Pontifícia Universidade Católica
do Rio de Janeiro

Percy Enrique Rivera Salas

OLAP2Datacube: An On-Demand transformation Framework from OLAP to RDF Data Cubes

TESE DE DOUTORADO

Thesis presented to the Programa de Pós-Graduação em Informática of the Departamento de Informática, PUC-Rio as partial fulfillment of the requirements for the degree of Doutor em Ciências - Informática

Advisor: Prof. Marco Antonio Casanova

Rio de Janeiro
September 2015

Pontifícia Universidade Católica
do Rio de Janeiro

Percy Enrique Rivera Salas

OLAP2Datacube: An On-Demand transformation Framework from OLAP to RDF Data Cubes

Thesis presented to the Programa de Pós-Graduação em Informática, of the Departamento de Informática do Centro Técnico Científico da PUC-Rio, as partial fulfillment of the requirements for the degree of Doutor.

Prof. Marco Antonio Casanova
Advisor
Departamento de Informática — PUC–Rio

Prof[a]. Karin Koogan Breitman
EMC

Prof[a]. Marta Lima de Queiros Mattoso
UFRJ

Prof. Fabio Andre Machado Porto
LNCC

Prof[a]. Giseli Rabello Lopes
UFRJ

Prof. José Eugenio Leal
Coordinator of the Centro Técnico Científico da PUC–Rio

Rio de Janeiro September 18^{th}, 2015

Percy Enrique Rivera Salas

Graduated in Systems Engineering from Universidad Católica de Santa Maria (UCSM), Arequipa – Peru in 2009. He received his master's degree in Computer Science at Pontifical Catholic University of Rio de Janeiro (PUC-Rio) on Databases and Semantic Web in 2011. Currently, he is a researcher at EMC Brazil Research and Development Center working with Cloud Computing and Big Data technologies. His current research areas are Databases, Semantic Web and Linked Data.

Bibliographic data

Rivera Salas, Percy Enrique

OLAP2Datacube: An On-Demand transformation Framework from OLAP to RDF Data Cubes / Percy Enrique Rivera Salas; advisor: Marco Antonio Casanova. – 2015.

95 f.: il. (color.) ; 30 cm

Tese (Doutorado em Informática) – Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2015.

Inclui bibliografia

1. Informática – Teses. 2. Operações OLAP. 3. Mashup de Dados. 4. R2RML. 5. Triplification. 6. Integração de Dados. 7. Modelos Multidimensionais. 8. Linked Data. 9. Dados Estatísticos. I. Casanova, Marco Antonio. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 510

Acknowledgments

First and foremost, I would like to express my sincerest gratitude to my advisor, Professor Dr. Marco Antonio Casanova, whose encouragement, guidance and everyday kindness made possible the realisation of this work.

I also would like to make a special reference to Dr. Karin Breitman, who unconditionally helped and supported me during my studies. Her advice has been crucial to my professional and academic growth.

I owe my deepest gratitude to my beloved family, for encouraging me to face the challenges and for giving me the strength to carry on.

I would also like to thank my beloved partner and future wife, Clemence, for her love, valuable advice, strength and continuous support, standing by my side without fail, either from here or from across the world.

I remain indebted to many colleagues and professors at PUC–Rio, Globo.com and EMC for providing me the means to learn and understand.

To CAPES, for the financial support, without which this work would not have been possible.

Abstract

Rivera Salas, Percy Enrique; Casanova, Marco Antonio. **OLAP2Datacube: An On-Demand transformation Framework from OLAP to RDF Data Cubes**. Rio de Janeiro, 2015. 95p. DSc Thesis, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Statistical data is one of the most important sources of information, relevant to a large number of stakeholders in the governmental, scientific and business domains alike. A statistical data set comprises a collection of observations made at some points across a logical space and is often organized as what is called a data cube. The proper definition of the data cubes, especially of their dimensions, helps processing the observations and, more importantly, helps combining observations from different data cubes. In this context, the Linked Data principles can be profitably applied to the definition of data cubes, in the sense that the principles offer a strategy to provide the missing semantics of the dimensions, including their values. In this thesis we describe the process and the implementation of a mediation architecture, called OLAP2DataCube On Demand, which helps describe and consume statistical data, exposed as RDF triples, but stored in relational databases. The tool features a catalogue of Linked Data Cube descriptions, created according to the Linked Data principles. The catalogue has a standardized description for each data cube actually stored in each statistical (relational) database known to the tool. The tool offers an interface to browse the linked data cube descriptions and to export the data cubes as RDF triples, generated on demand from the underlying data sources. We also discuss the implementation of sophisticated metadata search operations, OLAP data cube operations, such as slice and dice, and data cube mashup operations that create new cubes by combining other cubes.

Keywords

OLAP Operators; Data Cubes Mashup; R2RML; Triplification; Database Integration; Multidimensional Model; Linked Data; Statistical Data.

Resumo

Rivera Salas, Percy Enrique; Casanova, Marco Antonio. **OLAP2Datacube: Um Framework para Transformações em Tempo de Execução de OLAP para Cubos de Dados em RDF**. Rio de Janeiro, 2015. 95p. Tese de Doutorado — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Dados estatísticos são uma das mais importantes fontes de informações, relevantes para um grande número de partes interessadas nos domínios governamentais, científicos e de negócios. Um conjunto de dados estatísticos compreende uma coleção de observações feitas em alguns pontos através de um espaço lógico e muitas vezes é organizado como cubos de dados. A definição adequada de cubos de dados, especialmente das suas dimensões, ajuda a processar as observações e, mais importante, ajuda a combinar observações de diferentes cubos de dados. Neste contexto, os princípios de Linked Data podem ser proveitosamente aplicados na definição de cubos de dados, no sentido de que os princípios oferecem uma estratégia para fornecer a semântica ausentes nas dimensões, incluindo os seus valores. Nesta tese, descrevemos o processo e a implementação de uma arquitetura de mediação, chamada OLAP2DataCube On Demand Framework, que ajuda a descrever e consumir dados estatísticos, expostos como triplas RDF, mas armazenados em bancos de dados relacionais. O Framework possui um catálogo de descrições de Linked Data Cubes, criado de acordo com os princípios de Linked Data. O catálogo tem uma descrição padronizada para cada cubo de dados armazenado em bancos de dados (relacionais) estatísticos conhecidos pelo Framework. O Framework oferece uma interface para navegar pelas descrições dos Linked Data Cubes e para exportar os cubos de dados como triplas RDF geradas por demanda a partir das fontes de dados subjacentes. Também discutimos a implementação de operações sofisticadas de busca de metadados, operações OLAP em cubo de dados, tais como slice e dice, e operações de mashup sofisticadas de cubo de dados que criam novos cubos através da combinação de outros cubos.

Palavras–chave

Operações OLAP; Mashup de Dados; R2RML; Triplification; Integração de Dados; Modelos Multidimensionais; Linked Data; Dados Estatísticos.

Contents

1 Introduction **12**
- 1.1 Motivation . . . 12
- 1.2 Problem Setting . . . 13
- 1.3 Goal . . . 14
- 1.4 Contributions . . . 15
- 1.5 Organization . . . 15

2 Background **16**
- 2.1 Linked Data . . . 16
- 2.2 Database Multidimensional Representation . . . 17
- 2.3 OLAP . . . 18
- 2.4 Representation of Statistical Data in RDF . . . 20
- 2.5 The RDF Data Cube Vocabulary . . . 21
- 2.6 The QB4OLAP Vocabulary . . . 22
- 2.7 R2RML: RDB to RDF Mapping Language . . . 24

3 Related Work **26**
- 3.1 Triplification of Relational Data . . . 26
- 3.2 Frameworks for Publishing of Multidimensional and Statistical Data . . . 27
- 3.3 OLAP Manipulations on RDF Multidimensional Data . . . 28
- 3.4 Mashup Frameworks . . . 30
- 3.5 Linked Governmental Data . . . 30

4 Olap2DataCube On Demand **32**
- 4.1 Introduction . . . 32
- 4.2 Client Application . . . 34
- 4.3 Catalogue . . . 35
  - 4.3.1 Linked Data Cube Description . . . 35
  - 4.3.2 Organization of the Catalogue . . . 37
- 4.4 Mediator . . . 39
- 4.5 Wrapper . . . 40
- 4.6 Linked Data Cube Enrichment . . . 40
- 4.7 Data Cube Discovery (DCD tool) . . . 42

5 Olap2DataCube On Demand Processes **45**
- 5.1 Catalogue Construction Process . . . 45
  - 5.1.1 Overview . . . 45
  - 5.1.2 Catalogue Exploration . . . 46
  - 5.1.3 Metadata Enrichment . . . 47

5.1.4 Linked Data Cube Generation 47
5.1.5 Linked Data Cube Enrichment 48
5.2 Data Cube Consumption Process 49
5.2.1 Overview 49
5.2.2 Search and Choose 51
5.2.3 Production and Request 52
5.2.4 Transform and Respond 52

6 Olap Operations and Data Cube Mashup 54
6.1 Introduction 54
6.2 Definition of the OLAP Operations 56
6.3 OLAP Operations Implementation 58
6.3.1 Pre-Processing: Relational Metadata Extraction 58
6.3.2 A General Algorithm for Roll-up/Drill-down 59
6.3.3 A General Algorithm for Slice/Dice 62
6.3.4 Post-Processing: R2RML Mapping Rules Generation . . 63
6.4 Data Cube Mashup 66

7 Using the Olap2DataCube on Demand Framework 72
7.1 Introduction 72
7.2 Open Government Data in Brazil 72
7.3 Catalogue Construction Process 74
7.4 Data Cube Consumption Process 79

8 Conclusions and Future Work 84
8.1 Contributions 84
8.2 Publications 86
8.3 Future Work 87

Bibliography 89

List of Figures

2.1 A Star Schema [Chaudhuri et al. 1997] 18
2.2 A Snowflake Schema [Chaudhuri et al. 1997] 18
2.3 Multidimensional data [Chaudhuri et al. 1997] 19
2.4 Outline of the Data Cube Vocabulary [Cyganiak & Reynolds 2014] 22
2.5 Outline of the QB4OLAP Vocabulary [Etcheverry & Vaisman 2012] 23
2.6 An overview of R2RML [Das et al. 2012] 24
2.7 An R2RML mapping example [Das et al. 2012] 25

4.8 Olap2DataCube On Demand Architecture 33
4.9 Brazilian population multidimensional model 33

5.10 Workflow Catalogue Exploration 46
5.11 Workflow Metadata Enrichment 47
5.12 Workflow Linked Data Cube Generation 48
5.13 Workflow Linked Data Cube Enrichment 49
5.14 Workflow Search and Choose 50
5.15 Workflow Production and Request 51
5.16 Workflow Transform and Respond 52

6.17 OLAP operation general process 54
6.18 Mashup Data Cube general process 66
6.19 Conformed Dimension identification process 67
6.20 Granularity and Schema compatibility of Data Cube `C1` & `C2` . . 68
6.21 Data compatibility of Data Cube `C1` & `C2` 68
6.22 Data Cube `C1` & `C2` conformed dimensions 68
6.23 Mashup Data Cube *C3* 69

7.24 Dashboard OLAP2Datacube Framework 75
7.25 List of databases 75
7.26 Multidimensional models in database 76
7.27 Multidimensional model schema of the “population” dataset . . 76
7.28 Semantic annotation of the “state” table 77
7.29 Linked Data Cube description of the “population” dataset . . . 77
7.30 R2RML mapping rules of dimensions in the “population” dataset 78
7.31 Ontology matching tool 78
7.32 Silk link discovery tool 79
7.33 Mashup multidimensional schemes 80
7.34 Mashup visualization between the population and the mortality data cubes. 82
7.35 Radial mashup visualization 83

List of Acronyms

DCV	Data Cube Vocabulary
DW	Data warehouse
ETL	Extract, Transform and Load
GUI	Graphical User Interface
HTTP	Hypertext Transport Protocol
LOD	Linked Open Data
LDC	Linked Data Cube
OLAP	Online Analytical Processing
R2RML	RDB to RDF Mapping Language
RDF	Resource Description Framework
RDFS	RDF Schema
SCOVO	Statistical Core Vocabulary
SDMX	Statistical Data and Metadata Exchange
SKOS	Simple Knowledge Organization System
SPARQL	SPARQL Protocol and RDF Query Language
SQL	Structured Query Language
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium

Ad Majorem Dei Gloriam

San Ignacio de Loyola, *(1491 – 1556).*

1
Introduction

1.1
Motivation

Statistical data is one of the most important sources of information, relevant to a large number of stakeholders. In the governmental domain, statistical data provides an anatomy of society outlining strong and weak points of governance thus providing crucial input for policy and decision makers. In science, statistical data representing observations or measurements is often a fundamental artifact to verify or refute scientific theories. In the business domain, statistical data about product sales, market developments or economic indicators provide crucial input for strategic decisions by management. However, the elicitation of statistical data is usually quite costly in terms of time and resources, especially in scenarios where different organizations are involved. This is particularly true for public statistical data, where local, regional, state-level, national/federal and supranational organizations are involved in the definition of statistical criteria and the elicitation of statistic ground truth.

Most of the statistical data are produced by official governmental agencies and use some methodology to address issues such quality and accuracy of data. After collecting and processing these data, they are disseminated to the public to allow an overview of the data collected. This audience may be the general public, which usually displays the results in tables and charts, or advanced users, including researchers, analysts and statistical experts, who prefer to get as close as possible to raw data and view data in a format that facilitates digital analysis [Cyganiak et al. 2011].

Applications dealing with statistical data usually include Online Analytical Processing (OLAP), a set of tools and algorithms for querying large statistical databases. In OLAP, statistical data are conveniently organized as multidimensional structures known as data cubes [Etcheverry & Vaisman 2012a], which are in turn typically stored in relational databases. The advantage of using such structures is based on the possibility of obtaining different perspectives

of the data, transforming the data cubes through OLAP operations. Statistical data are frequently enriched with extensive metadata that unveil the semantics of the dimensions and of the dimension values, among others. Such metadata are essential for proper consumption of the data cubes, especially when the user is interested in comparing two or more data cubes. Furthermore, statistical data are often distributed in a simple format. For example, data cubes are commonly exported as spreadsheets with elaborate headers indicating what the columns mean.

However, both the metadata and the publication formats are typically prepared for (human) users. As a consequence, they are not adequate for software tools, which must process the metadata along with the data cubes. Standardized vocabularies and formats, such as SDMX [SDMX 2005] and the RDF Data Cube Vocabulary (DCV) [Cyganiak & Reynolds 2014], have been developed to mitigate this situation.

Although the motivation is around publishing statistical data, this cube model is very general and so the vocabularies can be used for other data sets such as survey data, spreadsheets, OLAP data cubes and multidimensional data in general.

1.2 Problem Setting

In the process of analyzing statistical data, some characteristics are essential to ensure that data are consumed in a simple but efficient way. The main characteristics are: (i) the data should be published in a simple format, without undue complexity that could become a barrier to their use, and in a standardized way, so that they can be reused and processed by automated tools; (ii) the data should be contextualized with other existing data to enrich the quality of the statistics.

In this context, the Linked Data principles [Heath & Bizer 2011] can be profitably applied to statistical data, in the sense that the principles offer a strategy to provide the missing semantics of the data. Intuitively, if followed, the Linked Data principles will include the data in a context, i.e., will connect statistical data with related data sources, creating a globally interconnected data space that enables a rich analysis of the data [Cyganiak et al. 2011]. One example would be to link demographic data collected for a region with data about the region already existing in other data sources, such as DBpedia [Auer et al. 2008]. However, it should be clear that the linkage process should be applied to the data cube dimensions, dimension values and attributes, whose

semantics must be elicited, and not to the data cube observation values.

To represent the data, the Linked Data principles recommend using RDF (Resource Description Framework). This simple and flexible model has features that facilitate data merging, even if the underlying schemas differ, and it specifically supports the evolution of schemas over time without requiring all the data consumers to be changed [Manola & Miller 2004]. The usage of RDF thus allows structured and semi-structured data to be mixed, exposed, and shared across different applications, thereby facilitating data interoperability. In fact, RDF has inspired interesting mashup tools [Bizer et al. 2007b].

1.3 Goal

In this thesis, we propose a mediation architecture, called OLAP2DataCube On Demand, which helps describe, integrate and consume statistical data, exposed as RDF triples, but stored in relational databases. The architecture features a *Catalogue* of Linked Data Cube descriptions, created according to the Linked Data principles. The *Catalogue* has a standardized description for each data cube actually stored in each statistical (relational) database known to the mediation environment. The *Mediator* offers an interface to browse the Linked Data cube descriptions, transform and integrate data cubes through OLAP operations, and export the data cubes as RDF triples, generated on demand from the underlying data sources. We also describe the different stages involved in the consumption of a Linked Data cube and discuss the representation of data cubes in RDF.

The framework proposed herein has been designed to enable a deployment environment for:

- OLAP operations (such as roll-up, drill-down, slice and dice) on multi-dimensional data cubes, which provide the users with the flexibility to view data from different perspectives.
- Dynamic mashup between multiple data cubes, to provide an aggregate view of data by combining different aspects of a phenomenon in a single representation. Recent studies show that well-crafted mashups are more than nice visual representations, but rather an important analysis tool, that enables users to make observations that are not evident from individual datasets alone.

Finally, in order to validate our approach we used a large use case of statistical government data made available by the Information Organizing Committee of

the Presidency (COI-PR)[1].

1.4 Contributions

The main contributions of this thesis are the following:

1. The overall *OLAP2Datacube on Demand* framework implementation.
2. The compilation of OLAP operations over data cubes building customized R2RML mapping rules on demand.
3. The implementation of sophisticated mashup operations to combine multiple data cubes.
4. The validation of our framework with a large-scale use case of statistical government data.

1.5 Organization

The rest of this thesis is organized as follows. In Chapter 2, we discuss the background concepts involved in the process of publishing statistical data in RDF. In Chapter 3, we discuss related work. In Chapter 4, we describe the architecture of the OLAP2DataCube on Demand framework and outline the implementation of the central modules. In Chapter 5, we describe the interaction between modules when data is consumed. In Chapter 6, we describe general transformations and data cube mashups. In Chapter 7, we validate the framework with a real case study related to publishing statistical government data. Finally, in Chapter 8, we conclude by summarizing the contributions of this work and suggesting future work.

[1] `https://i3gov.planejamento.gov.br`

2
Background

This chapter briefly reviews basic multidimensional modeling and Linked Data concepts that will be used throughout the thesis.

2.1
Linked Data

The term Linked Data refers to a set of best practices for publishing and linking structured data on the Web. These best practices [Berners-Lee 2009] enable all published data to become part of a single global data space. These principles are the following:

1. Use URIs as names for things.
2. Use HTTP URIs, so that people can look up those names.
3. When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL).
4. Include links to other URIs, so that they can discover more things.

The second principle recommends the use of HTTP URIs to name entities and concepts so that consumers of the data can lookup those URIs to get more information, including links to other related URIs. RDF [Manola & Miller 2004] provides a standard for the representation of the information that describes those entities and concepts, and is returned by dereferencing the URIs.

RDF describes resources and their relationships through triples of the form (s, p, o), where: s is the subject of the triple, which is an RDF URI reference or a blank node; p is the predicate or property of the triple, which is an RDF URI reference and specifies how s and o are related; and o is the object, which is an RDF URI reference, a literal or a blank node. A triple (s, p, o) may also be denoted as "$< s >< p >< o >$".

According to Cyganiak & Reynolds [Cyganiak & Reynolds 2014], there are a number of benefits to being able to publish multidimensional data, such as statistics, using RDF and the linked data approach:

The individual observations, and groups of observations, become (Web) addressable. This allows publishers and third parties to annotate and link to this data; for example a report can reference the specific figures it is based on allowing for fine grained provenance trace-back.

Data can be flexibly combined across sets of datasets (for example find all religious schools in census areas with high values for national indicators pertaining to religious tolerance). The statistical data becomes an integral part of the broader Web of linked data.

2.2 Database Multidimensional Representation

A *multidimensional database*, or a data warehouse (DW), is a repository of multiple heterogeneous data sources, organized under a unified schema in order to facilitate management decision making [Han 2005].

Most Data Warehouses use a *star schema*, proposed by Kimball [Kimball & Ross 2002], to represent the multidimensional data model for data warehouse design. The term "star schema" is derived from the fact that a graphical depiction of the schema resembles a star. The star schema consists of a single *fact* table in the middle connected to a number of *dimension* tables. A *fact* table contains measurement records such as the "total price" in the fact table of the star schema given in Figure 2.1. These records model the business process and provide us with measurements (or facts) in terms of the key dimensions in our data warehouse. *Dimensions* are data warehouse "subjects". For instance, dimensions in our example are City, Product, Customer and Date tables. In practice, *fact* tables are typically massive, holding perhaps billions of records (or facts), while *dimension* tables are relatively small and contain information about the entries of a particular attribute in the *fact* table.

Note that the *dimension* tables are generally denormalized, meaning that the tables maintain some of the redundancy that a good OLTP (OnLine Transaction Processing) system typically eliminates.

The *snowflake schema* provides a refinement of a star schema, where the dimensional hierarchy (levels of *dimension*) is explicitly represented by normalizing the *dimension* tables, as illustrated in Figure 2.2. This leads to advantages in maintaining the *dimension* tables. However, the denormalized structure of the dimensional tables in star schemes may be more appropriate

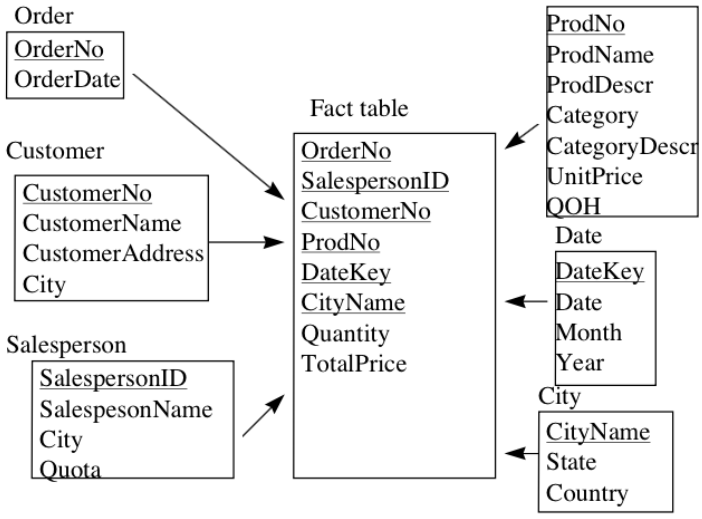


Figure 2.1: A Star Schema [Chaudhuri et al. 1997]

for browsing the *Dimensions*.

The *fact constellations schema* is an example of more a complex structure in which multiple fact tables share dimensional tables, viewed as a collection of star schemes.

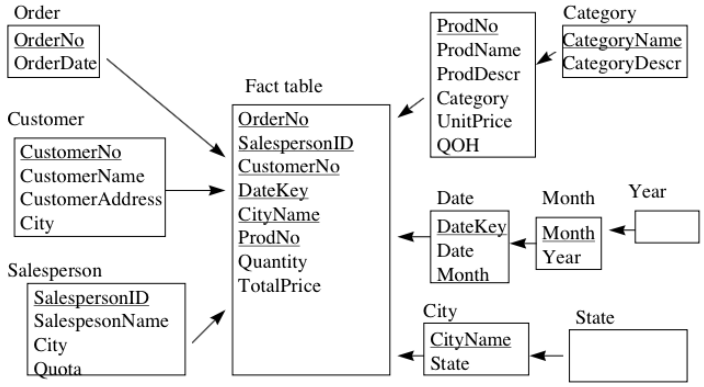


Figure 2.2: A Snowflake Schema [Chaudhuri et al. 1997]

2.3 OLAP

Business Intelligence comprises a collection of techniques used for extracting and analyzing business data to support decision-making. Applications dealing with these data usually include a set of tools and algorithms for querying large Data Warehouses. These tools are known as On-Line Analytical Processing (OLAP) [Codd et al. 1993, Etcheverry & Vaisman 2012a].

In OLAP, data are perceived as multidimensional structures known as *data cubes*. Such structures are organized according to their components: *dimensions*, *attributes* and *measures*. Each data cube contains a set of *measures* that are the objects of analysis (representing *facts*), such as the sale amount in our example (see Figure 2.3). Each measure depends on a set of *dimensions*, which provide the context for the measure. For example, the dimensions associated with a sale amount can be the city, product name, and the date when the sale was made. The dimensions together are assumed to uniquely determine the measure. Thus, multidimensional data views a measure as a value in the multidimensional space of dimensions. Each *dimension* is described by a set of *attributes*. For example, the Product dimension may consist of four attributes: the category and the industry of the product, year of its introduction, and the average profit margin.

The *dimensions* can be structured in hierarchies of *levels* that enable the analysis of the data at different levels of aggregation. The values in a dimension level are called *members*, which can also have properties. Members in a dimension level must have a corresponding member in the upper level in the hierarchy. In the above example, the city is related to its state and the country level through such a hierarchical relationship. [Chaudhuri et al. 1997]

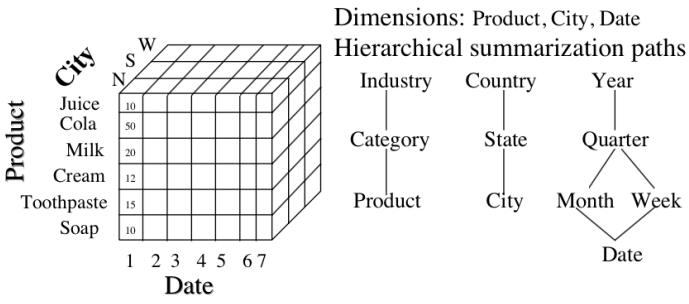


Figure 2.3: Multidimensional data [Chaudhuri et al. 1997]

Another distinctive feature of the conceptual model for OLAP is that it offers *aggregation* of measures by one or more dimensions as one of the key operations. For example, one may compute and rank the total sales by country (or by each year). Other popular operations include comparing two measures (e.g., sales and budget) aggregated by the same dimensions. Time is a dimension that is of particular significance to decision support (e.g., trend analysis).

The advantage of using such structures is the ability to obtain different perspectives of the data. One can also transform the cubes through OLAP operations, such as [Kimball & Ross 2002].

- *Roll-Up* summarizes data at a higher level in the hierarchies of a dimension, creating a cube that contains instance data on a higher aggregation level.
- *Drill-Down* summarizes data at a lower level of a dimension hierarchy, thereby viewing data in a more specialized level within a dimension. It performs the opposite of what Roll-up does. It decomposes the aggregation at a finer level of detail.
- *Slice* produces a subcube with one less dimension by choosing a single value for one of its dimensions. A *slice* is a subset of a multidimensional cube corresponding to a single attribute on one of the dimensions of the data cube while allowing the other dimensions to vary.
- *Dice* produces a subcube by allowing a filter for specific values of multiple dimensions. The *dice* operation is a *slice* on more than two dimensions of a data cube.
- *Drill-Across* link two or more fact tables at the same granularity. It is important to notice that the Drill-Across operator requires related *conformed dimensions* to combine the data cubes. A *conformed dimension* is a dimension shared among several data cubes with the same semantics.

2.4 Representation of Statistical Data in RDF

Following Cyganiak & Reynolds [Cyganiak & Reynolds 2014], a *statistical data set* comprises a collection of *observations* made at some points across some logical space. The collection can be characterized by a set of *dimensions* $d_1, \ldots, d_m$ that define what the observations apply to, along with metadata attributes $a_1, \ldots, a_n$ describing what has been measured, how it was measured and how the observation measures o are expressed. The values of each dimension d_i (of each attribute a_j or of the observation measures o) are taken from a dimension domain D_i (an attribute domain A_j or an observation measures domain O, respectively).

A statistical data set therefore defines a relation $R \subseteq D_1 \times \cdots \times D_m \times A_1 \times \cdots \times A_n \times O$, commonly referred to as a *data cube* or simply as a *cube*. A tuple of values from the dimension domains identifies an observation measure value and the associated attribute values, that is, R is actually a function of the form $R : D_1 \times \cdots \times D_m \to A_1 \times \cdots \times A_n \times O$.

According to Noy and Rector [Noy & Rector 2006], we may represent R by reification ('Pattern 1: Introducing a new class for a relation' in [Noy & Rector 2006]), that is, by creating a new class r and

treating the dimensions, attributes and observation measure as properties. Thus, a tuple $(x_1, \ldots, x_m, y_1, \ldots, y_n, z)$ in R is represented by $m + n + 1$ triples $(u, d_1, x_1), \ldots, (u, d_m, x_m), \ldots, (u, a_1, y_1), \ldots, (u, a_n, y_n), (u, o, z)$. The OLAP2DataCube approach follows this reification strategy.

Cubes are often exported as spreadsheets, which are bi-dimensional matrices. This is possible by selecting a dimension D_i and treating $R : D_1 \times \cdots \times D_m \rightarrow A_1 \times \cdots \times A_n \times O$ as a function with two arguments $R : (D_1 \times \cdots \times D_{i-1} \times D_{i+1} \times \cdots \times D_m) \times D_i \rightarrow A_1 \times \cdots \times A_n \times O$. Then, a tuple $(x_1, \ldots, x_m, y_1, \ldots, y_n, z)$ in R is represented by a tuple of values $(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_m)$ taken from the spreadsheet heading, a value x_i taken from a line of the spreadsheet and a tuple of values $(y_1, \ldots, y_n, z)$ obtained from the corresponding cell (usually just the observation measure value z). With this interpretation, one can then extract $m + n + 1$ triples $(u, d_1, x_1), \ldots, (u, d_m, x_m), \ldots, (u, a_1, y_1), \ldots, (u, a_n, y_n), (u, o, z)$ to represent the tuple $(x_1, \ldots, x_m, y_1, \ldots, y_n, z)$ in R.

2.5 The RDF Data Cube Vocabulary

The RDF Data Cube vocabulary (DCV) [Cyganiak & Reynolds 2014] is a recommendation of the W3C Government Linked Data (GLD) Working Group, focused on the publication of statistical data and metadata using RDF and adhering to Linked Data principles. The vocabulary was specifically designed to publish data cubes on the Web in such a way that it can be linked to related RDF datasets. The model that underlies the DCV is compatible with the cube model that is supported by SDMX (Statistical Data and Metadata eXchange), an ISO standard for exchanging and sharing statistical data and metadata among organizations.

DCV can be extended with other vocabularies that support the publication of additional context of statistical data, such as SKOS, SCOVO, VoID, FOAF and Dublin Core. We also remark that DCV also allows representing data cube slices, which is a very useful feature [Cyganiak & Reynolds 2014].

Very briefly, to encode structural information about the observations, DCV contains a set of concepts, such as `qb:DataStructureDefinition`, `qb:DataSet` and `qb:Slice`. It represents data cube dimensions, attributes, and measures as RDF properties. Each property is an instance of the abstract class `qb:ComponentProperty`, which in turn has sub-classes `qb:DimensionProperty`, `qb:AttributeProperty` and `qb:MeasureProperty`. Observations (in OLAP terminology, facts) are described as instances of

`qb:Observation`, representing points in a multidimensional data space indexed by dimensions.

The Figure 2.4 summary the key terms and their relationship of the DCV.

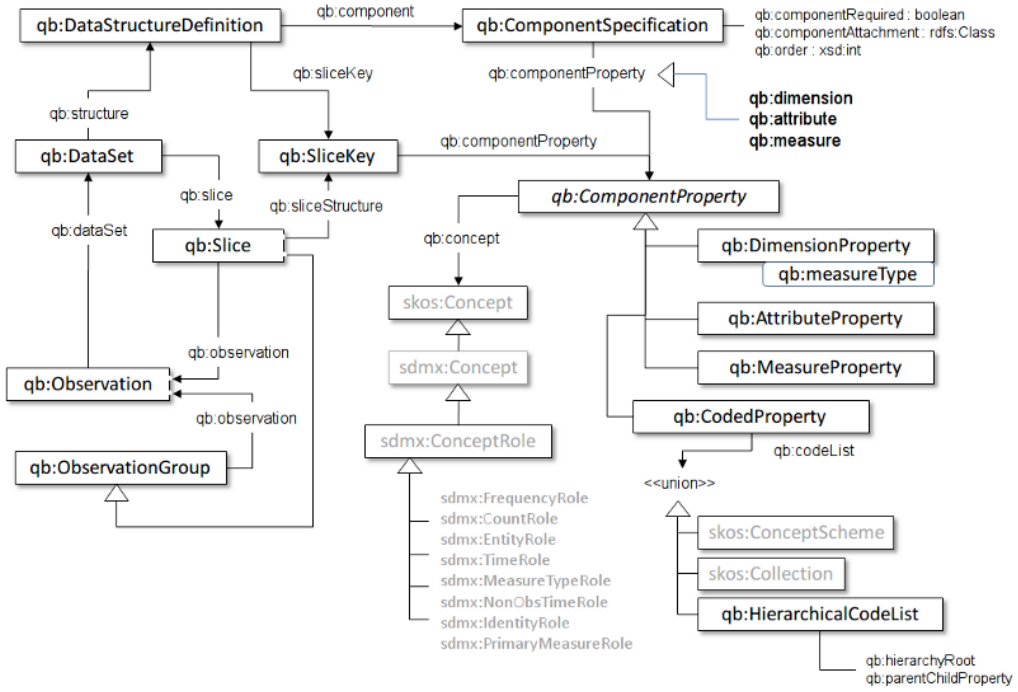


Figure 2.4: Outline of the Data Cube Vocabulary [Cyganiak & Reynolds 2014]

2.6 The QB4OLAP Vocabulary

The QB4OLAP vocabulary proposes to extend DCV in order to support the following concepts, defined in classic multidimensional models for OLAP and not modeled in DCV:

1. Dimension structure: the structure of a dimension is defined in terms of levels, which are hierarchically organized through rollup relations.

2. Dimension instances: level instances are called level members, and there is a relation between level members from different levels. In QB4OLAP the relationship between level members (from most specific to more general concepts) is modeled using the `skos:broader` property.

3. Aggregate functions: aggregate functions are used to compute measure aggregate values when performing OLAP operations (e.g: Roll-Up).

The main components of the QB4OLAP vocabulary used in this thesis are listed below:

- The class `qb4o:LevelProperty` models dimension levels.
- The property `qb4o:parentLevel` represents relations between dimension levels.
- The property `qb4o:inDimension`, links dimension levels to its corresponding dimension.
- The class `qb4o:AggregateFunction` represents aggregate functions.
- The property `qb4o:hasAggregateFunction` represents the association between measures and aggregate functions in a data cube.
- The property `qb4o:level` is used to specify the cube schema.

The Figure 2.5 shows classes, properties and instances of the QB4OLAP vocabulary (grey background).

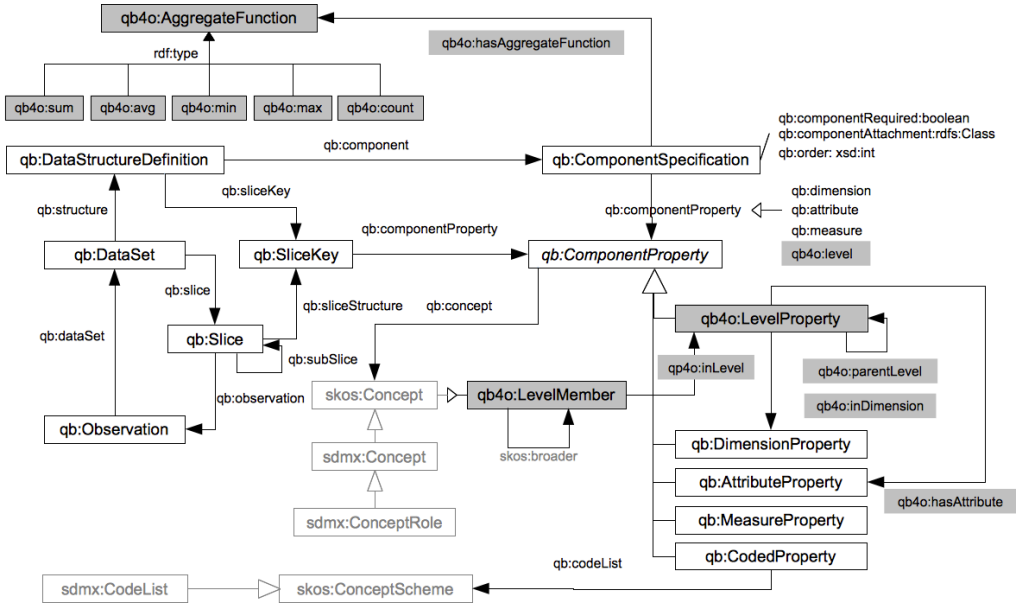


Figure 2.5: Outline of the QB4OLAP Vocabulary [Etcheverry & Vaisman 2012]

2.7 R2RML: RDB to RDF Mapping Language

The RDB to RDF Mapping Language (R2RML) [Das et al. 2012] is a W3C Recommendation since September 2012, used for expressing customized mappings from relational databases to RDF datasets created by the RDB2RDF working group. Such mappings provide the ability to view existing relational data in the RDF data model, expressed in a structure, and using a target vocabulary of the mapping author's choice. Figure 2.6 presents the main classes and properties of the vocabulary.

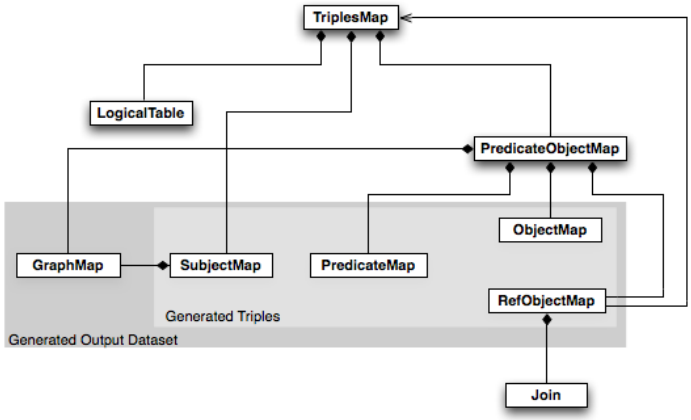


Figure 2.6: An overview of R2RML [Das et al. 2012]

The R2RML mappings are conceptual and are expressed as RDF graphs written with Turtle syntax. This language is designed to meet the use cases and requirements identified in "Use Cases and Requirements for Mapping Relational Databases to RDF" [Auer et al. 2010].

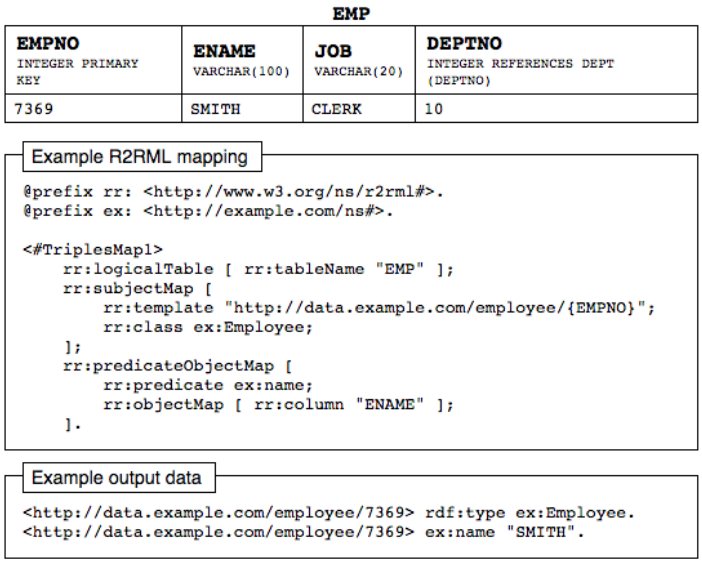
An R2RML mapping refers to logical tables to retrieve data from the input database. A logical table can be: a base table; a view; or a valid SQL query (called an "R2RML view" because it emulates an SQL view without modifying the database).

Each logical table is mapped to RDF using a *triples map*, which is a rule to map each tuple in the logical table to a set of RDF triples. Each rule is composed of two main parts: a *subject map* and multiple *predicate-object maps*.

– The *subject map* generates the subject of all RDF triples that will be generated from a logical table row. The subjects are often URIs generated from the primary key values.

– The *predicate-object maps* in turn consist of *predicate maps* and *object maps* (or referencing object maps).

Triples are produced by combining the *subject map* with a *predicate-object maps*, and applying these maps to each logical table row.



EMP

EMPNO INTEGER PRIMARY KEY	**ENAME** VARCHAR(100)	**JOB** VARCHAR(20)	**DEPTNO** INTEGER REFERENCES DEPT (DEPTNO)
7369	SMITH	CLERK	10

Example R2RML mapping

```
@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix ex: <http://example.com/ns#>.

<#TriplesMap1>
    rr:logicalTable [ rr:tableName "EMP" ];
    rr:subjectMap [
        rr:template "http://data.example.com/employee/{EMPNO}";
        rr:class ex:Employee;
    ];
    rr:predicateObjectMap [
        rr:predicate ex:name;
        rr:objectMap [ rr:column "ENAME" ];
    ].
```

Example output data

```
<http://data.example.com/employee/7369> rdf:type ex:Employee.
<http://data.example.com/employee/7369> ex:name "SMITH".
```

Figure 2.7: An R2RML mapping example [Das et al. 2012]

Figure 2.7 shows a R2RML mapping to produce RDF triples from the EMP table. The pattern to generate the subject URIs is defined by the `rr:template` property and contains the primary key of the table. The first triple is generated by the property `rr:class`, which generates the `rdfs:type` property. The object of this triple is the class `ex:Employer` of the custom target vocabulary. The second triple is composed of the predicate `ex:name` and has as an object the value of the column ENAME of the logical table.

3
Related Work

To facilitate reading, we group related work into five topics: triplification of relational data; frameworks for publishing of multidimensional and statistical data; OLAP manipulations on RDF multidimensional data; mashup frameworks; and linked governmental data.

In the triplification and publication of multidimensional and statistical data sections, we discuss related work where the main focus is on mapping of data to RDF datasets, generating RDF triples as well as methods and approaches for publishing data. In the OLAP manipulation section, we discuss the OLAP operations over RDF multidimensional data described using Data Cube Vocabulary (DCV) or related vocabularies. In the mashup section, we talk about approaches to create homogenized views of linked data sources. Finally, in the linked governmental data section, we present the lessons learned, by different governments, when publishing statistical government data in RDF.

3.1
Triplification of Relational Data

Currently most of the work in the area of triplification focuses on generating RDF from relational database content. There is a wide range of approaches developed in this regard ranging from very simple scripts such as *Triplify* [Auer et al. 2009] over standalone solutions such as *D2R* [Bizer & Cyganiak 2006] up to integrated tools such as *Virtuoso RDF Views* [Erling 2009].

Under the auspices of the W3C, the *RDB2RDF working group* is currently standardizing the *R2RML* [Das et al. 2012] mapping language for the mapping and transformation of relational data to RDF. The *DB2Triples*[2] tool, implements the R2RML and Direct Mapping standards defined by the group to convert relational database content to RDF triples.

[2] `https://github.com/antidot/db2triples`

3.2 Frameworks for Publishing of Multidimensional and Statistical Data

The Statistical Data and Metadata eXchange (SDMX, [SDMX 2005]) is an initiative started in 2001 to foster standards for the exchange of statistical information. The SDMX sponsoring institutions are the Bank for International Settlements, the European Central Bank, Eurostat, the International Monetary Fund (IMF), the Organization for Economic Co-operation and Development (OECD), the United Nations Statistics Division and the World Bank. The SDMX message formats have two basic expressions, SDMX-ML (using XML syntax) and SDMX-EDI (using EDIFACT syntax and based on the GESMES/TS statistical message). Experiences and best practices regarding the publication of statistics on the Web in SDMX have been published by the United Nations [UNECE 2000] and the Organization for Economic Cooperation and Development [OECD 2006].

The representation of statistics in RDF started with SCOVO [Hausenblas et al. 2009, Cyganiak et al. 2010a] and continued with the successor RDF Data Cube Vocabulary [Cyganiak et al. 2010a]. The Data Cube Vocabulary is closely aligned with SDMX [Cyganiak & Reynolds 2014]. Examples of statistics published as RDF adhering to the Data Cube vocabulary and visualized for human consumption include the EC's INFSO Digital Agenda Scoreboard[3] and the LOD2 Open Government Data stakeholder survey [Martin et al. 2011].

Zancanaro et al. [Zancanaro et al. 2013] describe a process for identifying sources, ontology generation, mapping and publishing statistical linked data. The authors suggest the use of domain ontologies to provide a semantic meaning to the data cube. A similar work is described by Petrou et al. [Petrou et al. 2014]. The authors describe a methodology for publishing Linked Open Statistical Data from tabular data sources or relational databases. Both works use the RDF Data Cube Vocabulary to describe the output triples.

Ghasemi et al. [Ghasemi et al. 2014] present the design and implementation of a middleware between an application and an OLAP server which manages multidimensional data. The authors propose a new vocabulary, called *Multidimensional to RDF Mapping Language (M2RML)* in reference to the R2RML mapping language. The middleware use this mapping language to transform data persisted in OLAP servers to RDF dataset described in RDF Data Cube Vocabulary.

[3] `http://ec.europa.eu/information_society/digital-agenda/scoreboard/`

Janev et al. [Janev et al. 2014] present the LOD2 Statistical Workbench, an integrated set of tools for accessing, manipulating, exploring and publishing statistical data. The data representation and processing is based on the RDF Datacube Vocabulary. The authors validate the use of the Workbench with a case study of the Statistical Office of the Republic of Serbia.

In our approach we publish multidimensional data dynamically upon request while in [Zancanaro et al. 2013, Petrou et al. 2014] and [Janev et al. 2014] they publish the entire multidimensional datasets once and expose them statically as an end-point. Unfortunately the later approach generates redundantly problems and require an extra effort maintaining both the relational data and their triplifications. Similar to our approach Ghasemi et al. in [Ghasemi et al. 2014] publish multidimensional data dynamically. However, while they consider as input an MDX query, we expect a data cube request in the form of RDF Data Cube triples best suitable for Semantic Web applications.

3.3 OLAP Manipulations on RDF Multidimensional Data

We can identify two categories of approaches for manipulating RDF multidimensional data. A first category considers an ETL process for extracting and transforming the RDF multidimensional data into a specific structure, usually a relational model, before using standard OLAP systems [Kämpgen & Harth 2011]. The second category aims at manipulating OLAP operations directly on RDF data collections without using an ETL process [Etcheverry & Vaisman 2012, Kämpgen et al. 2012, Saad et al. 2013].

In [Kämpgen & Harth 2011], a module transforms RDF data (described using RDF Data Cube Vocabulary) into a multidimensional model. The resulting structure is further manipulated using Mondrian OLAP system and MDX queries. Hence, the OLAP manipulations are performed on the multidimensional data source and not directly on the RDF data collection. The inconvenience of this approach is the ETL process has to repeat in order to propagate changes in the raw data.

In order to overcome this issue, Kämpgen et al. [Kämpgen et al. 2012] defined a mechanism for translating common OLAP operations into SPARQL queries for direct manipulation of RDF data described in RDF Data Cube Vocabulary. However, this work took into account neither the hierarchical structure at multiple levels nor the multiple hierarchies in a dimension.

Considering that the RDF Data Cube Vocabulary (DCV) was not sufficient for modeling and fully perform OLAP operations over RDF data cubes (e.g. representing multiple hierarchies in a dimension), Etcheverry & Vaisman [Etcheverry & Vaisman 2012a] defined a new RDF vocabulary called *Open Cubes (OC)* for the multidimensional modeling of RDF data. OC provides a set of classes and properties to model the different structures of the multidimensional model (dimensions, attributes, measures), including hierarchical relationships between attributes of dimensions. From RDF collections described using OC, different OLAP manipulations can be performed directly via queries expressed in SPARQL. Although this solution is based on a multidimensional modeling of RDF data and allows expressing OLAP operations in terms of SPARQL, its main limitation is the use of a specific and non-standard vocabulary. To address this issue, Etcheverry & Vaisman [Etcheverry & Vaisman 2012] proposed a new vocabulary, denoted QB4OLAP, which extends DCV to fully support multidimensional modeling including hierarchical relationships and OLAP manipulations. The authors also created algorithms to transform cubes based on QB into equivalent QB4OLAP cubes (and vice versa).

In a related work, Saad et al. [Saad et al. 2013] presented a formalization of a multidimensional model in terms of RDF data, addressing the representation of multiple hierarchies in a dimension, using RDF Data Cube, SKOS and RDFS vocabularies on the basis of this model, the authors defined a mechanism for translating common OLAP operations into SPARQL queries.

Kämpgen & Harth [Kämpgen & Harth 2014] presented *OLAP4LD*, a framework for building analysis applications with Linked Data reusing the RDF Data Cube Vocabulary (DCV). OLAP4LD provides access to multidimensional datasets published in DCV. The framework translates MDX queries, from an *olap4j* client, to SPARQL queries over Linked Data sources, and translates results from Linked Data sources to representations understandable by the client. In a similar work Ghasemi et al. [Ghasemi et al. 2014] provides access and manipulates multidimensional datasets through MDX queries and returns as the answer an RDF dataset using DCV and a new DCV extension defined by the authors.

Our approach describes a totally different category for manipulating RDF multidimensional data. In our approach the OLAP operations are performed directly over the underlying databases using standard SQL queries. Then the result of the SQL query is mapped at runtime to RDF triples using the DCV and QB4OLAP standard vocabularies.

3.4 Mashup Frameworks

Building Linked Data Mashup applications, in many cases, requires fetching and assembling pieces of information from multiple Linked Data sources (including SPARQL endpoints). Then, these pieces can be used to create an homogenized view, called a *Linked Data mashup view* [Vidal et al. 2015]. Mountantonakis et al. [Mountantonakis et al. 2014] prefer to use the term *Semantic Warehouse* to refer to this concept.

In this context, Knap et al. [Knap et al. 2012] proposed the *ODCleanStore (ODCS)* framework. This framework offers various transformations for cleaning RDF graphs (deduplication, conflict resolution), linking it to existing resources, and assessing the quality of the outcome.

Schultz et al. [Schultz et al. 2012] proposed the *LDIF - Linked Data Integration Framework*. The framework implements a mapping language, deals with URIs remapping and uses named graphs for registering data provenance. *Sieve* [Mendes et al. 2012], which is part of LDIF, assesses the quality of the integrated data and subsequently decide which values to keep, discard or transform according to user-configured metrics and fusion functions.

Tzitzikas et al. [Tzitzikas et al. 2014] described the requirements, presented a process and a tool for constructing semantic warehouses. The authors developed a tool called *MatWare* (Materialized Warehouse) to validate the warehouse construction process. They also reported their experience, using this tool, to build a semantic warehouse for the marine domain.

Vidal et al. [Vidal et al. 2015] proposed an ontology-based framework for formally specifying *Linked Data mashup views*, and a strategy for the incremental maintenance of such views, based on specifications.

In contrast with most of the aforementioned approaches, this thesis describe an automatic and on-demand mashup algorithm which checks the adequacy of the input data cubes for merging and compiles a new RDF Data Cube to be used for further exploration and analysis.

3.5 Linked Governmental Data

Several governments started to publish governmental data on the Web. Tim-Berners Lee discussed a set of Design Issues [Berners-Lee 2009] on how to publish governmental information in a re-usable way.

One of the first providers to publish governmental Linked Data was the UK Government (`http://data.gov.uk/`), hosting information about different governmental sectors of Great Britain including transportation, legislation and finance [Sheridan & Tennison 2010].

A further provider of governmental data is the US Government (`http://data.gov/`). Due to the fact that this information was not made available as Linked Data, external groups started to transform and publish the information according the Linked Data principles [Ding et al. 2010].

Hoxha et al. [Hoxha et al. 2011] described the use of a new vocabulary, called *ODA - Open Data Albania*, in the process of publishing open government data for Albania. The authors converted the raw statistical datasets into RDF triples, based on the ODA ontology, and then exposed these triples online together with a textual description.

Recent research work also aims facilitating government data ecosystems through specialized portals [Ding et al. 2011] and distributed dataset catalogs [Erickson et al. 2011]. Another important issue, which is particularly tackled by this paper for the statistics domain, is enabling interoperability of government data catalogs [Maali et al. 2010].

Breitman et al. [Breitman et al. 2012] presented the lessons learned in the publication of government data in RDF and highlighted the ease of creating mashups when the vocabulary used to describe the data is the same. The paper also presented some comparisons between the American and Brazilian data.

4
Olap2DataCube On Demand

This chapter first describes the Olap2DataCube On Demand framework architecture and introduces a running example that will be used in the remaining chapters. Then, it briefly describes each module and the interaction between them in the proposed framework. Chapter 5 describes in detail the major processes of the framework, (i) the *Catalogue Construction* process and (ii) the *Data Cube Consumption* process.

4.1
Introduction

The framework *Olap2DataCube On Demand* enables the dynamic generation, consumption and integration of statistical data, exposed as RDF triples, but stored on relational databases across different institutions. This framework provides a dynamic generation of RDF triples describing the data cube observations, which can be consumed by the user on demand. The dynamic generation of RDF triples avoids storing the complete triplification of all data cubes redundantly, which otherwise lead to problems such as storage space, velocity of consumption, synchronization and redundancy maintenance costs.

The purpose of this framework is to provide a unified deployment environment for federated queries, dynamic mashups and OLAP operations on multidimensional data cubes. The intent of this approach is to grant users or software agents access to multiple sources of independent data in an integrated fashion using the standard RDF Data Cube Vocabulary. The architecture of the proposed framework comprises the following components (see Figure 4.8): the *Client Application*, the *Linked Data Cube Enrichment*, the *Catalogue*, the *Mediator*, the *Data Cube Discovery* and *Wrappers*. Queries submitted by the client are sent to the *Mediator*, which uses information stored in the *Catalogue* to locate the data sources containing data that will produce the result of a query. Based on the information obtained from the *Catalogue*, the *Mediator* breaks down the original query into subqueries, pointing each of them via *Wrappers* to the corresponding data sources. Each *Wrapper* receives

a subquery, executes it on the data source and converts the result into RDF triples using multiple vocabularies. After recovering the RDF triples from different *Wrappers*, the *Mediator* consolidates the query response and delivers consolidated information for client consumption.

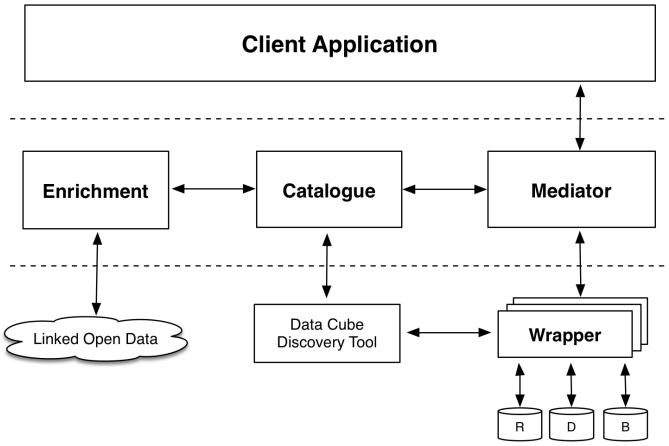


Figure 4.8: Olap2DataCube On Demand Architecture

In order to illustrate the use of the framework we will use a small multidimensional model extracted from DadosGov[4], which describes population projections in Brazil (a measure) along the dimensions geographic location and date. Let us call these dimensions *geoDim* and *dateDim*, respectively. Dimension *geoDim* is organized in a hierarchy of levels: City and State; dimension *dateDim* has only one level: year. Figure 4.9 depicts the example of a multidimensional schema using graphical notations.

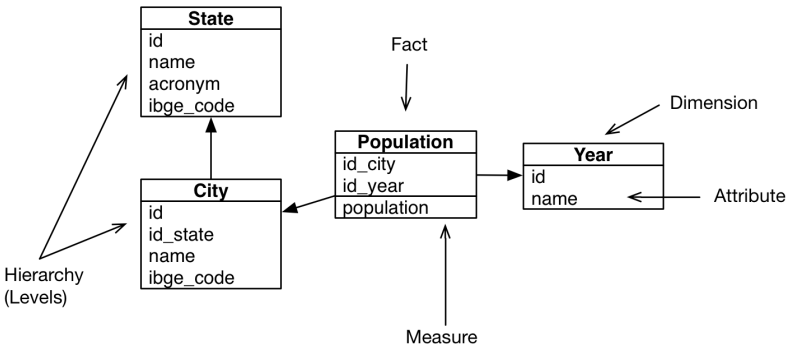


Figure 4.9: Brazilian population multidimensional model

[4] https://i3gov.planejamento.gov.br/

4.2 Client Application

The client applications that communicate with the *Mediator* are located in this module. The communication protocol between the applications and the *Mediator* is based on the RDF Data Cube Vocabulary, QB4OLAP, SKOS and RDFS vocabularies.

The *Client Application* is responsible for:

1. The construction of requests to the *Mediator* for the manipulation or creation of data cubes; and

2. Supplying mechanisms which allow the client to retrieve and manipulate the data cube structure and the records stemming from the Mediator response.

The *Client Application* should be able to request that the *Mediator*:

1. Retrieve data from a specific data cube or combined data cubes;

2. Formulate new data cubes by OLAP operations embedded in the request itself (e.g., Slice, Roll-up, Drill-down); and

3. Apply comparison functions among series of combined data cubes (e.g., correlation, percentage).

The Listing 4.1 illustrates a *Client Application* request corresponding to our running example. In this request the client would like to obtain a data cube with the total population for each state in each year. This corresponds to the application of the Roll-up operator on the city dimension.

```
1 eg:populationDS a qb:DataStructureDefinition;
2     qb:component [ qb4o:level eg:state];
3     qb:component [ qb4o:level eg:year];
4     qb:component [ qb:measure eg:population;
5                    qb4o:hasAggregateFunction qb4o:sum].
```

Listing 4.1: RDF Data Cube request.

4.3 Catalogue

4.3.1 Linked Data Cube Description

Recall from Section 2.3 that a *data cube* is organized according to a set of dimensions, measures and attributes. Collectively, they are called *components* of the data cube. A tuple of values from the dimension domains identifies an observation. A *measure* represents an *observation* value. An *attribute* allows qualifying and interpreting an observation. Attributes specify the units of measure, scaling factors and generic metadata, such as the status of the observation.

To enable the easy consumption of the data cubes, we adopt well-known vocabularies for the *Linked Data Cube* descriptions, as recommended in [Bizer et al. 2007]. We use the RDF Data Cube Vocabulary (DCV) [Cyganiak & Reynolds 2014] as the foundation to describe the *Linked Data Cube*, and some terms from the QB4OLAP vocabulary [Etcheverry & Vaisman 2012] to fully support multidimensional modeling. The DCV and the QB4OLAP are described in more detail in Sections 2.5 and 2.6 respectively. The motivation behind using the QB4OLAP vocabulary is due to the limitations of DCV in representing dimension hierarchies, that allows analysis at different aggregation levels. For instance, in our running example, consider the geoDim dimension with City, State and Country levels. Cities can be regrouped into parent levels such as States or Countries whereas the DCV dimension allows only a single granularity level. Therefore, the QB4OLAP vocabulary adds to DCV the capability of representing hierarchies through the definition of dimension levels, level members, rollup relations between dimensions, hierarchies, levels and level members, and associating aggregate functions to measures.

Consequently, we use some classes and properties from the QB4OLAP and DCV vocabularies to describe the Linked Data Cube. In addition to the classes and properties of the DCV described in Section 2.5, we use basically the classes `qb4o:LevelProperty` and `qb4o:AggregateFunction` and the properties `qb4o:parentLevel`, `qb4o:inDimension`, `qb4o:AggregateFunction` and `qb4o:level` from the QB4OLAP vocabulary. Listing 4.2 illustrates the definition of the Linked Data Cube corresponding to a single dimension, level and measure of our running example.

Together with the triples describing the dimensions, measures and attributes. A Linked Data cube description contain triples of dimension domain values (dimension instances). Listing 4.3 illustrates dimension domain values triples of city and state levels corresponding to our running example.

```
# Data Cube Description
eg:populationDS a qb:DataStructureDefinition;
    qb:component [ qb4o:dimension eg:year];
    qb:component [ qb4o:dimension eg:city];
    qb:component [ qb4o:level eg:state];
    qb:component [ qb:attribute sdmx-attribute:unitMeasure];
    qb:component [ qb:measure eg:population;
                   qb4o:hasAggregateFunction qb4o:sum].

eg:populationDataset  a qb:DataSet;
    rdfs:label "Brazilian Cities Population"@en;
    qb:structure eg:populationDS.

eg:populationMeasure a qb:MeasureProperty;
    rdfs:subPropertyOf sdmx-measure:obsValue;
    sdmx-attribute:unitMeasure foaf:Person.

# City Dimension structure
eg:city a qb:DimensionProperty;
    qb4o:parentLevel eg:State;
    rdfs:range eg:CityClass.

eg:CityClass a rdfs:Class;
    rdfs:subClassOf <http://dbpedia.org/ontology/City>;

eg:cityLabel a rdf:Property;
    rdfs:domain eg:CityClass;
    rdfs:range rdfs:Literal;
    rdfs:subProperyOf rdfs:label.

# State Level structure
eg:state a qb4o:LevelProperty;
    rdfs:range eg:StateClass.

eg:StateClass a rdfs:Class;
    rdfs:subClassOf <http://dbpedia.org/ontology/State>;

eg:stateLabel a rdf:Property;
    rdfs:domain eg:StateClass;
    rdfs:range rdfs:Literal;
    rdfs:subProperyOf rdfs:label.
```

Listing 4.2: RDF triples of the Linked Data Cube description.

```
# City Dimension Instances
ns2:58f1 a eg:CityClass;
   rdfs:label "Petropolis"@en;
   qb4o:inLevel eg:City;
   skos:broader ns1:6b2c;
   owl:sameAs <http://dbpedia.org/resource/Petropolis>;

ns2:6b2c a eg:StateClass;
   rdfs:label "Rio de Janeiro"@en;
   qb4o:inLevel eg:State;
   owl:sameAs <http://dbpedia.org/resource/Rio_de_Janeiro>
```

Listing 4.3: Dimension domain values triples.

4.3.2 Organization of the Catalogue

The *Catalogue* contains public and private data. *Public data* refers to the data cube descriptions that are exposed to the applications and linked to external data sources. *Private data* refers to the information required internally by the framework. The *Catalogue* is in turn divided into three submodules: *External Catalogue*, *Internal Catalogue* and *Triplification*.

The *External Catalogue* and *Internal Catalogue* submodules are framework data repositories. While the *External Catalogue* serves as an interface for communication with the LOD, the *Internal Catalogue* functions as a source of internal data that is essential to the operation of the other framework modules. Taking these features into account, *External Catalogue* data are stored in a triple database (TripleStore), while *Internal Catalogue* data are stored in a relational database.

The following triple groups are stored in the *External Catalogue* submodule:

- *Linked Data Cube* descriptions, i.e., sets of triples that describe data cubes in RDF. A Linked Data cube description contains triples describing the dimensions, measures, attributes and dimension domain values of a data cube. Thus, a *Linked Data Cube* description does not contain triples that capture the observations, which remain in the relational database.
- `owl:sameAs` and `owl:equivalentClass` triples that relate resources and *Linked Data Cube* descriptions with resources located in external data sources. (e.g. DBpedia, Geonames).
- Triples of dimension domain values, stored as resources, if the user wants to link such resources to those in outside data sources, again using `owl:sameAs` triples. As a simple example, consider the Country dimension, whose domain would consist of a set of URIs representing countries, linked to the respective entries in DBpedia

The *External Catalogue* is organized in such a way that *Linked Data Cube* descriptions may share the definition of dimensions, dimension domain values, measures and attributes. This simple strategy reduces the overhead of creating *Linked Data Cube* descriptions.

Metadata from the source dimensional models, essential to proper functioning of the framework modules, are stored in the *Internal Catalogue* submodule.

- Metadata from the dimensional models in the source database (e.g., table name, column name, column data type).

- Semantic annotations of domain ontologies in the dimensional model metadata (e.g, we can annotate the Country table with the class `http://dbpedia.org/ontology/Country`).
- Relationships between the metadata of the dimensional model entities and components of the *Linked Data Cube* structures, i.e., IRIs of *Linked Data Cubes* stored in the *External Catalogue* with the column/table of the source database.

These data, stored in the *Internal Catalogue*, are used by the *Mediator* to construct SQL federated queries, which are embedded in the R2RML mapping triples, and sent to the *Wrappers* for execution.

Linked Data Cube descriptions are produced in the *Triplification* submodule, i.e., a set of triples describing the structure of dimensional models and cubes in RDF triples using DCV, QB4OLAP and RDFS vocabularies. The R2RML mapping triples are also produced in the submodule to create the triples of dimension domain values, stored in the *External Catalogue*. The mappings are produced from the data stored in the *Catalogue* (Internal and External Catalogues). Metadata stored in the *Internal Catalogue* are used to create SQL statements while we use Linked Data Cube descriptions stored in the *External Catalogue* to describe the triples resulting from the SQL statement.

The Listing 4.4 illustrates the R2RML mapping triples of the City dimension from our running example.

In addition to the *Mediator*, which basically uses the data stored in this module, the *Catalogue* also interacts with two other framework modules: the *Linked Data Cube Enrichment* which aligns *Linked Data Cube* concepts and instances in the *External Catalogue* with LOD sources; and the *DCD Tool* which extracts the metadata (e.g., table name, column name, column data type) of the dimensional models in the source database and semantically records them using domain ontologies.

```
# R2RML Dimension domain values
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix qb4o: <http://purl.org/qb4olap/cubes#> .
@prefix dbpedia: <http://dbpedia.org/ontology/>

ex:tm1 a rr:TriplesMap ;
    rr:logicalTable [ rr:sqlQuery
    """
      SELECT distinct
             c.name, c.ibge_code,
             md5(concat(c.id,"City")) as md5,
             md5(concat(s.id,"State")) as md5_state
      FROM State as s, City as c
      WHERE c.id_state = s.id
    """ ] ;
    rr:subjectMap [
        rr:template "http://example.org/city/{md5}" ;
        rr:class eg:CityClass ] ;
    rr:predicateObjectMap [
        rr:predicate rdfs:label ;
        rr:objectMap [ rr:column "name" ]
    ],[
        rr:predicate dbpedia:code ;
        rr:objectMap [ rr:column "ibge_code" ]
    ],[
       rr:predicate qb4o:inLevel ;
       rr:object eg:City
    ], [
       rr:predicate skos:broader ;
       rr:objectMap
       [ rr:template
         "http://example.com/state/{md5_state}" ]
    ] .
```

Listing 4.4: R2RML mapping triples of the City dimension.

4.4 Mediator

The *Mediator*, the main module of the framework, is responsible for the communication and orchestration between the framework modules. Given the central characteristics of this module, it heavily interacts with the *Catalogue* and the *Wrapper* modules to be able to respond to requests made by client applications.

The *Mediator* is composed of three submodules: *Mashups On Demand*, *Federated Queries* and *R2RML On Demand*. These submodules are detailed below.

The *Mashups On Demand* submodule is responsible for analyzing the feasibility of mashups through the validation of certain dimension properties, for instance, the feasibility of executing OLAP operations on the data cubes for performing the mashup. A detailed description of this submodule and the complete list of dimension properties are discussed in Section 6.4.

The *Federated Queries* submodule receives the requested data cube from the *Mashups On Demand* submodule and identifies the different sources

necessary to carry out the data cube construction. For each source identified, the submodule makes a request to the *R2RML on Demand* submodule to create at runtime the set of R2RML mapping triples necessary to retrieve the instances that will compose the requested cube.

The *R2RML on Demand* submodule is invoked by the *Federated Queries* submodule. This submodule seeks further information in the *Internal and External Catalogues* to construct SQL statements, independently of the OLAP conversion involvement, which will be wrapped in R2RML mapping triples and sent to the *Wrappers* of different source databases.

4.5 Wrapper

A *Wrapper* is a module responsible for the communication between the framework and a source database where a dimensional model is stored. In order to ensure secure access to the source database, a *Wrapper* is implemented as a Web Service and split into two submodules: *Client* and *Server*.

The *Client* submodule is responsible for pointing the data retrieval requests to the correct *Server*. The *Server* submodule is configured on a server provided by the owner of the relational database where the statistical information resides. This submodule runs the DB2Triple triplification tool and a tiny Web-based application which register the connection information of the multidimensional database. This Client-Server architecture ensures the privacy of sensible data, such as the database connection information.

The *Wrapper* interacts with the *Mediator* that supplies the R2RML mapping triples. These mappings are used by the DB2Triple tool to generate triples of the observations and dimensions described by the RDF Data Cube Vocabulary. The module also interacts heavily with *DCD Tool* module described in Section 4.7. It is through the *Wrapper* that this module accesses the source databases to search its catalogues for possible existing dimensional models.

4.6 Linked Data Cube Enrichment

The *Enrichment* module is responsible for identifying the alignments between concepts and instances by using Ontology Matching techniques [Euzenat & Shvaiko 2007] and Record Linkage [Dunn 1946], respectively. Concept alignment is performed between *Linked Data Cube* descriptions, while instances alignment is performed between instances of the data cube dimen-

sions and data sources external to the framework. It is by means of these alignments that the framework is able to use data from external sources to build mashups and federated queries. This module is also in charge of the integration of the contents of the *External Catalogue* with the LOD cloud.

This module is composed of the submodules: *Data Source Recommendation*, *Ontology Matching* and *Record Linkage*.

The *Data Source Recommendation* sub-module offers a recommendation service to help select related and relevant triplesets to create links. The implementation of the sub-module is based on the TRT [Caraballo et al. 2013] and TRTML [Caraballo et al. 2014] Web-based tools. Both tools define the task of recommending datasets as a task of ranking the existing tripleset according to the probability that one can define links between resources of the triplesets. Consequently the submodule ranks the top ten tripleset from the popular LOD tripleset and the External Catalogue agains the dimension instances of the Linked Data Cubes; then the submodule sends this top ten rank as a recommendation list to the Record Linkage sub-module.

The *Ontology Matching* submodule is in charge of detecting the alignments between concepts of the descriptions of Linked Data cubes stored in the *External Catalogue*. The submodule performs SPARQL queries [Prud'hommeaux & Seaborne 2008] at the endpoint of the *External Catalogue* to obtain the Linked Data cube descriptions. Ontology matching techniques are applied to these data and the triples based on `owl:equivalentClass` and `owl:equivalentProperty` of the aligned concepts are generated.

In the *Record Linkage* submodule, applies record linkage techniques between the *External Catalogue* and the list of recommended external sources created by the *Data Source Recommendation* submodule. This submodule can use existing general purpose link discovery tools (such as LIMES [Ngonga Ngomo & Auer 2011] or SILK [Volz et al. 2009]) for linking of statistical data.

The result is expressed as a set of triples `owl:sameAs` which link instances of Linked Data cube dimensions stored in the *External Catalogue* with instances located in data sources external to the framework and scattered throughout the Web (e.g., DBpedia, Geonames). We illustrate record linkage with the help of an example of a SILK configuration for discovering links between dados.gov.br and DBpedia in Listing 4.5.

Recall from Section 2.3 that the components of a data cube are the dimensions `qb:DimensionProperty`, measures `qb:MeasureProperty` and attributes `qb:AttributeProperty`. In addition, we also consider as a component of a Linked Data Cube description, the instances of the domain of a dimen-

sion. According to Cabrera [Cabrera 2013] any of these components may be enriched or not depending on their type, defined as follows:

- Type 1: The component is identified by a URI and it has enough properties to make it possible to match the component to external resources.
- Type 2: The component is identified by a URI but it does not have enough properties that make it is possible to match the component to external resources.
- Type 3: The component is not identified by a URI.

As examples, one may consider: (1) a Type 1 component would be a URI denoting a country, with a label indicating the name of the country in English; (2) a Type 2 component would be a URI denoting a dimension, with a label indicating its name, taken from the name of the corresponding table in the original relational database (which does not have any meaning outside the database); and (3) a Type 3 component would be a string representing a date pertaining to the domain of the Age dimension.

The Enriching module stores the resulting `owl:sameAs` triples in the External Catalogue, along with the Linked Data Cube description.

4.7 Data Cube Discovery (DCD tool)

Recall from Section 2.2 that a data cube is stored in a relational database using tables typically organized in the shape of a star or a snowflake. *Star schemes* are composed of one or more fact tables that reference dimension tables. *Snowflake schemes*, on the other hand, are a more complex variation, where dimension tables are normalized into multiple, related tables.

Dimensional modeling must follow certain rules [Ross 2009] and avoid certain pitfalls [Kimball 2001]. Based on these rules and practices, the Data Cube Discovery module identifies data cubes stored in a relational database, using the following heuristics:

1. A table F is considered a fact table if:
   (a) All columns of F have numeric data types.
   (b) The primary key of F is composite, i.e., a list of columns.
   (c) Each column of its primary key is a foreign key to some other table.

(d) F has at least a non-key column, that is, a column that is not part of a key.

(e) Columns not in the primary key of F capture the measures of the observations represented in F.

2. *A* table D is considered a dimension table if:

(a) The primary key of D is a foreign key of a fact table.

(b) If D has a foreign key to some other table H, then the data cube is represented as a snowflake schema. Table H has a hierarchical relationship with D.

The user is free to switch on and off some of these rules, depending on how adherent the relational schema is to the typical practices of dimensional modeling. We remark that the heuristics may return false positives, that is, tables considered as fact tables, but which represent instead n-ary relationships. A detailed discussion of this issue is outside the scope of this work. For more details, please refer to [Ortiga 2013].

After identify the data cubes, the *Data Cube Discovery* module extracts the data cube relational metadata (e.g. table name, column name, column datatype) from the underlying relational database. Then, the data administrator updates, complements and associates semantic concepts of domain ontologies (e.g., FOAF, Dublin Core, DBpedia Ontology) to the relational metadata elements with the assistance of the *DCD tool* GUI. This is a metadata enrichment process prior to the Linked Data Cube Enrichment of the *Enrichment* module.

This module interacts with the *Internal Catalogue*, to save the enriched dimensional models; and depends heavily on the *Wrapper* module to be able to carry out its functions.

```
<Interlink id="States_DBpedia">

  <SourceDataset dataSource="http://lod2.inf.puc-rio.br/sparql" var="a">
    <RestrictTo>
            ?a <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> ?class.
            ?class <http://www.w3.org/2000/01/rdf-schema#subClassOf> <http://
                purl.org/ontology/places#State> .
    </RestrictTo>
  </SourceDataset>

  <TargetDataset dataSource="http://dbpedia.org/sparql" var="b">
    <RestrictTo>
            ?b <http://dbpedia.org/ontology/type> <http://dbpedia.org/resource
                /States_of_Brazil> .
    </RestrictTo>
  </TargetDataset>

  <LinkageRule linkType="&lt;http://www.w3.org/2002/07/owl#sameAs&gt;">
    <Compare id="label_distance" required="false" weight="1" metric="
        levenshteinDistance" threshold="0.0" indexing="true">
      <Param name="minChar" value="0"/>
      <Param name="maxChar" value="z"/>
      <TransformInput id="replace" function="replace">

        <TransformInput id="lower_case" function="lowerCase">
          <Input id="input_b" path="?b/<http://www.w3.org/2000/01/rdf-schema#
              label>"/>
        </TransformInput>
        <Param name="search" value=" (brasil)"/>
        <Param name="replace" value=""/>
      </TransformInput>

      <TransformInput id="replace" function="replace">
        <TransformInput id="lower_case" function="lowerCase">
          <Input id="input_a" path="?a/<http://www.w3.org/2000/01/rdf-schema#
              label>"/>
        </TransformInput>
        <Param name="search" value=" (brasil)"/>
        <Param name="replace" value=""/>
      </TransformInput>
    </Compare>
  </LinkageRule>
</Interlink>
```

Listing 4.5: SILK spec. for discovering links between dados.gov.br and DBpedia.

5
Olap2DataCube On Demand Processes

The operation of the proposed framework is structured as two main processes: *Catalogue Construction* and *Data Cube Consumption*. The process of *Catalogue Construction* is a prerequisite for the proper functioning of the *Data Cube Consumption* process, because the latter relies heavily on the information stored in the *Catalogue*.

5.1
Catalogue Construction Process

5.1.1
Overview

Catalogue Construction is supported by the *Wrapper*, *Mediator*, *DCD Tool*, *Catalogue* and *Linked Data Cube Enrichment* modules. This process is in charge of the generation and insertion of data into the Internal and External Catalogues.

The process is subdivided into four stages:

1. *Catalogue Exploration:* responsible for identifying dimensional models in the source databases and inserting the identified models into the *Internal Catalogue*.

2. *Metadata Enrichment:* responsible for semantic annotation, using domain ontologies, of persistent dimensional models in the *Internal Catalogue*.

3. *Linked Data Cube Generation:* responsible for generating dimensional model descriptions in triples, generating R2RML mappings for triplification of the dimension instances, and insertion of the triples into the *External Catalogue*.

4. *Linked Data Cube Enrichment:* responsible for aligning the Linked Data Cubes descriptions with standard ontologies and dimension domain

values with external sources; also responsible for inserting the alignments into the *External Catalogue*.

The stages of the *Catalogue Construction* process are elaborated in detail in the following sections.

5.1.2 Catalogue Exploration

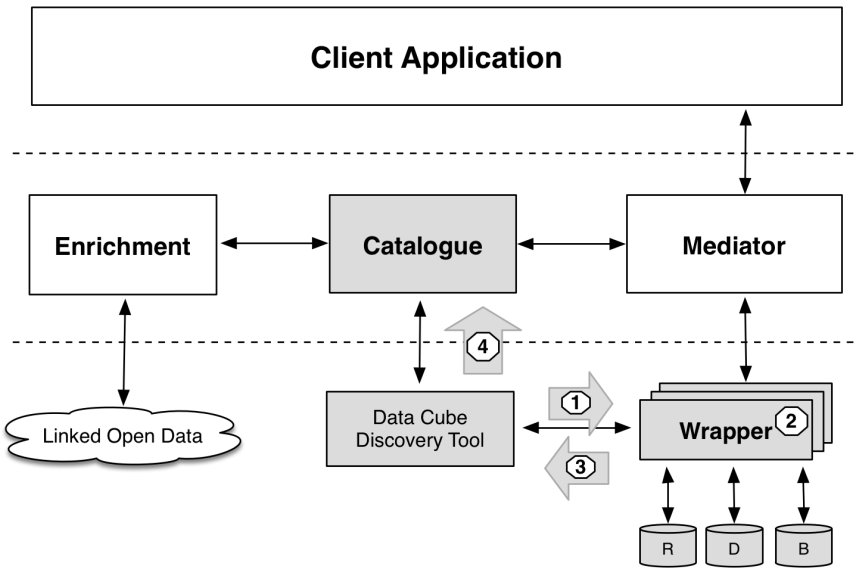


Figure 5.10: Workflow Catalogue Exploration

In the *Catalogue Exploration* stage, the dimensional models are identified and stored in the *Internal Catalogue*. The workflow (see Figure 5.10) of this stage is as follows:

- Step 1: The *DCD Tool* module sends a request to the *Wrapper* to explore the registered databases for dimensional models.
- Step 2: The *Wrapper* executes the exploration heuristics in the database.
- Step 3: The *Wrapper* returns the metadata of the identified dimensional models to the *DCD tool* module.
- Step 4: The *DCD tool* inserts the metadata of the identified dimensional models into the *Internal Catalogue*.

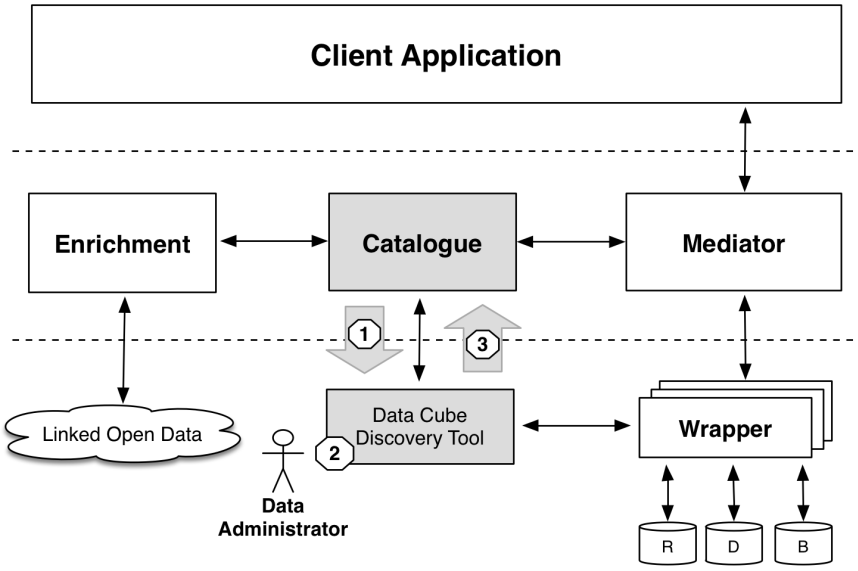


Figure 5.11: Workflow Metadata Enrichment

5.1.3 Metadata Enrichment

The *Metadata Enrichment* stage consists of the enrichment of the dimensional models by the data administrator with the assistance of the *DCD tool* GUI. The workflow (see Figure 5.11) of this stage is as follows:

- Step 1: The GUI of the *DCD tool* module retrieves information from the *Catalogue* of the dimensional model selected by the data administrator.
- Step 2: In the *DCD tool*, the data administrator updates, complements and associates semantic concepts of domain ontologies (e.g., FOAF, DublinCore, GoodRelations) to *Internal Catalogue* dimensional models.
- Step 3: The *DCD tool* inserts new information about the dimensional models into the *Internal Catalogue*.

5.1.4 Linked Data Cube Generation

At this stage, the *Triplification* submodule of the *Catalogue* generates triples that describe a Linked Data cube and stores them in the *External Catalogue*. The workflow (see Figure 5.12) of this stage is as follows:

- Step 1: The GUI of the *DCD tool* module retrieves information from the multidimensional models stored in the *Internal Catalogue*.

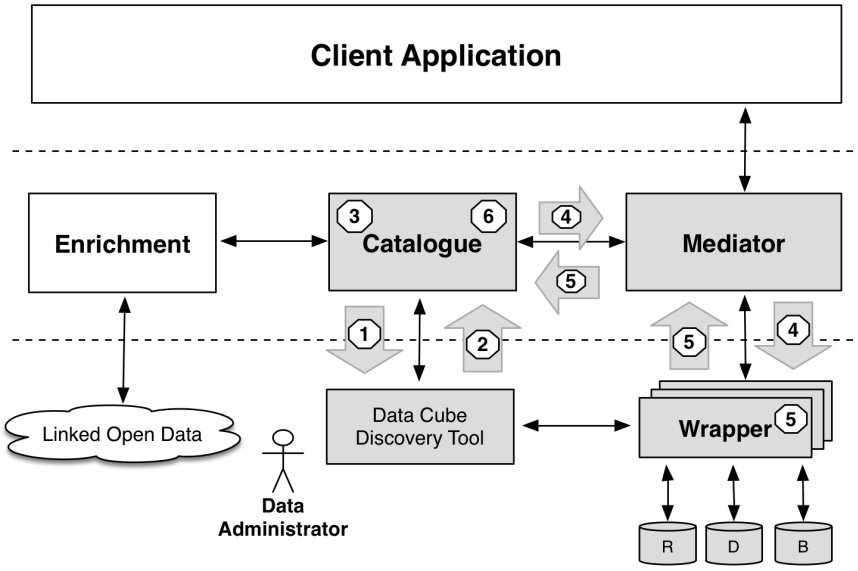


Figure 5.12: Workflow Linked Data Cube Generation

- Step 2: In the *DCD tool*, the data administrator of the authorized institution chooses one of many different multidimensional models and requests the generation of Linked Data Cube triples for the *Catalogue*.
- Step 3: The *Triplification* submodule performs the multidimensional model and data cube triplification using the DCV, QB4OLAP, SKOS and RDFS vocabularies.
- Step 4: The *Triplication* submodule also generates the R2RML mapping triples for the triplification of the dimension instances, and sends this R2RML mapping triples via the *Mediator* to be run in the *Wrapper*.
- Step 5: The *Wrapper* execute the R2RML mapping using the DB2Triple tool and returns the resulting triples to the *Catalogue* through the *Mediator*.
- Step 6: The *Catalogue* stores the triples resulting from Steps 3 and 5 in the *External Catalogue*.

5.1.5 Linked Data Cube Enrichment

The Linked Data Cube Enrichment is responsible for the alignment of the Linked Data Cubes with LOD vocabularies and triplesets. The workflow (see Figure 5.13) of this stage is as follows:

- Step 1: The *Enrichment* module identifies and recovers the classes, properties and instances using SPARQL queries on the *External Catalogue*.

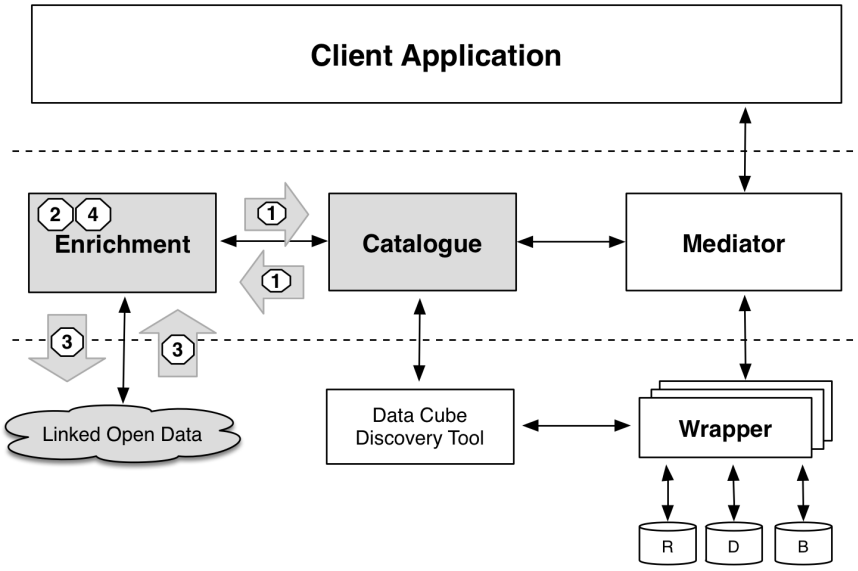


Figure 5.13: Workflow Linked Data Cube Enrichment

- Step 2: The *DataSources Recommendation* submodule generates a list of tripleset recommendations to perform the alignment operations.
- Step 3: The *Enrichment* has two types of alignments. The Ontology Matching which is responsible for aligning the Linked Data Cube (TBox) descriptions with standard vocabularies published on the Web. And the Record Linkage which is responsible for aligning the instances of the Linked Data Cube (ABox) with the list of recommended triplesets from the previous step. Thus in this step the *Ontology Matching* and *Record Linkage* submodules are configured to run the alignment techniques and to detect similar concepts and instances.
- Step 4: The *Enrichment* module inserts the alignments into the *External Catalogue*, i.e., the triples written with the `owl:sameAs`, `owl:equivalentClass` and `owl:equivalentProperty` OWL properties resulting from the alignment algorithms.

5.2 Data Cube Consumption Process

5.2.1 Overview

Data Cube Consumption is supported by the *Client Application*, *Mediator*, *Catalogue* and *Wrapper* modules. This process is responsible for responding

to all requests made by client applications. It uses the data available in the *Catalogue* to inform which data cubes it contains, the structure of these data cubes. It also allows queries to be performed on the data stored in the data cubes. Optionally the data cubes can be combined through mashup heuristics or undergo conversions through OLAP operations.

The process of generating observations of the selected cubes is divided into three stages:

1. *Search and Choose:* stage responsible for the search of cubes available in the *Catalogue*, and for the selection of the desired cubes through the *Client application*.

2. *Production and Request:* stage responsible for the on-demand generation of R2RML mappings which create the selected data cubes, and for the *Wrapper* mapping execution requests.

3. *Transform and Responds:* stage responsible for triplification and consolidation of the observations of the selected data cubes, which returns the resulting triples to the *Client application*.

The stages of the Data Cube Consumption process are outlined in the following sections.

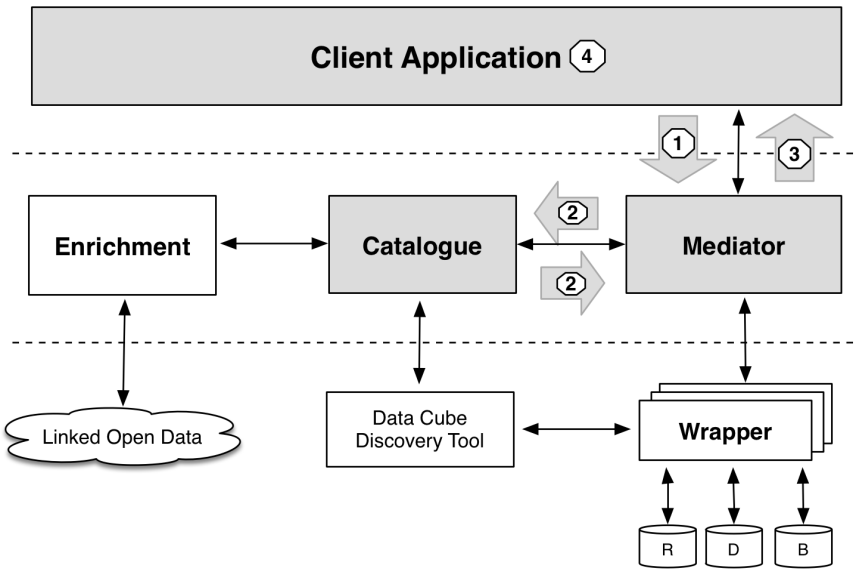


Figure 5.14: Workflow Search and Choose

5.2.2 Search and Choose

In the *Search and Choose* stage, the user searches for cubes about a particular matter by providing keywords; the user then selects the desired cube, having the ability to consider cube changes or combinations if necessary. The workflow (see Figure 5.14) of this stage is as follows:

- Step 1: The *Client Application* module sends a data cube fetch request to the *Mediator*.
- Step 2: The Mediator offers two search interfaces. The first interface is a SPARQL endpoint to the External Catalogue, through which an application may submit SPARQL queries to locate Linked Data Cube descriptions. The second is a keyword search interface that an application may use to submit keywords that are matched against Linked Data Cube descriptions stored in the *External Catalogue*.
- Step 3: The *Mediator* returns to the *Client Application* the RDF triples that represent a set of possible Linked Data Cubes to be handled.
- Step 4: The user then selects one of these Linked Data Cubes. Optionally, the *Client Application* may also decide that it needs the Linked Data Cube after a certain transformation, such as a slice of the cube (fixing a single value for one of its dimensions), or roll-up a dimension (summarizing the data along a dimension).

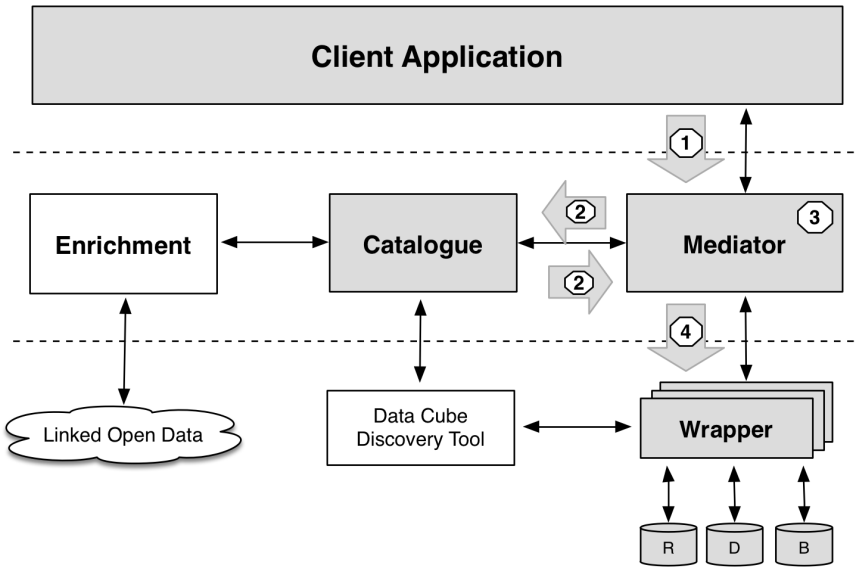


Figure 5.15: Workflow Production and Request

5.2.3 Production and Request

The *Production and Request* stage constructs the R2RML mapping from the information stored in the *Catalogue*, in order to triplify the observations of selected cubes, and then sends the mappings to be run by the appropriate *Wrappers*. The workflow (see Figure 5.15) of this stage is as follows:

- Step 1: The *Client Application* sends a fetch request to the *Mediator*. A fetch request specifies a cube and optionally a cube transformation.
- Step 2: The *Mediator* queries the *Catalogue* to retrieve information (hostname, database name, among others) about the cube and the underlying database where the cube is stored and how it is stored.
- Step 3: The *Mediator* then creates a R2RML mapping triples with an SQL statement embedded. Optionally the *Mediator* modifies the SQL statement to account for the transformations.
- Step 4: The *Mediator* then sends the R2RML mapping triples to the *Wrapper* for the underlying database to fetch the data cube.

5.2.4 Transform and Respond

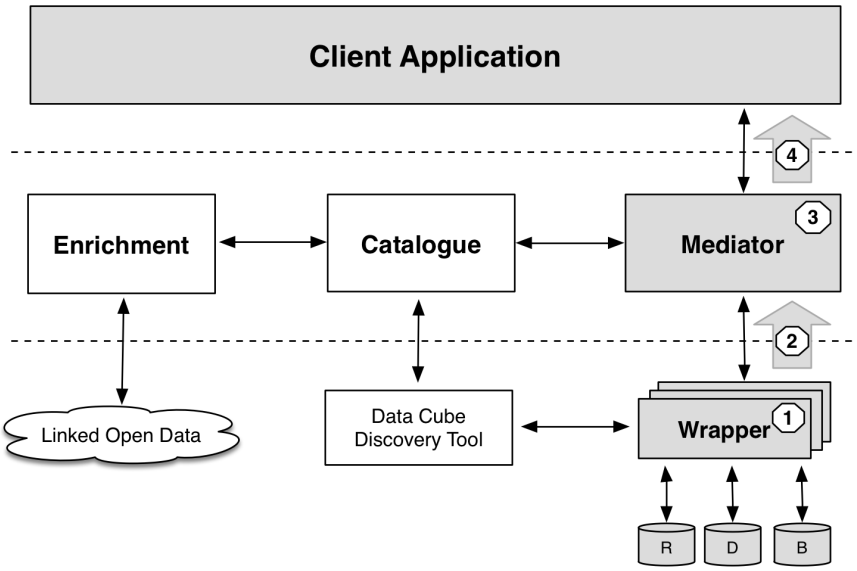


Figure 5.16: Workflow Transform and Respond

In the *Transform and Respond* stage, triples of the selected Linked Data Cube observations are generated, after which these triples are consolidated (changing

or combining the cubes as desired), and ultimately forwarded to the user in response to the request made. The workflow (see Figure 5.16) of this stage is as follows:

- Step 1: The *Wrapper* receives the R2RML mapping triples and uses them in the DB2Triples tool to convert the data stored in the source databases into the requested Linked Data Cube triples.
- Step 2: The *Wrapper* sends the triples generated by the DB2Triples to the *Mediator*.
- Step 3: The *Mediator* consolidates the triples sent by the various *Wrappers*. The *Mediator* may optionally modifies the set of resulting triples to fulfill the request for cube conversion or mashup.
- Step 4: Finally, the *Mediator* sends the resulting triples set described by the DCV and QB4OLAP vocabularies to the *Client Application*.

Using the set of incoming triples, the *Client application* displays to the user the returned data, enabling the user to manipulate these data within the realm of possibilities existing in the *Client application*.

6
Olap Operations and Data Cube Mashup

6.1
Introduction

The general mechanism to compile the OLAP operation over RDF data cube encompasses three stages, as illustrated in Figure 6.17.

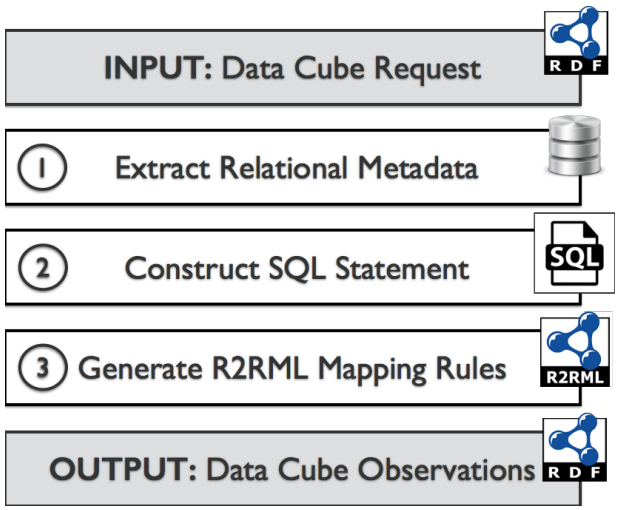


Figure 6.17: OLAP operation general process

In the first stage, we parse the RDF data cube request to obtain the IRIs of the components (dimensions, dimension levels, measures) and extract their relational metadata from the *Internal Catalogue*. Then, we select the proper OLAP operation required to fulfill the request. Listing 6.1 illustrates a RDF data cube request for a roll-up operation over city level corresponding to our running example (see Figure 4.9).

```
@prefix qb: <http://purl.org/linked-data/cube#> .
@prefix sdmx: <http://purl.org/linked-data/sdmx/2009/attribute#> .
@prefix qb4o: <http://purl.org/qb4olap/cubes#> .
@prefix eg: <http://lod2.inf.puc-rio.br/olap2datacube/> .

eg:dsd-87b1fc9aa278fd9dc861355a54357819
  a qb:DataStructureDefinition ;
  qb:component
  [
    qb:measure eg:428bd4f770abb36319debc6b9545d637;
    qb4o:hasAggregateFunction qb4o:sum ],
  [
    qb:attribute sdmx:unitMeasure; qb:componentAttachment qb:MeasureProperty
  ],
  [ qb4o:level eg:dim1446dim_time199 ],
  [ qb4o:level eg:out1443dim_state199 ].
```

Listing 6.1: Data cube request with roll-up operation over city level.

In the second stage, we use the relational metadata, extracted in the previous step, as building blocks to construct a SQL statement that implements the appropriate OLAP operation to query the underlying relational databases. Listing 6.2 illustrates the SQL query corresponding to the previous RDF data cube request.


```
SELECT distinct
 md5(concat(dim_time_199.year,"214dim_time_199")) as dim_time_199_md5,
 md5(concat(dim_state_199.code,"214dim_state_199")) as dim_state_199_md5,
 md5(concat(dim_time_199.year,dim_state_199.code)) as md5,
 SUM(fact_population.var_city) AS agg_var_city
FROM
 fact_population,
 dim_time_199,
 dim_state_199,
 dim_city_199
WHERE
 dim_time_199.year = fact_population.year AND
 dim_state_199.code=dim_city_199.code_state AND
 dim_city_199.code=fact_population.code
GROUP BY
 md5(concat(dim_time_199.year,"214dim_time_199")),
 md5(concat(dim_state_199.code,"214dim_state_199")),
 md5(concat(dim_time_199.year,dim_state_199.code))
```

Listing 6.2: SQL query for roll-up operation over city level.

In the third stage, we generate customized R2RML mapping rules for the SQL statement, obtained in the previous step, to convert the resulting tuples returned by the SQL query to RDF triples, expressed using the DCV and QB4OLAP vocabularies. Listing 6.3 illustrates the customized R2ML mapping rules corresponding to the previous SQL statement.

```
@prefix sdmx: <http://purl.org/linked-data/sdmx/2009/attribute#> .
@prefix rr: <http://www.w3.org/ns/r2rml#> .
@prefix qb: <http://purl.org/linked-data/cube#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix eg: <http://lod2.inf.puc-rio.br/olap2datacube/> .
@prefix eg-res: <http://lod2.inf.puc-rio.br/olap2datacube/resources/> .

eg:TriplesMapfa6517a3ab6cb030c481fedaeae8f7c6 a rr:TriplesMap ;
  rr:logicalTable [ rr:sqlQuery """ --- SQL QUERY --- """ ] ;
  rr:subjectMap [
    rr:template "eg-res:{md5}" ;
    rr:class qb:Observation
  ] ;
  rr:predicateObjectMap [
    rr:predicate eg:dim1446dim_time199 ;
    rr:objectMap [ rr:template "eg-res:dim_time_199/{dim_time_199_md5}" ]
  ], [
    rr:predicate eg:out1443dim_state199 ;
    rr:objectMap [ rr:template "eg-res:dim_state_199/{dim_state_199_md5}" ]
  ], [
    rr:predicate eg:428bd4f770abb36319debc6b9545d637 ;
    rr:objectMap [ rr:column "agg_var_city" ]
  ], [
    rr:predicate qb:dataSet ;
    rr:objectMap [ rr:constant eg:87b1fc9aa278fd9dc861355a54357819 ]
  ], [
    rr:predicate sdmx:unitMeasure ;
    rr:objectMap [ rr:constant foaf:Person ]] .
```

Listing 6.3: R2RML mapping rules for roll-up operation over city level.

6.2 Definition of the OLAP Operations

The multidimensional manipulation of data cubes requires the definition of a conceptual model over which the OLAP operations will be specified. Therefore, before describing the OLAP operation we define the elements of a data cube.

Recall from Section 2.3 that a data cube is organized according to a set of *dimensions*, *measures* and *attributes*. The set of *dimensions* is denoted $D = < D_1, \ldots, D_m >$ and defines what the observations apply to.

According to Vassiliadis in [Vassiliadis 1998], a *dimension* is a lattice $(H, \prec)$ of *levels*. Each path in the lattice, beginning from its least upper bound and ending in its greatest lower bound, is called a *dimension path*. Each dimension path is linear, total order of levels. The total order allows us to use comparison operators for the dimension levels. For instance, if we consider the dimension path [year, month, day], then $day \prec month \prec year$ holds.

For each dimension level, there is a set of *values*. For example, the dimension level "City" has "Rio de Janeiro", "Paris" and "Rome" as members. The set of all values of a dimension level dl_i is denoted as $DOM(dl_i)$.

A value x can have *ancestors* and *descendants*. Let x belong to a specific dimension level dl_x. Then, there are specific instances related to x, at higher

(lower) dimension level in the lattice, corresponding to more general (detailed) terms. We denote $ancestor(x, dl_y) = y$, with $y \in DOM(dl_y)$ and $dl_x \prec dl_y$ $descendants(x, dl_y) = \{y_1, y_2, \ldots, y_k\}$, with $y_2, \ldots, y_k \in DOM(dl_y)$ and $dl_y \prec dl_x$.

The correspondence between a value and its *ancestors* is defined through so-called *rollup* functions. For instance, given a pair of levels (dl_j, dl_k) of dimension D_i such that $dl_j \prec dl_k$, a relation (denoted *rollup*) is defined, associating values from level dl_j with values of level dl_k.

Based on the conceptual model presented above, we define an SQL query mechanism for performing OLAP operations on the relational databases. This mechanism is based on the OLAP operations briefly summarized in the rest of this section (see [Vassiliadis 1998, Etcheverry & Vaisman 2012] for detailed definitions).

- `Roll-Up` summarizes data at a higher level in the lattice of a dimension. Given a cube C, a dimension $D_i \in C$, with levels dl_o and dl_u such that $dl_o \prec dl_u$, and an aggregate functions f, `Roll-Up(`C`,` D_i`,` dl_u`,` f`)` returns a new cube C' whose measure values are aggregated along D_i up to level dl_u, using the aggregate function f. Aggregation is performed according with the *rollup* function associated with the relation $dl \prec dl_u$.
- `Slice` removes one dimension from a cube C. Intuitively, given a cube C, a dimension $D \in C$, and an aggregation function f, the result of applying the `Slice` operation returns a new cube $C' =$ `Slice(`C`,`D`,`f`)` whose dimensions are those of C, except for D. Measures in C' are the result of applying the aggregation function f along dimension D, up to the top level of D, before removing the dimension.
- `Dice` selects a subset of the instances of a cube. Intuitively, given a cube C, a dimension $D \in C$, and a first order formula σ over the levels in D, the operation `Dice` returns a new cube $C' =$ `Dice(`C`,` D`,` σ`)` whose dimensions are the same as in C, and such that the elements in C are those that satisfy σ.
- `Drill-Across` links two or more cubes at the same granularity level. Intuitively, given two cubes C_1 and C_2 such that
  - C_1 has p dimensions with s dimension levels such that k of which are conformed;
  - C_2 has q dimensions with t dimension levels such that k of which are conformed.

Let C'_1 and C'_2 respectively be C_1 and C_2 summarized over their conformed dimension levels. The new data cube C_X is the combined data of C'_1 and C'_2 summarized over the common values of their conformed dimension levels, using the natural join.

6.3 OLAP Operations Implementation

6.3.1 Pre-Processing: Relational Metadata Extraction

Before the execution of the OLAP operations, we need to parse the data cube definition in order to identify the metadata elements of the relational databases involved and select the proper OLAP operation required to fulfill the request.

In order to identify the metadata elements of the data cube description, we use Algorithm 1 to query the Internal Catalogue and extract the relational metadata of the requested data cube components. Later these relational metadata will be used as building blocks to construct the SQL statements of the OLAP operations.

We now describe Algorithm 1 which receives the data cube description as input, where *dsd* is the data structure definition of a data cube c in DCV and QB4OLAP, DL is the set of dimension level components in *dsd*, M is the set of pairs (m_i, ag_i) where m_i is a measure component and ag_i is its corresponding aggregate function. We use the following functions:

- `add(`$A, Index, B$`)` appends B into a particular position $Index$ of the data structure A.
- `getMetadata(`iri`)` returns all the relational metadata (e.g. table name, column name, column restrictions) related to the iri element.
- `aggFunction(`iri`)` returns the aggregate function (e.g. SUM, AVG, MEDIAN) associated to the iri element.
- $levels(dl)$ returns a set of levels l_j, l_k up to level dl in the hierarchy, such that each $l_j \preceq l_k$ (meaning that l_j is lower than l_k in the hierarchy)

The algorithm traverses the data cube description (dsd, DL, M) to extracts the relational metadata and adds them to the $(Metadata_M, Metadata_{DL})$ data structures. Lines 3 through 5 extract the metadata of the measure component m_i and its associated aggregate function ag_i. Then it adds the metadata into the data structure $Metadata_{m_i}$. In lines

6 through 23 it extracts the metadata of the dimension levels dl_i related to each measure component m_i, and adds them into the structure $Metadata_{dl_i}$. Line 9 obtains a set of pair of levels l_j, l_k up to dl_i in the hierarchy (all the pair of levels in the hierarchy of dl_i that satisfied l_j `qb4o:parentLevel` l_k). Lines 10 through 13 extract the metadata of the pair of levels l_j, l_k and add them into the structure $Metadata_{dl_i}$. Lines 14 through 16 obtain the dl_o which represents the bottom dimension level in the hierarchy level of dl_i. Line 18 add the bottom level dl_o of each dimension level dl_i into the structure $Metadata_{dl_i}$. Lines 19 through 22 obtain the dimension level instances $dom(dl_i)$ and add them into the structure $Metadata_{dl_i}$.

6.3.2 A General Algorithm for Roll-up/Drill-down

Once we extracted the relational metadata of the data cube description we can use it as building blocks to construct traditional SQL statements to query the underlying relational databases. In this section, we describe Algorithm 3 that compiles Roll-Up/Drill-Down operations over an RDF data cube into SQL code. We define the following functions:

- `get(`$A, Index$`)` returns the data stored in a particular position $Index$ from the data structure A.
- `value(`t, c`)` returns the value stored in the column c of the table t.
- `SQLMeasurePart(`$Metadata_{m_i}, qSQL$`)` adds the column and table (fact table) of the measure metadata $Metadata_{m_i}$ into the SELECT and FROM clauses of the SQL query structure $qSQL$. In this function, f represents the SQL function corresponding to the aggregate function for each measure and $f(value(Table_{m_i}, Column_{m_i}))$ is the string that should be included to calculate the aggregated value. By default SUM(value(table, column)) is the aggregate function.
- `SQLDimensionPart(`$Metadata_{dl_i}, qSQL$`)` adds the columns and table (dimension table) of the dimension level metadata $Metadata_{dl_i}$ into the SELECT and FROM clauses of the SQL query structure $qSQL$.
- SQLWhereBottomLevel($Metadata_{m_i}, Metadata_{dl_i}, qSQL$) adds the cross-reference $FK_{m_i \to dl_o}$ between the measure table $Table_{m_i}$ (fact table) and the bottom dimension level table $Table_{dl_o}$ of the dimension dl_i (dimension table) into the WHERE clause of the SQL query structure $qSQL$.

Algorithm 1: Relational metadata extraction from data cube definition.

input : dsd is the data structure definition of a data cube c described in QB and QB4OLAP vocabularies, DL is the set of dimension level components in dsd, M is the set of pairs (m_i, ag_i) where m_i is a measure component and ag_i is its corresponding aggregate function.

output: $Metadata_{DL}$ is the metadata of the dimension level, and the $Metadata_M$ is the metadata of the measure components.

```
1  Function ExtractRelationalMetadata(dsd, DL, M)
2      foreach (m_i, ag_i) ∈ M
       such that (dsd qb:component [qb:measure m_i;qb4o:hasAggregateFunction ag_i])
       do
3          Metadata_{m_i} = getMetadata(m_i)
4          f = aggFunction(ag_i)
5          add(Metadata_{m_i}, "aggFunction", f)
6          foreach dl_i ∈ DL
           such that (dsd qb:component [qb4o:level dl_i]) do
7              Metadata_{dl_i} = getMetadata(dl_i)
8              dl_o = dl_i
9              L_j, L_k = levels(dl_i)
10             foreach (l_j, l_k) ∈ L_j, L_k do
11                 Metadata_{l_j} = getMetadata(l_j)
12                 Metadata_{l_k} = getMetadata(l_k)
13                 add(Metadata_{dl_i}, "LevelsPath", (Metadata_{l_j}, Metadata_{l_k}))
14                 if dl_o ⪯ l_j then
15                     dl_o = l_j
16                 end
17             end
18             add(Metadata_{m_i}, ("BottomLevel", dl_i), dl_o)
19             Obtain all the dimension level instances DOM(dl_i) of the
               dimension level dl_i.
20             foreach dom(dl_i) ∈ DOM(dl_i) do
21                 add(Metadata_{dl_i}, "instances", dom(dl_i))
22             end
23         end
24     end
25     return (Metadata_M, Metadata_DL)
```

The algorithm incrementally builds the SQL query, using the add function. Line 3 adds the SELECT and FROM clauses to the $qSQL$ structure for each measure component m_i. Lines 4 through 16 add the SQL clauses (SELECT, FROM, and WHERE) of the dimension levels dl_i related to each measure component m_i into the $qSQL$ structure. Line 6 obtains the hierarchy dimension levels path metadata HL_{path} of the dimension level dl_i. Line 7 through 14 add the tables and columns needed to navigate the dimension

level hierarchy. Lines 10 and 11 add tables $Table_{dl_j}$ and $Table_{dl_k}$ of the dimension levels dl_j and dl_k to the FROM clause, and line 13 adds the foreign key columns $FK_{dl_j \to dl_k}$ of the table $Table_{dl_j}$ (child table) and the table $Table_{dl_k}$ (parent table) into the WHERE clause of the SQL structure $qSQL$. Finally, line 17 adds the SELECT clause of the dimension levels into the GROUPBY clause of the SQL query structure $qSQL$.

Component	**IRI Generation Rule**
Dimension property	**"dimension"** $+ID_{Dimension} + Name_{Dimension}$
Level property	**"level"** $+ID_{Level} + Name_{Level})$
Measure property	**md5**(**"measure"** $+ID_{Measure} + Name_{Measure})$

Component Instance	**IRI Generation Rule**
Dataset ($\boldsymbol{IRI_{Dataset}}$)	**md5**$(ID_{Model} + Name_{Model})$
TriplesMap ($\boldsymbol{tMap}$)	**md5**$(PK_{All-Dimension} + Column_{All-Measures})$
Dimension ($\boldsymbol{IRI_{Dim}}$)	**md5**(**value**$(PK_{Dimension}) + ID_{Model} + Name_{Dimension})$
Observation ($\boldsymbol{IRI_{Obs}}$)	**md5**(**value**$(PK_{All-Dimension}) + ID_{Model})$

Table 6.1: IRI Generation Rules.

Algorithm 2: Functions used in the operators construction.

1 **Function** `SQLMeasurePart`$(Metadata_{m_i}, qSQL)$
2 $\quad Table_{m_i} =$ `get`$(Metadata_{m_i}, \text{“}Table\text{”})$
3 $\quad Column_{m_i} =$ `get`$(Metadata_{m_i}, \text{“}Column\text{”})$
4 $\quad f =$ `get`$(Metadata_{m_i}, \text{“}aggFunction\text{”})$
5 $\quad$ `add`$(qSQL, \text{“}Select(measure)\text{”}, f(value(Table_{m_i}, Column_{m_i})))$
6 $\quad$ `add`$(qSQL, \text{“}From\text{”}, Table_{m_i})$
7 $\quad$ **return** $qSQL$
8 **Function** `SQLDimensionPart`$(Metadata_{dl_i}, qSQL)$
9 $\quad Table_{dl_i} =$ `get`$(Metadata_{dl_i}, \text{“}Table\text{”})$
10 $\quad Column_{dl_i} =$ `get`$(Metadata_{dl_i}, \text{“}Column\text{”})$
11 $\quad$ `add`$(qSQL, \text{“}Select\text{”}, (Table_{dl_i}.Column_{dl_i}))$
12 $\quad$ `add`$(qSQL, \text{“}From\text{”}, Table_{dl_i})$
13 $\quad$ **return** $qSQL$
14 **Function** `SQLWhereBottomLevel`$(Metadata_{m_i}, Metadata_{dl_i}, qSQL)$
15 $\quad IRI_{dl_i} =$ `get`$(Metadata_{dl_i}, \text{“}IRI\text{”})$
16 $\quad IRI_{dl_o} =$ `get`$(Metadata_{m_i}, (\text{“}BottomLevel\text{”}, IRI_{dl_i}))$
17 $\quad FK_{m_i \to dl_o} =$ `get`$(Metadata_{m_i}, (\text{“}ForeignKey\text{”}, IRI_{dl_o}))$
18 $\quad$ `add`$(qSQL, \text{“}Where\text{”}, FK_{m_i \to dl_o})$
19 $\quad$ **return** $qSQL$

Algorithm 3: Generates the SQL query that builds the *Roll-Up/Drill-Down* operators.

input : $Metadata_{DL}$ is the metadata of the dimension level, and the $Metadata_M$ is the metadata of the measure components.
output: $qSQL$ is the SQL query that builds the *Roll-Up/Drill-Down* operators.

```
1  Function RollUpOrDrillDown(Metadata_DL, Metadata_M)
2      foreach Metadata_mi ∈ Metadata_M do
3          qSQL = SQLMeasurePart(Metadata_mi, qSQL)
4          foreach Metadata_dli ∈ Metadata_DL do
5              qSQL = SQLDimensionPart(Metadata_dli, qSQL)
6              HL_Path = get(Metadata_dli, "LevelsPath")
7              foreach (Metadata_dlj, Metadata_dlk) ∈ HL_Path do
8                  Table_dlj = get(Metadata_dlj, "Table")
9                  Table_dlk = get(Metadata_dlk, "Table")
10                 add(qSQL, "From", Table_dlj)
11                 add(qSQL, "From", Table_dlk)
12                 FK_dlj→dlk = get(Metadata_dlj, ("ForeignKey", IRI_dlk))
13                 add(qSQL, "Where", FK_dlj→dlk)
14             end
15             qSQL = SQLWhereBottomLevel(Metadata_mi, qSQL, Metadata_dli)
16         end
17         add(qSQL, "GroupBy", get(qSQL, "Select"))
18     end
19     return qSQL
```

6.3.3 A General Algorithm for Slice/Dice

In this section, we describe Algorithm 4 that compiles slice/dice operators over an RDF data cube into SQL code. The SQL construction process of these operations uses several functions already defined in the roll-up/drill-down algorithm. However we define the function `getDimRule(`$Metadata_{m_i}, Metadata_{dl_i}$`)` which uses the metadata $Metadata_{m_i}, Metadata_{dl_i}$ as input to construct the IRI template for the dimension level dl_i following the proper IRI generation rule in Table 6.1.

As mentioned before, the general algorithm that implements the slice/dice operators share several construction steps with the roll-up/drill-down algorithm. However between lines 6 and 10, the Algorithm 4 implements the main functionality of the slice/dice operators. Line 6 obtains the IRI template IRI_{Dim} for each dimension level dl_i. Line 7 obtains a set of instances Dom_{dl_i} of the $Metadata_{dl_i}$. Line 9 adds the pair $(iriRule_{Dim}, dom(dl_i))$ for each instance $dom(dl_i)$ to the WHERE clause of the SQL query structure $qSQL$.

Algorithm 4: Generates the SQL query that builds the *Slice/Dice* operators.

input : $Metadata_{DL}$ is the metadata of the dimension level, and the $Metadata_M$ is the metadata of the measure components.
output: $qSQL$ is the SQL query that builds the *Slice/Dice* operators.

```
1  Function SliceOrDice(Metadata_DL, Metadata_M)
2      foreach Metadata_mi ∈ Metadata_M do
3          qSQL = SQLMeasurePart(Metadata_mi, qSQL)
4          foreach Metadata_dli ∈ Metadata_DL do
5              qSQL = SQLDimensionPart(Metadata_dli, qSQL)
6              IRI_Dim = getDimRule(Metadata_mi, Metadata_dli)
7              DOM(dl_i) = get(Metadata_dli, "instances")
8              foreach dom(dl_i) ∈ DOM(dl_i) do
9                  add(qSQL, "Where", (IRI_Dim, dom(dl_i)))
10             end
11             qSQL = SQLWhereBottomLevel(Metadata_mi, qSQL, Metadata_dli)
12         end
13     end
14     return qSQL
```

6.3.4 Post-Processing: R2RML Mapping Rules Generation

After the construction of the SQL query that implements the OLAP operation, we generate the R2RML mapping rules to convert the SQL query results to RDF triples, expressed using DCV and QB4OLAP vocabularies.

We now show an algorithm for the generation of the R2RML mapping rules. Algorithm 5 takes as input the SQL structure $qSQL$, the metadata of the dimension levels $Metadata_{DL}$ and the measure component $Metadata_M$ and produces the R2RML mapping rules $tMap$ that generate the RDF triples of the data cube requested. We define the following functions:

- `newBnode(`v`)` generates a unique blank node[5] v.
- `value(`v`)` returns the value stored in variable v.
- `addTriple(`g, s, p, o`)` adds a triple $< s, p, o >$ to the RDF graph g
- `predObjMap(`$g, s, p, oType, o$`)` creates a predicate-object mapping rule for the predicate p and the object o of type $oType$. Then the mapping rule is associate with the subject s. Finally it adds the mapping rule to the graph g.

Similary to the construction of the SQL query detailed in Section 6.3.2 the algorithm R2RMLGeneration incrementally builds the R2RML mapping rules,

[5]Blank node identifiers are local identifiers that are used in some concrete RDF syntaxes

using the addTriple function. Line 10 obtains the IRI templates $IRI_{TriplesMap}$, $IRI_{Dataset}$ and IRI_{Obs} following the proper IRI generation rules in Table 6.1 using as input the $Metadata_{m_i}$ and the *Select* part of the $qSQL$ structure. Line 11 adds a triple to the $tMapGraph$ graph, that defines the set of triples map rules. Lines 12 through 14 add triples that defines the mapping data source, in our case the SQL query $qSQL$. Line 15 through 18 define the subject map rule with the class `qb:Observation` and the observation IRI template IRI_{Obs}. Lines 19 to 29 define a set of predicate-object mapping rules. Lines 19 and 21 add a predicate-object mapping rule for the `qb:dataset` and the `sdmx:unitMeasure` with the IRI template $IRI_{Dataset}$ and the IRI_A respectively. Lines 22 through 24 extract the $Select_{m_i}$ part of $qSQL$ and the measure component m_i, and use them to construct the predicate-object mapping rule for the measure with the m_i as predicate and the $Select_{m_i}$ as object. Lines 25 to 30 define the predicate-object mapping rule of the dimension levels with the dimension level dl_i as predicate and the IRI template IRI_{Dim} as the object, for each dimension levels $dl_i \in D$.

It is important to notice that the values of the IRI templates are replaced in runtime with the appropriate values of the SQL query result.

Algorithm 5: Generates the R2RML mapping rules to convert the SQL query result to RDF triples.

input : $Metadata_{DL}$ is the metadata of the dimension level, the $Metadata_M$ is the metadata of the measure components, and the $qSQL$ is the SQL query to define the 'R2RML view'.
output: $tMapGraph$ is the set of R2RML mapping rules.

1 **Function** predObjMap($tMapGraph, tMap, Pred, Obj_{Type}, Obj$)
2 newBnode(pom)
3 addTriple($tMapGraph, tMap$, rr:predicateObjectMap , $value(pom)$)
4 addTriple($tMapGraph, value(pom)$, rr:predicate , $Pred$)
5 newBnode(om)
6 addTriple($tMapGraph, value(pom)$, rr:objectMap , $value(om)$)
7 addTriple($tMapGraph, value(om), Obj_{type}, Obj$)
8 **Function** R2RMLGeneration($Metadata_{DL}, Metadata_M, qSQL$)
9 **foreach** $Metadata_{m_i} \in Metadata_M$ **do**
10 Get the IRI templates $tMap$, $IRI_{Dataset}$ and IRI_{Obs} following the proper IRI generation rules using the $Metadata_{m_i}$, and the *Select* clause of $qSQL$.
11 addTriple($tMapGraph, tMap$, rdf:type , rr:TriplesMap)
12 newBnode(lt)
13 addTriple($tMapGraph, tMap$, rr:logicalTable , $value(lt)$)
14 addTriple($value(lt)$, rr:sqlQuery , $qSQL$)
15 newBnode(sm)
16 addTriple($tMapGraph, tMap$, rr:subjectMap , $value(sm)$)
17 addTriple($tMapGraph, value(sm)$, rr:template , IRI_{Obs})
18 addTriple($tMapGraph, value(sm)$, rr:class , qb:Observation)
19 predObjMap($tMapGraph, tMap$, qb:dataset , rr:constant , $IRI_{Dataset}$)
20 IRI_A = get($Metadata_{m_i}$, "*Attribute*")
21 predObjMap($tMapGraph, tMap$, smdx:unitMeasure , rr:constant , IRI_A)
22 Get the $Select_{m_i}$ from the $qSQL$ structure.
23 IRI_{m_i} = get($Metadata_{m_i}$, "*IRI*")
24 predObjMap($tMapGraph, tMap, IRI_{m_i}$, rr:column , $Select_{m_i}$)
25 **foreach** $Metadata_{dl_i} \in Metadata_{DL}$ **do**
26 Get the $Select_{dli}$ from the $qSQL$ structure.
27 Get the IRI template IRI_{Dim} for dl_i using the $Metadata_{m_i}$ and $Select_{dli}$
28 IRI_{dli} = get($Metadata_{dl_i}$, "*IRI*")
29 predObjMap($tMapGraph, tMap, IRI_{dli}$, rr:template , IRI_{Dim})
30 **end**
31 **end**
32 **return** $tMapGraph$

6.4 Data Cube Mashup

Data analysts would greatly benefit from the ability to navigate and combine data cubes from multiple sources. However, this requires that the dimensions of the data cube be in some sense aligned, or that they have *conformed dimensions*. The strict requirements of conformity restrict the feasibility of mashup construction between external unfamiliar multidimensional databases.

Indeed, according to Kimball & Ross [Kimball & Ross 2002], a key requirement to combine or integrate multidimensional data from heterogeneous sources is that they must have conformed dimensions. Their view of conformed dimensions, which is widely adopted, requires the conformed dimensions to be identical in terms of schema and data.

In this section we present the *Data Cube Mashup* operator that extends the navigation operation *Drill-Across* to include apparently non-conformed dimensions. This new operator takes advantage of the dimensions "enriched" with external data (links between the *Linked Data Cube* descriptions stored in the *External Catalogue* and the LOD cloud) to discover possible relationships between seemingly non-conformed dimensions.

The process of combining two data cubes using the *Data Cube Mashup* operator encompasses three stages (see Figure 6.18). In the first stage, the submodule *Mashups On Demand* identifies the conformed dimensions (cDL) in the data cubes.

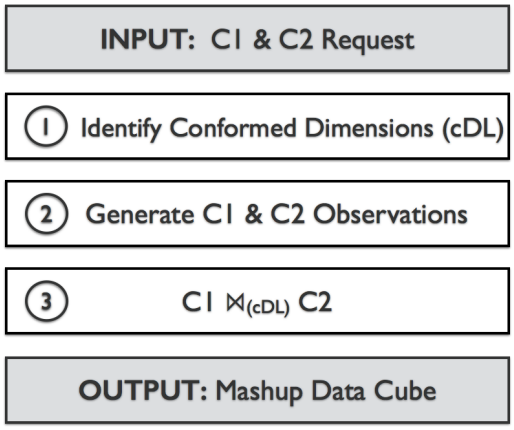


Figure 6.18: Mashup Data Cube general process

In order to identify the conformed dimensions the submodule analyzes the feasibility of integrating their dimensions identifying a number of desirable properties that aligned dimensions should satisfy. The workflow (see Figure 6.19) of this stage is as follows:

1. *Granularity compatibility:* the submodule searches for the property `qb4olap:parentLevel` in the *Linked Data Cube* description to determine the level of hierarchy of the *Linked Data Cube* dimensions. Based on this information the submodule analyzes the feasibility of executing OLAP operations (e.g., Roll-up, Drill-down) on the data cubes for performing the mashup.

2. *Schema compatibility:* the submodule searches for the property `owl:equivalentClass` between the dimensions of the Linked Data Cubes to identify the presence of semantic relationship between dimension concepts and determine the *Schema compatibility*.

3. *Data compatibility:* the submodule searches for the property `owl:sameAs` between the dimension instances, previously filtered by the schema compatibility, to calculate the similarity between the *Linked Data Cube* dimension instances of the potential mashup.

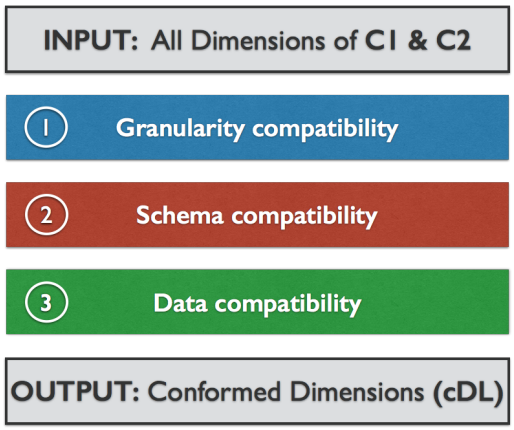


Figure 6.19: Conformed Dimension identification process

For instance, Figure 6.20 depicts the *Granularity and Schema compatibility* process between the dimensions `vocab-x:State` and `vocab-x:City` of data cube `C1` and the dimension `vocab-y:State` of data cube `C2`. In order to align the dimensions of data cube `C1` and `C2`, we need execute a Roll-up operation on the dimension `vocab-x:City`. Figure 6.21 depicts the *Data compatibility* between the dimension instances of data cubes `C1` and `C2`.

Once the conformed dimensions have been identified as shown in Figure 6.22, the submodule *Mashups On Demand* starts the second stage requesting to the *Federated Queries* submodule to determine the sources needed to construct each one of the data cubes. Then, for each source identified,

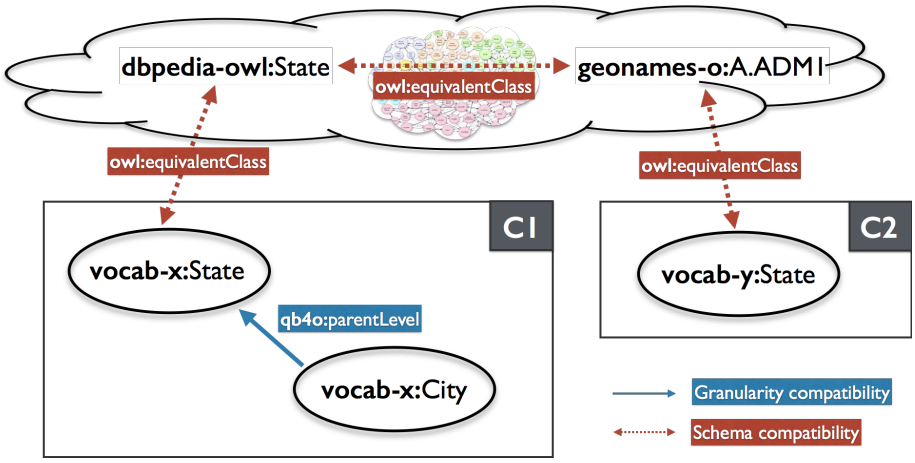


Figure 6.20: Granularity and Schema compatibility of Data Cube `C1` & `C2`

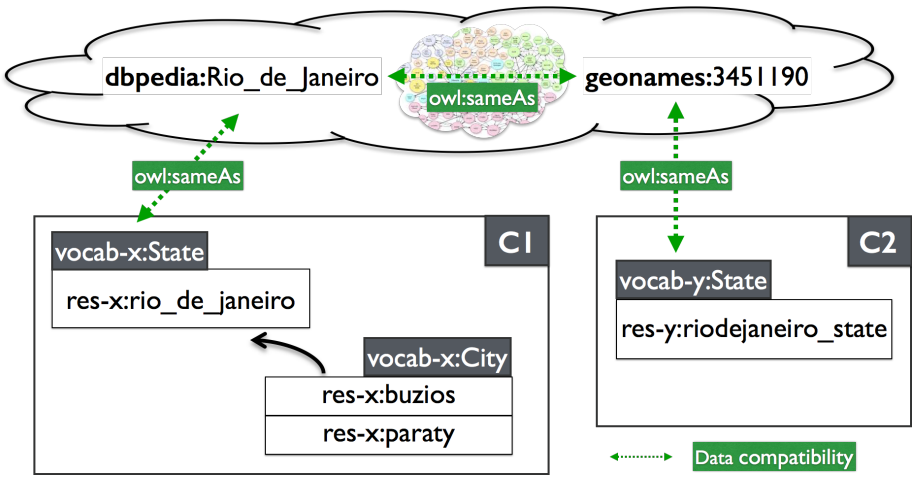


Figure 6.21: Data compatibility of Data Cube `C1` & `C2`

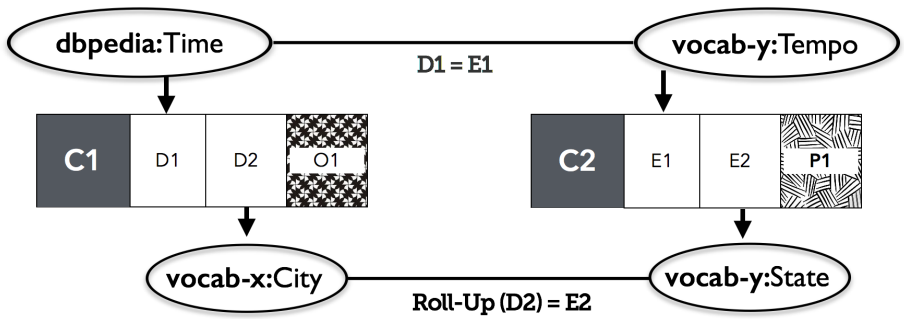


Figure 6.22: Data Cube `C1` & `C2` conformed dimensions

the submodule invokes the *R2ML on Demand* submodule to create R2RML mapping triples and sends it to the proper *Wrapper* for execution. The *Wrapper* executes the R2RML mapping triples (using the DB2Triples tool) and

generates a set of triples with the data cube observations. Finally, in the third stage the submodule *Federated Queries* consolidates the observation triples applying a JOIN over the conformed dimensions to generate the *Data Cube Mashup*. Figure 6.23 depicts the final stage to obtain the *Data Cube Mashup* `C3` merging the data cubes `C1` and `C2` over their conformed dimensions.

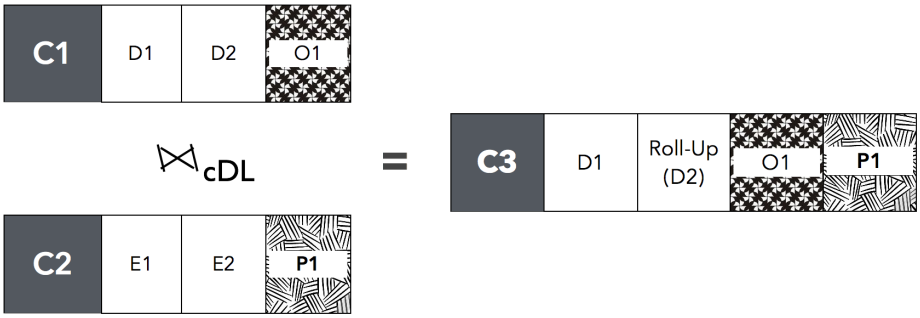


Figure 6.23: Mashup Data Cube *C3*

We now describe Algorithm 6 for the generation of *Data Cube Mashup*. The algorithm takes as input the data structure definition dsd_1 and dsd_2 of the data cubes c_1, c_2 and produces a set of triples OT_{Mashup} of the *Data Cube Mashup*.

The algorithm iteratively analyzes the dimension levels of c_1 and c_2 to identify the pair of dimension levels that are compatible. To do that, it uses several functions already described, such as `ExtractRelationalMetadata` in Section 6.3.1, `SliceOrDice` in Section 6.3.3, `R2RMLGeneration` in Section 6.3.4. Lines 2 and 3 obtain the set of dimension levels components DL_1, DL_2 and the set of measure components M_1, M_2 from the data structures dsd_1 and dsd_2 respectively. Lines 4 and 5 obtain the metadata $Metadata_{DL_1}$, $Metadata_{DL_2}$ and $Metadata_{M_1}$, $Metadata_{M_2}$ of the dimension levels DL_1, DL_2 and the set of measures M_1, M_2 using the `ExtractRelationalMetadata` function. Lines 6 through 22 traverse each pair of dimension levels (dl_i, dl_j) from $DL_1 \times DL_2$ to verify their compatibility. Lines 7 and 8 get the set of classes CL_i and CL_j of the dimension levels dl_i and dl_j, applying a SPARQL query over the *External Catalogue*. Lines 9 and 10 evaluate the schema compatibility of each pair $(cl_i, cl_j) \in CL_i \times CL_j$ verifying the connectivity between cl_i and cl_j through the *property path*[6] `owl:equivalentClass` of arbitrary length. Line 11 adds the filtered dimension levels to the conformed dimension level structure cDL. Lines 12 and 13 get the set of instances $DOM(dl_i)$ and $DOM(dl_j)$ from the *External Catalogue*. Lines 14 and 15 evaluate the data

[6]A property path is a possible route through a graph between two graph nodes. A trivial case is a property path of length exactly 1, which is a triple pattern.

compatibility of each pair $(dom(dl_i), dom(dl_j)) \in DOM(dl_i) \times DOM(dl_j)$ verifying the connectivity between $dom(dl_i)$ and $dom(dl_j)$ through the *property path* `owl:sameAs` of arbitrary length. Lines 16 and 17 add the filtered instances $dom(dl_i)$ and $dom(dl_j)$ into the metadata $Metadata_{DL_1}$ and $Metadata_{DL_2}$ respectively. These instances will be used in the SQL statement to filter the query result. Therefore in lines 23 and 24 it uses the `SliceOrDice` function to construct the SQL queries $qSQL_1$ and $qSQL_2$ filtering the instances found in the previous step. Lines 25 and 26 use the function `R2RMLGeneration` to generate the triples map rules $tMap_1$ and $tMap_2$. Line 27 uses these mapping rules $tMap_1$ and $tMap_2$ to generate the set of observation triples OT_1 and OT_2 for the dsd_1 and dsd_2 respectively. Finally line 28 executes a JOIN between these sets of observations OT_1 and OT_2 over their conformed dimension levels cDL to generate the new *Data Cube Mashup* OT_{Mashup}.

Algorithm 6: Generates a Data Cube Mashup.

input : dsd_1 and dsd_2 are the data structure definition of the data cubes c_1 and c_2 respectively

output: OT_{Mashup} is the set of observations of the new data cube mashup.

Function `Mashup`(dsd_1, dsd_2)
- Get the set of dimension level components DL_1 and DL_2 from dsd_1 and dsd_2 respectively
- Get the set of measure components M_1 and M_2 from dsd_1 and dsd_2 respectively
- $Metadata_{DL_1}, Metadata_{M_1}$ = `ExtractRelationalMetadata`(dsd_1, DL_1, M_1)
- $Metadata_{DL_2}, Metadata_{M_2}$ = `ExtractRelationalMetadata`(dsd_2, DL_2, M_2)
- **foreach** $(dl_i, dl_j) \in DL_1 \times DL_2$ **do**
  - Obtain the set of classes CL_i for dl_i in the External Catalogue
  - Obtain the set of classes CL_j for dl_j in the External Catalogue
  - **foreach** $(cl_i, cl_j) \in CL_i \times CL_j$ **do**
    - **if** $cl_i \xrightarrow[owl{:}equivalentClass]{*} cl_j$ **then**
      - Add equivalent levels (dl_i, dl_j) to cDL
      - Obtain the set of instances $DOM(dl_i)$ of type cl_i
      - Obtain the set of instances $DOM(dl_j)$ of type cl_j
      - **foreach** $(dom(dl_i), dom(dl_j)) \in DOM(dl_i) \times DOM(dl_j)$ **do**
        - **if** $dom(dl_i) \xrightarrow[owl{:}sameAs]{*} dom(dl_j)$ **then**
          - `add`$(Metadata_{DL_1}, \text{“}instances\text{”}, dom(dl_i))$
          - `add`$(Metadata_{DL_2}, \text{“}instances\text{”}, dom(dl_j))$
        - **end**
      - **end**
    - **end**
  - **end**
- **end**
- $qSQL_1$ = `SliceOrDice`$(Metadata_{DL_1}, Metadata_{M_1})$
- $qSQL_2$ = `SliceOrDice`$(Metadata_{DL_2}, Metadata_{M_2})$
- $tMap_1$ = `R2RMLGeneration`$(Metadata_{DL_1}, Metadata_{M_1}, qSQL_1)$
- $tMap_2$ = `R2RMLGeneration`$(Metadata_{DL_2}, Metadata_{M_2}, qSQL_2)$
- Execute the R2RML triplesMap $tMap_1$ and $tMap_2$ to generate the set of observation triples OT_1 and OT_2 for the dsd_1 and dsd_2 respectively.
- Execute a `JOIN` between OT_1 and OT_2 over their *conformed dimension levels* cDL to generate the new Data Cube Mashup OT_{Mashup}
- **return** OT_{Mashup}

7
Using the Olap2DataCube on Demand Framework

7.1
Introduction

This chapter presents a large-scale use case where the OLAP2Datacube framework was applied to Brazilian open government data.

Then, in Section 7.3 we step through the interfaces of the framework to illustrate the *Catalogue Construction Process* described in Section 5.1.

Finally, in Section 7.4 we apply the *Data Cube Consumption Process* described in Section 5.2 using some OLAP operations, specified in Chapter 6, to the aforementioned use case.

7.2
Open Government Data in Brazil

Efforts towards the publication of Open Government Data (OGD) in Brazil can be traced back to 2009, when the Information Organizing Committee of the Presidency (COI-PR) started to amass large amounts of aggregated government data for digital publication. The goal of the committee was to create a central information catalog of public activity, with the intent of improving governance, and monitoring government activity. This catalog was originally created to serve the President of the Republic and his team of advisors, as a reliable source of official data. The project was so successful that, reflecting open data principles, the catalog was made available to the general public in 2010[7].

In September 2011 Brazil became a member of the Open Government Partnership[8], a multinational initiative to promote worldwide adoption of OGD. As a participating member, Brazil committed to public transparency

[7] `https://i3gov.planejamento.gov.br/`
[8] `http://www.opengovpartnership.org/`

Table 7.2: Multidimensional Data Models Statistics (DadosGov).

Criterion	**Measurement**
Data Models	
Datasets	921
Fact Tables	921
Dimension Tables	6,333
Data Entries	
Fact Tables (tuples)	4,704,066
Geography Granularity	
Municipality	4,638,971
State	62,592
Brazil	2,503
Time Granularity	
Year	4,658,702
Month	45,364
Dimension Tables (tuples)	871,146
Data About	
Municipalities	5,320
States	28
Years	27
Governmental Agency Providers	77
Series	551

and action in securing open publication of official data. The commitment comprises political, as well as technical landmarks, including a presidential mandate for the launch of the Brazilian open government data portal[9].

The *DadosGov* information catalog comprises over 1,500 historic data series that reflect government activity during the mandate of President Luiz Inacio 'Lula' da Silva (2003 to 2010). The COI management team proposed a standard organization to classify the data, based on two dimensions: territorial (country, states, cities) and time (year or month). Data series were classified in several hierarchical thematic trees, that branched from general to more specific subjects, e.g., infrastructure, citizenship and social inclusion, as well as more specific subjects that define third and fourth level trees. Data (not in Linked Data format) is publicly available[10].Table 7.2 summarizes the *DadosGov* datasets which comprises more than 4 million observations covering three levels of administration in Brazil.

In addition to serving the general public via the website, *DadosGov* data is also published in XML and JSON. These formats allow data to be directly accessed at a persistent URL, so that it can be consumed and processed by

[9] `http://www.dados.gov.br/`
[10] `https://i3gov.planejamento.gov.br/`

software applications. However, neither XML nor JSON offer the semantic expression of RDF, nor interoperability with the rest of the LOD cloud. Thus to enjoy a true Linked Data experience, we needed to generate DadosGov RDF datasets that were "mashup-ready", i.e., represented using standard vocabularies that enable the creation of relevant local mashups, but also allow for the correlation to other datasets produced globally. The *OLAP2Datacube Framework* solution is given in the format of a process, to guide the production of the *DadosGov* datasets in RDF triples.

In order to illustrate the two major process of the *OLAP2Datacube Framework*, we will use two small datasets from the *DadosGov*.

1. *Population in Brazil by Municipalities*, the dataset was provided by the Ministry of Planning (MP), which describes the population projection in Brazil by municipalities (a measure) along the dimensions territorial and time. The territorial dimension is organized in a hierarchy of levels: municipality, state, country; while the time dimension has only one level: year.

2. *Mortality rates in Brazil by States*, the dataset was provided by the Ministry of Health, which describes the mortality rate in Brazil by state (a measure) along the dimensions territorial and time. The territorial dimension is organized in a hierarchy of levels: state, country; while the time dimension has only one level: year.

7.3 Catalogue Construction Process

Figure 7.24 illustrates the dashboard of the *OLAP2Datacube Framework*. From this interface we can navigate to the different modules of the framework. The dashboard also shows some metrics of the *Catalogue*, such as the number of datasets, dimensions, dimension levels and measures.

Figure 7.25 shows a list of databases registered in our framework. Using this interface we can add a new database, edit or remove an existing database. A similar interface is used to manage the framework users.

Figure 7.26 shows a list of data cubes identified by the *DCD tool* module following the heuristics described in Section 4.7, in the *Catalogue Exploration* stage (detailed in Section 5.1.2). Then, Figure 7.27 depicts the snowflake model schema of the population dataset inserted into the *Internal Catalogue* by the *DCD tool* module.

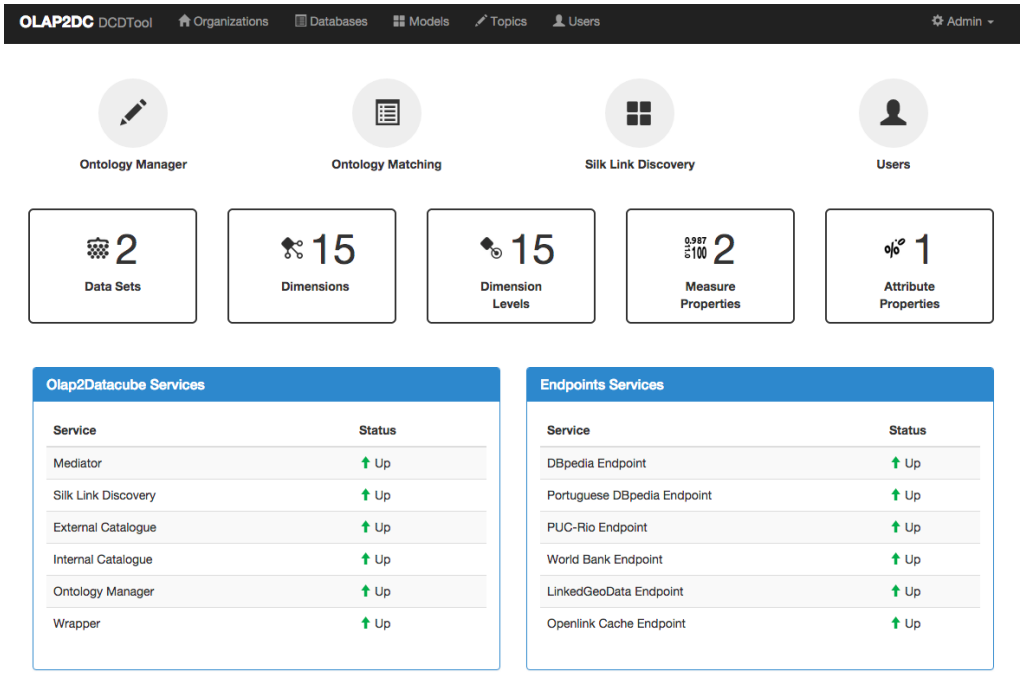


Figure 7.24: Dashboard OLAP2Datacube Framework

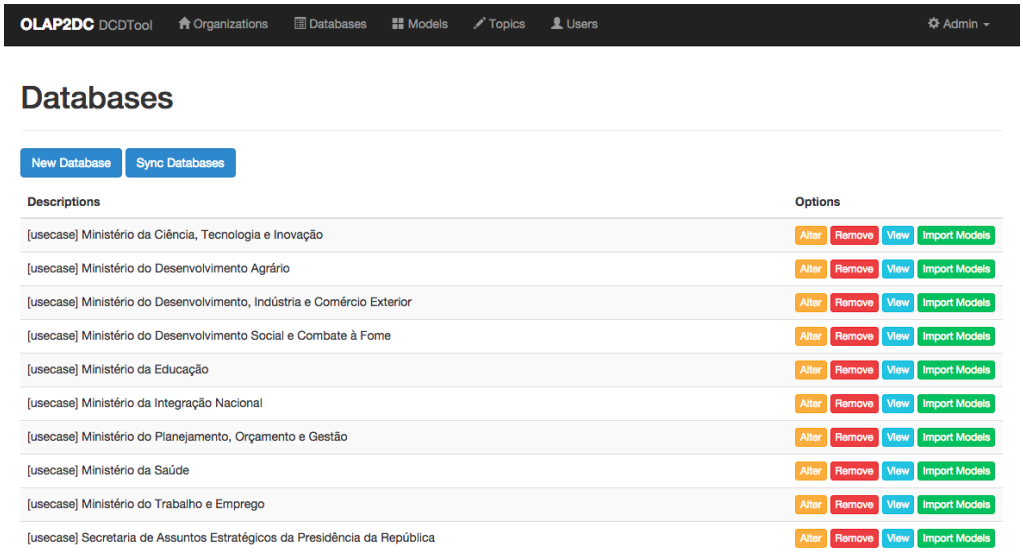


Figure 7.25: List of databases

Figure 7.28 depicts the semantic annotation of the State table with the class http://dbpedia.org/ontology/State. The figure exemplifies the *Metadata Enrichment* stage (detailed in Section 5.1.3) where the data administrator updates, complements and annotates the dimensional model components.

Figure 7.29 depicts the Linked Data Cube description of the population

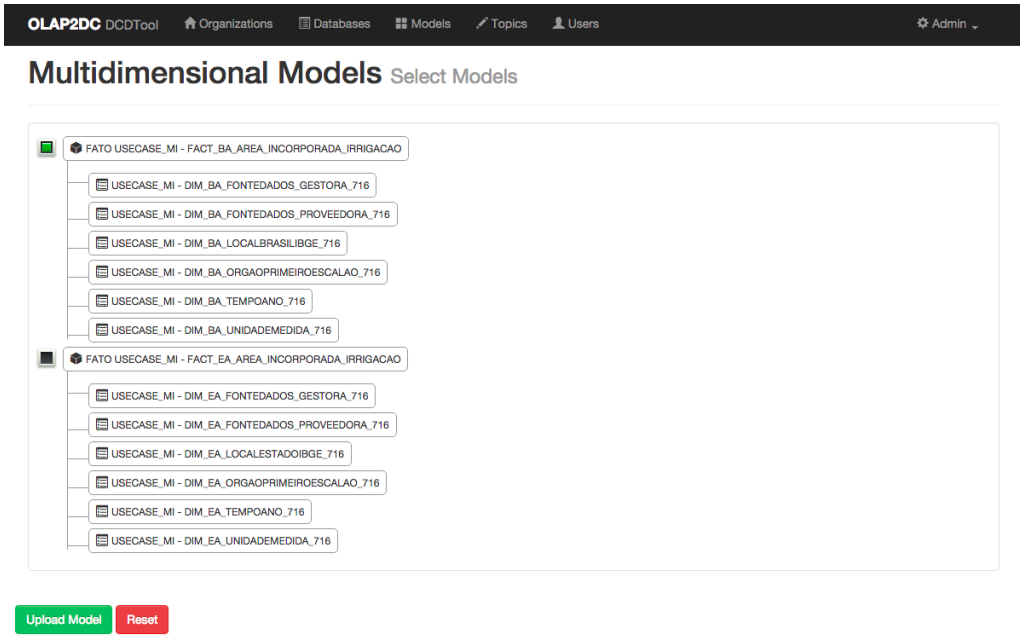


Figure 7.26: Multidimensional models in database

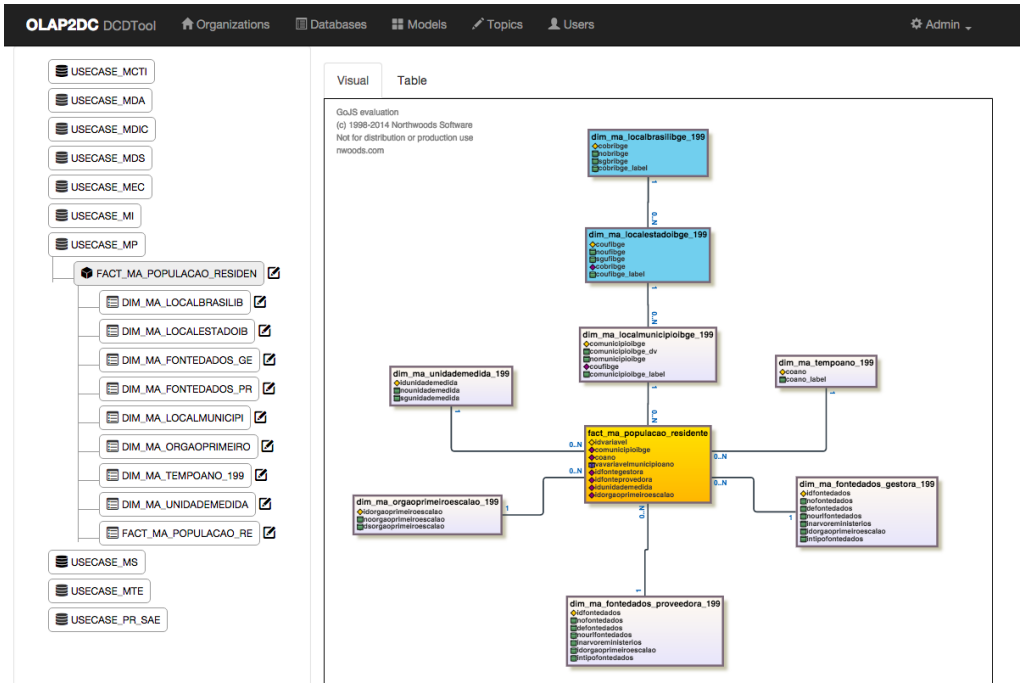


Figure 7.27: Multidimensional model schema of the “population” dataset

dataset as a result of the *Linked Data Cube Generation* stage (detailed in Section 5.1.4). Figure 7.30 shows the R2RML mapping rules used for the triplification of the dimension instances.

The *Linked Data Cube Enrichment* (detailed in Section 5.1.5) is illus-

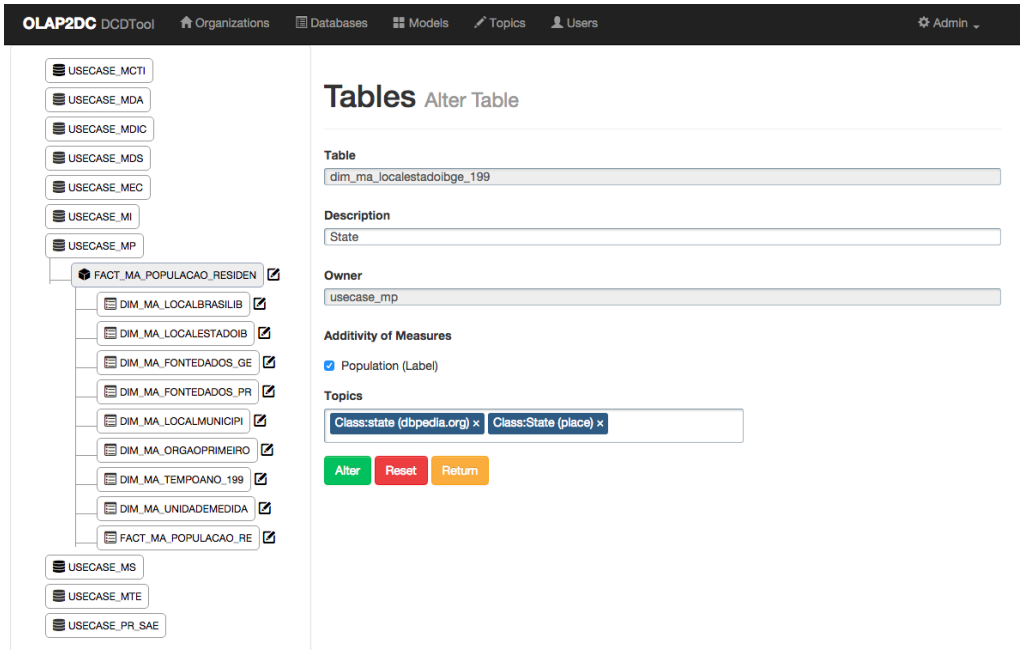


Figure 7.28: Semantic annotation of the "state" table

trated by Figure 7.31, which shows the interface of the ontology matching tool for the alignment of the Linked Data Cube concepts, and the Figure 7.32, which depicts the *Silk Record Linkage* tool for the alignment of the Linked Data Cube dimension instances.

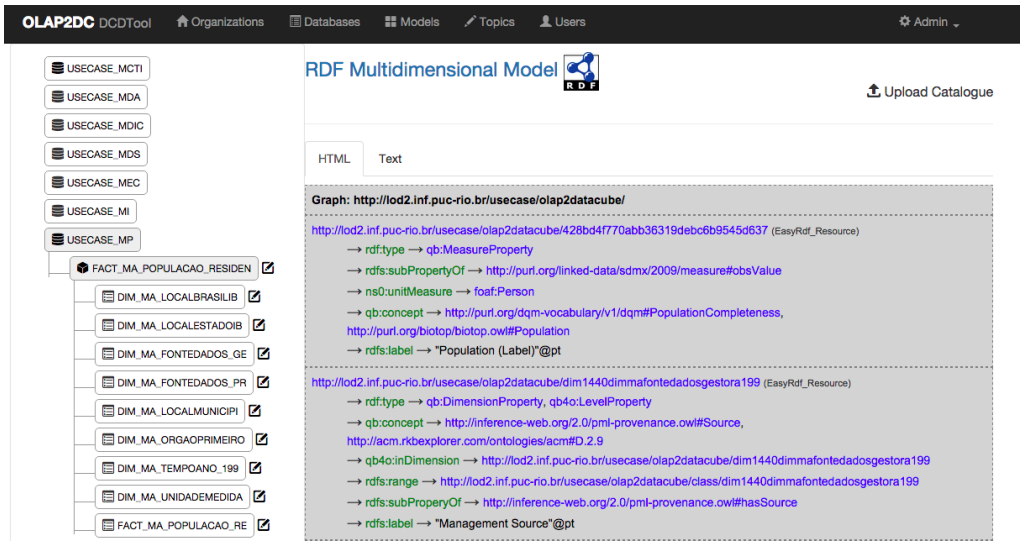


Figure 7.29: Linked Data Cube description of the "population" dataset

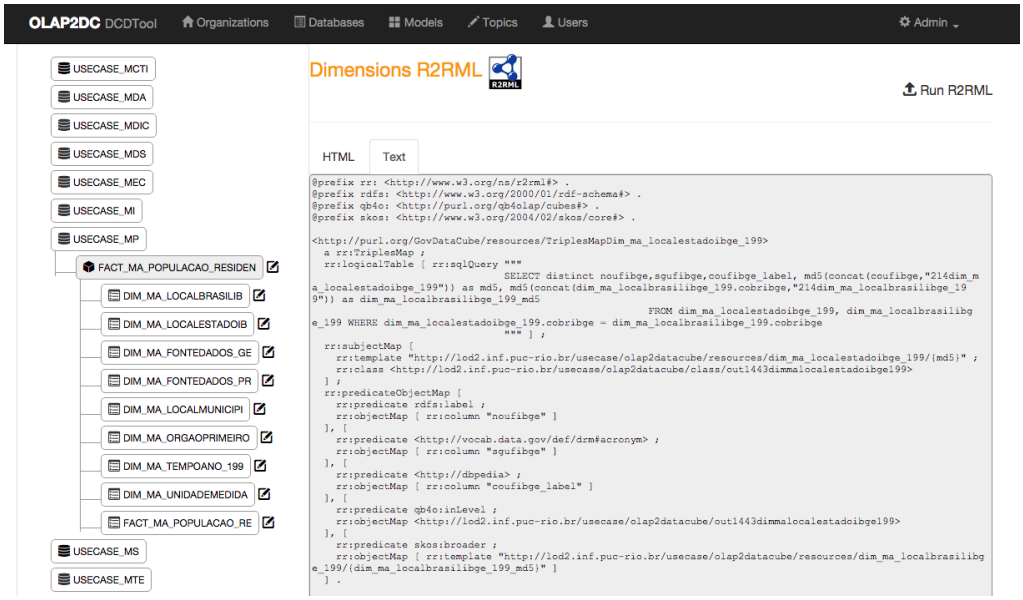


Figure 7.30: R2RML mapping rules of dimensions in the "population" dataset

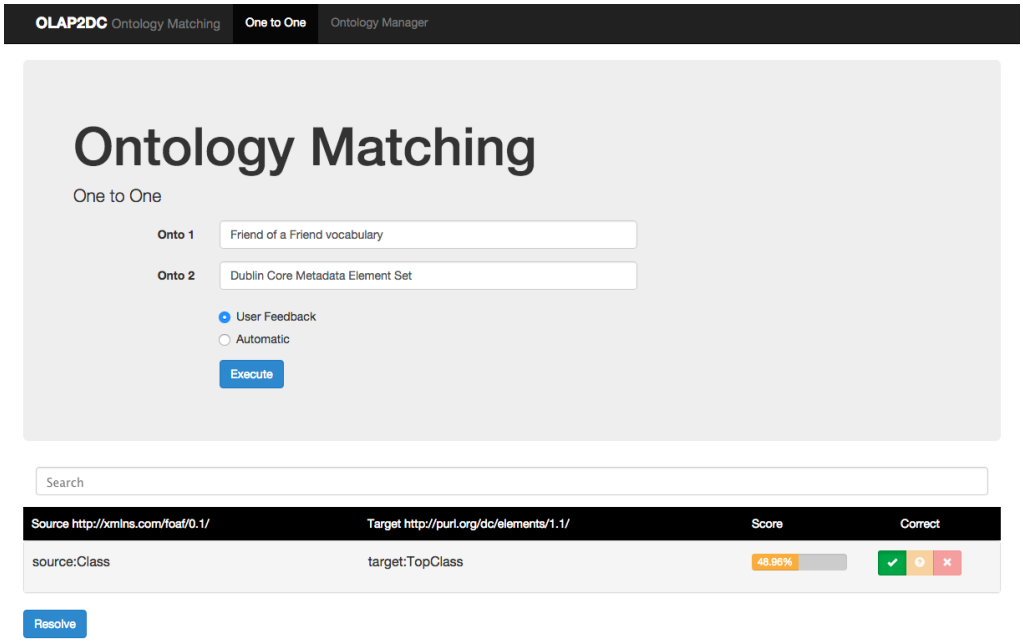


Figure 7.31: Ontology matching tool

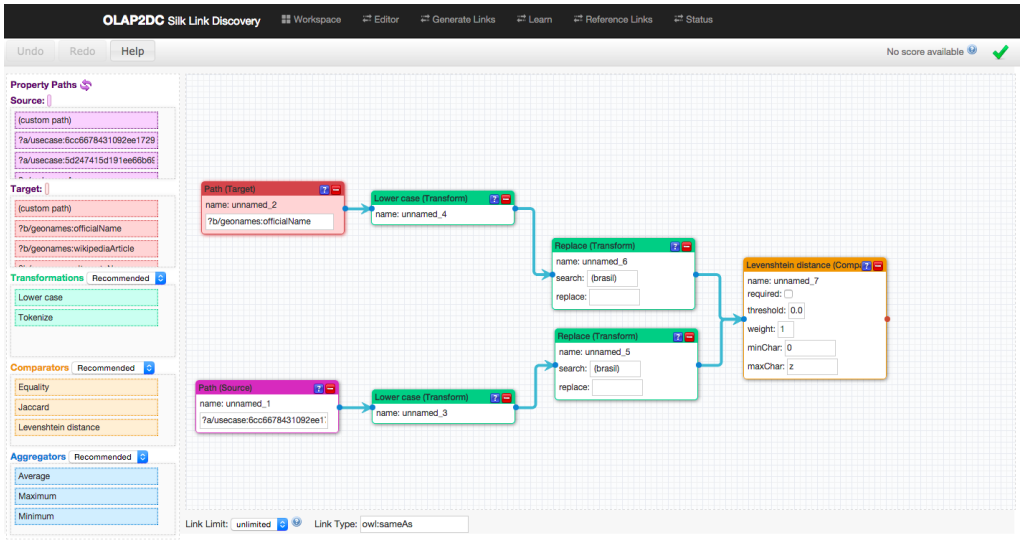


Figure 7.32: Silk link discovery tool

7.4 Data Cube Consumption Process

The *Mediator* offers an interface to browse the Linked Data Cubes registered in the *Catalogue* through a RESTful Web service. The keyword search HTTP method browses the *Catalogue* for a Linked Data Cubes description that satisfied a given word. For instance, Listing 7.1 shows the JSON response of the keyword search HTTP request for 'population'.

GET `http://host:port/mediator/getDatacubes?seed=population`

The user employs the JSON response to construct a data cube request. The request can involve a data cube transformation defined by OLAP operations, a more sophisticated mashup operation between different data cubes or a simple access to the data cubes observations on-the-fly (detailed in Section 5.2.2). Figure 7.33 depicts the multidimensional schemes used to demonstrate the mashup operation. Listing 7.2 shows the mashup request between the population and the mortality data cubes.

```
{ "models": [
    { "prefix" : [
      {"ex": "http://lod2.inf.puc-rio.br/usecase/olap2datacube/"}],
      "dimensions": [
        { "levels": [
            { "uri": "ex:dim1446dim_time199",
              "label": "Date and Time Dimension"}
        ]},
        { "levels": [
            { "uri": "ex:out1443dim_state199",
              "label": "State"}
        ]},
        { "levels": [
            { "uri": "ex:dim1444dim_city199",
              "label": "Municipality",
              "parent": "ex:out1443dim_state199"}
        ]}],
      "measures": [
        { "uri": "ex:428bd4f770abb36319debc6b9545d637",
          "label": "Population (Label)"}],
      "type": "fact",
      "uri": "ex:87b1fc9aa278fd9dc861355a54357819",
      "label": "http://purl.org/biotop/biotop.owl#Population"}
]}
```

Listing 7.1: JSON response of the keyword search data cube.

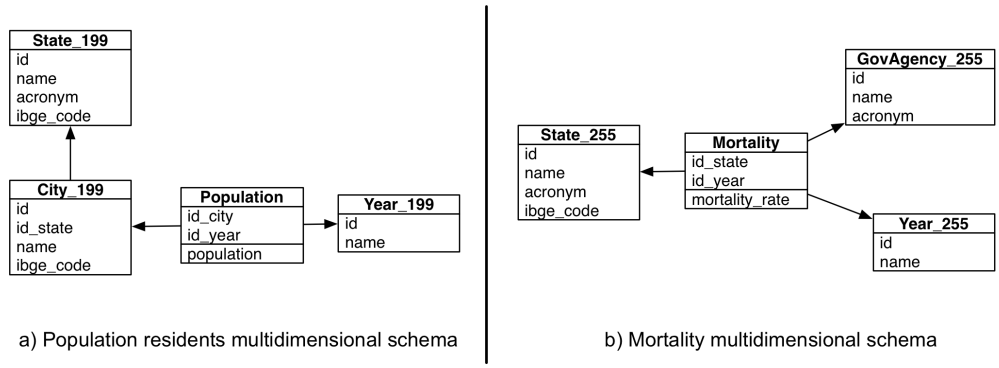


Figure 7.33: Mashup multidimensional schemes

The *Mediator* receives the data cube request and parse it to extract the relational metadata and identify the proper OLAP operation to fulfill the request (detailed in Section 6.3.1). In our use case, before combine the population and mortality data cubes, the *Mediator* applies a roll-up operation, described in Section 6.3.2, over city level in the population data cube. Once it has the dimension levels of the data cubes at the same granularity, the submodule *Mashup on Demand* applies the mashup operation using the algorithm described in Section 6.4. Listing 7.3 shows two observations (2 out of 378 observations) of the new data cube mashup 'population-mortality'.

```
@prefix qb: <http://purl.org/linked-data/cube#> .
@prefix sdmx: <http://purl.org/linked-data/sdmx/2009/attribute#> .
@prefix qb4o: <http://purl.org/qb4olap/cubes#> .
@prefix gdc: <http://purl.org/GovDataCube/>.
@prefix eg: <http://lod2.inf.puc-rio.br/usecase/olap2datacube/> .

eg:dsd-87b1fc9aa278fd9dc861355a54357819>
  a qb:DataStructureDefinition ;
  qb:component
  [
    qb:measure eg:428bd4f770abb36319debc6b9545d637>;
    qb4o:hasAggregateFunction qb4o:sum, gdc:percentage ],
  [
    qb:attribute sdmx::unitMeasure ;
    qb:componentAttachment qb:MeasureProperty
  ]
  [ qb4o:level eg:dim1446dim_time199> ],
  [ qb4o:level eg:out1443dim_state199> ].

eg:dsd-b12141325e2485c50ab88aea7ebcc98f>
  a qb:DataStructureDefinition ;
  qb:component
  [
    qb:measure eg:ac01dac8b20046c496caa2ec9d097641>;
    qb4o:hasAggregateFunction qb4o:sum, gdc:percentage ],
  [
    qb:attribute sdmx::unitMeasure ;
    qb:componentAttachment qb:MeasureProperty
  ]
  [ qb4o:level eg:dim1452dim_state2615> ]
  [ qb4o:level eg:dim1454dim_time2615> ].
```

Listing 7.2: Mashup data cube request.

The *Client Application* module uses the RDF mashup observations to create different types of visualization. For instance, Figure 7.35 depicts the comparison using different formats: geographical distribution, time series graph, and bar graphs that compare the situation per state and nationwide. More interesting still are the pop-ups, that can be observed in the map depictions. They present the total population of each state, the mortality rate, and also derived data, such as the percentage of the state population mortality. The coloring of the maps provides an overview of both series, the greener the state, the greater the percentage of its mortality rate. The bar graph on the right and the time series graph on the bottom provide a detailed, per state, analysis.

```
@prefix db-onto: <http://dbpedia.org/ontology/> .
@prefix eg: <http://lod2.inf.puc-rio.br/usecase/olap2datacube/> .
@prefix eg-res:  <http://lod2.inf.puc-rio.br/usecase/olap2datacube/resources/>
    .
@prefix qb: <http://purl.org/linked-data/cube#>
@prefix mashup: <http://lod2.inf.puc-rio.br/usecase/mashup/>

mashup:000a16a73fd3b73b7969a7ff3ed76788 a qb:Observation ;
    db-onto:percentage "0.6566" ;
    eg:428bd4f770abb36319debc6b9545d637 "3815000" ;
    eg:ac01dac8b20046c496caa2ec9d097641 "25050" ;
    eg:dim1446dimmatempoano199 eg-res:dim_time_199/6553
        f93773a01ea088e5c826fd6413d0 ;
    eg:out1443dimmalocalestadoibge199 eg-res:dim_state_199/
        c2d52a3c3bf97db94eb303e00ee14007 .

mashup:00741717299fd6d0966056d9e0772249 a qb:Observation ;
    db-onto:percentage "0.4405" ;
    eg:428bd4f770abb36319debc6b9545d637 "758800" ;
    eg:ac01dac8b20046c496caa2ec9d097641 "3343" ;
    eg:dim1446dimmatempoano199 eg-res:dim_time_199/6553
        f93773a01ea088e5c826fd6413d0 ;
    eg:out1443dimmalocalestadoibge199 eg-res:dim_state_199/66
        c7fe992cdffff58c9683aad281c4ba .
```

Listing 7.3: RDF data cube observations sample of the new data cube mashup.

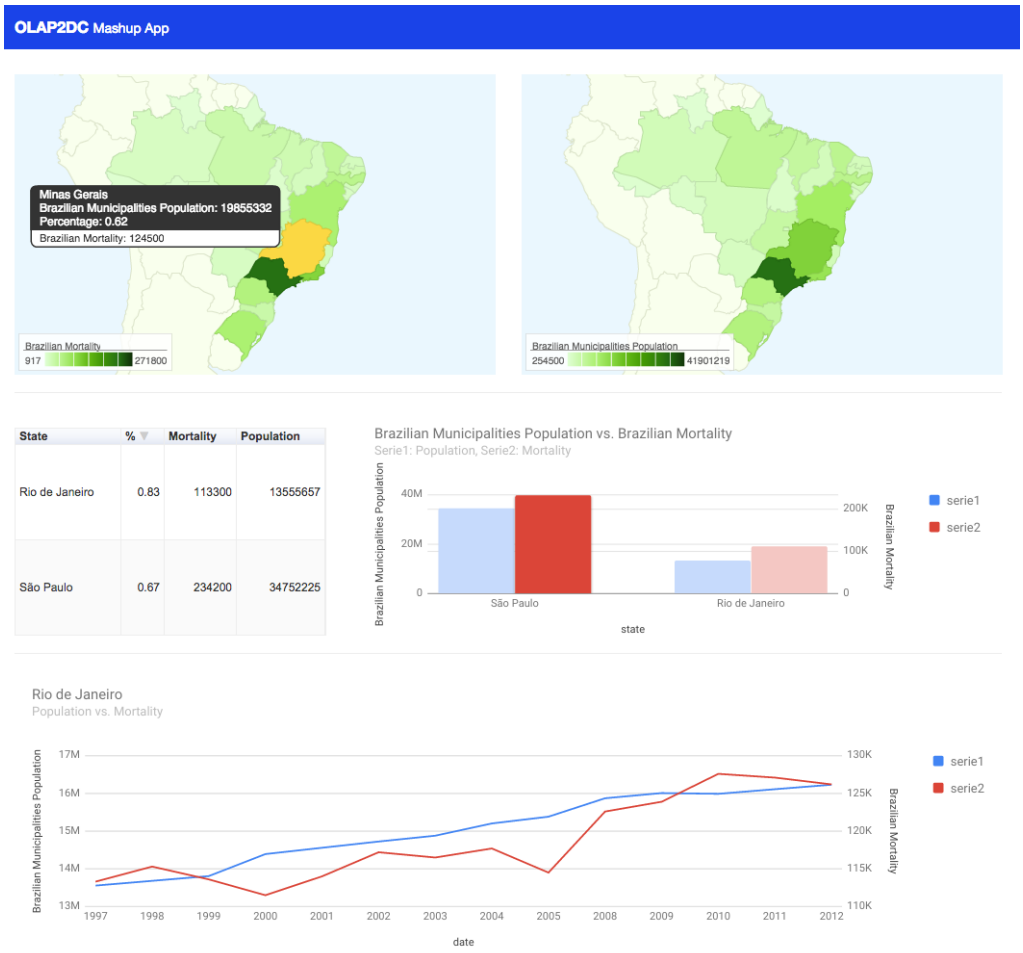


Figure 7.34: Mashup visualization between the population and the mortality data cubes.

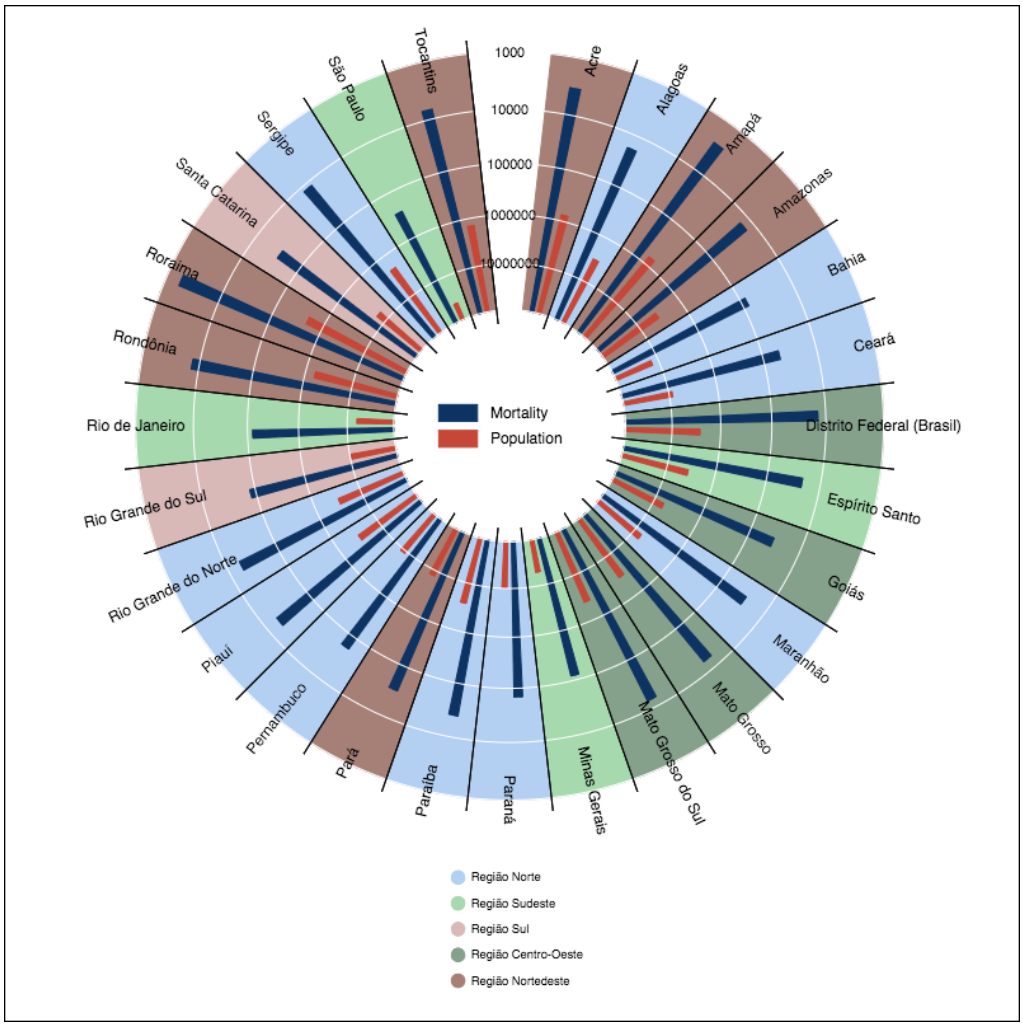


Figure 7.35: Radial mashup visualization

8
Conclusions and Future Work

8.1
Contributions

In [Salas et al. 2012], we introduced tools which facilitate statistical data management on the Web. We used two tools to create triples in the RDF DataCube Vocabulary. The tool *Olap2DataCube* supported collaborative creation, maintenance and publishing of triples via relational databases, and the CSV2DataCube was used for the conversion of statistical data stored in CSV files.

Using the experience gained during the development of the OLAP2DataCube tool, we produced the first version of the framework *Olap2DataCube On Demand* [Ruback et al. 2013]. This approach focused on mitigating the problems created by data redundancy, e.g., data synchronization. A mediation architecture was then developed for the description and consumption of statistical data in relational databases, exposed in RDF triples. The architecture has a catalog of data cube descriptions created in accord with Linked Data principles. The mediator provides an interface to navigate through the data cube descriptions and exports its observations in RDF triples, generated at runtime by the underlying relational databases.

Some limitations became apparent during the development of the framework *Olap2DataCube On Demand*. The first limitation is related to the queries for the creation of customized R2RML mappings. The mappings were static and returned the full content of a data cube in RDF triples. The second limitation was the lack of support for carrying out OLAP transformations/operations in the data cubes. The framework also did not support the integration of data cubes through data cube mashup operations. Finally, a separate installation and utilization of each individual module was necessary to operate the framework, mainly due to having been developed separately, without the benefit of an overview of the framework operation.

The main contribution of this thesis is a second version of the Olap2Datacube on Demand framework that addresses the limitations of the first version just summarized.

To this end the functionalities assigned to each module have been extended, supplemented and redistributed, in addition to creating new functionalities which enable the following runtime operations:

1. Creation and distribution of federated queries encapsulated in customized R2RML mappings;

2. Performing transformations and OLAP operations in data cubes; and

3. Building mashups from multiple data cubes.

In order to perform the OLAP operations on the data cubes, we had to modify the way in which the descriptions of Linked Data Cubes were created in *Catalogue*, as detailed in Section 4.3.1. Compiling the OLAP operations was also necessary, as detailed in Section 6.3. To accomplish this, we developed new submodules of the Mediator, viz., *Federated Queries* and *R2RML on Demand*. The first is in charge of the creation of federated queries in SQL through RDF requests, and the second generates the customized R2RML mapping from the SQL query.

To generate the data cube mashups, the use of ontology and instance alignment tools was required in order to identify the common dimensions that enable the combination of the data cubes. The module responsible for these alignment tasks is the *Linked Data Cube Enrichment*. This module was implemented using the LogMap[11] and SILK[12] tools as a basis, which implement the alignment of ontology and instance matching, respectively. The submodule *Mashups On Demand* of the *Mediator* was also implemented, which analyzes the feasibility of the creation of mashups between data cubes using the alignments stored in the *Catalogue* and combines the observations of the Linked Data Cubes generated separately in a new Data Cube Mashup.

The modules *Catalogue*, *Linked Data Cube Enrichment*, *Data Cube Discovery*, *Mediator* and *Wrapper* were implemented and integrated into a single solution to provide a unified management console in order to facilitate the use of the proposed framework. The OLAP2Datacube modules can be accessed through the unified console available at `http://lod2.inf.puc-rio.br/DCDTool/adm`.

[11] `https://code.google.com/p/logmap-matcher/`
[12] `http://wifo5-03.informatik.uni-mannheim.de/bizer/silk/`

To demonstrate the operation of the OLAP2Datacube on Demand Framework, this thesis introduced the two main processes: the *Catalogue Construction Process* and the *Data Cube Construction Process*, using a large-scale use case of statistical data published in Brazil.

8.2 Publications

The framework proposed in this thesis consists of different modules. The first version of each module was developed and described in detail in a separate master dissertation. In this thesis we implemented a second version of each module and an orchestration mechanism to integrate them into a single framework.

- The design and implementation of the first *Catalogue* version, created according to the Linked Data principles, was presented in [Manso 2013].
- The first version of the *Linked Data Cube Enrichment* was implemented and described in [Cabrera 2013];
- A primitive version of the *Mediator* was implemented and detailed in [Ruback et al. 2013].
- Finally, the first version of the *Data Cube Discovery Tool* was developed and described in [Ortiga 2013].

The publications of this work are listed below in two groups: journal articles and conference papers. Each group is presented in a chronological order.

- Journal Papers

  1. K. Breitman, P. E. R. Salas, M. A. Casanova, J. Viterbo, R. P. Magalhaes *Open government data in Brazil.* In IEEE Intelligent Systems, volume 27, pages 45–49, 2012.
  2. P. E. R. Salas, M. Martin, F. M. D. Mota, S. Auer, K. Breitman and M. A. Casanova. *Publishing Statistical Data on the Web.* In International Journal of Semantic Computing, volume 06, number 04, pages 373–388, 2012.
  3. E. Marx, P. E. R. Salas, K. Breitman, J. Viterbo, and M. A. Casanova. *RDB2RDF plugin: Relational to RDF plugin for eclipse.* Software: Practice and Experience, 2013.

– Conference Papers

1. P. E. R. Salas, M. Martin, F. M. D. Mota, S. Auer, K. Breitman and M. A. Casanova. *OLAP2DataCube: An ontowiki plug-in for statistical data publishing.* In Proceedings of the Second International Workshop on Developing Tools as Plug–Ins (TOPI'12), 2012.
2. P. E. R. Salas, M. Martin, F. M. D. Mota, S. Auer, K. Breitman and M. A. Casanova. *Publishing Statistical Data on the Web.* In 2012 IEEE Sixth International Conference on Semantic Computing, pages 285–292. IEEE, 2012.
3. L. Ruback, M. Pesce, S. Manso, S. Ortiga, P. E. R. Salas and M. A. Casanova. *A mediator for statistical linked data.* In Proceedings of the 28th Annual ACM Symposium on Applied Computing – SAC'13, 2013.

8.3 Future Work

As for future research, we suggest some improvements to the *Olap2Datacube on Demand* architecture, such as:

– Open source distribution of the *Olap2Datacube on Demand* framework. There was an effort to integrate the modules of the framework and make them easy to use and deploy. We created an API, called Olap2Datacube API, which provides several services of the *Mediator* to the general public on the Web. This framework could be distributed as open source software.

– Create a module to store the R2RML mapping history used to create data cubes, thus obviating the need to generate an R2RML mapping each time that a recurring request for the the same data cube is made.

– Create new wrappers that can extract statistical data from other data formats (e.g., JSON, CSV, XML). This would require creating a statistical data crawler to seek out cube sources on the Web and store them temporarily in a relational database using an OLAP schema.

– Create a submodule in the module *Linked Data Cube Enrichment* to generate the "authority" dimensions. Such dimensions would be the representative dimensions of a given concept. For instance, the IBGE (Brazilian Institute of Geography and Statistics) would be responsible for defining the "authority" dimension of geography (regions, states, municipalities, neighborhoods).

– Associate a metric for degree of completeness (confidence) at the time of roll-up operations on the data cubes. For example, a user would like to perform the roll-up operation on the state dimension level to a country dimension level, but the cube does not contain records for some states in the country. The degree of completeness of aggregation would therefore be less than 100%, depending on the number of missing records. This metric would be sent to the user along with the data cube triples.

Bibliography

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R. and Ives, Z. **DBpedia: A nucleus for a web of open data**. In *Proceedings of the 6th International Semantic Web Conference (ISWC)*, volume 4825 of *Lecture Notes in Computer Science*, pages 722–735. Springer, 2008.

Auer, S., Dietzold, S., Lehmann, J., Hellmann, S. and Aumueller, D. **Triplify: light-weight linked data publication from relational databases**. In *WWW '09: Proceedings of the 18th international conference on World wide web*, pages 621–630, New York, NY, USA, 2009. ACM.

Auer, S., Feigenbaum, L., Miranker, D., Fogarolli, A., Sequeda, J., Prud'hommeaux, E. and Hausenblas, M. **Use cases and requirements for mapping relational databases to RDF, W3C working draft**. Retrieved May 15, 2015, from `http://www.w3.org/TR/rdb2rdf-ucr/`, 2010.

Berners-Lee, T. **Putting Government Data online**. W3C Design Issue, 2009. `http://www.w3.org/DesignIssues/GovData.html`.

Bizer, C. and Cyganiak, R. **D2R Server - Publishing Relational Databases on the Semantic Web**. Poster at ISWC, 2006.

Bizer, C., Cyganiak, R. and Gauß, T. **The RDF book mashup: from web APIs to a web of data**. In *The 3rd Workshop on Scripting for the Semantic Web (SFSW 2007)*, Innsbruck, Austria, 2007.

Bizer, C., Cyganiak, R. and Heath, T. **How to publish linked data on the web**. Retrieved December 14, 2010, from `http://www4.wiwiss.fuberlin.de/bizer/pub/LinkedDataTutorial/`, 2007.

Breitman, K., Salas, P., Casanova, M. A., Saraiva, D., Gama, V., Viterbo, J., Magalhaes, R. P., Franzosi, E. and Chaves, M. **Open government data in Brazil**. In *IEEE Intelligent Systems*, volume 27, pages 45–49, May 2012.

Cabrera, X. **EnLiDa:Enriquecimento das descrições de Linked Data Cubes**. Master's thesis, *Pontifical Catholic University of Rio de Janeiro*, Rio de Janeiro, Brazil, 5 2013.

Caraballo, A. A. M., Nunes, B. P., Lopes, G. R., Leme, L. A. P. P., Casanova, M. A. and Dietze, S. **TRT - A tripleset recommendation tool**. In *Proceedings of the ISWC 2013 Posters & Demonstrations Track, Sydney, Australia, October 23, 2013*, pages 105–108, 2013.

Caraballo, A. A. M., Arruda, J., Narciso Moura, Nunes, B. P., Lopes, G. R. and Casanova, M. A. **Trtml - a tripleset recommendation tool based on supervised learning algorithms**. In Presutti, V., Blomqvist, E., Troncy, R., Sack, H., Papadakis, I. and Tordai, A., editors, *The Semantic Web: ESWC 2014 Satellite Events*, volume 8798 of *Lecture Notes in Computer Science*, pages 413–417. Springer International Publishing, 2014.

Chaudhuri, S. and Dayal, U. **An overview of data warehousing and olap technology**. *SIGMOD Rec.*, 26(1):65–74, Mar. 1997.

Codd, E. F., Codd, S. B. and Salley, C. T. **Providing OLAP (On-Line Analytical Processing) to User-Analysts: An IT Mandate**. E. F. Codd and Associates, 1993.

Cyganiak, R., Field, S., Gregory, A., Halb, W. and Tennison, J. **Semantic statistics: Bringing together sdmx and scovo.** In *LDOW*, volume 628 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2010.

Cyganiak, R., Hausenblas, M. and McCuirc, E. **Official statistics and the practice of data fidelity**. In *Linking Government Data*, 2011.

Cyganiak, R. and Reynolds, D. **The RDF Data Cube Vocabulary. W3C Recommendation**. Retrieved May 15, 2015, from `http://www.w3.org/TR/vocab-data-cube/`, 2014.

Das, S., Sundara, S. and Cyganiak, R. **R2RML: RDB to RDF Mapping Language. W3C RDB2RDF Working Group**. Retrieved December 15, 2014, from `http://www.w3.org/TR/r2rml/`, 2012.

Ding, L., DiFranzo, D., Graves, A., Michaelis, J., Li, X., McGuinness, D. L. and Hendler, J. **Twc data-gov corpus: incrementally generating linked government data from data.gov.** In *WWW*, pages 1383–1386. ACM, 2010.

Ding, L., Lebo, T., Erickson, J. S., DiFranzo, D., Williams, G. T., Li, X., Michaelis, J., Graves, A., Zheng, J. G., Shangguan, Z., Flores, J., McGuinness, D. L. and Hendler, J. **Twc logd: A portal for linked open government data ecosystems**. *J. of Web Semantics*, 2011.

Dunn, H. **Record linkage***. In *American Journal of Public Health and the Nations Health*, 1946.

Erickson, J. S., Rozell, E., Shi, Y., Zheng, J., Ding, L. and Hendler, J. A. **Twc international open government dataset catalog**. In *Proceedings of the 7th ICSS, I-Semantics '11*. ACM, 2011.

Erling, O. **Automated Generation of RDF Views over Relational Data Sources with Virtuoso**, 2009.

Etcheverry, L. and Vaisman, A. **QB4OLAP: A new vocabulary for olap cubes on the semantic web**. In *Proceedings of the Third International Workshop on Consuming Linked Data (COLD 2012) in 11th International Semantic Web Conference 2012 (ISWC 2012)*, 2012.

Etcheverry, L. and Vaisman, A. A. **Enhancing OLAP analysis with web cubes**. In Simperl, E., Cimiano, P., Polleres, A., Corcho, O. and Presutti, V., editors, *In Proceedings of the 9th international conference on The Semantic Web: research and applications, ESWC'12*, volume 7295 of *Lecture Notes in Computer Science*, pages 469–483, Berlin, Heidelberg, May 2012. Springer Berlin Heidelberg.

Euzenat, J. and Shvaiko, P. **Ontology matching**. Springer-Verlag, Heidelberg (DE), 2007.

Ghasemi, S., Luk, W.-S. and Alrayes, N. **M2rml: Multidimensional to rdf mapping language**. In *Database and Expert Systems Applications (DEXA), 2014 25th International Workshop on*, pages 263–267, Sept 2014.

Han, J. **Data Mining: Concepts and Techniques**. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

Hausenblas, M., Halb, W., Raimond, Y., Feigenbaum, L. and Ayers, D. **Scovo: Using statistics on the web of data.** In *In Proceedings of the 6th European Semantic Web Conference on The Semantic Web: Research and Applications, ESWC'2009*, volume 5554 of *LNCS*. Springer, 2009.

Heath, T. and Bizer, C. **Linked Data: Evolving the Web into a Global Data Space (1st edition)**. Morgan & Claypool Publishers, Feb 2011.

Hoxha, J., Brahaj, A. and Vrandečić, D. **Open.data.al: Increasing the utilization of government data in albania**. In *Proceedings of the 7th International Conference on Semantic Systems, I-Semantics '11*, pages 237–240, New York, NY, USA, 2011. ACM.

Janev, V., Nuffelen, B., Mijovi, V., Kremer, K., Martin, M., Miloševi, U. and Vrane, S. **Supporting the linked data publication process with the lod2 statistical workbench**. *Semantic Web Journal (under review), http://www.semantic-web-journal.net/content/supporting-linked-data-publication-process-lod2-statistical-workbench*, 2014.

Kämpgen, B. and Harth, A. **Transforming statistical linked data for use in olap systems**. In *In Proceedings of the 7th International Conference on Semantic Systems, I-Semantics'11*, pages 33–40, New York, NY, USA, 2011. ACM.

Kämpgen, B., O'Riain, S. and Harth, A. **Interacting with statistical linked data via OLAP operations**. In *Proceedings of the Workshop on Interacting with Linked Data (ILD 2012) in the 9th Extended Semantic Web Conference (ESWC 2012)*, 2012.

Kämpgen, B. and Harth, A. **OLAP4LD – A Framework for Building Analysis Applications Over Governmental Statistics**. In *The Semantic Web: ESWC 2014 Satellite Events*, volume 8798 of *Lecture Notes in Computer Science*, pages 389–394. Springer International Publishing, 2014.

Kimball, R. **What not to do**. Retrieved August, 2013, from `http://www.kimballgroup.com/2001/10/24/what-not-to-do/`, 2 2001.

Kimball, R. and Ross, M. **The data warehouse toolkit: The complete guide to dimensional modeling**. John Wiley & Sons, Inc., New York, NY, USA, 2nd edition, 2002.

Knap, T., Michelfeit, J. and Daniel, J. **ODCleanStore: a framework for managing and providing integrated linked data on the web**. In *... Engineering-WISE 2012*, 2012.

Maali, F., Cyganiak, R. and Peristeras, V. **Enabling interoperability of government data catalogues**. In *Proc. of the 9th IFIP, EGOV'10*, 2010.

Manola, F. and Miller, E. **Rdf primer, w3c recommendation**. Retrieved January 18, 2011, from `http://www.w3.org/TR/rdf-primer/`, 2 2004.

Manso, S. **Catalogue of Linked Data Cube Descriptions**. Master's thesis, *Pontifical Catholic University of Rio de Janeiro*, Rio de Janeiro, Brazil, 6 2013.

Martin, M., Kaltenböck, M., Nagy, H. and Auer, S. **The Open Government Data Stakeholder Survey**. In *OKCon*. OKFN, 2011.

Mendes, P. N., Mühleisen, H. and Bizer, C. **Sieve: Linked data quality assessment and fusion**. In *Proceedings of the 2012 Joint EDBT/ICDT Workshops*, *EDBT-ICDT '12*, pages 116–123, New York, NY, USA, 2012. ACM.

Mountantonakis, M., Allocca, C., Fafalios, P., Minadakis, N., Marketakis, Y., Lantzaki, C. and Tzitzikas, Y. **Extending void for expressing connectivity metrics of a semantic warehouse**. In *Proceedings of the 1st International Workshop on Dataset PROFIling & fEderated Search for Linked Data co-located with the 11th Extended Semantic Web Conference, PROFILES@ESWC 2014, Anissaras, Crete, Greece, May 26, 2014.*, 2014.

Ngonga Ngomo, A. and Auer, S. **LIMES - A Time-Efficient Approach for Large-Scale Link Discovery on the Web of Data**. In *Proc. of IJCAI*, 2011.

Noy, N. and Rector, A. **Defining N-ary Relations on the Semantic Web**. Technical report, W3C, 2006.

Management of Statistical Metadata at the OECD, 2006.

Ortiga, S. **LDCD Tools: Um conjunto de ferramentas para descoberta e triplificação de cubos de dados estatísticos**. Master's thesis, *Pontifical Catholic University of Rio de Janeiro*, Rio de Janeiro, Brazil, 5 2013.

Petrou, I., Meimaris, M. and Papastefanatos, G. **Towards a methodology for publishing linked open statistical data**. *eJournal of eDemocracy & Open Government*, 6(1), 2014.

Prud'hommeaux, E. and Seaborne, A. **SPARQL Query Language for RDF. W3C Recommendation**. Retrieved May 15, 2015, from `http://www.w3.org/TR/rdf-sparql-query/`, 2008.

Ross, M. **The 10 essential rules of dimensional modeling**. Retrieved August, 2013, from `http://www.kimballgroup.com/2009/05/29/the-10-essential-rules-of-dimensional-modeling/`, 2 2009.

Ruback, L., Pesce, M., Manso, S., Ortiga, S., Salas, P. E. R. and Casanova, M. A. **A mediator for statistical linked data**. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing - SAC '13*, page 339, New York, New York, USA, Mar. 2013. ACM Press.

Statistical data and metadata exchange (SDMX). Technical report, Standard No. ISO/TS 17369:2005, 2005.

Saad, R., Teste, O. and Trojahn, C. **OLAP Manipulations on RDF Data following a Constellation Model**. In *Proceedings of the First International Workshop on Semantic Statistics (SemStats'13) in conjunction with the 12th International Semantic Web Conference (ISWC'13)*, 2013.

Salas, P. E. R., Martin, M., Mota, F. M. D., Auer, S., Breitman, K. and Casanova, M. A. **Publishing Statistical Data on the Web**. In *2012 IEEE Sixth International Conference on Semantic Computing*, pages 285–292. IEEE, Sept. 2012.

Schultz, A., Matteini, A. and Isele, R. **LDIF-A Framework for Large-Scale Linked Data Integration**. In *21st International World Wide Web Conference (WWW2012)*, 2012.

Sheridan, J. and Tennison, J. **Linking UK Government Data**. In *WWW2010 Workshop on Linked Data on the Web (LDOW)*, 2010.

Tzitzikas, Y., Minadakis, N., Marketakis, Y., Fafalios, P., Allocca, C., Mountantonakis, M. and Zidianaki, I. **MatWare: Constructing and Exploiting Domain Specific Warehouses by Aggregating Semantic Data**. In *The Semantic Web: Trends and Challenges*, volume 8465 of *Lecture Notes in Computer Science*, pages 721–736. Springer International Publishing, 2014.

Guidelines for statistical metadata on the internet. Technical report, United Nations, Economic Commission for Europe (UNECE), 2000.

Vassiliadis, P. **Modeling multidimensional databases, cubes and cube operations**. In *Proceedings of the 10th International Conference on Scientific and Statistical Database Management, SSDBM '98*, pages 53–62, Washington, DC, USA, 1998. IEEE Computer Society.

Vidal, V. M., Casanova, M. A., Arruda, N., Roberval, M., Leme, L. P., Lopes, G. R. and Renso, C. **Specification and incremental maintenance of linked data mashup views**. In Zdravkovic, J., Kirikova, M. and Johannesson, P., editors, *Advanced Information Systems Engineering*, volume 9097 of *Lecture Notes in Computer Science*, pages 214–229. Springer International Publishing, 2015.

Volz, J., Bizer, C., Gaedke, M. and Kobilarov, G. **Discovering and maintaining links on the web of data**. In *Proceedings of the 8th*

International Semantic Web Conference, *ISWC '09*, pages 650–665, Berlin, Heidelberg, 2009. Springer-Verlag.

Zancanaro, A., Pizzol, L. D. and Speroni, R. **Publishing Multidimensional Statistical Linked Data**. In *In Proceedings of the Fifth International Conference on Information, Process, and Knowledge Management (eKNOW 2013)*, 2013.