MDM: Governing Evolution in Big Data Ecosystems

Sergi Nadal
Universitat Politècnica de Catalunya
Barcelona, Spain
snadal@essi.upc.edu

Alberto Abelló
Universitat Politècnica de Catalunya
Barcelona, Spain
aabello@essi.upc.edu

Oscar Romero
Universitat Politècnica de Catalunya
Barcelona, Spain
oromero@essi.upc.edu

Stijn Vansummeren
Université Libre de Bruxelles
Brussels, Belgium
svsummer@ulb.ac.be

Panos Vassiliadis
University of Ioannina
Ioannina, Greece
pvassil@cs.uoi.gr

ABSTRACT

On-demand integration of multiple data sources is a critical requirement in many Big Data settings. This has been coined as the data variety challenge, which refers to the complexity of dealing with an heterogeneous set of data sources to enable their integrated analysis. In Big Data settings, data sources are commonly represented by external REST APIs, which provide data in their original format and continously apply changes in their structure (i.e., schema). Thus, data analysts face the challenge to integrate such multiple sources, and then continuosly adapt their analytical processes to changes in the schema. To address this challenges, in this paper, we present the Metadata Management System, shortly MDM, a tool that supports data stewards and analysts to manage the integration and analysis of multiple heterogeneous sources under schema evolution. MDM adopts a vocabulary-based integration-oriented ontology to conceptualize the domain of interest and relies on *local-as-view* mappings to link it with the sources. MDM provides user-friendly mechanisms to manage the ontology and mappings. Finally, a query rewriting algorithm ensures that queries posed to the ontology are correctly resolved to the sources in the presence of multiple schema versions, a transparent process to data analysts. On-site, we will showcase using real-world examples how MDM facilitates the management of multiple evolving data sources and enables its integrated analysis.

1 INTRODUCTION

In recent years, a vast number of organizations have adopted data-driven approaches that align their business strategy with advanced data analysis. Such organizations leverage Big Data architectures that support the definition of complex data pipelines in order to process heterogeneous data, from multiple sources, in their original format. External data (i.e., neither generated nor under control of the organization) are commonly ingested from third party data providers (e.g., social networks) via REST APIs with a fixed schema. This requires data analysts to tailor their processes to the imposed schema for each source. A second challenge that data analysts face is the adaptation of such processes upon schema changes (i.e., a release of a new version of the API), a cumbersome task that needs to be manually dealt with. For instance, in the last year Facebook's Graph API[1] released four major versions affecting more than twenty endpoints each, many of them breaking changes. The maintenance of such data analysis processes is critical in scenarios integrating tenths of sources and exploiting them in hundreds of analytical processes, thus its automation is badly needed.

The definition of an integrated view over an heterogeneous set of sources is a challenging task that Semantic Web technologies are well-suited for to overcome the *data variety* challenge [3]. Given the simplicity and flexibility of ontologies, they constitute an ideal tool to define a unified interface (i.e., global vocabulary or schema) for such heterogeneous environments. This family of systems, that perform data integration using ontologies, propose to define a global conceptual schema (i.e., by means of an ontology) over the sources (i.e., by means of mappings) in order to rewrite ontology-mediated queries (OMQs) to the sources. The state of the art approaches for such integration-oriented ontologies are based on generic reasoning algorithms, that rely on certain families of description logics (DLs). Such approaches rewrite an OMQ, first to an expression in first-order logic and then to SQL. This approach, commonly referred as *ontology-based data access* (OBDA) [8], does not consider the management of changes in the sources, and thus such variability in their schema would cause OMQs either crash or return partial results. This issue, which is magnified in Big Data settings, is caused because OBDA approaches represent schema mappings following the *global-as-view* (GAV) approach, where elements of the ontology are characterized in terms of a query over the source schemata. GAV ensures that the process of query rewriting is tractable and yields a first-order logic expression, by just unfolding the queries to the sources, but faulty upon source schema changes [2]. To overcome this issues a desiderata is to adopt the *local-as-view* (LAV) approach. Oppositely to GAV, LAV characterizes elements of the source schemata in terms of a query over the ontology, making it inherently more suitable for dynamic environments [4]. LAV flexibility, however, comes at the expense of computational complexity in the query answering process.

To address these challenges, we adopt a vocabulary-based approach for data integration. These approaches are not necessarily restricted to the expressiveness of a DL and its generic reasoning algorithms. Such settings rely on rich metamodels for specific integration tasks, here focused on schema evolution. Under certain constraints when instantiating the metamodel, it is possible to define specific efficient algorithms that resolve LAV mappings without ambiguity. To this end, we created the Metadata Management System, or shortly MDM[2], an end-to-end solution to assist data stewards and data analysts during the Big Data integration lifecycle. Data stewards are provided with mechanisms to

[1] https://developers.facebook.com/docs/graph-api/changelog

[2] http://www.essi.upc.edu/~snadal/mdm.html

semi-automatically integrate new sources and accomodate schema evolution into a global schema. In turn, data analysts have means to pose OMQs to such global schema by making transparent the underlying mechanisms to query the sources with LAV mappings.

MDM implements a vocabulary-based integration-oriented ontology, represented by means of two RDF graphs, specifically the global graph and the source graph [5]. The former representing the domain of interest (also known as domain ontology) and the latter the schema of the sources. The key concepts are *releases*, which represent a new source or changes in existing sources. A relevant element of releases are *wrappers* (from the well-known mediator/wrapper architecture in data integration), the mechanism enabling access to the sources (e.g., an API request or a database query). Upon new releases the schemata of wrappers are extracted and their RDF-based representation stored in the source graph. Afterwards, the data steward is aided on the process of linking such new schemata to the global graph (i.e., define the LAV mapping). Orthogonally, data analysts pose OMQs to the global graph. The current de-facto standard to query ontologies is the SPARQL query language, however to enable non-expert analysts to query the sources MDM offers an interface where OMQs are graphically posed as subgraph patterns of the global graph, which are automatically translated to SPARQL. A specific query rewriting algorithm takes care of how to properly resolve LAV mappings, a process that consists on the discovery of joins amongst wrappers and their attributes, regardless of the number of wrappers per source.

Motivational use case. As motivational use case, and for the sake of understandability, we will analyse information related to european football teams. This represents the simple use case that will be demoed on-site amongst others with higher complexity (i.e., the SUPERSEDE project). Precisely, we aim to ingest data from four data sources, in the form of REST APIs, respectively providing information about players, teams, leagues and countries. The integrated schema of this scenario is conceptualized in the UML depicted in Figure 1, which we use as a starting point to provide a high-level representation of the domain of interest, used to generate the ontological knowledge captured in the global graph.

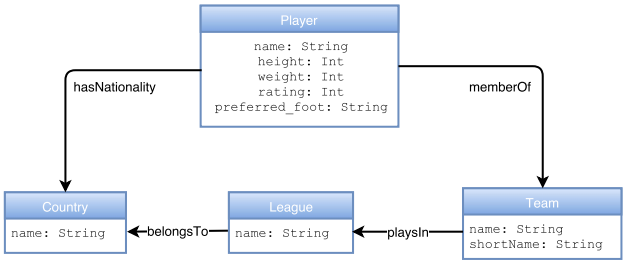

Figure 1: UML of the motivational use case

Each of the APIs is independent from each other, and thus they differ in terms of schema and format. Thus, for instance, the *Players API* provides data in JSON format while the *Teams API* in XML. An excerpt of the content provided by such two APIs is depicted in Figure 2. Next, the goal is to enable data analysts to pose OMQ to the ontology-based representation of the UML diagram (i.e., global graph) by navigating over the classes. Specifically, we aim the sources to be automatically accessed under multiple schema versions. An exemplary query would be, *"who are the players that play in a league of their nationality?"*.

Outline. In the rest of the paper, we will introduce the demonstrable features to resolve the motivational and other exploratory


```
{
  "id": 6176,
  "name": "Lionel Messi",
  "height": 170.18,
  "weight": 159,
  "rating": 94,
  "preferred_foot": "left",
  "team_id": 25
}
```

```
<team>
  <id>25</id>
  <name>FC Barcelona</name>
  <shortName>FCB</shortName>
</team>
```


Figure 2: Sample data for *Players API* and *Teams API*

queries. We first provide an overview of MDM and then, we present its core features to be demonstrated. Lastly, we outline our on-site presentation, involving the motivational use case and a complex real-world use case.

2 DEMONSTRABLE FEATURES

MDM presents an end-to-end solution to integrate and query a set of continuously evolving data sources. Figure 4 depicts a high-level overview of the approach. Its pillar is the Big Data integration (BDI) ontology [7], the metadata model (i.e., set of design guidelines) that allow data stewards to semantically annotate the integration constructs that enable automating the evolution process and unambiguously resolve query answering.

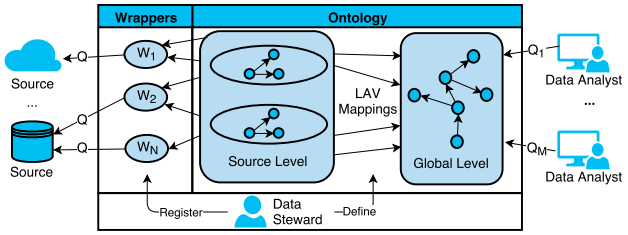

Figure 4: High-level overview of our approach

We devise four kinds of interaction with the system, which are in turn the offered functionalities: *(a) definition of the global graph*, where data stewards define the domain of interest for analysts to query; *(b) registration of wrappers*, either in the presence of a new source or the evolution of an existing one; *(c) definition of LAV mappings*, where LAV mappings between the source and the global graphs are defined; and *(d) querying the global graph*, where data analysts pose OMQs to the global graph which are automatically rewritten over the wrappers. In the following subsections, we describe how MDM assists on each of them.

2.1 Definition of the global graph

The global graph, whose elements are prefixed with `G`, reflects the main domain concepts, relationships among them and features of analysis. To this end, we distinguish between two main constructs *concepts* and *features*. Concepts (i.e., instances of `G:Concept`) are elements that group features (i.e., `G:Feature`) and do not take concrete values from the sources. Only concepts can be related to each other using any user-defined property, we also allow to define taxonomies for them (i.e., `rdfs:subClassOf`). It is possible to reuse existing vocabularies to semantically annotate the data at the global graph, and thus follow the principles of Linked Data. This, enables data to be self-descriptive as well as it opens the door to publish it on the Web [1]. Furthermore, we restrict features to belong to only one concept.

MDM supports the definition of the global graph avoiding the need to use external ontology modeling tools (e.g., *Protégé*). Figure 5 depicts an excerpt of the global graph for the demo use case, focusing on the concepts *Player* and *Team*. Like we said, we reuse vocabularies as much as possible, hence the concept *Team* is reused from http://schema.org/SportsTeam. When no reuse is possible, we define the example's custom prefix `ex`. As data stewards interact with MDM to define the global graph, the corresponding RDF triples are being generated automatically.

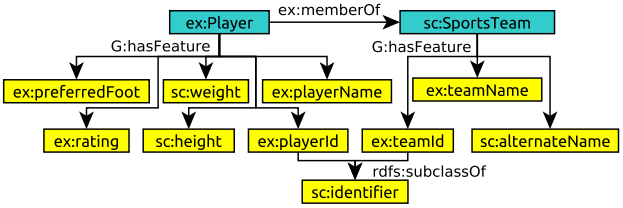

Figure 5: Global graph for the motivational use case. Blue and yellow nodes denote concepts and features

2.2 Registration of new data sources

New wrappers are introduced either because we want to consider data from a new data source, or because the schema of an existing source has evolved. Nevertheless, in both cases the procedure to incorporate them to the source level, whose elements are prefixed with `S`, is the same. To this end, we define the data source (i.e., `S:DataSource`) and wrapper (i.e., `S:Wrapper`) metaconcepts. Data stewards must provide the definition of the wrapper, as well as its signature. We work under the assumption that wrappers provide a flat structure in first normal form, thus the signature is an expression of the form $w(a_1, \ldots, a_n)$ where w is the wrapper name and $a_1, \ldots, a_n$ the set of attributes. With such information, MDM extracts the RDF-based representation of the wrapper's schema (i.e., creates elements of type `S:Attribute`) which are incorporated to the existing source level. In the case of a wrapper for an existing data source, MDM will try to reuse as many attributes as possible from the previous wrappers for that data source. However, this is not possible among different data sources as the semantics of attributes might differ. In the case of attributes in the source graph, as they are not meant to be shared, oppositely to features in the global graph, there is no need to reuse external vocabularies.

Figure 6 depicts an excerpt of the source graph for the sources related to players and teams, the former with a wrapper's signature $w_1(id, pName, height, weight, score, foot, teamId)$ and the latter $w_2(id, name, , shortName)$. Note that, for w_1, some attribute names differ from the data stored in the source (see Figure 2), this is due to the fact that the query contained in the wrapper might rename (e.g., $foot$) or add new attributes (e.g., $teamId$). The definition of a wrapper (e.g., a MongoDB query, a Spark job, etc.) is out of the scope of MDM and should be carried out by the data steward.

2.3 Definition of LAV mappings

LAV mappings are encoded as part of the ontology. We represent them as two components, (a) a subgraph of the global graph, one per wrapper, and (b) a function linking attributes from the source graph to features in the global. The former are achieved thanks to RDF named graphs, which allow to identify subsets of other RDF

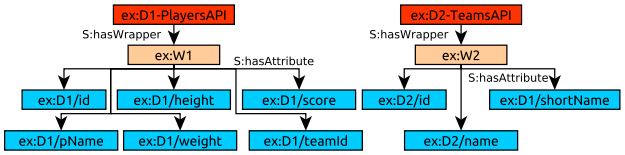

Figure 6: Source graph for the motivational use case. Red, orange and blue denote data sources, wrappers and attributes

graphs identified by an URI. In this case, the URI will be the one for the wrapper. The latter are achieved via the `owl:sameAs` property. Note that, traditionally, the definition of LAV mappings was a difficult task even for IT people. However, in MDM LAV mappings can be easily asserted through the interface: each wrapper must map to a named graph (i.e., a subset of the global graph), and a set of `owl:sameAs` from attributes to features. The task consists on first selecting a wrapper, and then, with the mouse, drawing a contour around the set of elements in the global graph that this wrapper is populating (including concept relations).

Figure 7 depicts the LAV mappings for wrappers w_1 and w_2, respectively in red and green. Note the intersection in the concept `sc:SportsTeam` and its identifier, this will be later used when querying in order to enable joining such concepts. However, this joins are only restricted to elements that inherit from `sc:identifier`.

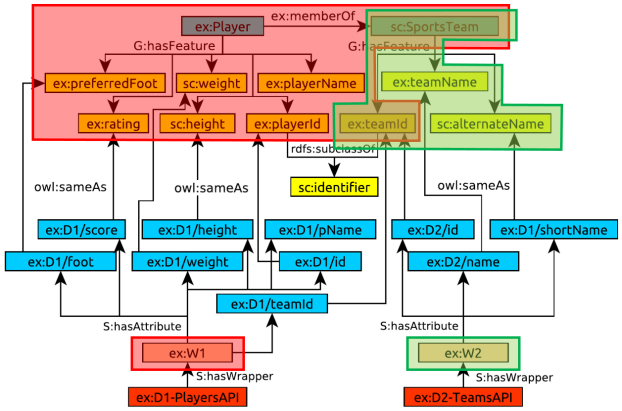

Figure 7: LAV mappings for the motivational use case

2.4 Querying the global graph

To overcome the complexity of writing SPARQL queries over the global graph, MDM adopts a graph pattern matching approach to enable non-technical data analysts perform their OMQs. Recall that the *WHERE* clause of a SPARQL query consists of a graph pattern. To this end, the analyst can graphically select a set of nodes of the global graph representing such pattern, we refer to it as a *walk*. Then, a specific query rewriting algorithm takes as input a walk and generates as a result an equivalent union of conjunctive queries over the wrappers resolving the LAV mappings [7]. Such process consists of three phases: *(a) query expansion*, where the walk is automatically expanded to include concept identifiers that have not been explicitely stated; *(b) intra-concept generation*, that generates partial walks per concept indicating how to query the wrappers in order to obtain the requested features for the concept at hand; and *(c) inter-concept generation*, where all partial walks are joined to obtain a union of conjunctive queries.

Using the excerpt of the ontology depicted in Figure 7, we could graphically pose an OMQ fetching the name of the players and their teams. Figure 8 shows how such query can be defined in MDM by drawing a contour (in red) around the concepts and features of interest in the global graph. On the right hand side, it is depicted the equivalent SPARQL query, as well as the generated relational algebra expression over the wrappers. Table 1 depicts a sample of the output resulting of the execution of the query.

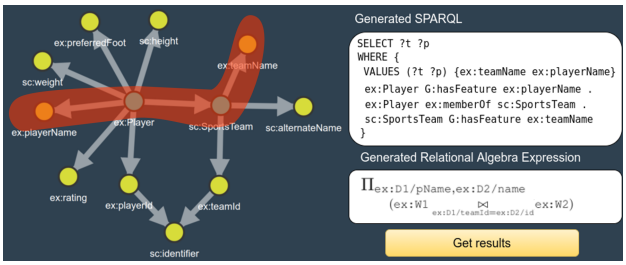

Figure 8: Posing an OMQ in MDM

ex:teamName	ex:playerName
FC Barcelona	Lionel Messi
Bayern Munich	Robert Lewandowski
Manchester United	Zlatan Ibrahimovic

Table 1: Sample output for the exemplary query.

2.5 Implementation details

MDM has been developed at UPC BarcelonaTech in the context of the SUPERSEDE[3] project using a service-oriented architecture. It is the cornerstorne of the Big Data architecture supporting the project, and a central component of its Semantic Layer [6]. On the frontend, MDM provides the web-based component to assist the management of the Big Data evolution lifecycle. This component is implemented in *JavaScript* and resides in a *Node.JS* web server. The interface makes heavy use of the *D3.js* library to render graphs and enables the user to interact with them. Web interfaces are defined using the *Pug* template engine, and a number of external libraries are additionally used. The backend is implemented as a set of REST APIs defined with *Jersey* for *Java*, thus the frontend interacts with the backend by means of HTTP REST calls. This enables extensibility of the system and a separation of concerns in such big system. The backend makes heavy use of *Jena* to deal with RDF graphs, as well as its persistence engine *Jena TDB*. Additionally, a *MongoDB* document store is responsible of storing the system's metadata. Concerning the execution of queries, the fragment of data provided by wrappers is loaded into temporal SQLite tables in order to execute the federated query.

3 DEMONSTRATION OVERVIEW

In the on-site demonstration, we will present the functionality of MDM relying based on two use cases. First, we will focus on the paper's motivational scenario, in order to comprehensively show the functionalities offered by MDM. Next we will focus on the SUPERSEDE use case, a real-world scenario of Big Data integration under schema evolution in order to show the full potential and benefits of MDM. We will cover the four possible kinds of interactions with MDM, taking the role of both data steward (definition of the global graph, registration of new wrappers, definition of LAV mappings) and data analyst (querying the global graph). We will showcase how MDM aids on each of the processes, considering as well the input from participants. Precisely, the following scenarios will be covered:

System setup. In the first scenario we will take the role of a data steward that has been given a UML diagram (likewise Figure 1), and assigned the task of setting up a global schema to enable integrated querying of a disparate set of sources. Thus, we will show how MDM supports the definition of its equivalent global graph (likewise Figure 5) within the interface. Once finished, we will introduce the four sources (i.e., the players API, teams API, etc.) and a wrapper for each. We will show how MDM automatically extracts the schemata of wrappers to automatically generate the source graph (likewise Figure 6). Finally, we will show how MDM supports the graphical definition of named graphs, which are the basis for LAV mappings, and thus properly maps the source and global graphs (likewise Figure 7).

Ontology-mediated queries. With the global graph set up and a set of data sources and wrappers in place, now we can act as data analysts in order to pose OMQs to the system. We will encourage participants to propose their queries of interest, this is possible because MDM presents the global graph and allows to graphically draw a walk around its nodes. This is later automatically translated to its SPARQL form (likewise Figure 8), and to a relational algebra expression derived from the query rewriting process. MDM presents the execution of the query in tabular form.

Governance of evolution. In Big Data ecosystems, changes in the structure of the data sources will frequently occur. In this scenario, we will release a new version of one of the APIs including breaking changes that would cause the previously defined queries to crash. First, we will showcase how MDM easily supports the inclusion of this new source into the existing global graph and the definition of its LAV mappings. Next, we will execute again the queries that were supposed to crash showing how MDM has adapted the generated relational algebra expressions, where the two schema versions are now fetched and yield correct results.

ACKNOWLEDGMENTS

This work was partly supported by the H2020 SUPERSEDE project, funded by the EU Information and Communication Technologies Programme under grant agreement no 644018, and the GENESIS project, funded by the Spanish Ministerio de Ciencia e Innovación under project TIN2016-79269-R.

REFERENCES

[1] Christian Bizer, Tom Heath, and Tim Berners-Lee. 2009. Linked Data - The Story So Far. *Int. J. Semantic Web Inf. Syst.* 5, 3 (2009), 1–22.
[2] Loredana Caruccio, Giuseppe Polese, and Genoveffa Tortora. 2016. Synchronization of Queries and Views Upon Schema Evolutions: A Survey. *ACM Trans. Database Syst.* 41, 2 (2016), 9:1–9:41.
[3] Ian Horrocks, Martin Giese, Evgeny Kharlamov, and Arild Waaler. 2016. Using Semantic Technology to Tame the Data Variety Challenge. *IEEE Internet Computing* 20, 6 (2016), 62–66.
[4] Petar Jovanovic, Oscar Romero, and Alberto Abelló. 2016. A Unified View of Data-Intensive Flows in Business Intelligence Systems: A Survey. *T. Large-Scale Data- and Knowledge-Centered Systems* 29 (2016), 66–107.
[5] Maurizio Lenzerini. 2002. Data Integration: A Theoretical Perspective. In *PODS*. 233–246.
[6] Sergi Nadal, Victor Herrero, Oscar Romero, Alberto Abelló, Xavier Franch, Stijn Vansummeren, and Danilo Valerio. 2017. A software reference architecture for semantic-aware Big Data systems. *Information & Software Technology* 90 (2017), 75–92.
[7] Sergi Nadal, Oscar Romero, Alberto Abelló, Panos Vassiliadis, and Stijn Vansummeren. 2018. An Integration-Oriented Ontology to Govern Evolution in Big Data Ecosystems. *Submitted to Information Systems* (2018). Available at https://arxiv.org/abs/1801.05161.
[8] Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. 2008. Linking Data to Ontologies. *J. Data Semantics* 10 (2008), 133–173.

[3] https://www.supersede.eu