# 3
# Proposed Architecture for the Catalogue of Linked Data Cube Descriptions

## 3.1.
## Overview of the Proposed Architecture

Figure 6 illustrates the proposed architecture for the Catalogue of Linked Data Cube Descriptions. The architecture comprises the following major components: *Catalogue, SameAs Generator*, *Data Cubes Charger*, *External Interface*, *SPARQL endpoint* and *Internal Interface*.

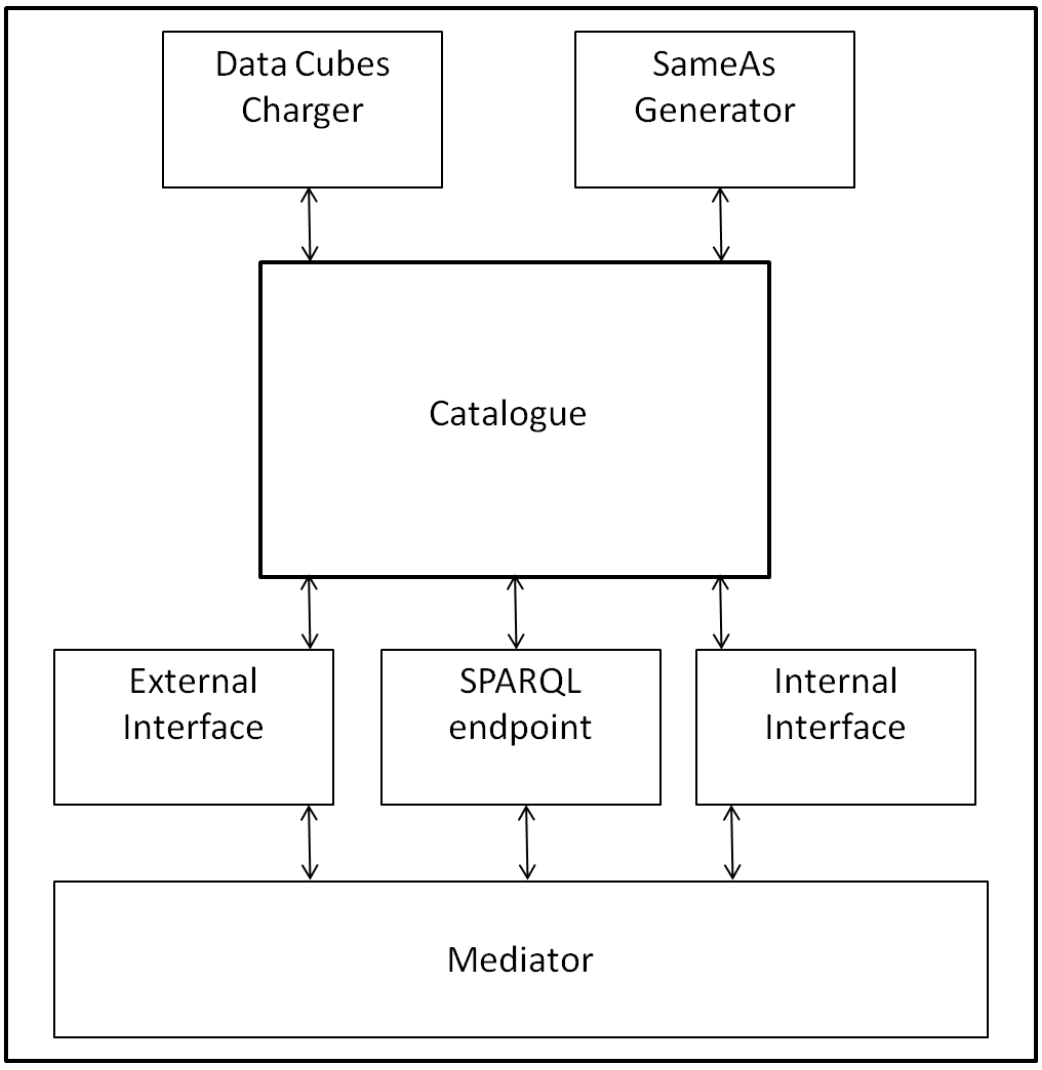


Figure 6 - Overview of the Proposed Architecture

The following sections will be dedicated to describe each one of the components listed above.

## 3.2. Catalogue of Linked Data Cube Descriptions

The *Catalogue of Linked Data Cube Descriptions* is created according to the Linked Data principles (Heath & Bizer 2011). This catalogue contains standardized descriptions in RDF about each data cube stored in a statistical (relational) database known through the mediation environment. Thus, a data cube description, called a *linked data cube description*, is nothing more than a set of RDF triples.

The triples stored in the catalogue describe the dimensions and the attributes of each data cube, including dimension domain values and their connections to external databases. However, a linked data cube description does not contain triples that capture the observations, i.e., it is not a complete materialization of a data cube in RDF; the data cube observations still remain in the relational database. Therefore, the catalogue contains metadata about the data cubes but not the observations themselves.

To allow the connection to other external data sources, the materialization of the dimensions can also be stored in the catalogue. This is essential to insert the data cubes in other contexts – following the Linked Data principles (Heath & Bizer 2011) – thereby helping making the semantics of the data cubes explicit.

Considering that the catalogue contains different types of information, it is divided into two parts, *public* and *private* data. Public data refers to the data cube descriptions, dimension values and owl:*sameAs* triples (Bechhofer et al. 2004) (that relate resources in the linked data cube descriptions with resources located in external data sets). Private data refers to the information required internally, i.e., connections to the databases and mappings from the underlying relational databases to the RDF data cubes. For each linked data cube description in the catalogue, there is at least one mapping to a star-shaped view schema of an underlying database.

To enable the easy consumption of the data, well known vocabularies were used for the linked data cube descriptions in the catalogue, as recommended in (Heath & Bizer 2011). The *RDF Data Cube Vocabulary* (Cyganiak et al. 2013) was adopted for the linked data cube descriptions and the *R2RML Mapping Language* (Das et al. 2010) for the mappings. Some terms from the *D2RQ*

*Mapping Language* (Cyganiak et al. 2012) were also used, due to the fact that the R2RML Mapping Language does not include terms related to the connection to the databases, such as user name and password, for example.

The next tables contain different examples of the information stored inside of the catalogue, to illustrate how the vocabularies are used.

```
@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-residents a qb:DataSet;
      rdfs:label "Number of residents";
      rdfs:comment "The number of residents in Brazil of each sex, race and range of age by area and time.
Dimensions: DimRace, DimSex, DimAge, DimArea and DimYear.";
      qb:structure ex-resource:dsd-residents.

ex-resource:dsd-residents a qb:DataStructureDefinition;
     # The dimensions
     qb:component [qb:dimension ex-resource:dimRace];
     qb:component [qb:dimension ex-resource:dimSex];
     qb:component [qb:dimension ex-resource:dimAge];
     qb:component [qb:dimension ex-resource:dimYear];
     qb:component [qb:dimension ex-resource:dimArea];
     # The measure
     qb:component [qb:measure ex-property:numberResidents];
     # The attributes
     qb:component [qb:attribute sdmx-attribute:unitMeasure; qb:componentAttachment qb:DataSet;].
```

Table 1 - Definition of the Cube Residents

Table 3 illustrates the description of a data cubes.. The structure of the data cube explains how the cube is formed and shows its components. In this example the components are five dimensions, one measure and one attribute.

Table 4 shows the connection information to a relational database (RDB). The connection presented in this example is to the RDB *project1* and it is named *ex-resource:databaseResidents*. This information is used when the R2RML mapping to this RDB is established.

Tables 5 and 6 show the R2RML mappings.

```
@prefix ex-resource:      <http://purl.org/GovDataCube/resources/> .
@prefix d2rq:             <http://www.wiwiss.fu-berlin.de/suhl/bizer/D2RQ/0.1#>.

ex-resource:databaseResidents a d2rq:Database;
      d2rq:username "root";
      d2rq:password "root";
      d2rq:jdbcDSN "jdbc:mysql://localhost/project1";
      d2rq:jdbcDriver "com.mysql.jdbc.Driver".
```

Table 2 - Definition of the connection to the RDB *project1*

```
@prefix ex-resource:       <http://purl.org/GovDataCube/resources/> .
@prefix ex-property:       <http://purl.org/GovDataCube/properties/>.
@prefix rdf:               <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rr:                <http://www.w3.org/ns/r2rml#>.
@prefix d2rq:              <http://www.wiwiss.fu-berlin.de/suhl/bizer/D2RQ/0.1#>.
@prefix qb:                <http://purl.org/linked-data/cube#>.
@prefix sdmx-attribute:    <http://purl.org/linked-data/sdmx/2009/attribute#>.

ex-resource:TriplesMapFactResidents a rr:TriplesMap;
      d2rq:dataStorage ex-resource:databaseResidents;
      rr:logicalTable [ rr:sqlQuery """
                              SELECT DISTINCT idPerson, idArea, idTime, numberResidents
                              FROM factResidents
                              WHERE numberResidents<>0 """; ];
      rr:subjectMap [
          rr:template
"http://purl.org/GovDataCube/resources/Observations/{idPerson}_{idArea}_{idTime}_{numberResidents}";
           rr:class qb:Observation; ];
      rr:predicateObjectMap [
          rr:predicate qb:dataSet;
          rr:objectMap [rr:constant ex-resource:dataset-residents]; ];
       rr:predicateObjectMap [
           rr:predicate ex-resource:dimSex;
           rr:objectMap [
               rr:parentTriplesMap ex-resource:TriplesMapSex;
               rr:joinCondition [
                   rr:child "idPerson";
                   rr:parent "idPerson"; ];];];
       rr:predicateObjectMap [
           rr:predicate ex-resource:dimRace;
           rr:objectMap [
               rr:parentTriplesMap ex-resource:TriplesMapRace;
               rr:joinCondition [
                   rr:child "idPerson";
                   rr:parent "idPerson"; ];];];
       rr:predicateObjectMap [
           rr:predicate ex-resource:dimAge;
           rr:objectMap [
               rr:parentTriplesMap ex-resource:TriplesMapAge;
```

```
            rr:joinCondition [
                rr:child "idPerson";
               rr:parent "idPerson"; ];];];
    rr:predicateObjectMap [
        rr:predicate ex-resource:dimArea;
        rr:objectMap [
            rr:parentTriplesMap ex-resource:TriplesMapArea;
            rr:joinCondition [
                rr:child "idArea";
                rr:parent "idArea"; ];];];
    rr:predicateObjectMap [
        rr:predicate ex-resource:dimYear;
        rr:objectMap [
            rr:parentTriplesMap ex-resource:TriplesMapYear;
            rr:joinCondition [
                rr:child "idTime";
                rr:parent "idTime"; ];];];
    rr:predicateObjectMap [
        rr:predicate ex-property:numberResidents;
        rr:objectMap [rr:column "numberResidents"]; ];
    rr:predicateObjectMap [
        rr:predicate sdmx-attribute:unitMeasure;
        rr:objectMap [rr:constant <http://dbpedia.org/ontology/Person>]; ].
```

Table 3 - Mapping of Cube Residents

```
@prefix rdfs:           <http://www.w3.org/2000/01/rdf-schema#>.
@prefix rdf:            <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rr:             <http://www.w3.org/ns/r2rml#>.
@prefix ex-class:       <http://purl.org/GovDataCube/classes/>.
@prefix ex-resource:    <http://purl.org/GovDataCube/resources/>.
@prefix d2rq:           <http://www.wiwiss.fu-berlin.de/suhl/bizer/D2RQ/0.1#>.

ex-resource:TriplesMapRace a rr:TriplesMap;
    d2rq:dataStorage ex-resource:databaseResidents;
    rr:logicalTable [ rr:sqlQuery """
                      SELECT DISTINCT md5(race) as race_md5, race, idPerson
                      FROM dimperson
                      WHERE race IS NOT NULL """; ];
    rr:subjectMap [
        rr:template "http://purl.org/GovDataCube/resources/Race/{race_md5}";
        rr:class ex-class:Race; ];
    rr:predicateObjectMap [
        rr:predicate rdfs:label;
        rr:objectMap [ rr:column "race"; rr:language "pt" ];].
```

Table 4 - Mapping of Dimension Race

One of the catalogue features is the reuse of dimensions. Reusing dimensions decreases the number of triples stored in the catalogue and enables the standardization of the triples of the same dimension in all different cubes from all RDBs.

Table 5 describes the mapping of the Residents cube and illustrates the question of reuse. It is easily seen that every dimension of the Residents cube calls a triples map from a dimension previously defined. Table 6 shows the mapping of one of the dimensions called in the mapping of Residents cube.

It is important to remark that the dimension values can be stored in the same or different RDB, i.e., some cubes can use dimensions that are stored in an RDB different from the RDB where the cube is stored. This is the reason why it was needed to use D2RQ Mapping Language inside the mapping defined in R2RML. The use of D2RQ in the triples map explicitly informs where the mapping must be executed.



## 3.3. SameAs Generator

The *SameAs Generator* is an application that creates links between data cube descriptions – especially the dimensions - and other data sources. The links are also stored in the catalogue and are expressed using owl:sameAs (Bechhofer et al. 2004).

The application groups different *sameAs* according to the context involved. Each dimension from a data cube has a different range and a different target of external data sources to be linked, thus a different sameAs rule. Therefore, there are as many sameAs groups as dimensions. To create the links with the external data sources, the SameAs Generator uses the *Link Discovery Framework for Metric Spaces- LIMES* (Ngomo & Auer 2011).

LIMES (Ngomo & Auer 2011) provides a semi-automatic approach for the discovery of links between Link Data sources and implements, for large-scale link discovery, time-efficient approaches based on the characteristics of metric spaces. By using approximations, a large number of instance pairs that do not conform to these conditions are filtered out. And just the real similarities of the remaining instances pairs will be computed to return the matching instances. LIMES uses a significantly smaller number of comparisons than brute force approaches by using synthetic data.

For each class, i.e. context, different conditions are applied as it has connections to different data sources. To decide to which external data sources to connect, the SameAs Generator uses a tool to recommend the RDF knowledge bases that are relevant to the context and that are worth to link to (Herrera 2012).

An example of a recommendation of a data source to link with the class Country is DBpedia (Auer et al. 2007). Table 7 shows a triple generated to express this linkage.

| |
|---|
| **Object:** http://purl.org/GovDataCube/classes/Country |
| **Predicate:** http:// www.w3.org/2002/07/owl#sameAs |
| **Subject:** http:// dbpedia.org/ontology/Country |

Table 5 - RDF Triple representing the owl:sameAs of the Class Country

All resources from the Country class are likewise linked with DBpedia. Table 8 illustrates an example of the linkage of one of the resources from this class.

| |
|---|
| **Object:** http://purl.org/GovDataCube/resources/Country/232bf11cb81bcdb269f76a08fde8b947 |
| **Predicate:** http:// www.w3.org/2002/07/owl#sameAs |
| **Subject:** http://dbpedia.org/resource/Angola |

Table 6 - RDF Triple of a resource from Class Country

The triples that the SameAs Generator create are stored in the catalogue along with information about their provenance. It is always recommendable that primary data is stored together with provenance metadata, which enables consulting the origin and the quality of the data. The vocabulary used to express this metadata is *Dublin Core Metadata Initiative (DCMI) Metadata Terms* (DCMI 2012), which provides terms such as *creator* and *date*.

## 3.4. Data Cubes Charger

The *Data Cubes Charger* is an application that generates R2RML mappings between the relational database schemas and the RDF description of the data cubes and stores those mappings inside of the catalogue.

To help specify the mappings between the RDF descriptions and the RDBs, correspondence assertions are used. It is important to remark that just the

mappings related to the instances of the dimensions of the data cubes and their observations are specified. Then, based on these correspondence assertions, the R2RML mapping is generated (Vidal et al. 2012).

There are three types of correspondence assertions: (i) class correspondence assertion (CCA), matching a class and a relation schema; (ii) object property correspondence assertion (OCA), matching an object property with attributes or paths of a relation schema; (iii) datatype property correspondence assertion (DCA), matching a datatype property with attributes or paths of a relation schema (Vidal et al. 2012).

A running example is provided below to better understand how the *Data Cubes Charger* operates. The running example considers a data cube that has already been defined in the catalogue.

The *Data Cubes Charger* generates a Cube ID for the user, when prompted to do so. For example, if the ID is “cube1”, the URI generated takes the form http://purl.org/GovDataCube/resources/cube1.

The observations of a data cube have URIs which are generated by concatenating the cube URI with the primary keys from the fact table of the RDB, ensuring the uniqueness of the URIs for the observations.

To create the data cube, the user has to select one fact table and the dimension tables desired. The user must also fill in the description of the data cube to be used as a *rdfs:label* and *rdfs:comment*.

The next figures illustrate the triplification of Residents cube.

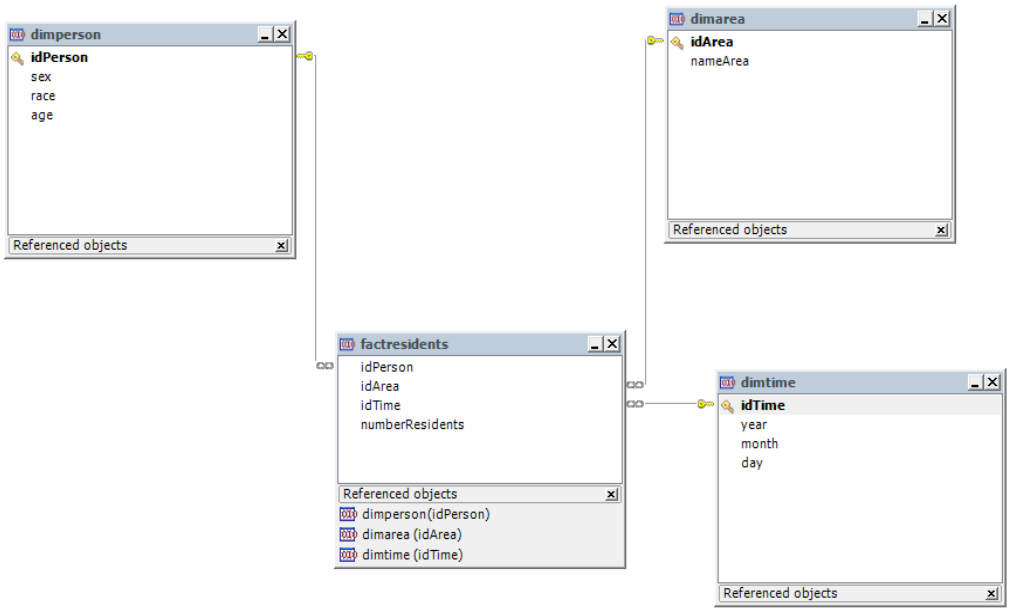


Figure 7 - Database Schema

```
@prefix rdf:                <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs:               <http://www.w3.org/2000/01/rdf-schema#>.
@prefix xsd:                <http://www.w3.org/2001/XMLSchema#>.
@prefix qb:                 <http://purl.org/linked-data/cube#>.
@prefix ex-class:           <http://purl.org/GovDataCube/classes/>.
@prefix ex-property:        <http://purl.org/GovDataCube/properties/> .
@prefix ex-resource:        <http://purl.org/GovDataCube/resources/> .
@prefix sdmx-dimension:     <http://purl.org/linked-data/sdmx/2009/dimension#>.
@prefix sdmx-attribute:     <http://purl.org/linked-data/sdmx/2009/attribute#>.
@prefix sdmx-measure:       <http://purl.org/linked-data/sdmx/2009/measure#>.
@prefix sdmx-concept:       <http://purl.org/linked-data/sdmx/2009/concept#>.

ex-resource:dataset-residents a qb:DataSet;
    rdfs:label "Number of residents";
    rdfs:comment "The number of residents in Brazil of each sex, race and range of age by area and time.
Dimensions: DimRace, DimSex, DimAge, DimArea and DimYear.";
    qb:structure ex-resource:dsd-residents.

ex-resource:dsd-residents a qb:DataStructureDefinition;
    # The dimensions
    qb:component [qb:dimension ex-resource:dimRace];
    qb:component [qb:dimension ex-resource:dimSex];
    qb:component [qb:dimension ex-resource:dimAge];
    qb:component [qb:dimension ex-resource:dimYear];
    qb:component [qb:dimension ex-resource:dimArea];
    # The measure
    b:component [qb:measure ex-property:numberResidents];
    # The attributes
    qb:component [qb:attribute sdmx-attribute:unitMeasure; qb:componentAttachment qb:DataSet;].

ex-property:numberResidents a rdf:Property, qb:MeasureProperty;
    rdfs:label "number of residents";
    rdfs:subPropertyOf sdmx-measure:obsValue;
    rdfs:range xsd:integer .

ex-resource:dimRace a qb:DimensionProperty, rdf:Property;
    rdfs:label "dimension Race";
    rdfs:range ex-class:Race.

ex-resource:dimSex a qb:DimensionProperty, rdf:Property;
    rdfs:label "dimension Sex";
    rdfs:range ex-class:Sex.

ex-resource:dimAge a qb:DimensionProperty, rdf:Property;
    rdfs:label "dimension Age";
    rdfs:range ex-class:Age.

ex-resource:dimYear a qb:DimensionProperty, rdf:Property;
    rdfs:label "dimension Year";
    rdfs:subProperyOf sdmx-dimension:refTime;
```

```
        qb:concept sdmx-concept:refTime;
        rdfs:range ex-class:Year.

ex-resource:dimArea a qb:DimensionProperty, rdf:Property;
       rdfs:label "dimension Area";
       rdfs:range ex-class:Area.

ex-class:Race rdf:type rdfs:Class;
        rdfs:comment "Represents a race";
        rdfs:label "Race".

ex-class:Sex rdf:type rdfs:Class;
        rdfs:comment "Represents a sex gender";
        rdfs:label "Sex".

ex-class:Age rdf:type rdfs:Class;
       rdfs:comment "Represents a age";
       rdfs:label "Age".

ex-class:Year rdf:type rdfs:Class;
        rdfs:comment "Represents a year";
        rdfs:label "Year".

ex-class:Area rdf:type rdfs:Class;
        rdfs:comment "Represents a area";
        rdfs:label "Area“.
```

Table 7 - Description in RDF related with the Cube Residents

The Correspondence Assertions between the relation schema presented in Figure 7 and the RDF description of the Cube Residents from Table 9 are represented in the next table:

| | |
|---|---|
| CCA1 | qb:Observation ≡ factresidents [idPerson, idArea, idTime] |
| CCA2 | ex-class:Race ≡ dimperson [race] |
| CCA3 | ex-class:Sex ≡ dimperson [sex] |
| CCA4 | ex-class:Age ≡ dimperson [age] |
| CCA5 | ex-class:Year ≡ dimtime [year] |
| CCA6 | ex-class:Area ≡ dimarea [nameArea] |
| OCA1 | ex-resource:dimRace ≡ dimperson / NULL |
| OCA2 | ex-resource:dimSex ≡ dimperson / NULL |

| | |
|---|---|
| OCA3 | ex-resource:dimAge ≡ dimperson / NULL |
| OCA4 | ex-resource:dimYear ≡ dimtime / NULL |
| OCA5 | ex-resource:dimArea ≡ dimarea / NULL |
| DCA1 | rdfs:label ≡ dimperson / race |
| DCA2 | rdfs:label ≡ dimperson / sex |
| DCA3 | rdfs:label ≡ dimperson / age |
| DCA4 | rdfs:label ≡ dimtime / year |
| DCA5 | rdfs:label ≡ dimarea / nameArea |
| DCA6 | ex-property:numberResidents ≡ factresidents / numberResidents |

Table 8 - Correspondence Assertions



With the information provided by the correspondence assertions listed in Table 10, the R2RML mappings can be generated.

An R2RML mapping consist of a subject map and multiple predicate-object maps that in turn consist of predicate maps and object maps (or referencing object maps). A subject map is defined for each CCA, a predicate map for each DAC and an object map for each OCA (Vidal et al. 2012).

The next tables show the R2RML generated for the running example.

```
1  <#TriplesMapRace>
2      rr:logicalTable [
3        rr:sqlQuery """
4        SELECT DISTINCT md5(race) as race_md5, race, idPerson
5        FROM dimperson
6        WHERE race IS NOT NULL """; ];
7      rr:subjectMap [
8        rr:template "http://purl.org/GovDataCube/resources/Race/{race_md5}";
9        rr:class ex-class:Race; ];
10     rr:predicateObjectMap [
11       rr:predicate rdfs:label;
12       rr:objectMap [ rr:column "race" ];].
```

Table 9 - R2RML Mappings generated from *CCA2* and *DCA1*

In the R2RML mapping shown in Table 11, the *CCA2* translates to the subject map for the target class *ex-class:Race* (7-9) and the *DCA1* translates to the predicate map for the datatype property *rdfs:label* (10-12). This mapping

specifies that each tuple *t* from the *R2RML view* (3-6) produces two RDF triples (the translations from attribute values to RDF literals are omitted for simplicity):

*<http://purl.org/GovDataCube/resources/Race/t.race_md5> rdf:type ex-class:Race*.
*<http://purl.org/GovDataCube/resources/Race/t.race_md5> rdfs:label "t.race"*.

```
1  <#TriplesMapSex>
2      rr:logicalTable [
3        rr:sqlQuery """
4        SELECT DISTINCT md5(sex) as sex_md5, sex, idPerson
5        FROM dimperson
6        WHERE sex IS NOT NULL """; ];
7      rr:subjectMap [
8        rr:template "http://purl.org/GovDataCube/resources/Sex/{sex_md5}";
9        rr:class ex-class:Sex; ];
10     rr:predicateObjectMap [
11       rr:predicate rdfs:label;
12       rr:objectMap [ rr:column "sex" ];].
```

Table 10 - R2RML Mappings generated from *CCA3* and *DCA2*

The R2RML mapping shown in Table 12 specifies the subject map for target class *ex-class:Sex*, generated from the *CCA3* (7-9), and the predicate map for datatype property *rdfs:label*, generated from the *DCA2* (10-12). This mapping specifies that each tuple *t* from the *R2RML view* (3-6) produces two RDF triples:

*<http://purl.org/GovDataCube/resources/Sex/t.sex_md5> rdf:type ex-class:Sex*.
*<http://purl.org/GovDataCube/resources/Sex/t.sex_md5> rdfs:label "t.sex"*.

```
1  <#TriplesMapAge>
2      rr:logicalTable [
3          rr:sqlQuery """
4        SELECT DISTINCT md5(age) as age_md5, age, idPerson
5        FROM dimperson
6        WHERE age IS NOT NULL """; ];
7      rr:subjectMap [
8        rr:template "http://purl.org/GovDataCube/resources/Age/{age_md5}";
9        rr:class ex-class:Age; ];
10     rr:predicateObjectMap [
11       rr:predicate rdfs:label;
12       rr:objectMap [ rr:column "age" ];].
```

Table 11 - R2RML Mappings generated from *CCA4* and *DCA3*

Table 13 contains the R2RML mapping, translating the subject map for target class *ex-class:Age* from the *CCA4* (7-9) and the predicate map for

datatype property *rdfs:label* from the *DCA3* (10-12). This mapping specifies that each tuple *t* from the *R2RML view* (3-6) produces two RDF triples:

*<http://purl.org/GovDataCube/resources/Age/t.age_md5> rdf:type ex-class:Age*.
*<http://purl.org/GovDataCube/resources/Age/t.age_md5> rdfs:label "t.age"*

```
1  <#TriplesMapYear>
2      rr:logicalTable [
3        rr:sqlQuery """
4        SELECT DISTINCT md5(year) as year_md5, year, idTime
5        FROM dimtime
6        WHERE age IS NOT NULL """; ];
7      rr:subjectMap [
8        rr:template "http://purl.org/GovDataCube/resources/Year/{year_md5}";
9        rr:class ex-class:Year; ];
10     rr:predicateObjectMap [
11       rr:predicate rdfs:label;
12       rr:objectMap [ rr:column "year" ];].
```

Table 12 - R2RML Mappings generated from *CCA5* and *DCA4*

In Table 14, the R2RML mapping describes the subject map for target class *ex-class:Year*, generated from the *CCA5* (7-9), and the predicate map for datatype property *rdfs:label*, generated from the *DCA4* (10-12). This mapping specifies that each tuple *t* from the *R2RML view* (3-6) produces two RDF triples:

*<http://purl.org/GovDataCube/resources/Year/t.year_md5> rdf:type ex-class:Year*.
*<http://purl.org/GovDataCube/resources/Year/t.year_md5> rdfs:label "t.year"*.

```
1  <#TriplesMapArea>
2      rr:logicalTable [
3        rr:sqlQuery """
4        SELECT DISTINCT md5(nameArea) as area_md5, nameArea, idArea
5        FROM dimarea
6        WHERE age IS NOT NULL """; ];
7      rr:subjectMap [
8        rr:template "http://purl.org/GovDataCube/resources/Area/{area_md5}";
9        rr:class ex-class:Area; ];
10     rr:predicateObjectMap [
11       rr:predicate rdfs:label;
12       rr:objectMap [ rr:column "nameArea" ];].
```

Table 13 - R2RML Mappings generated from *CCA6* and *DCA5*

The R2RML mapping shown in Table 15 translates to the subject map for the target class *ex-class:Area*, generated from the *CCA6* (7-9), and translates to the predicate map for the datatype property *rdfs:label*, generated from the *DCA5* (10-12). This mapping specifies that each tuple *t* from the *R2RML view* (3-6) produces two RDF triples:

*<http://purl.org/GovDataCube/resources/Area/t.area_md5> rdf:type ex-class:Area*.
*<http://purl.org/GovDataCube/resources/Area/t.area_md5> rdfs:label "t.nameArea"*.

```
1  <#TriplesMapObservations>
2      rr:logicalTable [
3        rr:sqlQuery """
4        SELECT DISTINCT idPerson, idArea, idTime, numberResidents
5        FROM  factresidents
6        WHERE  numberResidents<>0 """; ];
7      rr:subjectMap [
8         rr:template
9  "http://purl.org/GovDataCube/resources/cube1/Observations/{idPerson}_{idArea}_{idTime}";
10        rr:class qb:Observation; ];
11     rr:predicateObjectMap [
12        rr:predicate ex-resource:dimRace;
13        rr:objectMap [
14           rr:parentTriplesMap <#TriplesMapRace>;
15           rr:joinCondition [
16              rr:child "idPerson";
17              rr:parent "idPerson";];];];
18     rr:predicateObjectMap [
19        rr:predicate ex-resource:dimSex;
20        rr:objectMap [
21           rr:parentTriplesMap <#TriplesMapSex>;
22           rr:joinCondition [
23              rr:child "idPerson";
24              rr:parent "idPerson"; ]; ];];
25     rr:predicateObjectMap [
26       rr:predicate ex-resource:dimAge;
27       rr:objectMap [
28          rr:parentTriplesMap <#TriplesMapAge>;
29          rr:joinCondition [
30            rr:child "idPerson";
31            rr:parent "idPerson"; ]; ];];
32     rr:predicateObjectMap [
33       rr:predicate ex-resource:dimArea;
34       rr:objectMap [
35          rr:parentTriplesMap <#TriplesMapArea>;
36          rr:joinCondition [
37             rr:child "idArea";
38             rr:parent "idArea";];];];
```

```
39    rr:predicateObjectMap [
40       rr:predicate ex-resource:dimYear;
41       rr:objectMap [
42          rr:parentTriplesMap <#TriplesMapYear>;
43          rr:joinCondition [
44            rr:child "idTime";
45            rr:parent "idTime";];];];
46    rr:predicateObjectMap [
47       rr:predicate ex-property:numberResidents;
48       rr:objectMap [rr:column "numberResidents"];].
```

Table 14 - R2RML Mappings generated from *CCA1*, *OCA1*, *OCA2*, *OCA3*, *OCA4*, *OCA5* and *DCA6*

Table 16 shows the R2RML mapping of the data cube. The *CCA1* translates to the subject map for the target class *qb:Observation* (7-10). It specifies the object map for the object properties ex-resource:dimRace, generated from the *OCA1* (11-17), ex-resource:dimSex generated from the *OCA2* (18-24), ex-resource:dimAge generated from the *OCA3* (25-31), ex-resource:dimYear generated from the *OCA4* (39-45) and ex-resource:dimArea generated from the *OCA5* (32-38). The *DCA6* translates to the predicate map for the datatype property *ex-property:numberResidents* (46-48).

This mapping performs joins between *factresidents* and *dimperson*, *dimarea* and *dimtime*. It also specifies the pairs <*t,t'*>, <*t,t''*>, <*t,t'''*>, where *t* is a tuple in *factresidents*, *t'* is a tuple in *dimperson*, *t''* is a tuple in *dimarea* and *t'''* is a tuple in *dimtime.* For each tuple *t* from the *R2RML view* (3-6), this R2RML mapping produces the following RDF triples:

<http://purl.org/GovDataCube/resources/cube1/Observations/t.idPerson_idArea_idTime> *rdf:type qb:Observation*.

<http://purl.org/GovDataCube/resources/cube1/Observations/t.idPerson_idArea_idTime> *ex-resource:dimRace <http://purl.org/GovDataCube/resources/Race/t'.race_md5>.*

<http://purl.org/GovDataCube/resources/cube1/Observations/t.idPerson_idArea_idTime> *ex-resource:dimSex <http://purl.org/GovDataCube/resources/Sex/t'.sex_md5>.*

<http://purl.org/GovDataCube/resources/cube1/Observations/t.idPerson_idArea_idTime> *ex-resource:dimAge <http://purl.org/GovDataCube/resources/Age/t'.age_md5>.*

<http://purl.org/GovDataCube/resources/cube1/Observations/t.idPerson_idArea_idTime>

*ex-resource:dimArea <http://purl.org/GovDataCube/resources/Area/t''.area_md5>.*

<http://purl.org/GovDataCube/resources/cube1/Observations/t.idPerson_idArea_idTime>
*ex-resource:dimYear <http://purl.org/GovDataCube/resources/Year/t'''.year_md5>.*

<http://purl.org/GovDataCube/resources/cube1/Observations/t.idPerson_idArea_idTime>
*ex-property:numberResidents "t.numberResidents"*.

The R2RML mappings generated from this application are stored in the Catalogue of Linked Data Cube Descriptions for consumption.

## 3.5. Interfaces

The *SPARQL endpoint* allows the mediator or the user to query the catalogue via the SPARQL query language. The SPARQL query language for RDF is designed to meet the use cases and requirements identified by the RDF Data Access Working Group in (Clark 2005). The results of SPARQL queries can be results sets or RDF graphs (Prud'hommeaux & Seaborne 2008a).

The SPARQL Protocol and RDF Query Language (SPARQL) is a query language and protocol for RDF. The SPARQL Protocol contains one interface, *SparqlQuery*, containing one operation, *query*. The SPARQL Protocol is described in two ways: primarily as an abstract interface independent of any concrete realization, implementation, or binding to another protocol; and secondly, as HTTP and SOAP bindings of this interface (Prud'hommeaux & Seaborne 2008b).

A *SPARQL endpoint* implements the SPARQL protocol and makes it possible for users, which can be humans or machines, to query a knowledge base via the SPARQL query language. Results are typically returned in one or more machine-processable formats. The endpoints are primarily for serving requests from applications and not for manual interactions via browser, as they were mostly conceived to be a machine-friendly interface towards a knowledge base.

The *External Interface* is an application that returns the descriptions of the data cubes according to a keyword provided by the mediator. These descriptions consist of the information about the cubes, except their mappings to the underlying relational databases. A more detailed description will be given in Section 4.2.2.

The *Internal Interface* is an application that returns to the mediator the information about the mapping of a certain data cube and the information about the connection to the RDB where it is stored. A more detailed description will be given in Section 4.2.3.

## 3.6. Summary of Chapter 3

This chapter presented the architecture of the Catalogue of Linked Data Cube Descriptions. The catalogue comprises the following modules: Catalogue Store, SameAs Generator, Data Cubes Charger, External Interface, SPARQL endpoint and Internal Interface. These modules were described in detail.