An Open Data Approach to Publish Relational Data

Miguel Bento Alves

ESTG, Instituto Politécnico de Viana do Castelo
4900-348 Viana do Castelo
Atlanta, Georgia 30332–0250
NOVA-LINCS, Universidade Nova de Lisboa
2829-516 Caparica, Portugal
Email: mba@estg.ipvc.pt

João Ferreira Nunes

ESTG, Instituto Politécnico de Viana do Castelo
4900-348 Viana do Castelo
Email: joao.nunes@estg.ipvc.pt

***Abstract*—Open Data is a principle that defines that data should be freely available to all, without any kind of restrictions from copyright, patents or other mechanisms of control. During this work, we've applied this concept into data that was modelled using a relational methodology. Thus, we've transformed the relational data into Open Data using a Semantic Web approach, namely RDF data. Furthermore, we've implemented a set of relational restrictions in the RDF data by means of semantic rules. These rules are used to guarantee the integrity of the Open Data repository. Tools were developed to manipulate the Open Data repository, ensuring the data integrity.**

***Index Terms*—Open Data; Semantic Web; Semantic Rules;**

I. Introduction

Open Data's principle [1] claims that data should be freely available, without any kind of restrictions from copyright, patents or other mechanisms of control. Another key concept that it is implicit to this ideal, is the interoperability, which refers to the capability of several systems and organizations in working together. In this specific case, it refers to the capability to combine - or inter-operate - different sets of data.

The concept of Open Data derives in a sense from Semantic Web [2], introduced by Berners-Lee [3]. Semantic Web is realized by assigning some meaning to the published content over the Internet in a way that it becomes discernible both to humans and computers. In this way, interoperability and cooperation between systems is enhanced. The meaning of the content is achieved by its classification and its relation with ontologies, which is a model that represents a set of concepts within a domain and the relationships between them.

Data publication in open format and its usage has been a hot topic within the scientific community over the past years. The Linked Open Data (LOD) project [4] is a good example of this practice. It aims to create structured and interconnected datasets, generating a data cloud. The LOD project contains more than 31 billion facts, linking more than 500 million facts to each other. A central dataset in the Linked Open Data project is the DBpedia [5], created from data extracted from Wikipedia containing over 1 billion facts. The LOD project proposes the publication of data using Web standards along with *links* to other data sources, giving a semantic context that allows easy access and easy interpretation of data. Linked Data also implies the use of standards, such as HTTP, RDF [6] or SPARQL [7], making it easier to use on the Web.

In our project, we created a data repository capable to publish information using the Open Data philosophy, so that it could be used externally either by humans and machines. One of the requirements imposed was that the project information should be centralized and consolidated through Semantic Web principles.

This project arises with the need to share and make public the data produced under the TREASURE project - a Research & Innovation Action financed by European Commission under the Horizon 2020 (grant agreement no. 634476). The aim of the project is to improve knowledge, skills and competences necessary to develop existing and create new sustainable pork chains based on European local pig genetic resources (local breeds). Initially, the information requirements were analyzed based on a relational model approach [8] to create a relational database. In order to enable the reuse of all the work produced during the initial phase of the project, it was decided to replicate the relational model for a Semantic Web approach. Therefore, the entire relational model was transformed into an RDF model. The challenge is to represent the relational structures, consisting of tables, fields and the stored data, in triple *Subject-Property-Subject* inherent to the RDF model. Although the transformation of the relational model into a RDF model is not a new topic (as can be seen in Section V), none of the approaches studied fulfilled our requirements for this project. In the best of our knowledge, we do not identify any work that follows the approach we have done in this work and which will be detailed throughout this document. We've also developed two tools to manipulate data in the open data repository. The first one allows select one local database and transfer the data to the Open Data repository. The second one is a SPARQL endpoint, that can be used either in a program or with a Web interface, that allows the execution of SPARQL commands in the central repository. Both tools guarantees the integrity of the data considering the relational constraints

 ISBN: 978-1-61208-636-1

implemented.

This paper is organized as follows: in Section II, we describe the ontologies created in our approach and the data structure in datasets. In Section III the tools that were developed are described and, in Section IV, we present the created semantic rules to guarantee the integrity of the data. In Section V, some of the related work described is presented and compared to our approach. Conclusions and some ideas to be developed in the near future are presented in Section VI.

II. Ontologies and Datasets

Although the focus of our work was to publish the produced data from the TREASURE project on an Open Data approach, the developed system that we designed is adaptable to any relational database. To accomplish this we proposed a three layers model, where at the upper-level the most important (or most used) concepts of the relational model are modeled; at the middle-level the meta-model of the relational database is modeled; and at the lower-level the database information is represented. With the two highest layers, we have all the knowledge on how the information in a database is organized, and from that we can extract information about what is modeled. This will also allows to support reasoning on the data model, as will be demonstrated throughout this work.

A. Relational Model Ontology

In the upper-level, we have created an ontology to represent the concepts of relational databases. Not all the concepts of relational databases were modeled, since we decided to model the most commonly used concepts of our project. The ontology was modeled in OWL [9] and it is represented in Figure 1.

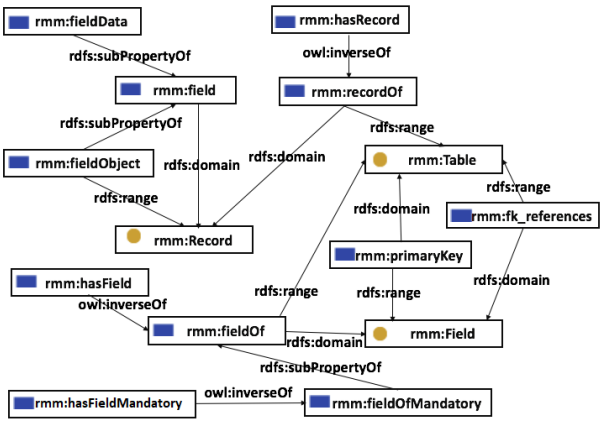

Fig. 1. Relational Model Ontology

B. Database meta-model

At the middle-level is represented the meta-model of a database, namely and as an example, which tables were created, which fields have each table, the primary keys of the tables and the foreign keys. In the Figure 2, a sub-model of the data model of this project is represented. In the list Listing 1 is encoded in OWL the sub-model shown in Figure 2 (due lack of space, we do not list all triples).

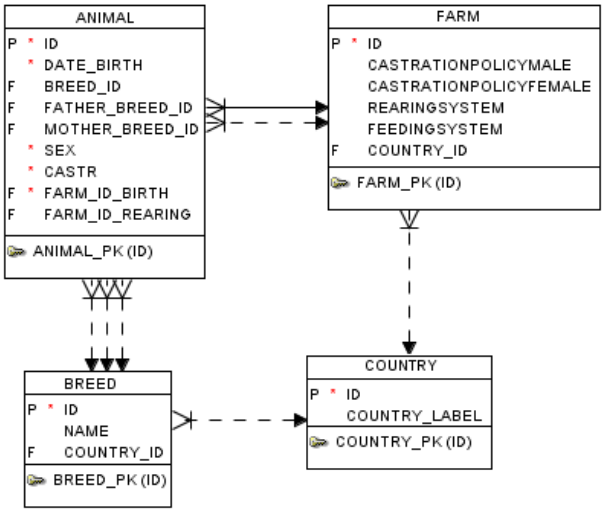

Fig. 2. A sub-model of the project Data Model

```
exa_mm:Country rdf:type rmm:Table ;
      rmm:primaryKey <http://www.example.com/exa_mm.ttl/Country#id>.
<http://www.example.com/exa_mm.ttl/Country#id>rmm:fieldOf exa_mm:Country ;
      rdfs:subPropertyOf rmm:fieldData .
<http://www.example.com/exa_mm.ttl/Country#country_label>
      rmm:fieldOf exa_mm:Country ;
      rdfs:subPropertyOf rmm:fieldData .

exa_mm:Breed rdf:type rmm:Table ;
      rmm:primaryKey <http://www.example.com/exa_mm.ttl/Breed#id>.
<http://www.example.com/exa_mm.ttl/Breed#id>rmm:fieldOf exa_mm:Breed ;
      rdfs:subPropertyOf rmm:fieldData .
<http://www.example.com/exa_mm.ttl/Breed#country_id>
      rmm:fieldOf exa_mm:Breed ;
      rdfs:subPropertyOf rmm:fieldObject ;
      rmm:fk_references exa_mm:Country .

exa_mm:Farm rdf:type rmm:Table ;
      rmm:primaryKey <http://www.example.com/exa_mm.ttl/Farm#id>.
<http://www.example.com/exa_mm.ttl/Farm#id>rmm:fieldOf exa_mm:Farm ;
      rdfs:subPropertyOf rmm:fieldData .
<http://www.example.com/exa_mm.ttl/Farm#castrationPolicyMale>
      rmm:fieldOf exa_mm:Farm ;
      rdfs:subPropertyOf rmm:fieldData .
<http://www.example.com/exa_mm.ttl/Farm#country_id>
      rmm:fieldOf exa_mm:Farm ;
      rdfs:subPropertyOf rmm:fieldObject ;
      rmm:fk_references exa_mm:Country .

exa_mm:Animal rdf:type rmm:Table ;
      rmm:primaryKey <http://www.example.com/exa_mm.ttl/Animal#id>.
<http://www.example.com/exa_mm.ttl/Animal#id>
      rmm:fieldOf exa_mm:Animal ;
      rdfs:subPropertyOf rmm:fieldData .
<http://www.example.com/exa_mm.ttl/Animal#breed_id>
      rmm:fieldOfMandatory exa_mm:Animal ;
      rdfs:subPropertyOf rmm:fieldObject ;
      rmm:fk_references exa_mm:Breed .
<http://www.example.com/exa_mm.ttl/Animal#father_breed_id>
      rmm:fieldOf exa_mm:Animal ;
      rdfs:subPropertyOf rmm:fieldObject ;
      rmm:fk_references exa_mm:Breed .
```

Listing 1 - Encoding of the relational sub-model presented in Figure 2

The middle-level layer that represents the objects of a given database contains the same information that can be found in the data dictionary of the database manager system. Although the catalogs (data dictionaries) of the different database management systems are not standardized, the essential information is available in all of them. In this way, we can create automatisms so that this middle-level layer can be created in an automatic way. In Figure 3 we can see a SQL command that

 ISBN: 978-1-61208-636-1

extracts the metadata (in Oracle) from the relational sub-model represented in the Figure 2 and the result of this command. Comparing the result of the SQL command with the OWL encoding listed in Listing 1 makes it obvious that the OWL encoding can be done automatically.

```
Select utc.table_name, utc.column_name, utc.data_type,
sql.constraint_type, sql.foreignTable, sql.foreignColumn
from user_tab_columns utc
   left join
   (select constraint_type, uc.table_name,
        ucc.column_name, uccr.table_name foreignTable,
        uccr.column_name foreignColumn
    from user_constraints uc
        inner join user_cons_columns ucc
        on (uc.constraint_name = ucc.constraint_name)
        left join user_cons_columns uccr
        on (uc.r_constraint_name = uccr.constraint_name)) sql
    on (utc.table_name = sql.table_name
        and utc.column_name = sql.column_name)
order by table_name, column_id;
```

	TABLE_NAME	COLUMN_NAME	DATA_TYPE	CONSTRAINT_TYPE	FOREIGNTABLE	FOREIGNCOLUMN
1	ANIMAL	ID	NVARCHAR2	P	(null)	(null)
2	ANIMAL	DATE_BIRTH	DATE	C	(null)	(null)
3	ANIMAL	BREED_ID	NVARCHAR2	R	BREED	ID
4	ANIMAL	FATHER_BREED_ID	NVARCHAR2	R	BREED	ID
5	ANIMAL	MOTHER_BREED_ID	NVARCHAR2	R	BREED	ID
6	ANIMAL	SEX	NUMBER	C	(null)	(null)
7	ANIMAL	CASTR	NUMBER	C	(null)	(null)
8	ANIMAL	FARM_ID_BIRTH	NVARCHAR2	R	FARM	ID
9	ANIMAL	FARM_ID_BIRTH	NVARCHAR2	C	(null)	(null)
10	ANIMAL	FARM_ID_REARING	NVARCHAR2	R	FARM	ID
11	BREED	ID	NVARCHAR2	P	(null)	(null)
12	BREED	NAME	NVARCHAR2	C	(null)	(null)
13	BREED	COUNTRY_ID	CHAR	R	COUNTRY	ID
14	COUNTRY	ID	CHAR	P	(null)	(null)
15	COUNTRY	COUNTRY_LABEL	NVARCHAR2	(null)	(null)	(null)
16	FARM	ID	NVARCHAR2	P	(null)	(null)
17	FARM	CASTRATIONPOLICYMALE	NUMBER	(null)	(null)	(null)
18	FARM	CASTRATIONPOLICYFEMALE	NUMBER	(null)	(null)	(null)
19	FARM	REARINGSYSTEM	NUMBER	(null)	(null)	(null)
20	FARM	FEEDINGSYSTEM	NUMBER	(null)	(null)	(null)
21	FARM	COUNTRY_ID	CHAR	R	COUNTRY	ID

Fig. 3. Example of a SQL command that extracts the metadata from the relational sub-model

C. Dataset

In the most specific layer, the data themselves are represented. In our project, the information produced by the TREASURE project, was initially stored in the database. In the list Listing 2 some example data is presented. Note that in the case of foreign keys, we decided to point to the record instead of storing a key value, as was implicit in the high-level ontological model.

```
country:pt rmm:recordOf exa_mm:Country ;
      country:id "PT" ;
      country:country_label "Portugal" .

exa_mm:Country rmm:hasRecord [
      country:id "FR" ;
      country:country_label "French"
].

breed:b2703 rmm:recordOf exa_mm:Breed ;
      breed:id "b2703" ;
      breed:name "Bisaro" ;
      breed:country_id country:pt .
```

Listing 2 - Example of data contained in the dataset

III. System Developed

Our system was developed using the Jena Framework [10], a free and open source Java framework for building Semantic Web applications. It provides a programmatic environment for RDF, RDFS [11], OWL, a query engine for SPARQL and it includes a rule-based inference engine. Jena is widely accepted for Semantic Web applications because it offers an "all-in-one" Java solution. Our system consists by two sub-systems. One of them allows to select one local database and transfer the data to the Open Data repository. The other one is a SPARQL endpoint, that can be used either in a program or with a Web interface, that allows the execution of SPARQL commands in the central repository. We decided to develop our own SPARQL endpoint instead of using Fuseki [12], the SPARQL server of Jena package, because we implemented several relational constraints over our central repository and we want to control the integrity of the data against the relational constraints. Every SPARQL command that change the data content must carry the central repository from an integrity state to other integrity state. The relational constraints implemented are detailed in Section IV. In the Figure 4 is showed the interface of our SPARQL endpoint.

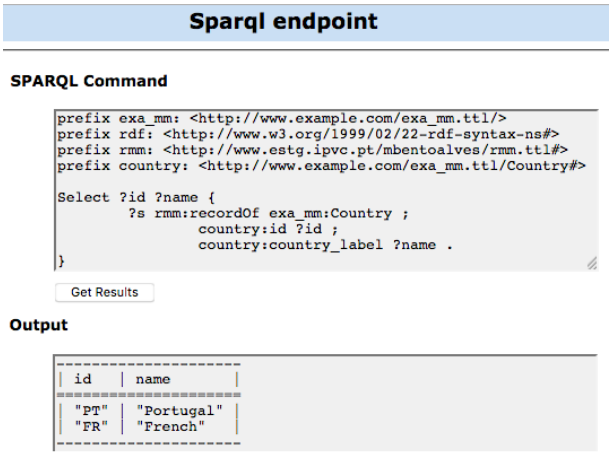

Fig. 4. Sparql Endpoint

IV. Rules to Implement Relational Constraints

When we convert a relational database into an Open Data repository, the main concern is to ensure its integrity, taking into account the inherent constraints of the relational model. In OWL, we do not have all the necessary mechanisms to impose the constraints of the relational model. Therefore, it was necessary to implement relational constraints using semantic rules. A rule language is needed for several reasons, at least because of the limitations of OWL [13].

The process to ensure the integrity of the Open Data repository is as follows: each time a command that can change the data content of the repository is invoked, a SPARQL command is executed to verify if any rule or constraint has not been met; if this occurs, the reverse command is executed in order to reset the database to the last integrity state identified, and an error is issued. All invoked commands that might change the contents of the data repository go to a queue in order to run sequentially. In this approach, the integrity of the data repository with concurrent commands was not thought of.

 ISBN: 978-1-61208-636-1

In general, a row in a table represents a relationship among a set of values where each element is termed an attribute value or a field. Also, in a table, all its fields are distinct, i.e., each table cannot have the same field more than one time. The Rule 1 expresses this condition and if for some reason it occurs, an error message is returned.

```
(err:MultipleFieldError err:violation ?Msg) <-
      (?Record1 ?Field ?Value),
      (?Record1 rdf:type rmm:Record),
      (?Field rdf:type rmm:Field),
      (?Record1 ?Field ?Value2),
      (?Record1 rmm:recordOf ?Table),
      notEqual(?Value, ?Value2),
      strConcat('Table -> ', ?Table, ';record -> ', ?Record1, ', field -> ', ?Field,
?Msg1),
      strConcat(?Msg1, '; values -> (', ?Value, ', ', ?Value2, ')', ?Msg) .
```

Rule 1 - Rule to avoid repeated fields in a table

Each table should have a primary key, a key, or a set of keys, that identifies univocally a row. If two distinct rows have the same primary key, then its constraint is violated. The Rule 2 ensures that the primary key isn't violated. We defined that all primary keys must be constituted by single fields and the Rule 2 assumes this assumption. Furthermore, in sub-section IV-A we discuss about composite keys.

```
(err:PrimaryKeyError err:violation ?Msg) <-
      (?Record ?Field ?Value),
      (?Record rdf:type rmm:Record),
      (?Field rdf:type rmm:Field),
      (?Table rmm:primaryKey ?Field),
      (?Record1 ?Field ?Value),
      notEqual(?Record, ?Record1),
      strConcat('Table -> ', ?Table, ' ; records -> (', ?Record, ', ', ?Record1, ')',
?Msg1),
      strConcat(?Msg1, '; value -> ', ?Value, ')', ?Msg) .
```

Rule 2 - Rule to avoid violation of primary key constraint

When a given table has a foreign key, then its value either is null or must exist in case it is a primary key. Our approach was to point into the record in the related table, i.e., where the key is primary. The Rule 3 ensures that a reference made in a foreign key exists in the table where the key is primary.

```
(err:ForeignKeyError err:violation ?Msg) <-
      (?Record ?Field ?Value),
      (?Record rdf:type rmm:Record),
      (?Field rmm:fk_references ?TablePK),
      (?TablePK rmm:primaryKey ?FieldPK),
      noValue(?Value rmm:recordOf ?TablePK),
      (?Record rmm:recordOf ?Table),
      strConcat('Table -> ', ?Table, ' ; record -> ', ?Record, ' ; value -> ', ?Value,
?Msg1),
      strConcat(?Msg1, ' ; TablePK -> ', ?TablePK, ?Msg) .
```

Rule 3 - Rule to avoid violation of foreign key constraint

In the ontological model of the relational constraints errors, the classes *err:MultipleFieldError*, *err:PrimaryKeyError* and *err:ForeignKeyError* are sub-classes of the class *err:Error*. So, after a SPARQL command that change the data of the repository, we look for all errors whose class is a sub-class of *err:Error*. In Figure 5 we give an example of a command that violates the primary key.

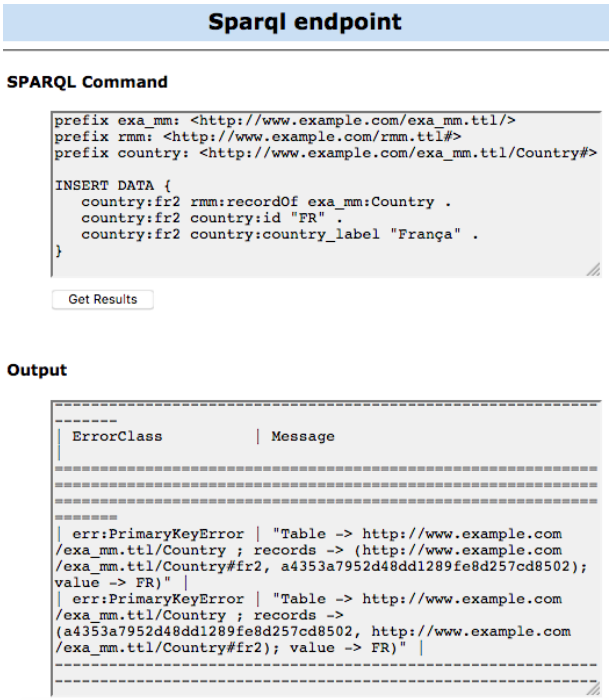

Fig. 5. Sparql endpoint integrity error

A. *Composite keys*

As previously mentioned, during the modeling process of the relational database we assumed that all keys, primary and foreign, are single keys. Composite keys were not in the scope of our project in this first approach. However, as future work we want to deal with composite keys. We can already introduce some solutions to deal with it: the primary key could be a field, when it is a single key, or a list of fields in case of composite keys. Considering this assumption, the Rule 4 defines that an error is returned whenever a primary composite key is violated.

```
(err:PrimaryKeyError err:violation ?Msg) <-
      (?Record rmm:recordOf ?Table),
      (?Record1 rmm:recordOf ?Table),
      notEqual(?Record, ?Record1),
      (?Table rmm:primaryKey ?ListFields),
      (err:PrimaryKeyError err:validate validatePK(?Record, ?Record1, ?ListFields)),
      strConcat('Table -> ', ?Table, ' ; records -> (', ?Record, ', ', ?Record1, ')',
?Msg) .

-> table(err:validate).

(err:PrimaryKeyError err:validate validatePK(?Record, ?Record1, rdf:nil)) <-
IsTrue().

(err:PrimaryKeyError err:validate validatePK(?Record, ?Record1, ?ListFields)) <-
      notEqual(?ListFields, rdf:nil),
      (?ListFields rdf:first ?Field),
      (?Record ?Field ?Value),
      (?Record1 ?Field ?Value),
      (?ListFields rdf:rest ?RestFields),
      (err:PrimaryKeyError err:validate validatePK(?Record, ?Record1, ?RestFields))
.
```

Rule 4 - Rule to avoid the violation of primary key constraint in composite keys

 ISBN: 978-1-61208-636-1

Considering our approach that a foreign key points to a record of the primary key, the problem of a composite key doesn't happen. However, as we want to foreseen in the future the two possibilities (point to a record or keep the value of the key), we need a rule with the same approach done to the Rule 4. Another solution is construct a specific rule to a table with composite keys. In this case we loose the generality of the rule but we can create the rule in a automatic manner from a template and from the database catalog.

V. Related work

A W3C Recommendation that describes R2RML, a language for expressing customized mappings from relational databases to RDF datasets is presented in [14]. In [15], it is presented a transformation of relational model to RDF model, a two-step approach where is extracted the semantics of relational model and then it is transformed in RDF models. In general, the approach followed in this work is done by us when we create the middle-tier, the representation of the meta-model in RDF. We also refer that we can create this middle-tier automatically by the information extracted from the database catalog. In [16], it is presented a tool that transforms relational data into OWL2 and performs data validation to report errors in the data. This validation is accomplished through rules. In [17], it is presented a set of mappings that transforms relational data and schema to Semantic Web models. OWL and SWRL are used to express relational constraints satisfaction or validation conditions. SPARQL query is used to verify these constraints. This work can be considered close to our work in the relational constraints approach since they used rules in backward mode to verify possible violation of relational constraints. However, our rules are generalized to all kind of data models and are based on meta-model while in their work the rules are designed for a specific data model. In [18], it is performed a practical evaluation of existing approaches in automatic generation of ontology from data models. The purpose of this work is the evaluation of the availability of existing approaches for automatic or semi-automatic generation of ontology from data models, the evaluation of the tools according to their operability and the evaluation of the resulting ontologies to assess their quality in supporting semantic interoperability. In [19], it is performed a survey of the works about the creation of an ontology from an existing database instance and the discovery of mappings between an existing database instance and an existing ontology. It is also presented the motivation, benefits, challenges and solutions. Some solutions that are presented in our work, could be developed using Shapes Constraint Language (SHACL) [20], which is a World Wide Web Consortium (W3C) specification for describing and validating RDF graphs data against a set of conditions. However, developing generalized SHACL rules for all models it is not possible, in the best of our knowledge. Furthermore, even for a specific data model it is not possible develop SHACL rules for constraints that includes composite fields.

VI. Conclusion

The main goal of this work was to produce a public repository of the information retrieved from the TREASURE project using the principles inherent to the Open Data philosophy. Initially, we modeled the information using a relational approach, having created a data model that would take into account the requirements of the project in terms of information. During the evolution process to an Open Data repository, and in order to avoid another analysis process, we decided to transform the relational information into RDF information. Afterwards, a Semantic Web approach was followed. The data repository is organized in a three layer schema. In a more generic layer, is the ontological model that represents the relational model and where its objects are characterized. In an intermediate layer is the meta-model of the database, where the data structure is described. Finally, in a more specific layer it is the data itself, which in our case, is the data generated by the TREASURE project. We created a system that allows the data repository to be automatically fed from a database, as well as a SPARQL endpoint enabling both updating and selecting data from the data repository. This SPARQL endpoint can be used either in a program or through a Web interface. Moreover we implemented a set of constraint mechanisms of the relational model by means of logical rules, ensuring the consistency of the data repository. Any change of data conducts the data repository from one *integrity state* to another *integrity state*. If any command does not respect the defined rules, a rollback of the operation is executed. Although this work was developed under TREASURE project, we have always been concerned with generalizing ontological models and the developed systems for any domain.

As future work we intend to implement more constraint mechanisms of the relational model for a more comprehensive use. In the way that our system is structured, this involves the development of more semantic rules. The semantic rules are kept in plain text file, therefore, we do not need any programming code change. As we refer before, composite keys in primary and foreign keys will be implemented. With regard to the TREASURE project, we intend to make an analysis using an OWL approach, instead of a relational approach, in order to have richer descriptions of the data. In addition, we intend to make mappings so that it can be framed as Linked Open Data, namely linking the main concepts to DBpedia.

References

[1] O. K. Foundation, Ed., *The Open Data Handbook / Das Open Data Handbuch*, 2012. [Online]. Available: http://opendatahandbook.org/

[2] T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web a new form of web content that is meaningful to computers will unleash a revolution of new possibilities," *Scientific American*, vol. 284, pp. 28–37, May 2001.

[3] T. Berners-Lee, "Linked-data design issues," W3C design issue document, June 2009. [Online]. Available: http://www.w3.org/DesignIssues/LinkedData.html

[4] C. Bizer, T. Heath, and T. Berners-Lee, "Linked data - the story so far." *International Journal on Semantic Web and Information Systems*, vol. 5, no. 3, pp. 1–22, March 2009.

 ISBN: 978-1-61208-636-1

[5] S. Auer et al., "Dbpedia: A nucleus for a web of open data," *The Semantic Web*, pp. 722–735, November 2008.
[6] R. Cyganiak, D. Wood, and M. Lanthaler Resource description framework (rdf): Concepts and abstract syntax. February 2014. [Online]. Available: https://www.w3.org/TR/rdf11-concepts/
[7] E. Prud'hommeaux and A. Seaborne, "SPARQL Query Language for RDF," W3C, Tech. Rep., March 2013. [Online]. Available: http://www.w3.org/TR/rdf-sparql-query/
[8] E. F. Codd, "A relational model of data for large shared data banks," *Commun. ACM*, vol. 13, no. 6, pp. 377–387, Jun. 1970. [Online]. Available: http://doi.acm.org/10.1145/362384.362685
[9] S. Bechhofer et al., Owl web ontology language reference. February 2004. [Online]. Available: http://www.w3.org/TR/owl-ref/.
[10] A. Seaborne. Jena, a Semantic Web Framework. November 2010. [Online]. Available: http://wiki.apache.org/incubator/JenaProposal
[11] P. Hayes and P. Patel-Schneider. Rdf semantics. 2014 February. [Online]. Available: https://www.w3.org/TR/rdf11-mt/
[12] Jena fuseki. 2014 [Online]. Available: https://jena.apache.org/documentation/serving_data/
[13] B. Parsia et al., Cautiously approaching swrl. Preprint submitted to Elsevier Science. 2005. [Online]. Available: http://www.mindswap.org/papers/CautiousSWRL.pdf
[14] S. Das, S. Sundara, and R. Cyganiak. R2rml: Rdb to rdf mapping language. September 2012. [Online]. Available: https://www.w3.org/TR/r2rml/
[15] Y. Lv and Z. Ma, "Transformation of relational model to RDF model," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Singapore*, October 2008, pp. 506–511. [Online]. Available: https://doi.org/10.1109/ICSMC.2008.4811327
[16] A. Yajai and G. Sriharee, "Eertoowl2: A tool for transforming RDB data to OWL2 for data validation," in *IEEE 24th International Conference on Tools with Artificial Intelligence, ICTAI 2012, Athens, Greece*, November 2012, pp. 970–975. [Online]. Available: https://doi.org/10.1109/ICTAI.2012.137
[17] X. Fan, P. Zhang, and J. Zhao, "Transformation of relational database schema to semantics web model," in *Second International Conference on Communication Systems, Networks and Applications*. IEEE, June 2010. [Online]. Available: https://doi.org/10.1109/iccsna.2010.5588750
[18] B. E. Idrissi, S. Baïna, and K. Baïna, "Automatic generation of ontology from data models: A practical evaluation of existing approaches," in *IEEE 7th International Conference on Research Challenges in Information Science, RCIS 2013, Paris, France*, May 2013, pp. 1–12. [Online]. Available: https://doi.org/10.1109/RCIS.2013.6577694
[19] D.-E. Spanos, P. Stavrou, and N. Mitrou, "Bringing relational databases into the semantic web: A survey," *Semant. web*, vol. 3, no. 2, pp. 169–209, April 2012. [Online]. Available: http://dx.doi.org/10.3233/SW-2011-0055
[20] H. Knublauch and D. Kontokostas. Shapes constraint language (shacl). W3C. July 2017. [Online]. Available: https://www.w3.org/TR/shacl/

 ISBN: 978-1-61208-636-1