A Service Oriented Approach for Integrating Metadata from Heterogeneous Digital Libraries

Bernhard Haslhofer*

Abstract.

Metadata in today's digital library systems are accessible through various incompatible interfaces and correspond to heterogeneous schemas. Clients that want to search over a number of distributed metadata sources require a solution that provides uniform and location transparent access to these systems without replicating the metadata stored therein. In this paper we describe a metadata integration approach that conceives these data sources as services in the sense of a service oriented architecture. Without forcing clients to adopt a pre-defined mediation schema, we want to give them the possibility to query multiple autonomous and distributed metadata services by formulating SPARQL queries.

1. Introduction

As a result of numerous digitisation activities, especially in the last decade, the number of cultural institutions like libraries or museums that maintain digital library systems to expose digital objects on the Internet is growing rapidly. These systems store bibliographic metadata records about digital objects to organize their often quite extensive collections. To access digital objects, clients typically formulate queries over the available metadata.

Different institutions use different technical solutions to store their metadata. As a result, their digital objects are now accessible in an inconsistent fashion through a variety of interfaces, ranging from web- or domain-specific protocols to common database query languages. However, applications that need to integrate metadata from multiple digital libraries require easy and uniform access to these systems. Thus, they need a framework that provides interoperability among existing digital library systems.

In this paper we present a metadata integration approach that provides the necessary features to fulfill three main requirements: *support for any kind of data in and any type of interface to a data source*, *integration of heterogeneous metadata schemas*, and *location transparency*. Because Service Oriented Architectures (SOAs) have become a well-accepted design pattern for decentralized architectures, we apply this pattern and consider each metadata source that must be integrated as an independent service. For modelling the semantics of metadata, querying the metadata sources and representing the results returned to the clients, we build our solution on the techniques provided by the W3C Semantic Web Initative.

*Research Studios – Studio Digital Memory Engineering, ARC Seibersdorf research, Thurngasse 8/20, A-1090 Vienna, email: bernhard.haslhofer@researchstudio.at

2. Requirements

From the institutions' perspective there is a strong will to keep already existing systems and the metadata stored therein in place. Due to technical but also because of legal reasons (e.g. digital rights management) many institutions do not want to export their metadata or adjust their systems to requirements coming from external systems. So rather than *materializing* metadata from various sources into a central data store, we need a *virtual integrated system* which builds on top of existing architectures and integrates metadata on demand. In the following we describe the institutions' requirements for such a system. Although most of them originate from the well-known problems of data integration [1], we believe that it is worth to investigate them further from the perspective of digital libraries.

2.1. Any Kind of Data in and any Type of Interface to a Data Source

A given digital library system may store highly structured metadata in a relational database, maintain semi-structured metadata descriptions like XML or HTML documents, or even store completely unstructured metadata, e.g. files on the file system. In case that two data sources maintain structured metadata, the applied data models might still differ in their usage of constraints. Depending on the data model's structural properties, data sources are accessible through different interfaces including Web browsers, structural query languages (e.g. SQL), unstructured queries (e.g. Google-like full-text search), or domain specific interfaces such as metadata exchange protocols (e.g. OAI-PMH[1], Z.39.50[2]).

Since our goal is to build a metadata integration framework that can be used by higher level applications to access metadata in various sources, providing a full text search interface is insufficient. We rather need to offer a more expressive, at least semi-structured, query language.

2.2. Integration of Heterogeneous Metadata Schemas

In the digital library domain, metadata schemas define the semantics of metadata. Many institutions use standardized schemas (e.g. MARC-21[3]), others apply their own schemas which are tailored to their specific needs. The complexity of metadata schemas ranges from flat element lists (e.g. Dublin Core[4]) to complex ontologies (e.g. CIDOC-CRM[5], FRBR[6]). As soon as there is more than one metadata schema involved it is likely that semantic conflicts occur. The common approach to deal with this issue is to specify *operational mappings* between schema elements. Since (semi-)automatic schema matching is another highly active research topic, we assume that mapping tools and solutions are available and we refer to two surveys [2, 3] covering this topic. In fact, in the digital library domain such mappings are called *crosswalks* and are available at least for many standardized schemas[7].

Figure 1 illustrates semantic heterogeneity among two metadata records corresponding to distinct metadata schemas: the first record represents metadata about the book "Hamlet" by "William Shakespeare" using elements from the FRBR standard. The element "Manifestation.Title" for instance

[1] The Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH), http://www.openarchives.org/oai

[2] Z.39.50 Gateway: http://www.loc.gov/z3950/gateway.html

[3] MARC 21 Format for Bibliographic Data, http://www.loc.gov/marc/status.html

[4] Dublin Core Metadata Initiative, http://dublincore.org/

[5] CIDOC Conceptual Reference Model (CRM), http://cidoc.ics.forth.gr/

[6] Functional Requirements for Bibliographic Records, http://www.ifla.org/VII/s13/frbr/frbr.htm

[7] http://www.ukoln.ac.uk/metadata/interoperability/

represents what is commonly referred to as the "title" of a book. The other record stores the same information using the much more comprehensive MARC-21 schema. In this case, the field with identifier "100a" is used to represent the title. The mapping table shows the crosswalk definitions between the FRBR and the MARC-21 schema.

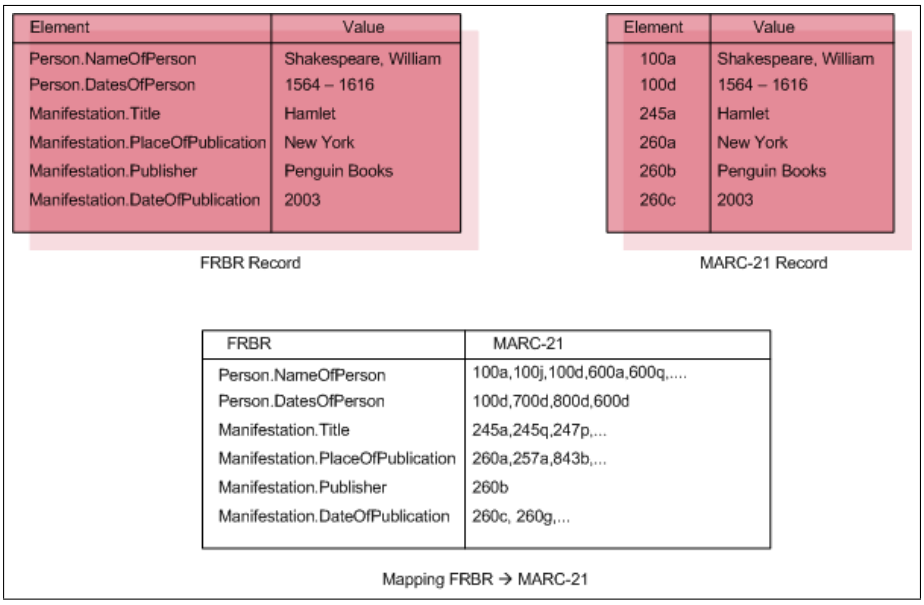

Figure 1. Sample Record in FRBR and MARC-21

Clients usually use a certain metadata schema for formulating their queries. If for instance an application provides a graphical query interface for FRBR, all queries will contain elements from the FRBR schema. It is the system's task to translate between queries formulated over a chosen schema into queries over other source-specific schemas.

2.3. Location Transparancy

The metadata themselves may not be replicated or moved to other systems but must remain in the institutions' systems. However, for a client which queries multiple systems the location of these systems must be transparent. As a technical prerequisite we can assume here that each data source is accessible via a certain Internet or domain specific protocol.

Transparent access to decentralised data sources requires *discovering* and *retrieving* information from relevant sources without client intervention. Factors like connection speed or the technical capabilities of a data source can affect query response time so the integration system must take these factors into account when formulating a query plan.

3. A Service-Oriented Metadata Integration Approach

The overall goal of our metadata integration architecture is to provide a homogeneous view and uniform access to a multitude of autonomous data sources. *Mediated Query Systems* are a well established design pattern [4] for architectures that should integrate metadata with all kinds of structural properties (unstructured, semistructured, structured) from heterogeneous sources. In our approach we apply this pattern and extend it with requirements of a decentralized service oriented integration architecture.

3.1. Architecture Overview

Our integration framework comprises the two main architectural components that are typical for a mediated query system: *wrappers* and *mediators*. Additionally, we employ an *integration registry* that is aware of available components and their technical as well as semantic query capabilities. All components in our integration architecture expose their functionality via a Web Service interface. Since Web Services allow platform-independent and location-transparent access to SOA components, we can achieve a certain degree of technical interoperability by using this technology.

3.1.1. Wrappers

Wrappers encapsulate local data sources and export their functionalities and the metadata stored therein. They accept queries in a certain query language and return metadata in a unified form. The semantics of metadata is expressed in terms of an *exported schema* and it is the wrapper's task to translate between the exported schema and native data descriptions.

It is well known [5, 6] that using ontologies for modeling the semantics of data is a suitable approach to cope with semantic heterogeneity. By describing the meanings of terms and the relationship between those terms we can make the semantics of data explicit and expose an exported schema. The benefits of ontology-based over traditional schema-based approaches include knowledge-reuse, improved maintainability, and the possibility to apply automated inference to derive new information [7, 8]. However, within a mediated query system we must agree on a common language for modeling the semantics of data and a common query language, which in turn requires a common data model.

By creating wrappers we can lift the involved data sources to a common technical level which in turn helps us to overcome the technical heterogeneities among systems without changing the existing systems.

3.1.2. Mediators

Mediators handle queries from the application layer, unfold them into sub-queries, disperse them to local data sources where they are executed, and finally combine and present the results to the client. A mediator defines a *mediation schema* which is exposed to the client and can be used for formulating queries. The process of unfolding queries relies heavily on previously defined *operational mappings* between the mediation and the exported schemas exposed by the wrappers. Mappings are part of a more extensive *integration specification* which contains all the information required for the query unfolding process. Integration specifications are plug-able and used to set up a mediator; the mediator itself is a generic component.

3.1.3. Integration Registry

Each component that is part of the integration architecture is registered with the integration registry. By introducing such a registry we not only ensure that data can be discovered but also enforce a certain community aspect in metadata integration by allowing institutions or organisations that elaborate an integration scenario to publish their integration specifications. Other institutions can re-use these specifications and the integration registry could derive additional operational mappings from existing ones.

3.2. Workflow

In the following we describe the integration workflow using the example from Section 2.2. and assume that a client requires search functionalities on multiple institutions based on the FRBR metadata schema. To simplify the scenario, we focus on a single wrapper which encapsulates a relational database containing MARC-21 metadata. Figure 2 illustrates the workflow.

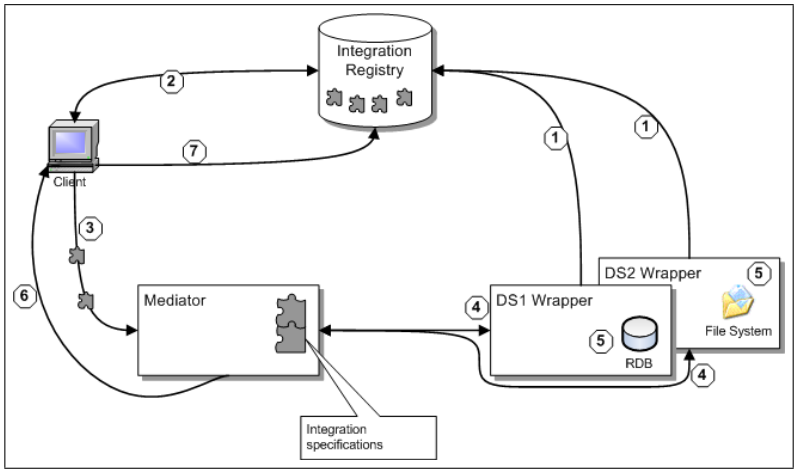

Figure 2. Metadata Integration Workflow

1. each wrapper that is built to expose metadata from a certain institution is registered at the integration registry. It publishes a basic profile, which is a human readable description, an interface description and the wrapper's exported schema. In our example the published profile contains the MARC-21 schema.

2. a client who wants to query multiple autonomous sources using a specific mediation schema (in our case FRBR) contacts the integration registry and checks if there are existing integration specifications for the wrappers to be considered. In case that there is no specification available for a certain wrapper, it must be defined.

3. the client uses the integration specification to set up the generic mediator component, expresses a query over the mediation schema and executes it at the mediator.

4. the mediator rewrites the query over the mediation schema into queries over the exported schema. This requires that the operational mappings between the mediation (FRBR) and the exported schema (MARC-21) are available as part of the integration specification. Based on the mappings, the mediator unfolds the original query into rewritten sub queries and forwards them to wrappers.

5. the wrapper receives a query from the mediator and answers this query from the data available in the encapsulated data source. Since in our example, the data source is a relational database, the wrapper must translate the mediator's query into a native SQL query.

6. the mediator collects the query results that are returned from the wrappers, combines them and returns them to the client in a uniform way.

7. for reusability purposes, the client publishes the integration specification in the integration registry.

4. Design Details

4.1. Standards-based Metadata Integration

As we have seen in the previous section, building a mediated metadata integration system requires a common ontology language, a common data model and a query language that operates on this data model. We believe that the usage of standards in integration scenarios is a major step forward towards interoperability, at least on a technical level. Therefore we have decided to rely on: the *Resource Description Framework (RDF)*, the *Web Ontology Language (OWL)*, and *SPARQL*.

The main design goal of RDF [9] is to provide a framework for representing metadata about arbitrary resources in a way that machines can exchange and "understand" the meanings. Another often unregarded aspect of RDF is that is was also designed for metadata integration and aggregation[8] purposes. The RDF data model itself is simply a directed labeled graph. Such a model manages to be simple, and yet powerful enough to allow the description of metadata of any kind originating from various heterogeneous sources [10].

OWL [11] is a language for modeling ontologies, which provides all the constructs and the expressiveness to describe the semantics of data. Although it was originally not intended to use OWL for data integration, we can benefit from it's popularity, which is reflected in the availability of various OWL-specific tools.

Providing uniform access to multiple heterogeneous metadata sources requires a single query language which is understood by all system components. Recently, SPARQL [12], a query language which operates on the RDF data model, has been specified. Given that wrappers can translate between RDF and native data models, we can use this query language for accessing the data sources in an integrated fashion. Therefore, in our architecture each wrapper and mediator exposes a SPARQL query interface in terms of a Web Service definition.

4.2. Integration Specification

Integration specifications are the building blocks required by the mediator in order to integrate with a given data source. A single specification contains all the information a mediator requires for rewriting and forwarding queries from the client. Hence, in addition to technical information like the wrapper's access point the specification must cover two main aspects: *semantic mismatches* between the source and the mediation schema and *different contents* in the data sources.

Semantic mismatches occur whenever the mediation and an export schema are different; in our sample scenario the mediation schema is FRBR while the exported schema of a data source is MARC-21. Of course, a prerequisite for handling semantic mismatches is to know semantic mappings between two schemas and to express them in a machine processable way. As already mentioned in Section 2.2. we can assume that in the digital library domain these mappings are partially available.

Furthermore, it is likely that two data sources, even if they expose the same schema, do not contain the same contents. It might happen that queries simply cannot be answered by all data sources. Hence, the mediator needs a mechanism to determine if a certain wrapper is relevant to a query.

[8]The W3C RDF Data Access Working Group is working on this issue, has defined data access use cases and has released a candidate recommendation of the SPARQL query language in April 2006

In order to provide for both aspects, we follow the approach of parameterized views [13] and define a *capability description* as part of an integration specification. A capability description contains a set of *query templates*, each defining how to rewrite the conditions of a query over the mediation schema into conditions over the exported schema. A query transformer, which is part of the mediator, takes the capability description and unfolds the original query by substituting its conditions. In this way templates reflect the operational mappings as well as the set of possible queries and the mediator can handle semantic mismatches and determine if a data source is relevant to a query.

Since we apply SPARQL, which is a graph pattern matching language, as query language, we can regard templates as translation specifications between triple patterns. With "CONSTRUCT" queries, SPARQL defines a mechanism which allows us to express such templates.

Figure 3 demonstrates a query over the FRBR mediation schema, which asks for the title of all books written by William Shakespeare. Figure 4 illustrates a part of SPARQL query template which has been defined according to the crosswalks between FRBR and MARC-21.

```
SELECT ?t
WHERE {
  ?x frbr:Title ?t
  ?x frbr:NameOfPerson "Shakespeare, William"
}
```

Figure 3. Incoming query over the mediation schema (FRBR)

```
CONSTRUCT {?x frbr:Title ?y}
WHERE {
  ?x marc:245a ?y
  ...
}
```

```
CONSTRUCT {?x frbr:NameOfPerson ?y}
WHERE {
  ?x marc:100a ?y
  ...
}
```

Figure 4. SPARQL query templates

The templates we defined here are fairly simple because there exists a direct one-to-many mapping between each attribute of the mediation and the exported schema. In reality, integration scenarios are much more complex. For instance it is likely that a metadata description in schema A requires a different number of elements than a description in schema B to express the same information. In such a case, we must extend the SPARQL template mechanism to work with functions operating on certain data types. For instance, we can apply a function *concat* or even a more complex ones (e.g. *if-else* conditions) in a template specification to virtually create the values for an element of one schema from one or more elements from another schema. Figure 5 demonstrates how to create the value for an element "fullname" in schema s1 from the values of two elements in schema s2.

```
CONSTRUCT {?x s1:fullName ?NAME}
WHERE {
  ?x s2:firstName ?first
  ?x s2:lastName ?last
  (?first ?last) op:concat ?NAME

}
```

Figure 5. More complex SPARQL query template including operator

5. Related Work

As in other approaches [14], we apply ontologies to model the semantics of metadata. In contrast to other systems we do not follow a *single ontology* approach where one global ontology provides a shared vocabulary for the sources, but rather let the client choose a mediation schema for formulating queries. Since the mediation and the source ontologies can be different from each other, we rely on operational mappings provided by the user.

Query rewriting algorithms, in particular view-based approaches, originate from the database domain [15] and have been developed further for semi-structured data models in [16]. Ontology based query rewriting algorithms have been proposed by [17, 18], but both rely on quite simple mapping assertions between the mediation and the exposed ontologies.

In addition to mappings between the different schemas used in a system, wrappers in our integration approach also depend on mappings between the exported ontologies and the information in the data sources. Because we rely on RDF, which is a de-facto standardized model, we can rely on existing approaches, each operating on a particular type of data source: [19] shows how to define mappings between relational databases and Semantic Web technologies, and [20, 21] provide solutions for querying these databases using SPARQL. An approach for integrating XML data has been proposed by [22].

6. Conclusions and Future Work

In this paper, we have described an ontology-based approach for the integration of metadata from various heterogeneous digital library systems. We have elevated the architecture of traditional mediated query systems to the level of service-oriented architectures and defined them as Web Service components. The next step will be to implement a metadata integration framwork based on this approach.

Determining ontology mappings is probably the most difficult task. In the digital library domain we have the advantage that we can rely on the existence of predefined metadata crosswalks, which are available at least for standardized schemas. We believe that the integration of non-standardized schemas - especially in productive environments - remains an intellectual task which of course can be supported by the output of state-of-the-art ontology mapping algorithms. However, especially for more complex schemas we will need more powerful mapping capabilities such as complex functions that operate on instance level values and various datatypes. Currently we are working on a language which allows us to express such mappings.

In our approach we rely on templates to describe the capabilities of a data source and to rewrite queries coming from the mediator. For larger ontologies writing such templates manually does not scale. Therefore we are working on a semi-automatic wrapper generator which allows us to generate query templates from previously defined ontology mappings.

7. Acknowledgments

The work reported in this paper was partly funded by the Austrian Federal Ministry of Economics and Labour and the European Union as part of the 6th Framework Program.

—-

References

[1] Amit P. Sheth and James A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Comput. Surv.*, 22(3):183–236, 1990.

[2] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350, 2001.

[3] Yannis Kalfoglou and Marco Schorlemmer. Ontology mapping: the state of the art. *Knowl. Eng. Rev.*, 18(1):1–31, 2003.

[4] Ruxandra Domenig and Klaus R. Dittrich. An overview and classification of mediated query systems. *SIGMOD Rec.*, 28(3):63–72, 1999.

[5] Tom Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisitions*, 5:199–220, 1993.

[6] Natalya F. Noy. Semantic integration: a survey of ontology-based approaches. *SIGMOD Rec.*, 33(4):65–70, 2004.

[7] Natalya F. Noy and Michel Klein. Ontology evolution: Not the same as schema evolution. *Knowl. Inf. Syst.*, 6(4):428–440, 2004.

[8] Michael Uschold and Michael Gruninger. Ontologies and semantics for seamless connectivity. *SIGMOD Rec.*, 33(4):58–64, 2004.

[9] W3C Semantic Web Activity – RDF Core Working Group. Resource Description Framework (RDF), 2004.

[10] Yannis Papakonstantinou, Hector Garcia-Molina, and Jennifer Widom. Object exchange across heterogeneous information sources. pages 251–260, 1995.

[11] W3C Semantic Web Activity – Web Ontology Working Group. Web Ontology Language (OWL), 2004.

[12] W3C Semantic Web Activity – RDF Data Access Working Group. SPARQL Query Language for RDF, 2006.

[13] Yannis Papakonstantinou and Vasilis Vassalos. Query rewriting for semistructured data. In *SIGMOD '99: Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, pages 455–466, New York, NY, USA, 1999. ACM Press.

[14] H. Wache, T. Vögele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Hübner. Ontology-based integration of information — a survey of existing approaches. In H. Stuckenschmidt, editor, *IJCAI–01 Workshop: Ontologies and Information Sharing*, pages 108–117, 2001.

[15] Alon Y. Halevy. Answering queries using views: A survey. *The VLDB Journal*, 10(4):270–294, 2001.

[16] Serge Abiteboul. Querying semi-structured data. pages 1–18, 1997.

[17] Huiyong Xiao and Isabel F. Cruz. Ontology-based query rewriting in peer-to-peer networks. In *In Proceedings of the 2nd International Conference on Knowledge Engineering and Decision Support, 2006*, 2006.

[18] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. What to ask to a peer: Ontology-based query reformulation. In *Proc. of the 9th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2004)*, pages 469–478, 2004.

[19] Eric Prud'hommeaux. Optimal RDF access to relational databases, aug 2004. http://www.w3.org/2004/04/30-RDF-RDB-access/.

[20] Christan Bizer and Andy Seaborne. D2RQ - Treating non-RDF databases as virtual RDF graphs, 2004. http://www.wiwiss.fu-berlin.de/suhl/bizer/D2RQ/.

[21] Damian Steer. SquirrelRDF - Querying existing SQL data with SPARQL. http://jena.sourceforge.net/SquirrelRDF/.

[22] Patrick Lethi and Peter Frankhauser. XML data integration with OWL: Experiences and Challenges. In *2004 Symposium on Applications and the Internet (SAINT 2004)*, pages 160–170, 2004.