Contextually-Enriched Querying of Integrated Data Sources

Giacomo Cavallo [#1], Francesco Di Mauro [#2], Paolo Pasteris [#3], Maria Luisa Sapino [*4], K. Selçuk Candan [#5]

[#] *Computer Science Department, University of Torino; Torino- Italy*
[1] `cavallo@di.unito.it`; [2] `dimauro@di.unito.it`; [3] `pasteris@di.unito.it`; [4] `mlsapino@di.unito.it`
[*] *Arizona State University; Tempe-USA*
[5] `candan@asu.edu`

***Abstract*—In many applications, users face the need to integrate multiple informations sources to solve their tasks. While most information sources can be seen as stable, as they contain factual, objective knowledge (which is the case for most relational databases), there can be personalized contextual knowledge which enriches the available factual data and contributes to their interpretation, *in the context of the knowledge of the user who queries the system*. In this paper, we present *CrowdSourced Semantic Enrichment (CroSSE)*, a social knowledge platform supporting semantic enrichment and integrated services (such as content personalization, preview, and social recommendations) within the context of scientific investigations. Semantic enrichment is especially useful in applications where data (schema and facts) evolve faster than the database itself and, while essential to the operation of the enterprise, the database lacks the flexibility to (a) prevent going stale or (b) capture user-provided and/or crowdsources data and knowledge for more effective decision support. More specifically, in this paper, we focus on the SESQL language and the supporting system architecture, which enables users to (a) enrich a databank with semantic tagging information and (b) leverage contextualised queries to the databank for enabling contextualised data analysis.**

I. Introduction

In this paper, we present *Crowd Sourced Semantic Enrichment (CroSSE)*, a social knowledge platform and integrated services supporting semantic enrichment and content personalization within the context of scientific investigations. While our ultimate goal is that the CroSSE platform will be available for scientists of different backgrounds, our work is within the context of the *SmartGround - SMART data collection and inteGRation platform to enhance availability and accessibility of data and infOrmation in the EU territory on SecoNDary Raw Materials*[1] - European project, which aims to develop a databank platform, providing access to a broad spectrum of data relevant to decision making in the context of waste collection and management.

A. Use Case: SmartGround

Developed countries produce increasing amounts of wastes which, if not properly recycled, have a significant impact on pollution and sustainability of the economic production process. Both urban and mining wastes contain, on the other

[1]Partially supported by EU Horizon 2020 Research and Innovation Programme under Grant Agreement No 641988 (http://www.smart-ground.eu/index.php).

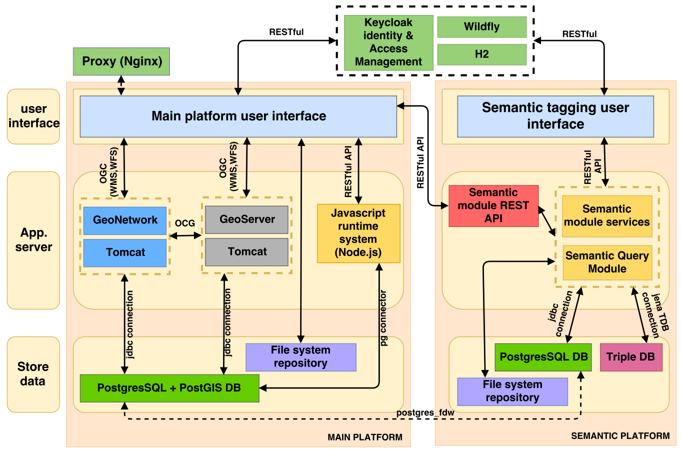


Fig. 1. The SmartGround databank and semantic enrichment

hand, materials that can still be useful. The challenge for the scientific community is to define rational waste management practices which enable the use of that part of waste which can still be useful and should be recovered from industrial, mining, and municipal landfills [2]. A key aspect to encourage reuse of raw materials stored in the many existing landfills is the availability of data and information about their existence: “What is available where?” “Is there an advantage of acquiring a given material from a specific landfill instead of getting it from another one?” “How is the quality of the material (chemical pureness) vary over landfills?” “Are there other materials of interest in the candidate landfills?”

The SmartGround platform integrates existing information from national and international databanks (national agencies, public bodies data bases, European statistics) and provides the data to all types of researchers and decision makers (city level, state level, European level) to enable perform hypothetical reasoning, possibly within different contexts, representing, for example, the rules and constraints enforced in different countries. As currently implemented, the SmartGround platform (Figure 1) is comprised of the following modules:

Main Platform collects the data on the landfills in a database

[2]In Europe, there are from 150.000 to 500.000 very variable landfills. In 2008, 49% of the almost 3 billion tons of total waste generated in the EU-27 was disposed in dedicated landfills, thus the secondary raw material potential of various landfills is very high.

and provides the users with the tools to explore and update its content.

Semantic Platform collects and manages the ontological information provided by the users, offering the tools to perform enriched queries on the main platform database.

Integration the integration between the two platforms is managed by means of RESTful APIs, while the communication between data sources relies on the `postgres_fdw` extension.

While the advantages of bringing scientific data into a uniform integrated platform is clear, in this paper we note that this only solves part of the problem. We see that, in order to make the databank truly useful for a diverse group of researchers, analysts, and decision makers, the data needs to be complemented by one or more knowledge-bases, that describe the contexts in which the data is queried and explored. Moreover, given that it is not realistic to assume that tailor-made knowledge-bases will exist for all relevant contexts, it is critical that such knowledge (ontologies as well as assumptions and hypotheses) can be extended individually and collaboratively by the users of the system (Figure 2).

The Knowledge Base at the core of Smartground consists of rules and ontologies that formally describe the relationships among key concepts at different levels of abstraction. Such knowledge potentially includes observational concepts related to context, sampling, classification, and measurement. Ontological knowledge may represent identity or hierarchy information or can define an accepted standard or scientist-specific index (e.g., relating region names to their geographical coordinates). Importantly, users are allowed to enter concepts into the knowledge base and to relate them to known concepts or concepts declared by other researchers.

B. Vision: Crowdsourced Semantic Enrichment and Context-Aware Exploration and Analysis

Once the data are properly integrated and semantically-enriched, the system can provide rich integrated and contextualized data access, exploration, and analysis services to its users. We see that the current tools and systems have the following shortcomings:

a) Lack of personal context-aware search and exploration: For effective data and resource discovery, searches need to be highly precise and informed of user's search context. Let us consider two users who are searching for materials having to do with the concept "pollution". For the sake of the example, let us assume that the first user is a researcher, who is interested in finding materials related to "pollution" within a scientific context. Let us further assume that the second user is a city planner, who, naturally, interprets the term within its urban planning context. For these two users, the concepts in the database and documents in the collection as well as their respective queries carry different meanings and connotations. When the contexts are taken into account, the search should rank and organize results differently for these two users.

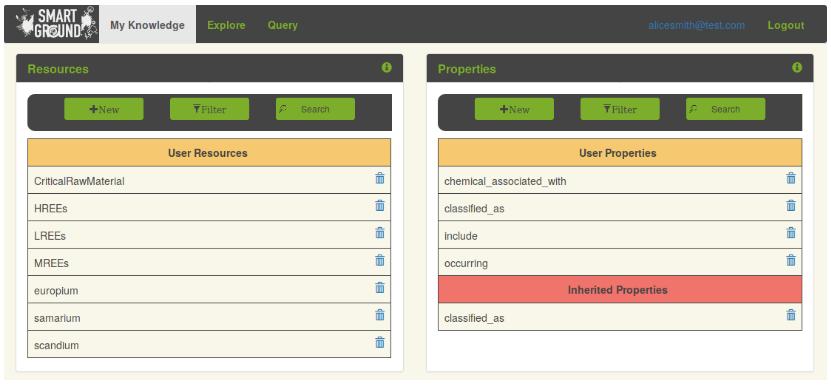
b) Lack of peer-networking and recommendations: Consider a researcher who is preparing a report on "pollutant elements" with particular emphasis on "asbestos". This researcher will need to explore and weed out many irrelevant content before identifying a suitable set of materials. This is because content-based searches are not sufficiently precise. Context-aware exploration can help this researcher in two complementary ways: (a) peer recommendation: Based on this researcher's interactions with the system (including her past queries, exploration emphasis she has declared, and the concepts she has explored), the system can help the researcher locate other individuals (or peers) with similar interests or who have similar goals; (b) data recommendations based on peer networks: the system can recommend the researcher resources that were explored and used by others within similar contexts.

c) Lack of content previews, exploration support, and context-aware extension: When searching for data, simple query interfaces may not be effective. Instead, the system needs to provide snippets, summaries, and other previews to help the user explore resources effectively. Consider a researcher who is searching for data and documents on "mining secondary materials". When this user is provided with a long list of results to her query, she faces a number of challenges: First of all, each result in the list is likely to require further investigation as there is a chance that only a fraction of the results be relevant to the user's context. Secondly, long list of results increase the overhead on this user while sifting through the list. Instead the system should provide (a) context-aware ranking, (b) snippet extraction, (c) key concept highlighting, and (d) context-aware knowledge extension.

To support resource discovery within rich information spaces, we develop services for context-driven filtering and enrichment for user requests and explorations. We note that accessing scientific data effectively requires a proper understanding of the personal activity context, context-aware resource discovery, and peer-network driven data and knowledge sharing and collaborative recommendations. In particular, building on our prior scientific and educational open crowd platforms, Hive [1] and MiNC [2], we are developing a *Crowd Sourced Semantic Enrichment (CroSSE)* platform, with the goal of supporting data and peer services, including (a) understanding the personal activity context through analysis of access patterns and of user's own annotations, (b) context-aware data and resource discovery, including search, presentation, exploration, and knowledge extension support within the knowledge structure (e.g. concept maps), and (c) peer discovery, peer-network management, and peer driven resource and knowledge sharing and collaborative data analytics.

C. Focus of this Paper: Semantic Tagging and Semantic Query Enrichment

In order to support the above, the system enables users to enrich the information stored in the databank with their own knowledge, to personalise queries and reasoning tasks. For example the director of a specific laboratory might be interested in combining the information about the analysis



User Resources

- CriticalRawMaterial
- HREEs
- LREEs
- MREEs
- europium
- samarium
- scandium

User Properties

- chemical_associated_with
- classified_as
- include
- occurring

Inherited Properties

- classified_as

Fig. 2. The SmartGround Semantic Enrichment Platform

on a landfill (information stored in the database), with other information relevant to her but not stored in the database (for example, the data and role in the laboratory of the person who has signed the analysis report), and might be interested in *querying the database in the context of her personal knowledge*. To support such contextualised querying process, we are developing a semantic tagging module (see Section III-A) in which users can insert their own knowledge (and possibly share knowledge already inserted and made available by other users) in the form of RDF statements [3], and a query engine which combines SPARQL queries on each user's knowledge base and SQL queries for the data stored in the relational databank. In [4], we discussed the need for semantic tagging and contextualised queries to enable crowdsourced participation to decision making processes, and we provided an overview of the semantics of the enriched queries we were targeting. In this paper, we introduce the syntax of our contextually enriched query language SESQL, and focus on our existing architecture which enables SESQL queries.

D. Organization of the Paper

The paper is organised as follows: In Section II we discuss related work. Section III introduces the semantic tagging component of our system, which enables users to insert their own contextual annotations. Section IV presents syntax and semantics of the SESQL, showing how it allows contextualised queries. Section V concludes the paper and discusses future work.

II. Related Works

In many domains, especially in the research communities, users feel the need to share information and knowledge, to enhance their collaboration. Great opportunities in this direction can potentially be offered by dedicated and thematic social networks and crowdsourcing platforms, such as MiNC [2] and LabBook [5]. In this section, we provide an overview of the related work and enabling technologies.

Ontology Management and CrowdSourcing: [6] and [7], focus on crowdsourcing ontology verification and engineering, in the biomedical domain. They apply ontological verification to large biomedical ontologies, in which the class hierarchy not only is the core structure, but is the only semantic relationship created by ontology developers. Using a crowdsourcing method for ontology verification (in which workers answer computer-generated questions based on ontology axioms) the hierarchy verification is subdivided in micro-tasks and the results are measured. So crowd-workers can collaborate with the domain experts, improving the quality of the enriched ontology, while reputation and altruism are forms of incentive models.

In [8] the problem of ontology based information reuse is oriented to the realization of knowledge-based digital ecosystems. In particular, the authors present techniques based on linguistic analysis that, starting from the vocabularies contained in each source ontology and relating them with the initial (or proto) ontology, can facilitate the process of ontology construction, automating the selection and reuse of existing data models. [9] presents the NeOn methodology for ontology engineering. Without a rigid framework, this approach considers the ontological development as the construction of networks of ontologies, in which resources may be managed by people in different organizations.

Data Integration: In general, there are three types of information-integration systems. In source-centric systems, the sources are defined in terms of the global schema and are referred to as local-as-view, or LAV, systems (Information Manifold [10], Emerac [11]). The LAV approach, while flexible, assumes a consistent integrated view. An alternative approach is to define the global schema in terms of the sources. This is called global-as-view, or GAV (HERMES [12], SIMS [13], TSIMMIS [14]), and WEBBASE [15], [16], by Davulcu). In GAV systems, whenever a source changes or is added, the global schema needs to be modified. The third class is a hybrid referred to as a GLAV system [17]. Orchestra [18] and FICSR [19] are systems that focus on managing disagreements that arise (at both schema and instance levels) during data sharing. In Orchestra, each participant has a (locally) consistent database instance, containing the set of tuples (possibly originated from other participants) that it accepts. FICSR creates a data structure that captures all interpretations of a conflicting database and can provide different views, ranked with the user's individual assumptions and preferences to different users. [20] offers a survey of different DB integration techniques. Among them, mediated query systems enable a uniform data access solution by providing a single point for read-only query of heterogeneous data sources: the global query processor sends sub-query to local/distributed sources and manages the reconciliation of the results.

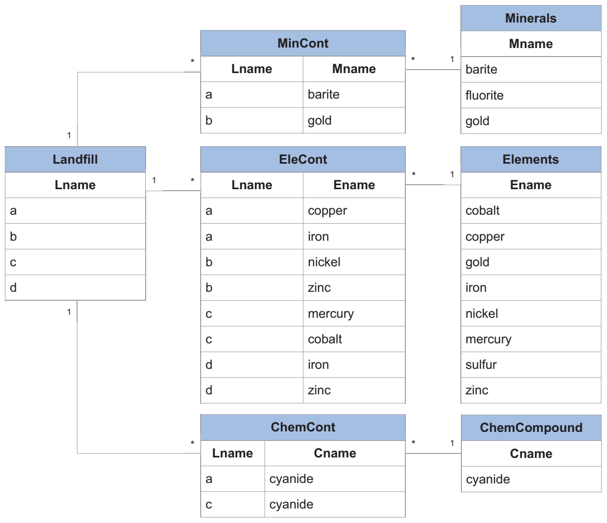
Ontology Driven Query Formulation: The Ontology-Based Data Access (ODBA) is the focus of [21], devoted to the understanding how reasoning on the ontology affects the query answering process. ODBA can be implemented as a three level architecture consisting of the ontology, the data sources, and the mapping between them. Answers to a query are not only a data structure that collects (in terms of data integration) the various sources, but also include semantically rich descriptions of the relevant concepts in the domain of interest. [22] deals with ontology-driven query formulation, in which the intensional description of a relational database is mapped to a OWL-DL description, the language in which the domain experts express their specific knowledge. On this common OWL-DL formalization, the user may formulate ontological queries that are then translated into the corresponding relational SQL statements.

Available ontologies can be used in web site management and integration scenarios; in particular, [23] describes a SEmantic portAL (SEAL) which presents a three-layer architecture encompassing: (a) *heterogeneous data sources* (DB, XML, HTML); (b) a *wrapper* that aggregates the sources in a common data model; (c) *integration modules* (and specific mediators for the dynamic case) able to reconcile the data sources. The ontology can offer support to user query targeted to different sources, and the intensive use of schema information can facilitate the activity of *integration*, *selection* and *presentation* requested by a web tool that is based on a semantic conceptual model. The central aspect of this family of semantic portals (and other similar system, like SmartGround) is the help offered to a community of users, each one *contributing* to the global knowledge base while also *consuming* the common enriched knowledge.

III. Semantic Tagging and Enrichment

Regarding the information handled in our platform, we distinguish between (1) *data*, which are stored in the database and represent *factual information* shared by the different partner institutions and taken as certain knowledge by all the users, and (2) *personal, contextual knowledge*, which reflects the users' interpretation of the data, or the available contextual meta-knowledge that the users might want to use in combination with the stored data. While data are commonly accepted as true, personal knowledge reflects the users' individual experience and know-how.

The semantic tagging module enables annotation of existing data with user-provided and crowd-sourced knowledge and integration of different information sources, with the help of one or more domain experts and/or data users. Thanks to this module, data stored in the relational data back-end can be



Landfill
Lname
a
b
c
d

MinCont	
Lname	Mname
a	barite
b	gold

Minerals
Mname
barite
fluorite
gold

EleCont	
Lname	Ename
a	copper
a	iron
b	nickel
b	zinc
c	mercury
c	cobalt
d	iron
d	zinc

Elements
Ename
cobalt
copper
gold
iron
nickel
mercury
sulfur
zinc

ChemCont	
Lname	Cname
a	cyanide
c	cyanide

ChemCompound
Cname
cyanide

Fig. 3. Sample fragment from the SmartGround database

fused (a) with ontological knowledge asserting properties of the concepts mentioned in the database relations, (b) information of interest to the user, that can complement the database content, such as files with users notes, pictures, information about the data provenance, etc., and (c) user-provided and crowd-sourced knowledge bases to provide rich, open user-enriched data.

A. Semantic Tagging

The semantic tagging interface provides users the option of inserting their own knowledge or accepting as their own (part of) the contextual knowledge already inserted by other users. The inserted (or acquired from other users' statements) knowledge *enriches*, i.e, *contextualises* and *extends* the information already available in the database and the conclusions that can be drawn from it.

Notice that there is no centralized control on the correctness and/or consistency of the crowdsourced knowledge (e.g. conflicting information regarding different concepts). This is because we aim to give each user the freedom to express her own beliefs, assumptions, or hypotheses about the domain, and query the database within the context of such additional information not readily available in the database.

Example 3.1: The schema of our SmartGround database includes tables to collect information about the elements, minerals and/or chemical compounds that can be found in different mine landfills (Figure 3). While those waste items are described in terms of their chemical properties and of the available amounts in the various considered landfills, the database schema does not capture, for example, information about the different labs which conducted the analyses, and their internal hierarchical organizations. Similarly, information about the fact that some elements (maybe if co-located with some others) might be considered as pollutant might depend on local (to the states or the regions) rules and regulations, fixing thresholds for acceptable amounts of specific elements in space units. Once suitably tagged, the users can query the database and obtain information about the existence of pollutant elements (extracted from the database) in some areas, according to the analysis conducted by some specific lab (information derived from the semantic knowledge base). ♢ As we see in the above example, semantic tagging enables users to extend the knowledge base by annotating data with contextually relevant information. and have personalized experiences during the exploration of the database content. In particular, we identify several annotation scenarios:

- *Integrated annotation scenario:* Users who are interacting with the platform have the possibility of highlighting a concept of interest to them, among the ones visualized by the platform, and annotate them. Annotations can be of different nature: (i) they can express properties about the concepts, to be inserted in the knowledge base and possibly used at (enriched) query time; (ii) they can be general notes the user is interested in storing for future use, for exploration purposes only.
- *Independent annotation scenario:* Users can directly access the semantic module and state their properties to be inserted in the knowledge base. The system offers two interchangeable input interfaces: users can either directly insert $<\texttt{subject}, \texttt{property}, \texttt{object}>$ triplets, or leverage a graph-based visualization tool which supports knowledge insertion in a more user friendly way.
- *Crowdsources annotation scenario:* Being a platform aiming at enhancing information sharing and collaboration, users' annotations are public (i.e, visible to other registered users). Users can explore the knowledge made available by their peers, and choose to import (part of) it in their own knowledge base. It is the personal knowledge base that will constitute the context in which a user's query will be evaluated.

A major difference between the integrated and the independent annotation scenarios is that while integrated annotation forces the subject of the annotation triplets being inserted to be a concept extracted from the original data source, independent annotation gives the users the freedom to insert any additional knowledge into the knowledge base, for future exploration and use. More specifically, the users can explore the existing annotations, both their own ones and those made available by other users and use the statements included in the knowledge base to enrich the results, by combining data extracted from the database and knowledge extracted from the knowledge base. In the next subsection, we provide an overview of the RDF based representation of the semantic enrichment and in Section IV, we describe the semantically-enriched query language, SESQL.

B. RDF Representation of Contextual Information

Annotations expressing contextual information are expressed as RDF statements, $< subject, property, object >$ triples, stating that the concepts associated to $subject$ and $object$ are related through the relationship $property$. $< Mercury, is-a, element >$is an example of such a triple, stating that Mercury belongs to the class of elements. In our platform, to create and query the triple store database we used the Jena library, written in Java [24]. In CroSSE, RDF statements may or may not be shared. Each statement is annotated with information about its "source", the users who inserted it into the system and the users who have chosen to accept this statement as theirs. Figure 4 illustrates the RDF schema we devised in order to allow the storage and querying of the contextual metadata (i.e., the RDF triples) while differentiating the ones defined by different users (associating metadata defined by different users to their authors/owners).

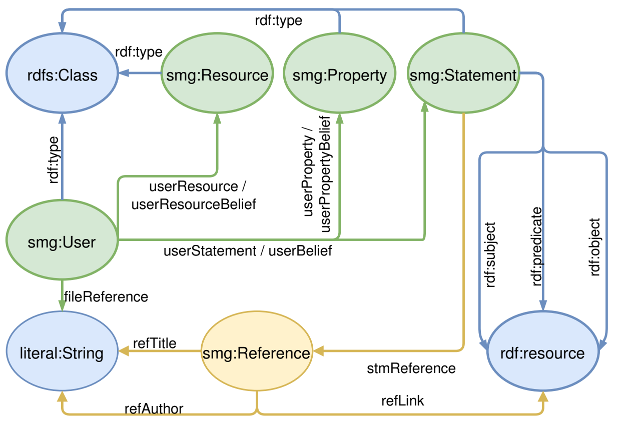


Fig. 4. RDF triple store for semantic enrichment

The basic idea behind query enrichment consists in exploiting the ontology in order to identify semantic patterns not directly recognizable in the knowledge contained in the database, thus providing the user a more informative result set, which contains data derived by the common/shared knowledge along with the one that the user herself put in the system. The RDF formalism allows, through the SPARQL language, to navigate the ontology by following the edges of the graph, thus allowing the definition of complex paths connecting various concepts. In particular, the user can choose one or more attributes (from a relation involved in the query or from the query result set) and run a SPARQL query for any of its values, to obtain a set of replacements (which may or may not contain the initial value according to the user preferences) to enrich the scope of the query or the result set itself.

In the next section we will see how such query enrichment can be expressed through the SESQL language.

IV. Semantically Enriched Querying

In this section, we introduce the *Semanticaly Enriched SQL (SESQL)* language supporting contextually-relevant query processing. SESQL queries the relational database in the context of the ontological information. In addition to all SQL statements, SESQL offers statements to specify (i) the desired type of enrichment (either addition/removal of attributes in/from the query result table or use of ontological knowledge in the query filtering condition); (ii) the attributes (from the relational schema) to be enriched; and (iii) the ontological properties on which the enrichment has to be based.

Terminal Symbols:

```
T = {ENR, SEXS, BSEXS, SREXP, BSREXP, RCEXP, RVEXP,
AS, STRING, OPAR, CPAR, VIR}
```

Non-terminal Symbols:

```
VT = {body, exp, wexp, seexp, bseexp, srexp, bsrexp,
rcexp, rvexp, map, property, cond, name, concept, s}
(here s is the start symbol)
```

Production Rules:

```
s → ENRICH body
body→ exp body ‖ ex ‖ wexp
xp → seexp ‖ bseexp ‖ srexp ‖ bsrexp
wexp → rcexp ‖ rvexp
seexp → SCHEMAEXTENSION(map, property)
bseexp → BOOLSCHEMAEXTENSION(map, property,
concept)
srexp → SCHEMAREPLACEMENT(map, property)
bsrexp → BOOLSCHEMAREPLACEMENT(map, property,
concept)
rcexp → REPLACECONSTANT(map, property)
rvexp → REPLACEVARIABLE(map, property)
map → STRING
cond → STRING
property → STRING
concept → STRING
name → STRING
```

Fig. 5. BNF Grammar for SESQL

A. SESQL Query Language

SESQL is a query language which extends SQL to enable users to query a relational database and enrich the returned results with contextual information modelled as ontological knowledge. A SESQL query (Figure 5) consists of two parts, the first one corresponding to a traditional SQL query, and the second one specifying the type of semantic enrichment the user is interested in. More specifically, a SESQL query follows the pattern

```
SELECT ...
FROM ...
WHERE ...
ENRICH
SCHEMA EXTENSION(...)
SCHEMA REPLACEMENT(...)
```

where the clause `ENRICH` plays the role of the separator between the two query components. The extended SQL syntax describes the desired enrichment by specifying (i) the relational attribute the user wants to enrich (thus possibly changing the set of values returned by the query for that attribute) and (ii) which contextual, ontological knowledge the user wants to leverage for the enrichment process.

We introduce six types of enrichments, four of them affecting the `SELECT clause`, and thus the schema of the returned result, with possible addition/replacement of attributes, and the remaining two affecting the evaluation of the `WHERE clause`.

1) Schema Extension: `SCHEMA EXTENSION(attr, prop)` is the clause that enables the user to add an attribute (not coming from the database schema) to the relation returned by the SQL part of the query, and the ontological properties based on which the values for the new attribute will be computed.

Informally, at the query time the enrichment is obtained by (i) creating a SPARQL query to find subject and object of all the RDF triples containing the specified property `prop`; (ii) comparing the values of the attribute `attr`, an attribute occurring in the SELECT clause of the SQL query, with the subjects of the returned RDF triples. In case of match, the corresponding objects are returned as the values for the new column of the SESQL query result.

Example 4.1: The query

```
SELECT elem_name, landfill_name
FROM elem_contained
WHERE landfill_name = 'a'
ENRICH
SCHEMAEXTENSION( elem_name, dangerLevel)
```

extends the result of the SQL query which returned the elements contained in the landfill "a", extracted from the table *elem_contained*, with the information about the level of danger associated to such elements according to the user's knowledge, coded in her RDF knowledge base.

◇

2) Schema Replacement: The clause `SCHEMA REPLACEMENT(attr, prop)` enables the users to replace a column from the results of the SQL query with another column. The values of the replacing attribute are computed from the ontological knowledge which states instances of the property `prop` on values originally appearing in the relational column `attr`.

Example 4.2: The query

```
SELECT name, city
FROM landfill
ENRICH
SCHEMAREPLACEMENT(city, inCountry)
```

extends the result of the SQL query which returns the list of landfills in the database and their city, by replacing the information about the city with the information about the country the city belongs to, according to the user's knowledge, coded in her RDF statements having `inCountry` as the property. Such query can be useful if the user is interested in geographical information at a different, in this case lower, level of details.

◇

3) Boolean Schema Extension: A special case of schema extension is the *Boolean Schema extension*, in which the result of the initial SQL query is extended with a new column which can only assume boolean values, `BOOLSCHEMAEXTENSION(attr, prop, concept)`. Intuitively, given a relational attribute `attr` (from the schema resulting from the SQL query, i.e., listed in the `SELECT` clause), an ontological property `prop` and an ontological `concept`, for every value of attribute `attr` which is related to the given concept through the specified property in the ontological knowledge base the value *true* will appear in the extension boolean column, all the other values will be associated to the value *false*.

Example 4.3: The query

```
SELECT elem_name
FROM elem_contained
WHERE landfill_name = 'a'
ENRICH
BOOLSCHEMAEXTENSION( elem_name, isA,
HazardousWaste)
```

returns the list of elements contained in the landfill 'a', according to the given relational database, each one couples with the boolean information about it being dangerous, according to the user's ontological knowledge.

◊

4) Boolean Schema Replacement: Similarly, the clause `BOOLSCHEMAREPLACEMENT(attr, prop, concept)` introduces in the query result a boolean attribute, but instead of addend it to the schema of the result, it replaces it to the attribute `attr` appearing as a parameter.

Example 4.4: The query

```
SELECT name, city
FROM landfill
ENRICH
BOOLSCHEMAREPLACEMENT(city, inCountry,
Italy)
```

extracts the list of *landfills* and the *city* in which they are located. Instead of returning the name of the city in the enriched result, it returns the boolean information about it being in Italy or not.

◊

5) Replace Constant: The clause `REPLACECONSTANT (cond, attr, prop)` is an enrichment which affects the semantics of the `WHERE` clause. Given (the identifier of) a filter condition `cond`, an attribute `attr` appearing in the `SELECT` clause and a property `prop`, the RDF triples having `prop` as their property element are extracted from the knowledge base. Subjects and objects of such triples are stored and used in such a way that, whenever constant (appearing as subject of the triple) is mentioned in the condition `cond`, the corresponding object value is replaced to it, before the condition is evaluated.

Example 4.5: Consider the query

```
SELECT landfill_name
FROM elem_contained
WHERE ${elem_name = HazardousWaste:cond1}
ENRICH
REPLACECONSTANT(cond1, HazardousWaste,
dangerQuery)
```

The equality in the `WHERE` clause refers to "HazardousWaste", which is not an attribute in the database schema, while it is a concept appearing in the user's contextual ontology. The property `dangerQuery`, on the other hand, in this case is not a property name occurring in stored triples, while it refers to a SPARQL query which extracts from the contextual ontology the list of dangerous elements (more details on the implementation of such queries can be found in [25].

The intended meaning of this query is to treat the list of HazardousWastes as if it was a relational attribute, and compare, while evaluating the `WHERE` clause, the values in the column `elem_name` with the values returned by the SPARQL query.

◊

6) Replace Variable: Another possibility to enrich the `WHERE` clause relies on the clause `REPLACEVARIABLE(cond, attr, prop)`. The behaviour is similar to the previous clause. Also in this case, `cond` is the identifier of the condition being affected by the enrichment, `attr` denotes the variable on which the condition is tested, while `prop` refers to either a property from the contextual onthology, or the identifier of a previously stored SPARQL query.

Example 4.6: Assume the user is interested in returning the names of the landfills in which some common element appears. Moreover, assume the user wants to leverage, in addition to the information stored in the relational database, her domain knowledge about elements which typically occur together, captured in her ontological knowledge by the property *oreAssembage*. The desired result can be obtained through the query

```
SELECT Elecond1.landfill_name AS l_name1,
Elecond2.landfill_name AS l_name2,
Elecon1.elem_name
FROM elem_contained AS Elecond1,
elem_contained AS Elecond2
WHERE ${ Elecond1.elem_name <>
Elecond2.elem_name:cond1} AND
Elecond1.elem_name = Elecond2.elem_name
ENRICH
REPLACEVARIABLE(cond1, Elecond2.elem_name,
oreAssemblage)
```

◊

Remark 4.1: Notice that the last two enrichment strategies, operating on the `WHERE` clause, require a parameter to denote the condition to be enriched. This is because, in general, the `WHERE` clause might contain conjunction/disjunction of multiple sub conditions, including joins.

To obtain the desired behaviour (i) we identify the condition through a syntax construct which uses characters which wouldn't be accepted at that point by standard SQL; (ii) a dedicated scanner recognised such characters, generates the condition syntax tree and identifies the conditions; (iii) the query is then "clean" by removing the non SQL identification part, so that a syntactically correct SQL query can be processed, and then extended as specified. ◦

B. Processing SESQL Queries

Figure 6 illustrates the semantically enriched query processing module of the the CroSSE platform, with its two main components: the semantic query parser (SQP) and the Semantic Query Module (SQM). The SQP module is the parser for the enriched queries. Given a SESQL query, it identifies its two subcomponents, the SQL query (to be enriched), and the enrichment specification. This last is analysed and associated to its Syntax Tree that describes the structure of the request. The SQL query, and enrichment specification syntax tree are the input to the Semantic Query Module (SQM). The Semantic Query Module uses the Syntax Tree it received in input to detect the requested enrichment strategies and constructs a corresponding SPARQL query, to extract the relevant knowledge form the ontology. At this point the SQL query and the SPARQL query are independently issued on the relational database and on the ontological knowledge base, respectively. A JoinManager module combines the partial results returned by the two independent queries, leveraging the resource mapping described in an XML file. A temporary support database stores the results in temporary tables, on which a final SQL query (obtained by leveraging the enrichment syntax tree) is issued to generate the final result of the SESQL query.

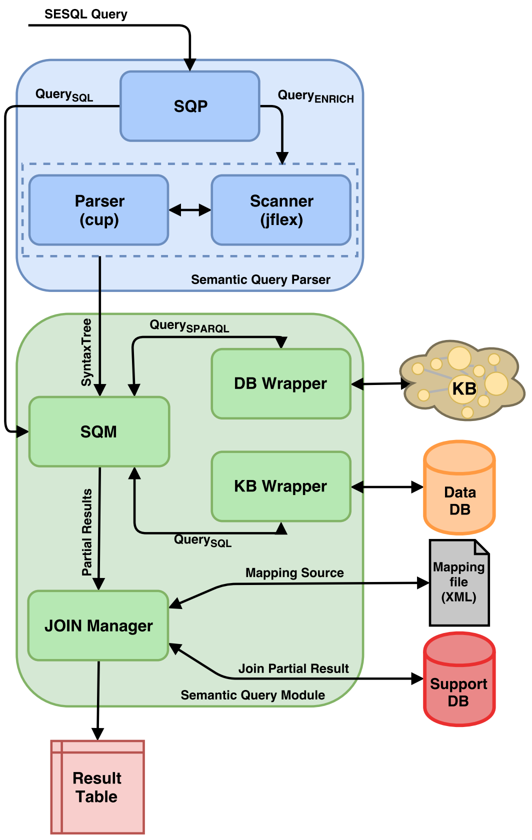


Fig. 6. Architecture of the enriched query processing module of CroSSE

V. Conclusions

In this paper, we introduced *Crowd Sourced Semantic Enrichment (CroSSE)*, a crowdsourced knowledge platform supporting semantic enrichment and context-aware data access for scientific investigations. The semantic tagging module provides a set of functionalities that implement the belief-based knowledge expansion, allowing each user the possibility to (a) explore the common meta-knowledge, which is shared among all users; (b) extend common knowledge according to her domain of expertise (in particular by means of RDF statements connecting existing concepts through suggested properties and/or by defining new concepts and new properties); and (c) borrow (part of) the knowledge inserted by other users, possibly leading to an enrichment of the common knowledge. The proposed SESQL language allows users to enrich a relational databank with semantic tagging information and pose contextualised queries to support contextualised data analysis. SESQL has been proposed and tested in the domain of tracking the urban and mining secondary raw materials. We are now planning to package the semantic enrichment and query modules as a general purpose product, to be used in other domains that could benefit from social information sharing.

Acknowledgments

We are grateful to all the partners of the *SmartGround* H2020 EU project, especially to our colleagues from the Earth Sciences Department at the University of Torino, Piergiorgio Rossetti, Giovanna Antonella Dino and Giulio Biglia, for their input and feedback as domain experts.

References

[1] J. H. Kim, X. Chen, K. S. Candan, and M. L. Sapino, "Hive open research network platform," in *Joint 2013 EDBT/ICDT Conferences, EDBT '13 Proceedings, Genoa, Italy, March 18-22, 2013*, 2013, pp. 733–736.
[2] "Minc: A social platform for fostering educational interactions," 2017. [Online]. Available: https://hive.asu.edu/minc/
[3] D. Beckett, Ed., *RDF/XML Syntax Specification (Revised)*, ser. W3C Recommendation, 2004. [Online]. Available: http://www.w3.org/TR/rdf-syntax-grammar/
[4] F. D. Mauro, P. Pasteris, M. L. Sapino, K. S. Candan, G. A. Dino, and P. Rossetti, "Crowdsourced semantic enrichment for participatory e-government," in *Proceedings of the 8th International Conference on Management of Digital EcoSystems, MEDES 2016, Biarritz, France, November 1-4, 2016*, 2016, pp. 82–89.
[5] E. Kandogan, M. Roth, P. M. Schwarz, J. Hui, I. G. Terrizzano, C. Christodoulakis, and R. J. Miller, "Labbook: Metadata-driven social collaborative data analysis," in *2015 IEEE International Conference on Big Data, Big Data 2015, Santa Clara, CA, USA, October 29 - November 1, 2015*, 2015, pp. 431–440.
[6] J. Mortensen, M. Musen, and N. Noy., "Developing crowdsourced ontology engineering tasks: An iterative process," in *CrowdSem*, 2013, pp. 79–88.
[7] J. Mortensen, P. R. A. andM. A. Musen, and N. F. Noy, "Crowdsourcing ontology verification," in *ICBO*, 2013, pp. 40–45.
[8] E. G. Caldarola, A. Picariello, and A. M. Rinaldi, "An approach to ontology integration for ontology reuse in knowledge based digital ecosystems," in *MEDES*, 2015.
[9] M. C. Suárez-Figueroa, A. Gómez-Pérez, E. Motta, and A. Gangemi, *Ontology Engineering in a Networked World*. Springer, 2012, ch. 2.
[10] A. Levy, "The information manifold approach to data integration," *IEEE Intelligent Systems*, pp. 1312–16, 1998.
[11] S.Kambhampati, E. Lambrecht, U. Nambiar, Z. Nie, and G. Senthil, "Optimizing recursive information gathering plans in emerac," *JIIS*, pp. 22–119, 2004.
[12] S. Adali, K. Candan, Y. Papakonstantinou, and V. Subrahmanian, "Query processing in the sims information mediator," in *SIGMOD*, 1996, pp. 137–148.
[13] Y. Arens, C. Knoblock, and C. Hsu, "Query processing in the sims information mediator," in *AAAI*, 1996.
[14] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajararnan, Y. Sagiv, J. Ullman, V.Vassalos, and J. Widom, "The tsimmis approach to mediation: Data models and languages," *JIIS*, vol. 2, 1997.
[15] H. Davulcu, J. Freire, M. Kifer, and I. Ramakrishnan, "A layered architecture for querying dynamic web content," in *SIGMOD*, 1999.
[16] H. Davulcu, M. Kifer, G. Yang, and I. Ramakrishnan, "Design and implementation of the physical layer in webbases: The xrover experience," in *DOOD*, 2000.
[17] L. Xu and D. W. Embley, "Combining the best of global-as-view and local-as-view for data integration," in *ISTA*, 2002, pp. 123–136.
[18] N. E. Taylor and Z. G. Ives, "Reconciling while tolerating disagreement in collaborative data sharing." in *SIGMOD*, 2006.
[19] K. S. Candan, H. Cao, Y. Qi, and M. L. Sapino, "System support for exploration and expert feedback in resolving conflicts during integration of metadata." *VLDB Journal*, vol. 17, no. 6, pp. 22–119, 2008.
[20] P. Ziegler and K. R. Dittrich, *Data Integration - Problems, Approaches, and Perspectives*, 2007, ch. 3.
[21] F. Di Pinto, D. Lembo, M. Lenzerini, R. Mancinu, A. Poggi, R. Rosati, M. Riccardo, Ruzzi, and D. Savo, "Optimizing query rewriting in ontology-based data access," in *EDBT*, 2013, pp. 561–572.
[22] K. Munir, M. Odeh, and R. Mcclatchey, "Ontology-driven relational query formulation using the semantic and assertional capabilities of OWL-DL," *Know.-Based Syst.*, vol. 35, pp. 144–159, Nov. 2012.
[23] A. Maedche, S. Staab, R. Studer, Y. Sure, and R. Volz, "Seal – tying up information integration and web site management by ontologies," *IEEE Data Engineering Bulletin*, vol. 25, 2002.
[24] [Online]. Available: https://jena.apache.org/
[25] G. Cavallo, "Integrazione di conoscenza di contesto in database relazionali," Master's thesis, Universita di Torino, Italia, 7 2017, advisor: Prof. M.L. Sapino.