2
Data Cube Description in RDF

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 were introduced by Tim Berners-Lee (Berners-Lee 2009) and enable all the 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.

These principles are known as the *Linked Data principles* and provide a basic recipe for publishing and connecting data using the infrastructure of the Web while adhering to its architecture and standards (Bizer et al. 2009b).

Linked Data has been adopted by an increasing numbers of data providers and application developers, leading to the creation of this global, interconnected data space known as *the Web of Data* (Bizer et al. 2009b). This *Web of Data*, which is part of the *Semantic Web* (Berners-Lee et al. 2001), presents a revolutionary opportunity for deriving insight and value from data (Heath & Bizer 2011).

A standard mechanism is required to link data across the Web, for specifying the existence and meaning of connections between items described in this data. This mechanism is provided by the Resource Description Framework (RDF). RDF provides a flexible way to describe things in the world – such as people, locations, or abstract concepts – and how they relate to other things. These statements of relationships between things are, in essence, links connecting things in the world. RDF enables the data publisher to state explicitly the nature of the connection. Rather than simply connecting documents, as it happens with in the hypertext Web with HTML (HyperText Markup Language) documents connected by untyped hyperlinks, Linked Data uses RDF to make typed statements that link arbitrary things in the world (Heath & Bizer 2011), (Bizer et al. 2009b). Therefore, a Web in which data is both published and linked using RDF is a Web where data is significantly more discoverable, and therefore more usable. Just as hyperlinks in the classic Web connect documents into a single global information space, Linked Data enables links to be set between items in different data sources and therefore connect these sources into a single global data space. Linked Data builds directly on Web architecture and applies this architecture to the task of sharing data on global scale (Heath & Bizer 2011).

Technically, Linked Data refers to data published on the Web in such a way that it is machine-readable, its meaning is explicitly defined, it is linked to other external data sets, and can in turn be linked to from external data sets (Bizer et al. 2009b).

The use of Web standards and a common data model enables the Web to transcend different technical architectures and operate over the complete data space. This is the essence of *Linked Data* (Heath & Bizer 2011).

In summary, the Linked Data provides a more generic, more flexible publishing paradigm which makes it easier for data consumers to discover and integrate data from large numbers of data sources. In particular, Linked Data provides: a unifying data model, a standardized data access mechanism, hyperlink-based data discovery and self-descriptive data (Heath & Bizer 2011).

2.2. RDF

The Resource Description Framework (RDF) is a language for representing information about resources in the World Wide Web (Manola et al. 2004). RDF provides a simple graph-based data model that has been designed for the integrated representation of information originated from multiple sources in the context of the Web.

In RDF, a description of a resource is represented as a number of *triples*. An RDF triple is composed by three components: *subject*, *predicate* and *object*. The subject is URI identifying the described resource or a blank node. The predicate is an URI expressing the relationship between subject and object. The object is an URI of another resource that is somehow related with the subject, a literal or a blank node.

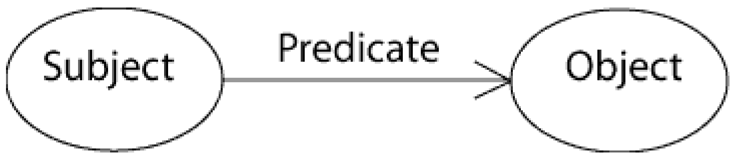

Figure 2 - Representation of an RDF Triple (Klyne et al. 2004)

A set of such triples is called an RDF graph. The nodes of an RDF graph are its subjects and objects and the directed arcs connecting them are the predicates that express the relationship between them. The direction of the arc is significant: it always points toward the object (Klyne et al. 2004).

Heath and Bizer (2011) mention that two principal types of RDF triples can be distinguished: *Literal Triples* and *RDF Links*. *Literal Triples* have an RDF literal such as a string, number, or date as the object. This kind of triples is used to describe the properties of resources. On the other hand, *RDF Links* describe the relationship between two resources. RDF links consist of three URI references. The URIs in the subject and the object position of the link identify the related resources. The URI in the predicate position defines the type of relationship between the resources. The RDF links can be divided in *internal* and *external. Internal RDF links* connect resources within a single Linked Data source. *External RDF links* connect resources that are served by different Linked Data sources. External RDF links are crucial for the Web of Data as they are they allow to navigate between data sources and discover additional data, creating a global, interconnected data space. Thus, Linked Data can be seen as one *giant global graph*, as proposed by Tim Berners-Lee in (Berners-Lee 2007).

According to Heath and Bizer (2011), the main benefits of using the RDF data model in a Linked Data context are that:

- By using HTTP URIs as globally unique identifiers for data items as well as for vocabulary terms, the RDF data model is inherently designed for being used at global scale and enables anybody to refer to anything.

- Clients can look up any URI in an RDF graph over theWeb to retrieve additional information. Thus each RDF triple is part of the globalWeb of Data and each RDF triple can be used as a starting point to explore this data space.

- The data model enables you to set RDF links between data from different sources.

- Information from different sources can easily be combined by merging the two sets of triples into a single graph.

- RDF allows you to represent information that is expressed using different schemata in a single graph, meaning that you can mix terms for different vocabularies to represent data.

- Combined with schema languages such as RDF-Schema (Brickley et al. 2004) and OWL (McGuinness & Harmelen 2004), the data model allows the use of as much or as little structure as desired, meaning that tightly structured data as well as semi-structured data can be represented.

To conclude, it is important to remark that RDF is not a data format but a data model for describing resources. Thus, to publish an RDF graph on the Web, it must first be serialized using a RDF syntax. There are several RDF serialization formats, among which we may quote: RDF/XML (Beckett & McBride 2004), Turtle (Beckett et al. 2013), N-triples (Grant et al. 2004), N3 (Berners-Lee & Connolly 2011).

2.3. Formal Definition of Data Cubes

According to Cyganiak et al. (2013), 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. An example of a data cube is given in Section 2.7, including the notions of dimension, measure and attribute.

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 of the form:

$$R \subseteq D_1 \times \ldots \times D_m \times A_1 \times \ldots \times A_n \times \mathrm{O}$$

commonly referred to as a *data cube* or simply as a *cube* (without meaning there are exactly 3 dimensions, as it can be seen in the relation presented above). Actually, a statistical data set is a multi-dimensional space, or hyper-cube, indexed by those dimensions.

Many data cubes include time as a dimension. Any such cube is known as a *time series*.

A tuple of values from the dimension domains identifies a single observation measure value and the associated attribute values. Hence, *R* is actually a function of the form:

$$R: D_1 \times \ldots \times D_m \rightarrow A_1 \times \ldots \times A_n \times O$$

The cubes are commonly represented in a relational database (RDB) as a set of tables organized in a *star schema* or in a *snowflake schema* (Salas et al. 2012). A star schema is a combination of one or more fact tables with its dimension tables; the fact table is at the center of the star while each dimension table referenced by the fact table represents a point of the star. Snowflake schemas are more complex, as dimension tables are normalized into multiple, related tables (Salas et al. 2012).

For both schemas, fact tables have key fields connecting the fact table to a dimension table for each of the dimensions. In turn, dimension tables have key fields joining the dimension tables to the fact table (and to other dimension tables when snow flaking) (Upadhyaya & Singh 2011).

2.4. Data Cube Vocabulary

There is strong interest to publish multi-dimensional data, such as statistics, on the Web in such a way that it can be linked to related data sets and concepts. The Data Cube vocabulary (Cyganiak et al. 2013) enables such information to be represented on the Web using the W3C RDF standard and published following the principles of linked data.

The model that underlies the Data Cube vocabulary is compatible with the cube model that is supported by SDMX (Statistical Data and Metadata eXchange) (Sdmx 2009), an ISO standard for exchanging and sharing statistical data and metadata (Cyganiak et al. 2013).

The Data Cube vocabulary is based on the existing RDF vocabularies:

- *Simple Knowledge Organization System* (SKOS) (Miles & Bechhofer 2009) – for concept schemes
- *Statistical Core Vocabulary* (SCOVO) (Hausenblas et al. 2012) - for core statistical structures
- *Dublin Core Metadata Initiative (DCMI) MetadataTerms* (DCMI 2012) – for metadata
- *Vocabulary of Interlinked Datasets* (VoID) (Alexander et al. 2011) - for data access
- *Friend-of-a-Friend* (FOAF) (Brickley & Miller 2010) – for agents
- *Core Organization Ontology* (ORG) (Reynolds 2012) – for organizations

The Data Cube vocabulary is a core foundation which supports extension vocabularies to enable publication of other aspects of statistical data flows or other multi-dimensional data sets (Cyganiak et al. 2013).

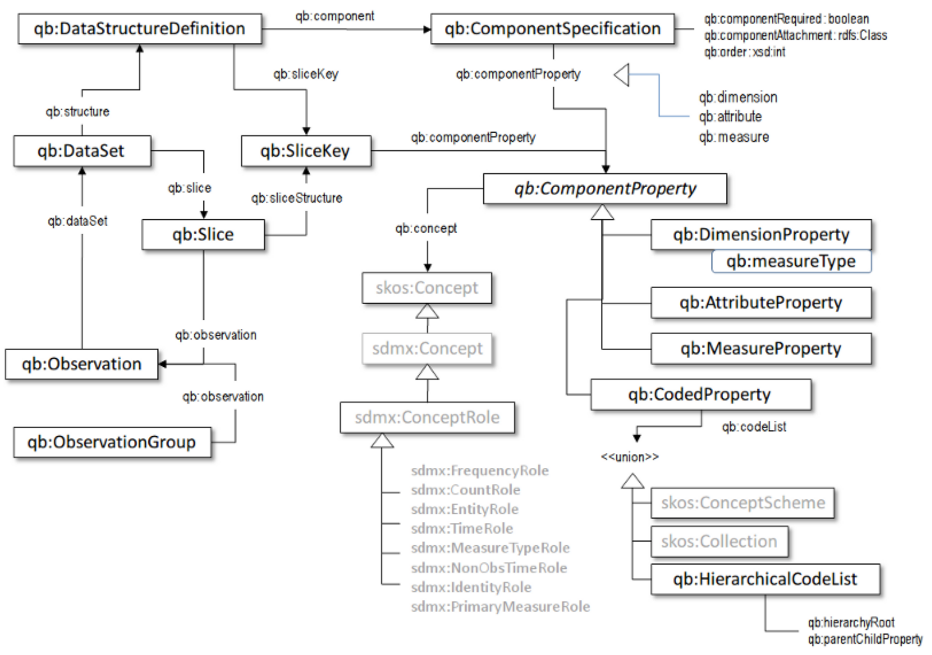

Figure 3 - Outline of the Data Cube Vocabulary (Cyganiak et al. 2013)

The names of RDF entities - classes, predicates, individuals - are URIs. Usually these URIs are represented using a compact notation, just using prefixes to identify the namespaces.

The prefixes and its correspondent namespaces used in Figure 3 are shown in Table 1.

PREFIX	NAMESPACE
qb	http://purl.org/linked-data/cube#
skos	http://www.w3.org/2004/02/skos/core#
rdfs	http://www.w3.org/2000/01/rdf-schema#
xsd	http://www.w3.org/2001/XMLSchema#
sdmx	http://purl.org/linked-data/sdmx#

Table 1 – Prefixes and Namespaces

To better understand Figure 3, the RDF Data Cube vocabulary is summarized in what follows (Cyganiak et al. 2013).

1 DataSets

Class: qb:DataSet Sub class of: qb:Attachable *Equivalent to:* scovo:Dataset

Represents a collection of observations, possibly organized into various slices, conforming to some common dimensional structure.

2 Observations

Class: qb:Observation Sub class of: qb:Attachable *Equivalent to:* scovo:Item

A single observation in the cube, may have one or more associated measured values.

Property: qb:dataSet (*Domain:* qb:Observation -> *Range:* qb:DataSet)

Indicates the data set of which this observation is a part.

Property: qb:observation (*Domain:* qb:Slice -> *Range:* qb:Observation)

Indicates a observation contained within this slice of the data set.

3 Slices

Class: qb:ObservationGroup

A, possibly arbitrary, group of observations.

Class: qb:Slice Sub class of: qb:Attachable, qb:ObservationGroup

Denotes a subset of a DataSet defined by fixing a subset of the dimensional values, component properties on the Slice.

Property: qb:slice (*Domain:* qb:DataSet -> *Range:* qb:Slice; *sub property of:* qb:observationGroup)

Indicates a subset of a DataSet defined by fixing a subset of the dimensional values.

Property: qb:observationGroup (*Domain:* -> *Range:* qb:ObservationGroup)

Indicates a group of observations. The domain of this property is left open so that a group may be attached to different resources and need not be restricted to a single DataSet.

4 Dimensions, Attributes, Measures

Class: qb:Attachable

Abstract superclass for everything that can have attributes and dimensions.

Class: qb:ComponentProperty Sub class of: rdf:Property

Abstract super-class of all properties representing dimensions, attributes or measures.

Class: qb:DimensionProperty Sub class of: qb:ComponentProperty, qb:CodedProperty

The class of component properties which represent the dimensions of the cube.

Class: qb:AttributeProperty Sub class of: qb:ComponentProperty

The class of component properties which represent attributes of observations in the cube, e.g. unit of measurement.

Class: qb:MeasureProperty Sub class of: qb:ComponentProperty

The class of component properties which represent the measured value of the phenomenon being observed.

Class: qb:CodedProperty Sub class of: qb:ComponentProperty

Superclass of all coded component properties.

5 Reusable general purpose component properties

Property: qb:measureType (*Domain:* -> *Range:* qb:MeasureProperty)

Generic measure dimension, the value of this dimension indicates which measure (from the set of measures in the DSD) is being given by the observation.

6 Data Structure Definitions

Class: qb:DataStructureDefinition Sub class of: qb:ComponentSet

Defines the structure of a DataSet or slice.

Property: qb:structure (*Domain:* qb:DataSet -
> *Range:* qb:DataStructureDefinition)

Indicates the structure to which this data set conforms.

Property: qb:component (*Domain:* qb:DataStructureDefinition -> *Range:* qb:ComponentSpecification)

Indicates a component specification which is included in the structure of the dataset.

7 Component specifications - for qualifying component use in a DSD

Class: qb:ComponentSpecification Sub class of: qb:ComponentSet

Used to define properties of a component (attribute, dimension etc) which are specific to its usage in a DSD.

Class: qb:ComponentSet

Abstract class of things which reference one or more ComponentProperties

Property: qb:componentProperty (*Domain:* qb:ComponentSet -
> *Range:* qb:ComponentProperty)

Indicates a ComponentProperty (i.e. attribute/dimension) expected on a DataSet, or a dimension fixed in a SliceKey.

Property: qb:order (*Domain:* qb:ComponentSpecification -> *Range:* xsd:int)

Indicates a priority order for the components of sets with this structure, used to guide presentations - lower order numbers come before higher numbers, un-numbered components come last.

Property: qb:componentRequired (*Domain:* qb:ComponentSpecification -> *Range:* xsd:boolean)

Indicates whether a component property is required (true) or optional (false) in the context of a DSD. Only applicable to components corresponding to an attribute. Defaults to false (optional).

Property: qb:componentAttachment (*Domain:* qb:ComponentSpecification -> *Range:* rdfs:Class)

Indicates the level at which the component property should be attached, this might be an qb:DataSet, qb:Slice or qb:Observation, or a qb:MeasureProperty.

Property: qb:dimension (*Domain: -> Range:* qb:DimensionProperty ; *sub property of:* qb:componentProperty)

An alternative to qb:componentProperty which makes explicit that the component is a dimension.

Property: qb:measure (*Domain: -> Range:* qb:MeasureProperty ; *sub property of:* qb:componentProperty)

An alternative to qb:componentProperty which makes explicit that the component is a measure.

Property: qb:attribute (*Domain: -> Range:* qb:AttributeProperty ; *sub property of:* qb:componentProperty)

An alternative to qb:componentProperty which makes explicit that the component is a attribute.

Property: qb:measureDimension (*Domain: -> Range:* qb:DimensionProperty ; *sub property of:* qb:componentProperty)

An alternative to qb:componentProperty which makes explicit that the component is a measure dimension.

8 Slice definitions

Class: qb:SliceKey Sub class of: qb:ComponentSet

Denotes a subset of the component properties of a DataSet which are fixed in the corresponding slices.

Property: qb:sliceStructure (*Domain:* qb:Slice -> *Range:* qb:SliceKey)

Indicates the slice key corresponding to this slice.

Property: qb:sliceKey (*Domain:* qb:DataSet -> *Range:* qb:SliceKey)

Indicates a slice key which is used for slices in this dataset.

9 Concepts

Property: qb:concept (*Domain:* qb:ComponentProperty -
> *Range:* skos:Concept)

Gives the concept which is being measured or indicated by a ComponentProperty.

Property: qb:codeList (*Domain:* qb:CodedProperty -
> *Range:* owl:unionOf(skos:ConceptScheme skos:Collection qb:HierarchicalCodeList))

Gives the code list associated with a CodedProperty.

10 Non-SKOS Hierarchies

Class: qb:HierarchicalCodeList

Represents a generalized hierarchy of concepts which can be used for coding. The hierarchy is defined by one or more roots together with a property which relates concepts in the hierarchy to their child concept . The same concepts may be members of multiple hierarchies provided that different qb:parentChildProperty values are used for each hierarchy.

Property: qb:hierarchyRoot (*Domain:* qb:HierarchicalCodeList)

Specifies a root of the hierarchy. A hierarchy may have multiple roots but must have at least one.

Property: qb:parentChildProperty (*Domain:* qb:HierarchicalCodeList -
> *Range:* rdf:Property)

Specifies a property which relates a parent concept in the hierarchy to a child concept. Note that a child may have more than one parent.

2.5. R2RML

R2RML is a mapping language from relational databases to RDF (Das et al. 2010) as it expresses customized mappings from relational databases to RDF datasets. With a direct mapping (Arenas et al. 2012) of a database, the structure of the resulting RDF graph directly reflects the structure of the database, the target RDF vocabulary directly reflects the names of database schema elements, and neither structure nor target vocabulary can be changed. R2RML, in turn, provides the ability to view existing relational data in the RDF data model, expressed in a structure and target vocabulary of the mapping author's choice; a mapping author can define highly customized views over the relational data.

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 row in a logical table to a set of RDF triples. The rule has 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 map and object map, and applying these three maps to each logical table row.

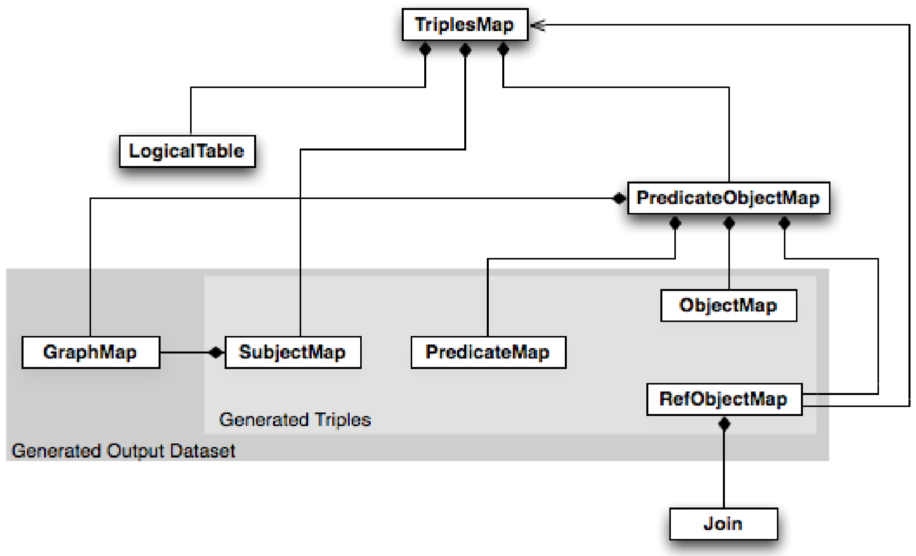

Figure 4 - An overview of R2RML

R2RML has the namespace http://www.w3.org/ns/r2rml# and it has the namespace prefix rr. The R2RML vocabulary has the following classes:

- rr:TriplesMap is the class of triples map.
- rr:LogicalTable is the class of logical tables. It has two subclasses:
    - rr:R2RMLView is the class of R2RML views.
- rr:BaseTableOrView is the class of SQL base tables or views.
- rr:TermMap is the class of term maps. It has four subclasses:
    - rr:SubjectMap is the class of subject maps.
    - rr:PredicateMap is the class of predicate maps.
    - rr:ObjectMap is the class of objecy maps.
    - rr:GraphMap is the class of graph maps.
- rr:PredicateObjectMap is the class of predicate-object maps.
- rr:RefObjectMap is the class of referencing object maps.
- rr:Join is the class of join conditions.

The members of these classes are collectively called mapping components. The R2RML vocabulary also represents the following properties:

- rr:child represents a child column.
- rr:class represents a class IRI.
- rr:column represents a column name.

- rr:datatype represents a specified datatype.
- rr:constant represents a constant value.
- rr:graph represents a constant shortcut property.
- rr:graphMap represents a graph map.
- rr:inverseExpression represents a inverse expression.
- rr:joinCondition represents a join condition.
- rr:language represents a specified language tag.
- rr:logicalTable represents a logical table.
- rr:object represents a constant shortcut property.
- rr:objectMap represents a object map, referencing object map.
- rr:parent represents a parent column.
- rr:parentTriplesMap represents a parent triples map.
- rr:predicate represents a constant shortcut property.
- rr:predicateMap represents a predicate map.
- rr:predicateObjectMap represents a predicate-object map.
- rr:sqlQuery represents a SQL query.
- rr:sqlVersion represents a SQL version identifier.
- rr:subject represents a constant shortcut property.
- rr:subjectMap represents a subject map.
- rr:tableName represents a table or view name.
- rr:template represents a string template.
- rr:termType represents a term type.

Other terms existing in this vocabulary are listed below.

- rr:defaultGraph denotes a default graph.
- rr:SQL2008 denotes a Core SQL 2008.
- rr:IRI denotes an IRI.
- rr:BlankNode denotes a blank node.
- rr:Literal denotes a literal.

2.6. Triplification of Data Cubes

One might follow different strategies to represent a data cube in RDF. On one extreme, there is the *complete triplification strategy* that separately represents each observation in the cube. On the other extreme, there is the *single-triple strategy* that represents the cube as a single triple.

In more detail, let $R \subseteq D_1 \times \ldots \times D_m \times A_1 \times \ldots \times A_n \times O$ be a relation that represents a data cube, as in Section 2.3.

The complete triplification strategy is as follows. According to Noy et al. (2006), it is possible to represent *R* by reification, 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 $(u, d_1, x_1), \ldots, (u, d_m, x_m), (u, a_1, y_1), \ldots, (u, a_n, y_n), (u, o, z)$.

The problem with this strategy is that the number of triples might be huge.

For example, for a simple cube with 3 dimensions, each one with 10 values, there are 10**3, i.e., 1,000 observations and, so, 1,000 triples.

In view of this observation, one question arises: “Should the Linked Data principles be strictly applied to triplify data cubes?” Indeed, one of the Linked Data principles recommends using URIs as names for things. Assigning one URI to each observation probably will not be needed as it is likely that there will be no situation where a specific observation has to be separately identified.

To address this problem, one may adopt the single-triple strategy, that is, to represent the whole cube as a single triple. The triple would have a subject that would be some URI that identifies the cube, the predicate would be a property such as hasObservations and the object that would be a XML document *X* containing a compact representation of all observations in the cube. The only disadvantage of this approach is that it would require the application that consumes the triples to know the XML schema that describes *X.* This would add one more layer of complexity to the process of consuming the data.

In this dissertation just the *complete triplification strategy* is considered. To generate the set of RDF triples to represent the cube, it was adopted the R2RML Mapping Language.

There are different ways to generate the mappings between the relational databases and the RDF vocabularies. One important decision to make is when to treat the values of the columns of a relational table as an object property or as a datatype property.

Object properties require a definition of an URIs for the resources and allow the use of owl:sameAs to link resources to external RDF databases, enabling the resources to be in a global context. On the other hand, datatype properties do not require the definition of the URIs, which can simplify the mapping. However, without the definition of URIs, owl:sameAs links cannot be created, not allowing the resources to be connected to other resources in external databases. Also, the range of a datatype property must be defined as a XML Schema datatype (Biron & Malhotra 2004).

The choice between an object property or a datatype property basically depends on whether it is desirable to connect resources and give them a semantic meaning or not. For instance, it is useful to create an URI to represent a country but there is no reason to represent a number with the help of an URI.

2.7.
Data Cube Explanation and Example

A data cube (Cyganiak et al. 2013) is organized according to a set of dimensions, measures and attributes. Collectively they are called components of a data cube.

The dimension components serve to identify the observations. A set of values for all the dimension components is sufficient to identify a single observation.

The measure components represent the phenomenon being observed.

The attribute components allow qualifying and interpreting the observed value. They enable specification of the units of measure, any scaling factors and metadata such as the status of the observation.

The Data Cube vocabulary represents the dimensions, attributes and measures as RDF properties. RDF (Manola et al. 2004) provides a standard for the representation of the information that describes those entities and concepts, and is returned by dereferencing the URIs.

The rest of this section describes an example of how to represent a data cube from relational databases in RDF. The Relational Schema for this example is presented in Figure 5.

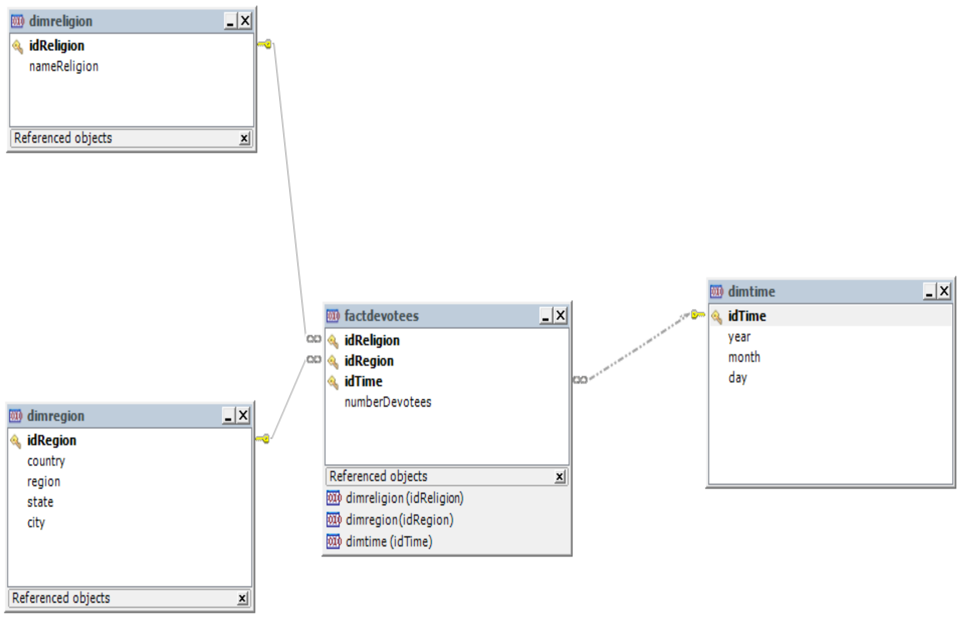

Figure 5 - Relational Schema for Cube Devotees

The relational schema presented in Figure 5 shows a fact table *factdevotees* and the dimensional tables *dimreligion*, *dimtime* and *dimregion*. It is desirable for Cube Devotees to be composed by the values of certain columns from the dimensional tables – numberDevotees (factdevotees), nameReligion (dimReligion), state (dimregion) and year (dimtime).

Using the data cube vocabulary, the Cube Devotees will take the form shown in Table 2.

```
@prefix ex-resource:      <http://purl.org/GovDataCube/resources/>.
@prefix ex-property:      <http://purl.org/GovDataCube/properties/>.
@prefix rdfs:             <http://www.w3.org/2000/01/rdf-schema#>.
@prefix qb:               <http://purl.org/linked-data/cube#>.
@prefix sdmx-attribute:   <http://purl.org/linked-data/sdmx/2009/attribute#>.


ex-resource:dataset-devotees a qb:DataSet;
    rdfs:label "Number of devotees";
    rdfs:comment "The number of devotees in Brazil of each religion by state and time. Dimensions:
DimReligion, DimState and DimYear.";
    qb:structure ex-resource:dsd-devotees.

ex-resource:dsd-devotees a qb:DataStructureDefinition;
     #The dimensions
     qb:component [qb:dimension ex-resource:dimReligion];
     qb:component [qb:dimension ex-resource:dimState];
     qb:component [qb:dimension ex-resource:dimYear];
     # The measures
     qb:component [qb:measure ex-property:numberDevotees];
     # The attributes
     qb:component [qb:attribute sdmx-attribute:unitMeasure; qb:componentAttachment qb:DataSet;].
```

Table 2 - Definition of the Cube Devotees

Table 2 shows the use of the following terms from the data cube vocabulary: qb:DataStructureDefinition, qb:DataSet, qb:component, qb:dimension, qb:measure, qb:attribute and qb:componentAttachment.

As discussed in Section 2.4, qb:DataStructureDefinition defines the structure of the datasets. In particular it defines the dimensions, attributes and measures used in the dataset.

The notion of a Data Structure Definition makes possible to define just once the structure and then reuse it for each publication in the series. Consumers can then be confident that the structure of the data has not changed.

qb:DataSet represents a collection of observations, conforming to some common dimensional structure.

qb:component specifies the components that constitute the data cube. These components are qb:dimension, qb:measure and qb:attribute (which makes explicit that the components are dimensions, measures or attributes).

qb:componentAttachment indicates the level at which the component property should be attached.

2.8. Summary of Chapter 2

This chapter discussed the theoretical foundations which were the basis for the development of this dissertation. Topics such as Linked Data, RDF, the formal definition of data cubes, the Data Cube Vocabulary, R2RML and the triplification of data cubes were addressed to understand how to describe data cubes in RDF.