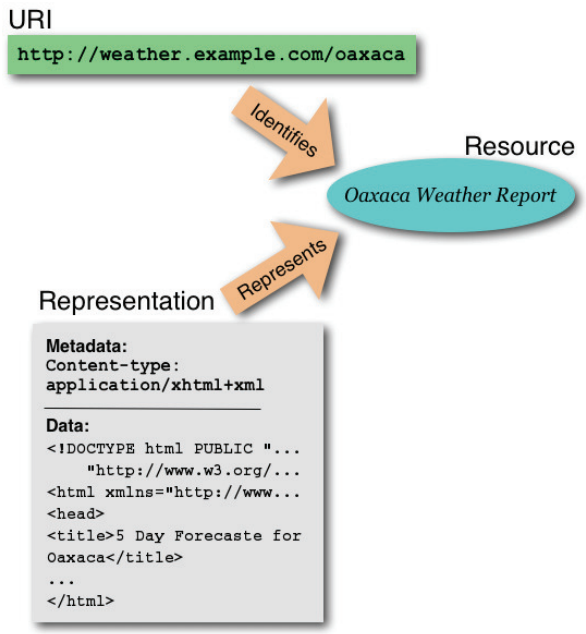
2
Background

This chapter provides an overview of the main concepts related to this dissertation. Section 2.1 introduces the key concepts of the architecture of the World Wide Web: resources, their representations and the notion of uniform identifier (URI). Section 2.2 presents the main features of the application-level protocol used by the WWW, the HTTP Protocol. Section 2.3 presents how the Linked Data Principles emerged from these same architectural bases, how they fit in the Semantic Web and presents the main W3C research group related to Linked Data: the Linking Data Open Project. Section 2.4 introduces the RDF model, the building block of the Semantic Web and Section 2.5 presents the SPARQL Query Language, the language created to express queries across RDF graphs. Section 2.6 concerns data cubes, and includes a definition and the RDF vocabulary created to represent them. Section 2.7 presents the two approaches to convert relational databases into RDF models: the direct mapping approach and the customized mapping approach, also known as the R2RML approach. Finally, Section 2.8 presents the Web Service Architecture, its architecture stack and the REST approach to consuming Web services.

2.1
The World Wide Web architecture

The *World Wide Web* (*WWW*, or simply *Web*) is defined as “an information space in which the items of interest, referred to as *resources*, are identified by global identifiers called *Uniform Resource Identifiers* (*URI*)” (Ian Jacobs 2004).

The Web is composed of three architectural bases: *identifier* (URI), *resource,* and *representation*. Figure 1 shows the relationship between these components using a weather report scenario. The *resource* in this example is the report on the weather in Oaxaca, which is regularly updated. This resource is uniquely identified by an *identifier*, the URI `http://weather.example.com/oaxaca`. It is important to notice that the term *resource* is used in a general sense for anything that might be identified by a URI, which can include Web pages, images, and product catalogs.



```
Metadata:
Content-type:
application/xhtml+xml

Data:
<!DOCTYPE html PUBLIC "...
    "http://www.w3.org/...
<html xmlns="http://www...
<head>
<title>5 Day Forecaste for
Oaxaca</title>
...
</html>
```

Figure 1: The three architectural bases of the Web (Ian Jacobs 2004).

To access this resource, the client (a browser or a Web agent) sends a request to the server at `weather.example.com` and the server returns a message that contains the representation of the resource at the time that the representation was generated, for instance a HTML or a XML document (Ian Jacobs 2004). The communication between the client and the server is specified by the HTTP protocol.

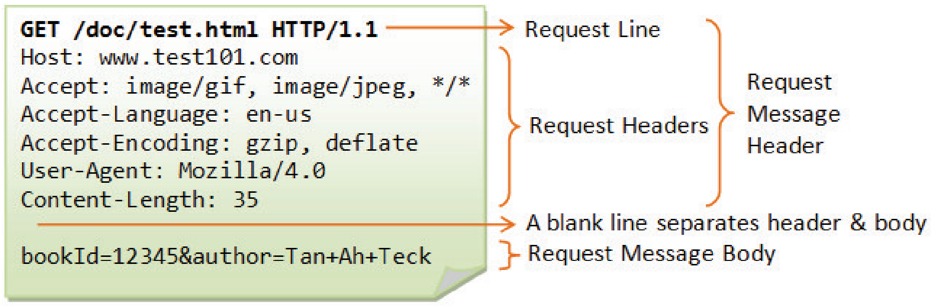
2.2 The HTTP Protocol

HTTP has been used by the Web since 1990 and is specified in the RFC (Request for Comments) 2068[1]. According to W3C, “the Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless protocol which can be used for many tasks beyond its use for hypertext, such as name servers and

[1] http://www.ietf.org/rfc/rfc2616.txt

distributed object management systems, through extension of its request methods, error codes and headers".

The basic unit of the HTTP communication is the *HTTP message*, which is a request type (from client to server) or a response type (from server to client). Both message types consist of a *start-line*, zero or more header fields (or just *headers*), an *empty line* indicating the end of the header fields and possibly a *message-body*. Figure 2 illustrates a request type HTTP message. The request headers of this figure indicate which data formats and language the client prefers.



```
GET /doc/test.html HTTP/1.1
Host: www.test101.com
Accept: image/gif, image/jpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
Content-Length: 35

bookId=12345&author=Tan+Ah+Teck
```

Figure 2: A HTTP request method example[2]

The process of selecting the best representation out of the multiple representations available for a given response is one of the most important features of the HTTP protocol and is called *content negotiation* (Heath and Christian Bizer 2011). The *accept* request-header field is used to specify certain media types that are acceptable for the response; the *accept-language* is similar to accept, but it restricts the set of preferred natural languages; the accept-encoding restricts the content-codings; the *user-agent* is concerned with the user agent originating the request and the *content-length* indicates the size of the body.

The HTTP message in Figure 2 also contains a GET method, one of the common methods defined by the HTTP specification, known as *HTTP verbs*. Table 1 shows the main HTTP methods and their descriptions (Ian Jacobs 2004).

Table 1: The main HTTP methods

HTTP method	Method description
GET	Retrieves a representations of a resource
POST	Creates a new resource

[2] http://www.bindichen.co.uk/post/Fundamentals/HTTP%20Basics.html

PUT	Updates an existing resource
DELETE	Deletes the resource itself

When any of these methods are requested by the client, the server returns a code with information about the status of the request, called the *HTTP status code*. Table 2 describes some main HTTP status codes, defined in the RFC[3].

Table 2: Some of the HTTP status codes[2]

Status class	**Status code**	**Status name**	**Status description**
1xx (provisional)	100	Continue	The client SHOULD **continue** with its request
	101	Switching Protocols	The client has asked the server to **switch protocols**
2xx (successful)	200	OK	The **request has successful**. For GET requests, the body contains the entire object requested
	201	Created	The request has been fulfilled and a **new resource is created**, which can now be referenced
3xx (redirection)	300	Multiple Choices	The resource is available **at multiple representations**
	304	Not Modified	The **resource has not changed**. Is used when the client makes a conditional GET request (e.g If-modified-since)
4xx (client error)	400	Bad Request	The request could not be understood by the server due to **malformed sintax**
	401	Unauthorized	The request **requires user authentication**. The response includes a authenticate header field
	403	Forbidden	The server understood the request, but is refusing to fulfill it. (for example, due to file permissions)
	404	Nod found	The requested **resource does not exist** on the server. The server has not found anything matching the requested URI.
5xx (server error)	500	Internal Server Error	The server **encountered an unexpected condition** which prevented it from fulfilling the request.
	501	Not Implemented	The **server does not support the functionality** required to fulfill the request. It does not recognize the method.
	503	Service unavailable	The server **is unable to handle the request** due to a temporary overloading or maintenance of the server.

[3] http://www.ietf.org/rfc/rfc2616.txt

2.3 The Linked Data concept

The classic Web or *Web of documents* presented in section 2.1 connects documents through hyperlinks creating a single global information space. A Web document is based on the following architectural standards: the HTTP protocol, detailed in section 2.2, as the universal access mechanism; URIs, as a globally unique identification mechanism; and the HTML (Hypertext Markup Language), as a content format (Heath and Christian Bizer 2011).

2.3.1. The Linked Data Principles

Basically, the Linked Data applies the general architecture of the WWW to share structured data on a global scale (Heath and Christian Bizer 2011). It was conceived as a set of principles published as a Web architecture note[4] by Berners-Lee in 2007 that contains a set of good practices when publishing and interlinking structured data on the Web. The four Linked Data Principles are:

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, provides useful information, using the standards (RDF, SPARQL).
4. Include links to other URIs, so they can discover more things.

The first principle uses URIs on the Web to *identify any object or concept* in the world, and not just to identify Web documents as in the classic Web. URIs must reference things such as people, places or even abstract concepts as a type of relationship (for instance, “knowing somebody”).

The second principle combines HTTP and URIs to better identify these objects and concepts, in such a way that the *URIs can be dereferenced over the HTTP* protocol into a description. This differs from the classic Web because the HTTP URIs are used as a simple retrieval mechanism.

[4] http://www.w3.org/DesignIssues/LinkedData.html

The third principle introduces *standardizing content format* to enable different applications to process Web content. It advocates the use of a simple graph-based model to represent and share structured data on the Web, known as RDF (Resource Description Framework) (Heath and Christian Bizer 2011) (Manola and Miller 2013). Section 2.4 describes this model in more detail.

The last principle essentially details what it means to *insert the data in a context*. While on the classic Web the hyperlinks are not typed, in a Linked Data context RDF hyperlinks have typed descriptive relationships between things or concepts, such as the relation "friend of" between two people. Therefore, the hyperlinks in Linked Data, known as RDF hyperlinks, connect not only documents, but any type of thing on the Web as well (Heath and Christian Bizer 2011).

Following the aforementioned principles based on the general architecture of the Web, data can be published in such a manner that they are machine-readable, that their meaning is explicitly defined, and that they can be linked to other external data sets (Heath and Christian Bizer 2011). Organizations have adopted Linked Data to publish their data by not just placing it *on* the Web, but also by using Linked Data to ground it *in* the Web (Heath and Christian Bizer 2011). By grouping data following this basic recipe, the data become part of a single global data space called the *Web of Data* (Heath 2009).

2.3.2. The Linking Open Data Project

The *W3C Linking Open Data Project*[5] started on February 2007 by Chris Bizer and Richard Cyganiak. The aim of the project was to bootstrap the Web of Data by identifying existing datasets available under open licenses and to publishing them in RDF according to the Linked Data Principles (Heath and Christian Bizer 2011). The project included the participation of universities (FU Berlin, MIT, KMi/The Open University, Universities of Pennsylvania, Leipzig, London, Hannover, Galway, among others) and some companies (OpenLink Software, Talis, Zitgist, BBC) (Heath 2008).

[5] http://linkeddata.org/

Figure 3 shows the datasets that have been published by contributors to the project until September 2011. The metadata collected is curated by the Data Hub[6]members. Each node of the diagram represents a dataset published as Linked Data and the arcs represent links between items from one dataset to another. Currently, the LOD Cloud encompasses 337 datasets[7] and is maintained by the LOD within the Comprehensive Knowledge Archive Network (CKAN[8]), a generic catalog that lists open-license datasets.

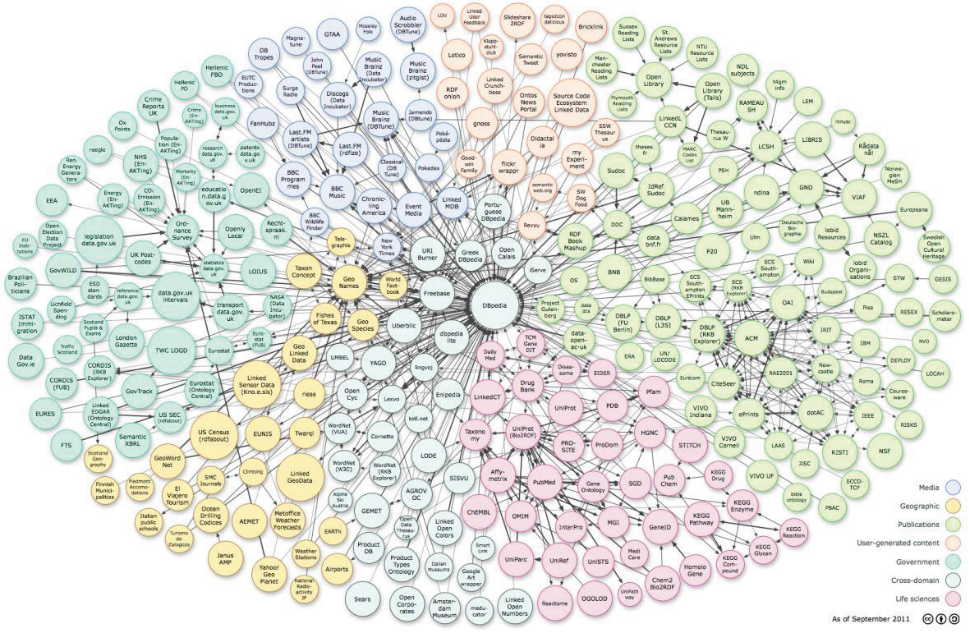


Figure 3: The LOD Cloud Diagram at September 2011[9]

In order to deal particularly with government-related datasets, the *Linking Open Government Data (LOGD) project*[10] investigates opening and linking government data using Semantic Web technologies.

2.4 RDF

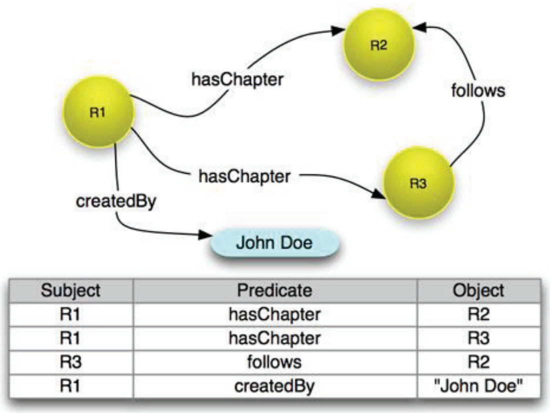
The third of the Linked Data Principles listed in the previous section is to provide useful information. This topic is essential in the context of consuming

[6] http://datahub.io/
[7] http://datahub.io/dataset?groups=lodcloud
[8] http://ckan.org/
[9] http://lod-cloud.net/
[10] http://logd.tw.rpi.edu/

data on the Web because of data reuse. The more regular and well-defined the structure of the data is, the easier people and software agents can reliably process them for reuse (Heath and Christian Bizer 2011).

To enable heterogeneous applications to consume and reuse Web content, it is essential to agree on a standardized data format. To address this issue when publishing Linked Data on the Web, data are represented using the *Resource Description Framework* (RDF) (Manola and Miller 2013). RDF is a simple graph-based data model, which describes a resource as a *triple*. Each triple has three parts: a *subject*, a *predicate* and an *object*. The triple *subject* is the URI that identifies the resource (for instance, `http://weather.example.com/oaxaca`). The *object* can also be another URI or can be a literal value. Finally, the *predicate* represents the type of relation between both the object and the subject (Heath and Christian Bizer 2011).

RDF graphs contain subjects related to more than one object, shown in Figure 4. Resource R1 is part of three triples in the subject role: <R1, hasChapter, R2>, <R1, hasChapter, R3> and <R1, createdBy, “John Doe”>. Resource R3 is also the subject of another triple, <R3, follows, R2>.



Subject	Predicate	Object
R1	hasChapter	R2
R1	hasChapter	R3
R3	follows	R2
R1	createdBy	"John Doe"

Figure 4: An RDF graph example[11]

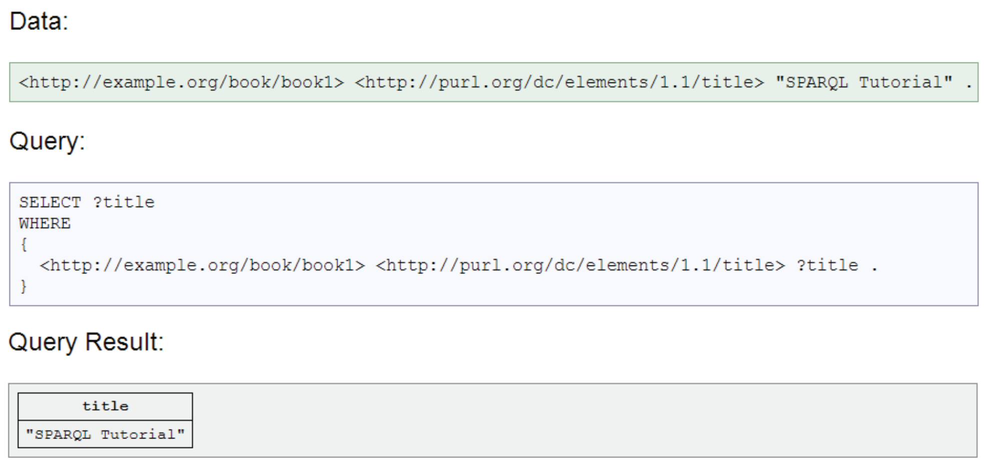
The main benefits of using the RDF model in Linked Data are to facilitate data merging and to support the evolution of schemas over time without requiring all the data consumers to change (Manola and Miller 2013). Because RDF is not a data format, rather an abstract model for representing structured data as triples, RDF triples must be serialized using one of the following RDF syntaxes:

[11] http://www.openarchives.org/ore/1.0/primer

RDF/XML, RDFa, Turtle, N-Triples, or RDF/JSON (Heath and Christian Bizer 2011).

2.5 SPARQL Query Language

The SPARQL Query Language was designed to query RDF datasets, that is, collections of RDF graphs (Prud'hommeaux and Seaborne 2008). SPARQL is based on graph patterns in the sense that a SPARQL query is a combination of graph patterns, including their conjunctions and disjunctions. Figure 5 presents an example of a simple SPARQL query, which was designed to find the title of a book from the given data graph with a simple graph pattern (Prud'hommeaux and Seaborne 2008).



Data:

```
<http://example.org/book/book1> <http://purl.org/dc/elements/1.1/title> "SPARQL Tutorial" .
```

Query:

```
SELECT ?title
WHERE
{
  <http://example.org/book/book1> <http://purl.org/dc/elements/1.1/title> ?title .
}
```

Query Result:

title
"SPARQL Tutorial"

Figure 5: A simple SPARQL query (Prud'hommeaux and Seaborne 2008)

Figure 5 contains the triplestore to be accessed, the SPARQL query itself and the query result. The first part of the query, the `SELECT` clause, identifies the variables that will appear in the result (in this example, `?title`). The `WHERE` clause contains a graph pattern, which is matched with the data graph. The pattern in this example is a single triple, in which the object position contains the variable of the `SELECT` clause, `?title`.

The pattern subject is the resource identified by the URI `<http://example.org/book/book1>` and the pattern predicate is the resource representing the relation, identified by the URI `<http://purl.org/dc/elements/1.1/title>`. Thus, only the triples that have this subject and this predicate will appear in the results (Prud'hommeaux and Seaborne 2008).

2.6 Data Cubes

2.6.1. OLAP Data Cube

Business intelligence (BI) consists of a set of techniques 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 multidimensional databases known as data warehouses (DW). These tools are known as On-Line Analytical Processing (*OLAP*) (Etcheverry and Vaisman 2012a).

In OLAP, data are perceived as multidimensional structures known as *data cubes*, which can be understood as a star schema view of the relational database. Such structures are organized according to their components: dimensions, attributes and measures. Each data cube has one (or more) specific *measure* corresponding to the observed phenomenon, such as life expectancy. Each measure also has an attribute to define the units, for instance years.

A data cube also contains contextual information called *dimensions*. For instance, assume that a cube has three dimensions - time period, region and gender. These dimensions can be structured in hierarchies of levels that enable the analysis of the data at different levels of aggregation (Etcheverry and Vaisman 2012a).

The advantage of using such structures is the ability to obtain different perspectives on the data. One can also transform the cubes through OLAP operations, such as: *slice*, which removes a dimension from the cube and aggregates over its members; *dice*, which enables a filter for certain dimension members and aggregates over them; and *roll-up*, which creates a cube that contains instance data on a higher aggregation level (Kampgen et al. 2012).

2.6.2. RDF Data Cube Vocabulary

The RDF Data Cube vocabulary[12] was specifically designed to publish data cubes on the Web in such a way that they can be linked to related RDF datasets. It is basically a simplified version of the SDMX Information Model (Statistical Data and Metadata eXchange), an ISO standard for exchanging and sharing statistical data and metadata among organizations (Richard Cyganiak et al. 2010).

This vocabulary can be extended to other vocabularies that help publish additional context of statistical data, such as SKOS, SCOVO, VoID, FOAF and Dublin Core. The RDF Data Cube Vocabulary also enables to represent the result of data cube slice operations (Tennison 2012).

Figure 6 shows the classes and properties of the RDF Data Cube vocabulary. Basically, to encode structural information about the observations, the RDF Data Cube vocabulary contains a set of concepts, such as `qb:DataStructureDefinition`, `qb:DataSet` and `qb:Observation`. The `qb:DataStructureDefinition` defines the structure of one or more datasets, which includes the dimensions, attributes and measures. The `qb:DataSet` represents a collections of observations that may or not may be organized into slices. The `qb:Observation` is a single observation in the cube (Sun 2009).

The data cube dimensions, attributes, and measures are represented 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`.

[12] http://purl.org/linked-data/cube#

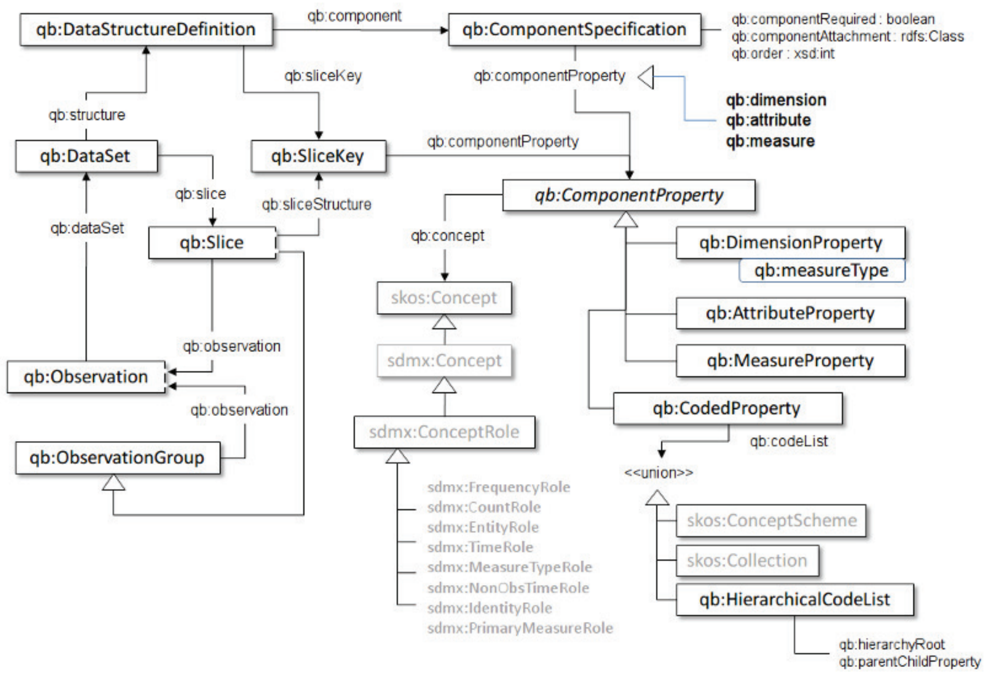
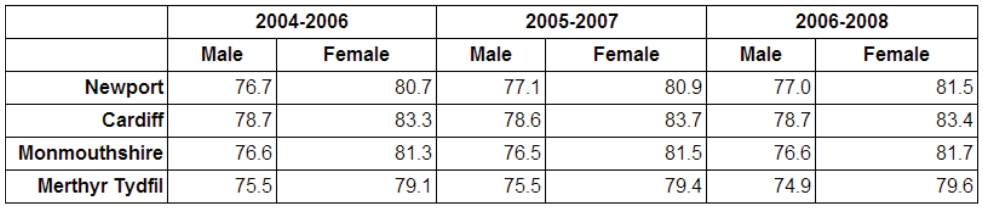


Figure 6: The RDF Data Cube Vocabulary key terms

To illustrate the use of the RDF Data Cube Vocabulary, we can consider a small data set that describes life expectancy broken down by region, age and time, depicted in Figure 7.



	2004-2006		2005-2007		2006-2008	
	Male	Female	Male	Female	Male	Female
Newport	76.7	80.7	77.1	80.9	77.0	81.5
Cardiff	78.7	83.3	78.6	83.7	78.7	83.4
Monmouthshire	76.6	81.3	76.5	81.5	76.6	81.7
Merthyr Tydfil	75.5	79.1	75.5	79.4	74.9	79.6

Figure 7: A Data cube example (Sun 2009)

This cube has three dimensions - time period, region and gender. It contains 24 observations (4 values of region x 2 values of gender x 3 values of time period). Each observation represents the life expectancy for that population with regards to the three dimensions.

Each data cube has a specific measure corresponding to the phenomenon being observed, in this example, life expectancy. Each measure also has an attribute to define the units, in this example, years (Tennison 2012).

To represent a set of common statistical concepts that is intended to be reused across data sets, the SDMX standard includes a set of *content oriented* *guidelines*, the SDMX-COG[13]. These concepts are recommended to be used when representing data cubes dimension, measures and attributes by the RDF Data Cube Vocabulary.

```
eg:refPeriod  a rdf:Property, qb:DimensionProperty;
    rdfs:label "reference period"@en;
    rdfs:subPropertyOf sdmx-dimension:refPeriod;
    rdfs:range interval:Interval;
    qb:concept sdmx-concept:refPeriod .

eg:refArea  a rdf:Property, qb:DimensionProperty;
    rdfs:label "reference area"@en;
    rdfs:subPropertyOf sdmx-dimension:refArea;
    rdfs:range admingeo:UnitaryAuthority;
    qb:concept sdmx-concept:refArea .
```

Figure 8: Dimension properties (Sun 2009)

Figure 8 presents two dimensions of the aforementioned data cube, *Time* and *Region*, described using the SDMX-COG concepts.

The Time dimension is described by the component property `sdmx-dimension:refPeriod`. The `qb:concept` predicate has the object `sdmx-concept:refPeriod`, the REF_PERIOD concept of the SDMX-COG (Richard Cyganiak et al. 2010).

The same idea is applicable to the Region dimension. Again, there is a suitable SDMX-COG concept (`sdmx-concept:refArea`) and component (`sdmx-dimension:refArea`) that can be used to represent this dimension. The *admingeo* vocabulary[14] was also used to customize the range of the component (Sun 2009).

The *measure property* of the data cube vocabulary gives a value to each observation. Figure 9 presents the measure life expectancy using the `sdmx-measure:obsvalue` property.

```
eg:lifeExpectancy  a rdf:Property, qb:MeasureProperty;
    rdfs:label "life expectancy"@en;
    rdfs:subPropertyOf sdmx-measure:obsValue;
    rdfs:range xsd:decimal .
```

Figure 9: A measure property (Sun 2009)

With this representation of dimensions and measures using the RDF Data Cube Vocabulary, one can create a data cube description. A `qb:DataSet` resource

[13] http://sdmx.org/
[14] http://data.ordnancesurvey.co.uk/ontology/admingeo/

represents the entire dataset, a collection of statistical data and it is linked with its structure by the `qb:structure` property, presented in Figure 10.

```
eg:dataset-le1 a qb:DataSet;
    rdfs:label "Life expectancy"@en;
    rdfs:comment "Life expectancy within Welsh Unitary authorities
    qb:structure eg:dsd-le ;
    .
```

Figure 10: A data cube description (the dataset definition part) (Sun 2009)

The structure of each dataset is defined by a `qb:DataStructureDefinition` resource, which combines its components - dimensions, measure(s) and attributes - into a specification. Figure 11 shows a description of a data cube, including all its components: dimensions, measure(s) and attributes. The advantage of separating the data cube description into a dataset part and its structure is that the structure is reusable across other datasets which share the same components (Sun 2009).

```
eg:dsd-le a qb:DataStructureDefinition;
    # The dimensions
    qb:component [qb:dimension eg:refArea;         qb:order 1];
    qb:component [qb:dimension eg:refPeriod;       qb:order 2];
    qb:component [qb:dimension sdmx-dimension:sex; qb:order 3];
    # The measure(s)
    qb:component [qb:measure eg:lifeExpectancy];
    # The attributes
    qb:component [qb:attribute sdmx-attribute:unitMeasure;
                  qb:componentRequired "true"^^xsd:boolean;
                  qb:componentAttachment qb:DataSet;] .
```

Figure 11: A data cube description (the data structure part) (Sun 2009)

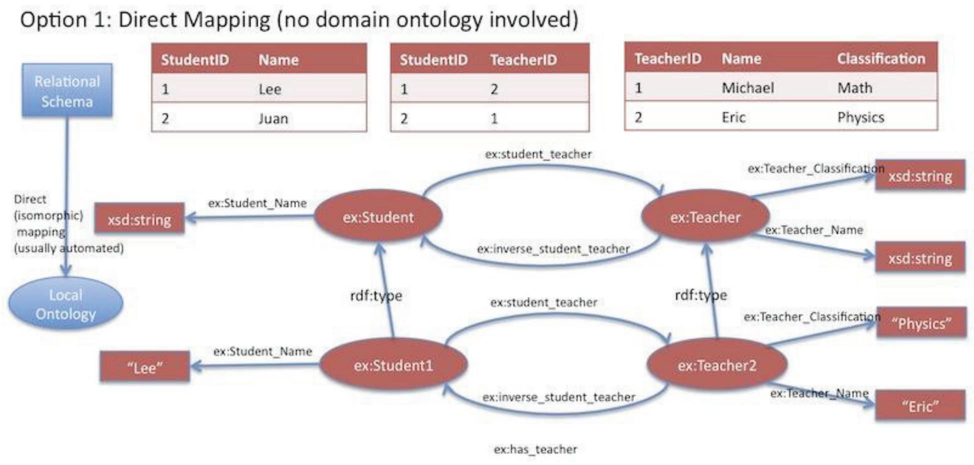
2.7 RDF to RDB approaches

The mission of the RDB2RDF Working Group[15] is to standardize languages for mapping relational data and relational database schemas into RDF and OWL. They proposed two approaches: Direct Mapping (DM) and the RDB2RDF Mapping Language (R2RML).

[15] http://www.w3.org/2001/sw/rdb2rdf/

2.7.1. Direct Mapping

In the direct mapping from relational databases to RDF, the RDF graph generated directly reflects the database structure. In other words, neither structure nor target vocabulary change. Figure 12 shows an example of direct mapping when no domain ontology is involved.



StudentID	Name
1	Lee
2	Juan

StudentID	TeacherID
1	2
2	1

TeacherID	Name	Classification
1	Michael	Math
2	Eric	Physics

Figure 12: Direct Mapping example[16]

2.7.2. R2RML Vocabulary

R2RML (Das et al. 2013) is a W3C Recommendation since September 2012, used for expressing customized mappings from relational databases to RDF datasets created by the RDB2RDF working group (Herman et al. 2012). 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 13 presents the main classes and properties of the vocabulary.

---

[16] http://www.w3.org/TR/rdb2rdf-ucr/

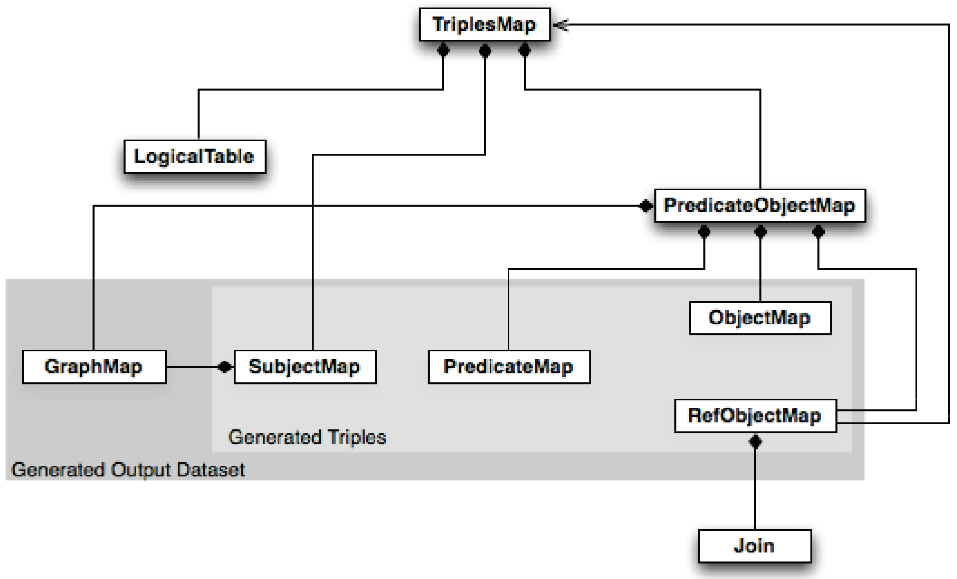


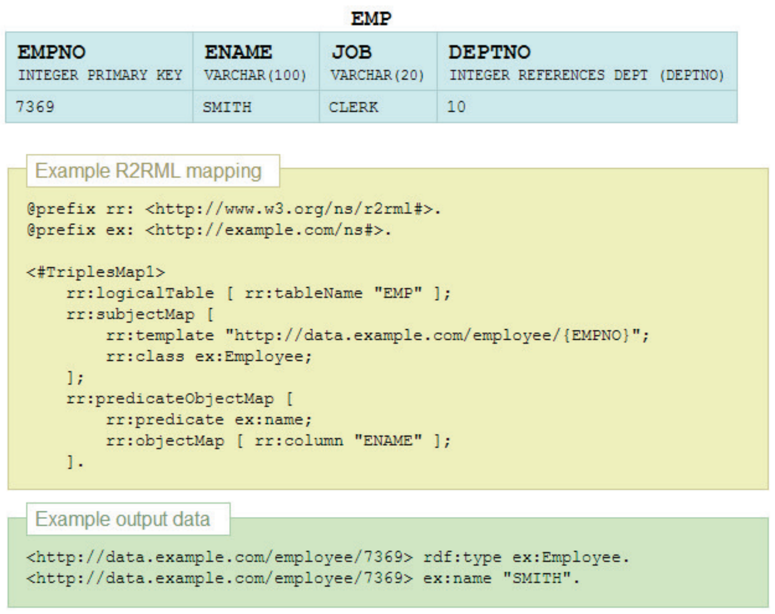
Figure 13: Overview of the R2RML Vocabulary[17]

Briefly, a *triple map* is a rule that maps each tuple in the logical table to RDF triples. It is composed of two main parts: the *subject map* and the *predicate-object maps*. The subject map generates the subject of all RDF triples that will be generated from each logical table row. The multiple predicate-object maps contain *predicate maps* and *object maps* (or *referencing object maps*).

Figure 14 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.

[17] http://www.w3.org/TR/r2rml/



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 14: An R2RML mapping example (Das et al. 2013)

2.8 The Web Service Architecture

According to W3C, a *Web service* is, “a software system designed to support interoperable machine-to-machine interaction over a network”. Its interface must be described in a machine-processable format (Booth et al. 2004).

The Web Service Architecture, defined by the W3C Web Services Architecture Working Group, is an interoperability architecture: it identifies the elements of the Web services network required to ensure interoperability between them (Booth et al. 2004). The Web Service Architecture uses several technologies that are interrelated, but divided into layers, shown below in Figure 15:

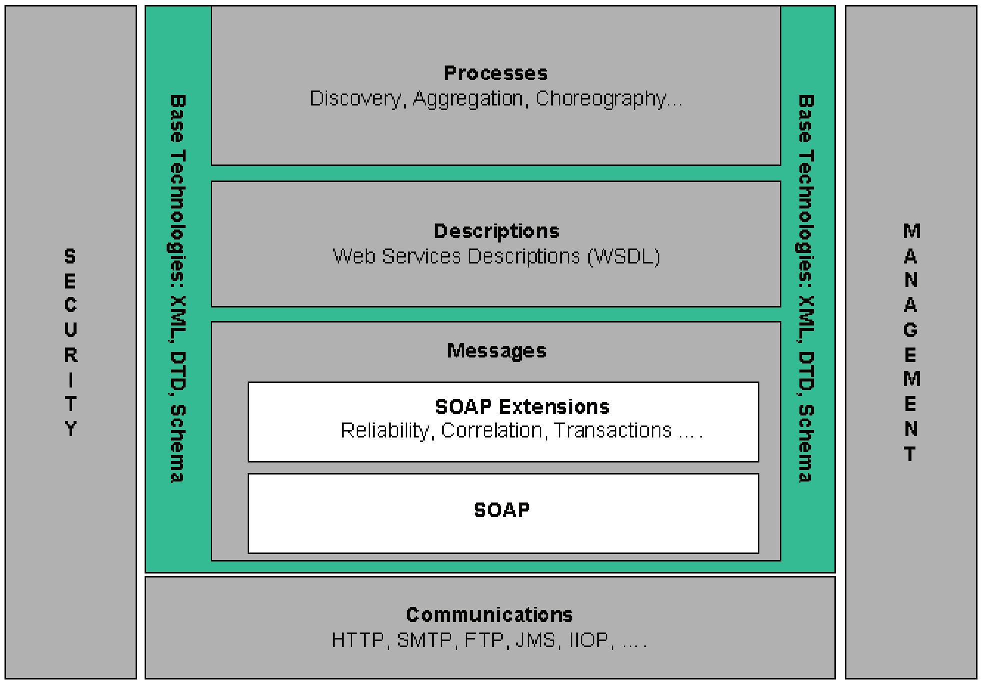
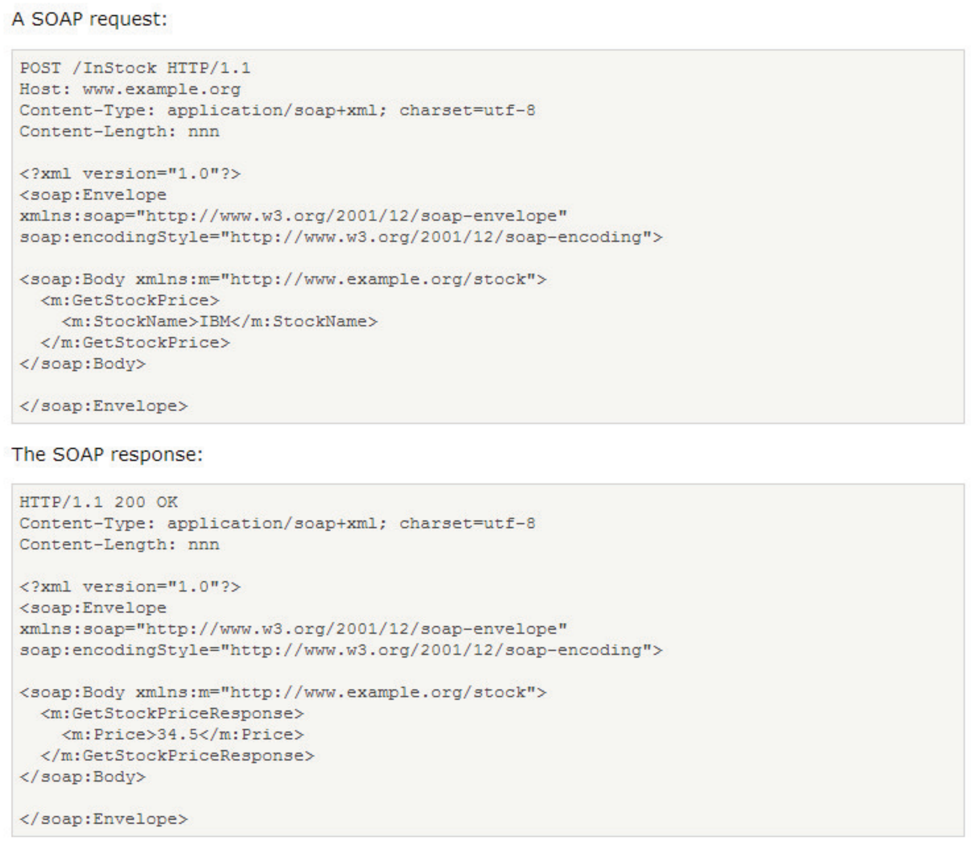

S E C U R I T Y
Base Technologies: XML, DTD, Schema
Processes
Discovery, Aggregation, Choreography...
Descriptions
Web Services Descriptions (WSDL)
Messages
SOAP Extensions
Reliability, Correlation, Transactions ... .
SOAP
Base Technologies: XML, DTD, Schema
M A N A G E M E N T
Communications
HTTP, SMTP, FTP, JMS, IIOP, ...

Figure 15: Web Services Architecture Stack (Booth et al. 2004)

2.8.1. XML/SOAP/WSDL

The main technologies of the Web Service Architecture are: XML, SOAP and WSDL. Extensible Markup Language (XML) offers a standard, flexible and inherently extensible data format (Quin 2004).

Simple Object Access Protocol (SOAP) is a W3C recommendation for exchanging structured information in a decentralized and distributed environment (Gudgin et al. 2007). It is based on XML and allows for exchanging messages over a variety of underlying protocols. Figure 16 offers an example of a simple SOAP message that contains a SOAP request and a SOAP response.


A SOAP request:

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPrice>
    <m:StockName>IBM</m:StockName>
  </m:GetStockPrice>
</soap:Body>

</soap:Envelope>
```

The SOAP response:

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPriceResponse>
    <m:Price>34.5</m:Price>
  </m:GetStockPriceResponse>
</soap:Body>

</soap:Envelope>
```

Figure 16: A SOAP request and response example (w3schools 2012)

The request in Figure 16 has a `StockName` parameter and a `Price` parameter, which are returned in the response message. Both messages are enveloped by the SOAP protocol.

Web Service Description Language (WSDL) is a language based on XML and is used for describing network services such as collections of communication endpoints capable of exchanging messages (Christensen et al. 2007). Figure 17 presents an overview of the WSDL classes and properties.

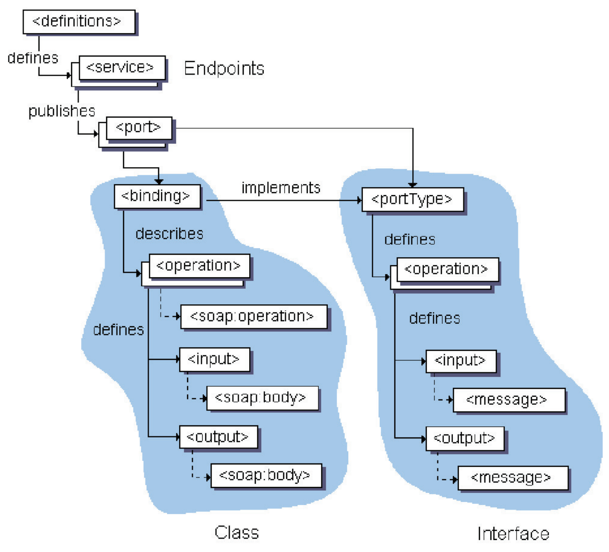


Figure 17: A WSDL structure[18]

According to the Web Service Architecture specification, a Web service must use XML to tag data, the SOAP protocol to transfer its messages, and WSDL to describe how applications can access Web services.

2.8.2.
REST

2.8.2.1. REST architecture

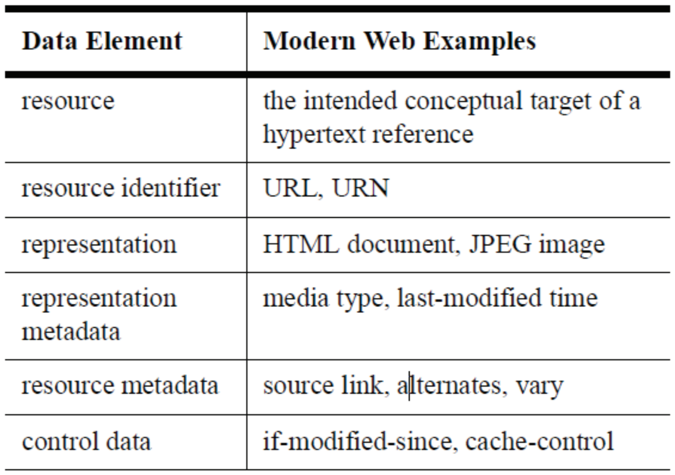
The Representational State Transfer (REST) style is an abstraction of the architectural elements within a distributed hypermedia system. It was conceived by Roy Fielding, one of the authors of the HTTP specification, in his dissertation in 2000 (Fielding and Taylor 2002).

The definition of the REST architecture was inspired by three classes of architectural elements defined by Perry and Wolf: *processing elements* or components, *data elements*, and connecting elements or *connectors* (Perry and Wolf 1992).

REST connectors offer an interface for component communication and provide a separation of concerns, while hiding the underlying implementation of resources. The main REST connectors are: the *client*, the connector that initiates the communication by making a request; the *server*, which listens for connections

18 http://download.oracle.com/otn_hosted_doc/jdeveloper/1012/web_services/ws_wsdlstructure.html

and responds to requests to supply access to its services; and the *cache* connector, which saves cacheable responses to be reused for other requests (Fielding and Taylor 2002). The data elements of the REST architecture are described in Figure 18 and explained hereafter.



Data Element	Modern Web Examples
resource	the intended conceptual target of a hypertext reference
resource identifier	URL, URN
representation	HTML document, JPEG image
representation metadata	media type, last-modified time
resource metadata	source link, alternates, vary
control data	if-modified-since, cache-control

Figure 18: REST Data Elements (Fielding and Taylor 2002)

As in the architecture of the Web described in Section 2.1, the key abstraction of information in REST is the *resource*, which is defined as any information that can be identified as a document, an image, or a weather service. REST also uses a *resource identifier* to identify a specific resource involved in a message exchange. (Fielding and Taylor 2002).

Each resource has a *representation* that captures the resource's current or intended state. An HTML document and a JPG image are examples of such a representation. It is important to emphasize that resources can have multiple representations. When this is the case, the HTML content negotiation feature is used to select the proper representation (Trick 2007).

Both representation and resource have metadata to describe themselves. An example of the *representation metadata* is the date of the last representation update. The *resource metadata* consists of information about the resource that is not specific to the representation requested (Fielding and Taylor 2002). Lastly, the *control data* is related to the purpose of the message. Depending on the control data, a representation may indicate the current state of the requested resource.

REST interactions are always *stateless*, i.e., each message is self-contained. In other words, each request contains all the information needed for any connector to process the request. Thus, the server does not need to keep any information related to any individual request (Trick 2007).

Another important REST feature is the *layered system*. REST interactions are structured in a layered client-server style, such that a component cannot interact directly with any other component, restricting the knowledge of the system to a single layer (Sun 2009).

2.8.2.2. RESTful Web services

In the traditional Web Service approach (SOAP/WSDL), a service publish an endpoint that exposes the set of available operations, which have particular semantics that must be known in advance, in order to be properly requested by the client (Alarcon and Wilde 2010).

An application that follows the REST principles is called *RESTful*. RESTful Web services have no endpoint. Instead, they are comprised of a set of resource URIs and a set of standard methods that enable access and change the state of the resources, which reduce the interaction latency between clients and servers and simplify the overall system architecture (Sun 2009).

Currently, due to its simplicity, REST has inspired the design of several services on the Web to serve as an alternative approach to using the SOAP protocol. They are used for science, reference, and medical research APIs, including Dropbox REST API[19], Mendeley REST API[20], and Twitter REST API[21].

2.9 Summary

This chapter presented the main concepts related to this dissertation. Section 2.1 presented the architecture of the Wide World Web. Section 2.2 summarized the HTTP Protocol and its main features. Section 2.3 introduced the Linked Data Principles. Section 2.4 presented the RDF model, the building block of the Semantic Web. Section 2.5 introduced the SPARQL Query Language, created to

[19] https://www.dropbox.com/developers/core/docs
[20] http://apidocs.mendeley.com/
[21] https://dev.twitter.com/docs/api

query the RDF graphs. Section 2.6 showed the data cubes, its components – dimensions, measure and attributes - and presented the RDF vocabulary built to represent data cubes as RDF. Section 2.7 presented the approaches to convert relational databases into RDF models. Section 2.8 introduced the Web Service Architecture - the traditional Web service approach - and the REST approach to consuming Web services.