Rights Enforcement and Licensing Understanding for RDF Stores Aggregating Open and Private Data Sets

Pierfrancesco Bellini, Lorenzo Bertocci, Filippo Betti, Paolo Nesi
Distributed Systems and Internet Technology, DISIT Lab, University of Florence, Italy
Tel 0039-055-2758517, fax: 0039-055-2758570, http://www.disit.dinfo.unifi.it, paolo.nesi@unifi.it

***Abstract*—Several applications are going to aggregate data on triple stores coming from different data sets and presenting different licenses. Semantic queries should provide only allowed triples, while most of the RDF stores have strong limitations in providing support for access control, licensing, rights enforcement and supporting the developers in providing tutoring information about what is possible and what is not. In this paper, a specific solution is proposed for supporting developers in understanding the licensing level of the requested triples, and the RDF stores in enforcing rights. The proposed solutions can be integrated into a range of different RDF stores for removing their limitations and assisting developers. The proposed solution has been developed and tested in the case of large smart city solution called Km4City and adopted in a number of projects: Sii-Mobility SCN, RESOLUTE H2020 and REPLICATE H2020.**

Keywords— RDF store, rights enforcement, smart city, data aggregation licensing

I. INTRODUCTION

Many smart city solutions are focused on data aggregation, reasoning and delivering of services via Smart City APIs. Smart City data can be coarsely aggregated by using solutions such as CKAN [1], OpenDataSoft [2], ArcGIS and OpenData [3]. These solutions are suitable for indexing data and metadata sources, providing support for browsing and querying data. In more advanced cases, they provide access to data sets as Linked Data (LD), Linked Open Data (LOD), from an RDF store endpoint [4], via SPARQL queries [5] exploiting some ontology. The access to RDF stores for data search and browsing can be performed by using visual browsers such as Linked Open graph, LOD of [6].

In most cases, the effectiveness of data services for Smart City is enabled by the availability of private data owned and managed by *City Operators* addressing specific domains: mobility, energy, business services (health, water), telecom, tourism, school and university, welfare, health, etc. City Operators are stakeholders also providing data and services with different granularities and size, and in some cases with under certain access conditions. Different "data granularity" implies different methods for collecting, licensing and providing access to those data. Real Time data are provided by city operators through some APIs such as Web Services or REST calls. The effective deploy of smart services for city users is very frequently viable only by exploiting the semantic integration of open data, private data and real time data coming from administrations and different city operators. This approach of data aggregation implies semantic processes of reconciliation and the adoption of unifying data models and ontologies as in Km4City [7]. Thus, aggregated data can be exploited to implement a large number of services and applications by structuring the Smart City Architecture and the corresponding Smart City APIs. For open data, as well as for private data, several different licensing models can be adopted [8], [9] enabling or preventing some business models, or simply their usage.

Services on smart city are typically based on a large amount of data sets (static and real-time) coming from different sources. Most of the data sets coming from governmental entities are licensed with some open license as CC [10], while data sets coming from city operators are frequently provided with some limitations. For example with: CC "no-commercial" constraint to limit their usage in business applications or even with no-distribution and/or no-derivative restriction: preventing the aggregator to distribute the data and/or providing derivative works from them. Therefore, a data aggregation service providing access to aggregated data with the aim of setting up mobile and web applications has to be very careful and provide support for managing the different licensing models by: **Aggregating data** coming from different operators and governmental entities by accepting input data sets with different licensing models, which in some cases may refer to the same entities. For example, the position of a bus stop is provided as open data, while the real time delays of busses are no-commercial and no-derivative.

Providing aggregated data and services via API with coherent licenses with respect to: (i) those of the original data sets, that cannot be violated [11], [12]; (ii) the different user kinds/profiles (e.g., city operators, citizen, tourist, student, researcher, policeman), (iii) the different application domains and contexts (cultural heritage information city promotion, commercial advertising, education, civil protection emergency advertising, car sharing, education of citizens, etc.); other constraints can be imposed as well.

Facilitating the work of developers in setting up their mobile and web applications via API according to the application domain and user kind. The same application may access to different data according to the application domain context and user role kind.

The license compatibility table adopted included license families of CC, ODC, OS Open Data, and takes into account duties and permissions. **Duties** are: Attribute, ShareAlike, and

Notice; and Permissions are: Derive, Distribute, Reproduce, and Commercialize. See for the adopted table the web page accessible from http://www.disit.org/6877. Relevant restrictions are present when derivative is not permitted, and when data set with share alike are tried to combine with n*on commercial share alike* licenses. In order to control the exploitation/access of content/data (and specifically on open data via RDF stores) a number of problems have to be addressed and solved by using different approaches as explained in the following paragraphs. The problems are related to data *service or right formalization*, *license formalization*, *grant computing*, and *right enforcement*. Additional problems can be related to RDF versioning [13].

In this paper, the focus is on query analysis for right enforcement since these aspects are not addressed in a suitable manner by RDF storage, which leaves the developer without assistance during the development of queries, which should respect the data set licenses.

The paper is structured as follow. Section II describes the related work. In Section III, the reference architecture of RDF stores with supportive tools for developing SPARQL queries respecting data licenses and understating limitations and constraints is presented. Section IV presents experimental results by providing an annotated example of query enrichment, control and aggregated license generation. Conclusions are drawn in Section V.

II. Related Work

Data services or rights formalization can include data access, read/write, change, derive, distribute, copy, adapt, etc. To this end, right ontologies or vocabularies could be used such as MPEG-21 RDD (Right data dictionary) [14], the ODRL vocabulary [15] or the Access Management Ontology [16]. The simplest solution to control right exploitation may consist of a conditional access system, CAS, based on the authentication via credentials and/or certificates.

License formalization may include the capability of describing complex conditions with temporal constraints, number of accesses, user localization, prices, etc., for example following the MPEG-21 standard [17], ODRL [15], XACML (an XML-based language used to specify policies on web resources in terms of polices) [https://wiki.oasis-open.org/xacml/], OASIS XAMCL [18], etc. Licenses may include permitted and/or forbidden functionalities and are substantially logical equations. Most of the above mentioned technological solutions have been developed for content and media, and may be applied to open data and private data. For example, Creative Commons licensing framework (a set of licenses) [10], [19] allows users to formalize the usage of data/content that the owner is sharing but applying some legally formalized duties, restrictions, permissions (e.g., no commercial use, attribution, no derivative, share alike). On the contrary, a number of specific open data licenses have been defined, some of them directly inherited from content and media. Other set of licenses are: ODC [20], Open Government License [21], Italian Open Data Licenses [22]. Most of the above licensing models are adopted for licensing open data sets, and when they are integrated and provided as Linked Data from stores the problems is reduced to the access at the RDF stored data in terms of triples [7].

The **grant computing** for the exploitation of rights can be performed on the basis of a combination of (i) ***user roles*** associated with actions to be performed on content (i.e., data sets, objects and/or services); (ii) ***license*** can be associated with a given user or user category, for content, condition and rights. Conditions may include: context, location, times, intervals, etc. The grant computing in accessing to multiple data sets having different licenses has to take into account license combination compatibility such as in [11], [12], and in their extended version provided by the authors on http://www.disit.org/6877. As regards of license formalization and grant computing in [23], a model for license formalization and access control of triples with context, related to time, location, credentials has been presented. The grant is computed by processing facts into an RDF store, including licensing models. The Access Management Ontology (AMO) [16] defines licenses as a role-based access control model and a reasoning model for processing rules/licenses via OWL. Similarly also [24] presented a license server for grant computing for RDF store where the access policies are expressed in a descriptive language formalizing the triple patterns to be licensed. In [25] a Relation Based Access Control model (RelBAC), providing a formal model of permissions based on description logics has been proposed. In [9] an access control model based on S3AC ontology (Social Semantic SPARQL Security for Access Control) has been presented. In [26], Privacy Preference Ontology (PPO) expressed access control policies to RDF in SPARQL.

The **right enforcement** can be regarded as the set of technical solutions to assure that certain rights adopted in licenses are not violated (respected) by using technological solutions as: encryption, certification and authentication, filtering, etc. For example, the right enforcement to control the exploitation of right to play a video is a set of technical solutions into the media player so as to guarantee that a user would be capable to play only the videos for which the user has received the grant authorization. Thus, the right enforcement is implemented in the tools used for content access, i.e., media player, browser, decoder, databases, mp3 player, etc. The above models integrating licenses formalization and grant computing collected policies/licenses into RDF triples and the grant computing (verification of access policies) is formalized via a ASK clause of SPARQL. The obtained results may be true of false to give at the user the rights on performing the corresponding action on the real RDF store containing data (that is conceptually different from that containing the license information).

The **enforcement** of **right** to access at triples on **RDF store** implies to guarantee that a given user can access only to triples for which he/she is authorized to access. When a SPARQL query for access is performed, as a result: several different triples may be provided, and may be identified exploiting some inference. The resulting triples may belong to different data sets, i.e., RDF graphs, presenting different licenses (i.e., the car sharing position coming from different providers). This means that the license computing based on triple patterns cannot be enough for fine-right enforcement. Therefore, according to the

above presented licensing computing models, the license is associated with triple (triple patterns) and the solutions proposed are focused on grant computing and not on the right enforcement, that is totally demanded to the RDF store implementation or on the filtering of triples on the basis of patterns, or on the rewriting of SPARQL queries so as to avoiding requesting triples for which the user is not authorized to access. Most of the RDF stores provide support for access control to the whole repository, such as Fuseky-Jena [27], GraphDB [28]. ORACLE RDF store provide support to perform access control to users at level of triple and model. Jena [27] provides API to write JAVA processes for filtering triples. Virtuoso [29] and Stardog [30] allow to formalize simple licenses (as read/write permissions) at level of data set (RDF graph), and associate them to users. This means that in the case of Virtuoso and Stardog an user performing a SPARQL query get back only triples for which is authorized without any explanation of filtered triples, and thus of potentially accessible data set with a different user profile and license. The RDF store supporting quadruples can be a solution, storing in the fourth element the ID of the dataset and recovering from its metadata the original license. This means that the results are dependent on data, and thus the programmer accessing to the RDF store via API and SPARQL queries have to test in advance if the query and API used will provide them data when a user category or role will be the requester. For developers it is very important to understand the involved RDF graphs (data sets) for each query, and the corresponding licenses to be acquired for exploiting the data of their interest, for example during the mobile APP development and query/data identification. The same problems arise when a developer make a query with wrong object, reference, subject patterns. The logical engine will not find any triple without any explanation.

III. Reference Architecture and Purpose

The reference architecture is referring to a phase in which the developer have to tune a SPARQL query on a complex RDF store in which a number of graphs (data sets) are stored, and where each of them may have a different license. Therefore, the triples resulting from a query may belong to multiple data sets. And thus the developer needs to understand if the data requested can be obtained for an application deployed for a given category of users. For example, if a final user App may access to the position of sensitive and critical services on the city or only to those that are generic and not critical. To this end, according to **Figure 1**, the developer can access to the SPARQL Query License Verification Tool to test each defined query for each specific user profile to see a reports (point (7)) describing which triples (belonging to a number of data sets) would be provided in response to the query. This approach solves the development problems identified in Virtuoso, Smartdog and in all the other RDF stores that do not provide a detailed list of needed licenses to obtain all possible data potentially resulting from a SPARQL query. To this end, the SPARQL query is analyzed (2) and enriched according to an algorithm described in Section III.A. The enriched query (addressing inference, inspecting all data sets, etc.) is applied to the RDF Store (3) to get as a result the list of involved Data Sets (4). This list is furtherly analyzed (by the License Verification Engine) to see if the User Kind involved can access to those data sets, thus producing the report at point (7). Both **SPARQL Analyzer** and **License Verification Engine** are described in Sections III.A and III.B, respectively.

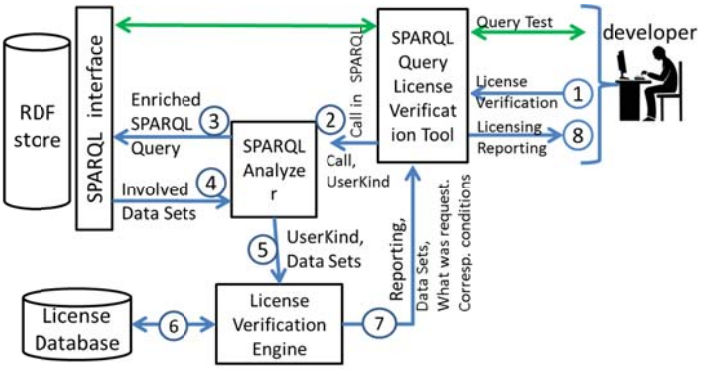

Figure 1 – Architecture of the SPARQL Query License Verification Tool

Once the set of queries have been finalized (according to Figure 1 process), they can be used into final applications if the RDF store provides support for right enforcement. In that case, Virtuoso and Stardog allow associating with graphs specific rights of read/write.

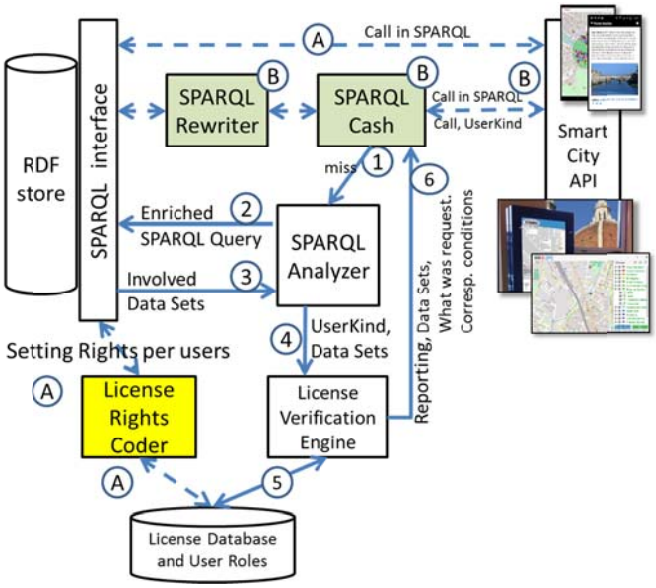

Figure 2 Reference architecture for SPARQL query usage with rights enforcement

According to **Figure 2**, they can be automatically set by using information collected into the License Database (line (A) of the figure). Thus, in that case, the SPARQL queries finalized by the developer, which is aware about the results according to **Figure 1** process, can be directly applied and triples will be filtered by the store. On the other hand, for the RDF stores that do not present a right enforcement support, solution (B) can be adopted. In case (B), the solution implies to process each query

for recovering in real time the datasets involved (3) and those that can be accessed case by case (path from (4) to (6)). The results for each [Call,UserKind] may be cashed to increase performances. The computing of the results (6) allows rewriting the SPARQL queries that violate licenses by using FROM clauses of SPARQL syntax to limit the access to certain datasets (graphs).

A. *SPARQL Analyzer Algorithm*

In this section, the SPARQL Analyzer Algorithm is described. It is used to extract the list of RDF graphs used in a given SPARQL Query. The solution adopted involves the analysis of the SPARQL query for its rewriting to produce an enriched SPARQL query that allows get the graphs used from the RDF store. The following example is reported for triple access, while different rights could be addressed similarly.

For example, in case in which the following query is used (to find all the service with name containing "CASA" and having a geo position):

```
SELECT ?name ?lat ?long WHERE {
  ?s a km4c:Service.
  ?s schema:name ?name.
  ?s geo:lat ?lat.
  ?s geo:long ?long.
  FILTER(contains(?name, "CASA"))
}
```

The above query is transformed/enriched into a query to find all the distinct graphs (data sets) where the different triples matching the constraints are stated:

```
SELECT DISTINCT ?g1 ?g2 ?g3 ?g4 WHERE {
    GRAPH ?g1 {?s a km4c:Service. }
    GRAPH ?g2 {?s schema:name ?name. }
    GRAPH ?g3 {?s geo:lat ?lat. }
    GRAPH ?g4 {?s geo:long ?long. }
    FILTER(contains(?name, "CASA"))
}
```

In general, two different cases have to be taken into account:

- *result graphs* which are those used to provide a value in the SELECT projection variables (e.g. g1, g2, g3 in the above example). The data provided from *result graphs* is provided to the user and thus need to be accessible in the results;
- *connection graphs* which are used to connect different pattern matching. These constraints do not involve a projection variable (e.g., g1 in the above example). Thus, c*onnection graphs* which are used in the computation of the result may not be accessible from the user and can be exploited to find other results (providing that the license for the usage of the involved data sets allows to exploit them).

The SPARQL Analyzer algorithm for query rewrite has to take into account the following cases, since the query may contain some SPARQL constructs of:

- UNION of triplets constraints;
- OPTIONAL constraints;
- property path expressions;
- GRAPH constraints;
- other sub-queries.

In cases of UNION and OPTIONAL constructs, the rewriting is performed recursively on the triplet constraints. In case a triplet constraint uses one or more property path expressions they are substituted with the equivalent form. For example, when a triplet constraint is multiple as: ?s p1/p2/p3 ?o, then it is rewritten / decomposed to:

```
GRAPH ?g1 {?s p1 ?x1}
GRAPH ?g2 {?x1 p2 ?x2}
GRAPH ?g3 {?x2 p3 ?o}
```

In case of unbounded property paths as with p1* and p1+ operators, it is not possible to rewrite the query with a finite set of triplet constraints. Therefore, (A) and (B) cases have to be addressed differently. In case (A), the construct is not a problem since the triples are not provided in any case by the RDF Store and the only limitation is on the analysis of the graphs involved. In case (B), the right enforcement is not provided by the RDF store. So that the above constructs cannot be accepted in the queries since they can cause the violation of licenses. Thus the developer is informed that the query rewriting has to be performed, for example making explicit the number of successive paths in the query. Thus permitting their rewriting, and re-conducting the issues to the other cases.

When a query presents a GRAPH constraint, it is rewritten as it is, thus, the graph variable is used as the other graph variables generated from the rewrite process. When a query contains one or more sub-queries, the sub-queries are analyzed to find the graphs used and graphs used by the upper query is the union of the graphs of all sub-queries plus other graphs used on the sub-queries results.

$$Graphs(Q) = \bigcup_{i=1}^{n} Graphs(Q.subQ_i) \cup G(Q)$$

Where *Graphs(Q)* is the set of graphs used by a query, $Q.subQ_i$ is the i-th sub-query of Q, and G(Q) is the set of graphs used by query Q not rewriting the sub-queries and adding the GRAPH constraint to the other triplets constraints.

To find the *result graphs* and the *connection graphs*, the graphs involving a variable in the projection list of the SELECT statement are considered. It should be noted that the use in the projection list of the graph content may not be directly in the projection list but indirectly via a BIND variable. This approach is used to compute *ResultGraph(Q)* that is the result graph of query Q.

B. *License Verification Engine*

Therefore, the graphs involved into a query can be extracted by executing the produced query generated by the SPAQRL Analyzer Algorithm. For each dataset a

corresponding license can be recovered from the License Database; making possible to compute the license needed for accessing or addressing the query result.

Let *P* be the set of Permissions that a license may allow or not for a dataset, *D* the set of Duties that the license may require or not to use a dataset, C the set of User Categories that will use the data. Please note that among ShareAlike (intended as in CC) is a Special Duty (SD) since it can be applied only in the cases in which the re-distribution is possible. Thus, a license for datasets in *G* is modeled with two functions:

$$allow: G \times C \times P \rightarrow \{true, false\}$$
$$require: G \times C \times D \rightarrow \{true, false\}$$

The *allow* function associates for a dataset in *G* an use category in C, and a permission in *P* a value true or false meaning that this permission is allowed or not on the dataset for an user of a specific category, and function *require* associates for a dataset, an user category and a duty in *D* true or false if the duty has to be required or not for a user on the specified category. The computing of permissions can be estimated by using one of the solutions proposed by the state of the art as [9], [26].

On the other hand, if the license models are limited to those for open data, the functions *allow()* and *require()* to **estimated bounds** on permission and duties can be defined on the results of query *Q* as follows:

$$allow(Q, c, p) = \bigwedge_{g \in ResultGraphs(Q)} allow(g, c, p)$$
$$require(Q, c, d) = \bigvee_{g \in ResultGraphs(Q)} require(g, c, d)$$

Where: *ResultGraphs(Q)* are the graphs/datasets used by query Q. Meaning that permission *p* is granted on results of query *Q* only if it is granted on all used dataset and duty *d* is required if at least one dataset requires it. Using this procedure and knowing the permissions associated with the application (e.g., a commercial application) the *License Verification Engine* can produce a report for the developer putting in evidence the datasets that can/cannot be used for each specific query.

In some cases, a global license cannot be provided, for example of Table 1, for the presence of protected content, or when data forbidding derivative (i.e., no-derivative), or when data sets with Share alike should be combined with data sets with no-commercial share alike data sets. In these cases, the developer is informed to reformulate the query with FROM construct, for the right enforcement (case B), the triples are not provided.

IV. Experimental Results

A SPARQL Analyzer and License Verifier Engine have been implemented according to the architecture of Figure 1, and it is available at http://log.disit.org/sparql-license-checker. The service has been validated using the queries from a smart city RDF store benchmark accessible from [31], [32]; which is based on queries similar to those used in Km4City API by http://servicemap.disit.org web service to discover services in Florence and in the Tuscany region.

For example, a query involving several datasets searches for all services distant less than 100m from a geographic coordinate (the Florence main train station). It gets the service name and an optional street address as well as its geographical coordinates:

```
SELECT DISTINCT ?name ?addr ?lat ?long ?dist WHERE {
  ?s a km4c:Service OPTION (INFERENCE "urn:ontology").
      ?s schema:name ?name.

  OPTIONAL{ ?s schema:streetAddress ?addr }
  { ?s geo:lat ?lat; geo:long ?long;
    geo:geometry ?geo
  } UNION {  ?s km4c:hasAccess [ geo:lat ?lat; geo:long ?long;
                                 geo:geometry ?geo ] }
  BIND(bif:st_distance(?geo, bif:st_point(11.2484,43.7765)) AS
?dist)
  FILTER(?dist<=0.1)
} ORDER BY ?dist
```

Thus the SPARQL query is transformed into:

```
SELECT DISTINCT ?g1 ?g2 ?g3 ?g4 ?g5 ?g6 ?g7 ?g8 ?g9 ?g10
WHERE {
    GRAPH ?g1{?s a km4c:Service OPTION (inference
  "urn:ontology"). }
    GRAPH ?g2{?s schema:name ?name. }
    {
      GRAPH ?g3{?s geo:lat ?lat. }
      GRAPH ?g4{?s geo:long ?long. }
      GRAPH ?g5{?s geo:geometry ?geo. }
    } UNION {
      GRAPH ?g6{?s km4c:hasAccess _:Nf7. }
      GRAPH ?g7{_:Nf7 geo:lat ?lat. }
      GRAPH ?g8{_:Nf7 geo:long ?long. }
      GRAPH ?g9{_:Nf7 geo:geometry ?geo. } }
  OPTIONAL {GRAPH ?g10{?s schema:streetAddress ?addr. }}
  BIND(bif:st_distance(?geo,bif:st_point(11.2484,43.7765)) AS
?dist)
  FILTER(?dist<=0.1) }
```

The tool, for each variable for GRPAH clause generate a table in which the graphs associated with the variable al listed. Thus, by using the above rewritten query on Km4City, 8 different datasets have been identified, and the final most restrictive license computed as reported in Table 1.

TABLE 1 – Licenses for the example described.

dataset/graph description	license	SD	Duties		permisions				user categories				
		sharealike	attribution	notice	derivative	commecialize	redistribute	reproduce	Citizen	Tourist	Police	Civil protection	Firefighters
DigitalLocation	CC-By-NC-SA	✔	✔	✔	✔	✘	✔	✔	✔	✔	✔	✔	✔
Eneergy Cabins	protected	✘	✘	✘	✘	✘	✘	✔	✘	✘	✔	✔	✔
Commercial firms	CC-By	✘	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Graph street	CC-By	✘	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Services on the city	CC-By-NC	✘	✔	✔	✔	✘	✔	✔	✔	✔	✔	✔	✔
Renting bikes	CC-By	✘	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Taxi	CC-By	✘	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Enogastronomy	CC-By	✘	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔

Thus the developer may verify if the resulting license is suitable for the application under development or it could be better to restrict the access at only some data sets. For example,

the resulting query can provide all triples for Police personnel which are also informed of the data kind and attribution; while Citizens and Tourists cannot access to those data and a FROM clause has to be added for scenario of case (B) of Figure 2. And, commercial applications are not going to access and visualize at data sets regarding "Energy cabins". The developer is assisted by the tool providing full information of what is viable and what cannot be done according to the different application and user category. Thus the developers is also assisted in producing coherent queries receiving suggestion on how the query cam be modified by using clauses FROM and FROM NAMED.

V. Conclusions

RDF store based applications aggregating several data sets with different licenses may lead to certain complexity in data access for developers. Most of the RDF stores present strong limitations in providing access control, licensing, rights enforcement and thus on supporting the developers in getting information about what is possible and what is not. For example, those supporting access control, limit the number of triples accessible without informing the developer that the results triple set is empty only for him and not in general. In this paper, a specific solution has been proposed for supporting (i) developers in understanding the licensing level of performed queries, and (ii) RDF stores in enforcing rights. The proposed solutions can be integrated into a range of different RDF stores (in different manner for different RDF store kinds as described in the paper); thus removing their limitations and assisting developers. The proposed solution has been developed and tested in the case of large smart city solution called Km4City and adopted in a number of projects: Sii-Mobility SCN, RESOLUTE H2020 and REPLICATE H2020.

Acknowledgment

This works has been developed in the context of Sii-Mobility Smart City National project (http://www.sii-mobility.org), for RESOLUTE H2020, and for REPLICATE H2020 European Commission Projects.

References

[1] CKAN: http://ckan.org

[2] OpenDataSoft: https://www.opendatasoft.com/

[3] ArcGIS OpenData: http://opendata.arcgis.com/

[4] RDF https://www.w3.org/RDF/

[5] W3C Consortium, "SPARQL 1.1 Query Language", W3C Recommendation, 21 March: https://www.w3.org/TR/rdf-sparql-query/

[6] P. Bellini, P. Nesi, A. Venturi, "Linked Open Graph: browsing multiple SPARQL entry points to build your own LOD views", http://log.disit.org Int. Journ. of Visual Language and Computing, Elsevier, 2014, DOI: http://dx.doi.org/10.1016/j.jvlc.2014.10.003

[7] P. Bellini, M. Benigni, R. Billero, P. Nesi and N. Rauch, "Km4City Ontology Building vs Data Harvesting and Cleaning for Smart-city Services", Int. Journal of Visual Language and Computing, Elsevier, 2014, http://dx.doi.org/10.1016/j.jvlc.2014.10.023

[8] Korn, N., Oppenheim, C.. "Licensing Open Data: A Practical Guide", Discvery on line, June 2011 http://discovery.ac.uk/files/pdf/Licensing_Open_Data_A_Practical_Guide.pdf

[9] Villata S., Delaforge N., Gandon F., Gyrard A., "An Access Control Model for Linked Data", OTM Workshops, Oct 2011, Heraklion, Greece. Springer, 7046, pp.454-463, 2011, LNCS.

[10] Creative Commons, http://creativecommons.org

[11] CC compatibility http://wiki.creativecommons.org/Wiki/cc_license_compatibility

[12] "The ODI license compatibility", https://github.com/theodi/open-data-licensing/blob/master/guides/licence-compatibility.md

[13] P. Bellini, I. Bruno, P. Nesi, N. Rauch, "Graph Databases Methodology and Tool Supporting Index/Store Versioning", publication on JVLC, Journal of Visual Languages and Computing, Elsevier, 2015 doi:10.1016/j.jvlc.2015.10.018

[14] Xin Wang, Thomas DeMartini, Barney Wragg, M. Paramasivam and Chris Barlas, "The MPEG-21 rights expression language and rights data dictionary", IEEE Transactions on Multimedia, vol. 7, no. 3, pp. 408–417, June 2005.

[15] R. Iannella and S. Guth, "ODRL version 2.0 common vocabulary", Specification, W3C ODRL Community Group, 04 2012. http://www.w3.org/community/odrl/two/vocab

[16] Buffa, M., Faron-Zucker, C., Kolomoyskaya, A., "Gestion semantique des droits d'acces au contenu: l'ontologie AMO", In: Yahia, S.B., Petit, J.M. (eds.) EGC. Revue des Nouvelles Technologies de l'Inf, vol. RNTI-E-19, pp. 471-482, 2010.

[17] MPEG21-REL, "MPEG-21 Part 5: Rights Expression Language", ISO/IEC 21000-5:2004.

[18] T. Moses, "Privacy policy profile of XACML v2.0," Oasis standard, OASIS", 02 2005. http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-privacy_profile-spec-os.pdf

[19] [Abelson et al., 2008] H. Abelson, et al. "ccREL: the Creative Commons Rights ExpressionLanguage," wiki.creativecommons.org/images/d/d6/Ccrel-1.0.pdf, 2008.

[20] [Open Data Commons, 2009] Open Data Commons. "Legal tools for Open Data", 2013, Retrieved from http://opendatacommons.org/licenses/

[21] "Open Government Licence, 2014" , Retrieved from http://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/

[22] Italian Open Data Licenses https://data.gov.uk/

[23] Abel, F., Coi, J.L.D., Henze, N., Koesling, A.W., Krause, D., Olmedilla, D., "Enabling advanced and context-dependent access control in rdf stores", Proc. of the 6th Int. Semantic Web Conf. (ISWC-2007), LNCS 4825. pp.1-14, 2007.

[24] Muhleisen, H., Kost, M., Freytag, J.C., "SWRL-based Access Policies for Linked Data" , Proc. of the 2nd Workshop on Trust and Privacy on the Social and Semantic Web (SPOT-2010), 2010.

[25] Giunchiglia, F., Zhang, R., Crispo, B. "Ontology driven community access control", Proc. of the 1st Workshop on Trust and Privacy on the Social and Semantic Web, 2009.

[26] Sacco, O., Passant, A., "A Privacy Preference Ontology (PPO) for Linked Data" , Proc. of the 4th Workshop about Linked Data on the Web (2011)

[27] Jena, https://jena.apache.org

[28] GraphDB, http://ontotext.com/products/ontotext-graphdb/

[29] O. Erling and I. Mikhailov. "Virtuoso: RDF Support in a Native RDBMS". Semantic Web Information Management, pp. 501-519, Springer, 2009.

[30] Startdog, http://stardog.com/

[31] DISIT Smart City RDF BenchMark, http://www.disit.org/smartcityrdfbenchmark

[32] P. Bellini, P. Nesi and G. Pantaleo, "Benchmarking RDF Stores for Smart City Services," *2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)*, Chengdu, 2015, pp. 46-49. doi: 10.1109/SmartCity.2015.45