessons learned (Section 9.3.3). We describe related work in Section 9.3.4 and conclude in Section 9.3.5.

9.3.1 Financial Information Observation System (FIOS)

In this section, we describe our Linked-Data-based XBRL approach. As illustrated in Figure 9.7, the architecture of the *Financial Information Observation System* (FIOS) is separated into two types of components, the offline ETL and online analysis components.

The *offline ETL components* make financial information available as Linked Data and crawl, pre-process, and load such data to an RDF triple store. The *online analysis components* provide three different interfaces over financial data in the triple store.

In the following, we explain the modelling and mapping of data cubes, as well as the query processing and query optimisation over those data cubes done in the ETL or analysis components of FIOS.

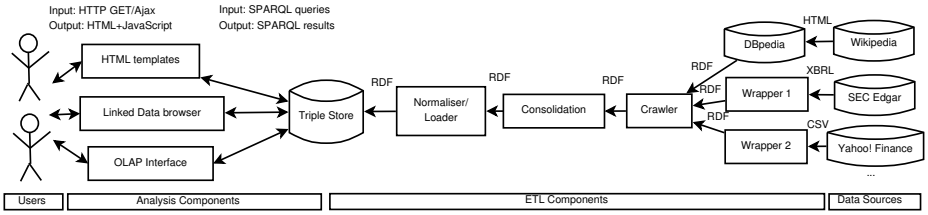


Figure 9.7: Flow diagram illustrating architecture of Financial Information Observation System (FIOS).

9.3.1.1 Modelling of Data Cubes

FIOS uses the Linked Data principles to identify and retrieve relevant information spread across different web servers as described in the following.

Identification of relevant finance data from the Web. We uniquely name things/entities with URIs: XBRL balance sheets from the SEC Edgar Database, including their taxonomies, and daily stock quotes from the Yahoo! Finance Web API; companies and industries listed by the SEC, Yahoo! Finance and Wikipedia/DBpedia. The SEC uniquely identifies the companies using a Central Index Key (CIK, e.g., Mastercard has "1141391"); Yahoo! Finance uses Ticker symbols (e.g., "MA" for Mastercard); and Wikipedia uses their own non-standardised identifiers that are typically based on the name of the company, e.g., "Mastercard".

If looked up, URIs provide useful information in RDF either by originating from Linked Data providers or created by wrappers around data sources not publishing Linked Data. Wrappers mint (introduce, create) new URIs and internally transform available information about such entities in the data source to RDF. Since the actual URIs are application-specific, in the following, we simply abbreviate URIs from our selected data sources using intuitive namespaces (abusing CURIE syntax): `edgar` for entities from the SEC Edgar Database, `yahoo` for Yahoo! Finance Web API and `dbpedia` for Wikipedia. Table 9.2 shows example mappings between things/entities, data sources with useful information about these entities and URI identifying those entities in Linked Data.

For a holistic view on selected companies from the SEC Edgar Database, FIOS defines as relevant data: data from resolving the URIs of the companies and of entities linked

[27] http://www.sec.gov/Archives/edgar/data/1141391/000119312511207804/0001193125-11-207804-xbrl.zip, last accessed 2014-11-07.

[28] http://ichart.yahoo.com/table.csv?s=MA&a=11&b=01&c=2010&d=11&e=01&f=2010&g=d&ignore=.csv, last accessed 2014-11-07.

from the companies, as well as starting from QB datasets the relevant data for each multidimensional dataset (Definition 7).

Table 9.2: Example mappings between things/entities, data sources and URIs in XBRL approach.

Entity	Original data source	URI
Company Mastercard	Mastercard from DBpedia	`dbpedia:Mastercard`
Company Mastercard	SEC Edgar company Mastercard with CIK 1141391	`edgar:cik/1141391#id`
Company Mastercard	Yahoo! Finance company Mastercard with Ticker MA	`yahoo:ticker/MA#id`
Balance sheet	XBRL document from SEC Edgar[27]	`edgar:archive/1141391/0001193125-11-207804#ds`
Stock Quotes table	Stock Quotes table from Yahoo! Finance Web API[28]	`yahoo:archive/MA/2010-12-01#ds`

The RDF Data Cube Vocabulary is used to model financial data as data cubes as described in the following.

Modelling of finance data. To allow FIOS to use the retrieved information, we model financial data reusing existing Linked Data vocabularies and link entities from different sources.

Whereas there are well-adopted vocabularies for all kinds of metadata, e.g., SKOS, FOAF, and the DBpedia ontology, there is no standard way to represent XBRL data as Linked Data [GG10, BRLD10, WTB11]. *XBRL* distinguishes instance and taxonomy documents. An *XBRL instance document* (also called "filing") contains financial facts with a numeric value and a unit such as USD. A fact has a context, e.g., describing the issuing company such as Mastercard, the time period of a financial fact (often, a quarter of a year or full fiscal year) and so-called segment information, e.g., allowing to specify subgroups of financial facts, e.g., that facts are published for subsidiary members. Most importantly, a fact specifies a certain disclosed financial concept such as "Total Assets". Financial concepts are taken from *XBRL taxonomy documents*. XBRL taxonomies can be standardised, e.g., the US-GAAP, and their concepts used across many instance documents. Also, companies may create their own taxonomies and financial concepts. Within taxonomies, concepts may be given additional information, e.g., labels, and may have relations to other concepts, e.g., "part of" relationships.

We model every XBRL instance and taxonomy as a multidimensional dataset, i.e., collection of facts with independent dimension variables and dependent measure variables, using the RDF Data Cube Vocabulary (QB) as follows: for any XBRL instance with taxonomy a multidimensional dataset (`qb:DataSet`) and data structure definition (`qb:DataStructureDefinition`) are created. For any single financial fact

within an XBRL instance, an observation (`qb:Observation`) is created with dimensions issuer, time period (`edgar:dtstart`, `edgar:dtend`), the financial concept (`edgar:subject`), segment information (`edgar:segment`), and one decimal measure with a unit.

Similarly, stock quotes from Yahoo! Finance can be modelled using QB: every daily collection of values is a dataset, every stock quote contains as dimensions the company (`yahoo:issuer`), the date the value is valid and the stock quote type such as price at stock market opening (Open).

9.3.1.2 Mapping of Data Cubes

We use QB for the following reason: Given datasets contain observations with certain companies, certain financial concepts and certain periods in time, financial data integration boils down to identifying and consolidating equivalent dimensions and dimension values in multidimensional datasets.

See Figure 9.8 for an illustration of the linking between different entities or properties. Here, the fact of an XBRL instance document disclosing Total Assets (`edgar:vocab/-us-gapp-2009-01-31#Assets`) and an Opening stock quote are linked via equivalent dimensions, e.g., `dcterms:date` / `ical:dtstart` and `edgar:issuer` / `yahoo:issuer`, and via equivalent dimension members, e.g., Mastercard `edgar:-cik/1141391#id` / `yahoo:ticker/MA#id`.

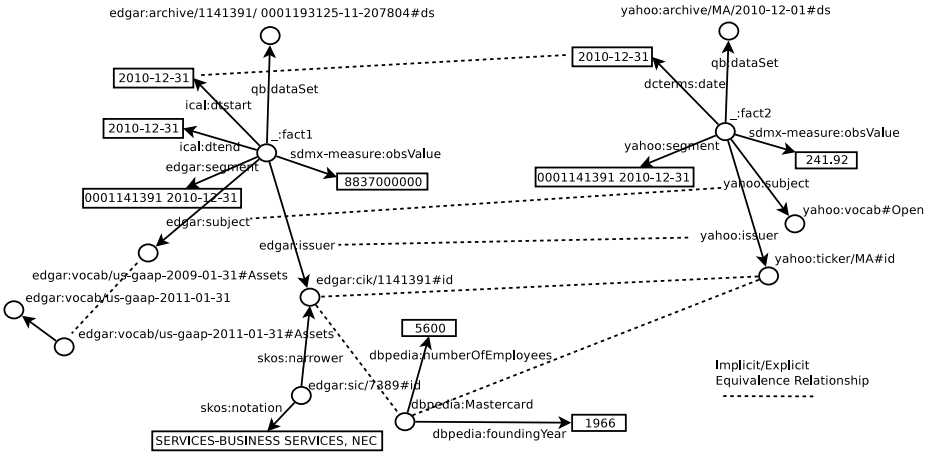


Figure 9.8: Illustration of modelling of finance data in FIOS; most importantly, equivalence relationships exist between Total Asset fact from the SEC (top left), opening stock quote from Yahoo! Finance (top right), and Mastercard in DBpedia (bottom center).

Whereas time periods can easily be matched by comparing canonical representations of time, for linking between different URI for companies and financial concepts across data sources, mappings need to be available. Entities or properties in RDF can explicitly be stated as equivalent via `owl:sameAs` or `owl:equivalentProperty` relationships between their URIs.

To model finance related metadata, e.g., about companies and industries, we use widely-adopted Linked Data vocabularies, e.g., FOAF and the DBpedia ontology. The industry of a company can be represented using SKOS classification hierarchies, e.g., the SEC provides for companies the Standard Industrial Classification (SIC) hierarchy, e.g., SIC concept "SERVICES-BUSINESS SERVICES, NEC" `skos:narrower` "MASTERCARD INC".

If XBRL balance sheets from the SEC and stock quotes from Yahoo! Finance are modelled as data cubes, and their dimensions and dimension values are mapped, their observations can be integrated in a multi-cube "FIOS 2.0 Data Cube for SEC/YHOF".

9.3.1.3 Query Processing Over Data Cubes

For query processing, FIOS pre-processes and stores data in a triple store as described in the following.

Consolidation, normalisation, loading and validation of finance data. FIOS allows to pre-process and store acquired data for fast access as well as to check its quality.

Entity consolidation in FIOS – making explicit and merging all available information about an entity, so as access to that information is available independently from a specific distribution across sources – results in simpler queries and is only different from Hogan et al. [HHU+11] in that we also consider equivalent relationships between predicates such as dimensions of datasets.

Queries on our consolidated data are complicated by the fact that all entities described by FIOS use distributed namespaces. Resources from FIOS thus link to external servers with non-preprocessed data, confusing the user or complicating the application. Therefore, we mint resolvable URIs for all entities, including URIs in the predicate position, in an own FIOS namespace `fios`. For provenance reasons, we create `owl:sameAs` and `owl:equivalentProperty` links from FIOS entities to the original entities.

After pre-processing, data is loaded into a triple store that supports SPARQL 1.1 for analytical aggregate queries and is indexed for performance.

We use SPARQL queries for quality checks, e.g., validating QB integrity constraints or XBRL-specific integrity constraints such as defined between financial concepts in XBRL calculation relationships.

Given relevant finance data pre-processed and loaded to a triple store by the ETL components, the online analysis components can provide three different interfaces over the triple store.

Analysis of integrated financial Linked Data. Semantic Search engines are too general for financial analysis (e.g., [HHU+11]) and data analysis tools such as the SPARQL Package for R[29] are too complicated for domain experts. FIOS uses three different kinds of interfaces for views on financial Linked Data.

SPARQL Templates with Visualisations, i.e., webpages that show results of SPARQL 1.1 queries on the triple store in visualisations, give a general overview of data in FIOS, e.g., number of datasets. Also, we create domain-specific reports about companies that require data integration. Templates can be parameterised with input by the analyst, e.g., a company identifier.

A *Linked Data Browser*, i.e., webpages of things/entities in RDF that show all ingoing and outgoing triples of a resource and support follow-your-nose browsing from resource to resource, provides a more detailed view on any RDF data in FIOS.

An *OLAP Interface*, i.e., an intuitive and explorative data analysis method allows analysts to create own visualisations on multidimensional datasets. Since (parameterised) SPARQL templates have a fixed structure and Linked Data browsing does not aggregate triples, we use our OLAP-to-SPARQL Algorithm (Chapter 6) to evaluate OLAP operations using SPARQL over RDF about QB datasets.

9.3.1.4 Query Optimisation

No specific query optimisation methods are applied.

In the next section, we apply FIOS to our financial data analysis scenario.

9.3.2 Accepting the XBRL Challenge

We successfully submitted implementations of FIOS to the XBRL Challenge 2012[30] and 2013[31]. For evaluation, we first describe the most current implementation of FIOS, then a case study applying FIOS to the company performance analysis scenario.

[29] http://linkedscience.org/tools/sparql-package-for-r/, last accessed on 2014-10-26.

[30] See http://xbrl.us/research/appdev/Pages/275.aspx, including screencast at http://youtu.be/e3XTh54O_5E, last accessed 2014-11-07.

[31] See http://xbrl.us/research/appdev/Pages/423.aspx, including screencast at http://www.youtube.com/watch?v=zLqsQ-YHMvk, last accessed 2014-11-07.

From a FIOS start page[32], analysts get information about the ETL process and an overview of available entities: publishing companies (`fios:issuer` / 64 different values), valid time periods (`ical:dtstart` / 234, `ical:dtend` / 223, and `dcterms:date` / 5,937) financial concepts (`subject` / 3,781), and specific information (`segment` / 58,395). From linked histograms, one sees that most observations are from the time period between 2008 and 2013. Also, it becomes visible that FIOS contains an evenly spread number of observations for each company. Also, analysts get a good understanding of what financial concepts are published very often, e.g., `us-gaap-2009:Revenues`. Both FIOS ETL and analysis components run on a Virtual Machine with QEMU Virtual CPU version 0.12.3 with 2673.330 CPU MHz and 1GB memory and are described in the following:

ETL components. For the `edgar`[33] and `yahoo`[34] namespaces, we use the SEC Edgar Wrapper (Edgarwrap) and the (Yahoofinancewrap[35]) that publish Statistical Linked Data (implemented with the Google App Engine platform). Some information, e.g., XBRL calculation linkbases and footnotes currently are not considered, however, could be extracted and published as Linked Data to provide additional interesting information [OCB+13, BHH+11].

Yahoo companies link to Edgar companies using a Ticker-to-CIK mapping provided by the Yahoo! Finance API. Edgar companies link to DBpedia companies via Freebase. Datasets from SEC and Yahoo! Finance are linked by manually stating the equivalence of dimensions, such as the company, the valid time period and the financial concept. In cases where structures of datasets are less similar, approaches for data warehouse integration could be applied [TC05].

We created a Java program fios-etl[36] containing separate components for crawling data, applying consolidation and normalisation algorithms to the collected data, and loading the data into a triple store. As crawler, we used the Open Source software LDSpider (Stable Version 1.1e).

For each run, fios-etl automatically fills a seed list with pre-selected companies and new balance sheets from where LDSpider starts to crawl. We selected company URIs from several industries, e.g., "finance, insurance and real estate" companies such as Visa and Mastercard. New balance sheet URIs are taken from an SEC RSS feed.

[32] http://fios.linked-data-cubes.org/FIOS_2_0/Queries/, last accessed 2014-11-07.
[33] http://edgarwrap.ontologycentral.com/, last accessed 2014-11-07.
[34] http://yahoofinancewrap.appspot.com/, last accessed 2014-11-17.
[35] Developed within a Master thesis by Tobias Weller [Wel13], co-supervised by the author, available at http://www.aifb.kit.edu/images/4/4b/Masterarbeit_Dominik_Siegele.pdf, last accessed 2014-12-17.
[36] https://code.google.com/p/fios-etl/, mainly developed by Tobias Weller, last accessed 2014-11-07.

For example, LDSpider would start crawling at the URI of Mastercard in Yahoo! Finance Wrapper that provides links to stock quote datasets from 1990-01-01 to today and `owl:sameAs` links to Mastercard in Edgarwrap. From there, further `owl:sameAs` links to Mastercard in DBpedia and links to SEC balance sheets would be followed.

We setup LDSpider to crawl with breadth-first strategy and a depth of the traversal of 3, with a maximum number of 10 URIs crawled per round per pay-level domain. Consolidation and normalisation algorithms we implemented as described for FIOS. Experiments with differently-sized datasets show that consolidation time increases exponentially with the number of equivalence statements, normalisation time increases linearly with the number of triples. Data was then bulk-loaded to an OpenLink Virtuoso Server v06.01.3127 running in Apache/2.2.14.

For our case study, we run fios-etl daily during the XBRL Challenge 2013 submission time from 15 Feb 2013 to 27 Feb 2013 GMT. On average crawling, pre-processing and loading took 25min; loading can be done offline and could further be accelerated using differential loading. In total, we crawled 1,238,041 triples.

Furthermore, we execute a SPARQL CONSTRUCT query within fios-etl to link observations from several integrated datasets (datasets with the same structure) to a new integrated multidimensional dataset "FIOS 2.0 Data Cube for SEC/YHOF" containing both observations from balance sheets and stock quote tables. This pre-processing of creating a new dataset results in a multi-cube allowing to query over values from balance sheets and stock quotes.

Moreover, we created integrity constraints using SPARQL ASK queries that can be manually run, e.g., evaluating whether Earnings per Share for a company in fact is computed by the ratio of net income and outstanding shares. Since we have not found an automatic way of retrieving and validating integrity constraints, we have only implemented few checks.

Analysis components. The SPARQL Templates with Visualisations for overviews and domain-specific reports we implemented using the JavaScript library SPARK[37]. For some templates, especially the company template, users may need to wait several minutes before all results are displayed, due to large number of separately issued SPARQL queries. As Linked Data Browser we deployed the Open Source software Pubby. For the OLAP Interface we use the Open Source OLAP client Saiku and our OLAP engine OLAP4LD. The OLAP Interface shows long loading times due to a large number of multidimensional elements such as financial concepts (3,781) that need to be loaded in memory and displayed to the user.

In the remainder of this section, we show how FIOS fulfils the three user requirements of our XBRL scenario.

[37] http://km.aifb.kit.edu/sites/spark/, last accessed 2014-11-07.

9.3.2.1 Integrating Data Across Sources (XBRL User Requirement 1)

In the following, we describe four exemplary analyses integrating entities across data sources.

1) Background information analysis: The FIOS start page provides a link to analyse companies in a *SPARK company template*. After inserting the CIK for a company in the parameterised template, e.g., "1141391" for MASTERCARD INC, the user is presented with information from various sources, e.g., address and number of employees from Wikipedia, and various overviews of available KPIs from SEC Edgar Database and Yahoo! Finance Web API. Note, since companies from SEC, Yahoo! Finance and DBpedia are explicitly stated as equivalent in RDF and consolidated, we have one identifier for MASTERCARD INC that summarises all information from those data sources; queries do not need to consider equivalent links and thus are easier to write.

2) Multi-company KPI analysis: On the SPARK company template for a company, an overview of "Adjusted Closing Price" over time is given that interactively can be extended with companies from the same industry via the SIC classification as provided by SEC Edgar.

See Figure 9.9 for the adjusted closing price for MASTERCARD INC and other companies in SERVICES-BUSINESS SERVICES, NEC (SIC) industry. We see that MASTERCARD INC stock quotes always have been higher than VISA INC and COMSCORE INC stock quotes and at the beginning of 2013 were at an all-time-high with over 500 USD per share.

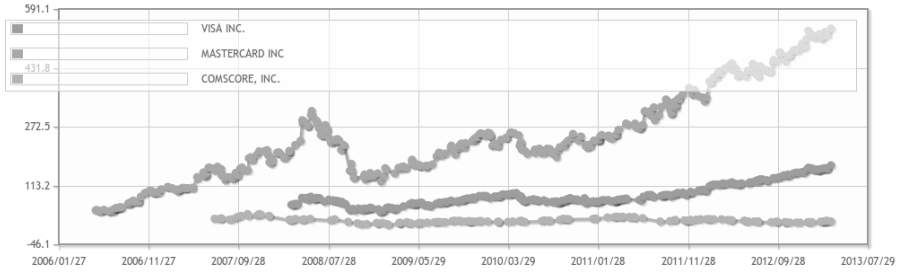


Figure 9.9: Example screenshot in FIOS of multi-company KPI analysis of adjusted closing price for Mastercard (top line), Visa (middle line), and Comscore (bottom line) in industry SERVICES-BUSINESS SERVICES, NEC (SIC).

3) Cross-data-sources KPI analysis: On the SPARK company template we also show an analysis taking into account "Earnings per Share" from SEC balance sheets and the "Opening Price per Share" from Yahoo! Finance stock market data. Earnings per Share

is considered the single most important variable in determining a share's price, thus an analyst may be interested to check for an obvious correlation for a company.

In Figure 9.10, we return for each reporting end date the maximum Earnings per Share as published in quarterly or yearly balance sheets from the SEC together with the maximum opening stock market price of values between the valid start and end data of the Earnings per Share financial ratio for MASTERCARD INC from Yahoo! Finance. Since numbers are not normalised and SPARK visualisations would not allow several separate y-axes, it is difficult to see correlations in the figure; however, the visualisation illustrates the successful integration of values from SEC and Yahoo! Finance. Another interesting analysis is the % rate of increase (decrease) for comparable periods, however, the SPARQL query and visualisations were not easily doable since the Edgar Wrapper does not explicitly represent the sequence of balance sheets.

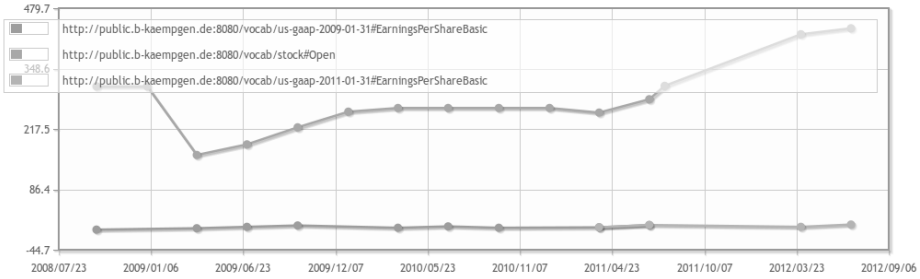


Figure 9.10: Illustration of integrating SEC and Yahoo! Finance data; screenshot from FIOS of example cross-data-sources KPI analysis of Earnings per Share from balance sheets (bottom line) versus price per share (top line) for MASTERCARD INC from stock quotes.

Since balance sheets and stock quote tables are integrated, it suffices to ask for specific values for the financial concept dimension `fios:subject` to query for financial concepts across the SEC Edgar Database and Yahoo! Finance.

As illustrated in Figure 9.11 for MASTERCARD INC, we can also query across different taxonomy versions: If we browse from the SPARK company template to a "Balance Sheet" template and click on "Total Assets", the company's Total Assets KPI over time is shown in a diagram from 2009 to 2012 although balance sheets from 2011 use a different US-GAAP taxonomy version. For that, Total Assets from US-GAAP-2009 and US-GAAP-2011 are stated as equivalent (either by consolidation or by adding UNION graph patterns to SPARQL queries). We see that MASTERCARD INC only twice has reduced its number of assets, at the end of 2009 (from 7.4B USD to 7.3B USD) and the end of 2010 (from 8.8B USD to 8.5B USD), for instance indicating reduced profits and a re-organisation.

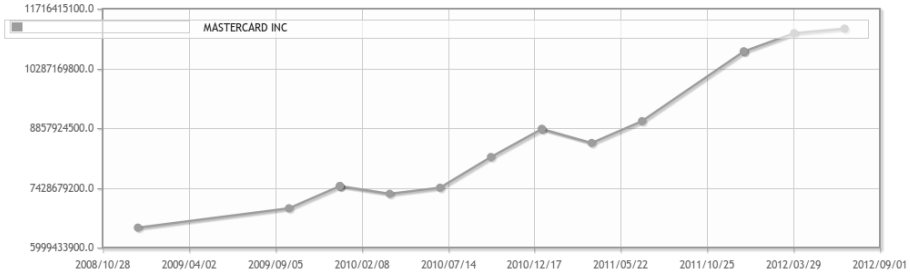


Figure 9.11: Illustration of integrating different taxonomies; example screenshot of cross-taxonomy analysis of Total Assets for Mastercard in FIOS.

9.3.2.2 Overview First, Zoom, Details on Demand (XBRL User Requirement 2)

We demonstrate the capability of FIOS to show the same data using our three different interfaces. As an example, we again visit from the SPARK company template the assets over time for MASTERCARD INC as displayed in Figure 9.11.

From the top of the SPARK company template, via "Pubby Link to Data", we can then start browsing all information related to MASTERCARD INC in the Linked Data Browser Pubby. For instance, we can browse to the balance sheets and there find the single observations visualised in the line chart. For instance, see Figure 9.12 for a screenshot of an observation found in Pubby with the total asset on 2011-12-31.

Anonymous Resource #285

Property	Value
qb:dataSet	<http://public.b-kaempgen.de:8080/pubby/archive/1141391/0001141391-12-000006%23ds>
dcterms:date	2011-12-31 (xsd:date)
?:issuer	<http://public.b-kaempgen.de:8080/pubby/cik/1141391%23id>
sdmx-measure:obsValue	10693000000 (xsd:double)
is rdfs:seeAlso of	<http://public.b-kaempgen.de:8080/pubby/archive/1141391/0001141391-12-000006%23ds>
?:segment	0001141391 2011-12-31
?:subject	<http://public.b-kaempgen.de:8080/pubby/vocab/us-gaap-2011-01-31%23Assets>
rdf:type	qb:Observation

Figure 9.12: Linked Data browser view on total asset in 2011 for Mastercard.

From the SPARK company template, we can also visit the OLAP Interface, Saiku, to create the same report as shown in the total asset line chart. For that, we create a pivot table with the issuer dimension filtered by Mastercard on columns, date dimension on rows and filtered on subject dimension with `us-gaap-2009:Assets` and `us-gaap-2011:Assets` on columns, as can be seen in Figure 9.13. The last row in the pivot table shows the value also displayed in the Pubby example in Figure 9.12.

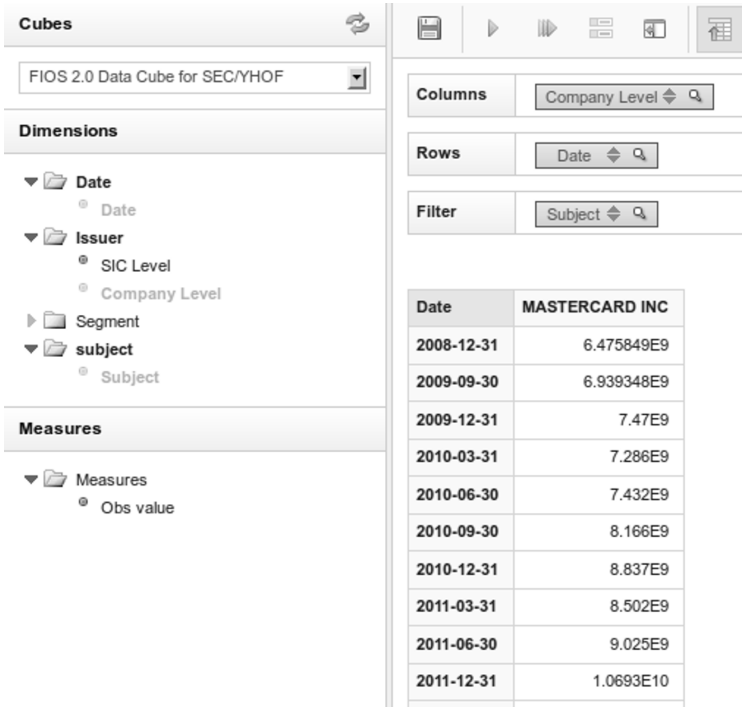
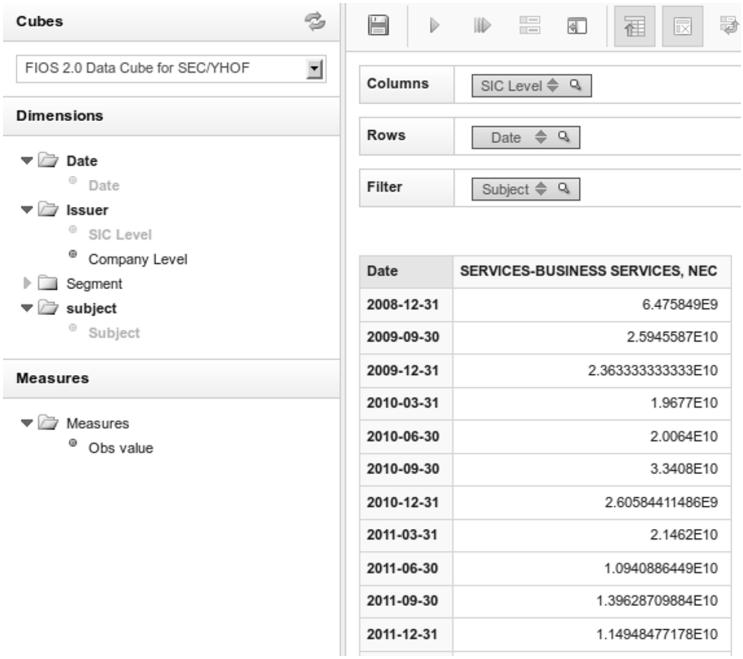


Figure 9.13: OLAP Interface query on Total Assets over time for Mastercard.

Although the three provided interfaces are connected through their underlying data, switching from one interface to another is difficult due to technical problems: SPARK tables did not allow to show browseable links; SPARK diagrams often contained aggregated or densely-displayed facts that are difficult to select for browsing; single facts often were modelled as blank nodes in QB and thus are not directly referenceable; and to browse a URI from FIOS, Pubby required adding "pubby" and converting "#" to "%23". One can think of implementations with a more seamless interlinking of interfaces.

9.3.2.3 Intuitively Create Own Reports and Analyses (XBRL User Requirement 3)

We show that a user can create a typical report on integrated financial data with intuitive OLAP operations: requesting a pivot table showing the Total Assets over time, similarly as for the total asset line and pivot charts, but this time aggregated to the industry level of Mastercard as visible in Figure 9.14.



Date	SERVICES-BUSINESS SERVICES, NEC
2008-12-31	6.475849E9
2009-09-30	2.5945587E10
2009-12-31	2.363333333333E10
2010-03-31	1.9677E10
2010-06-30	2.0064E10
2010-09-30	3.3408E10
2010-12-31	2.60584411486E9
2011-03-31	2.1462E10
2011-06-30	1.0940886449E10
2011-09-30	1.39628709884E10
2011-12-31	1.14948477178E10

Figure 9.14: OLAP Interface query on Total Assets over time for Business Services SIC.

Projection: By drag & drop of a measure to Columns, Rows or Filter fields in the pivot table, a user can select a certain measure. Since our data cube only contains one measure, projection is not necessary.

Dice: A user can filter for certain facts by clicking on the magnifier symbol of a dimension on the Columns or Rows fields. In our case, the user filters for certain subjects (`us-gaap-2009:Assets` and `us-gaap-2011:Assets`) as well as certain companies (Mastercard).

Slice: Any dimension that a user does not drag & drop to either Columns or Row fields gets sliced, i.e., removed and aggregated over. Since QB does not provide means to describe aggregation functions, FIOS uses the AVERAGE function as default; for numeric values the average returns an easy-to-understand and meaningful measurement.

Roll-Up: Any dimension listed on the left side can exhibit a hierarchy of several levels. For instance, for the issuer dimension, either Company or SIC Level can be selected. SIC Level groups companies by their SIC industry classification. In our example, we rolled-up to SIC Level and filtered for the SIC of Mastercard, “SERVICES-BUSINESS SERVICES, NEC”.

Drill-Across: Although not required in our example, an analyst may request a pivot table containing both observations from balance sheets and stock quote tables. The Saiku interface only allows to select one dataset per pivot table; an explicit Drill-Across can only be issued directly by using MDX and special-character-separated lists of data cubes. Instead, the multi-cube "FIOS 2.0 Data Cube for SEC/YHOF" pre-defined and created for FIOS contains both observations from balance sheets and stock quote tables.

9.3.3 Discussions and Lessons Learned

FIOS benefits from Semantic Web technologies in various ways: for instance, existing vocabularies such as QB and SKOS (based on OWL and RDFS) allow for intuitive modelling and integration of balance sheets from the SEC Edgar Database, stock quotes from the Yahoo! Finance Web API as well as company metadata from Wikipedia/DBpedia; the Linked Data principles ensure access to data in a standard and modular way. Since the schema of RDF is flexible, new data can easily be added by allowing the crawler to reach further entities. SPARQL can be used for quality checks and is sufficiently expressive for background information, multi-company and cross-data-sources analyses. Formal semantics such as explicit equivalence statements simplify access via entity consolidation. Three interfaces with different purposes use the same backend: any data that is added to the triple store can directly be visualised in SPARQL templates, browsed using the Linked Data Browser and queried using the OLAP Interface. Consequently, we argue that Semantic Web technologies allow a continuous integration of new data. With more heterogeneous datasets and frequent addition and updates of data sources, FIOS will develop its full potential if research resolves the following challenges:

Design interfaces and visualisations sufficiently specific to provide added value and generic to have new data immediately considered. If new information such as from text or structured databases, e.g., subsidiary relationships, product classifications, and organisational structures, are continuously added to FIOS, specialists are needed to adapt or create new SPARQL templates; the Linked Data Browser provides data only on a triple level; and the OLAP Interface requires integrated QB datasets such as in multi-cubes. Ideally, new data sources seamlessly and without much effort result in extended visualisations, e.g., providing more detailed provenance information, adding new data points or allowing additional interaction capabilities such as roll-up and drill-across.

Increase coverage and quality of information by continuously integrating sources. Integrity constraint checks need to be manually extracted and run. There may still be errors in the data, e.g., companies that share CIKs or ticker symbols because of a merger. Debreceny et al. [DdAF+10] have shown that some information may be derived only in a best-guess fashion. New data sources promise to reduce data quality

issues if integrated to one well-interlinked model. Then, the same KPIs can be calculated in different ways to identify differences between data sources, e.g., DBpedia "Operating Income" and the last yearly balance sheet net income loss. FIOS would need to consider uncertainty, to draw declarative knowledge from experts or other data sources, and to describe both static and dynamic relationships between financial data. One possible solution may be Complex Correspondences (Chapter 8).

Improve query processing performance. Although currently no issue in FIOS, preprocessing and integration will take too long for continuously updated and larger data sources. FIOS' current performance bottlenecks are large numbers of separately issued SPARQL queries and large numbers of multidimensional elements to load into the OLAP user interface. In more complex data integration and analysis scenarios optimisations such as parallelisation will be required. For instance, analytical queries that scan a large number of observations, contain filters of varying selectivity and compute aggregation functions on schema-flexible and heterogeneous data require specific data processing optimisations such as possible with RDF Aggregate Views(Chapter 7).

9.3.4 Related Work

We distinguish other financial data integration and analysis applications and related work about modelling XBRL data using Semantic Web technologies.

The Rhizomik Semantic XBRL demo [GG10] ties RDF representations of XBRL close to the original XML data which make mixing with other data sources difficult. The Business Intelligence Cross-lingual XBRL (BIXL) demonstrator [OCB+13] focuses on retrieving facts from unstructured text in filings as well as a multi-lingual interface, however does not consider data integration of XBRL balance sheets with stock quotes. Midas [BHH+11] implements a pipeline similar to FIOS without using Semantic Web technologies. Their main focus lies in extracting and linking of information about entities such as company and key people from semi-structured XML documents. However, it is unclear to what extent information from SEC and FDIC sources were integrated and what efforts would be needed to add new data sources such as Wikipedia.

Although judges saw potential, FIOS did not win the XBRL Challenge. Other submissions, in particular the winners – Calcbench and Sector3 – were more robust (e.g., FIOS is limited to certain browsers), provide keyword search or filtering for companies (e.g., "revenue higher than"), include a larger number of companies and filings (also non-balance-sheets), exhibit short update intervals with new filings (10-15min) and often provide MS Excel exports for further processing and analysis (Saiku also provides that, but this feature was not used).

In summary, although important for a holistic view on companies, current systems do not focus on integration of different data sources: whereas multi-company KPI analysis with an Excel export often is possible, background information, such as from

Wikipedia, rarely is embedded in the interfaces. Calcbench shows the actual stock quote of a company, yet, no other system allows for comparison of balance sheet KPIs with other numbers such as stock quotes over time. If systems find correspondences between companies or financial concepts, it is unclear whether the matching is hard-coded or flexibly represented with a formalism such as equivalence statements.

Several recent papers have proposed Semantic Web technologies as a suitable way to manage and model XBRL data. Wenger et al. [WTB11] consider the interoperability problems of different taxonomy versions, but apart from proposing the criteria they do not evaluate their approach. Bao et al. [BRLD10] try to fully keep the semantics of XBRL in an RDF/OWL representation; however, the authors do not describe the benefits of their representation in realistic case studies as done with FIOS. Similar to our work, Spies [Spi10] suggests to use RDF for modelling and data warehouses to analyse XBRL data. However, these works did not implement their solutions and did not demonstrate the benefits of using Semantic Web technologies in realistic use cases. Our FIOS tool shows the applicability and usefulness of modelling XBRL filings and taxonomies using the RDF Data Cube vocabulary.

9.3.5 Conclusions

We have described the Financial Information Observation System (FIOS) that models XBRL data using the RDF Data Cube Vocabulary; consolidates financial data for background, multi-company, and cross-data-sources KPI analysis; and provides intuitive and exploratory analysis interfaces. The benefit of Semantic Web technologies are a flexible schema, standard access, expressive queries and formal semantics. Main challenges to scaling-up those benefits in continuous integration scenarios are to design interfaces sufficiently specific to provide added value and generic to have new data immediately considered; to increase coverage and data quality with added data sources; and to optimise analytical operations on flexible schemas and heterogeneous data. In future work we are interested to investigate such challenges and to apply the FIOS approach to other domains such as medical decision support.

9.4 OGD Approach

After previously having introduced the Open Government Data (OGD) scenario (Section 2.3), we present a Linked-Data-based approach that partly fulfils the following user requirements:

- User Requirement 1: Citizens need to explore any of the Governmental Statistics datasets published on the Web in pivot tables.

- User Requirement 2: Citizens want to have confirmed important statistical indicators by as many datasets from the Web as possible.

Current approaches [SMDMM+12, Hoe13, MML+13] do not allow OLAP-like analyses on general datasets. In the next section, we present the Linked Data Cubes Explorer (LDCX) based on OLAP4LD and allowing the exploration of governmental statistics.

LDCX loads relevant data about datasets to an embedded triple store with SPARQL endpoint using our directed crawling strategy, validates the modelling, and executes metadata and analytical queries over the SPARQL engine.

In Section 9.4.2, for evaluation and demonstration, we present a user and performance study of LDCX. After describing related work in Section 9.4.3, we conclude in Section 9.4.4.

9.4.1 Linked Data Cubes Explorer (LDCX)

As illustrated in Figure 9.15, the *Linked Data Cubes Explorer* (LDCX) consists of several components: an HTML webpage using the JavaScript library xmla4js to connect via XMLA to an OLAP engine and to issue MDX queries (top of figure); a mediator olap4j-xmlaserver[38] to provide an XMLA interface for any olap4j-based OLAP engine such as OLAP4LD; and OLAP4LD to answer metadata and OLAP queries over Statistical Linked Data (bottom of figure).

In the following, we explain the modelling and mapping of data cubes, as well as the query processing and query optimisation over those data cubes in LDCX.

Modelling of data cubes. LDCX gets as input a comma-separated list of QB dataset URIs. Every single dataset is considered a data cube according to the MDM-QB Mapping. Implicitly, all selected data cubes form the global cube. Queries over the global cube will be interpreted as a nested set of Drill-Across operations generating a multi-cube according to the OLAP-to-SPARQL Algorithm.

Mapping of data cubes. Explicitly shared dimensions and dimension members such as between different datasets of Eurostat do not require additional mappings and are directly integrated in the global cube.

For implicitly shared dimensions and members LDCX finds equivalence `owl:sameAs` links directly in the RDF graphs of resolved URIs while loading the data cubes, e.g., when loading `allbus:geo.rdf#00`, LDCX finds mappings between Germany in Eurostat and Gesis.

[38] Part of olap4j technology stack, https://github.com/olap4j/olap4j-xmlaserver, last accessed 2014-11-07.

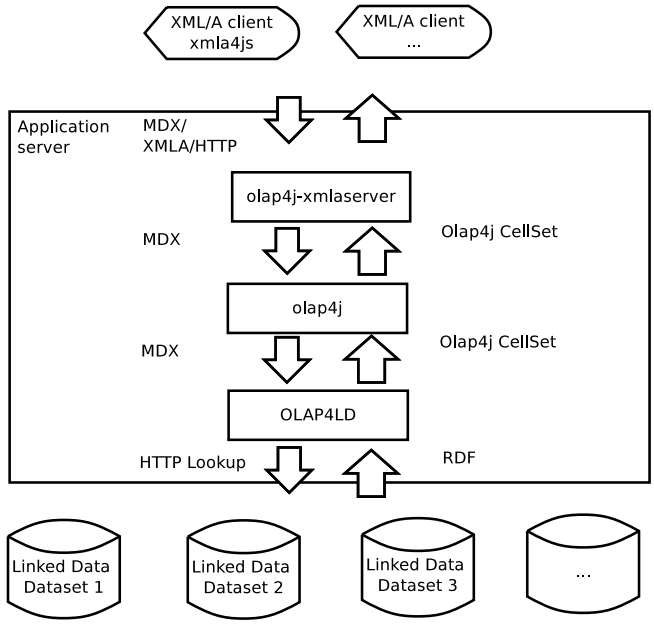


Figure 9.15: Architecture of the Linked Data Cubes Explorer.

Complex mappings between data cubes would allow LDCX to automatically increase the size of the global cube by computing conversions such as from "Million Euro" to "Euro" and compound measures such as the GDP Per Capita. So far, there is no RDF representation for such conversion and merging correspondences (Chapter 8); if a representation is adopted by publishers, LDCX will be able to find such mappings while loading the data.

Complex Correspondences – though implemented in OLAP4LD – are not activated in LDCX since even with a small number of correspondences, the global cube cannot be built sufficiently fast for ad-hoc and interactive analysis.

Query processing over data cubes. Along our Okun's law example from the introduction, we explain how the three step interface allows for exploring datasets: 1) A user selects one or more comma-separated URIs of *qb:DataSet*s. With "Explore Dataset…", metadata queries are issued to populate the user interface; Figure 9.16 shows a screenshot of how the URIs of the datasets for real GDP growth rate and employment growth are inserted as a comma-separated list.

2) The user selects measures to be displayed in the pivot table cells; in our case for each dataset the average measure is chosen. 3) The user selects dimensions to add member combinations to rows and columns of the pivot table and clicks "Update Table…".

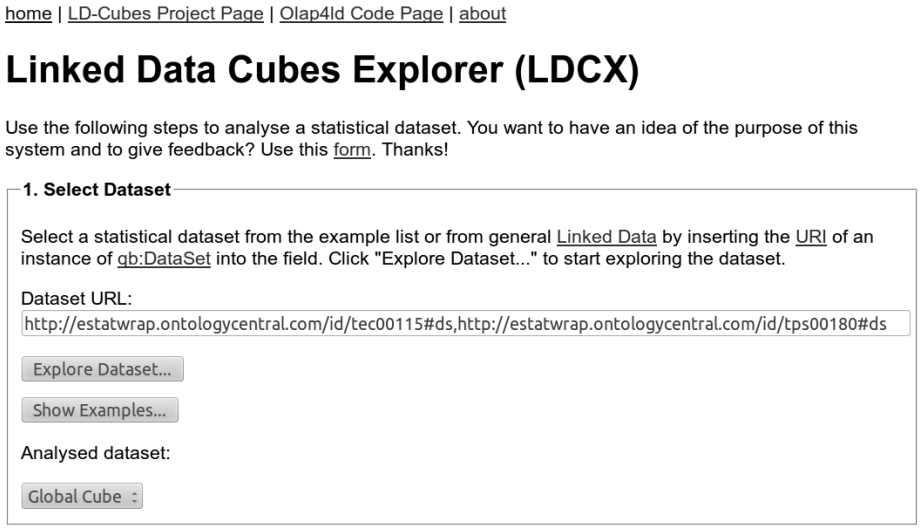


Figure 9.16: First step in three-step interface of LDCX; URIs for Real GDP Growth Rate and Employment Growth are inserted as a comma-separated list.

LDCX automatically queries every dimension on the most granular level since multi-level hierarchies are rarely used and users can still slice dimensions to view datasets on a higher aggregation level. In our example, as shown in a screenshot in Figure 9.17, the measures are to be shown in the columns and the dates (years) in the rows.

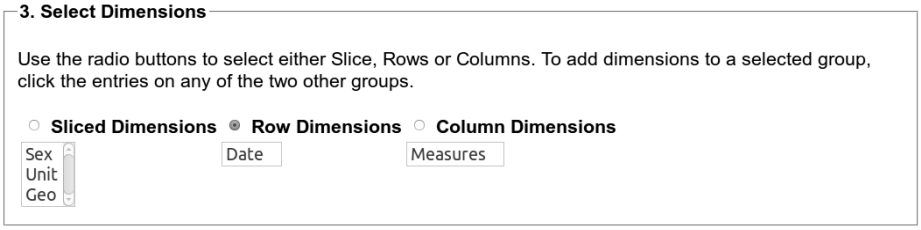


Figure 9.17: Third step in three-step interface of LDCX; measures are to be displayed in the columns, years in the rows of the pivot table.

The results of our example query are depicted in Figure 9.18. On top of the values in the pivot table, the correlation between the real GDP growth rate and employment growth can be computed to confirm or oppose Okun's law.

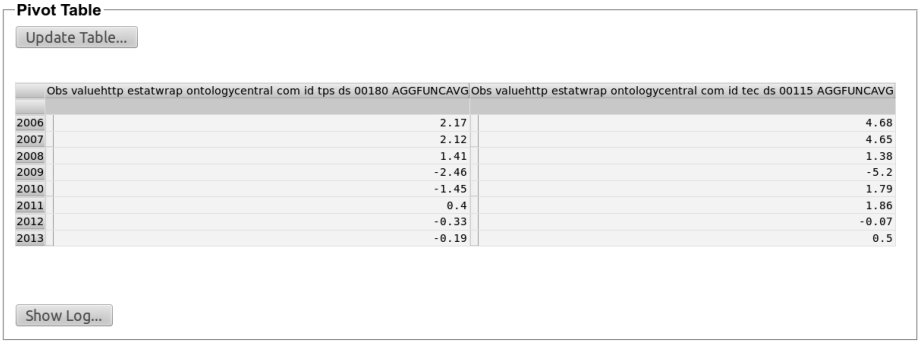


Figure 9.18: Query result in three-step interface of LDCX; real GDP growth rate and employment growth are displayed in pivot table as a basis for correlation analysis (left and right column, column names are concatenations of dataset URI and used aggregation function).

For LDCX we use our Linked Data Cubes Engine for OLAP4LD that uses an embedded Sesame triple store for query processing. *EmbeddedSesameEngine*[39] implements the MDM-QB Mapping.

To evaluate analytical queries, for a given MDX query created by the interface, LDCX implements our OLAP-to-SPARQL Algorithm, including drill-across operations to join the results.

Before executing a metadata or analytical query with SPARQL, the `EmbeddedSesameEngine` executes our directed crawling strategy and automatically loads relevant data for multidimensional datasets into an embedded Sesame RDF store. For that, `EmbeddedSesameEngine` first resolves all queried dataset URIs, then in turn asks SPARQL queries to its store for additional URIs to resolve and load; EmbeddedSesameEngine resolves all instances of concepts defined in the QB specification in the order they can be reached from the dataset URI, from `qb:DataStructureDefinitions` over `qb:ComponentProperty` to single `qb:Concepts`. Since there is no standard way to publish QB observations, the engine assumes that the observations are represented as blank nodes and stored at the location of the dataset URI.

This directed crawling has the advantage that necessary data is found quickly and not all information has to be given in one location, but can be distributed and reused, e.g., the range for the `ical:dtstart` dimension is provided by its URI. EmbeddedSesameEngine ensures that the entire QB dataset is loaded and well-formed according to the QB specification by executing SPARQL ASK queries for integrity constraints defined by the specification.

[39] https://github.com/bkaempgen/olap4ld/blob/master/OLAP4LD-trunk/src/org/olap4j/driver/olap4ld/linkeddata/EmbeddedSesameEngine.java, last accessed on 2014-10-22.

Those SPARQL ASK queries and implicitly shared dimensions and members require the SPARQL engine to evaluate OWL semantics. For that, EmbeddedSesameEngine uses the equivalence duplication strategy and materialises implicit information with repeatedly executed SPARQL INSERT queries.

Query optimisation. Some integrity constraints require to go through all observations, e.g., when checking that no two observations in the same `qb:DataSet` may have the same value for all dimensions (IC-12). Therefore, EmbeddedSesame allows to set a maximum number of triples for which those more complex integrity constraints are checked.

Also, LDCX currently loads anew the relevant data of queried data cubes for every query. This is because we expect that in theory data changes frequently in the original source and because we do not want to fill the repository with data that may not be needed anymore. Since historic data often does not change and different users possibly issue similar queries, heuristics-based performance optimisations would be possible.

9.4.2 Exploring Governmental Statistics

A demonstration system of LDCX is available online[40]. To show that we successfully applied LDCX, we first describe a user study to evaluate the usability and usefulness of LDCX. Then we describe a performance evaluation to gain an understanding of possible performance bottlenecks of LDCX.

To confirm the usefulness of LDCX, we have put LDCX online and have asked 20 business engineering students at the Karlsruhe Institute of Technology (KIT) to fill in an online questionnaire[41] that is adapted from the Computer System Usability Questionnaire (CSUQ) [Lew95]. With part 1 – a set of tasks that students had to solve using the system – we want to investigate whether participants can successfully use LDCX for data analysis and ensure that participants use all functionality of LDCX. With part 2 – a set of statements for which students had to indicate how much they like the system (from strongly agree to strongly disagree, 1 to 7) – we intend to assess the usability of LDCX and identify potential usability issues.

To increase the probability of students participating in the study, we have only used 11 questions – out of 19 from the original CSUQ – most relevant to our system. For instance, the expected results from the question "The interface of this system is pleasant." for LDCX are already covered by other questions, e.g., "I feel comfortable using this system." and "I like using the interface of this system.". Also the question "The information is effective in helping me complete my work." does not fit the evaluation's

[40] http://ldcx.linked-data-cubes.org/projects/ldcx/, last accessed 2014-11-07.

[41] https://docs.google.com/forms/d/1J721_e_eCCHjghLny1ZPoeb4X5g2h3RAetK7te_Vw6I/viewform, last accessed on 2014-11-07.

context of having students complete tasks that they would not normally execute. Having a reduced amount of questions and participants not experienced or familiar with the type of tasks is a limitation to our evaluation; still, as a tool targeting citizens for exploring Governmental Statistics, we expect the results to give an impression of the usability and usefulness of LDCX.

Eight students have taken part in the evaluation. According to the results of the first part of the evaluation, overall, students were able to fulfil the tasks. For instance, one task was to answer the following question: "What was the average GDP per capita in PPS for Germany?". All students were able to understand the three-step interface and the visualisation as pivot tables (Task 1, 2, 3, 4, 8); selecting datasets (Task 5, 8, 9, 10); projection (Task 1, 7); and slice (Task 5, 6, 7). For Task 8, 10, 11, two students missed to further drill-down to specific stock market and balance sheet values which probably was because the respective dimensions were not labelled intuitively. For the more complex Task 10 and 11, three students did not wait long enough for the answers. Only one participant could name the failed integrity constraint in Task 9, all others only noticed the general error message. This was expected since those students were not familiar with the RDF Data Cube Vocabulary.

The second part of the user study evaluation gave a more detailed idea of how usable the students found LDCX: We computed an overall CSUQ score by averaging over all answers. An overall score of 2.5 shows that students were satisfied with LDCX. Students mentioned that they understood the three-step interface within minutes and found the system easy to use. Worst scores are given in questions related to error messages (Feedback 6) and performance of the system (Feedback 3 and 10). Also, apparently, the search for datasets can be improved: students sometimes chose the wrong dataset (Task 9) and complained about the copy-paste of URIs needed to select a dataset.

Since performance of OLAP systems is crucial, we investigated possible bottlenecks of LDCX in a performance evaluation. We logged elapsed query times in ms per query processing step for a specific workload and computed the average over 5 runs of the same query after a warm-up.

As query processing steps we selected all steps related to our research contributions implemented in OLAP4LD. Therefore, the time to issue queries and to process the results in xmla4js and olap4j-xmlaserver we do not investigate.

More concretely, we are interested in the time 1) for loading and validating a dataset (load URIs, run normalisation, run integrity constraints), 2) generating a logical query plan (parse MDX, transform MDX parse tree and run metadata queries after loading dataset) and 3) executing the logical query plan and returning the results in the Linked Data Cubes Engine (generating and executing physical query plan, caching results).

The workload includes 1) all datasets from the user study to consider differently sized (from around 100 to 10,000 triples and 10 to 1000 observations) and distributed (static[42] or dynamically created files) datasets and 2) for each dataset a slice and drill down query testing the performance of the system regarding different aggregation levels. For every slice query, a pivot table was queried that shows the first dimension on the rows, all measures on the columns, and slices all other dimensions; for every drill down query, a pivot table was queried that uses a crossjoin of all dimensions on the rows and all measures on the columns. More details about the workload can be found on our evaluation page[43]. We run the experiments from a Java test case issuing XMLA queries directly on the LDCX XMLA server deployed on a VM in an OpenNebula framework with QEMU Virtual CPU version 0.9.1; 1x CPU with 2266.808 MHz; 32 KB cache size; and 4GB Memory. Figure 9.19 shows the results.

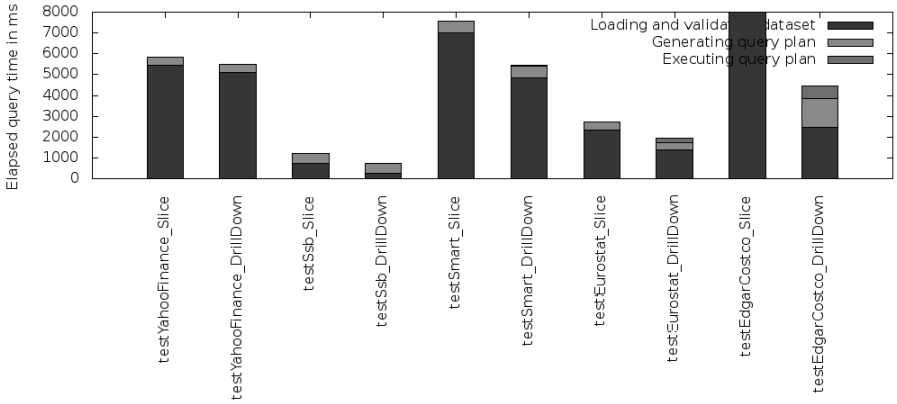


Figure 9.19: Elapsed query time in ms per query processing step for queries on datasets ordered by increasing number of triples; loading and validating dataset (bottom part of every bar) and generating query plan (middle part of bars) are always shown; executing query plan (top part of bar) often is too small to be displayed.

As expected, we see that dynamically generated datasets (Yahoo, Smart) take much longer to load and validate than static datasets (Ssb, Eurostat, Edgar). For the static datasets loading and validating increases with the size of the dataset. Because of a large number of metadata queries, for larger datasets LDCX needs more time to generate the query plan. Those many metadata queries to the Linked Data Cubes Engine are needed to search for the multidimensional element of an identifier in the MDX query. To overcome this bottleneck, we could include path information in identifiers to faster retrieve

[42] RDF files uploaded to Google Code source repository of OLAP4LD at olap4ld.googlecode.com/git/OLAP4LD-trunk/tests/, last accessed 2014-11-07.

[43] http://www.linked-data-cubes.org/index.php/LDCX_Evaluation_for_ESWC_2014_Demo, last accessed on 2014-11-13.

an element from the multidimensional model. Executing the query plan requires an OLAP query to the Linked Data Cubes Engine which takes longer for larger datasets (Eurostat, Edgar). We note that aggregated slice queries take less time for executing the query plan which may be because they require to transmit and cache fewer results. Also, we note that slice queries always take more time for loading and validating the dataset, although both slice and drill down query load the same QB dataset. The server (Google Code repository) from where the datasets are loaded seems to return results faster if asked in short intervals repeatedly for the same data. The Edgar slice query takes more than 20,000ms, mainly for loading; to make the other bars more visible in the diagram, we have fixed the y-axis to 8,000ms.

At the time of conducting the user study and performance evaluation, the drill-across operator was not available in LDCX. However, we demonstrated its usefulness for comparing values between datasets in a demonstration session at ESWC[44]. Since not all equivalence links can be found in Statistical Linked Data, we manually added such links to LDCX via a hard-coded list of file URIs and via direct insertion of `owl:sameAs` triples.

We allowed visitors to validate and explore governmental statistics with the Linked Data Cubes Explorer (LDCX). We showed how changes in modelling are propagated to LDCX by live modifying a QB dataset published via the platform Pastebin[45].

Also, we showed common modelling errors in existing QB datasets such as missing dimension `rdfs:range` or `qb:codeList`, no resolvable URIs, no data structure definition, and errors in the data structure definition.

In summary, according to the user study, the system seems usable and robust; improvements are possible regarding the performance, error messages and selection of datasets. The performance evaluation indicate that the main bottlenecks of LDCX lie in loading the dataset and in many metadata queries needed to identify elements in queries.

The Linked Data Cubes Explorer (LDCX) can partly fulfil our user requirements. The three-step-interface allows citizens to explore any of the Governmental Statistics datasets published on the Web (**OGD User Requirement 1**); LDCX accepts as input several datasets from which a Multi-Cube is generated using Drill-Across and values from several cubes can be compared (**OGD User Requirement 2**). The requirement is only partly fulfilled since the computation of compound measures and the consideration of conversion and merging relationships between datasets is not activated in LDCX since building and querying the global cube takes too long for exploratory analysis.

[44] See paper [KH14] and website describing demo at http://www.linked-data-cubes.org/index.php/OLAP4LD_Demo_at_ESWC_2014, last accessed 2014-10-28.

[45] http://pastebin.com/raw.php?i=839G2u72#ds, last accessed on 2014-11-05.

9.4.3 Related Work

The most common format to share datasets is XML. Other representations such as the Google Dataset Publishing Language, SDMX and XBRL require specific tools or focus on specific domains and provide few possibilities to link, and less-widely adopted mechanisms to access data over the Web.

Applications are available that, similar to LDCX, try to hide most RDF-specificities from the user to analyse a QB dataset. In the *stats.270a.info* analysis platform [CAR13] users can select two datasets from a fixed set for integration on the time and location dimension and for finding correlations in a scatter plot. McCusker et al. [MML+13] present *qb.js* to analyse the effect of tobacco policies on consumption. Though presenting useful systems to analyse QB datasets in specific data integration scenarios, it is unclear how well such approaches can be applied to more general use cases. Other systems provide more general analyses: Salas et al. [SMDMM+12] present *CubeViz* that offers faceted-browsing and visualizations of QB datasets but no OLAP analysis and integration of datasets via Drill-Across. The authors mention that in a use case around Open Government data in Brazil before the data could be analysed with CubeViz, data pre-processing took a lot of effort, e.g., modelling the data as cubes.

Hoefler [Hoe13] present the *CODE Visual Analytics Wizard* that automatically suggests appropriate chart types for QB datasets. LDCX automatically loads and checks the modelling of datasets and allows exploration of general datasets in pivot tables. Although the interface of LDCX is not as nice as of other systems, we argue that OLAP4LD reduces the costs of building analysis applications since OLAP clients, SPARQL engines as backends of Linked Data Cubes Engines, and existing Statistical Linked Data sources are separated and can be reused.

9.4.4 Conclusions

We applied OLAP4LD for the Linked Data Cubes Explorer (LDCX) that – different from current tools for analysing QB datasets – automatically starts an ETL process to load and validate necessary data for a query and allows users to explore arbitrary QB datasets in pivot tables via OLAP operations. According to a user study with eight participants, the system is robust and usable, improvements are possible regarding performance, error messages and selection of datasets. A performance evaluation revealed bottlenecks in loading the dataset and in querying for metadata elements.

Drill-across was not included in the evaluation; in general, from our experiences with LDCX we believe that users – for reasons of usability and intuition – do not want to manually select the right datasets in which to explore the facts. For future work, more potential we see if LDCX automatically builds and allows users to explore the global cube, a multi-cube generated from a large portion of all datasets available as

Statistical Linked Data. For flexible and efficient building and querying, Conversion Correspondences and RDF Aggregate Views may be necessary.

9.5 Discussion of Contributions

In this section, we discuss whether our contributions can help to fulfil the general requirements derived from our scenarios.

Table 9.3 shows an overview of general requirements where "X" indicates a contribution helping to fulfil a requirement. Every requirement is covered by at least two contributions.

Table 9.3: Requirements coverage analysis with "X" indicating research contributions (in columns) applicable to fulfil requirements (in rows).

Requirement	**Contribution**			
	MDM-QB Mapping	OLAP-to-SPARQL Algorithm	RDF Aggregate Views	Complex Correspondences
Flexibility				
R1: Integrate different data formats and models	X	X		
R2: Consider different identifiers for the same entities	X	X		
R3: Store provenance trail of datasets		X	X	
R4: Integrate datasets of different levels of detail	X	X	X	
R5: Consider datasets with different units of measurements	X	X		X
R6: Merge datasets for compound measures	X			X
Efficiency				
R7: Scale to large datasets	X		X	X
R8: Scale to large number of datasets	X	X		
R9: Efficiently filter for values with varying selectivity	X		X	
R10: Efficiently execute aggregation functions and formulas	X		X	X

In the following, we give more detail on how solving Requirement R1 to R6 can be supported by our contributions.

R1: Integrate different data formats and models. The MDM-QB Mapping and OLAP-to-SPARQL Algorithm help to integrate different data formats and models.

In every scenario, we were able to develop or use Linked Data wrappers to make datasets available to our systems as Statistical Linked Data. We used the MDM-QB Mapping to automatically map Statistical Linked Data to data cubes, multi-cubes, and the global cube.

As part of the MDM-QB Mapping, we also applied our definition of relevant data and implemented the directed crawling strategy. Correct modelling of data cubes was evaluated in the Financial Information Observation System (FIOS) and the Linked Data Cubes Explorer (LDCX).

The OLAP-to-SPARQL Algorithm was successfully applied in each of the scenario to allow queries over datasets originally represented in different data formats and models. For the SMART Knowledge Base (SKB) and FIOS, the Open Virtuoso triple store, for LDCX an embedded Sesame triple store were queried using SPARQL.

R2: Consider different identifiers for the same entities. The MDM-QB Mapping and OLAP-to-SPARQL Algorithm were helpful to integrate datasets that used different identifiers for identical dimensions and dimension values. Our solution proposes to use OWL semantics to evaluate mappings between dimensions and dimension values in Statistical Linked Data.

We used different ways to evaluate OWL equivalence semantics. In SKB, we used the reasoning capabilities of Open Virtuoso. In FIOS, we used entity consolidation. In LDCX, we used the equivalence duplication strategy with SPARQL INSERT queries over the Sesame triple store.

Equivalence statements such as `owl:sameAs` links were already published as Linked Data or manually added as RDF to the respective system.

R3: Store provenance trail of datasets. The OLAP-to-SPARQL Algorithm and RDF Aggregate Views execute OLAP queries using SPARQL engines and as such do not require a fixed schema such as a star schema in an RDBMS. Therefore, such approaches allow for storing arbitrary provenance information with the datasets.

Provenance information typically is modelled with triples outgoing from a QB dataset or observation URI. For instance, the publisher and the licence are typically stored with the dataset URI. Provenance information can be captured in different ways, e.g., directly from ETL tools generating datasets (Freitas et al. describe a possibility to track provenance within Java programs [FKaGO$^+$12]).

Provenance information also can be attached to entities to which datasets and observations refer. For instance, the triple store for SKB would not only contain numeric

data but also background information about locations, indicators, and scenarios added to Dropedia. Although not tried for SKB, background information such as the formula for an indicator could directly be displayed in the OLAP client (Diamantini and Potena show background information in pivot tables [DP10]).

In FIOS, we allow business analysts to first explore datasets using an OLAP interface and second to zoom into single datasets with a Linked Data browser. Also, FIOS presents background information from Wikipedia and Freebase. The directed crawling and the interface of LDCX could be extended to also load and visualise other Linked Data sources, e.g., reachable via `rdfs:seeAlso`.

R4: Integrate datasets of different levels of detail. The MDM-QB Mapping, OLAP-to-SPARQL Algorithm and RDF Aggregate Views allow for integrating datasets of different granularity.

Numeric data can be put on a lower granularity with the slice operation reducing the dimensionality of a data cube, and the roll-up operation aggregating facts along multi-level hierarchies. The MDM-QB Mapping ensures summarisability of lean datasets with no redundant facts and meaningful chosen aggregation functions.

For instance, in LDCX measures from several data cubes with different dimensions (e.g., the Unemployment and GDP Growth data cubes) can be compared if non-shared dimensions are sliced.

R5: Consider datasets with different units of measurements. The MDM-QB Mapping, OLAP-to-SPARQL Algorithm and Complex Correspondences support querying of datasets with different units of measurements.

Units of measurements can be modelled as dimensions of data cubes such as `estat-wrap:unit` with members such as `eurostat:dic/unit#EUR`.

Aggregating over values with different units seldom makes sense and our approach assumes users to choose meaningful aggregation functions. Using the MDM-QB Mapping and the OLAP-to-SPARQL Algorithm, data cubes can be fixed to a certain unit via the Dice operation.

Although not applied in the scenarios, Conversion Correspondences would allow for a more formal specification of units as a basis for integration. For instance, there may be many scientific units in hydrology and many different currencies in financial data analysis scenarios.

In SKB, for instance, conversion correspondences could be described in Dropedia using forms and from there retrieved by OLAP4LD.

R6: Merge datasets for compound measures. Conversion and Merging Correspondences allow for resolving more complex semantic conflicts than different identifiers

for identical dimensions and dimension values. Whereas Convert-Cube allows to transform single dimension values such as units, Merge-Cubes allows to combine facts from heterogeneous datasets.

Although not applied to the scenarios, the MDM-QB Mapping provides the possibility for compound measures using MDX and given an OLAP engine capable of such functionality (an example we have shown in the evaluation of the MDM-QB Mapping where we computed the "Percentage of Nos").

Increasing the size of the global cube using Complex Correspondences seems promising in the scenarios but due to a focus on exploratory analysis was not applied; for instance, indicators could be automatically computed in SKB, and the number of answers for the GDP Per Capita increased for LDCX but the pre-computation of the global cube would take a lot of time.

Similar as for conversion correspondences, in SKB, merging correspondences could be described in Dropedia using forms and from there retrieved by OLAP4LD.

Next, we discuss our contributions with respect to Requirement 7 to Requirement 10 on efficiency of approaches.

R7: Scale to large datasets. The MDM-QB Mapping allows for reusing existing OLAP engines that are developed for queries over large data warehouses. For instance, OLAP engines over relational databases and star schemas perform well with large datasets as we have shown in the Star Schema Benchmark.

Furthermore, RDF Aggregate Views are introduced for optimising queries over single large datasets.

Although materialised views were not used in the scenarios, query optimisations will be important in the future, especially if a materialised global cube is to be queried.

For instance, we were not able to explore the GDP per Capita from the World Bank[46] and the International Monetary Fund[47]. Both datasets contain values about the GDP or the population and could be used to compute the GDP per Capita. However, the World Bank dataset would link to a file with 2GB of RDF data (in Turtle format) with all observation URIs, the IMF dataset would link to a file with 240MB of RDF data (in Turtle) with the observation URIs. After loading such large files one still needs to resolve every single observation. With 1,285,448 observations in the IMF dataset alone and assuming 500ms per look-up, it will take more than a week to load all observations to a triple store. One remedy would be to provide data dumps in RDF that can be reached via properties, e.g., based on the VoiD vocabulary.

Larger datasets would further exacerbate bottlenecks of LDCX such as integrity constraint checks that require to go through many observations and the repeated loading of

[46] http://worldbank.270a.info/dataset/GDPPCKN, last accessed on 2014-08-18.
[47] http://imf.270a.info/dataset/PGI, last accessed 2014-11-18.

data cubes into the triple store. Also, we expect the equivalence duplication strategy to become a bottleneck if many mappings are available.

R8: Scale to large number of datasets. The MDM-QB Mapping and the OLAP-to-SPARQL Algorithm allow queries over several datasets. For SKB, we created a multi-cube "SMART-DB-DSD" with more than 100 data cubes from 27 analyses datasets in Dropedia, and from sensor datasets with 132 indicators and 6,517 locations. For FIOS, we created a multi-cube "FIOS 2.0 Data Cube for SEC/YHOF" with more than 100 data cubes from daily stock quote tables and quarterly and yearly balance sheets since 2009. Assuming sufficient amount of memory for the embedded Sesame triple store and time for downloading the data, in LDCX, multi-cubes with as many data cubes as available and relevant for a use case can be queried.

R9: Efficiently filter for values with varying selectivity. The MDM-QB Mapping allows for reusing existing OLAP engines that are able to efficiently execute the Dice operation. RDF Aggregate Views contain smaller number of facts than the original dataset and therefore allow for more efficient execution of filter operations.

In our scenarios, we did not use existing OLAP engines and RDF Aggregate Views but applied our OLAP-to-SPARQL Algorithm over a SPARQL engine that executed filtering sufficiently fast.

R10: Efficiently execute aggregation functions and formulas. Similar as for the filter operation in R9, the MDM-QB Mapping allows for reusing existing OLAP engines that are able to efficiently execute the Slice and Roll-Up operations. RDF Aggregate Views pre-compute aggregated facts and contain smaller number of facts than the original dataset and therefore allow for more efficient execution of aggregation operations.

Also, similar as for the filter operation, we did not use existing OLAP engines and RDF Aggregate Views in our scenarios but applied our OLAP-to-SPARQL Algorithm over a SPARQL engine that showed sufficiently fast with respect to aggregations.

In summary, we demonstrated that our contributions are useful to fulfil the general requirements in our scenarios. Statistical Linked Data showed useful for data integration and analysis scenarios. Standard and modular access mechanisms as well as standard vocabularies helped coordinating the publication of different data sources. The schema-flexibility of RDF proved useful to crawl and load data. SPARQL 1.1 allowed sufficiently expressive queries such as for integrity constraint checks to ensure that data is correctly modelled. Formal semantics provide various ways to evaluate implicit information such as the equivalence between identifiers. Also, OLAP analysis showed a useful interaction paradigm over multidimensional datasets from the Web. Given one global cube for exploration, our experiences suggest that domain experts can express their information need and issue queries for pivot tables. We have developed an OLAP engine, OLAP4LD, to implement our approaches and to help developers building integration and analysis applications. We have applied OLAP4LD in three scenarios and

have shown how the contributions can fulfil ten requirements about flexibility and efficiency. The MDM-QB Mapping and OLAP-to-SPARQL Algorithm were used in each of the scenarios to execute metadata and OLAP queries over Statistical Linked Data. RDF Aggregate Views were not necessary in the scenarios, but materialised aggregate views are promising in case new and larger datasets are added continuously. Though complex correspondences were not collected in the scenarios, automatically deriving implicit numeric values such as compound measures is promising in every scenario for further increasing the usefulness of the integration and analysis systems.

In summary, our contributions allow to successfully overcome overall requirements identified in three different scenarios. Various approaches combining our contributions can be used that find a suitable compromise between efficiency and flexibility. Three approaches have been implemented and the developed systems (SKB, FIOS, LDCX) demonstrate how to fulfil user requirements in the scenarios.

10 Conclusions

The overall goal of this work was to investigate approaches for flexible integration and efficient analytical queries over multidimensional datasets from the Web, as required by domain experts such as natural scientists, business analysts, and citizens interested in politics.

We summarise our results in the first section. Then, we reflect on the significance of the results in Section 10.2 and describe open questions in Section 10.3.

10.1 Summary of Results

In this work, we have presented complementary methods for flexibly building a unified view over multidimensional datasets from the Web and for efficiently querying over this global cube in different domains.

Our approach assumes publishers of datasets to use Linked Data and the RDF Data Cube Vocabulary and domain experts to be familiar with OLAP analysis. We provide the following contributions:

- The MDM-QB Mapping (Chapter 5) enables to build and query multi-cubes consisting of several QB datasets with existing OLAP engines.
- The OLAP-to-SPARQL Algorithm (Chapter 6) supports drill-across for building a multi-cube directly using a SPARQL engine.
- RDF Aggregate Views (Chapter 7) apply materialised aggregate views to analytical queries using a SPARQL engine.
- Complex Correspondences (Chapter 8) allow for deriving data cubes from existing ones and enable to increase the number of answers from the global cube.

When we described the State of the Art at the beginning of this thesis, we identified a gap of flexible and efficient approaches. In the previous overall evaluation chapter we showed that our approach of OLAP over Statistical Linked Data and our contributions can be applied to build flexible and efficient systems for three different scenarios.

Figure 10.1 zooms into the former gap (right upper quadrant in Figure 4.1) and gives an overview of approaches possible through our contributions. Since flexibility and efficiency put contradictory requirements to approaches, a compromise has to be found in specific scenarios. In the following, we describe how our contributions can be used separately or in combination to reach a suitable trade-off between flexibility and efficiency.

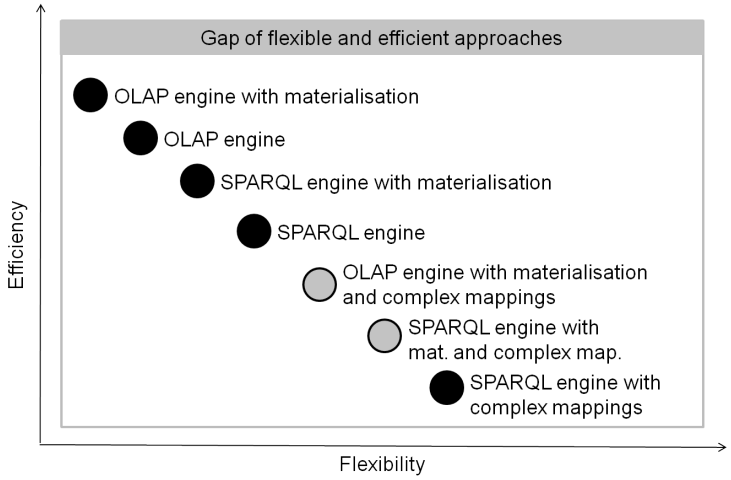


Figure 10.1: Overview of presented integration and analysis approaches over Statistical Linked Data filling gap of combined flexibility and efficiency; black circles refer to approaches that directly apply our contributions and are evaluated in our experiments, grey circles refer to promising approaches made possible with our contributions.

In our descriptions of approaches, we start with the approach prioritising efficiency in the flexibility-efficiency trade-off (left upper corner of Figure 10.1). Every approach suggests a specific trade-off that – given specific requirements – may be appropriate for a concrete scenario.

OLAP engine with materialisation. In this approach, the MDM-QB Mapping allows to use an existing OLAP engine for dataset integration and efficient query processing; furthermore, materialised aggregate views are used for query optimisation.

Flexibility is increased since based on the MDM-QB Mapping, relevant data for multidimensional datasets is automatically found in Statistical Linked Data. If datasets do not contain redundancies and a meaningful aggregation function is selected (lean QB dataset), this approach performs a correct evaluation of OLAP queries. Also, multi-cubes are automatically created from all available data cubes and equivalence statements between different identifiers of entities in data cubes are considered (Chapter 5).

Experiments with the ROLAP engine Mondrian, the RDBMS MySQL, and aggregate tables show that this approach enables efficient queries over large datasets (Chapter 7) for the following reason: aggregate tables reduce the complexity of queries and the number of facts to be processed in a query. Similarly, we can expect efficient analytical queries over a materialised global cube from many single cubes.

OLAP engine. Here, an OLAP engine but no materialisation is used for dataset integration and query processing. Flexibility is increased since no selection, computation, and maintenance of materialised aggregate views are needed. Our experiments show that efficiency is decreased since no materialisation is done (Chapter 7).

SPARQL engine with materialisation. In this approach, the MDM-QB Mapping enables executing metadata queries over Statistical Linked Data, the OLAP-to-SPARQL Algorithm translates OLAP to SPARQL queries over an existing SPARQL engine, and RDF Aggregate Views optimise query processing via materialisation.

In comparison to OLAP engines using a fixed schema (e.g., star schema) to store the multidimensional data model, this approach is more flexible since both numeric data and background information are accessed via a SPARQL engine. The OLAP-to-SPARQL Algorithm also allows for querying over multi-cubes from all available data cubes and for considering equivalence statements between different identifiers of entities in data cubes (Chapter 6).

Experiments with the SPARQL engine Open Virtuoso show that RDF Aggregate Views can lead to more efficient query processing in SPARQL engines. However, experiments also show that analytical query optimisation in SPARQL engines over schema-flexible RDF is more difficult than analytical query optimisation in the relational setting. For instance, whereas in an RDBMS aggregate views usually are stored in separate tables, RDF typically is stored and queried in one graph of self-descriptive RDF. Also, the flexible schema of RDF allows fewer assumptions about the data (Chapter 7).

SPARQL engine. Here, a SPARQL engine but no materialisation is used for dataset integration and query processing. Similar to the OLAP engine approach without materialisation, flexibility is increased since no selection, computation, and maintenance of materialised aggregate views are needed. Our experiments show that certain queries may be slower if no materialisation is done (Chapter 7).

OLAP engine with materialisation and complex mappings. In this approach we combine the MDM-QB Mapping with materialisation and complex mappings for integration and analytical queries using an existing OLAP engine. Aggregate views are materialised. Also, Complex Correspondences are directly described over the data in the data warehouse [CDL+01, SSR94] (Chapter 8).

Although not evaluated in scenarios (therefore, grey circle), our experiments with computing the GDP Per Capita have shown that materialisation and the evaluation of com-

plex correspondences can be combined to find a good compromise between efficiency and flexibility.

Flexibility is increased since heterogeneities between datasets can be resolved. Heterogeneities would be described on a logical level using Datalog and can automatically be used in query processing, e.g., query rewriting techniques. We believe that existing approaches to query rewriting [Gen10, see Query Folding Chapter] and complex correspondences [CDL+01, SSR94] can be applied on top of the MDM-QB Mapping; the abstract representation of mappings would be transformed to the logical representation used by the OLAP engine (e.g., star schema).

Efficiency is decreased due to the evaluation of Complex Correspondences but can be optimised with offline computation, materialisation techniques, and query rewriting.

SPARQL engine with materialisation and complex mappings. Here, all our contributions are combined in one approach. The MDM-QB Mapping allows for executing metadata, the OLAP-to-SPARQL Algorithm allows for executing OLAP queries over Statistical Linked Data using an existing SPARQL engine. Also, RDF Aggregate Views optimise query processing via materialisation and Complex Correspondences provide descriptions of how to resolve semantic conflicts between data cubes.

Flexibility is further increased since not only simple (equivalence statements) and complex (Conversion and Merging Correspondences) mappings between data cubes can be described (Chapter 8), but also the SPARQL engine approach allows to store both numeric data and background information in the same backend (Chapter 6).

Our experiments comparing analytical queries with existing OLAP and SPARQL engines indicate that a SPARQL engine is more difficult to optimise than the relational pendant when using an existing OLAP engine (Chapter 7) as in the previous approach.

Although not evaluated in experiments (therefore, grey circle), we believe that RDF Aggregate Views can be applied on top of Conversion Correspondences; for instance, the global cube could be materialised using RDF Aggregate Views. Existing selection and maintenance methods may be applied in this case [SDN00].

SPARQL engine with complex mappings. Here, based on the MDM-QB Mapping and the OLAP-to-SPARQL Algorithm, Complex Correspondences are evaluated using a SPARQL engine.

In comparison to the previous approach, flexibility is increased and efficiency is decreased since complex mappings are evaluated but no materialisation of pre-aggregated facts is done (Chapter 8).

In summary, there is a compromise between flexibility and efficiency: Using the OLAP-to-SPARQL Algorithm over SPARQL engines offers more flexibility but may be slower than optimised OLAP engines. Using RDF Aggregate Views may allow to speed up

query processing but is less flexible since aggregate views need to be maintained. Describing Complex Correspondences between datasets allows to flexibly resolve semantic conflicts but will largely increase the size of the Global Cube and thus reduce query performance. We believe that for specific scenarios, contributions can be combined leading to an appropriate trade-off between flexibility and efficiency.

10.2 Significance of Results

Before we describe the significance, we want to clarify three aspects of our work:

Despite the name of the RDF Data Cube Vocabulary, the mapping between Statistical Linked Data and the multidimensional data model of data cubes is not straightforward: for instance, we must identify all relevant data for datasets distributed over several URIs and we must ensure correct modelling with integrity constraint checks (directed crawling strategy); we assume datasets without redundancy for correct aggregations (lean dataset); and we allow different modelling of hierarchies. Most importantly, the MDM-QB Mapping defines multi-cubes of cubes with implicitly shared dimensions and dimension members. Based on this definition, the OLAP-to-SPARQL Algorithm allows to explicitly generate multi-cubes with Drill-Across; in turn, the global cube is defined as a nested set of drill-across operations over all available data cubes.

Since we reuse existing RDBMS, OLAP engines, and SPARQL engines to execute queries and assume efficient physical query plans generated, our contributions are broadly applicable. Our focus in this work was on modular and interoperable methods based on (Web) standards and widely-used query engines. For instance, relevant data is identified according to Linked Data principles. Also, the MDM-QB Mapping, the OLAP-to-SPARQL Algorithm, RDF Aggregate Views, and Complex Correspondences are defined using RDF and SPARQL. Although our scenarios provided limited amount of data, experiments indicate that query performance time – at least if complex correspondences between datasets are not considered – does not overly increase with the expected increase in size and number of datasets available in Statistical Linked Data. To increase query performance further, our OLAP and SPARQL engines can be replaced with engines providing SQL or SPARQL interfaces that deploy in-memory, column-oriented, multi-core, or NoSQL technology.

Our work does not pursue innovative user interfaces and does not report about interesting findings from actual analyses by domain experts but instead investigates data integration and analytical query processing approaches over both numeric and arbitrary background information. For example, whereas a star schema is fixed to a prespecified multidimensional data model, a SPARQL engine allows for querying over schema-flexible RDF data. Therefore, our approaches minimise assumptions concerning queried data. Flexible interfaces and visualisations are interesting research areas

beyond the scope of this work; FIOS with its three different interfaces on financial data over a single backend and other examples such as a provenance browser [FKaGO+12] sufficiently demonstrate the usefulness of a unified RDF representation. Also, the systems presented to potential users in scenarios were prototypes. We believe that with more focus on usability and robustness (for instance, the FIOS system could only be used with a specific browser), adoption and real-world analysis findings are possible.

We give three examples of where our work has proved relevant:

First, with this work we contributed to the effort in providing easier – since more homogeneous – access over statistics made openly available on the Web. Our experiences from using the RDF Data Cube Vocabulary (QB) for the SMART Research project, for the XBRL Challenge, and for consuming Open Government Data have supported the standardisation of QB by the World Wide Web Consortium (W3C)[1]. Also, our contributions are among the first to propose analytical query processing over QB datasets (MDM-QB Mapping, OLAP-to-SPARQL), to investigate a possible query optimisation method (RDF Aggregate Views), and to describe semantic mappings (equivalence statements for implicitly shared dimensions and members, Complex Correspondences).

Second, our contributions helped quantifying the advantages and disadvantages of using RDF for data analytics. For example, database vendor OpenLink repeated our experiments and confirms that the more flexible schema in SPARQL engines leads to difficulties in finding an efficient query plan[2]. The application of our contributions in three different domains give guidance on finding an appropriate trade-off between flexibility and efficiency in other scenarios.

Third, we have contributed to a discussion on the right conceptual model for domain experts dealing with multidimensional datasets from the Web. For instance, the Linked Open Data Extension of the widely-used Data Mining tool RapidMiner[3] uses OLAP4LD to query multidimensional datasets from the Web and to load results into algorithms such as for correlation analysis.

[1] See use cases and lessons in Working Group Note [KC13], edited by the author in collaboration with other researchers participating in W3C Government Linked Data Working Group.

[2] For more information see comments by Orri Erling made in ESWC 2013 Panel "Semantic Technologies for Big Data Analytics" about our work, http://www.openlinksw.com/dataspace/oerling/weblog/Orri%20Erling%27s%20Blog/1730, last accessed on 2014-10-29

[3] See "Accessing RDF Data Cubes" in RapidMiner Linked Open Data Extension, Manual, Version 1.5, 09/19/14, http://dws.informatik.uni-mannheim.de/fileadmin/lehrstuehle/ki/research/RapidMinerLODExtension/RapidMinerLODExtensionManual.pdf, last accessed 2014-10-29.

10.3 Open Questions

We see three main areas of future work about building and querying the Global Cube:

First, we envision a version of OLAP4LD that allows exploratory analysis of the global cube, that means all currently available datasets modelled using the RDF Data Cube Vocabulary. We expect that interesting queries will only be possible if more mappings are manually or automatically generated. Mappings include equivalence statements such as between companies from the SEC, Yahoo! Finance, and other data sources, as well as conversion and merging correspondences such as for computing the Earnings per Share, the Altman Z-score, and other interesting financial ratios. To evaluate such mappings over all available statistical datasets – especially if integrated with schema-flexible, and semi-structured background information – will not be possible without specific optimisations. One possibility are parallel computations using many machines. Another way to optimise performance may be to only load data structure definitions of cubes and to then load the actual numeric facts if requested by a query.

Second, it would be interesting to design analysis interfaces and backends that can cope with a continuous stream of new facts from datasets, of new datasets from data sources, and possibly also of new data sources. Such a system would need to be sufficiently specific to provide added value to domain experts and generic to have new information immediately considered. For example new data could allow views on an additional granularity level. Regarding efficiency, a polling mechanism may not suffice to immediately consider new facts from datasets. Regarding flexibility, new datasets, and especially new data sources, will continuously exhibit new, heterogeneous schemas that – to be considered in interfaces – need to be integrated. If information is provided and consumed in real-time decision makers can immediately learn about micro-data changes such as from climate sensors and financial transactions. Also, coverage and quality of information steadily improve; for instance, if the same statistical values are computed from different data sources, reliability is increased.

Third, having access to the Global Cube of statistical datasets and an abundance of semi-structured background information will allow for holistic analyses of domain models in decision support. Models range from Okun's law, over the water resources situation in the lower Jordan Valley, to possible macro-economic explanations for the financial crisis. Inputs to the system are hypothetical values from experts and what-if-analyses from background information. Then, different domain models can be simulated based on the Global Cube and predictions and even suggestions computed. Conversion and merging correspondences will not suffice to formalise complex models, but mathematical formalisations and uncertainties have to be considered. Self-improving, intelligent approaches would automatically learn to identify the appropriate models for specific, ad-hoc problems.

Further investigation of such open questions about building the global cube, considering a continuous stream of facts, evolving interfaces, and simulating complex domain models could eventually change the way many experts do their daily work. Natural scientists can define water management objectives and indicators and get immediate feedback on their assumptions without manual acquisition of relevant micro and macro data; business analysts have all finance-relevant information about companies at their fingertips without tedious data pre-processing tasks; and citizens and politicians can directly support claims by referring to datasets on the Web; datasets that in turn can transparently be confirmed or opposed by anyone publishing numeric data on the Web.

A Overview of Additional Information Provided on the Web

For several parts of the work, we make additional information openly available on the Web. Here, we give an overview of such additional information, including short descriptions and links.

- We published OLAP4LD as Open Source. Links to the source code and the documentation about OLAP4LD we maintain on the Linked Data Cubes website[1].
- Experiments for the evaluation of the Drill-Across operation in Chapter 6 as well as Convert-Cube and Merge-Cubes operations in Chapter 15 we provide on a page on the Linked Data Cubes website[2].
- More information about the Star Schema Benchmark and experiments for evaluating RDF Aggregate Views in Chapter 7, including required and generated data, we provide on a benchmark website[3].
- Background information to our user study and performance evaluation of the Linked Data Cubes Explorer we provide on a page on the Linked Data Cubes website[4].
- A demo of the Linked Data Cubes Explorer we make available at a website[5].

[1] http://www.linked-data-cubes.org/index.php/OLAP4LD, last accessed 2014-11-13.
[2] http://www.linked-data-cubes.org/index.php/Global_Cube_Evaluation_EKAW14, last accessed on 2014-11-13.
[3] http://people.aifb.kit.edu/bka/ssb-benchmark/, last accessed on 2014-11-13.
[4] http://www.linked-data-cubes.org/index.php/LDCX_Evaluation_for_ESWC_2014_Demo, last accessed on 2014-11-13.
[5] http://ldcx.linked-data-cubes.org/projects/ldcx/, last accessed 2014-11-13.

Bibliography

[ABD+99] Jens Albrecht, Andreas Bauer, O. Deyerling, Holger Günzel, Wolfgang Hümmer, Wolfgang Lehner, and Lutz Schlesinger. Management of Multidimensional Aggregates for Efficient Online Analytical Processing. In *International Database Engineering and Applications Symposium, IDEAS*, 1999.

[AdAB00] Anil Agarwal, Marian S. delos Angeles, and Ramesh Bhatia. Integrated Water Resources Management. Technical report, Global Water Partnership, Technical Advisory Committee (TAC), 2000.

[ADE+13] Alberto Abelló, Jérôme Darmont, Lorena Etcheverry, Matteo Golfarelli, Jose-Norberto Mazón, Felix Naumann, Torben Pedersen, Stefano Bach Rizzi, Juan Trujillo, Panos Vassiliadis, and Gottfried Vossen. Fusion Cubes: Towards Self-Service Business Intelligence. *International Journal of Data Warehousing and Mining*, 9(2), 2013.

[AF11] Darko Anicic and Paul Fodor. EP-SPARQL: A Unified Language for Event Processing and Stream Reasoning. In *World Wide Web Conference (WWW)*, 2011.

[AFR11] Alberto Abelló, Jaume Ferrarons, and Oscar Romero. Building Cubes with MapReduce. In *ACM 14th International Workshop on Data Warehousing and OLAP (DOLAP)*, 2011.

[AGS97] Rakesh Agrawal, Ashish Gupta, and Sunita Sarawagi. Modeling Multidimensional Databases. In *13th International Conference on Data Engineering*, 1997.

[AK07] José Luis Ambite and Dipsy Kapoor. Automatically Composing Data Workflows with Relational Descriptions and Shim Services. In *International Semantic Web Conference (ISWC)*, 2007.

[ALS+12] Ahmad Assaf, Eldad Louw, Aline Senart, Corentin Follenfant, Raphaël Troncy, and David Trastour. RUBIX: A Framework for Improving Data Integration with Linked Data. In *1st International Workshop on Open Data (WOD)*, 2012.

[AM08] Daniel J. Abadi and Samuel R. Madden. Column-Stores vs . Row-Stores: How Different Are They Really? In *ACM SIGMOD International Conference on Management of Data (SIGMOD)*, 2008.

[ASS03] Alberto Abelló, José Samos, and Fèlix Saltor. Implementing Operations to Navigate Semantic Star Schemas. In *ACM International Workshop on Data Warehousing and OLAP (DOLAP)*, 2003.

[BBNA12] Seyed-Mehdi-Reza Beheshti, Boualem Benatallah, Hamid R. Motahari Nezhad, and Mohammad Allahbakhsh. A Framework and a Language for On-Line Analytical Processing on Graphs. In *13th International Conference on Web Information Systems Engineering (WISE)*, 2012.

[BG97] Stephane Bressan and Cheng Goh. Semantic Integration of Disparate Information Sources over the Internet Using Constraint Propagation. Technical Report 02142, Massachusetts Institute of Technology, 1997.

[BHH+11] Douglas Burdick, Mauricio A. Hernández, Howard Ho, Georgia Koutrika, Rajasekar Krishnamurthy, Lucian Popa, Ioana Stanoi, Shivakumar Vaithyanathan, and Sanjiv R. Das. Extracting, Linking and Integrating Data from Public Sources: A Financial Case Study. *IEEE Data Engineering Bulletin*, 2011.

[BJK+12] Lukas Blunschi, Claudio Jossen, Donald Kossmann, Magdalini Mori, and Kurt Stockinger. SODA: Generating SQL for Business Users. *VLDB Endowment*, 5(10), 2012.

[BJL14] Laurence Ball, Joao Tovar Jalles, and Prakash Loungani. Do Forecasters Believe in Okun's Law? An Assessment of Unemployment and Output Forecasts. Technical report, International Monetary Fund, 2014.

[BP13] Stefan Bischof and Axel Polleres. RDFS with Attribute Equations via SPARQL Rewriting. In *10th Extended Semantic Web Conference (ESWC)*, 2013.

[BPZ11] Anja Bog, Hasso Plattner, and Alexander Zeier. A mixed transaction processing and operational reporting benchmark. *Information Systems Frontiers*, 13(3), 2011.

[BRLD10] Jie Bao, Graham Rong, Xian Li, and Li Ding. Representing Financial Reports on the Semantic Web: A Faithful Translation from XBRL to OWL. In *International Conference on Semantic Web Rules*, 2010.

[CAR13] Sarven Capadisli, Sören Auer, and Reinhard Riedl. Linked Statistical Data Analysis. In *Second International Workshop on Semantic Statistics (SemStats)*, 2013.

[CCGL02] Andrea Cal, Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Data Integration Under Integrity Constraints. *Information Systems*, 29(2), 2002.

[CCS93] Edgar Frank Codd, S.B. Codd, and C.T. Salley. Providing OLAP to User-Analysts: An IT Mandate. Technical report, E.F. Codd Associates, 1993.

[CD97] Surajit Chaudhuri and Umeshwar Dayal. An Overview of Data Warehousing and OLAP Technology. *ACM SIGMOD Record*, 26(1), 1997.

[CDL+01] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, and Riccardo Rosati. Data Integration in Data Warehousing. *International Journal of Cooperative Information Systems*, 10(3), 2001.

[CFG+10] Richard Cyganiak, Simon Field, Arofan Gregory, Wolfgang Halb, and Jeni Tennison. Semantic Statistics: Bringing Together SDMX and SCOVO. In *WWW Workshop on Linked Data on the Web (LDOW)*, 2010.

[CGLG04] Conor Cunningham, César A. Galindo-Legaria, and Goetz Graefe. PIVOT and UNPIVOT: optimization and execution strategies in an RDBMS. In *30th International Conference on Very Large Data Bases (VLDB)*, 2004.

[Cha98] Surajit Chaudhuri. An Overview of Query Optimization in Relational Systems. In *17th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 1998.

[CHH+13] Philippe Cudr, Albert Haque, Andreas Harth, Felix Leif Keppmann, Daniel P Miranker, Juan F Sequeda, and Marcin Wylot. NoSQL Databases for RDF: An Empirical Evaluation. In *International Semantic Web Conference (ISWC)*, 2013.

[CL10] Roger Castillo and Ulf Leser. Selecting Materialized Views for RDF Data. In *10th International Conference on Current Trends in Web Engineering*, 2010.

[CRB+06] Lei Chen, Raghu Ramakrishnan, Paul Barford, Bee-Chung Chen, and Vinod Yegneswaran. Composite Subset Measures. In *32nd International Conference on Very Large Data Bases (VLDB)*, 2006.

[CTG12] Héctor Carretié, Beatriz Torvisco, and Roberto García. Using Semantic Web Technologies to Facilitate XBRL-based Financial Data Comparability. In *International Workshop on Finance and Economics on the Semantic Web*, 2012.

[CYZ+08] Chen Chen, Xifeng Yan, Feida Zhu, Jiawei Han, and Philip S. Yu. Graph OLAP: Towards Online Analytical Processing on Graphs. In *8th IEEE International Conference on Data Mining (ICDM)*, 2008.

[dACCG+13] Cristina Dutra de Aguiar Ciferri, Ricardo Rodrigues Ciferri, Leticia I. Gómez, Markus Schneider, Alejandro A. Vaisman, and Esteban Zimányi. Cube Algebra: A Generic User-Centric Model and Query Language for OLAP Cubes. *International Journal of Data Warehousing and Mining*, 9(2), 2013.

[DCSW09] Umeshwar Dayal, Malu Castellanos, Alkis Simitsis, and Kevin Wilkinson. Data Integration Flows for Business Intelligence. In *12th International Conference on Extending Database Technology Advances in Database Technology (EDBT)*, 2009.

[DdAF+10] Roger Debreceny, d'Eri Alessandro, Carsten Felden, Stephanie Farewell, and Maciej Piechocki. Feeding the Information Value Chain: Deriving Analytical Ratios from XBRL filings to the SEC. Technical report, School of Accountancy Shidler College of Business, Hawaii, 2010.

[DKSU11] Songyun Duan, Anastasios Kementsietsidis, Kavitha Srinivas, and Octavian Udrea. Apples and Oranges: a Comparison of RDF Benchmarks and Real RDF Datasets. In *ACM International Conference on Management of Data (SIGMOD)*, 2011.

[DP08] Claudia Diamantini and Domenico Potena. Semantic Enrichment of Strategic Datacubes. In *11th ACM International Workshop on Data Warehousing and OLAP (DOLAP)*, 2008.

[DP10] Claudia Diamantini and Domenico Potena. Exploring Strategic Indexes by Semantic OLAP Operators. In *Management of the Interconnected World*. Physica Verlag Heidelberg, 2010.

[DP11] Claudia Diamantini and Domenico Potena. Thinking Structurally Helps Business Intelligence Design. In *Information Technology and Innovation Trends in Organizations*. Physica Verlag Heidelberg, 2011.

[DPS13] Claudia Diamantini, Domenico Potena, and Emanuele Storti. A Logic-Based Formalization of KPIs for Virtual Enterprises. In *Advanced Information Systems Engineering Workshops*, 2013.

[DPS14] Claudia Diamantini, Domenico Potena, and Emanuele Storti. Data Mart Reconciliation in Virtual Innovation Factories. In *Advanced Information Systems Engineering Workshops*, 2014.

[DR11] Aba-sah Dadzie and Matthew Rowe. Approaches to Visualising Linked Data: A Survey. *Semantic Web*, 2(2), 2011.

[EAS13] Ivan Ermilov, Sören Auer, and Claus Stadler. User-driven Semantic Mapping of Tabular Data. In *9th International Conference on Semantic Systems (I-SEMANTICS)*, 2013.

[Erl10] Orri Erling. Directions and Challenges for Semdata. In *Workshop on Semantic Data Management at VLDB (SemData@VLDB)*, 2010.

[Erl12] Orri Erling. Virtuoso, a Hybrid RDBMS/Graph Column Store. *IEEE Data Engineering Bulletin*, 35(1), 2012.

[ETBL12] Julian Eberius, Maik Thiele, Katrin Braunschweig, and Wolfgang Lehner. DrillBeyond: Enabling Business Analysts to Explore the Web of Open Data. *VLDB Endowment*, 5(12), 2012.

[EV12a] Lorena Etcheverry and Alejandro A. Vaisman. Enhancing OLAP Analysis with Web Cubes. In *9th Extended Semantic Web Conference (ESWC)*, 2012.

[EV12b] Lorena Etcheverry and Alejandro A. Vaisman. QB4OLAP: A Vocabulary for OLAP Cubes on the Semantic Web. In *International Workshop on Consuming Linked Data (COLD)*, 2012.

[EV12c] Lorena Etcheverry and Alejandro A. Vaisman. Views over RDF Datasets: A State-of-the-Art and Open Challenges. *The Computing Research Repository (CoRR)*, Nov 2012.

[FKaGO+12] André Freitas, Benedikt Kämpgen, Jo ao Gabriel Oliveira, Seán O'Riain, and Edward Curry. Representing Interoperable Provenance Descriptions for ETL Workflows. In *3rd International Workshop on Semantic Web in Provenance Management at ESWC*, 2012.

[Fow96] Martin Fowler. *Analysis Patterns: Reusable Object Models*. Addison-Wesley, Menlo Park, California (USA), 1996.

[FTC11] Corentin Follenfant, David Trastour, and Olivier Corby. A Model for Assisting Business Users along Analytical Processes. In *2nd Workshop on Semantic Personalized Information Management: Retrieval and Recommendation*, 2011.

[GAW+08] Bugra Gedik, Henrique Andrade, Kun-Lung Wu, Philip S. Yu, and MyungCheol Doo. SPADE: The System S Declarative Stream Processing Engine. In *ACM International Conference on Management of Data (SIGMOD)*, 2008.

[GCB+97] Jim Gray, Surajit Chaudhuri, Adam Bosworth, Andrew Layman, Don Reichart, Murali Venkatrao, Frank Pellow, and Hamid Pirahesh. Data Cube: A Relational Aggregation Operator Generalizing Group-By,

Cross-Tab, and Sub-Totals. *Data Mining and Knowledge Discovery*, 1(1), 1997.

[Gen10] Michael R. Genesereth. *Data Integration: The Relational Logic Approach*. Morgan & Claypool Publishers, San Rafael, California (USA), 2010.

[GG10] Roberto García and Rosa Gil. Triplificating and linking XBRL financial data. In *6th International Conference on Semantic Systems (I-SEMANTICS)*, 2010.

[GGV12] Leticia I. Gómez, Silvia A. Gómez, and Alejandro A. Vaisman. A Generic Data Model and Query Language for Spatiotemporal OLAP Cube Analysis. In *15th International Conference on Extending Database Technology (EDBT)*, 2012.

[GKLM11] François Goasdoué, Konstantinos Karanasos, Julien Leblay, and Ioana Manolescu. View Selection in Semantic Web Databases. *VLDB Endowment*, 5(2), 2011.

[GM95] Ashish Gupta and Inderpal Singh Mumick. Maintenance of Materialized Views: Problems, Techniques, and Applications. *IEEE Data Engineering Bulletin*, 18(3), 1995.

[GMP+12] Matteo Golfarelli, Federica Mandreoli, Wilma Penzo, Stefano Rizzi, and Elisa Turricchia. OLAP query reformulation in peer-to-peer data warehousing. *Information Systems*, 37(5), 2012.

[GMUW08] Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer Widom. *Database Systems: The Complete Book*. Prentice Hall, Upper Saddle River, NJ, USA, 2 edition, 2008.

[Gra93] Goetz Graefe. Query Evaluation Techniques for Large Databases. *ACM Computing Surveys (CSUR)*, 25(2), 1993.

[GWK+10] Steffen Gebhardt, Thilo Wehrmann, Verena Klinger, Ingo Schettler, Juliane Huth, Claudia Künzer, and Stefan Dech. Improving Data Management and Dissemination in Web Based Information Systems by Semantic Enrichment of Descriptive Data Aspects. *Computers & Geosciences*, 36(10), 2010.

[Har10] Andreas Harth. VisiNav: A system for visual search and navigation on web data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 8(4), 2010.

[HBF09] Olaf Hartig, Christian Bizer, and Johann-Christoph Freytag. Executing SPARQL Queries over the Web of Linked Data. In *8th International Semantic Web Conference (ISWC)*, 2009.

[HBH03] Wolfgang Hümmer, Andreas Bauer, and Gunnar Harde. XCube – XML for Data Warehouses. In *6th ACM International Workshop on Data Warehousing and OLAP (DOLAP)*, 2003.

[HGM05] Carlos A. Hurtado, Claudio Gutierrez, and Alberto O. Mendelzon. Capturing Summarizability with Integrity Constraints in OLAP. *ACM Transactions on Database Systems (TODS)*, 30(3), 2005.

[HHD04] Aidan Hogan, Andreas Harth, and Stefan Decker. Performing Object Consolidation on the Semantic Web Data Graph. In *Workshop at 16th International World Wide Web Conference (WWW)*, 2004.

[HHK+10] Andreas Harth, Katja Hose, Marcel Karnstedt, Axel Polleres, Kai-Uwe Sattler, and Jürgen Umbrich. Data Summaries for On-Demand Queries over Linked Data. In *19th International Conference on World Wide Web (WWW)*, 2010.

[HHR+09] Michael Hausenblas, Wolfgang Halb, Yves Raimond, Lee Feigenbaum, and Danny Ayers. SCOVO: Using Statistics on the Web of Data. In *6th European Semantic Web Conference (ESWC)*, 2009.

[HHU+11] Aidan Hogan, Andreas Harth, Jürgen Umbrich, Sheila Kinsella, Axel Polleres, and Stefan Decker. Searching and browsing Linked Data with SWSE: The Semantic Web Search Engine. *Web Semantics: Science, Services and Agents on the World Wide Web*, 9(4), 2011.

[HKR09] Pascal Hitzler, Markus Krötzsch, and Sebastian Rudolph. *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC, London, 2009.

[HKS+13] Andreas Harth, Craig A. Knoblock, Steffen Stadtmüller, Rudi Studer, and Pedro Szekely. On-the-fly Integration of Static and Dynamic Linked Data. In *ISWC Workshop on Consuming Linked Data (COLD)*, 2013.

[HMPR04] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. Design Science in Information Systems Research. *MIS Quarterly*, 28(1), 2004.

[Hoe13] Patrick Hoefler. Linked Data Interfaces for Non-expert Users. In *10th Extended Semantic Web Conference (ESWC)*, 2013.

[HRU96] Venky Harinarayan, Anand Rajaraman, and Jeffrey D. Ullman. Implementing Data Cubes Efficiently. In *ACM International Conference on Management of Data (SIGMOD)*, 1996.

[HS13] Stefan Hagedorn and Kai-Uwe Sattler. Efficient Parallel Processing of Analytical Queries on Linked Data. In *On the Move to Meaningful Internet Systems Conferences (OTM)*, 2013.

[HSS11] Peter Haase, Michael Schmidt, and Andreas Schwarte. The Information Workbench as a Self-Service Platform for Linked Data Applications. In *2nd International Workshop on Consuming Linked Data (COLD)*, 2011.

[IAK13] Hiroyuki Inoue, Toshiyuki Amagasa, and Hiroyuki Kitagawa. An ETL Framework for Online Analytical Processing of Linked Open Data. In *14th International Conference on Web-Age Information Management (WAIM)*, 2013.

[IUBH10] Robert Isele, Jürgen Umbrich, Christian Bizer, and Andreas Harth. LDspider: An Open-Source Crawling Framework for the Web of Linked Data. In *9th International Semantic Web Conference (ISWC) Posters and Demos*, 2010.

[KBZ86] Ravi Krishnamurthy, Haran Boral, and Carlo Zaniolo. Optimization of Nonrecursive Queries. *12th International Conference on Very Large Data Bases (VLDB)*, 1986.

[KC13] Benedikt Kämpgen and Richard 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.

[KD08] Georgi Kobilarov and Ian Dickinson. Humboldt: Exploring Linked Data. In *WWW Workshop about Linked Data on the Web (LDOW)*, 2008.

[KFvHH01] Michel Klein, Dieter Fensel, Frank van Harmelen, and Ian Horrocks. The relation between ontologies and schema-languages. *Linköping Electronic Articles in Computer and Information Science*, 6(4), 2001.

[KH11] Benedikt Kämpgen and Andreas Harth. Transforming Statistical Linked Data for Use in OLAP Systems. In *7th International Conference on Semantic Systems (I-SEMANTICS)*, 2011.

[KH12] Benedikt Kämpgen and Andreas Harth. Benchmark Document for No Size Fits All – Running the Star Schema Benchmark with SPARQL and RDF Aggregate Views – http://people.aifb.kit.edu/bka/ssb-benchmark/, 2012.

[KH13] Benedikt Kämpgen and Andreas Harth. No Size Fits All – Running the Star Schema Benchmark with SPARQL and RDF Aggregate Views. In *10th Extended Semantic Web Conference (ESWC)*, 2013.

[KH14] Benedikt Kämpgen and Andreas Harth. OLAP4LD - A Framework for Building Analysis Applications over Governmental Statistics. In *11th ESWC 2014 (ESWC2014)*, May 2014.

[KKEM10] Daniel Keim, Jörn Kohlhammer, Geoffrey Ellis, and Florian Mansmann. *Mastering the Information Age – Solving Problems with Visual Analytics*. Eurographics Association, Goslar, Germany, 2010.

[KOH12] Benedikt Kämpgen, Seán O'Riain, and Andreas Harth. Interacting with Statistical Linked Data via OLAP Operations. In *9th Extended Semantic Web Conference (ESWC) Satellite Events*, 2012.

[KR02] Ralph Kimball and Margy Ross. *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modelling*. Wiley, New York, USA, 2002.

[KRH+13] Benedikt Kämpgen, David Riepl, Bernd Herrmann, Denny Vrandecic, and Andreas Harth. Allowing Exchange, Integration and Analysis of IWRM Data via the Semantic Web. Deliverable 203 (D203), Institute of Applied Geoscience, Karlsruhe Institute of Technology (KIT), 2013.

[KRK14] Benedikt Kämpgen, David Riepl, and Jochen Klinger. SMART Research using Linked Data - Sharing Research Data for Integrated Water Resources Management in the Lower Jordan Valley. In *ESWC Workshop on Semantic Publishing (SePublica)*, 2014.

[KSH14] Benedikt Kämpgen, Steffen Stadtmüller, and Andreas Harth. Querying the Global Cube: Integration of Multidimensional Datasets from the Web. In *19th International Conference on Knowledge Engineering and Knowledge Management (EKAW)*, 2014.

[KSS12] Veit Köppen, Gunter Saake, and Kai-Uwe Sattler. *Data Warehouse Technologien*. Verlagsgruppe Hüthig-Jehle-Rehm, Heidelberg, Germany, 2012.

[KUBM12] Spyros Kotoulas, Jacopo Urbani, Peter Boncz, and Peter Mika. Robust Runtime Optimization and Skew-Resistant Execution of Analytical SPARQL Queries on Pig. In *11th International Semantic Web Conference (ISWC)*, 2012.

[KWO+14] Benedikt Kämpgen, Tobias Weller, Seán O'Riain, Craig Weber, and Andreas Harth. Accepting the XBRL Challenge with Linked Data for Financial Data Integration. In *11th Extended Semantic Web Conference (ESWC)*, 2014.

[Kä11] Benedikt Kämpgen. DC Proposal: Online Analytical Processing of Statistical Linked Data. In *11th International Semantic Web Conference (ISWC) Doctoral Consortium*, 2011.

[Lew95] James R. Lewis. IBM Computer Usability Satisfaction Questionnaires: Psychometric Evaluation and Instructions for Use. *Human-Computer Interaction*, 7(1), 1995.

[LHG04] Xiaolei Li, Jiawei Han, and Hector Gonzalez. High-Dimensional OLAP: A Minimal Cubing Approach. In *30th International Conference on Very Large Data Bases (VLDB)*, 2004.

[LHM09] Alexander Löser, Fabian Hueske, and Volker Markl. Situational Business Intelligence. In *2nd Workshop on Business Intelligence for the Real-Time Enterprise (BIRTE)*, 2009.

[LMTS06] Sergio Luján-Mora, Juan Trujillo, and Il-Yeol Song. A UML profile for multidimensional modeling in data warehouses. *Data & Knowledge Engineering*, 59(3), 2006.

[LS97] Hans-Joachim Lenz and Arie Shoshani. Summarizability in OLAP and Statistical Data Bases. In *9th International Conference on Scientific and Statistical Database Management (SSDBM)*, 1997.

[LT10] Günter Ladwig and Thanh Tran. Linked Data Query Processing Strategies. In *International Semantic Web Conference (ISWC)*, 2010.

[MCG11] Adriana Matei, Kuo-Ming Chao, and Nick Godwin. OLAP for Multidimensional Semantic Web Databases. In *Workshop on Business Intelligence for the Real-Time Enterprise (BIRTE)*, 2011.

[MG12] Abhijeet Mohapatra and Michael Genesereth. Aggregation in Datalog Under Set Semantics. Technical report, Stanford, 2012.

[MG14] Abhijeet Mohapatra and Michael Genesereth. Incremental Maintenance of Aggregate Views. In *8th International Symposium on Foundations of Information and Knowledge Systems (FoIKS)*, 2014.

[MJC+07] Jayant Madhavan, Shawn R. Jeffery, Shirley Cohen, Xin Luna Dong, David Ko, Cong Yu, and Alon Halevy. Web-scale Data Integration: You can only afford to Pay As You Go. In *3rd Conference on Innovative Data Systems Research (CIDR)*, 2007.

[MKIK07] Konstantinos Morfonios, Stratis Konakas, Yannis Ioannidis, and Nikolaos Kotsis. ROLAP Implementations of the Data Cube. *ACM Computing Surveys (CSUR)*, 39(4), 2007.

[MLT09] Jose-Norberto Mazón, Jens Lechtenbörger, and Juan Trujillo. A survey on summarizability issues in multidimensional modeling. *Data & Knowledge Engineering*, 68(12), 2009.

[MML+13] James P. McCusker, Deborah L. McGuinness, Jeongmin Lee, Chavon Thomas, Paul Courtney, Zaria Tatalovich, Noshir Contractor, Glen Morgan, and Abdul Shaikh. Towards Next Generation Health Data Exploration: A Data Cube-Based Investigation into Population Statistics for Tobacco. In *46th Hawaii International Conference on System Sciences (HICSS)*, 2013.

[MTSP07] Jose-Norberto Mazón, Juan Trujillo, Manuel Serrano, and Mario Piattini. Improving the Development of Data Warehouses by Enriching Dimension Hierarchies with WordNet. In *VLDB Conference on Ontologies-based Databases and Information Iystems (ODBIS)*, 2007.

[MVC+12] Gabriela Montoya, Maria-Esther Vidal, Oscar Corcho, Edna Ruckhaus, and Carlos Buil-Aranda. Benchmarking Federated SPARQL Query Engines: Are Existing Testbeds Enough? In *International Semantic Web Conference (ISWC)*, 2012.

[MZ06] Elzbieta Malinowski and Esteban Zimányi. Hierarchies in a multidimensional model: From conceptual modeling to logical representation. *Data & Knowledge Engineering*, 59(2), 2006.

[NB12] Victoria Nebot and Rafael Berlanga. Building data warehouses with semantic web data. *Decision Support Systems*, 52(4), 2012.

[NBP+09] Victoria Nebot, Rafael Berlanga, Juan Manuel Pérez, María José Aramburu, and Torben Bach Pedersen. Multidimensional Integrated Ontologies: A Framework for Designing Semantic Data Warehouses. *Data Semantics XIII*, 5530, 2009.

[NM11] Thomas Neumann and Guido Moerkotte. Characteristic sets: Accurate cardinality estimation for RDF queries with multiple joins. In *27th International Conference on Data Engineering (ICDE)*, 2011.

[NN09] Marko Niinimäki and Tapio Niemi. An ETL Process for OLAP Using RDF/OWL Ontologies. *Data Semantics XIII*, 5530, 2009.

[NN10] Tapio Niemi and Marko Niinimäki. Ontologies and summarizability in OLAP. In *ACM Symposium on Applied Computing (SAC)*, 2010.

[NSL11] Bernd Neumayr, Michael Schrefl, and Konrad Linner. Semantic Cockpit: An Ontology-Driven, Interactive Business Intelligence Tool for Comparative Data Analysis. In *Workshops on Advances in Conceptual Modeling. Recent Developments and New Directions*, 2011.

[OCB+13] Seán O'Riain, Barry Coughlan, Paul Buitelaar, Thierry Declerk, Uli Krieger, and Susan Marie-Thomas. Cross-Lingual Querying and Comparison of Linked Financial and Business Data. In *Extended Semantic Web Conference (ESWC) Satellite Events*, 2013.

[OCH12] Seán O'Riain, Edward Curry, and Andreas Harth. XBRL and open data for global financial ecosystems: A linked data approach. *International Journal of Accounting Information Systems*, 13(2), 2012.

[OOC09] Pat O'Neil, Betty O'Neil, and Xuedong Chen. Star Schema Benchmark – Revision 3. Technical report, UMass, Boston (USA), June 2009.

[Pau12] Heiko Paulheim. Generating Possible Interpretations for Statistics from Linked Open Data. In *9th Extended Semantic Web Conference (ESWC)*, 2012.

[PBAP08] Juan Manuel Pérez, Rafael Berlanga, María José Aramburu, and Torben Bach Pedersen. Integrating Data Warehouses with Web Data: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 20(7), 2008.

[PFH06] Axel Polleres, Cristina Feier, and Andreas Harth. Rules with Contextually Scoped Negation. In *3rd European Semantic Web Conference (ESWC)*, 2006.

[PGSJ09] Torben Bach Pedersen, Junmin Gu, Arie Shoshani, and Christian S. Jensen. Object-extended OLAP querying. *Data & Knowledge Engineering*, 68(5), 2009.

[PHDU13] Axel Polleres, Aidan Hogan, Renaud Delbru, and Jürgen Umbrich. RDFS and OWL Reasoning for Linked Data. In *Reasoning Web. Semantic Technologies for Intelligent Data Access*. Springer Berlin Heidelberg, 2013.

[PJD01] Torben Bach Pedersen, Christian S. Jensen, and Curtis E. Dyreson. A foundation for capturing and querying complex multidimensional data. *Information Systems*, 26(5), 2001.

[PM11] Jesús Pardillo and Jose-Norberto Mazón. Using Ontologies for the Design of Data Warehouses. *International Journal of Database Management Systems (IJDMS)*, 3(2), 2011.

[PMA12] Nicolas Prat, Imen Megdiche, and Jacky Akoka. Multidimensional Models Meet the Semantic Web: Defining and Reasoning on OWL-DL Ontologies for OLAP. In *15th International Workshop on Data Warehousing and OLAP (DOLAP)*, 2012.

[PMT08] Jesús Pardillo, Jose-Norberto Mazón, and Juan Trujillo. Bridging the Semantic Gap in OLAP Models: Platform-independent Queries. In *11th ACM International Workshop on Data Warehousing and OLAP (DOLAP)*, 2008.

[RA07a] Oscar Romero and Alberto Abelló. Automating Multidimensional Design from Ontologies. In *10th ACM International Workshop on Data Warehousing and OLAP (DOLAP)*. ACM Press, 2007.

[RA07b] Oscar Romero and Alberto Abelló. On the Need of a Reference Algebra for OLAP. In *9th International Conference on Data Warehousing and Knowledge Discovery (DaWaK)*, 2007.

[RALT06] Stefano Rizzi, A Abelló, J Lechtenbörger, and Juan Trujillo. Research in Data Warehouse Modeling and Design: Dead or Alive? In *9th ACM International Workshop on Data Warehousing and OLAP (DOLAP)*, 2006.

[Rie13] David Riepl. *Knowledge-Based Decision Support for Integrated Water Resources Management with an application for Wadi Shueib, Jordan*. KIT Scientific Publishing, Karlsruhe, Karlsruhe (Germany), 2013.

[RMA+11] Oscar Romero, Patrick Marcel, Alberto Abelló, Verónika Peralta, and Ladjel Bellatreche. Describing Analytical Sessions Using a Multidimensional Algebra. In *13th International Conference on Data Warehousing and Knowledge Discovery (DaWaK)*, 2011.

[RPM+13] Lívia Ruback, Marcia Pesce, Sofia Manso, Sérgio Ortiga, Percy E. Rivera Salas, and Marco A. Casanova. A Mediator for Statistical Linked Data. *28th Annual ACM Symposium on Applied Computing (SAC)*, 2013.

[RTZ08] Dariush Riazati, James A. Thom, and Xiuzhen Zhang. Drill Across & Visualization of Cubes with Non-Conformed Dimensions. In *19th Australian Database Conference (ADC)*, 2008.

[RTZ11] Dariush Riazati, James A. Thom, and Xiuzhen Zhang. Enforcing Strictness in Integration of Dimensions: Beyond Instance Matching. In *14th ACM International Workshop on Data Warehousing and OLAP (DOLAP)*, 2011.

[Saa80] Thomas L. Saaty. *The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation*. McGraw-Hill, New York (USA), 1980.

[SBC+07] Michael Stonebraker, Chuck Bear, Ugur Cetintemel, Mitch Cherniack, Tingjian Ge, Nabil Hachem, Stavros Harizopoulos, John Lifter, Jennie Rogers, and Stan Zdonik. One Size Fits All? – Part 2: Benchmarking Results. In *3rd International Conference on Innovative Data Systems Research (CIDR)*, 2007.

[SDN00] Amit Shukla, Prasad Deshpande, and Jeffrey Naughton. Materialized View Selection for Multi-cube Data Models. In *7th International Conference on Extending Database Technology (EDBT)*, 2000.

[Shn96] Ben Shneiderman. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In *IEEE Symposium on Visual Languages (VL)*, 1996.

[Sie12] Dominik Siegele. Constructing OLAP hierarchies from Statistical Linked Data. Master's thesis (Master Abschlussarbeit), Institute AIFB, Karlsruhe Institute of Technology (KIT), Germany, February 2012.

[Sim96] Herbert A. Simon. *The Sciences of the Artificial*. MIT Press, MA (USA), 1996.

[SMDMM+12] Percy E. Rivera Salas, Fernando Maia Da Mota, Michael Martin, Sören Auer, Karin Breitman, and Marco A. Casanova. Publishing Statistical Data on the Web. In *6th IEEE International Conference on Semantic Computing*, 2012.

[Spi10] Marcus Spies. An ontology modelling perspective on business reporting. *Information Systems*, 35(4), 2010.

[SSHS13] Steffen Stadtmüller, Sebastian Speiser, Andreas Harth, and Rudi 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.

[SSR94] Michael Siegel, Edward Sciore, and Arnon Rosenthal. Using Semantic Values to Facilitate Interoperability Among Heterogeneous Information Systems. *ACM Transactions on Database Systems (TODS)*, 19(2), 1994.

[STH02] Chris Stolte, Diane Tang, and Pat Hanrahan. Query, Analysis, and Visualization of Hierarchically Structured Data using Polaris. In *8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2002.

[TC05] F. Tseng and C. Chen. Integrating heterogeneous data warehouses using XML technologies. *Journal of Information Science*, 31(3), 2005.

[Tor08] Riccardo Torlone. Two approaches to the integration of heterogeneous data warehouses. *Distributed and Parallel Databases*, 23(1), 2008.

[TPL08] Christian Thomsen, Torben Bach Pedersen, and Wolfgang Lehner. RiTE: Providing On-Demand Data for Right-Time Data Warehousing. In *24th IEEE International Conference on Data Engineering (ICDE)*, 2008.

[Vas98] Panos Vassiliadis. Modeling Multidimensional Databases, Cubes and Cube Operations. In *10th International Conference on Scientific and Statistical Database Management (SSDBM)*, 1998.

[VLH+10] Denny Vrandečić, Christoph Lange, Michael Hausenblas, Jie Bao, and Li Ding. Semantics of Governmental Statistics Data. In *Web Science Conference (WebSci)*, 2010.

[VS99] Panos Vassiliadis and Timos Sellis. A Survey of Logical Models for OLAP Databases. *ACM SIGMOD Record*, 28(4), 1999.

[VS09] Panos Vassiliadis and Alkis Simitsis. Near Real Time ETL. In *New Trends in Data Warehousing and Data Analysis*. Springer, 2009.

[VTBL13] Elena Vasilyeva, Maik Thiele, Christof Bornhövd, and Wolfgang Lehner. Leveraging Flexible Data Management with Graph Databases. *1st International Workshop on Graph Data Management Experiences and Systems (GRADES)*, 2013.

[WB97] Ming-Chuan Wu and Alejandro P. Buchmann. Research Issues in Data Warehousing. In *Datenbanksysteme in Büro, Technik und Wissenschaft*. Springer, 1997.

[WC12] Cord Wiljes and Philipp Cimiano. Linked Data for the Natural Sciences: Two Use Cases in Chemistry and Biology. In *ESWC Workshop on Semantic Publishing (SePublica)*, 2012.

[Wel13] Tobias Weller. Linking SEC XBRL Data With Yahoo! Finance Data for Online Analytical Processing. Bachelor's thesis (Bachelor Abschlussarbeit), Institute AIFB, Karlsruhe Institute of Technology, Germany, May 2013.

[WPWCM11] Marcin Wylot, Jigé Pont, Mariusz Wisniewski, and Philippe Cudré-Mauroux. dipLODocus – Short and Long-Tail RDF Analytics for Massive Webs of Data. In *10th International Semantic Web Conference (ISWC)*, 2011.

[WS11] Kevin Wilkinson and Alkis Simitsis. Designing Integration Flows Using Hypercubes. In *14th International Conference on Extending Database Technology (EDBT/ICDT)*, 2011.

[WTB11] Mitchell Wenger, Manoj Thomas, and Jeffrey S. Babb. An Ontological Approach to XBRL Financial Statement Reporting. In *AMCIS Proceedings – All Submissions*, 2011.

[YP04] Xuepeng Yin and Torben Bach Pedersen. Evaluating XML-extended OLAP Queries Based on a Physical Algebra. *7th ACM International Workshop on Data Warehousing and OLAP (DOLAP)*, 2004.

[ZLXH11] Peixiang Zhao, Xiaolei Li, Dong Xin, and Jiawei Han. Graph Cube: On Warehousing and OLAP Multidimensional Networks. In *ACM International Conference on Management of Data (SIGMOD)*, 2011.

[ZM14] Benjamin Zapilko and Brigitte Mathiak. Object Property Matching Utilizing the Overlap between Imported Ontologies. In *11th Extended Semantic Web Conference (ESWC)*, 2014.

Index

If numeric data from the Web are brought together, natural scientists can compare sensor measurements with hydrological estimations, financial analysts can evaluate companies based on balance sheets and daily stock market values, and citizens can explore the GDP per Capita independently from several data sources. However, heterogeneities between datasets such as varying dimensions and different identifiers for the same entities remain a problem. Also, RDF graphs from the Web describing numerous and large datasets as well as arbitrary background information render analytical queries more complex than typical data analysis settings. This work presents methods to query a uniform view - the Global Cube - of available datasets from the Web. Contributions build on standard Linked Data vocabularies, existing OLAP and SPARQL engines, materialisation of aggregate views, and explicit mathematical relationships between datasets. Resulting approaches are applied to three different scenarios.

ISBN 978-3-7315-0379-8