Ontology Driven Data Integration in Heterogeneous Networks*

Isabel F. Cruz Huiyong Xiao

ADVIS Lab
Department of Computer Science
University of Illinois at Chicago, USA
{ifc | hxiao}@cs.uic.edu

Abstract

We propose a layered framework for the integration of syntactically, schematically, and semantically heterogeneous networked data sources. Their heterogeneity stems from different models (e.g., relational, XML, or RDF), different schemas within the same model, and different terms associated with the same meaning. We use a semantic based approach that uses a global ontology to mediate among the schemas of the data sources. In our framework, a query is expressed in terms of one of the data sources or of the global ontology and is then translated into subqueries on the other data sources using mappings based on a common vocabulary. Metadata representation, global conceptualization, declarative mediation, mapping support, and query processing are addressed in detail in our discussion of a case study.

1 Introduction

Data integration refers to the ability to manipulate data transparently across multiple data sources, such as networked data sources on the web. The concept of data integration is part of the broader concept of interoperability

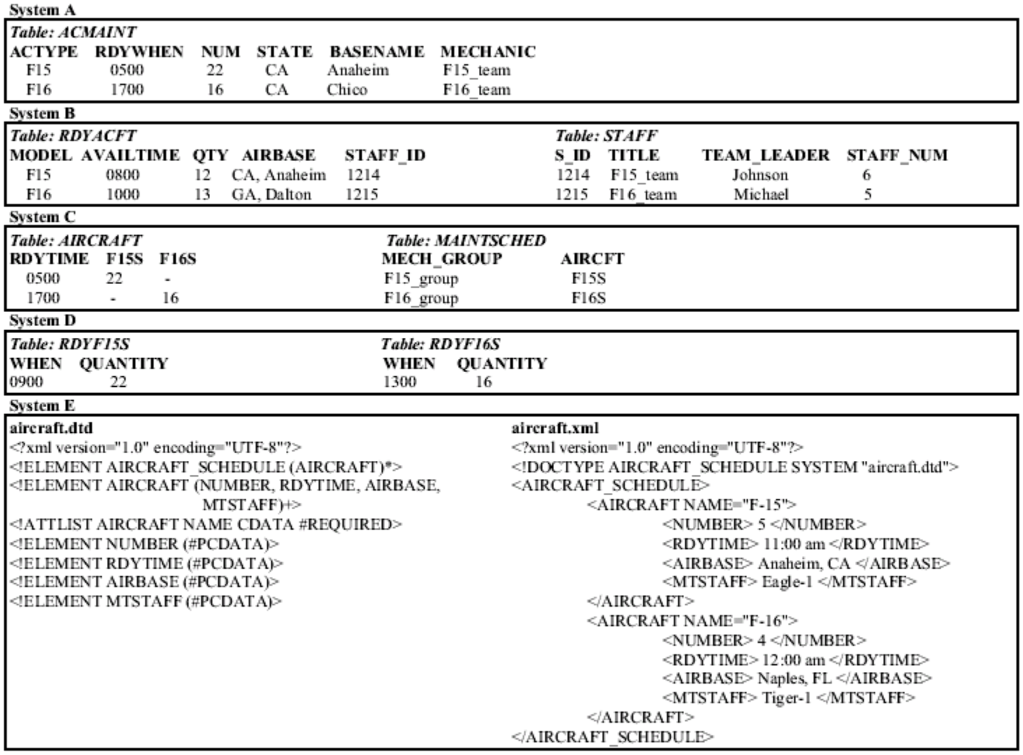
*This research was supported in part by the National Science Foundation under Awards ITR IIS-0326284, IIS-0513553, and IIS-0812258.

among systems, services, or programs [33]. However, because of their increasing number, reference models and standards that have been developed to enable interoperability stand paradoxically in the way of their intended goals. To address this problem, common metamodels and mappings have been proposed [44]. Metamodels support abstraction and generalization that aid in identifying problems that can use the same solutions. Mappings provide bridges between those alternative solutions. A compelling parallel has been made between models and metamodels: "What metadata and mediation services enable for data, metamodels and mappings can provide for models." [44]. The migration of solutions to new problems depends heavily on the ability to use metamodels and in particular on using a common layered metamodel to organize and map existing standards. An example of a layered model is the ISO/OSI model for networking, where layers support clearly defined functions and interfaces. It is in this structure that a solution can be devised.

The Levels of Conceptual Interoperability Model (LCIM) has been devised to measure the level of interoperability between systems: Level 6 (of maximum interoperability) is labeled "Conceptual Interoperability." The remaining levels (in decreasing order of interoperability that they support) are respectively labelled "Dynamic Interoperability," "Pragmatic Interoperability," "Semantic Interoperability," "Syntactic Interoperability," and "Technical Interoperability." Tolk *et al.* recognize the need of using ontologies for defining the meaning of data and of processes in a system and propose a ontology-based approach for the different layers of LCIM [45].

In this paper, we follow an approach that follows closely the objectives, if not the methods, of the above approaches. First, we focus on data operation and therefore on interoperability. Second, we distinguish levels of interoperability, with Semantic Interoperability being the highest level obtained thus far. Third, our structure for interoperability is not only layered but also follows closely the ISO/OSI model for networking. Fourth, we consider several types of data heterogeneity and deal with all of them within the same framework. Fifth, we deal with different data representation standards and would be able to accommodate emerging standards. Finally, we use an approach where ontologies play several roles. In particular, we use a metamodel (conceptually an ontology) to bridge across the ontologies that model the different interoperating data sources.

In the rest of the paper we focus on the particular problem of integrating data sources that can be heterogeneous in syntax, schema, or semantics, thus making data integration a difficult task [10]. More specifically, syntactic heterogeneity is caused by the use of different data models (e.g., relational, XML, or RDF). Within the same model, the structural differences between two schemas lead to schematic heterogeneity. Semantic heterogeneity results from different meanings or interpretations of data that may arise from various contexts.



System A

Table: ACMAINT

ACTYPE	RDYWHEN	NUM	STATE	BASENAME	MECHANIC
F15	0500	22	CA	Anaheim	F15_team
F16	1700	16	CA	Chico	F16_team

System B

Table: RDYACFT

MODEL	AVAILTIME	QTY	AIRBASE	STAFF_ID
F15	0800	12	CA, Anaheim	1214
F16	1000	13	GA, Dalton	1215

Table: STAFF

S_ID	TITLE	TEAM_LEADER	STAFF_NUM
1214	F15_team	Johnson	6
1215	F16_team	Michael	5

System C

Table: AIRCRAFT

RDYTIME	F15S	F16S
0500	22	-
1700	-	16

Table: MAINTSCHED

MECH_GROUP	AIRCFT
F15_group	F15S
F16_group	F16S

System D

Table: RDYF15S

WHEN	QUANTITY
0900	22

Table: RDYF16S

WHEN	QUANTITY
1300	16

System E

aircraft.dtd

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT AIRCRAFT_SCHEDULE (AIRCRAFT)*>
<!ELEMENT AIRCRAFT (NUMBER, RDYTIME, AIRBASE,
                    MTSTAFF)+>
<!ATTLIST AIRCRAFT NAME CDATA #REQUIRED>
<!ELEMENT NUMBER (#PCDATA)>
<!ELEMENT RDYTIME (#PCDATA)>
<!ELEMENT AIRBASE (#PCDATA)>
<!ELEMENT MTSTAFF (#PCDATA)>
```

aircraft.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE AIRCRAFT_SCHEDULE SYSTEM "aircraft.dtd">
<AIRCRAFT_SCHEDULE>
        <AIRCRAFT NAME="F-15">
                <NUMBER> 5 </NUMBER>
                <RDYTIME> 11:00 am </RDYTIME>
                <AIRBASE> Anaheim, CA </AIRBASE>
                <MTSTAFF> Eagle-1 </MTSTAFF>
        </AIRCRAFT>
        <AIRCRAFT NAME="F-16">
                <NUMBER> 4 </NUMBER>
                <RDYTIME> 12:00 am </RDYTIME>
                <AIRBASE> Naples, FL </AIRBASE>
                <MTSTAFF> Tiger-1 </MTSTAFF>
        </AIRCRAFT>
</AIRCRAFT_SCHEDULE>
```

Figure 1: Five syntactically, schematically, and semantically heterogeneous legacy databases for aircraft maintenance.

To demonstrate our approach we have adapted an example that considers legacy databases for aircraft maintenance scheduling [38] to illustrate syntactic, schematic, and semantic heterogeneities, which is shown in Figure 1. In this example, syntactic heterogeneity stems from data that is stored in relational and XML databases. The example also illustrates schematic heterogeneity in that the schemas of the relational databases differ substantially. For example, aircraft models are either table names, attribute names, or attribute values. Semantic heterogeneity is also apparent: "RDYF15S" (table name in System D), "F15S" (attribute name in System C), and "F15" (attribute value in System B) relate to the same kind of airplane.

In this paper, we achieve integration among syntactically, schematically, and semantically heterogeneous network data using an approach that is based on the layered model proposed by Melnik and Decker [31]. Their model, which is inspired by the ISO/OSI layered networking model, consists of four layers: the syntax, object, semantic, and application layers. This paper expands on our preliminary work [16].

We follow a *hybrid* approach, which is embodied in the semantic layer. In particular, we associate an ontology with each networked data source, called henceforth *local ontology*, and create a *global ontology* to mediate across the data sources. The global ontology relies on a common vocabulary shared by all data sources [47]. Considering the fundamental role of RDF as an ontology language, we express all the data models in our approach using RDF[1] and RDF Schema. [2] Queries are propagated across the semantic layers of the data sources and are expressed using a RDF query language. We use the RDF Query Language, RQL [25], but any other language for RDF could have been used, including SPARQL. [3] The queries are expressed using mappings established between pairs of ontologies, one called the *source* ontology and the other one the *target* ontology.

The rest of the paper is organized as follows. In Section 2 we describe the different ways under which ontologies can be used and the different roles they play in data integration. Our layered approach is presented in Section 3. In this section, we describe the different layers and the overall architecture of a system that integrates data from the networked sources. To demonstrate our layered approach, in Section 4 we expand the legacy example we have just introduced in light of the different kinds of roles that ontologies play. Section 5 describes in detail query processing across different data sources and lists the kinds of assumptions we have made. We discuss related work in Section 6 and finally draw some conclusions in Section 7.

[1] http://www.w3.org/RDF/.

[2] http://www.w3.org/TR/rdf-schema/.

[3] http://www.w3.org/TR/rdf-sparql-query/.

2 Ontologies in Data Integration

We call *semantic data integration* the process of using a conceptual representation of the data and of their relationships to eliminate heterogeneities. At the heart of this process is the concept of *ontology*, which is defined as an explicit specification of a shared conceptualization [21, 22].

Ontologies are developed by people or organizations to facilitate knowledge sharing and reuse [23]. For example, they can embody the semantics for particular domains, in which case they are called *domain ontologies*. Ontologies are semantically richer than traditional database schemas. In what follows we describe data integration architectures in terms of database schemas. Later we describe how ontologies can be used instead of database schemas.

We distinguish two data integration architectures: one that is central [2, 5, 12, 18, 32, 46] and the other one that is peer-to-peer [4, 8, 9, 19, 24, 34]. A central data integration system has a global schema, which provides the user with a uniform interface to access information stored in the data sources by means of queries posed in terms of the global schema [28]. In contrast, in a peer-to-peer data integration system, any peer (a data source) can accept user queries to access information in other peers.

To enable data integration in a central data integration system, mappings need to be created between the global schema and the data source schemas. In a peer-to-peer data integration system mappings are established between peers. The two main approaches to building such mappings are Global-as-View (GaV) and Local-as-View (LaV) [28, 46]. In the GaV approach, every entity in the global schema is associated with a view over the data sources. Therefore querying strategies are simple because the mappings are explicitly defined. However, every time that there is a change to the data sources, it could change the views. In contrast, the LaV approach allows for changes to the data sources that do not affect the global schema, since the local schemas are defined as views over the global schema. However, query processing is more complex.

Now that we have introduced the main concepts behind data integration, we describe the different forms in which ontologies can intervene [47]:

Single ontology approach. All source schemas are directly related to a shared global ontology, which provides a uniform interface to the user. However, this approach requires that all sources have similar characteristics, for example the same level of granularity. An example of a system that uses this approach is SIMS [5].

Multiple ontology approach. Each data source is described by its own local ontology. Instead of using a global ontology, local ontologies are mapped to one another. For this purpose, an additional representation formalism is necessary for defining the inter-ontology mappings. The OBSERVER system is an example of this approach [32].

Hybrid ontology approach. A combination of the two preceding approaches is used. First, a local ontology is built for each source schema, which is mapped to a global ontology. New sources can be easily added with no need for modifying existing mappings. Our layered framework is an example of this approach.

The single and hybrid approaches are appropriate for building central data integration systems, the former being more appropriate for GaV systems and the latter for LaV systems. A peer-to-peer system, where a global ontology exists in a "super-peer" can also use the hybrid ontology approach [19]. However, the multiple ontology approach is better suited to "pure" peer-to-peer data integration systems, where there are no super-peers.

We identify the following five roles of ontologies in a data integration process [17]:

Metadata representation. Each source schema can be explicitly represented by a local ontology. All ontologies use the same representation language and are therefore syntactically homogeneous.

Global conceptualization. The global ontology provides a conceptual view over the schematically heterogeneous source schemas.

Support for high-level queries. The global ontology provides a high-level view of the sources. Therefore, a query can be formulated without specific knowledge of the different data sources. The query is then rewritten into queries over the sources, based on the semantic mappings between the global and local ontologies.

Declarative mediation. A hybrid peer-to-peer system uses the global ontology as a mediator for query rewriting across peers.

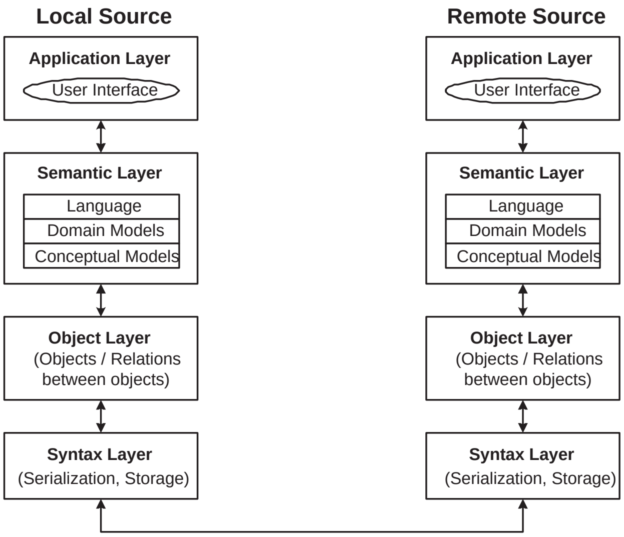


Figure 2: The layered model.

Mapping support. A common thesaurus or vocabulary, which can be formalized as an ontology, can be used to facilitate the automation of the mapping process.

In the following sections, we further elaborate on these five roles using our case study to exemplify them.

3 A Layered Data Interoperability Model

3.1 Overview

In this section, we present the layered approach for data interoperability proposed by Melnik and Decker [31], which we show in Figure 2. Of the four proposed layers, we concentrate on the semantic layer and identify three sublayers that are contained in it.

Application Layer. The application layer is used to express queries. For example, a visual user interface may be provided in this layer for users to submit their queries. Ideally, query results are integrated and shown to the users, so as to give the appearance that the distributed databases interoperate seamlessly.

Semantic Layer. This layer consists of three cooperating sublayers that accomplish different tasks:

- Languages. The main purpose of this sublayer is to accept the user queries and to interpret them so that they can be understood by the other interoperating systems. The language can be highly specialized or be a general purpose language.
- Domain Models. They include the ontologies for a particular application domain and can be different from one another even for the same domain [29]. Examples of domains include transportation, manufacturing, e-business, digital libraries, and aircraft maintenance.
- Conceptual Models. They model concepts, relationships and constraints using constructs such as generalization, aggregation, or cardinality constraints. RDF Schema and UML Foundation/Core are two examples.

Object Layer. The purpose of the object layer is to give an object-oriented view of the data to the application. This layer enables manipulations of objects and binary relationships between them. Every object in a data schema is mapped to a particular class. The object layer also forwards all the information that it receives from the syntax layer to the semantic layer.

Syntax Layer. The main purpose of the syntax layer is to provide a way of specifying both the semantic and object layer information using a common representation. XML has been used to represent RDF and RDF Schema. This layer is also responsible for data serialization and for mapping the queries from the object layer to the XML representation of the data schemas. The information that is extracted by the queries is returned to the object layer.

3.2 Architecture

Based on the layered approach discussed above, we concentrate now on the semantic layer. The semantic layer accepts user queries from the application layer and processes them. We use a hybrid ontology approach and RDF Schema both for the conceptual model of the data sources and for the domain model, as expressed by means of a global ontology. Given our choice of RDF and RDF Schema, a query language for RDF is appropriate. We chose RQL (RDF Query Language) [25] for the language sublayer.

The application layer can either support RQL or an application specific user interface. As for the implementation of the object layer and of the syntax layer, we use RSSDB (RDF Schema Specific Database), which is an RDF Store that uses schema knowledge to automatically generate an Object-Relational (SQL3) representation of RDF metadata and to load resource descriptions [1].

Figure 2 shows the architecture that is used to build applications based on our layered approach. The whole process can be further divided into three sub-processes as follows:

Constructing local ontologies. A component called *Schema Integrator* is used to transform source schemas and source data automatically into a single conceptual schema, expressed using a *local ontology.* We consider relational, XML, and RDF databases.

Mapping. We use a global ontology (using RDF Schema) to serve as the mediator between different local ontologies, each of which is mapped to the global ontology. We utilize a common vocabulary (which also uses RDF Schema) to facilitate this mapping process. This process is usually semi-automatic in that it may require input from users.

Query processing. When the user submits a query to the local ontology, the query will be executed directly over the local RSSDB. This process is called *local query processing.* Queries are transformed from one local ontology to all the other local ontologies by a query rewriting algorithm. This process, which is based on the mappings between the global ontology and the local ontologies, is called *remote query processing*; it can be performed automatically under some simplifying assumptions. The answers to both local query processing and remote query processing are assembled and returned to the user.

4 Case Study

In this section, we describe in detail the process for semantic data integration as illustrated by the case study already introduced in Section 1 and displayed

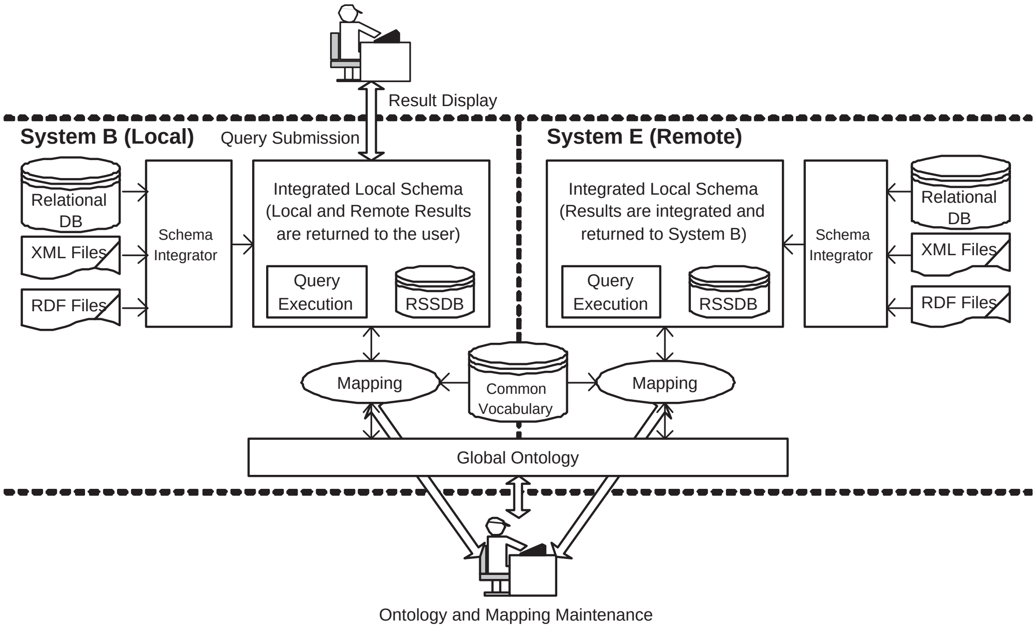


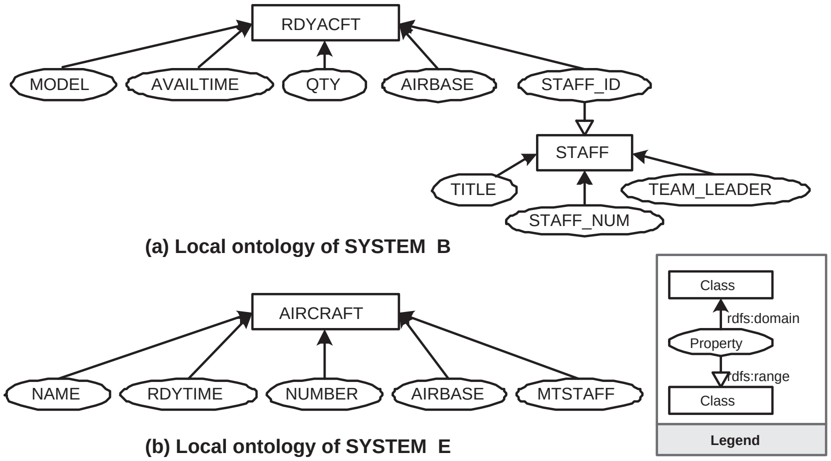
Figure 3: Architecture of the semantic layer.

in Figure 1. We also discuss the various ontology roles of Section 2 in the context of the case study.

4.1 Construction of Local Ontologies

The construction of a *local ontology* includes two phases. In the first phase, we integrate all source schemas into a single local ontology, which is represented in RDF Schema and stored in RSSDB. In the second phase, we transform data from the original relational or XML databases into RDF files, which are then also stored in RSSDB. When integrating RDF databases, this step is not needed.

When the underlying schema is relational, we analyze the attributes of each table and the dependency relationships between every pair of tables through their foreign keys. In particular, if two tables are connected using a foreign key, we create a schema tree for each table. The root of each tree represents one table with type *rdfs:Class* and its children represent the attributes of the table with type *rdf:Property*. Therefore, the arcs connecting the root and its children are *rdfs:domain* arcs. Then we connect two trees



(a) Local ontology of SYSTEM B

(b) Local ontology of SYSTEM E

Figure 4: Local ontology for legacy source schemas.

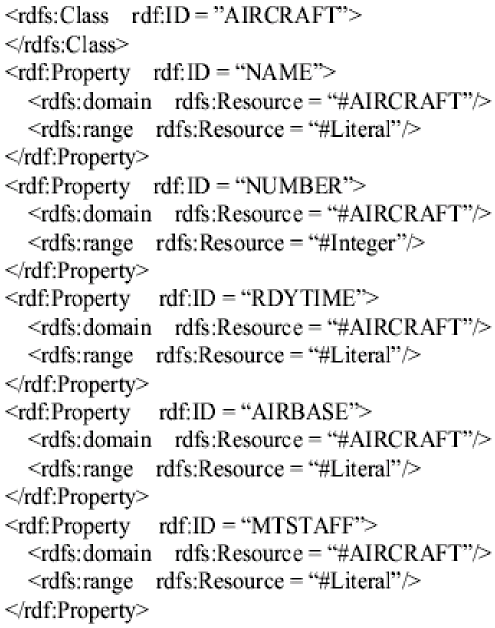
using an *rdfs:range* edge from the node corresponding to the foreign key to the root of the other tree. Hence, we obtain the local ontology that can be used by the system for data interoperation. An example is shown for System B where *STAFF_ID* is a foreign key, and the local ontology is shown in Figure 4(a).

In the case of System D, where there is no foreign key between two tables, we also create a schema for each table. We then build a new root with type *rdfs:Class* and two new nodes of type *rdf:Property*, one for each table as children of this new root. Then we connect each of these nodes to one of the nodes representing the tables using an arc of type *rdfs:range*.

If a system uses XML files, we analyze the DTD file corresponding to each XML file and find all the element and attribute tags and their hierarchy, which we use to construct the local ontology for the system. Figure 4(b) shows the schema tree for System E. In this way, each of the participating systems have their local schemas expressed as a single RDF Schema local ontology and data stored using RSSDB. Figure 5 gives a fragment of the underlying RDF Schema representation of the local ontology of System E.

Based on the local ontology, we consider two cases when transforming the data from a source schema into a local ontology.

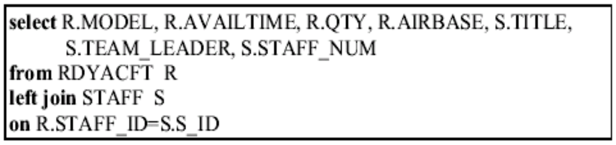
Relational database. We use the *left outer join* operation to unite the data from multiple tables that are connected through foreign keys. The table containing the foreign key acts as the left part of the left outer



```
<rdfs:Class   rdf:ID = ”AIRCRAFT”>
</rdfs:Class>
<rdf:Property   rdf:ID = “NAME”>
   <rdfs:domain   rdfs:Resource = “#AIRCRAFT”/>
   <rdfs:range   rdfs:Resource = “#Literal”/>
</rdf:Property>
<rdf:Property   rdf:ID = “NUMBER”>
   <rdfs:domain   rdfs:Resource = “#AIRCRAFT”/>
   <rdfs:range   rdfs:Resource = “#Integer”/>
</rdf:Property>
<rdf:Property    rdf:ID = “RDYTIME”>
   <rdfs:domain   rdfs:Resource = “#AIRCRAFT”/>
   <rdfs:range   rdfs:Resource = “#Literal”/>
</rdf:Property>
<rdf:Property    rdf:ID = “AIRBASE”>
   <rdfs:domain   rdfs:Resource = “#AIRCRAFT”/>
   <rdfs:range   rdfs:Resource = “#Literal”/>
</rdf:Property>
<rdf:Property    rdf:ID = “MTSTAFF”>
   <rdfs:domain   rdfs:Resource = “#AIRCRAFT”/>
   <rdfs:range   rdfs:Resource = “#Literal”/>
</rdf:Property>
```

Figure 5: RDF Schema for the local ontology of system E.

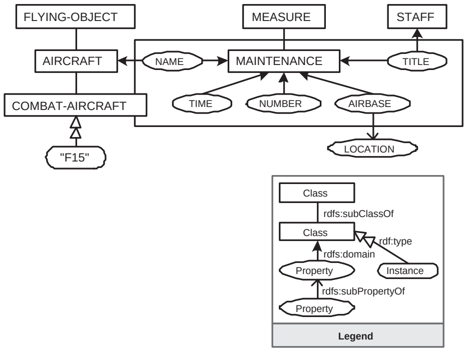
join operation. Figure 6 gives the example of a SQL query that expresses this transformation in System B. In the case where there is no foreign key relationship, we use a Cartesian product to realize the data transformation.



```
select R.MODEL, R.AVAILTIME, R.QTY, R.AIRBASE, S.TITLE,
       S.TEAM_LEADER, S.STAFF_NUM
from RDYACFT  R
left join STAFF  S
on R.STAFF_ID=S.S_ID
```

Figure 6: SQL query for data transformation in System B.

XML data. The data model uses XML Schema or a DTD. In both situations we use XSLT expressions to transform data from an XML file into an RDF file. After the data is saved into an RDF file, we can use RQL (or any of the APIs for RSSDB), to access the data.



```
<rdfs:Class rdf:ID="Maintenance">
   <rdfs:subClassOf rdfs:Resource="#Measure"/>
</rdfs:Class>
......
<rdf:Property rdf:ID="Name">
   <rdfs:domain rdfs:Resource="#Aircraft"/>
   <rdfs:domain rdfs:Resource="#Maintenance"/>
   <rdfs:range rdfs:Resource="#Literal"/>
   <rdfs:isDefinedBy rdfs:Resource="#Vocabulary_Aircraft"/>
</rdf:Property>
<rdf:Property rdf:ID="Time">
   <rdfs:domain rdfs:Resource="#Maintenance"/>
   <rdfs:range rdfs:Resource="#Date"/>
   <rdfs:isDefinedBy rdfs:Resource="#Vocabulary_ReadyTime"/>
</rdf:Property>
<rdf:Property rdf:ID="Number">
   <rdfs:domain rdfs:Resource="#Maintenance"/>
   <rdfs:range rdfs:Resource="#Integer"/>
   <rdfs:isDefinedBy rdfs:Resource="#Vocabulary_Number"/>
</rdf:Property>
<rdf:Property rdf:ID="Airbase">
   <rdfs:subPropertyOf rdfs:Resource="#Location"/>
   <rdfs:domain rdfs:Resource="#Maintenance"/>
   <rdfs:range rdfs:Resource="#Literal"/>
   <rdfs:isDefinedBy rdfs:Resource="#Vocabulary_Address"/>
</rdf:Property>
<rdf:Property rdf:ID="Title">
   <rdfs:domain rdfs:Resource="#Staff"/>
   <rdfs:domain rdfs:Resource="#Maintenance"/>
   <rdfs:range rdfs:Resource="#Literal"/>
   <rdfs:isDefinedBy rdfs:Resource="#Vocabulary_Machinist"/>
</rdf:Property>
......
```

Figure 7: The global ontology and its RDF Schema description.

4.2 Mapping Process

The global ontology is used for mediating among distributed schemas [29]. For this purpose, we set up the relationships between the global ontology and the local ontologies based on a common vocabulary. Figure 7 shows a fragment of the global ontology for the domain of aircraft maintenance and its partial RDF Schema representation. We focus on the class and properties that are contained in the box. We use *rdfs:isDefinedBy*, which is an RDF property, to make the connection between a class or a property and the associated vocabulary.

An important component of our approach is that all local ontologies share the common dictionary with the global ontology (see Figure 8). This dictionary stores the common vocabulary of all the concepts and the relationships between the global ontology and each local ontology. When presented with a new local ontology that needs to be mapped to the global ontology, the

AIRCRAFT –	Flighttype, Actype, Model, AC_type, FlightModel, F<number>S, Aeroplane, F-<number>, F_<number>, Airplane, Model_Flight
NUMBER –	Num, Qty, Quantity, Num_ready, ReadyNumber, Many, Size, Rdynumber, QuantityReady, Rdyqty
READYTIME –	Rdywhen, Availtime, When, RdyTime, ReadyTime, FinishRepair, Ready, CompletionTime, PickupTime, DeliveryTime
ADDRESS –	LocationRdywhen, AirportAddr, Airbase, Location, AirportLocation
MACHINIST –	Craftsman, Mechanic, Repairman, Technician, Staff

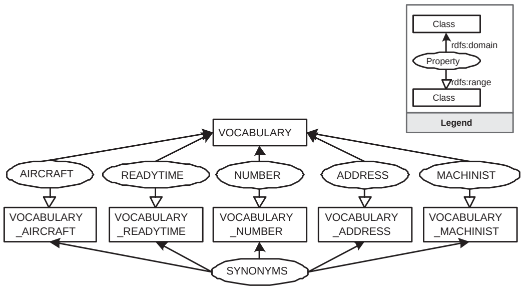


Figure 8: The common vocabulary.

system checks every concept name against the dictionary to obtain an appropriate matching for that concept and chooses the optimal one according to a score function. The construction of a dictionary and the determination of an appropriate matching is an important area of research [37]. In this paper, we consider that the dictionary has a simplified structure, in that it only supports one-to-one total mappings.

The components that participate in the mapping process (namely the local ontologies, the common vocabulary, and the global ontology) are all represented using RDF Schema. The mapping between the global ontology and a local ontology is realized according to the common vocabulary. Figure 9 shows the local ontologies of the five legacy systems of Figure 1, the relationships among the local ontologies, the global ontology, and the common vocabulary.

Similarly to the global ontology, each local ontology has its properties mapped to the properties in the common vocabulary through *rdfs:isDefinedBy*. When the local ontology becomes related to the common vocabulary, the RDF representation of the local ontology also needs to be updated by inserting *rdfs:isDefinedBy* into any *rdfs:Class* or *rdfs:Property* being mapped. We use System E as an example (refer to Figure 4 for the previous RDF description of its local ontology). Figure 10 shows a fragment of the new RDF representation.

4.3 Discussion

The case study illustrates the following three roles played by ontologies in data integration:

Metadata representation. To uniformly represent the metadata, which

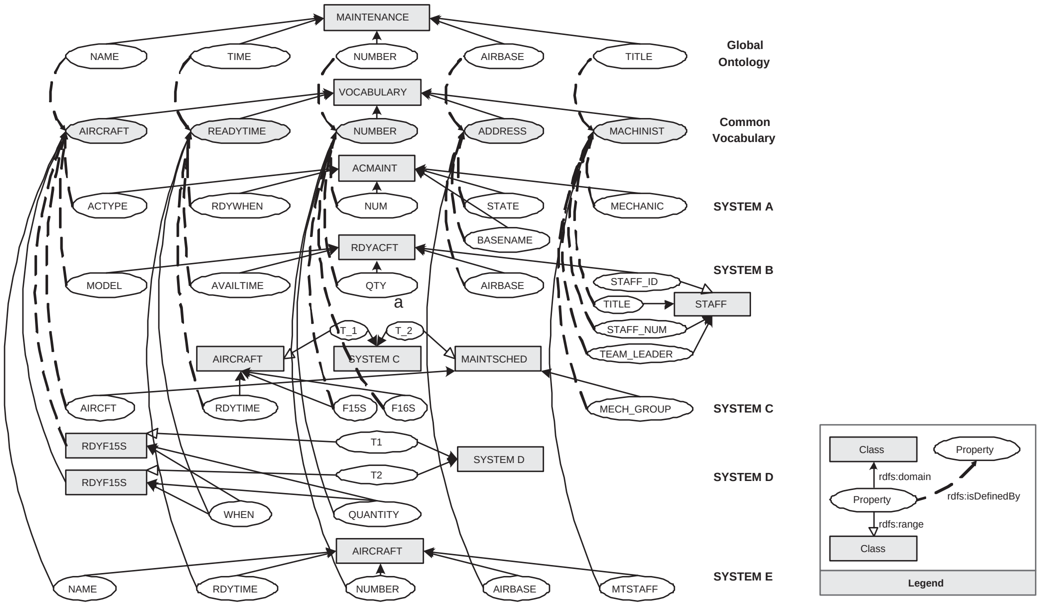
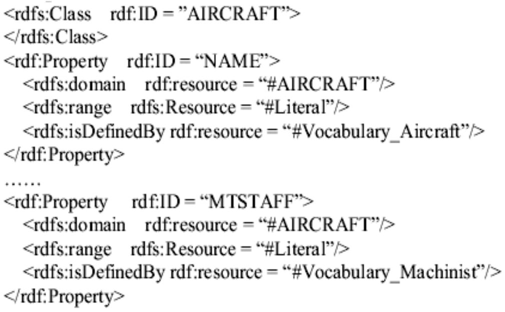


Figure 9: Mapping between the global ontology and the local ontologies.



```
<rdfs:Class    rdf:ID = ”AIRCRAFT”>
</rdfs:Class>
<rdf:Property    rdf:ID = “NAME”>
   <rdfs:domain    rdf:resource = “#AIRCRAFT”/>
   <rdfs:range    rdfs:Resource = “#Literal”/>
   <rdfs:isDefinedBy rdf:resource = “#Vocabulary_Aircraft”/>
</rdf:Property>
……
<rdf:Property     rdf:ID = “MTSTAFF”>
   <rdfs:domain    rdf:resource = “#AIRCRAFT”/>
   <rdfs:range    rdfs:Resource = “#Literal”/>
   <rdfs:isDefinedBy rdf:resource = “#Vocabulary_Machinist”/>
</rdf:Property>
```

Figure 10: RDF Schema for the mapping information of System E.

are source schemas in our case, a local ontology is constructed by means of a straightforward schema transformation process. Although a uniform syntax has been achieved, this process may seem too simplistic to encode rich and interpretable semantics. For example, names that do not correspond to words in a dictionary, such as *ACTYPE* in System A, may cause difficulties in the mapping process if not replaced by an English word or phrase, for example *Aircraft Type*. It remains an open issue how to generate an ontology that represents the metadata of a data source in a conceptual but semantically lossless way [30, 43].

Global conceptualization. In our architecture, the global ontology, which is mapped to the local ontologies using an LAV approach, provides the user with a conceptual high-level view of all the source schemas. We recall that our architecture uses an hybrid ontology approach where we associate an ontology with each networked data source and create a single ontology, the global ontology, whose properties are mapped to the properties in the common vocabulary.

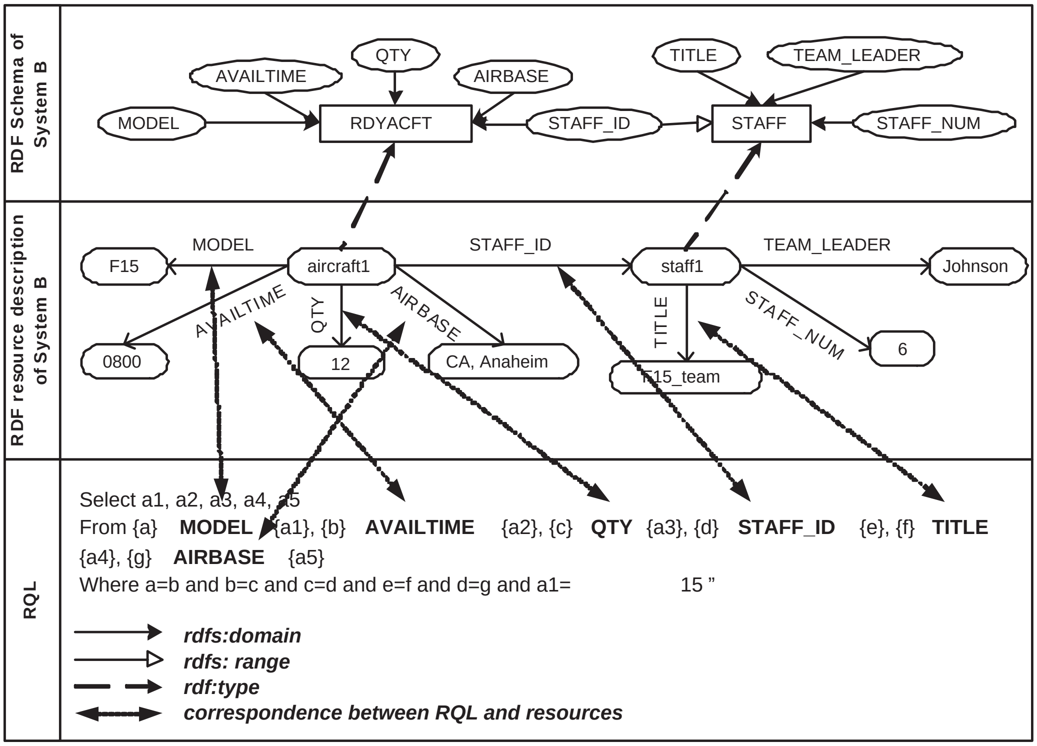
Mapping support. The support provided by the common vocabulary for the mapping process corresponds to one of the roles played by ontologies, as the vocabulary can be represented using an ontology, and play the role of a *meta-ontology*. It describes the semantic relations between the vocabularies of the global ontology and of the local ontologies, thus serving as the basis for the mappings.

5 Query Processing across Data Sources

In our layered approach, as in any data integration system, query processing plays a critical role. Query processing is performed by rewriting a query posed on one data source to all the other sources that are connected to it using established mappings.

5.1 RQL

RQL is a typed functional language and relies on a formal model for directed labeled graphs [25]. In Figure 11, we show the example of an RQL query on the local ontology of System B.



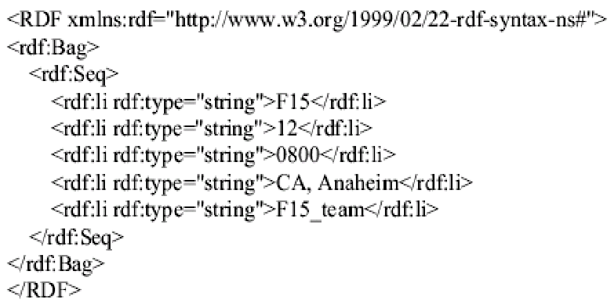
```
Select a1, a2, a3, a4, a5
From {a}  MODEL  {a1}, {b}  AVAILTIME  {a2}, {c}  QTY  {a3}, {d}  STAFF_ID  {e}, {f}  TITLE
{a4}, {g}  AIRBASE  {a5}
Where a=b and b=c and c=d and e=f and d=g and a1=                15 ”
```

Figure 11: A typical RQL query on System B.

In the *local query processing* phase the system executes the RQL query on the local ontology and on the resources that are stored in the local RSSDB; the answer to the query, shown in Figure 12, is expressed in RDF and returned to the user (a visual user interface can display the results in other formats). In the *remote query processing* phase, the query gets rewritten on the schemas of the other networked data sources.

5.2 Query Rewriting Algorithm

Our algorithm makes the following simplifying assumptions about the schemas, mappings, and queries: (1) We assume that the local ontology is either a hierarchy or can be converted into a hierarchy without losing important schematic or semantic information (see Figure 4). (2) We assume that the mappings between the global ontology and local ontologies are total mappings, that is, all the concepts (classes or properties) occurring in the query are mapped.



```
<RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<rdf:Bag>
   <rdf:Seq>
      <rdf:li rdf:type="string">F15</rdf:li>
      <rdf:li rdf:type="string">12</rdf:li>
      <rdf:li rdf:type="string">0800</rdf:li>
      <rdf:li rdf:type="string">CA, Anaheim</rdf:li>
      <rdf:li rdf:type="string">F15_team</rdf:li>
   </rdf:Seq>
</rdf:Bag>
</RDF>
```

Figure 12: Query results of executing an RQL query on local System B.

(3) We consider only one-to-one mappings, that is, a concept in one schema maps to a single concept in the other schema. (4) To keep the current discussion simple, we assume that the variables in the query only refer to "pure" classes or properties, that is, no functions such as *subClassOf*, *subPropertyOf*, *domain*, and *range* are applied to them; we consider queries with syntax {*class1*}*property*{*class2*} as shown in Figure 11. (5) We consider only schema or ontology mappings, not value mappings (which is our focus elsewhere [15]).

The rewriting algorithm uses the established mapping information between any pair of local ontologies, having the global ontology as mediator. Before the algorithm starts, we initialize the source ontology and the target ontology as schema trees using the following rules. For every pair of concepts (classes or properties), say S and O, if S *rdfs:domain* O, or S *rdfs:subClassOf* O, or S *rdfs:subPropertyOf* O, we make S a child of O; if S *rdfs:range* O or S *rdf:type* O, we incorporate S and O into a single node. In addition, we establish mappings between the source ontology and the target ontology according to their respective mappings to the global ontology. Figure 13 shows the schema trees of System B and System E and their mappings (refer also to Figure 9). In System B, for instance, the properties MODEL, AVAILTIME, and QTY are made children of the class RDYACFT. The property STAFF_ID and the class STAFF are incorporated into a single node.

In the following illustration of the query rewriting algorithm, Q is the source query and Q' is the target (rewritten) query, and we use the RDF terms *subject*, *object*, and *statement* [25]. As an example, we consider query Q on the local ontology of System B, which is shown in Figure 11. Q' is the target query on the local ontology of System E. The following steps will be

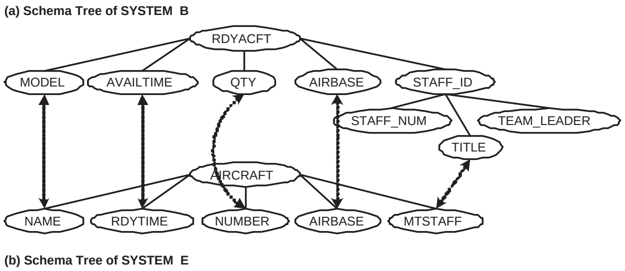


Figure 13: Local ontologies of System B and System E and their mappings.

executed:

Step 1. Get all the *statements* in the *from* clause that correspond to the objects in the select clause. In our example, we obtain (***MODEL***, ***AVAILTIME***, ***QTY***, ***AIRBASE***, ***TITLE***). Each of these *statements* corresponds to a class or property in the source ontology.

Step 2. Use the mapping information between the source ontology and the target ontology to find all the concepts (classes or properties) that correspond to the *statements* found in Step 1 and make these concepts statements in Q', as follows:

select $<object_1>$, ..., $<object_n>$
from $\{<subject_1>\}$ $<statement_1>$ $\{<object_1>\}$, ...,
$\{<subject_n>\}$ $<statement_n>$ $\{<object_n>\}$

In our particular example of Figure 13 we have:

select o'_1, o'_2, o'_3, o'_4, o'_5
from $\{s'_1\}$***NAME***$\{o'_1\}$, $\{s'_2\}$***RDYTIME***$\{o'_2\}$, $\{s'_3\}$***NUMBER***$\{o'_3\}$,
$\{s'_4\}$***AIRBASE***$\{o'_4\}$, $\{s'_5\}$***MTSTAFF***$\{o'_5\}$

Step 3. Consider each pair of nodes, E_i and E_j, corresponding to each pair of consecutive statements in Q', in the following cases:

1) If E_i and E_j are siblings (have the same parent), meaning that this pair of *statements* share a common *subject*, then we append the condition $<subject_i>=<subject_j>$ to the *where* clause of Q' using *and*.

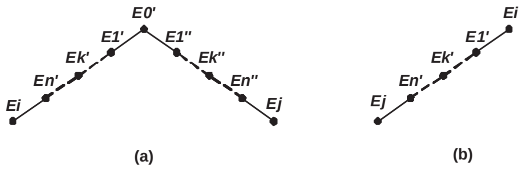


Figure 14: Relationships between E_i and E_j.

For example, in System E (see Figure 13), ***NAME*** and ***RDYTIME*** are siblings, therefore, $s'_1 = s'_2$ is to be appended to the *where* clause.

2) If E_i and E_j share a common ancestor E'_0 that is not their parent, as shown in Figure 14(a), then we append all the intermediate nodes ($statement_k$), between E'_0 and E_i and between E'_0 and E_j, to the *from* clause in the form $\{<subject_k>\}\, statement_k\{<object_k>\}$. In addition, we append new conditions to the *where* clause in the following way:

- $<object'_1> = <subject'_2>$, ..., $<object'_k> = <subject'_{k+1}>$, ..., $<object'_{n-1}> = <subject'_n>$, and $<object'_n> = <subject_i>$, which correspond to the path E'_0, E'_1, ..., E'_n, E_i.
- $<object''_1> = <subject''_2>$, ..., $<object''_k> = <subject''_{k+1}>$, ..., $<object''_{n-1}> = <subject''_n>$, and $<object''_n> = <subject_j>$, which correspond to the path E'_0, E''_1, ..., E''_n, E_j.
- $<subject'_1> = <subject''_1>$, as E'_1 and E''_1 share a common *subject*.

3) If E_i is an ancestor of E_j as shown in Figure 14(b), then we append all the intermediate nodes ($statement_k$) between E_i and E_j to the *from* clause in the form $\{<subject_k>\}statement_k\{<object_k>\}$. In addition, we append $<object_i> = <subject'_1>$, $<object'_1> = <subject'_2>$ ..., $<object'_k> = <subject'_{k+1}>$, ..., $<object'_{n-1}> = <subject'_n>$, and $<object'_n> = <subject_i>$ to the *where* clause.

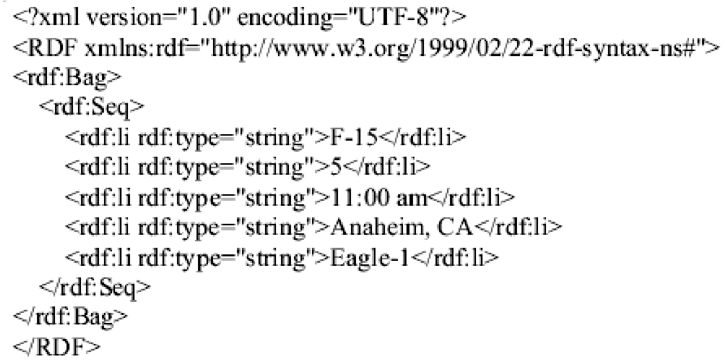
In our example, we only have the first case. Therefore, after Step 3 we get Q' as follows:

select o'_1, o'_2, o'_3, o'_4, o'_5
from $\{s'_1\}$***NAME***$\{o'_1\}$, $\{s'_2\}$***RDYTIME***$\{o'_2\}$, $\{s'_3\}$***NUMBER***$\{o'_3\}$, $\{s'_4\}$***AIRBASE***$\{o'_4\}$, $\{s'_5\}$***MTSTAFF***$\{o'_5\}$
where $s'_1 = s'_2$ and $s'_2 = s'_3$ and $s'_3 = s'_4$ and $s'_4 = s'_5$

Step 4. For each query condition $o_i =$<*string-value*> in Q, we append the condition $o'_i =$<*string-value*> to the *where* clause of Q', where o_i in Q is mapped to o'_i in Q'. In our example, we obtain the following Q':

$$\begin{array}{ll}
\textbf{select} & o'_1,\ o'_2,\ o'_3,\ o'_4,\ o'_5 \\
\textbf{from} & \{s'_1\}\boldsymbol{NAME}\{o_1\},\ \{s'_2\}\boldsymbol{RDYTIME}\{o_2\},\ \{s'_3\}\boldsymbol{NUMBER}\{o_3\}, \\
 & \{s'_4\}\boldsymbol{AIRBASE}\{o'_4\},\ \{s'_5\}\boldsymbol{MTSTAFF}\{o'_5\} \\
\textbf{where} & s'_1 = s'_2 \text{ and } s'_2 = s'_3 \text{ and } s'_3 = s'_4 \text{ and } s'_4 = s'_5 \text{ and } o'_1 = \text{“F15”}
\end{array}$$

Step 5. Finally, Q' is executed on RSSDB in System E. For our running example, we obtain the answer that is shown in Figure 15, where *rdf:Seq* is a tuple. This answer, which is returned by the *remote query processing* phase associated with System E, will be assembled by unioning all the tuples with the *local query processing* results that were shown in Figure 12.



```
<?xml version="1.0" encoding="UTF-8"?>
<RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<rdf:Bag>
  <rdf:Seq>
    <rdf:li rdf:type="string">F-15</rdf:li>
    <rdf:li rdf:type="string">5</rdf:li>
    <rdf:li rdf:type="string">11:00 am</rdf:li>
    <rdf:li rdf:type="string">Anaheim, CA</rdf:li>
    <rdf:li rdf:type="string">Eagle-1</rdf:li>
  </rdf:Seq>
</rdf:Bag>
</RDF>
```

Figure 15: Query results of executing Q' on System E.

5.3 Discussion

In our layered approach, the semantic layer enables the use of the above described algorithm for query rewriting in two ways—central and peer-to-peer—which correspond to two of the ontology roles described in Section 2:

Support for high-level queries. This role dependents on the *Global Conceptualization* role, which was discussed in Section 4. Given that the global ontology acts as a single query interface to the data sources, the user is then able to formulate queries by referring to the global ontology, which serves as a "super-peer". Both central and peer-to-peer queries are supported, based on the mappings between the global ontology and the local ontologies. Queries on the global ontology are automatically rewritten to a local query over each local ontology. A query on a peer is automatically rewritten to a query on another peer.

Declarative mediation. As shown in the above example illustrating the query rewriting algorithm, the rewriting of Q over System B to Q' over System E makes use of the global ontology as a mediator. In particular, the mapping information between System B and System E is derived by composing the mappings from System B to the global ontology and those from the global ontology to System E. It is possible that the composition of two mappings involves some semantic reasoning that may be supported by the global ontology. For example, suppose that System B has a concept *Transportation* mapped to Concept *Vehicle* in the global ontology, and that *Aircraft* (a child of *Vehicle*) is mapped to a concept *Plane* in System E. Then, the concept *Transportation* can be made a parent of *Plane*.

6 Related Work

The subject of interoperability and the related subject of data integration have been long-standing in database research. In the nineties, several system architectures and systems have been proposed, including TSIMMIS [13] and InfoHarness [39].

In the field of data integration there are a number of noteworthy proposals. For example, the MOMIS (Mediator Environment for Multiple Information Sources) system consists of a set of tools for the integration of information in heterogeneous sources ranging from traditional databases to XML databases [6, 7]. The integration process relies heavily on a common thesaurus derived from the WordNet lexical database. The inter-schema knowledge is expressed using the thesaurus in the form of relationships such as synonymy and hyponymy. One tool is the SI-Designer (Source Integrator Designer) [6], a designer support tool for E-commerce applications, which uses a semi-automatic (requiring some user intervention) approach for the integration of heterogeneous data sources. Another tool is the MIKS (Mediator agent for Integration of Knowledge Sources) system [7], an agent based middleware system that integrates data belonging to heterogeneous sources into a global virtual view and offers support for the execution of queries over the global virtual schema [11].

The Clio project creates mappings between two data representations semi-automatically for the purpose of managing and facilitating heterogeneous data transformation and integration [36]. Given the two schemas and the set of correspondences between them, Clio can generate queries (e.g., SQL, XSLT, XQuery) that drive the translation of data conforming to the source schema to data conforming to the target schema. However, with the exception of mapping queries, Clio does not provide a mechanism for users to associate semantic information with the data—the only semantics are those "embedded" in the queries.

There are a number of approaches addressing the problem of data integration among XML sources. In what follows we look at some of the main features found in existing approaches and describe those approaches.

6.1 Semantic Integration

High-level mediator Amann *et al.* propose an ontology-based approach to the integration of heterogeneous XML Web resources in the C-Web project [2, 3]. The proposed approach is very similar to our approach except for the following differences. For example, they use the local-as-view (LaV) approach with a hypothetical global ontology that may be incomplete.

Direct translation Klein proposes a procedure to transform XML data directly into RDF data by annotating the XML documents using external RDFS specifications [26].

Encoding semantics The Yin/Yang Web [35] proposed by Patel-Schneider and Siméon address the problem of incorporating the XML and RDF paradigms. They develop an integrated model for XML and RDF by integrating the semantics and inferencing rules of RDF into XML, so that XML querying can benefit from their RDF reasoner. But the Yin/Yang Web does not solve the essential problem of query answering across heterogeneous sources, that is, sources with different syntax or data models. Lakshmanan and Sadri also propose an infrastructure for interoperating over XML data sources by semantically marking up the information contents of data sources using application-specific common vocabularies [27]. However, the proposed approach relies on the availability of an application-specific standard ontology that serves as the global schema. This global schema contains information necessary for interoperability, such as key and cardinality information for predicates.

6.2 Query Languages

CXQuery [14] is an XML query language proposed by Chen and Revesz, which borrows features from both SQL and other XML query languages. It overcomes the limitations of the XQuery language by allowing the user to define views, specify explicitly the schema of the answers, and query multiple XML documents. However, CXQuery does not solve the issue of structural heterogeneities among XML sources. The user has to be familiar with the document structure of each XML source to formulate queries. Heuser *et al.* present the CXPath language based on XPath for querying XML sources at the conceptual level [12]. CXPath is used to write queries over a conceptual schema that abstracts the semantic content of several XML sources. However, they do not consider query rewriting from the XML sources to the global schema.

6.3 Query Rewriting

Query rewriting or query translation is the key issue for both mediator-based integration systems and peer-to-peer systems. As an example of the first case, the Clio approach [36] addresses the issue of schema mapping and data transformation between nested schemas and/or relational databases. It focuses on how to take advantage of schema semantics to generate the consistent translations from source to target by considering the constraints and structure of the target schema. The approach uses queries to express the mapping so as to transform the data into the target schema. The Piazza system [24] is a peer-to-peer system for interoperating between XML and RDF. The system achieves its interoperability in a low-level (syntactic) way using the interoperability of XML and the XML serialization of RDF.

7 Conclusions and Future Work

The work on data integration that we presented in this paper fits in the overall perspective of metamodels and mappings that address the problem of interoperability in the presence of multiple standards [44]. In particular, we presented an approach to the syntactic, schematic, and semantic integration of data sources using a layered model. Our layered model consists of four layers, of which the main component is the semantic layer. This layer uses a hybrid ontology-based approach to the integration problem that enables both central and peer-to-peer query processing. Our contribution involves the representation of metadata, global conceptualization, declarative mediation, mapping support, and query processing.

Future work will focus on the following topics:

- We will lift some of the restrictions we imposed on the query rewriting algorithm. An interesting extension, which will further use the conceptual modeling capabilities of RDF Schema, will consider the mapping of concepts related by subclassing.

- We will further explore the creation of joint local schemas to make it more general.

- We will consider the case where the answers to the queries may be expressed in the native schema of the legacy databases. For example, by using RDF, the nesting structure associated with XML is lost in the mapping from XML data to RDF data. We will look at which properties can be encoded using RDF Schema, and how they can be encoded. For example, RDF can be extended to encode the XML nesting structure using the property `rdfx:contained` [48].

- We will consider the problem of building vocabularies and of automatically matching ontologies that represent different vocabularies so as to align similar terms [40]. There have been significant advances in automatic ontology matching and in the evaluation of algorithms that implement those matchings [20], including of algorithms that consider the structure of the ontology graph (e.g., [41, 42]).

References

[1] S. Alexaki, V. Christophides, G. Karvounarakis, D. Plexousakis, and K. Tolle. The ICS-FORTH RDFSuite: Managing Voluminous RDF Description Bases. In *International Workshop on the Semantic Web (SemWeb)*, Hongkong, China, 2001.

[2] B. Amann, C. Beeri, I. Fundulaki, and M. Scholl. Ontology-Based Integration of XML Web Resources. In *International Semantic Web Conference (ISWC)*, pages 117–131, 2002.

[3] B. Amann, I. Fundulaki, M. Scholl, C. Beeri, and A.-M. Vercoustre. Mapping XML Fragments to Community Web Ontologies. In *International Workshop on the Web and Databases (WebDB)*, pages 97–102, 2001.

[4] M. Arenas, V. Kantere, A. Kementsietsidis, I. Kiringa, R. J. Miller, and J. Mylopoulos. The Hyperion Project: From Data Integration to Data Coordination. *SIGMOD Record*, 32(3):53–38, 2003.

[5] Y. Arens, C. A. Knoblock, and C. Hsu. Query Processing in the SIMS Information Mediator. In A. Tate, editor, *Advanced Planning Technology*. The AAAI Press, Menlo Park, CA, 1996.

[6] I. Benetti, D. Beneventano, S. Bergamaschi, F. Guerra, and M. Vincini. SI-Designer: an Integration Framework for E-commerce. In *IJCAI Workshop on E-Business and the Intelligent Web*, Seattle, WA, 2001.

[7] D. Beneventano, S. Bergamaschi, G. Gelati, F. Guerra, and M. Vincini. MIKS: An Agent Framework Supporting Information Access and Integration. In S. Bergamaschi, M. Klusch, P. Edwards, and P. Petta, editors, *Intelligent Information Agents Research and Development in Europe: An AgentLink Perspective*, volume 2586 of *Lecture Notes in Computer Science*, pages 22–49. Springer, 2003.

[8] S. Bergamaschi, F. Guerra, and M. Vincini. A Peer-to-Peer Information System for the Semantic Web. In *International Workshop on Agents and Peer-to-Peer Computing (AP2PC)*, pages 113–122, 2003.

[9] P. A. Bernstein, F. Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L. Serafini, and I. Zaihrayeu. Data Management for Peer-to-Peer Computing: A Vision. In *International Workshop on the Web and Databases (WebDB)*, pages 89–94, 2002.

[10] Y. A. Bishr. Overcoming the Semantic and Other Barriers to GIS Interoperability. *International Journal of Geographical Information Science*, 12(4):229–314, 1998.

[11] G. Cabri, F. Guerra, M. Vincini, S. Bergamaschi, L. Leonardi, and F. Zambonelli. MOMIS: Exploiting Agents to Support Information Integration. *International Journal of Cooperative Information Systems*, 11(3):293–314, 2002.

[12] S. D. Camillo, C. A. Heuser, and R. dos Santos Mello. Querying Heterogeneous XML Sources through a Conceptual Schema. In *International Conference on Conceptual Modeling (ER)*, pages 186–199, 2003.

[13] S. S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. D. Ullman, and J. Widom. The TSIMMIS Project: Integration of Heterogeneous Information Sources. In *Meeting of the Information Processing Society of Japan (IPSJ)*, pages 7–18, Tokyo, Japan, 1994.

[14] Y. Chen and P. Revesz. CXQuery: A Novel XML Query Language. In *International Conference on Advances in Infrastructure for Electronic Business, Science, and Medicine on the Internet (SSGRR 2002w)*, 2002.

[15] I. F. Cruz, A. Rajendran, W. Sunna, and N. Wiegand. Handling Semantic Heterogeneities Using Declarative Agreements. In *International Symposium on Advances in Geographic Information Systems (ACM GIS)*, pages 168–174, McLean, VA, 2002.

[16] I. F. Cruz and H. Xiao. Using a Layered Approach for Interoperability on the Semantic Web. In *International Conference on Web Information Systems Engineering (WISE)*, pages 221–232, December 2003.

[17] I. F. Cruz and H. Xiao. The Role of Ontologies in Data Integration. *Journal of Engineering Intelligent Systems*, 13(4):245–252, December 2005.

[18] I. F. Cruz, H. Xiao, and F. Hsu. An Ontology-based Framework for Semantic Interoperability between XML Sources. In *International Database Engineering and Applications Symposium (IDEAS)*, pages 217–226, 2004.

[19] I. F. Cruz, H. Xiao, and F. Hsu. Peer-to-Peer Semantic Integration of XML and RDF Data Sources. In *Workshop on Agents and Peer-to-Peer Computing (AP2PC 2004)*, volume 3601 of *Lecture Notes in Computer Science*, pages 108–119. Springer, 2005.

[20] J. Euzenat, A. Isaac, C. Meilicke, P. Shvaiko, H. Stuckenschmidt, O. Šváb, V. Svátek, W. R. van Hage, and M. Yatskevich. First Results of the Ontology Evaluation Initiative 2007. In *Second ISWC International Workshop on Ontology Matching*. CEUR-WS, 2007.

[21] T. R. Gruber. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

[22] T. R. Gruber and G. R. Olsen. An Ontology for Engineering Mathematics. In *International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 258–269, 1994.

[23] N. Guarino. Formal Ontology and Information Systems. In *International Conference on Formal Ontologies in Information Systems (FOIS 1998)*, pages 3–15, 1998.

[24] A. Y. Halevy, Z. G. Ives, P. Mork, and I. Tatarinov. Piazza: Data Management Infrastructure for Semantic Web Applications. In *International World Wide Web Conference (WWW)*, pages 556–567, 2003.

[25] G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, and M. Scholl. RQL: A Declarative Query Language for RDF. In *International World Wide Web Conference (WWW)*, pages 592–603, 2002.

[26] M. C. A. Klein. Interpreting XML Documents via an RDF Schema Ontology. In *International Conference on Database and Expert Systems Applications (DEXA)*, pages 889–894, 2002.

[27] L. V. Lakshmanan and F. Sadri. Interoperability on XML Data. In *International Semantic Web Conference (ISWC)*, pages 146–163, 2003.

[28] M. Lenzerini. Data Integration: A Theoretical Perspective. In *ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pages 233–246, Madison, WI, 2002.

[29] B. Ludächer, A. Gupta, and M. E. Martone. Model-Based Mediation with Domain Maps. In *IEEE International Conference on Data Engineering (ICDE)*, pages 81–90, 2001.

[30] A. Maedche and R. Volz. The Text-To-Onto Ontology Extraction and Maintenance System. In *Workshop on Integrating Data Mining and Knowledge Management co-located with the 1st International Conference on Data Mining*, 2001.

[31] S. Melnik and S. Decker. A Layered Approach to Information Modeling and Interoperability on the Web. In *ECDL Workshop on the Semantic Web*, Lisbon, Portugal, 2000.

[32] E. Mena, V. Kashyap, A. P. Sheth, and A. Illarramendi. OBSERVER: An Approach for Query Processing in Global Information Systems Based on Interoperation Across Pre-existing Ontologies. In *IFCIS International Conference on Cooperative Information Systems (CoopIS)*, pages 14–25, 1996.

[33] E. Morris, L. Levine, C. Meyers, P. Place, and D. Plakosh. System of Systems Interoperability (SOSI): Final Report. Technical Report CMU/SEI-2004-TR-004, ESC-TR-2004-004, Carnegie Mellon Software Engineering Institute, Pittsburgh, PA, April 2004.

[34] W. S. Ng, B. C. Ooi, K. L. Tan, and A. Zhou. PeerDB: A P2P-based System for Distributed Data Sharing. In *IEEE International Conference on Data Engineering (ICDE)*, pages 633–644, 2003.

[35] P. F. Patel-Schneider and J. Siméon. The Yin/Yang Web: XML Syntax and RDF Semantics. In *International World Wide Web Conference (WWW)*, pages 443–453, 2002.

[36] L. Popa, Y. Velegrakis, R. J. Miller, M. A. Hernández, and R. Fagin. Translating Web Data. In *International Conference on Very Large Databases (VLDB)*, pages 598–609, 2002.

[37] E. Rahm and P. A. Bernstein. A Survey of Approaches to Automatic Schema Matching. *VLDB Journal*, 10(4):334–350, 2001.

[38] S. A. Renner, A. S. Rosenthal, and J. G. Scarano. Data Interoperability: Standardization or Mediation. In *IEEE Metadata Conference*, 1996.

[39] L. A. Shklar, A. P. Sheth, V. Kashyap, and K. Shah. InfoHarness: Use of Automatically Generated Metadata for Search and Retrieval of Heterogeneous Information. In *International Conference on Advanced Information Systems Engineering (CAiSE)*, pages 217–230, 1995.

[40] P. Shvaiko and J. Euzenat. A Survey of Schema-Based Matching Approaches. In *Journal on Data Semantics IV*, volume 3730 of *Lecture Notes in Computer Science*, pages 146–171. Springer, 2005.

[41] W. Sunna and I. F. Cruz. Structure-Based Methods to Enhance Geospatial Ontology Alignment. In *International Conference on GeoSpatial Semantics (GeoS)*, volume 4853 of *Lecture Notes in Computer Science*, pages 82–97. Springer, 2007.

[42] W. Sunna and I. F. Cruz. Using the AgreementMaker to Align Ontologies for the OAEI Campaign 2007. In *Second ISWC International Workshop on Ontology Matching*. CEUR-WS, 2007.

[43] Y. A. Tijerino, D. W. Embley, D. W. Lonsdale, and G. Nagy. Ontology Generation from Tables. In *International Conference on Web Information Systems Engineering (WISE)*, pages 242–252, Rome, Italy, 2003.

[44] A. Tolk. Metamodels and Mappings–Ending the Interoperability War. In *Fall Simulation Interoperability Workshop*, pages 748–761. IEEE CS Press, 2004.

[45] A. Tolk, C. Turnitsa, and S. Diallo. Implied Ontological Representation Within the Levels of Conceptual Interoperability Model. *Intelligent Decision Technologies*, 2(1):3–20, 2008.

[46] J. D. Ullman. Information Integration Using Logical Views. In *International Conference on Database Theory (ICDT)*, pages 19–40, 1997.

[47] 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 *IJCAI Workshop on Ontologies and Information Sharing*, 2001.

[48] H. Xiao and I. F. Cruz. Integrating and Exchanging XML Data Using Ontologies. In *Journal on Data Semantics VI*, volume 4090 of *Lecture Notes in Computer Science*, pages 67–89. Springer, 2006.