Analytical Metadata Modeling for Next Generation BI Systems

Jovan Varga[a,*], Oscar Romero[a], Torben Bach Pedersen[b], Christian Thomsen[b]

[a] *Department of Service and Information System Engineering (ESSI), Universitat Politècnica de Catalunya, BarcelonaTech, Jordi Girona 1-3, Barcelona, Spain*
[b] *Department of Computer Science, Aalborg Universitet, Selma Lagerlöfs Vej 300, Aalborg, Denmark*

Abstract

Business Intelligence (BI) systems are extensively used as in-house solutions to support decision-making in organizations. Next generation BI 2.0 systems claim for expanding the use of BI solutions to external data sources and assisting the user in conducting data analysis. In this context, the Analytical Metadata (AM) framework defines the metadata artifacts (e.g., schema and queries) that are exploited for user assistance purposes. As such artifacts are typically handled in ad-hoc and system specific manners, BI 2.0 argues for a flexible solution supporting metadata exploration across different systems.

In this paper, we focus on the AM modeling. We propose *SM4AM*, an RDF-based Semantic Metamodel for AM. On the one hand, we claim for ontological metamodeling as the proper solution, instead of a fixed universal model, due to (meta)data models heterogeneity in BI 2.0. On the other hand, RDF provides means for facilitating defining and sharing flexible metadata representations. Furthermore, we provide a method to instantiate our metamodel. Finally, we present a real-world case study and discuss how *SM4AM*, specially the schema and query artifacts, can help traversing different models instantiating our metamodel and enabling innovative means to explore external repositories in what we call *metamodel-driven (meta)data exploration*.

*Corresponding author
Email addresses: jvarga@essi.upc.edu (Jovan Varga), oromero@essi.upc.edu (Oscar Romero), tbp@cs.aau.dk (Torben Bach Pedersen), chr@cs.aau.dk (Christian Thomsen)

November 14, 2018

Keywords: Business Intelligence, Metadata, Ontological Metamodeling

1. Introduction

Traditional BI systems enable data analysis to support decision-making. Their wide acceptance is greatly owed to the user-friendly front-ends (typically OLAP) built on top of data conforming to a predefined data model (e.g., multidimensional schemata). BI tools also exploit metadata (e.g., queries) to provide user assistance (e.g., query recommendations) such that even non-technical users are able to use such front-ends. With the arrival of Big Data, BI 2.0 aims to expand the analysis scope beyond the in-house data sources, that are traditionally used by BI tools, and consider publicly available data on the web and external sources. Although non-controlled and heterogeneous, external data sources should be analyzed in the same fashion (e.g., with a pivot table) as in traditional BI settings. Typical examples of external sources are social networks, forums, and Open data[1] that provide data in semi-structured (e.g., JSON, XML) and non-structured (e.g., textual) formats and use different (meta)data models. To overcome this heterogeneity, BI 2.0 promotes the use of Semantic Web (SW) technologies for representing and consolidating data semantics as well as for exchanging data [1]. As new possibilities attract an increasing number of individuals and groups of non-expert users (e.g., [2]), BI 2.0 emphasizes the need for user assistance so that users are as autonomous as possible in their analysis. Thus, metadata become an important asset to track system usage (e.g., by storing queries) and enable user assistance (e.g., interactive personalization). Different perspectives of BI 2.0 are discussed as Ad-hoc and Collaborative BI [3], Self-Service BI [2], Open BI [4], Situational BI [5], Exploratory OLAP [1] and others.

Although recent trends advocate for the exploitation of metadata artifacts (e.g., schema and queries) to assist the user, the modeling, organization, and

[1] `https://okfn.org/opendata/`

management of these artifacts are typically not systematically addressed and still handled in system-specific manners [6]. Nevertheless, as BI 2.0 requires interaction between systems, using metadata to capture the common semantics and support the automation of processes becomes a priority. As a first step in this direction, the Analytical Metadata (AM) framework [6] has been proposed and, based on a survey of the current state-of-the-art, it defines the user assistance process, the needed metadata artifacts as well as their processing to enable automatic user assistance when exploring and analyzing data in the context of BI 2.0. The AM artifacts also need to be shared/reused and automatically processed across different systems.

Data and metadata modeling approaches are widely applied in software engineering and database domains to enable systematic data organization and automation. Likewise, modeling of AM is necessary and highly desired, especially considering the BI 2.0 context. Indeed, some recent approaches already model certain AM artifacts. For instance, instead of keeping queries in logs, [7] represents queries according to a query metadata model and stores them in a common repository. However, strict modeling in BI 2.0 is hardly applicable due to the high level of heterogeneity of models and sources, e.g., relational and graph data models. Relevantly, semantic (or knowledge) graph models (i.e., RDF(S) and OWL) can represent several types of schemata [8] and due to their flexibility they are the cornerstone of the SW for creating the web of Linked Data [9]. Thus, semantic graphs are good candidates for handling structured and semi-structured sources in a unified way and, today, these formalisms are the basis of Ontology-Based Data Access (OBDA) [10], the main approach to tackle the so-called *Variety Challenge* in Big Data [11] (e.g., see [10] and [12]).

Thus, we follow this spirit to capture the semantics of AM artifacts in an RDF-based metamodel. Note, however, the AM is not an artifact to perform data integration, but a set of metadata artifacts to facilitate metadata exchange between different systems that must interact. Thus, we first advocate for linking the metadata models of independent systems via metamodeling. This approach, even if well-known in Software Engineering, is overlooked in the SW. We use

RDF [13] to represent metadata as a semantic graph that can be shared and reused across different systems via Linked Data. We use the metamodel abstraction level due to the heterogeneity of models. The RDF Schema vocabulary [14] built on top of RDF enables the representation of different *ontological modeling layers* (see [15]). Hence, our approach is *ontological metamodeling* [16] for BI 2.0 metadata and it includes a method defining steps to instantiate a metamodel with (existing) models that, in turn, have instances. To discuss the benefits of the approach, we present a case study. Overall, the main contributions of our work are as follows:

- We present the Semantic Metamodel for Analytical Metadata (*SM4AM*), an RDF-based metamodel for AM. The metamodel formalizes metadata artifacts needed to enable systems interaction and user assistance in BI 2.0.
- Given the challenge of metamodeling in RDF, we provide a method defining detailed steps on how to instantiate the metamodel for system-specific metadata models.
- We present a case study where we apply our approach to interlink two independent real-world data sets in order to allow cross-domain analysis. This case study shows the benefits of using *SM4AM* to, first, link their models via metamodeling and then, use such links to reduce the amount of (meta)data to be explored, in a metamodel-driven (meta)data exploration.

The present paper is a significant extension of an earlier workshop paper [17]. We extended and simplified the metamodel, added a detailed method for the metamodel instantiation, and presented a case study to show the practical benefits of *SM4AM*. In particular, Sections 1, 2, 5, 6, and major part of Section 3.3 are completely new, while the other sections are significantly updated.

The rest of the paper is organized as follows. Section 2 presents a motivating example and Section 3 explains related work and the necessary prerequisites to understand our approach. Then, the complete metamodel is presented in

Section 4. Section 5 defines a method comprising of steps for instantiation of an ontological metamodel and Section 6 elaborates on the application level case study. Finally, Section 7 concludes the paper.

2. Motivating Example

To exemplify a BI 2.0 scenario, let us consider a journalist named Joe who investigates development of countries. He is focusing on two real-world data sources from Linked Data that are illustrated in Figure 1. One data source is World Bank Linked Data (WBLD)[2] that provides data about countries and World Bank (WB) projects supporting countries around the world. The other data source is DBpedia[3] that is an RDF representation of data published on Wikipedia and covers a wide range of topics.

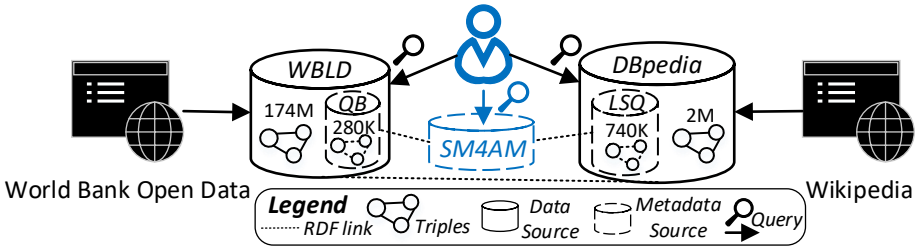

Figure 1: Motivating Example and Case Study Settings

Instead of simple explorations such as keyword search provided by WB and Wikipedia, using their RDF representation we want to enable wider querying possibilities, as typically done in BI. However, exploring such data sets is challenging as the data exposed are not known in advance by the user. Thus, Joe must first identify the relevant subset of his interest (e.g., population of countries) which is tedious due to the (meta)data volume. In the SW, this search is done by explorative querying over the available metadata. Specifically, WBLD contains 174 million triples, where 280 thousands are metadata, while DBpedia

[2] `http://worldbank.270a.info/.html`
[3] `http://wiki.dbpedia.org/`

contains more than 2 million triples about countries such as country description, name(s), and geographical coordinates. Thus, discovering what data is available in these data sources is complicated for a user such as Joe. Although reusing existing queries over DBpedia could help, it is still hard to find the country-related ones as there are 740,000 available queries. Clearly, these tasks require significant manual efforts.

In order to facilitate the querying, Joe can use *SM4AM* to lead the search over the sources, in what we call metamodel-driven (meta)data exploration. In particular, linking the WBLD and DBpedia metadata to *SM4AM* can enable Joe to directly identify the schema information for WBLD data (this metadata is defined using the QB vocabulary [18]), as well as to retrieve what other users searched about countries by reusing queries over DBpedia (metadata query artifacts for DBpedia are defined using the LSQ vocabulary[7]). The metadata of these two data sources are now linked through *SM4AM* and this facilitates identifying the corresponding metadata artifacts respectively. Also, note that not all metadata need to be linked to *SM4AM* but those artifacts to be exploited by Joe. This way, the amount of (meta)data triples and queries that Joe should explore prior to identifying his data of interest is drastically reduced, e.g., from 174 millions of total triples to 210 metadata elements for WBLD (i.e., relevant schema information). This setting is also used for the case study in Section 6 where we present further insights. The same principles can be applied to other cases where metadata are exposed using SW technologies and different RDF vocabularies such that they can be linked to *SM4AM*.

3. Background and Related Work

In this section, we introduce the background and discuss the related work necessary for understanding our approach. We first provide details about AM. Then, we explain the SW technologies and their role in BI 2.0. Finally, we focus on metamodeling as our choice for representing AM and discuss related work.

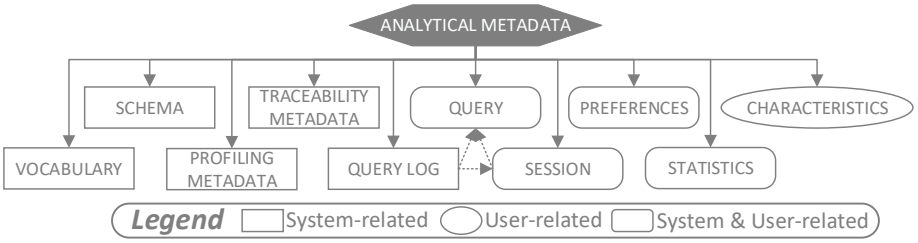

Figure 2: Analytical Metadata Artifacts

3.1. Analytical Metadata

The AM framework is presented in [6] explaining the role of AM for the user assistance in BI 2.0. It includes the AM taxonomy, defined according to a survey, that represents a set of metadata artifacts for this context. Each artifact is a concept that can be used for assisting the user with data analysis such as OLAP. Thus, the taxonomy consists of typical metadata artifacts related to traditional approaches (e.g., a query) and alternative ones (e.g., traceability metadata) that come from new BI 2.0 approaches. Figure 2 illustrates the AM artifacts classified into system-related and/or user-related artifacts.

In our context, a data source can be described by the AM artifacts that are defined as follows. *Vocabulary* defines a data set business terms and their relationships. It represents the end-users day-by-day vocabulary and their mappings to the data set schema. *Schema* represents the data model while *Profiling metadata* capture technical characteristics of the data set. *Traceability metadata* describe information about where the data come from and transformations made. Further, *Query* represents a user inquiry for certain data (disregarding the form it takes), *Query log* is a list of all queries ever posed, and *Session* is a sequence of queries posed by the user performing a certain analysis. *Preferences* refer to the user preferences about the result set selection (e.g., the year of analysis) and/or representation prioritization (e.g., visualization chart). Finally, *Statistics* captures data usage indicators (e.g., most queried piece of data) while *Characteristics* capture the explicitly stated information about the users (e.g., name, job position, etc.).

Due to the high variety of data sources in BI 2.0, the metadata models between different systems are typically heterogeneous. Thus, there is a need for a common yet flexible solution that enables identification and correlation of metadata concepts while supporting a high degree of customization entailed by the specific systems. Consequently, first, for any two data sources to be explored jointly their metadata artifacts must be aligned. The overall objective of *SM4AM* is to facilitate this task. Following good habits in Software Engineering, we formalize it at the *metamodel* abstraction level and using *RDF*-based formalisms.

3.2. Resource Description Framework

The means for flexible (meta)data representation range from XML[4] to semantically rich but computationally complex approaches such as the OWL[5] ontology language. OWL is the most expressive of them all, and provides powerful reasoning capabilities that comes at the cost of computational efficiency. Deactivating reasoning entails less expressivity but enables wider adoption and fosters sharing and reusing of (meta)data, as claimed in *the principle of least power* used by the SW [19]. Accordingly, in our approach we follow the strategy of RDF(S)-based vocabularies. Hence, we do not fully exploit the reasoning services provided by semantic graphs. Instead, we use their embedded semantics to create machine-readable annotations that facilitate the automation of exchanging metadata. Even if we considered OWL in our approach, we would use it as a vocabulary (in the spirit of RDF(S)), while exploiting its advanced reasoning capabilities is beyond the scope of this paper and remains for future work.

As BI 2.0 settings bring extremely large metadata volumes, computational efficiency, sharing, and reuse are major priorities. Thus, following the principle of least power, our choice is RDF(S) as it provides sufficient expressivity

[4] `http://www.w3.org/XML/`

[5] `http://www.w3.org/TR/owl2-overview/`

regardless of its simplicity. RDF constructs are very flexible for capturing data semantics as most information can be naturally represented as RDF triples and IRIs. A triple consists of a subject, a predicate, and an object, and represents a directed binary relationship (a predicate) between two resources (a subject and an object), e.g., see line 2 in Figure 3, or a resource (a subject) and a literal (an object), e.g., see line 3 in Figure 3.[6] The subject and predicate are represented with IRIs[7] that enable their unique identification, while the object can be an IRI or a literal value. A set of triples form an RDF graph. RDF vocabularies are used to define the semantics of IRIs and enable their (re)use across the (Semantic) Web. As discussed in [9], RDF is a standardized data model where data access is simplified as data are self-describing, thus supporting the same concepts reuse in independent systems.

```
1 ex:MDLevel rdf:type ex:SchemaComponent .
2 ex:refArea rdf:type ex:MDLevel .
3 ex:refArea ex:areaCode "08034"^^xsd:decimal .
```

Figure 3: Class and Instance Concept Example

The primary vocabulary for modeling in RDF is RDF Schema (RDFS) [14], which is an extension of the RDF vocabulary. Although quite simple, RDF and RDFS (jointly referred as RDF(S)) represent formalisms that can be used for data integration, mappings of business and technical terms, incorporation of external and heterogeneous sources, and other. In the context of metadata, we particularly outline the typing possibilities via the rdf:type property. As in an RDF graph both data and metadata with their classes and instances are stored together, defining data types is convenient to semantically distinguish the metadata and data instances. For instance, [20] uses data type links to extract the schema from data. RDF[8] graphs are queried with the SPARQL

[6] All prefixes used throughout the paper are available at `http://www.essi.upc.edu/~jvarga/sm4am_materials/sm4am_prefixes.txt`

[7] `http://en.wikipedia.org/wiki/Internationalized_resource_identifier`

[8] For the sake of simplicity, note that from here on RDF should be read as RDF(S).

query language [21]. It applies pattern matching techniques to retrieve subgraphs (i.e., set of triples) that fit the pattern (i.e., the query). Furthermore, SPARQL supports federated queries[9] for retrieving results from more than one data source. It provides a powerful framework for working with RDF graphs.

For a diverse environment such as BI 2.0, even existing non-RDF metadata repositories can be included if an RDF middleware for ontology-based data access is created. Furthermore, RDF is widely applied in the Linked Data initiative which is accepted by a significant number of participants including both companies, e.g., Thomson Reuters[10], and public government institutions, e.g., the European Union[11]. An overview of Linked Data and RDF can be found in [9]. Linked Data interlinking principles can also be applied to correlate the metadata of different systems in BI 2.0. Note that indeed RDF is already a mean for capturing different types of metadata (e.g., describing music, images, videos, etc.)[12]. Finally, an important characteristic of RDF models is that they can be extended by following the good practices of ontology evolution techniques (e.g., see [22]). Novel concepts can be incorporated and the metamodel can evolve according to the needs. This mechanism is already used in SW environments and BI 2.0 systems can strongly benefit from it.

3.3. Metamodeling

The heterogeneity of models for metadata artifacts (e.g., queries) in BI 2.0 hinders the use of a single metadata model [6]. Thus, our idea is to represent the AM artifacts at the metamodel abstraction level capturing the common semantics. Then, the system-specific metadata models can be defined as instances of the metamodel elements, both when creating new or enriching the existing models. Indeed, a metamodel is convenient for the settings where heterogeneous models can be created as instances of the metamodel [16]. Moreover, rather than

[9] `http://www.w3.org/TR/2013/REC-sparql11-federated-query-20130321/`
[10] `https://www.thomsonreuters.com`
[11] `http://ec.europa.eu/digital-agenda/en/open-data-0`
[12] `http://dublincore.org/`

classical metamodeling that defines linguistic *metatypes* that models must conform to, using RDF enables us to perform *ontological metamodeling*[13] such that ontological metamodel types do not require literal conformance at the model level [23]. Such an approach provides the flexibility needed for BI 2.0. We next dive into details on ontological metamodeling and position our approach with respect to the related work.

Traditional Object Management Group (OMG) modeling infrastructure has been the foundation for Model-Driven Development (MDD) and Model-Driven Architecture (MDA) approaches that promote the use of four layers for modeling [16]. These layers represent the following modeling abstraction levels: M0 – data, M1 – model, M2 – metamodel, and M3 – meta-metamodel. For each two adjacent levels, the lower is an instance of the higher level. Thus, these approaches follow the linguistic metamodeling where instances at the lower abstraction level strictly conform to the types of the higher level [23].

An ontological classification defines the semantics of an element but does not require literal syntactic conformance (e.g., an instance can have its own structure and properties). Furthermore, it enables the definition of ontological (i.e., semantic) types of elements within the same abstraction level [23]. Thus, we aim for ontological RDF metamodeling of AM (see Sections 4 and 5 for details).[14] In RDF, the class-instance relation between a meta class and a class, and of a class with a class instance can be defined with the *rdf:type* property. As discussed in [15], in this way we can distinguish between *ontological metamodel layers*. In RDF modeling there is no restriction that an instance cannot be a class at the same time. For example, in Figure 3 we can express that ex:MDLevel (i.e., a level in an MD schema) is a class instantiated by ex:refArea and an instance of ex:SchemaComponent at the same time. However, the fact that an RDF IRI can be, at the same time, a class and an instance yields basic modeling

[13]Considering that ontological metamodeling can be understood differently depending on the approach, note that the present paper considers the meaning given in [16].

[14]In this context, we use the terms *meta class* for meta type and *class* for type.

problems that have been formalised in terms of the *Russell paradox* formulated within the set theory [24]. This is due to the fact that RDF modeling does not follow well-known Software Engineering modeling principles. Thus, it is one of our objectives to provide foundations on how to properly use RDF for ontological metamodeling.

An RDF ontological metamodel can significantly reduce the manual efforts for aligning different data sets via metadata. For instance, we can automatically identify query elements in different systems [25] and align them via metamodeling. Note that we propose an alternative to the traditional instance-based data integration. Although querying is much more precise when materializing the integration of instances, it renders unfeasible in cases where the data is out of our control, which is the main scenario of BI 2.0 [1]. Alternatively, we propose to start by aligning metadata artifacts (i.e., models). Importantly, our approach facilitates the tedious task of ontology alignment as it involves only models, which are typically much smaller in size than instances. Hence, using ontologies for the metamodeling abstraction level drastically reduces the metadata search space (i.e., the number of pair checks to identify what metadata elements link to each other) and facilitates the alignment of the metadata models. Note that since RDF stores data and metadata together, once the models are aligned querying the instances can be done via rdf:type. We define the search space as the amount of (meta)data to be explored by the user. In the sequel, we discuss the related work considering other perspectives on the relation between metamodels and ontologies (summarized in Table 1), as well as the existing BI metadata metamodels (summarized in Table 2).

The use of metamodels for modeling ontologies. There is an OMG initiative for an Ontology Definition Metamodel (ODM). The initiative motivated approaches such as [26] making their own ODM proposals and finally resulted in an official ODM specification by OMG [27]. While these approaches focus on linguistic metamodeling and define language constructs at the M2 layer, our approach is focused on the ontological metamodels where both metamodel and model belong to the M1 layer.

The use of ontologies to provide semantics for metamodels. Discussions on the relation between ontologies and metamodels are surveyed and presented in [28, 29, 30]. These approaches acknowledge the previous perspective by discussing the creation of domain ontologies that are used as a concrete engineering artifact represented in a specific language. However, their emphasis is on considering the notion of ontology in a more philosophical manner where an ontology defines concepts independently of a modeling language. This perspective results in foundational ontologies describing the real-world knowledge without focusing on a particular modeling language. Examples of foundational ontologies are the Descriptive Ontology for Linguistic and Cognitive Engineering [31] focusing on "the ontological categories underlying natural language and human common sense"[15] and Unified Foundational Ontology [32] that can be used to (re)design and evaluate conceptual modeling, including ontology representation, languages. Moreover, approaches like [33] use ontologies to represent language metamodels and support their semantic integration and map different language metamodels. As such, these approaches can fit to the M3 layer since they can be used for defining the constructs used in a modeling language. Our approach clearly differs from these approaches as we focus on the ontological metamodeling of a domain ontology within the M1 layer.

Ontological engineering. These approaches focus on achieving desired computational properties when using domain ontologies [29]. Here, the use of extensive ontologies with complex logical foundations can be computationally expensive, leading to the use of lightweight ontologies and related languages (e.g., OWL) [29]. For example, OWL 2 uses "punning"[16] to support reasoning where a concept is interpreted as a class or an instance depending on the context. As this can still be computationally expensive, RDF(S) can, although limited in its reasoning capabilities, be used instead. Computational efficiency is crucial in our approach. Furthermore, reasoning is limited to the SPARQL

[15] `https://en.wikipedia.org/wiki/Upper_ontology`

[16] `http://www.mkbergman.com/913/metamodeling-in-domain-ontologies/`

RDF(S) entailment[17]. As previously explained, our primary focus is on exploiting the machine-readable semantics embedded in RDF(S) rather than its reasoning capabilities.

Multi-level modeling. Recent approaches such as [34] argue about the need for multi-level domain representation where a type of an element can be defined within the subject domain. Hence, [34] builds upon the ontology-based conceptual modeling language OntoUML. Such approaches focus on providing new ontological foundations to explicitly support such language constructs that, as discussed above, belong to the M3 layer and differ from our approach. Other multi-level modeling approaches such as [35], provide foundations (e.g., their modeling language) for an arbitrary number of abstraction levels for both linguistic and ontological metamodeling. As such they can fit to multiple modeling layers. A challenge with such approaches is that the two kinds of metamodeling can be mixed causing confusion [23]. Our approach differs in that it has a fixed number of domain abstraction levels, as well as that it uses RDF(S) as modeling language, which although less expressive is more widespread thanks to the Linked Data initiative.

Table 1: Comparison of the kinds of approaches using ontologies and metamodels

	Kind	Layer	Language specific	Traditional infrastruct.	Fixed #layers
Metamodels for ontologies	Linguistic	M2	✓	✓	✓
Ontologies for metamodels	Knowledge/ Linguistic	M3	X	✓	✓
Ontology engineering	Ontological	M1	✓	X	X
Multi-level modeling	Linguistic/ Ontological	Multi	✓	X	X
SM4AM	Ontological	M1	✓	X	✓

Finally, we summarize the discussion on the related work by revising some existing approaches in the traditional BI settings that already represent metadata

[17] https://www.w3.org/TR/sparql11-entailment/

at the metamodel abstraction level. Most importantly, the Common Warehouse Metamodel (CWM) [36] is a standardized solution for metadata modeling and management, focusing on the *schema*, *traceability*, and *vocabulary* AM artifacts. Another metamodeling solution is presented in [37] that focuses on the traceability between the data sources and the target schema and their relation with the user requirements in the data warehouse context. Similarly, an approach to model the *user preferences* and *schema* metadata is proposed in [38]. These approaches are based on the classical modeling standards (e.g., UML, MOF) that follow the strict modeling principle [23] where models must conform to the related metamodels. This is too restrictive for BI 2.0, and our approach provides more flexible means to define the semantics to guide the metadata exploration.

Table 2: Comparison of the BI metadata metamodels

	Query	Query Log	Session	Preferences	Schema	Statistics	Traceability	Vocabulary	User Charact.	Profiling	Ontological
Preference Metamodel[38]				✓	✓						
Trace Metamodel[37]							✓				
CWM[36]					✓		✓	✓			
SM4AM	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

4. The *SM4AM* Metamodel

In this section we present *SM4AM*. We explain the general design principles of the metamodel, elaborate on the metamodel elements, and exemplify the metamodel usage. Finally, we elaborate on the *SM4AM* definition in RDF.

4.1. The Metamodel Design

Considering that BI 2.0 is still in its infancy, many models of the AM artifacts are yet to be defined. Therefore, we perform a top-down knowledge modeling and, taking the high-level conceptualization of AM (see Section 3.1), represent the AM artifacts with a metamodel. Furthermore, instead of introducing new

modeling constructs, the metamodel is formalized in RDF. Thus, *SM4AM* benefits from the simple RDF metadata representation to increase the metadata interoperability and interchange, while its limited semantic expressivity needs to be compensated by following the method for *SM4AM* instantiation and use (see Section 5).

SM4AM aims at capturing atomic building elements for the artifacts. Thus, the AM artifacts are captured either directly, i.e., by a one-to-one mapping of an artifact to the metamodel element, or indirectly where an artifact is represented with more than one metamodel element. As some artifacts are more coarse grained than others (e.g., session vs. query), we also define complex metamodel elements that organize some of the atomic building elements into structurally organized collections (e.g., a schema organizing schema components). This way, different system-specific metadata models can be created by instantiating atomic elements that can be combined into an instance of a complex element. The complete metamodel is illustrated in Figure 4 and Table 3 summarizes how AM artifacts are covered by *SM4AM*.

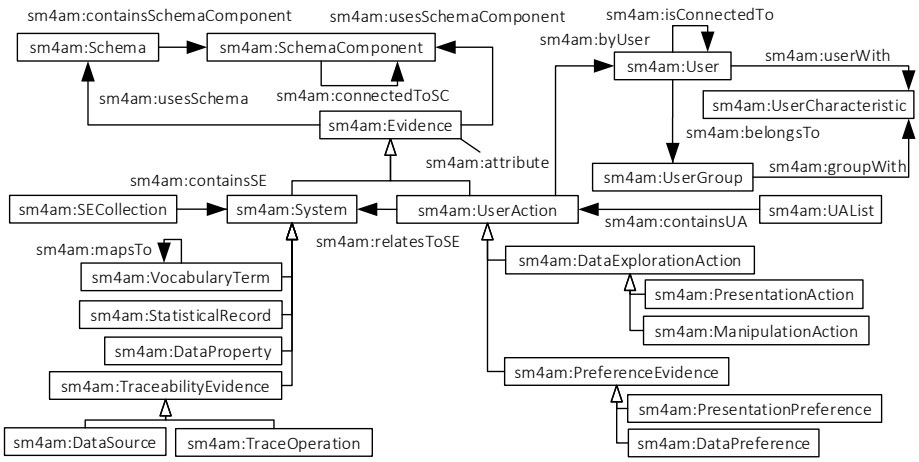

Figure 4: *SM4AM*: A Semantic Metamodel for Analytical Metadata

AM artifacts provide ***evidence*** for performing user assistance activities. AM artifacts are related to the system, user, or both system and user. Therefore,

the metamodel is designed around pieces of evidence about the system, user, or both. A piece of evidence is represented with the sm4am:Evidence abstract meta class (i.e., meta class without instances) that is the super class for all pieces of evidence. This abstract meta class is further sub-categorized into two (also abstract) meta classes, sm4am:System representing a piece of evidence related to the schema and sm4am:UserAction capturing both elements about explicit user actions over the schema (e.g., queries) and elements inferred from these actions (e.g., user preferences). Each piece of evidence is related to the schema while only sm4am:UserAction-related ones also affiliate to the user. The sm4am:UserAction element is linked to sm4am:System via sm4am:relatesToSE (i.e., relates to the system evidence) in order to capture situations when a user action relates to the system element, e.g., a preference over a vocabulary term. Importantly, each piece of evidence can have attributes (i.e., sm4am:attribute) which is how we model the situation where a property links a class with a datatype. In RDF, the literal value cannot be linked to its datatype via rdf:type. Thus, attribute properties are intended for relating a literal value with a class instance at the instance abstraction level, e.g., a value of a certain statistical record. Concrete attribute properties and datatypes are defined/specified at the model level. In the next subsections, we explain the remaining metamodel elements and exemplify how to use the metamodel.

4.2. Schema-Related and User-Related Elements

Schema-Related Elements. The *schema* AM artifact is modeled with the following two meta classes: sm4am:SchemaComponent represents schema components and sm4am:Schema refers to the schema as a whole organizing the components. Each piece of evidence relates to these meta classes via sm4am:usesSchemaComponent and sm4am:usesSchema properties, respectively. Both properties are need as a schema can have many schema components and a schema component can be linked to many schemata, while a piece of evidence can relate to either schema, schema component, or both. The sm4am:containsSchemaComponent property links the schema with schema components, while sm4am:connect-

Table 3: Capturing AM Artifacts with *SM4AM* Elements

AM Artifact	SM4AM element	System/User-related
Vocabulary	sm4am:VocabularyTerm	System
Schema	sm4am:Schema sm4am:SchemaComponent	System
Profiling metadata	sm4am:DataProperty	System
Traceability metadata	sm4am:DataSource sm4am:TraceOperation	System
Query log	sm4am:UAList	System
Query	sm4am:PresentationAction sm4am:ManipulationAction sm4am:UAList	Both
Session	sm4am:UAList	Both
Preferences	sm4am:PresentationPreference sm4am:DataPreference	Both
Statistics	sm4am:StatisticalRecord	Both
Characteristics	sm4am:UserCharacteristic sm4am:User sm4am:UserGroup	User

edToSC interlinks the schema components. This design is a generalization of a typical case where a complete integration schema, e.g., a database schema or an RDF graph, consists of components that can be mutually connected, e.g., interlinked tables of a relational database or nodes of an RDF graph. The example of schema instantiation for our motivating example (see Section 2) is illustrated in Figure 8, Section 6.

User-Related Elements. The (user) *characteristics* AM artifact is modeled with the following meta classes. First, sm4am:UserCharacteristic stands for a specific user characteristic. Second, sm4am:UserGroup models a group of users

with the characteristics to which it is linked via sm4am:groupWith. Third, a user is represented with the sm4am:User meta class. The user can have several characteristics linked via sm4am:userWith, be connected to other users via sm4am:isConnectedTo, and belong to a user group via sm4am:belongsTo. By now, user characteristics have typically been overlooked in the BI area. Existing approaches mostly focus on the user actions (e.g., queries). Inspired by the web recommender systems we believe that user characteristics are necessary to enable better user assistance possibilities in BI 2.0 [6]. Moreover, different social networks emphasize the need for keeping track of the user interconnections. The BI 2.0 systems need to follow this direction and benefit from these metadata for the user assistance features.

Figure 5 exemplifies a simple metadata model and its instances for two journalists exploring the motivating example data sources. In particular, the model level exemplifies the members (ex:OrganizationMember instantiating sm4am:User) of a non-profit organization (ex:NonProfitOrganization instantiating sm4am:UserGroup) that are interested in exploring the countries' populations. A member has an ID (i.e., ex:ID), a profession (i.e., ex:Profession), and a country of origin (i.e., ex:CountryOfOrigin) as instances of the user characteristics (i.e., sm4am:UserCharacteristic). Moreover, a non-profit organization gathers members that are of certain professions and from certain countries. All model elements are interlinked with the related properties as illustrated in the figure. This model has Joe and another journalist as instances of ex:OrganizationMember (i.e., ex:Joe and ex:Journalist2) with their IDs (i.e., ex:ID1 and ex:ID2, respectively) and countries of origin (i.e., ex:Spain and ex:Denmark, respectively), who are both journalists (i.e., ex:Journalist). These persons belong to a European journalist organization (i.e., ex:EuropeanJournalists) that gathers the journalists from Spain and Denmark.

4.3. User-Action-Related Elements

Data-Exploration-Action-Related Elements. Several AM artifacts are modeled by sub-classes of the sm4am:UserAction meta class as we explain in the

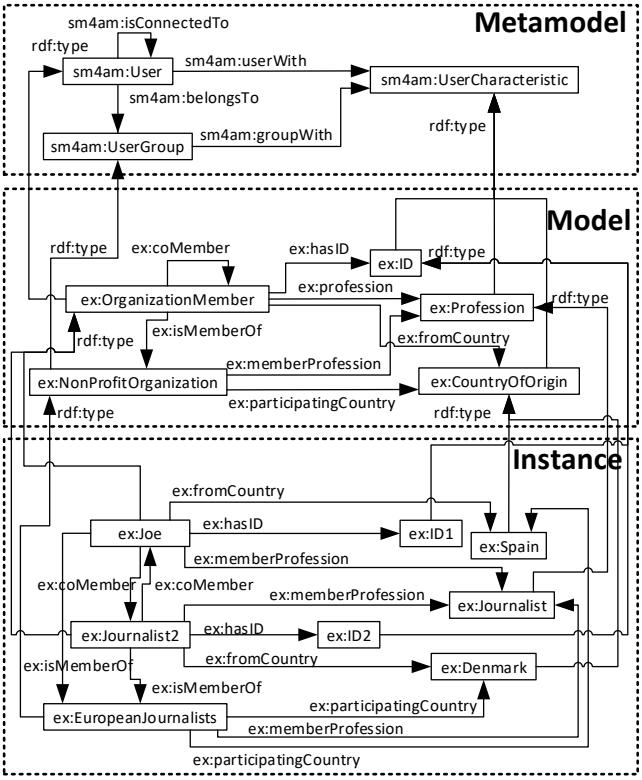

Figure 5: Example of User-related Elements

sequel. All user-action-related elements can be related to a user via sm4am:byUser. The *query*, *session*, and *query log* AM artifacts are considered as data exploration actions representing the explicit user actions when analyzing data (e.g., an operation in a query). As these artifacts are of different granularities (e.g., a query log consists of queries) and we focus to capture the atomic elements that can be composed in more complex structures, the sm4am:DataExplorationAction meta class with its subclasses represent the atomic elements that can be organized in a user action list (i.e., sm4am:UAList). The subclasses are sm4am:ManipulationAction capturing the actions for data handling (e.g., change of data granularity) and sm4am:PresentationAction describing the actions for data presentation (e.g., a diagram type selection). In general, what is to be considered as an atomic element depends on the model instantiating the metamodel (e.g.,

a SPARQL operation can be part of a SPARQL query). A simple example of query (i.e., sm4am:UAList) instantiation for our motivating example (see Section 2) is illustrated in Figure 8, Section 6, while an example of a complete query model instantiating *SM4AM* can be found in [25]. Note that examples of remaining metamodel elements are analogous and extensive examples can be found in [39].

Preference-Related Elements. The *preferences* AM artifact is modeled with the preferences evidence (i.e., sm4am:PreferenceEvidence) category capturing pieces of evidence for the personalization of the user data analysis that can either be stated explicitly, or that can be implied from explicit user actions. They capture the pieces of evidence that enable personalization of the user interaction with the system. We divide them into the following two categories, sm4am:PresentationPreference capturing preferences regarding the data presentation, typically visualization affinities, and sm4am:DataPreference modeling the information about the data interests that can be exploited for the result personalization and similar purposes.

Figure 6 depicts a simple presentation preference example for the metamodel, model, and instance levels. The focus of the example is to illustrate the use of an attribute. In particular, we capture Joe's preference of using a certain chart type when analyzing the population of countries in the motivating example. It has the ex:priority attribute property that links the preference expression with a decimal value defining the priority used for the ranking of preferences.

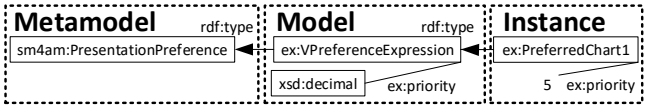

Figure 6: Example of Preference Evidence Elements

User Action List Element. After explaining the atomic elements of user actions, we now provide more details about the sm4am:UAList (i.e., user action list) meta class for composing them into ordered lists that represent different concepts. For instance, a *query* can be represented as an ordered list of: i) one or

more sm4am:ManipulationActions, ii) optionally one or more sm4am:PreferenceEvidence, and iii) one or more sm4am:PresentationActions. At the model level, the instances of user action elements should instantiate attributes (e.g., the ordering) to determine their organization inside of an instance of user action list. Furthermore, sm4am:UAList can be instantiated (at the model level) to model *sessions*, *query logs*, and *exploration patterns* (e.g., querying patterns) depending on the exploitation needs (e.g., query recommendation). The models depend on the concrete systems and the use of user action list for our motivating example (see Section 2) is illustrated in Figure 8, Section 6.

4.4. System-Related Elements

Traceability-Related Elements. The *traceability metadata* AM artifact is modeled with two subclasses of the sm4am:TraceabilityEvidence meta class that represent atomic metamodel elements. The first one is sm4am:DataSource capturing the source where the data come from. The second one is sm4am:TraceOperation and it represents an operation that can be performed over data or metadata before reaching the data/metadata repository. Note that the use of sm4am:attribute at the model level is useful for indicating if the data source is internal/external and trusted/not-trusted. Moreover, it can be used to link to the particular data values for the integration schema. An example of traceability-related elements can be that sm4am:DataSource is instantiated with a class of Linked Open Data sources that, in turn, has DBpedia as its instance.

Profiling-Related Elements. The *profiling metadata* AM artifact is modeled with the sm4am:DataProperty meta class representing technical quality characteristics (e.g., cardinality values). These metadata are typically obtained from data profiling processing in order to enhance the user understanding of the data set. Specific data properties are then defined at the model level depending on the particular system. For instance, the cardinality of schema components can be defined as an instance of data property.

Vocabulary-Related Element. The *vocabulary* AM artifact is modeled with the sm4am:VocabularyTerm meta class that represents a vocabulary entry

as a building block for vocabulary construction. Moreover, the sm4am:mapsTo property links two vocabulary terms and defines the mapping between them (e.g., a synonym relation). The sm4am:VocabularyTerm meta class is instantiated at the model level with the concrete vocabulary entry types (e.g., business terms) and their links that, in turn, have their instances.

Statistics-Related Element. The *statistics* AM artifact is modeled with sm4am:StatisticalRecord as an atomic element for constructing statistics. As before, sm4am:attribute should be instantiated for linking statistical records with their values. Then, the model level should be used to define the class representing the type of statistical indicators (e.g., a schema component usage counter) that are related to the specific numerical datatype as their value (e.g., decimal), while the instance level keeps track of the indicator instances and their value. Note that the values for the sm4am:StatisticalRecord-related metadata should come from system monitoring.

Complex System-Related Elements. After explaining the atomic system-related elements, we provide more details about the sm4am:SEList (i.e., system evidence list) meta class for composing them into ordered lists that represent different concepts. Atomic elements are composed into a complex structure via the sm4am:containsSE property. The attributes (i.e., sm4am:attribute) at the model level should be used to determine structural organization (e.g., ordering). For instance, we can have a complex *trace* composed of data sources and traceability operations aligned in an ordered trace structure. Similarly, we can also have a vocabulary composed of vocabulary terms, statistics composed of statistical records, and a data profile composed of data properties.

4.5. SM4AM and RDF

All meta classes in *SM4AM* are defined as instances of rdfs:Class. Furthermore, to be consistent with the RDF semantics and enable property typing between metamodel and model levels, each property in *SM4AM* is considered as both rdf:Property and rdfs:Class. This way, at the metamodel level it is used as property to link the meta classes and at the same time it can be instantiated at

the model level with a property. Note that examples of similar property formulation can be found in the QB [18] and QB4OLAP [40] vocabularies. Moreover, as they are properties we also define their domain and range meta classes using the rdfs:domain and rdfs:range properties, respectively. For the ease of distinction between properties at different abstraction levels, the terms *meta property*, *property*, or *property instance* refer to a property at the metamodel, model, or instance levels, respectively. Finally, the concept of *attribute* in *SM4AM* represents a meta property that links a meta class with a data type. The concrete data type should be defined at the model level and, therefore, this meta property defines only the domain while the range remains undefined. The definition of sm4am:attribute is related to the sm4am:Evidence meta class in *SM4AM*. [18]

5. A Method for Instantiating *SM4AM*

One of the challenges when using an RDF metamodel is to ensure that it is used in compliance with the RDF specification and in a consistent way when creating system-specific metadata models. This is a lesson learned from our experience with other RDF-based vocabularies (e.g., [40]) where we noticed that they can be used in inconsistent manners. Thus, the precise steps about how *SM4AM* should be instantiated must be defined, especially considering the context of RDF (meta)modeling (see Section 3.3 for details of well-known problems that may appear if metamodeling is not properly bounded in RDF). Hence, this method can be used as basis to implement other metamodels. The ultimate goal is to enable as uniform as possible use of the metamodel and thereby better exploitation possibilities of different models. We next consider the instantiation steps at the model and instance levels.

Model Level. The steps to define a model by instantiating *SM4AM* are defined in Algorithm 1. The algorithm takes the set of <class, meta class> tuples, the set of <property, meta property, domain, range> quadruples, and

[18] See `http://www.essi.upc.edu/~jvarga/sm4am-page.html` for all *SM4AM* triples.

the set of <attribute, domain, range> triples as inputs and returns a set of the resulting RDF triples. The input elements are retrieved by means of related get operations. In the sequel, we exemplify the results of Algorithm 1. Line 1 in Figure 7 exemplifies the class of organization member from Figure 5 as a result of line 4 in Algorithm 1. Moreover, lines 2-4 in Figure 7 exemplify the co-member property from Figure 5 as a result of lines 6 to 13 in Algorithm 1. Furthermore, lines 5-7 in Figure 7 exemplify the priority attribute from Figure 6 as a result of lines 15 to 22 in Algorithm 1. Note that all input elements are IRIs, except for the literal values of attributes at the instance level.

```
1  ex:OrganizationMember rdf:type sm4am:User .
2  ex:coMember rdf:type rdf:Property, sm4am:isConnectedTo .
3  ex:coMember rdfs:domain ex:OrganizationMember .
4  ex:coMember rdfs:range ex:OrganizationMember .
5  ex:priority rdf:type rdf:Property, sm4am:attribute;
6  rdfs:domain ex:VPreferenceExpression;
7  rdfs:range xsd:decimal .
8  ex:Joe rdf:type ex:OrganizationMember .
9  ex:Joe ex:coMember ex:Journalist2 .
10 ex:PreferredChart1 ex:priority "5"^^xsd:decimal .
```

Figure 7: Example of Model and Instance Level Triples

Instance Level. Once a model is defined, it can have its instances defined as detailed in Algorithm 2. The algorithm takes the set of <class instance, class> tuples, the set of <property, domain, range> triples, and the set of <attribute, domain, range> triples as inputs and returns a set of the resulting RDF triples. The input elements are retrieved by means of related get operations. In the sequel, we exemplify the results of Algorithm 2. Line 8 in Figure 7 exemplifies the IRI for our journalist Joe from Figure 5 as a result of line 4 in Algorithm 2. Moreover, line 9 in Figure 7 exemplifies the co-member property from Figure 5 as a result of lines 6 to 11 in Algorithm 2. Finally, line 10 in Figure 7 exemplifies the priority attribute from Figure 6 as a result of lines 13 to 18 in Algorithm 2. The complexity of both algorithms is linear with respect to their input size.

Additional Considerations. In addition to creating new models, the use of RDF enables linking of existing models with *SM4AM* as shown in Section 6.

Algorithm 1: Model Level Triples Definition Algorithm

```
Input: C, P, A ;                // class, property, and attribute inputs, respectively
Output: triples ;                                                  // resulting triples
begin
    triples = ∅;
    foreach cInput ∈ C do
        triples ∪= cInput.getClass() rdf:type cInput.getMetaClass();
    foreach pInput ∈ P do
        triples ∪= pInput.getProperty() rdf:type rdf:Property;
        triples ∪= pInput.getProperty() rdf:type pInput.getMetaProperty();
        if pInput.getDomain() is an instance of
          pInput.getMetaProperty().getDomain() then
            triples ∪= pInput.getProperty() rdf:domain pInput.getDomain();
        else Throw property domain exception ;
        if pInput.getRange() is an instance of pInput.getMetaProperty().getRange()
          then
            triples ∪= pInput.getProperty() rdf:range pInput.getRange();
        else Throw property range exception ;
    foreach aInput ∈ A do
        triples ∪= aInput.getAttribute() rdf:type rdf:Property;
        triples ∪= aInput.getAttribute() rdf:type sm4am:attribute;
        if aInput.getDomain() is an instance of non-abstract sm4am:Evidence
          subclass then
            triples ∪= aInput.getAttribute() rdf:domain aInput.getDomain();
        else Throw attribute domain exception ;
        if aInput.getRange() is a datatype then
            triples ∪= aInput.getAttribute() rdf:range aInput.getRange();
        else Throw attribute range exception ;
    return triples
```

This can be done by reverting the order of the steps in the algorithms above. Furthermore, automation of the metadata processing is crucial to enable stable populating of the metadata repository and further metadata exploitation, e.g., for the user assistance tasks. The metadata modeling is a starting point in this direction. Nevertheless, although the automation is desired, in certain cases the user might still want to state some of these metadata manually, e.g., the expert user can formulate her preferences manually and the system should support this. Moreover, in addition to the elements explicitly captured in *SM4AM*,

Algorithm 2: Instance Level Triples Definition Algorithm

```
Input: C, P, A ;      // class, property, and attribute instance inputs, respectively
Output: triples ;                                                  // resulting triples
begin
    triples = ∅;
    foreach cInst ∈ C do
        triples ∪= tuple.getClassInstance() rdf:type cInst.getClass();
    foreach pInst ∈ P do
        if pInst.getDomain() is an instance of pInst.getProperty().getDomain()
          then
            triples ∪= pInst.getProperty() rdf:domain pInst.getDomain();
        else Throw property instance domain exception ;
        if pInst.getRange() is an instance of pInst.getProperty().getRange() then
            triples ∪= pInst.getProperty() rdf:range pInst.getRange();
        else Throw property instance range exception ;
    foreach aInst ∈ A do
        if aInst.getDomain() is an instance of aInst.getAttribute().getDomain()
          then
            triples ∪= aInst.getAttribute() rdf:domain aInst.getDomain();
        else Throw attribute instance domain exception ;
        if aInst.getRange() is a literal of datatype aInst.getAttribute().getRange()
          then
            triples ∪= aInst.getAttribute() rdf:range aInst.getRange();
        else Throw attribute instance range exception ;
    return triples
```

more metadata are contained implicitly. For instance, statistics about the user actions can be retrieved by counting metadata instances and processing them. We consider this kind of metadata as *derived metadata* and it is up to the specific systems how to exploit this possibility [6].

6. Application Level Case Study

To illustrate the applicability of our approach, we present a case study built around the data sources from our motivating example (see Section 2). We first explain the case study design and internals. Then, we present the data collection and analysis, and discuss the threats to validity.

6.1. Case Study Design

The **setup** of the case study is illustrated in Figure 1 showing the logical relations between WBLD and DBpedia, their metadata and SM4AM. The schema metadata of WBLD are represented with the QB vocabulary [18] and the SPARQL query metadata for DBpedia are represented with the LSQ vocabulary [7]. Note, that SPARQL queries over DBpedia are available in the standalone LSQ data set [7]. WBLD is also linked with DBpedia and contains DBpedia country IRIs. In the case study, the QB metadata are linked to *SM4AM* enabling discovery of schema metadata of the WBLD data sets. Likewise, the LSQ metadata are linked to *SM4AM* enabling discovery of queries over DBpedia. We bridge between both metadata artifacts through *SM4AM*. Moreover, the WBLD to DBpedia links can be used to filter queries over DBpedia.

The **objective** of the case study is to show that the use of *SM4AM* can reduce the user efforts for exploring two independents data sources. In this context, the **research questions** to be answered in our case study are:

RQ1. What are the manual efforts required for using *SM4AM*?

RQ2. How much does the use of *SM4AM* reduce the user efforts for exploring the data sources?

The **metrics** related to the research questions are the number of triples that need to be created and the search space that the user needs to deal with.

Method. To answer the research questions, we first explain how each of the data sources relates to *SM4AM*. Here, we measure the effort to link them to *SM4AM*. Then, we define the queries to retrieve the total volumes of relevant (meta)data and the volumes of (meta)data retrieved considering *SM4AM*. Based on these results we discuss the user efforts for individual and combined analysis of the case study data sources. The details are presented in the sequel.

6.2. Linking the sources to SM4AM

Figure 8 illustrates the triples linking *SM4AM* to QB and LSQ at the metadata model level. QB and LSQ, in turn, link to their metadata instances and data examples for the case study. Note that all these triples already exist, except

for the ones between *SM4AM* and WBLD and LSQ that are discussed below. We start linking QB model elements to the *SM4AM* schema elements.[19] The QB schema structure is defined by the indicators:structure element which further relates to the sdmx-measure:obsValue measure as data being analyzed, and the sdmx-dimension:refPeriod (i.e., period), sdmx-dimension:refArea (i.e., area), and property:indicator (i.e., analyzed indicator; population in this case) dimensions representing data analysis perspectives. These components are linked with indicators:structure via blank nodes (see [13]) as proposed by the QB vocabulary. To facilitate understanding QB, an exemplary instance, the population of Serbia for year 2011, is provided (see [40] for more details on QB).

Next, the LSQ model is linked to *SM4AM*. The sm4am:UAList element is linked to sp:Query with its subclasses. Again, we present an example, in this case a query instance of sp:Select, which is, in turn, linked to the DBpedia IRI for Serbia. Linking to *SM4AM* is currently a manual process. The links are created once and should be provided by the data publisher that is familiar with the data source metadata such that the manual efforts only depend on the number of necessary links. For instance, the case study setup in Figure 8 can be achieved with the four triples presented in Figure 9, where lines 1-3 relate to QB (WBLD) and line 4 to LSQ (DBpedia).

6.3. Case Study Data Collection and Analysis

WBLD/QB data collection. Table 4 shows the number of (meta)data triples for WBLD[20]. Once linked to *SM4AM*, as previously discussed, its schemata can be automatically retrieved with Query 1 and dimensions and measures with Query 2, respectively. Thus, the metadata artifact model (in this case, the data set schema) can easily be retrieved guided by *SM4AM*.

[19] In particular, we use the data set about population of countries from WBLD available at `http://worldbank.270a.info/dataset/SP.POP.TOTL.html`

[20] The values are rounded and retrieved from WBLD website (http://worldbank.270a.info/about.html) or by querying the WBLD SPARQL endpoint.

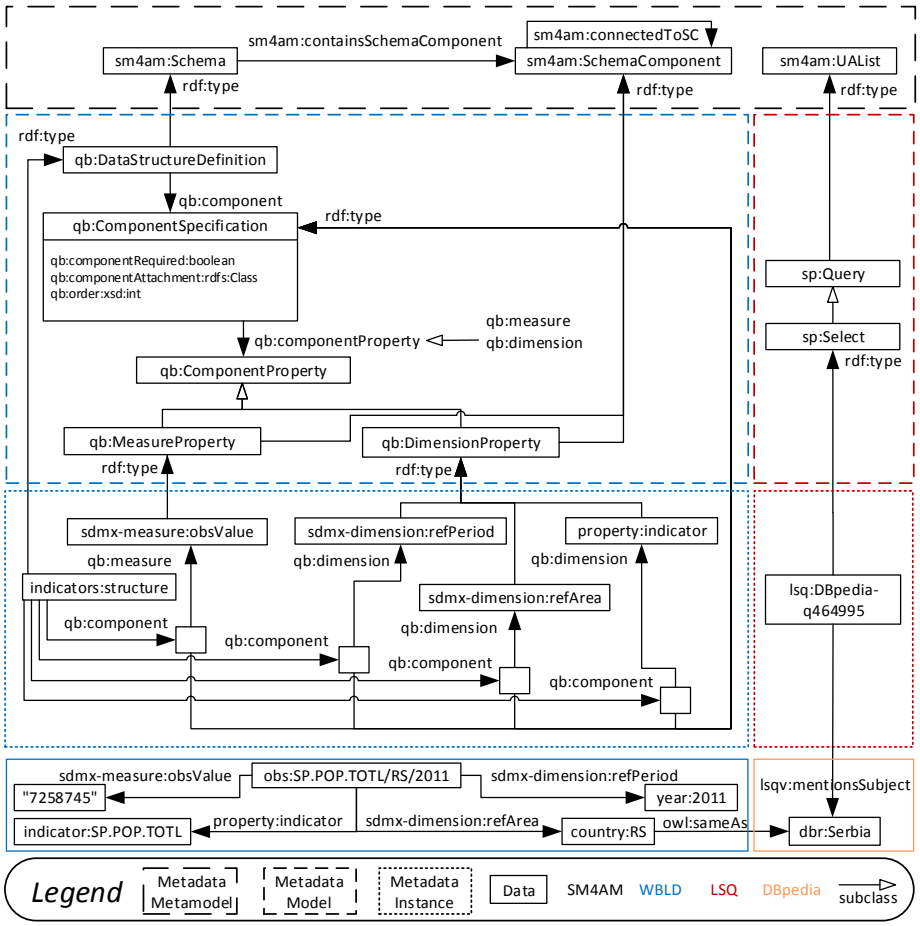

Figure 8: Case Study Internals

```
qb:DataStructureDefinition rdf:type sm4am:Schema .
qb:DimensionProperty rdf:type sm4am:SchemaComponent .
qb:MeasureProperty rdf:type sm4am:SchemaComponent .
sp:Query rdf:type sm4am:UAList .
```

Figure 9: Triples Adding SM4AM Semantics to QB (WBLD) and LSQ (DBpedia)

Query 1. *Retrieve Schemata*

```
SELECT DISTINCT ?schema
WHERE {
?schema a ?modelSchema .
?modelSchema a sm4am:Schema . }
```

Query 2. *Retrieve Schemata Components*

```
SELECT DISTINCT ?schemaComponent
WHERE {
?schemaComponent a ?modelSchemaComponent .
?modelSchemaComponent a sm4am:SchemaComponent . }
```

Table 4: WBLD Triple Numbers

Total number of triples	174M
Metadata number of triples	280K
Number of data sets	>9K
Number of schemata	59
Total number of dimensions	81
Total number of measures	70

Table 5: DBpedia Country-Related Triple Numbers

Total	2,358,094
Average	11,019.13
Max	467,819
Min	1

WBLD/QB discussion. To analyze a data set on the SW, the user typically needs to get familiar with the data organization. This is typically done by learning about schema models or ontologies. However, retrieving only the schema related triples can be a tedious task if the user does not know where to start exploring. For example, the total number of triples in Table 4 indicates that non-guided exploration is a burdensome task. However, these efforts can be significantly reduced if the schemata have additional semantics linking their schema and schema components to *SM4AM* such that they can be automatically retrieved. This way, the (meta)data search space is narrowed from 174 millions of triples (including 280 thousands of metadata triples) to 210 IRIs that can be retrieved with Queries 1 and 2. Moreover, this is enabled with only three additional triples (see Figures 8 and 9).

DBpedia/LSQ data collection. WBLD keeps track of 214 countries and provides their IRIs in both WBLD and DBpedia via owl:sameAs links (identifying the same resources) that can be retrieved with Query 3. Thus, the user can retrieve additional data (i.e., triples) about countries from DBpedia using the DBpedia IRIs. Table 5 shows the total number of country-related triples, as well as average, maximum, and minimum number of triples per country for the 214 countries on the DBpedia SPARQL endpoint.

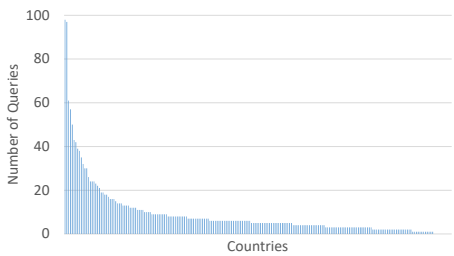

Figure 10: Number of Queries per Country

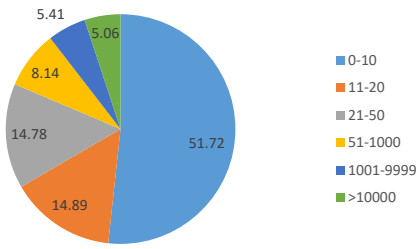

Figure 11: Percentage of Countries per Result Size Range

Query 3. *Retrieve WBLD Countries*

```
SELECT DISTINCT ?c ?cc
WHERE {
?c a <http://dbpedia.org/ontology/Country> .
?c <http://www.w3.org/2002/07/owl#sameAs> ?cc .
FILTER regex ( str(?cc), 'dbpedia.org') }
```

Query 4. *Retrieving Queries Related to an IRI*

```
SELECT DISTINCT ?query
WHERE {
?query ?p ?parameter? .
?query a ?modelQuery .
?modelQuery a sm4am:UAList . }
```

Additionally, the LSQ data set includes more than a million queries where approximately 740,000 are over the DBpedia endpoint. Moreover, Query 4 illustrates how the *SM4AM* metadata can be automatically exploited in metadata models where the query artifact is linked to *SM4AM*. Note that "?parameter?" represents an IRI parameter that is used for the filtering of queries. In our case study, it should be replaced with the DBpedia country IRI. This way, 1908 queries can be retrieved for 214 countries and Figure 10 illustrates the number of queries per country. The maximum number of queries per country is 98 for Germany. The average is approximately 9 queries, while there are only 10 countries with no related queries. These queries include searches like what are the country description, names, geographical coordinates, language, homepage, images, DBpedia class types, etc.

Furthermore, we analyze the result size for the queries previously retrieved. We focus on the 1719 SELECT queries which are typically used for data exploration and can retrieve more than one result. We consider the number of results for the SELECT clause of the query and Figure 11 illustrates the percentage of queries for result size ranges. It shows that half of the queries return at most

10 results, while there are approximately 15% in each of the ranges 11-20 and 21-50 results. Thus, 80% of the queries retrieve at most 50 results and only 5% retrieve 10,000 or more results (note that 10,000 results per query is the default limit on a SPARQL endpoint).

DBpedia/LSQ discussion. Retrieving additional data (i.e., triples) about countries from DBpedia can still be overwhelming if the user is not familiar with the LSQ data model as illustrated in Table 5. Thus, instead of exploring all the available triples related to a country, the user may be interested in other users' queries available in the LSQ data set. In this context, linking LSQ with *SM4AM* brings the following benefits.

First, the *SM4AM* semantics supports correlating the WBLD schema metadata with the DBpedia query metadata (i.e., LSQ) so that the user can reduce the search space for finding the relevant queries. This supports reducing the number of explored queries from 740,000 to approximately 9 queries per country on average. Second, the results in Figure 11 show that in most cases, the amount of query results is small and can be manually analyzed by the user. Thus, reducing the number of queries reduces the data search space from approximately 2.3 million of country-related triples to at most 50 results in 80% of the cases. Moreover, this is enabled with a single triple (see Figures 8 and 9).

Overall results summary. The case study shows that *SM4AM* can guide the search for (meta)data in both data sources and reduce the (meta)data search space that the user needs to deals with. The summary of the search space reductions with *SM4AM* is shown in Figure 12. As our approach is based on *SM4AM*, the same principles can be applied to metadata models other than WLDB/QB and DBpedia/LSQ. Reflecting on the research questions, the case study results show that the manual efforts for using *SM4AM* considered in RQ1 are small and require the creation of only four triples (see Figure 9). This is due to our choice for ontological metamodeling. Thus, linking of an existing metadata model to *SM4AM* can be done manually and it is performed only once. Furthermore, the user efforts for exploring the data sources considered in RQ2 are significantly reduced by using *SM4AM* to narrow the (meta)data

search space to only 210 schema-related IRIs for WBLD/QB and 9 queries per country on average which retrieve at most 50 results in 80% of the cases for DBpedia/LSQ. Intuitively, the user needs to explore much smaller volumes of (meta)data to find the details she is looking for.

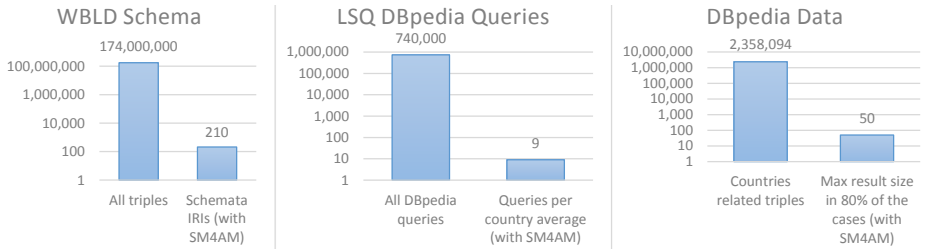

Figure 12: Search Space Reductions with *SM4AM*

6.4. Threats to Validity

Construct validity. Regarding the metamodel, someone may wonder what happens if a concept is not covered by *SM4AM*. However, the metamodel is based on a survey providing the most typical metadata artifacts used in this context. Furthermore, if required, the *SM4AM* extension is supported by means of ontology evolution techniques as suggested in Section 3. Although the case study only covers part of the AM elements, the same metamodeling principles can be followed for the remaining elements.

One threat may be whether a case study with RDF sources is representative for BI 2.0. Nevertheless, although using the same formalism, the case study involves different (meta)data models/sources that reflect the heterogeneity of BI 2.0. Furthermore, semantic formalisms are already used together with other database technologies as in the case of ontology-based data access. This use case is, in any case, representative for the Linked Open Data initiative, which is per se a huge wealth of knowledge to be explored. Another threat could be whether semantic technologies can cope with the performance requirement of BI 2.0. However, note that the volume of metadata is much smaller than data volumes, and the related semantic tools are constantly improving to provide

more efficient and scalable solutions.

Internal validity. When searching for queries, it might happen that no or too many queries are retrieved. Furthermore, a query and/or its results might not be of high quality. Nevertheless, the user still deals with much less (meta)data volumes which certainly facilitates the current means for (meta)data exploration. To improve the quality of the results, additional pre-processing steps (e.g., using another filtering parameter in Query 4) and even query recommendations algorithms can be applied. However, these issues are orthogonal to the discussion here and out of our scope. Current state-of-the-art solutions for these problems in the SW can seemingly be applied as complementary actions.

When analyzing case study steps a question may be whether queries could have been written without *SM4AM*. While at the model level queries can and even should be customized, they could hardly be generalized for other models as required by BI 2.0. In such case, the user is left with a large metadata search space to discover the concepts to be used. Thus, *SM4AM* is needed for BI 2.0.

External validity. The question of generalizing the case study results is another threat. However, using several already existing (meta)data sources provides evidence in this context. Furthermore, providing the metamodel usage guidelines should further contribute in this direction. Metamodeling solutions have already been widely used to support integration and information exchange, and they are the foundations for our approach.

Reliability. To minimize the potential bias in the presentation of the obtained results, the case study steps and constructs have been validated by all the authors of our approach. Furthermore, the public availability of all the used (meta)data sources enables anyone to perform related analysis.

7. Conclusions

Motivated by the need to better support and assist the user experience in the context of BI 2.0, this paper presents *SM4AM*: an RDF-based metadata metamodel. Using ontological metamodeling, *SM4AM* has been designed as a flexible

solution that can be easily shared among heterogeneous systems. Being represented in a semantic-aware format, it supports the metadata processing semi-automation. The practical benefits of *SM4AM* for narrowing the (meta)data search space in a metamodel-driven (meta)data exploration are shown on a case study with two real-world data sets. The case study demonstrates that *SM4AM* can be used not only with new metadata models but also with already existing ones. Furthermore, we have discussed the need to follow an *SM4AM* instantiation method to avoid potential well-known problems related to RDF and metamodeling.

Our approach proposes a compromise solution between the semantic expressivity of *SM4AM* and the performance and flexibility requirements of BI 2.0. Rather than providing strict constraints in the metamodel, *SM4AM* represents a high level abstraction which should be used for correlating the same or similar concepts across different models. This facilitates the metadata discovery and exploration, but it currently requires additional (manual) efforts for the complete alignment of different metadata models. Overall, in this paper we have shown that the SW and BI 2.0 should benefit from good habits from Software Engineering (meta)modeling in order to organize and facilitate cross-domain access to the available (meta)data.

Acknowledgments

This research has been funded by the European Commission through the Erasmus Mundus Joint Doctorate "Information Technologies for Business Intelligence - Doctoral College" (IT4BI-DC) and it has been partially supported by the Secretaria d'Universitats i Recerca de la Generalitat de Catalunya under 2014 SGR 1534.

References

[1] A. Abelló, O. Romero, T. B. Pedersen, R. B. Llavori, V. Nebot, M. J. A. Cabo, A. Simitsis, Using Semantic Web Technologies for Exploratory

OLAP: A Survey, IEEE Trans. Knowl. Data Eng. 27 (2) (2015) 571–588.

[2] A. Abelló, J. Darmont, L. Etcheverry, M. Golfarelli, J. Mazón, F. Naumann, T. B. Pedersen, S. Rizzi, J. Trujillo, P. Vassiliadis, G. Vossen, Fusion Cubes: Towards Self-Service Business Intelligence, IJDWM 9 (2) (2013) 66–88.

[3] H. Berthold, P. Rösch, S. Zöller, F. Wortmann, A. Carenini, S. Campbell, P. Bisson, F. Strohmaier, An Architecture for Ad-hoc and Collaborative Business Intelligence, in: EDBT/ICDT Workshops, 2010.

[4] J. Mazón, J. J. Zubcoff, I. Garrigós, R. Espinosa, R. Rodríguez, Open Business Intelligence: On the Importance of Data Quality Awareness in User-friendly Data Mining, in: EDBT/ICDT Workshops, 2012, pp. 144–147.

[5] A. Löser, F. Hueske, V. Markl, Situational Business Intelligence, 2009, pp. 1–11.

[6] J. Varga, O. Romero, T. B. Pedersen, C. Thomsen, Towards Next Generation BI Systems: The Analytical Metadata Challenge, in: DaWaK, 2014, pp. 89–101.

[7] M. Saleem, M. I. Ali, A. Hogan, Q. Mehmood, A. N. Ngomo, LSQ: The Linked SPARQL Queries Dataset, in: ISWC, 2015, pp. 261–269.

[8] D. Skoutas, A. Simitsis, Ontology-Based Conceptual Design of ETL Processes for Both Structured and Semi-Structured Data, Int. J. Semantic Web Inf. Syst. 3 (4) (2007) 1–24.

[9] C. Bizer, T. Heath, T. Berners-Lee, Linked Data - The Story So Far, Int. J. Semantic Web Inf. Syst. 5 (3) (2009) 1–22.

[10] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, R. Rosati, Linking Data to Ontologies, J. Data Semantics 10.

[11] R. Bean, Variety, Not Volume, Is Driving Big Data Initiatives, https://sloanreview.mit.edu/article/variety-not-volume-is-driving-big-data-initiatives/ (March 2016).

[12] S. Nadal, O. Romero, A. Abelló, P. Vassiliadis, S. Vansummeren, An Integration-oriented Ontology to Govern Evolution in Big Data Ecosystems, Inf. Syst.

[13] R. Cyganiak, D. Wood, M.Lanthaler, Resource Description Framework (RDF): Concepts and Abstract Syntax, `http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/` (2014).

[14] D. Brickley, R. Guha, RDF Schema 1.1, `http://www.w3.org/TR/rdf-schema/` (2014).

[15] S. Koide, H. Takeda, MetaModeling in OOP, MOF, RDFS, and OWL, in: SWESE, 2006.

[16] C. Atkinson, T. Kühne, Model-Driven Development: A Metamodeling Foundation, IEEE Software 20 (5) (2003) 36–41.

[17] J. Varga, O. Romero, T. B. Pedersen, C. Thomsen, SM4AM: A Semantic Metamodel for Analytical Metadata, in: DOLAP, 2014, pp. 57–66.

[18] R. Cyganiak, D. Reynolds, The RDF Data Cube Vocabulary (W3C Recommendation), `http://www.w3.org/TR/vocab-data-cube/` (January 2014).

[19] Tim Berners-Lee, Principles of Design, http://www.w3.org/DesignIssues/Principles.html (last accessed July, 2016).

[20] W. Zheng, L. Zou, W. Peng, X. Yan, S. Song, D. Zhao, Semantic SPARQL Similarity Search Over RDF Knowledge Graphs, PVLDB 9 (11) (2016) 840–851.

[21] E. Prud'hommeaux, A. Seaborne, SPARQL 1.1 Query Language for RDF, http://www.w3.org/TR/sparql11-query/ (2011).

[22] F. Zablith, G. Antoniou, M. d'Aquin, G. Flouris, H. Kondylakis, E. Motta, D. Plexousakis, M. Sabou, Ontology Evolution: A Process-centric Survey, The knowledge engineering review 30 (1) (2015) 45–75.

[23] C. Atkinson, T. Kühne, Demystifying Ontological Classification in Language Engineering, in: ECMFA, 2016, pp. 83–100.

[24] S. Koide, H. Takeda, Inquiry into RDF and OWL Semantics, in: JIST, 2016, pp. 15–31.

[25] J. Varga, E. Dobrokhotova, O. Romero, T. B. Pedersen, C. Thomsen, SM4MQ: A Semantic Model for Multidimensional Queries, in: ESWC, 2017, pp. 449–464.

[26] D. Gasevic, D. Djuric, V. Devedzic, MDA-based Automatic OWL Ontology Development, STTT 9 (2) (2007) 103–117.

[27] Object Management Group, Ontology Definition Metamodel Specification 1.1, https://www.omg.org/spec/ODM/1.1/PDF (last accessed May, 2018).

[28] B. Henderson-Sellers, Bridging Metamodels and Ontologies in Software Engineering, Journal of Systems and Software 84 (2) (2011) 301–313.

[29] G. Guizzardi, On Ontology, ontologies, Conceptualizations, Modeling Languages, and (Meta)Models, in: DB&IS, 2006, pp. 18–39.

[30] N. Guarino, The Ontological Level: Revisiting 30 Years of Knowledge Representation, in: Conceptual Modeling: Foundations and Applications - Essays in Honor of John Mylopoulos, 2009, pp. 52–67.

[31] S. Borgo, C. Masolo, Ontological foundations of DOLCE, in: Theory and applications of ontology: Computer applications, Springer, 2010, pp. 279–295.

[32] G. Guizzardi, Ontological foundations for structural conceptual models, CTIT, Centre for Telematics and Information Technology, 2005.

[33] G. Kappel, E. Kapsammer, H. Kargl, G. Kramler, T. Reiter, W. Retschitzegger, W. Schwinger, M. Wimmer, Lifting Metamodels to Ontologies: A Step to the Semantic Integration of Modeling Languages, in: MoDELS, 2006, pp. 528–542.

[34] V. A. de Carvalho, J. P. A. Almeida, C. M. Fonseca, G. Guizzardi, Multi-level Ontology-based Conceptual Modeling, Data Knowl. Eng. 109 (2017) 3–24.

[35] J. de Lara, E. Guerra, Deep Meta-modelling with MetaDepth, in: TOOLS, 2010, pp. 1–20.

[36] Object Management Group, Common Warehouse Metamodel Specification 1.1, http://www.omg.org/spec/CWM/1.1/PDF/ (last accessed September, 2016).

[37] A. Maté, J. Trujillo, A Trace Metamodel Proposal Based on the Model Driven Architecture Framework for the Traceability of User Requirements in Data Warehouses, Inf. Syst. 37 (8) (2012) 753–766.

[38] N. Kozmina, L. Niedrite, OLAP Personalization with User-Describing Profiles, in: BIR, 2010, pp. 188–202.

[39] J. Varga, Semantic Metadata for Supporting Exploratory OLAP, Ph.D. thesis, Universitat Politècnica de Catalunya (2017).

[40] J. Varga, A. A. Vaisman, O. Romero, L. Etcheverry, T. B. Pedersen, C. Thomsen, Dimensional Enrichment of Statistical Linked Open Data, J. Web Sem. 40 (2016) 22–51.